Volume 17 Number 4

ACTA CYBERNETICA

Szeged, 2006

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. Fifty reprints are supplied for each article published.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

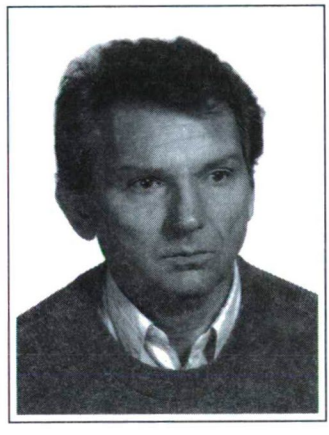
Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above informations along with the contents of past issues are available at the Acta Cybernetica homepage http://www.inf.u-szeged.hu/actacybernetica/ .

Editorial

In Memoriam Balázs Imreh (1945–2006)

On 8 August 2006, *Acta Cybernetica* lost one of its Editors. **Balázs Imreh** passed away that day at the age of 61.



Dr. Imreh was born in Szekszárd, Hungary. He received the MS degree from the *University of Szeged* (SZTE), formerly *József Attila University*, Szeged, Hungary in 1968 and the Candidate of Mathematical Sciences degree from the *Hungarian Academy of Sciences* in 1983. He joined the faculty of SZTE in 1969 and served as Head of the Institute of Informatics from 1991 to 1998. He also held positions as Head of the Department of Foundations of Computer Science and Head of the Department of Applied Informatics. He obtained the Habilitation degree in 2002.

During his service at the University, he was actively involved in both research and education. He wrote 4 textbooks on operations research and published more than 70 scientific papers in the fields of automata theory and operations research.

He was a member of the *Institute for Operations Research and the Management Sciences* (INFORMS), *János Bolyai Mathematical Society*, *John von Neumann Computer Society*, *Hungarian Operational Research Society*, and the *Committee of Computer Science of the Hungarian Academy of Sciences.*

Dr. Imreh served as Editor of *Acta Cybernetica* since 1996. His death is a great loss to all of us.

Zoltan Kato
Managing Editor

János Csirik
Editor-in-Chief

János Demetrovics
MTA SZTAKI
Budapest, Hungary
demetrovics@sztaki.hu

Bálint Dömölki
IQSOFT
Budapest, Hungary
domolki@iqsoft.hu

Zoltán Fülöp
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
fulop@inf.u-szeged.hu

Ferenc Gécseg
Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
gecseg@inf.u-szeged.hu

Jozef Gruska
Institute of Informatics/Mathematics
Slovak Academy of Science
Bratislava, Slovakia
gruska@savba.sk

Balázs Imreh
Department of Applied Informatics
University of Szeged
Szeged, Hungary
imreh@inf.u-szeged.hu

Helmut Jürgensen
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
helmut@csd.uwo.ca

Alice Kelemenová
Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun
Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa
Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa
Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga
Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
vogler@inf.tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Preface

The AFL (Automata and Formal Languages) conference series was initiated by Prof. István Peák (1936-1989). He organized the AFL conferences in 1980, 1982, 1984, 1986 and 1988, and started the organization of AFL'90. These conferences were all held in the hills around Salgótarján. In 1986 and 1988, the title of the conference was Automata, Languages and Programming Systems. Since the untimely death of Prof. István Peák in 1989, the AFL conferences have been organized in every third year. In 1993 and 1996, two more "Salgótarján conferences" took place. The last two conferences of the series were held in Vasszécsény (1999) and Debrecen (2002).

The 11th International Conference on Automata and Formal Languages, AFL 2005 took place in Dobogókő located at a distance of 30 kilometers from the center of Budapest in the Pilis mountains.

Topics of interest included grammars, acceptors and transducers for strings, trees, graphs, arrays, etc., algebraic theories for automata and languages, combinatorial properties of words and languages, formal power series, decision problems, efficient algorithms for automata and languages, relations of automata and language theory to computational complexity theory logic, picture description and analysis using automata theoretic tools, DNA computing, quantum computing, cryptography, automata and languages in relation to concurrency.

This issue of Acta Cybernetica contains the full version of 9 papers presented at the conference. All of them have been refereed according to the usual standards of the journal. I would like to thank all authors of the papers of this journal issue and all those who have contributed to the success of AFL 2005, including the members of the conference committees, all speakers and participants, and everybody who submitted a paper, or took part in the evaluation of the submissions.

Kyoto, October 2006

Zoltán Ésik

Acta Cybernetica **17** (2006) 665–684.

Finite State Evaluation of Logical Formulas : Jevons' Approach (1870) and Contemporary Description

Paul Amblard*

Abstract

In this paper, we describe a formal language for a class of logical expressions. We then present a Finite State Machine for recognition and evaluation of this language. The main interest of the language is its historical characteristic. This language invented by the British scholar W. Stanley JEVONS in 1865 is probably the earliest language in which expressions were evaluated by a Finite State Machine. The two outstanding contributions were the use of machinery to evaluate formulas and the evaluation of formulas with variables by several parallel evaluations with constants. The contribution of this paper is to present this ancient evaluation process in a contemporary framework, i.e. formal languages and finite state automata. The design of an evaluator is given in great detail.

Introduction and Related Works

The history of calculating machines is well known. Pascal and Babbage built machines that are considered as the mechanical ancestors of today's computers. But computers do not only compute numbers, they can also perform symbolic evaluations. At a certain level of abstraction, we may consider mechanisms based on logical choices `if ... then ...`, or `if ... then ... else ...` as the necessary complements of strictly arithmetic operations.

The first mechanical machine making these choices to " perform the logical inference " was designed by William Stanley JEVONS in 1865 and published in 1870 ([14]). He had been a student of De Morgan. He was at that time becoming a professor of Logic (and of Political Economy) at the Owens College of Manchester. Figure 1 shows Jevons' activity time w.r.t other well-known British logicians.

His work has often been presented in the same terms as in the original paper: logical evaluation [1, 5, 6, 16]. Burris' paper [5] gives interesting details about Jevons' logic and about his machine. But none of these papers establishes relations

*TIMA-CMP Lab. University of Grenoble, 46 Av. Felix Viallet, 38031 Grenoble CEDEX, France, E-mail: `Paul.Amblard@imag.fr`

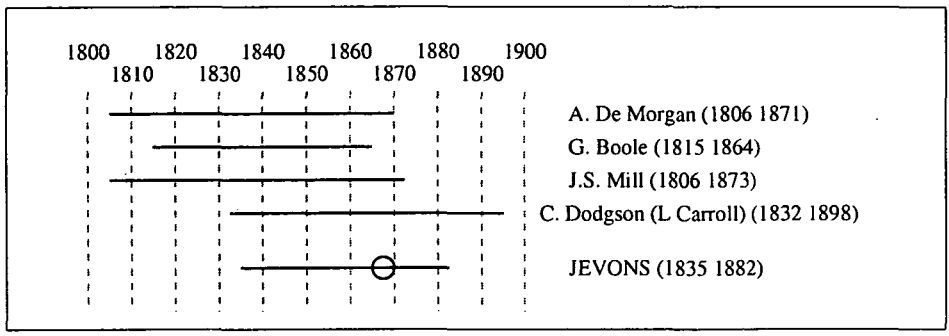


Figure 1: British logicians period. Jevons' time devoted to his machine is circled.

between Jevons' work and automata formalization. The only relations between Jevons' work and automata appear in Shepherdson's paper [19]. Unfortunately, after saying *And the key of the success of the whole endeavour was the discovery of a 'context free' algorithm which allowed the input proposition to be processed from left to right one symbol at a time*, the author did not explore in detail this 'context free algorithm'. He could certainly have discovered that it is in fact simpler than context free: no stack is used in this mechanical machine. However the paper is of great interest to us: Shepherdson describes the relations between the machine and the theory of Jevons, and he gives the drawings of the machine. A hardware implementation of this automaton by a V.L.S.I circuit had been studied in [2]. The present paper extends the presentation of [3].

In this paper we propose a description of Jevons' work in the framework of formal languages and automata. We shall see that Jevons can be considered as the inventor of Finite State transducers and evaluators. He also invented an ancestor in parallelism: input data are distributed (with modifications) to different "processors" running the same evaluation process, some kind of Single Instruction Multiple Data.

The paper is organized as follows: the first part describes a set of logical expressions. They constitute Jevons' formal language. The process of recognition is also described. The evaluation of this language is described in the second part. It is based on the Finite State paradigm. This part presents our technique and the technique used by Jevons himself to evaluate the formulas. It will then be possible in the third part to show that Jevons' evaluation, completed by syntactic analysis of formulas, meets our expectations. We shall give some details about the method used to obtain evaluation, based on composition of automata.

° In the paper some quotations from Jevons' presentation [14] will appear in this form. The number is the reference of the paragraph in the original text. °

1 Jevons' Language

1.1 Pseudo-natural Language

Jevons dealt with logical formulas organized as

iron is metal AND *metal is not_wood*

The goal being obviously to deduce that

iron is not_wood

If we maintain Jevons' terminology, in the sentence *iron is metal* *iron* is a "subject", *is* is the "copula" and *metal* is an "attribute". The copula is obviously an implication.

He also allowed disjunctions of conjunctions both in the subject part and in the attribute part. This conjunction is denoted by *and*. Disjunction is denoted by *or*.

Jevons' language contained conjunctions of sentences. This conjunction is denoted by AND.

He could then write formulas like

iron and heavy is metal AND *heavy or metal is not_wood* AND *wood is not_metal or not_iron*

1.2 Formal Language Implemented by Jevons

In his formal language, Jevons used four variables A, B, C and D, and their respective complements a, b, c and d instead of natural language names (iron, metal,..) so the previous sentences become

A is B AND *B is d*

Obviously, the conclusion remains :

A is d

Jevons also made explicit the distinction between a variable appearing in an attribute part or in a subject part. Conjunctions of variables were denoted by simple concatenation, where

$$A_s D_s b_s$$

simply denotes "A and D and not B" when this appears in a subject.

Similarly $A_a D_a b_a$ appears in an attribute.

The disjunction was the inclusive OR. Let us remember that Boole used at this time the exclusive OR and that he and Jevons exchanged arguments about this choice. The generalization of inclusive OR is also a contribution from Jevons. The disjunction had also the distinction between subject and attribute giving $+_s$ and $+_a$.

The conjunction between sentences was an AND and was written as a "Full-Stop". This AND is syntactically different from the *and* between variables.

The language of correct expressions is described by Jevons but he did not give any formal description of it. Formal grammars were only invented 80 years later. Similarly automata were not already known with the contemporary meaning. The word already existed in Homer's Iliad (ch. 5, v. 749 and ch. 2, v. 408) but the reality is not the same. It refers to things (The gates of heavens) or people (Menelas) moving by themselves.

The photo of the keyboard on Jevons' machine is available from the website of the Museum of the History of Science in Oxford. Due to its aspect, many descriptions present it as the "Logical Piano" (`www.mhs.ox.ac.uk/images/index.htm` then search for Jevons).

> 36
> The key board of the instrument is shown in fig. [..], where are seen two sets of term or letter keys, marked A, *a*, B, *b*, C, *c*, D, *d*, separated by a key marked COPULA–IS. The letter keys on the left belong to the subject of a proposition, those on the right to the predicate, and on either side just beyond the letter keys is a *Conjunction* key, appropriated to the disjunctive conjunction *or*, according as it occurs in the subject or predicate. The last key on the right hand is marked FULL STOP, and is to be pressed at the end of each proposition, where the full stop is properly placed. On the extreme left, lastly, is a key marked FINIS, which is used to terminate one problem and prepare the machine for a new one.

Example and transcription of Jevons' language in this paper

"A_sD_s OR$_s$ a_sC_s is c_aB_a Full-Stop B_s is D_a OR$_a$ A_ac_a Full-Stop" represent the formula nowadays written in standard logic as $(A \wedge D \vee a \wedge C \Rightarrow c \wedge B) \wedge (B \Rightarrow D \vee A \wedge c)$ (We could also write $\neg$ A instead of a). In this paper we shall use the following form : $A_sD_s +_s a_sC_s \Rightarrow c_aB_a \bullet B_s \Rightarrow D_a +_a A_ac_a \bullet$.

We take the conventions

A_s, a_s, B_s, b_s, C_s, c_s, D_s, d_s are the variables in a subject part,
A_a, a_a, B_a, b_a, C_a, c_a, D_a, d_a are the variables in an attribute part,
$+_s$ and $+_a$ are the disjunctions, in subject and attribute part,
$\Rightarrow$ is the IMPLIES named "is" by Jevons,
$\bullet$ is the AND between sentences named "Full-stop" by Jevons.

Jevons added a key *Finis* which was simply a "reset" key. We do not use it because it simply forces the machine into the initial state.

Jevons' language can then be described by the grammar of figure 2. The vocabulary is V_T.

$$V_T = \{A_s, a_s, B_s, b_s, C_s, c_s, D_s, d_s, +_s, \Rightarrow, A_a, a_a, B_a, b_a, C_a, c_a, D_a, d_a, +_a, \bullet\}$$

The grammar is described by noting that a problem is a sentence or a sentence followed by a problem, a sentence is a subject followed by an attribute, and so on. In the grammar, we use Var_s (resp. Var_a) for any variable in a subject (resp. attribute).

Problem	$\rightarrow$	Sentence	/ Sentence Problem
Sentence	$\rightarrow$		Subject $\Rightarrow$ Attribute $\bullet$
Subject	$\rightarrow$	Product_s	/ Product_s $+_s$ Subject
Attribute	$\rightarrow$	Product_a	/ Product_a $+_a$ Attribute
Product_s	$\rightarrow$	Var_s	/ Var_s Product_s
Product_a	$\rightarrow$	Var_a	/ Var_a Product_a

Figure 2: Grammar of Jevons' language

Another form of grammar is as follows :

Intermediate vocabulary is $V_N = \{ \text{J, K, L, M, N} \}$. The axiom is J. The rules are :

$\text{J} \rightarrow \text{Var}_s \text{ K}$
$\text{K} \rightarrow \text{Var}_s \text{ K} \ / \ +_s \text{ J} \ / \Rightarrow \text{L}$
$\text{L} \rightarrow \text{Var}_a \text{ M}$
$\text{M} \rightarrow \text{Var}_a \text{ M} \ / \ +_a \text{ L} \ / \bullet \text{ N}$
$\text{N} \rightarrow \text{Var}_s \text{ K} \ / \ \epsilon$

Let us note, as a comment, that a variable can be repeated any number of times in a product without changing the meaning. We could then think about an asynchronous automaton as in ([13]) but this property is not true for other symbols. We should have to admit "strange" expressions such as $1_s \Rightarrow \Rightarrow 0_a \bullet$.

Another comment is about redundancy between the indication subject-attribute and the correct alternation of the separators $\bullet$ and $\Rightarrow$.

The finite state recognizer of the language is represented by Automaton SA in figure 3.

2 Evaluation of Formulas

The main contribution of Jevons concerns evaluation of the aforesaid logical formulas. His method was obviously not explicitely based on Finite State Transducers, but, as we shall see, all the ideas were already present. His method was based on two levels: the first one consists of the evaluation of a formula containing variables by implementing several evaluations of formulas containing only constants (true 1, false 0). The second level is indeed a Finite State Evaluation process. The combination of the two levels could be described as a Single Instruction Multiple Data machine.

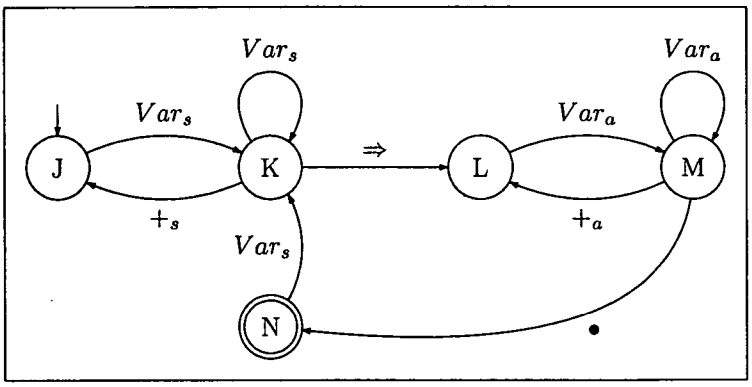


Figure 3: SA : Syntax Analyser. Recognition of Jevons' formulas. J is initial state, N is final state. The "dead" state is hidden. All transitions not described lead to this hidden state.

2.1 From Variables to Constants

Jevons considered evaluation of formulas with 4 logical variables A, B, C, D. To perform this evaluation, he considered 16 situations, corresponding to the 16 lines of a (nowadays) classic truth table. Truth tables were already known in 1870, in some forms, mainly presented by Leibniz. Jevons used truth tables under the name of *logical abecedarium.*

> 20
> Problems involving four distinct terms would similarly require a series of sixteen conceivable combinations, and if five or six terms enter, there will be thirty-two or sixty-four of such combinations. These series of combinations appear to hold a position in logical science at least as important as that of the multiplication table in arithmetic or the coefficients of the binomial theorem in the higher parts of mathematics. I propose to call any such complete series of combinations a *Logical Abecedarium...*

To evaluate a formula with N variables, Jevons simply evaluates 2^N formulas with constants. Each individual evaluator is labelled by a name representing a line in the truth table. Line ABCD represents the line where both A, B, C, D are true, line AbCd represents the line where A and C are true and B and D are false, and so on.

Jevons designed his machine with such a mechanism that evaluation of

$$A_s D_s \ +_s \ a_s C_s \ \Rightarrow \ c_a B_a \ \bullet \ B_s \ \Rightarrow \ D_a \ +_a \ A_a c_a \ \bullet$$

is implemented by evaluating

$1_s1_s +_s 0_s1_s \Rightarrow 0_a1_a \bullet 1_s \Rightarrow 1_a +_a 1_a0_a \bullet$ on line ABCD
$1_s0_s +_s 0_s1_s \Rightarrow 0_a1_a \bullet 1_s \Rightarrow 0_a +_a 1_a0_a \bullet$ on line ABCd
$1_s1_s +_s 0_s0_s \Rightarrow 1_a1_a \bullet 1_s \Rightarrow 1_a +_a 1_a1_a \bullet$ on line ABcD

and so on.
The fact that these 16 evaluations could be done in parallel was more obvious at that time. Jevons was not disturbed by the sequential activities scheme introduced by modern computers under the Von Neumann paradigm.

> The Logical Abacus was devised [..] and was constructed by placing the combinations of the *abecedarium* upon **separate moveable** slips of wood, which can then be easily classified, selected and arranged according to the conditions of the problem.

The mechanism moving the different slips of wood was slightly different for each line of the Abecedarium. So we see that the standard problem SAT, known to be NP-complete, was first solved by a system responding in constant time (in fact in time 0, the answer is given immediately at the end of the formula) but exponential in number of processors.

In Jevons' machine, however, the energy for activating the 16 evaluators was simply given by the user pressing the key of a keyboard. This energy limited the parallelism degree of the system.

2.2 Evaluation of Formulas with Constants

From a timed sequence of inputs (actions on mechanic keys) the Jevons' machine delivered, after each input, a result giving a temporary evaluation of the formula. This result contains 16 evaluations, on the 16 lines. The result of one basic evaluation is simply `true` or `false`. By giving these 16 results, the machine gives in a certain way the valuation which makes the formula `true`. At some pre-established instants, this evaluation is in adequation with the expected result. We may choose to consider results only after an AND (represented by a $\bullet$, separating sentences). We shall consider two techniques of evaluations: ours is based on an extension of the syntactic acceptor, then we shall give Jevons' proposal.

2.2.1 Our evaluation, based on syntactic recognition

We can give a value to any problem by the function Val. It gives a boolean result, based on the boolean values of the basic "Bo" atoms and the laws of boolean algebra. "Bo" stands for a boolean, 1 or 0. The description of Val is related to the grammar given in figure 2.

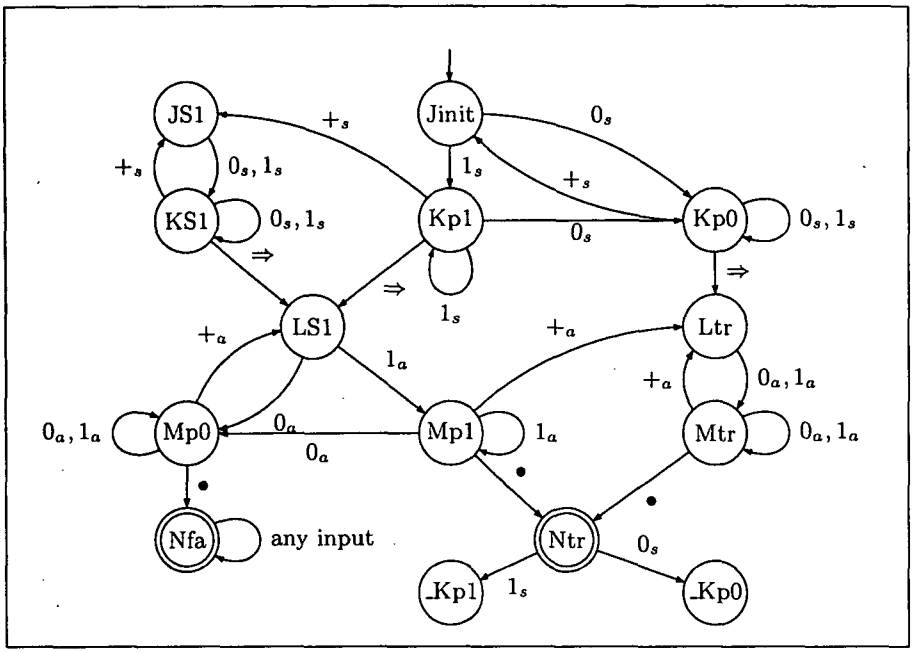


Figure 4: SE : Syntax-based Evaluation. States Kp1 and Kp0 are represented twice to make the figure easier to read. Ntr has the same successors as the initial state Jinit. In Ntr the expression evaluates to `true`, in Nfa, it evaluates to `false`

Val (Pr)	= Val (Sent) ;	Val (Pr)	= Val (Sent)	AND Val (Pr)
Val (Sent)	=		NOT Val (Sub)	OR Val (Attr)
Val (Sub)	= Val (Pro_s) ;	Val (Sub)	= Val (Pro_s)	OR Val (Sub)
Val (Attr)	= Val (Pro_a) ;	Val (Attr)	= Val (Pro_a)	OR Val (Attr)
Val (Pro_s)	= Val (Bo_s) ;	Val (Pro_s)	= Val (Bo_s)	AND Val (Pro_s)
Val (Pro_a)	= Val (Bo_a) ;	Val (Pro_a)	= Val (Bo_a)	AND Val (Pro_a)

We have extended the recognition automaton by considering the values of the interesting booleans evaluated in the machine. They are

- the current conjunction (or Product) of variables,
- the current disjunction (or Sum) of conjunctions, (and we must remember the subject Sum and the attribute Sum)
- these two sums give the value of the current sentence, by $x \Rightarrow y = \neg x \vee y$
- the current value of the conjunction (product) of sentences.

A further section (3.2) will describe the process to obtain this automaton. Let

us summarize the correspondence between the states of the syntactic recognizer (figure 3) and the states of our evaluator (figure 4) :

- In state J, either the subject is already certainly `true` after a first true product in a sum (state JS1), or the subject is not already certainly `true` (state Jinit).
- In state K, if the subject was already `true`, it remains (state KS1). If the subject is not yet `true`, either the current product is `true` (state Kp1) or the current product is `false` (state Kp0).
- In state L, either the subject was `true` (state LS1) or the sentence is already certainly `true` (state Ltr). This occurs either when the subject is `false` or when the subject is `true` and the attribute is already certainly `true`.
- In state M, if the sentence is already certainly `true`, it remains (state Mtr). In the other case, (the subject was certainly `true` and the attribute is not already certainly `true`), either the current product of the attribute is `false` (state Mp0) or this product is `true` (state Mp1).
- In state N, just after a •, the sentence is evaluated and the product of all the sentences is generated. When it has been `false` once (state Nfa), it remains false. If all the previous sentences have been `true` (state Ntr), evaluation goes on.

2.2.2 Jevons' evaluation

The goal of the present paper is not to present the method proposed by Jevons and its relations to "inductive" logic. This is done in ([6], [16] and [19]). It can also be understood from the original text. The basics is principally that we are interested in a prefix of expression. This prefix is a previous sentence, terminated by a •, followed by a premise. Then any line in the truth table can be classified in one of four categories with respect to the given prefix. (Line excluded by the premise, Line included and consistent with the premise, Line inconsistent with the premise, Line inconsistent with the previous sentence). These four situations can be modeled by four states.

We may consider how Jevons himself would have described the four states of one automaton (part 39) and the transitions due to action of the key "Full stop" (part 41).

> 39
> It is now necessary to explain that each rod has four possible positions fully indicated in the figs. [–]. The **first** of these positions is the neutral or initial position []. The **second** position is that into which a rod is thrown by a subject key ; the **third** position lies in the opposite direction, and is that into which a rod is thrown by a predicate key. The **fourth** position lies one half inch beyond the third. The **four** positions evidently correspond to the four classes into which combinations were classified in the previous part of the paper []

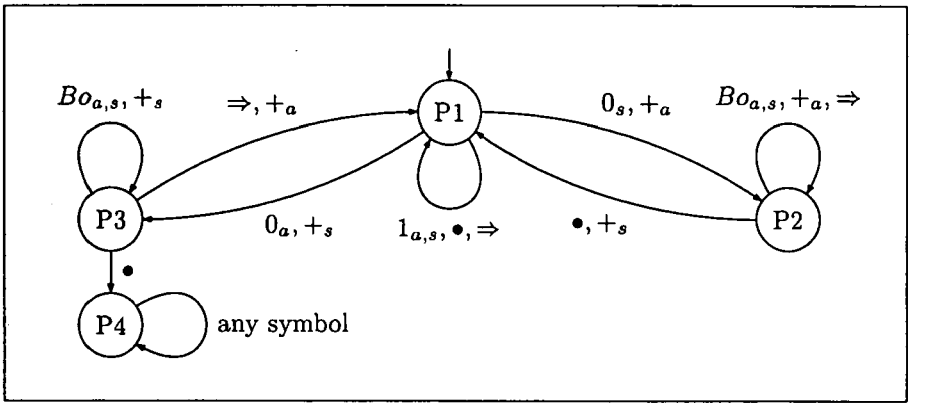


Figure 5: JE : Jevons' Evaluation : Reconstructed Jevons' automaton for evaluation. In state P1, the formula is `true`, in state P4, it is `false`. Bo is one of the booleans, 0 or 1.

His four positions are our four states P1, P2, P3 and P4 appearing in figure 5.

> 41
> The **full-stop** key being now pressed has a double effect. It acts [on the pins and rods of the machine] These pins we may distinguish as the α and β pins, the α pin being the uppermost. While a rod is in the **first** position the lever [] has **no effect** ; but if the rod be lowered $\frac{1}{2}$ inch into the **second** position, the lever will cause the rod to return to the **first** position by means of the α pin ; but if the rod be raised into the **third** position, the β pin will come into gear, and the rod will be pushed $\frac{1}{2}$ inch further into the **fourth** position.

Part 41 clearly describes a part of the transition function succ for the same input $\bullet$, and the four states.

$$\text{succ}(\text{P1}, \bullet) = \text{P1} \ ; \ \text{succ}(\text{P2}, \bullet) = \text{P1} \ ; \ \text{succ}(\text{P3}, \bullet) = \text{P4} \ ;$$

From these different explanations, we could infer the 4 state machine given by figure 5. In P1, the sentence is `true`. In P4, the sentence is `false`.

It is possible to follow the evaluation according to Jevons' technique on two tables: in these tables the initial state is P1. After a given input (first line), the new state is given under this input (second line).

For a formula giving a `true` result :

0_s	=>	0_a	$+_a$	$0_a 1_a$	$\bullet$	$0_s 0_s$	$+_s$	$1_s 0_s$	=>	$1_a 0_a$	$\bullet$
2	2	2	2	2 2	1	2 2	1	1 2	2	2 2	1

and similarly for a `false` result :

0_s	=>	1_a	$+_a$	0_a0_a	•	0_s1_s	$+_s$	1_s1_s	=>	0_a0_a	•
2	2	2	2	2 2	1	2 2	1	1 1	1	3 3	4

3 Jevons' Evaluation Coupled with Syntactic Recognition

Jevons did not verify the syntax of the formula entered on the keyboard. He was probably the only user of the machine in his courses of Logic. There were probably no syntax errors in his inputs! We have tried to compose evaluation and recognition by computing product automata. We used two evaluators: Jevons' one, of course, and an evaluator obtained by composing more basic evaluators.

3.1 Equivalence between two Evaluations

There are different definitions of the product automaton. We take the one of ([12], pp 134-137). The product computes the AND of the two composed automata. When we deal with recognition, it corresponds to intersection of languages. Here the interpretation is different, but AND is possible because the evaluator delivers a boolean (the value of the formula) and the recognizer can also be described with such a boolean output. If we name SA the syntactic analyser of figure 3 and JE the Jevons' evaluator of figure 5 the product SA × JE gives SE, the automaton of figure 4. (The correspondence between states is given by the Cartesian product of states in figure 6).

This composition is an interesting result. We can consider it as a validation of our syntactical evaluation method.

To enter more deeply into Jevons' technique, the reader may draw surrounding shapes on figure 4.

- One shape labelled P3 around states JS1, KS1, Mp0;
- One shape labelled P1 around Jinit, Kp1, LS1, Mp1, Ntr;
- One shape labelled P2 around Kp0, Ltr and Mtr.

The next section will give details about the design of SE.

3.2 Evaluation by Composition of Basic Evaluators

We have tried without success to obtain Jevons' automaton JE by composition of more basic understandable automata. We only obtained a composition evaluating *correct* formulas in the same way as Jevons' method (JE). We obtained CE our composed evaluator (figure 12), JE and CE are not equivalent. The (wrong) expression 0_s • does not give the same values in the two processes. If we compose them by the Syntax Analyser SA, the products SA × JE and SA × CE are equivalents. In both cases we obtain SE, the automaton of figure 4.

evaluation \ *syntax*	J	K	L	M	N
P1	Jinit	Kp1	LS1	Mp1	Ntr
P2		Kp0	Ltr	Mtr	
P3	JS1	KS1		Mp0	
P4	Nfa	Nfa	Nfa	Nfa	Nfa

Figure 6: Correspondence of states between the automaton of composite Syntax_Evaluation (fig. 4 SE) and the product of the Syntax Analyser (fig. 3 SA) by the Jevons' Evaluation automaton (fig. 5 JE)

Where does CE come from ? CE is the result of composing 5 automata. The organization of composition is given by figure 7. The composition has been computed with LUSTRE environment described in the next subsection. In the same way, the equivalences have been checked with this tool.

Automaton PROD (figure 8) is a Moore automaton, it receives all the inputs and delivers the product's value P.

Automaton SUM (figure 9) is a Mealy automaton, it receives symbols ($+_a$, $+_s$, $\Rightarrow$, $\bullet$) and the value P delivered by PROD. It delivers the sum of products value S.

Automata SUBJ (figure 10) and ATTR receive symbols ($\Rightarrow, \bullet$) and the value S of the sum of products. They deliver (respectively) the values Su and At of subject and attribute part. They are Mealy automata. SUBJ takes the value of S into account when $\Rightarrow$ occurs. In a symetric way, ATTR deals with S when $\bullet$ occurs.

Automaton EXPR (figure 11) is a Mealy automaton. It receives symbols ($\Rightarrow$, $\bullet$) and the values of Su and At. It delivers the global value Ex of Jevons' expression.

All these automata have two states as we could expect from boolean evaluators.

3.3 The Language Lustre and the Environment

The language LUSTRE has been designed in the '80s for real-time programming [10, 11]. The present description contains only some basic points useful to understand the composition made with the automata. The same approach is used for the environment. The use of LUSTRE in education is described in [4].

3.3.1 Boolean LUSTRE

Boolean LUSTRE has only one type : `boolean`. The boolean operations (`not`, `and`, `or` and `xor`) are defined in boolean LUSTRE. Two timed operators (`pre` and $- >$) allow us to deal with unitary delay and initialization. The synchronous hypothesis is that the automata update their states at the same clock ticks.

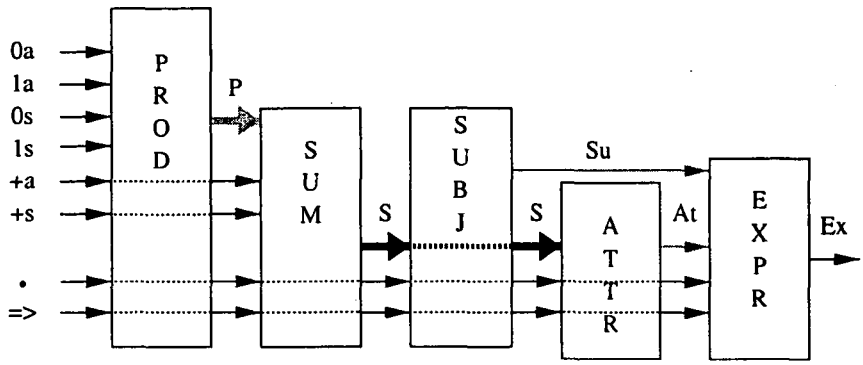


Figure 7: Composition of 5 basic automata to obtain an evaluator. It is inspired by the organization of a sequential circuit. Each input symbol is considered as a "wire", true or false at any instant. One and only one of these wires is true at any instant. These combinations represent the occurences of one symbol. P, S, Su and At are internal variables. The circuit would be a synchronous one, the clock being common.

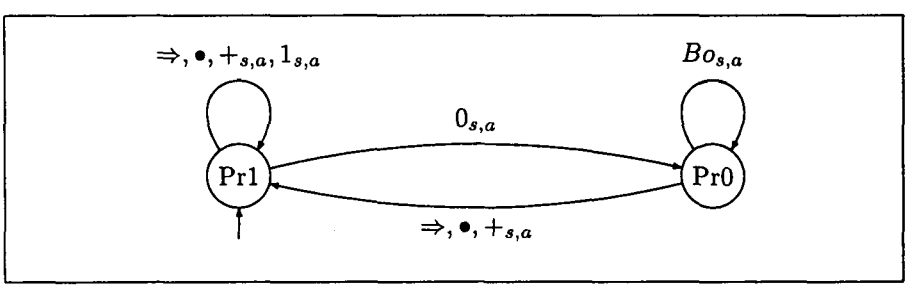


Figure 8: PROD : Evaluation of products. In state Pr1, the product P is 1, in state Pr0, the value is 0. The product is reset at 1 when a separator (⇒, •, +) occurs and this product becomes 0 only when a 0 boolean occurs.

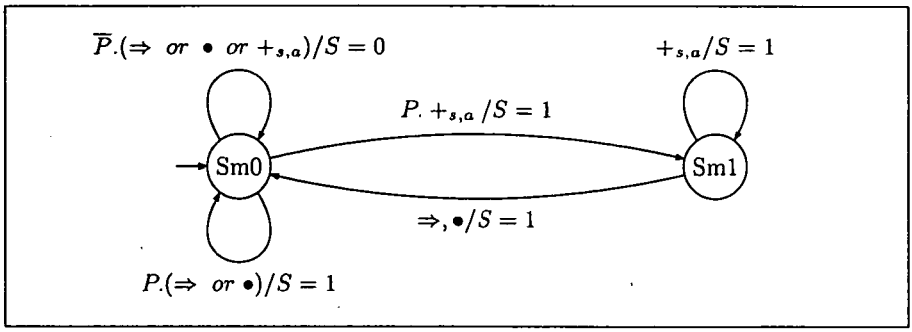


Figure 9: SUM : Evaluation of sum S, from the values of the products. P stands for (Product is 1) and $\overline{P}$ for (Product is 0). Booleans are not taken into account, updating only occurs on separator occurences.

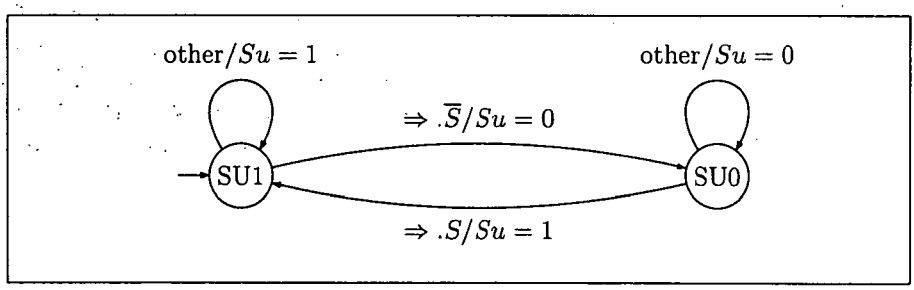


Figure 10: SUBJ : Evaluation of subjects Su from the value of the sum S. Updating occurs when $\Rightarrow$ occurs. Subject's value then receives the value of the sum S.

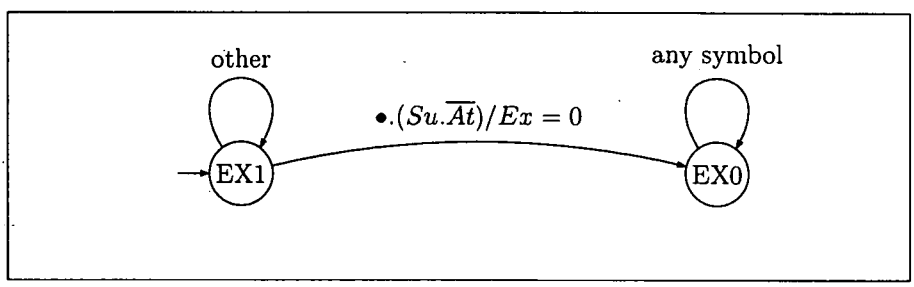


Figure 11: EXPR : Evaluation of a Jevons' expression Ex from Subject and Attribute's values. An expression remains `true` until occurence of a $\bullet$ when the current implication is `false`, i.e. the current subject Su is `true` and the current attribute At is `false`.

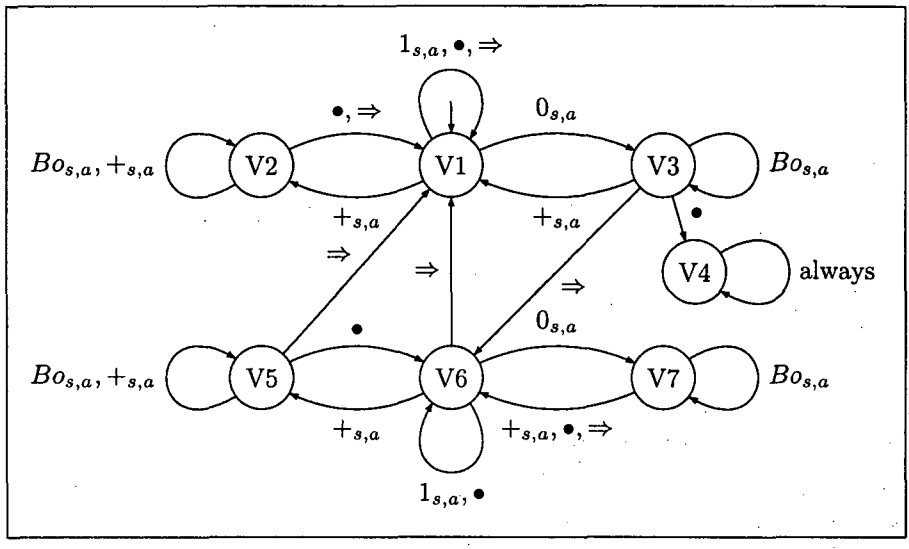


Figure 12: CE : Composed Evaluation. Evaluator obtained by combining PROD, SUM, SUBJ, ATTR, EXPR. In state V4, the expression is conclusively false. In states V1 and V6, it is true.

3.3.2 Basic descriptions of automata

Boolean LUSTRE makes it possible to describe finite automata in many different styles :

- The automaton can simply be described by a classical set of states, a description of the transition function and the description of the output function. This automaton may be deterministic or not, complete or not.
- We must introduce a comment about our mode of description of language recognizers. The only type being `boolean`, we cannot have a vocabulary based on characters. We solve this problem by introducing a set of boolean inputs such as {a, b, c, d}. We need to avoid the problem of 16 possible values of these four booleans. We use `assert` constructs to constrain one and only one amongst {a, b, c, d} to be true at any time.
- The general use of automata in formal languages studies distinguishes acceptor states and not_acceptor ones. This is obviously equivalent to having a boolean output defined in {0, 1} for each state. If we deal with more general automata, with not-simply boolean outputs, we may use the same approach. We then declare as many booleans as useful outputs.
- If the global automaton is not known, we can give *properties* of the automaton. This method is powerful but no systematic rules can be given. We may

experiment if the properties are adequate or not. Example of property is *For any transition due to input symbol X, a state and its successor never give the same output.* Experimentations could be simulation or formal proofs.

- When we deal with the *synchronous* sequential digital circuits, the description can easily be given in boolean LUSTRE. Logical gates are described by the operators. If we want to be close to the implementation we may describe `nand` or `nor` gates. Flip-flops are described by the timed operators.

- A systematic method of description of a regular grammar exists in boolean LUSTRE but there are restrictions on the form of the grammar: if A and B are non-terminal symbols and if x is a terminal, rules must be expressed in one of the two forms where we recognize initialization and unitary delay: $A \rightarrow \epsilon$ or $A \rightarrow B\ .\ x$

- A translator exists from a language allowing to describe regular expressions. This tool is described in [18].

3.3.3 Combinations of automata

Boolean LUSTRE allows to combine objects as it is the case in general LUSTRE. Different combinations of objects are possible :

- A very common case is that two automata A1, A2 are defined by LUSTRE nodes N1, N2 with the same inputs (*inp*). Both boolean LUSTRE nodes deliver one boolean output. A boolean operator `OP` allows us to define a new node as
  N3 (*inp*) = N1 (*inp*) `OP` N2 (*inp*). It creates a composed automaton A3. The language L3 recognized by A3 is a function of languages L1 and L2 recognized by A1 and A2. It also corresponds to the introduction of a logical gate on the two output signals of the two circuits.

  Correspondence between gates and language operations are obvious: `not` gate gives the complementary language, `and` gate gives the intersection of languages. ([12], p 135).
  `not xor` gate is particular. If two automata have always the same output, the composed automaton delivers always the output `true`. This corresponds to computing the equivalence of two automata. It is used to prove equivalence between two descriptions of automata that are assumed to be equivalent. (Similarly a $\Rightarrow$ operator is used to test inclusion of languages.)

- It is also possible to do cross-coupling of two nodes: some inputs of a node are outputs of the other one or vice-versa. It is very common in circuits. We must not include combinational loops.

- Any serial or parallel composition of automata may be described. An example appears in figure 7.

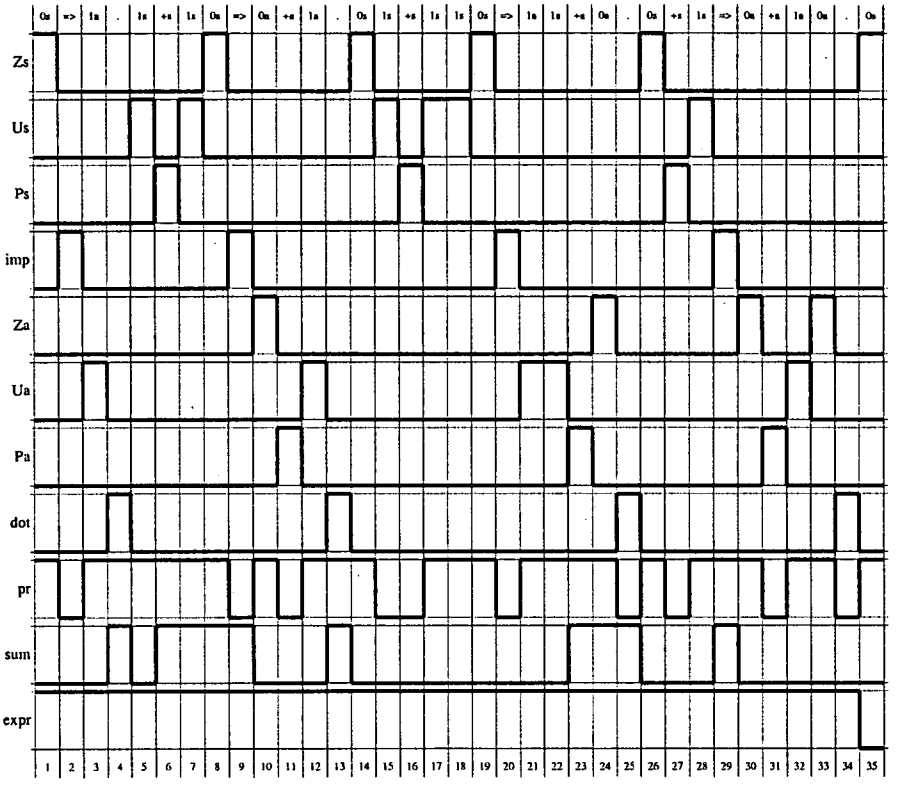


Figure 13: Simulation in Lustre. Zs represents 0_s, imp represents $\Rightarrow$, pr is the current value of the product. The input sequence begins with $0_s \Rightarrow 1_a \bullet$. The time slices are represented on the bottom line.

3.3.4 Environment and uses

The environment is available ([22]) under Solaris or Linux. Three tools are used in our work.

- The LUSTRE simulator allows to visualize the behaviour of the given object. The results are given in textual form or in timing diagrams form. This is particularly standard in digital circuits simulation. Figure 13 shows a simulation result of an evaluation. One character (represented by a boolean, true when occuring) is represented by one line of the oscilloscope.

- The LUSTRE combiner_minimizer computes the finite state machine described in input. The result of this compilation is a full definition: (list of states, list of all transitions). The result automaton is complete, deterministic and minimal. Obviously if we described a complete deterministic minimal automaton

as input, the compilation is only a state renaming ! It is particularly useful in composing automata. Obviously we must remain aware that the combinatorial explosion is possible.

- We use the Lesar tool in a particular case: for automata with only one boolean output, such as recognizers, Lesar computes if this output is always `true`. If it is not the case, Lesar gives a counterexample. This counterexample is very interesting when we test automata equivalence.

4 Conclusions

Obviously the contribution made by Jevons was an important step in the mechanization of Logic. The first machine devoted to artificial intelligence was his. The fact that syntactical aspects were not covered is easy to understand. But it is very pleasant to discover, by simple techniques, that his method could have been coupled with Finite State recognition. The present paper introduced the details about possibilities of such a composition with LUSTRE environment.

Part 46 of the original text opens a new problem: due to mechanical implementation, it was possible to press several keys simultaneously. Do we have to change automata theory to take such a feature into account?

> 46
> When several of the letter keys on the subject side only or the predicate side only are pressed in succession, the effect is to select the combinations possessing all the letters marked on the keys. Thus if the keys A, B, C be pressed there will remain
> o in the abecedarium only the combinations A B C D and A B C *d* ; and if the key o
> D be now pressed, the latter combination will disappear, and A B C D will alone remain. The effect will be exactly the same whatever the order in which the keys are pressed, and if they be pressed **simultaneously** there will be no difference in the result.

Acknowledgements

The drawing of automata used Latex packages made by Paul Gastin. The contribution of Lustre developers was obviously necessary.

References

[1] S.G. Akl, *Professor Jevons and his Logical Machine*, The Australian Computer Bulletin, June 1981, pp 28-30.

[2] P. Amblard, *A VLSI Implementation of the Earliest Specialized Logical Computer : the Jevons' Machine*, 4th International Workshop Mixed Design of integrated circuits and systems, Poznan, Poland, Mixdes97, June 1997, pp 55-66.

[3] P. Amblard, *The Earliest Formal Language and its Associated Finite State Evaluation Automaton : Jevons' Machine*, 11th International conference Automata and Formal Languages, Dobogókő, Hungary, May 2005, pp 59-68.

[4] P. Amblard *Using Lustre in Practical Educational Activities : Digital Circuits Design, Formal Languages*, ETAPS Workshop : Synchronous Language Applications Programming, SLAP 05 Edinburgh, April 2005.

[5] G.H. Buck, S.M. Munka, *W. Stanley Jevons, Allan Marquand, and the Origin of Digital Computing*, IEEE Annals of the History of Computing, Vol 21, No 4, October-December 1999, pp 21-27.

[6] S.N. Burris, *Contributions of the Logicians*, part 1 From Richard Whately to William Stanley Jevons, on-line : `www.thoralf.uwaterloo.ca/`

[7] I. Casltes, *Vice President's note*, Newsletter of the Academy of the Social Sciences in Australia, Vol 17, No 3, 1998, pp 5-12; `www.assa.edu.au/publications/Dialogue/dial31998.pdf`

[8] M. Gardner, *Logic Machines*, Scientific American, March 1952, pp 68-73.

[9] M. Gardner, **Logic Machines and Diagrams**, McGraw-hill 1958, and The Harvester Press, Brighton, 1983

[10] N. Halbwachs, **Synchronous Programming of Reactive Systems**, Kluwer Academic Pub., 1993

[11] N. Halbwachs, P. Caspi, P. Raymond and D. Pilaud, *The Synchronous Dataflow Programming Language Lustre*, Proceedings of the IEEE, September 1991, pp 1305-1320.

[12] J.E. Hopcroft, R. Motwani, J.D. Ullman, **Introduction to Automata Theory, Languages and Computation**, Addison Wesley, 2001.

[13] B. Imreh, *Some Remarks on Asynchronous Automata*, Conference DLT 2002, Lect. Notes in Comp. Science No 2450, pp 290-296.

[14] William Stanley Jevons, *On the Mechanical Performance of Inference*, Philosophical Transactions of the Royal Society, London, 1870, pp 497-518. Available on-line : `tima-cmp.imag.fr/~amblard/JEVONS/jevons_public.pdf`

[15] W. S. Jevons, **Papers and Correspondence**, (ed : R.D. Collison Black and Rosamond Kőnekamp), Vol 3, (correspondence 1863-1872), MacMillan Press, (London), 1977, pp 69-76.

[16] W. Mays, D. P. Henry, *Jevons and Logic*, Mind, Vol LXII, 1953, T. Nelson & sons, Edinburgh, pp 484-505.

[17] E. F. Moore, *Gedanken-experiments on Sequential Machines* in **Automata studies**, (Ed : C. Shannon, J. McCarthy) Princeton University Press, 1956, pp 129-153.

[18] P. Raymond, *Recognizing Regular Expressions by means of Dataflows Networks*, 23rd International Colloquium on Automata, Languages, and Programming, (ICALP'96) Paderborn, Germany, July 1996, LNCS 1099.

[19] J.C. Shepherdson, *W. S. Jevons: his Logical Machine and Work on Induction and Boolean Algebra*, Machine Intelligence 15, Oxford, July 1995, pp 489-505.

[20] **Dictionnary of National Biography**, vol XXIX, Smith, Elder and co, London, 1892, pp 374-378.

[21] **Sequential Machines, Selected papers**, Ed : E. F. Moore, Addison-Wesley, 1964.

[22] Web-site : `www-verimag.imag.fr/SYNCHRONE`

Acta Cybernetica 17 (2006) 685–700.

Parallel Communicating Watson-Crick Automata Systems

Elena Czeizler* and Eugen Czeizler*

Abstract

Watson-Crick automata are finite state automata working on double-stranded tapes, introduced to investigate the potential of DNA molecules for computing. In this paper we introduce the concept of parallel communicating Watson-Crick automata systems. It consists of several Watson-Crick finite automata parsing independently the same input and exchanging information on request, by communicating states to each other. We investigate the computational power of these systems and prove that they are more powerful than classical Watson-Crick finite automata, but still accepting at most context-sensitive languages. Moreover, if the complementarity relation is injective, then we obtain that this inclusion is strict. For the general case, we also give some closure properties, as well as a characterization of recursively enumerable languages based on these systems.

1 Introduction

Watson-Crick finite automata, introduced in [5], are a counterpart of finite automata working on double stranded sequences. As suggested by the name, these automata are mainly inspired from molecular computing and are intended as a formalization of DNA manipulation. The two strands of the input are separately scanned from left to right by read only heads controlled by a common state. The characters on the corresponding positions from the two strands are linked by a complementarity relation, inspired from the Watson-Crick complementarity of DNA nucleotides. Several variants of these automata were investigated in [11, 12, 13, 15], see also [14] for a comprehensive presentation.

Distributed computations play a major role in modern computer science; multiprocessor computers, distributed data bases, computer networks, etc., introduced notions such as distribution, parallelism, and communication. The theory of grammar and automata systems was developed as a mathematical model for distributed and parallel computations.

*Department of Mathematics, University of Turku and Turku Centre for Computer Science, Turku 20520, Finland, E-mail: elena.czeizler@utu.fi, euczei@utu.fi

An automata system is a set of automata working together on the same input, according to a well specified protocol, in order to accept one language. There are two basic classes of automata systems: *sequential* and *parallel*.

The sequential class is represented by *cooperating distributed* automata systems. Here, all components work on a common input tape and at each step only one automaton is active. An example of such systems is the *cooperating distributed push-down automata system*, introduced and studied in [3].

A *parallel communicating automata system* is a construct consisting of several automata working synchronously, each on its own input tape, and communicating on request. Special *query states* are provided, each of them pointing to exactly one component of the system. When a component i of the system reaches a query state K_j, the current state of the component j will be communicated to i and the computation continues. There are two important classifications of parallel communicating systems concerning the *communication graph* and the *returning feature*. An automata system is called *centralized* if only one component, the *master*, may introduce query states, and *non-centralized* otherwise. An automata system is called *returning* if after communicating, a component resumes the computation from its initial state, and *non-returning* if it remains in its current state. There are several papers in the literature investigating this class of systems. For example, parallel communicating push-down automata systems communicating by stacks were introduced in [2] and parallel communicating automata systems communicating by states were introduced in [10], see also [9] for a survey.

Cooperating distributed Watson-Crick automata systems were investigated in [1], where it was proved that distribution does not bring any change in the acceptance power of Watson-Crick finite automata, except for the case of one state automata, i.e. *stateless Watson-Crick automata*.

In this paper we introduce the notion of *parallel communicating Watson-Crick automata system* as a set of Watson-Crick finite automata working independently on their own input tape and communicating states on request. We consider only non-centralized and non-returning systems. Although every component has its own double-stranded tape, the input is the same on all of them. At the beginning, all components are in their initial states and start parsing synchronously the input from left to right. The communication between components is done using query states as described before for general parallel communicating automata systems. An input is accepted by the system if all components are in final states when they completely parsed the tape. Moreover, if one of the components stops before the others, the system halts and rejects the input. Hence, in order to accept, the components either finish at the same time or wait for each other at the end of the computation.

Combining the notions of Watson-Crick automata and parallel communicating systems comes naturally due to the new developments in DNA manipulation techniques. While classical Watson-Crick finite automata use just one of the essential features of DNA, i.e. the *Watson-Crick complementarity*, the systems introduced here open new possibilities in exploiting also the *massive parallelism* of DNA computations.

The structure of the paper is as follows. In Section 2 we fix our terminology and introduce some basic notions and results. Section 3 is devoted to the computational power of these systems. We start by giving an example of a parallel communicating Watson-Crick automata system proving that the accepting power is enhanced. We also prove that the languages accepted by these systems are at most context-sensitive. Moreover, if the complementarity relation is injective, as in the case of DNA nucleotides, then one letter-languages accepted by these systems are regular. In Section 4 we investigate some closure properties. We also give a characterization of recursively enumerable languages based on these systems. In Section 5 we present some open problems.

2 Preliminaries

In this section we give basic definitions and some already known results we need later on. We start by considering the classical Watson-Crick finite automata introduced in [5] and then define the parallel communicating version. We assume that the reader is familiar with the fundamental concepts from formal languages and automata theory. For more details we refer to [7], [14], and [16].

For a finite set Q, let $card(Q)$ and 2^Q denote the cardinality and the power set of Q, respectively. Let V be a finite alphabet. We denote by V^* the set of all finite words over V, by λ the empty word, and by V^+ the set of all nonempty finite words over V, $V^+ = V^* \backslash \{\lambda\}$. For $w \in V^*$ we denote by $|w|$ the length of w.

Given two alphabets V and U, we define a *morphism* as a function $h : V \to U^*$, extended to $h : V^* \to U^*$ by $h(\lambda) = \lambda$ and $h(w_1 w_2) = h(w_1)h(w_2)$, for $w_1, w_2 \in V^*$. If $h(a) \neq \lambda$ for each $a \in V$, then we say that h is a *λ-free* morphism. We define a *projection* associated to the alphabet V as the morphism $pr_V : (V \cup U)^* \to V^*$ such that $pr_V(a) = a$ for all $a \in V$ and $pr_V(a) = \lambda$ otherwise. For two morphisms $h_1, h_2 : V^* \to U^*$, we define the *equality set* of h_1, h_2 as:

$$EQ(h_1, h_2) = \{w \in V^* \mid h_1(w) = h_2(w)\}.$$

Let now $\rho \subseteq V \times V$ be a symmetric relation, called *the Watson-Crick complementarity relation on V*. As suggested by the name, this relation is biologically inspired by the Watson-Crick complementarity of nucleotides in the double stranded DNA molecule. We say that ρ is injective if any $a \in V$ has a unique complementary symbol $b \in V$ with $(a, b) \in \rho$. In accordance with the representation of DNA molecules, viewed as two strings written one over the other, we write $\binom{V^*}{V^*}$ instead of $V^* \times V^*$ and an element $(w_1, w_2) \in V^* \times V^*$ as $\binom{w_1}{w_2}$.

We denote $\begin{bmatrix} V \\ V \end{bmatrix}_\rho = \{\begin{bmatrix} a \\ b \end{bmatrix} \mid a, b \in V,\ (a, b) \in \rho\}$ and $WK_\rho(V) = \begin{bmatrix} V \\ V \end{bmatrix}_\rho^*$. The set $WK_\rho(V)$ is called *the Watson-Crick domain* associated to V and ρ. An element

$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \dots \begin{bmatrix} a_n \\ b_n \end{bmatrix} \in WK_\rho(V)$ can be also written in a more compact form as $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$, where $w_1 = a_1 a_2 \dots a_n$ and $w_2 = b_1 b_2 \dots b_n$.

The essential difference between $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ and $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ is that $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ is just an alternative notation for the pair (w_1, w_2), whereas $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ implies that the strings w_1 and w_2 have the same length and the corresponding letters are connected by the complementarity relation.

A *Watson-Crick finite automaton* is a 6-tuple $\mathcal{M} = (V, \rho, Q, q_0, F, \delta)$, where:

- V is the (input) alphabet,
- $\rho \subseteq V \times V$ is the complementarity relation,
- Q is a finite set of states,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states,
- $\delta : Q \times \begin{pmatrix} V^* \\ V^* \end{pmatrix} \to 2^Q$ is a mapping, called the *transition function*, such that $\delta(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) \neq \emptyset$ only for finitely many triples $(q, w_1, w_2) \in Q \times V^* \times V^*$.

We can replace the transition function with rewriting rules, by using

$$s \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s' \text{ instead of } s' \in \delta(s, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}).$$

We define transitions in a Watson-Crick finite automaton as follows. For $\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \in \begin{pmatrix} V^* \\ V^* \end{pmatrix}$ such that $\begin{bmatrix} v_1 u_1 w_1 \\ v_2 u_2 w_2 \end{bmatrix} \in WK_\rho(V)$ and $s, s' \in Q$ we write

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} s \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \Rightarrow \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} s' \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

if and only if $s' \in \delta(s, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix})$. If we denote by $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$, then the language accepted by a Watson-Crick automaton is:

$$L(\mathcal{M}) = \{w_1 \in V^* \mid q_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} s, \text{ with } s \in F, \ w_2 \in V^*,$$

$$\text{and } \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)\}.$$

Hence, a word w_1 is accepted by $\mathcal{M}$ if starting from the initial state, after parsing the whole input $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ we are in a final state.

Let us continue now by defining *parallel communicating Watson-Crick automata systems.*

A parallel communicating Watson-Crick automata system of degree n, denoted by $PCWK(n)$, is an $(n+3)$-tuple

$$\mathcal{A} = (V, \rho, A_1, A_2, \ldots, A_n, K),$$

where

- V is the input alphabet;
- ρ is the complementarity relation;
- $A_i = (V, \rho, Q_i, q_i, F_i, \delta_i)$, $1 \leq i \leq n$, are Watson-Crick finite automata, where the sets Q_i are not necessarily disjoint;
- $K = \{K_1, K_2, \ldots, K_n\} \subseteq \cup_{i=1}^{n} Q_i$ is the set of query states.

The automata $A_1, A_2, \ldots, A_n$ are called the *components* of the system $\mathcal{A}$. Note that any Watson-Crick finite automaton can be viewed as a parallel communicating Watson-Crick automata system of degree 1.

A configuration of a parallel communicating Watson-Crick automata system is a $2n$-tuple $(s_1, \begin{pmatrix} u_1 \\ v_1 \end{pmatrix}, s_2, \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}, \ldots, s_n, \begin{pmatrix} u_n \\ v_n \end{pmatrix})$ where s_i is the current state of the component i and $\begin{pmatrix} u_i \\ v_i \end{pmatrix}$ is the part of the input word which has not been read yet by the component i, for all $1 \leq i \leq n$. We define a binary relation $\vdash$ on the set of all configurations by setting

$$(s_1, \begin{pmatrix} u_1 \\ v_1 \end{pmatrix}, s_2, \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}, \ldots, s_n, \begin{pmatrix} u_n \\ v_n \end{pmatrix}) \vdash (r_1, \begin{pmatrix} u_1' \\ v_1' \end{pmatrix}, r_2, \begin{pmatrix} u_2' \\ v_2' \end{pmatrix}, \ldots, r_n, \begin{pmatrix} u_n' \\ v_n' \end{pmatrix})$$

if and only if one of the following two conditions holds:

- $K \cap \{s_1, s_2, \ldots, s_n\} = \emptyset$, $\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \begin{pmatrix} u_i' \\ v_i' \end{pmatrix}$, and $r_i \in \delta_i(s_i, \begin{pmatrix} x_i \\ y_i \end{pmatrix})$, $1 \leq i \leq n$;
- for all $1 \leq i \leq n$ such that $s_i = K_{j_i}$ and $s_{j_i} \notin K$ we have $r_i = s_{j_i}$, whereas for all the other $1 \leq l \leq n$ we have $r_l = s_l$. In this case $\begin{pmatrix} u_i' \\ v_i' \end{pmatrix} = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$, for all $1 \leq i \leq n$.

If we denote by $\vdash^*$ the reflexive and transitive closure of $\vdash$, then the language recognized by a parallel communicating Watson-Crick automata system $\mathcal{A}$ is

defined as:

$$L(\mathcal{A}) = \{w_1 \in V^* \mid (q_1, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, q_2, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \ldots, q_n, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}) \vdash^*$$
$$(s_1, \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}, s_2, \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}, \ldots, s_n, \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}),\ s_i \in F_i,\ 1 \leq i \leq n\}.$$

Intuitively, the language accepted by such a system consists of all words w_1 such that in every component we reach a final state after reading all input $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$.

In the next section we study the connection between the family of languages accepted by parallel communicating Watson-Crick automata systems and the class of *context-sensitive* languages. For doing this we use a special type of automata characterizing this class of languages.

Linearly bounded automata are a special class of Turing machines which have two extra symbols in their input alphabet, say # and \$, called the left and right end-markers, respectively. The automaton can neither overwrite these markers nor move left or right from them. Hence, this type of automata uses only a limited amount of tape. Similarly, *k-head linearly bounded automata* are a special class of k-tape Turing machines which use only limited amount of each tape. On each step the k heads move independently, according to the state of the automaton and the symbol read on each individual tape.

The following two results are well known in the literature, see for example [6] and [7].

Theorem 1. *$L \subseteq V^*$ is a context-sensitive language if and only if it is accepted by a linearly bounded automaton.*

We say that an automaton A is of *space complexity* $S(n)$ if, for every accepted input of length n there is some accepting computation in which at most $S(n)$ tape-cells are parsed by any read-write head.

Theorem 2. *If a language L is accepted by a k-tape Turing machine of space complexity $S(n)$, then L is accepted by some one-tape Turing machine of the same space complexity.*

The following lemma comes as a direct consequence of the previous two results and will be used in our future considerations.

Lemma 3. *A language L is context-sensitive if and only if it is accepted by a k-head linearly bounded automaton.*

3 Computational power

Let us start by giving an example of a language accepted by a parallel communicating Watson-Crick automata system of degree 3.

$$\begin{array}{lll}
\delta_1(q_1, \binom{x}{\lambda}) = q_1 & \delta_2(q_2, \binom{x}{x}) = q_2 & \delta_3(q_3, \binom{x}{x}) = q_3 \\
\delta_1(q_1, \binom{\#}{\lambda}) = r_1 & \delta_2(q_2, \binom{\#}{\#}) = p_1 & \delta_3(q_3, \binom{\#}{\#}) = s_1 \\
\delta_1(r_1, \binom{x}{\lambda}) = r_1 & \delta_2(p_1, \binom{x}{\lambda}) = p_1 & \delta_3(s_1, \binom{x}{x}) = s_1 \\
\delta_1(r_1, \binom{\#}{\lambda}) = r_2 & \delta_2(p_1, \binom{\#}{\lambda}) = p_2 & \delta_3(s_1, \binom{\#}{\#}) = s_2 \\
\delta_1(r_2, \binom{x}{\lambda}) = r_2 & \delta_2(p_2, \binom{x}{\lambda}) = p_2 & \delta_3(s_2, \binom{x}{\lambda}) = s_2 \\
\delta_1(r_2, \binom{\#}{\lambda}) = r_3 & \delta_2(p_2, \binom{\#}{\lambda}) = p_3 & \delta_3(s_2, \binom{\#}{\lambda}) = s_3 \\
\delta_1(r_3, \binom{x}{\lambda}) = r_3 & \delta_2(p_3, \binom{x}{\lambda}) = p_3 & \delta_3(s_3, \binom{x}{x}) = s_3 \\
\delta_1(r_3, \binom{\#}{\lambda}) = r_4 & \delta_2(p_3, \binom{\#}{\lambda}) = p_4 & \delta_3(s_3, \binom{\#}{\#}) = f_3 \\
\delta_1(r_4, \binom{x}{\lambda}) = r_4 & \delta_2(p_4, \binom{x}{x}) = p_4 & \delta_3(f_3, \binom{y}{z}) = f_3 \\
\delta_1(r_4, \binom{\#}{\lambda}) = r_5 & \delta_2(p_4, \binom{\#}{\#}) = f_2 & \\
\delta_1(r_5, \binom{x}{x}) = r_5 & \delta_2(f_2, \binom{y}{z}) = f_2 & \\
\delta_1(r_5, \binom{\lambda}{\#}) = f_1 & & \\
\delta_1(f_1, \binom{\lambda}{y}) = f_1 & &
\end{array}$$

with $x \in \{a, b\}$, $y, z \in \{a, b, \#, \lambda\}$

Table 1: The transition functions of Example 4

Example 4. Let $\mathcal{A} = (\{a, b, \#\}, \rho, A_1, A_2, A_3, \emptyset)$, where ρ is the identity relation, i.e., $\rho = \{(a, a), (b, b), (\#, \#)\}$, $A_1 = (\{a, b, \#\}, \rho, \{q_1, r_1, r_2, r_3, r_4, r_5, f_1\}, q_1, \{f_1\}, \delta_1)$, $A_2 = (\{a, b, \#\}, \rho, \{q_2, p_1, p_2, p_3, p_4, f_2\}, q_2, \{f_2\}, \delta_2)$, and $A_3 = (\{a, b, \#\}, \rho, \{q_3, s_1, s_2, s_3, f_3\}, q_3, \{f_3\}, \delta_3)$. The transition functions of the three components are defined in Table 1.

The system works as follows. The first component verifies that the input is of the form $\begin{bmatrix} w_1\#w_2\#w_3\#w_4\#w_5\#w_6 \\ w_1\#w_2\#w_3\#w_4\#w_5\#w_6 \end{bmatrix}$ with $w_i \in \{a, b\}^+$ for all $1 \leq i \leq 6$, and moreover $w_1 = w_6$. Simultaneously the second and the third component impose the constraints $w_2 = w_5$ and $w_3 = w_4$, respectively. Thus, the language accepted by $\mathcal{A}$ is $L = \{w_1\#w_2\#w_3\#w_3\#w_2\#w_1 \mid w_1, w_2, w_3 \in \{a, b\}^*\}$.

On the other hand, it was proved in [18] that the language L cannot be accepted by a 2-head finite automaton. Since Watson-Crick automata are equivalent with 2-head finite automata, see [14], we have the following result.

Theorem 5. *Parallel communicating Watson-Crick automata systems are more powerful than Watson-Crick finite automata.*

$$\delta_1(q_1, \begin{pmatrix} xy \\ z \end{pmatrix}) = q_1 \text{ for any } x, y, z \in V \qquad \delta_2(q_2, \begin{pmatrix} xy \\ \lambda \end{pmatrix}) = q_2 \text{ for any } x, y \in V$$

$$\delta_1(q_1, \begin{pmatrix} \# \\ x \end{pmatrix}) = q_x \text{ for any } x \in V \qquad \delta_2(q_2, \begin{pmatrix} \# \\ \lambda \end{pmatrix}) = K_1$$

$$\delta_1(q_x, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = K_2 \qquad \delta_2(q_x, \begin{pmatrix} \lambda \\ x \end{pmatrix}) = q_3 \text{ for any } x \in V$$

$$\delta_1(q_3, \begin{pmatrix} \lambda \\ x \end{pmatrix}) = q_x \text{ for any } x \in V \qquad \delta_2(q_3, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = K_1$$

$$\delta_1(q_3, \begin{pmatrix} \lambda \\ \# \end{pmatrix}) = q_{f_1} \qquad \delta_2(q_{f_1}, \begin{pmatrix} \lambda \\ x \end{pmatrix}) = q_{f_2} \text{ for any } x \in V$$

$$\delta_1(q_{f_1}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_{f_1} \qquad \delta_2(q_{f_2}, \begin{pmatrix} \lambda \\ x \end{pmatrix}) = q_{f_2} \text{ for any } x \in V \cup \{\#\}$$

Table 2: The transition functions of Example 6

Next, let us illustrate the communication between components by considering a parallel communicating Watson-Crick automata system accepting the non context-free language $L = \{ww\# \mid w \in V^+\}$, where $\# \notin V$ and $|V| \geq 2$.

Example 6. Let $\mathcal{A} = (V \cup \{\#\}, \rho, A_1, A_2, K)$ be a parallel communicating Watson-Crick automata system where $\rho = \{(a,a) \mid a \in V\} \cup \{(\#,\#)\}$, $K = \{K_1, K_2\}$, $A_1 = (V \cup \{\#\}, \rho, Q_1, q_1, \{q_{f_1}\}, \delta_1)$, and $A_2 = (V \cup \{\#\}, \rho, Q_2, q_2, \{q_{f_2}\}, \delta_2)$. The sets of states are $Q_1 = \{q_1, q_3, q_{f_1}, K_2\} \cup \{q_x \mid x \in V\}$ and $Q_2 = \{q_2, q_3, q_{f_1}, q_{f_2}, K_1\} \cup \{q_x \mid x \in V\}$, while the transition functions are defined in Table 2.

The first component finds the middle of the input word, by placing the two reading heads one at the end and the other in the middle of the input word. In parallel, to preserve the synchronization, the second component moves one reading head to the end of the input while the other one remains unmoved. At the same time we also check that the input has odd length. Then, by using communication between components we check letter by letter that the input is indeed of the form $\begin{bmatrix} ww\# \\ ww\# \end{bmatrix}$.

A natural question regarding these systems is the relation between the languages they accept and the family of context-sensitive languages.

We first need a generalization of a result already known for Watson-Crick finite automata, see [14].

Lemma 7. *Every parallel communicating Watson-Crick automata system is equivalent with a system where in every component we have only rules of the form* $s \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s'$, *with* $|w_1 w_2| \leq 1$.

Proof. Let $\mathcal{A} = (V, \rho, A_1, \ldots, A_n, K)$ be a parallel communicating Watson-Crick automata system with n components, where $A_i = (V, \rho, Q_i, q_i, F_i, \delta_i)$ for all $1 \leq i \leq n$. Let us first index by unique labels all transitions from all components and

define the constant $m = max\{|w_1| + |w_2| \mid s\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s'$ is a production in one of the components of the system$\}$.

We construct a parallel communicating Watson-Crick automata system $\mathcal{A}' = (V, \rho, A'_1, \ldots, A'_n, K)$, where $A'_i = (V, \rho, Q'_i, q_i, F_i, \delta'_i)$ are obtained from A_i as follows.

Let $s\begin{pmatrix} w_1 \ldots w_p \\ w'_1 \ldots w'_{p'} \end{pmatrix} \to \begin{pmatrix} w_1 \ldots w_p \\ w'_1 \ldots w'_{p'} \end{pmatrix} s'$, with $w_1, \ldots, w_p, w'_1, \ldots, w'_{p'} \in V$ be a transition rule from A_i, indexed with the unique label j. Then, in A'_i we introduce m new states $r_1^j, r_2^j, \ldots, r_m^j$ and the following transitions:

$$s\begin{pmatrix} w_1 \\ \lambda \end{pmatrix} \to \begin{pmatrix} w_1 \\ \lambda \end{pmatrix} r_1^j, \quad \ldots, \quad r_{p-1}^j\begin{pmatrix} w_p \\ \lambda \end{pmatrix} \to \begin{pmatrix} w_p \\ \lambda \end{pmatrix} r_p^j,$$

$$r_p^j\begin{pmatrix} \lambda \\ w'_1 \end{pmatrix} \to \begin{pmatrix} \lambda \\ w'_1 \end{pmatrix} r_{p+1}^j, \quad \ldots, \quad r_{p+p'-1}^j\begin{pmatrix} \lambda \\ w'_{p'} \end{pmatrix} \to \begin{pmatrix} \lambda \\ w'_{p'} \end{pmatrix} r_{p+p'}^j,$$

$$r_{p+p'}^j\begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \to \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} r_{p+p'+1}^j, \quad \ldots, \quad r_m^j\begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \to \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} s'.$$

Thus, any transition from A_i is replaced in A'_i by $m+1$ transitions of the form requested by the lemma. Also, since this construction preserves the synchronization between components, the system $\mathcal{A}'$ recognizes the same language as $\mathcal{A}$ but with linear time delay. □

Theorem 8. *The family of languages accepted by parallel communicating Watson-Crick automata systems is included in the family of context-sensitive languages.*

Proof. We can assume without loss of generality, that all components of the parallel communicating Watson-Crick automata system have only rules of the form described in Lemma 7. Then, for any such system $\mathcal{A}$ of degree n we can construct a $2n$-tape linearly bounded automaton $\mathcal{M}$ which recognizes the same language. Each $2p + i$ tape, with $0 \leq p \leq n - 1$ and $1 \leq i \leq 2$, simulates the i-th tape of the $(p + 1)$-th component of $\mathcal{A}$. All the states in $\mathcal{M}$, except the final one, encode information about the states of all n components of system $\mathcal{A}$. At each computational step, we read a character on each tape and either move the reading head one step to the right or remain on the same position, according to the evolution of system $\mathcal{A}$. For query steps, we just modify the information encoded in the state, i.e., we enter a new state in $\mathcal{M}$, whereas the input and positions of the reading heads remain unchanged. The final state of $\mathcal{M}$ is reached only from states encoding the information that all components of system $\mathcal{A}$ are in final states and all the $2n$ reading heads are positioned on the right end marker.

From this construction we obtain that automaton $\mathcal{M}$ accepts the same language as system $\mathcal{A}$. Hence, from Lemma 3, we obtain that $L(\mathcal{A})$ is a context-sensitive language. □

So far we considered only the general case where the complementarity relation ρ has no restrictions, except for symmetry. However, in [17] the case of an injective complementarity relation inspired by the real Watson-Crick complementarity of DNA nucleotides was discussed. For the rest of this section we restrict ourselves to this particular case. In order to investigate the computational power of these systems, we relate them to *k-head automata*, as they were defined in [10].

A *k-head automaton* is a 6-tuple $\mathcal{M} = (k, Q, V, f, q_0, F)$ where Q is the set of states, V is the input alphabet, $f : Q \times (V \cup \{\lambda\})^k \to 2^Q$ is the transition function, q_0 is the initial state, and $F \subseteq Q$ is the set of final states. Any computation starts in the initial state and with all the reading heads on the leftmost character of the input. Then, for any transition $q \in f(s, a_1, a_2, \ldots, a_k)$ and all $1 \leq i \leq k$, the i-th head reads a_i from the input tape and the automaton passes from state s into state q. A word w is accepted if after finitely many moves the automaton enters a final state, the input being completely read by all heads. In all the other cases the input word is rejected.

Theorem 9. *Any language recognized by a parallel communicating Watson-Crick automata system of degree n, with injective complementarity relation, can be also recognized by a 2n-head automaton.*

Proof. For the clarity of the proof, we consider only systems of degree 2, whereas the reasoning remains the same for the general case.

Let $\mathcal{A} = (V, \rho, A_1, A_2, K)$ be a parallel communicating Watson-Crick automata system of degree 2, accepting the language $L \subseteq V^*$, where $A_1 = (V, \rho, Q_1, q_1, F_1, \delta_1)$, $A_2 = (V, \rho, Q_2, q_2, F_2, \delta_2)$, and $K = \{K_1, K_2\}$. Since the relation ρ is injective, we can take it to be the identity relation; thus all components have on both tapes the same word $w \in V^*$. Also, by Lemma 7 we can suppose that in every component we have only rules of the form $s \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \to \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} s'$, with $|w_1 w_2| \leq 1$.

Let us construct now a 4-head automaton $\mathcal{M} = (4, Q, V, f, q_0, F)$ where $Q = Q_1 \times Q_2$, $q_0 = (q_1, q_2)$, $F = F_1 \times F_2$, and the transition function f is as follows:

- $f((p, q), w_1, w_2, w_3, w_4) = (p_1, q_1)$ whenever $p, q \notin K$, $\delta_1(p, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) = p_1$, and $\delta_2(q, \begin{pmatrix} w_3 \\ w_4 \end{pmatrix}) = q_1$;
- $f((K_2, q), \lambda, \lambda, \lambda, \lambda) = (q, q)$;
- $f((p, K_1), \lambda, \lambda, \lambda, \lambda) = (p, p)$.

At any step the automaton $\mathcal{M}$ simulates the corresponding moves of the two components of $\mathcal{A}$. If the components are not in a query state and they read $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ and $\begin{pmatrix} w_3 \\ w_4 \end{pmatrix}$ respectively from the input tape, with $|w_1 w_2| \leq 1$ and $|w_3 w_4| \leq 1$, then in $\mathcal{M}$ each head reads w_1, w_2, w_3, and w_4, respectively, and it enters into the

corresponding state. Otherwise, i.e., we are in a state (K_2, s) or (s, K_1), we just simulate the query by entering state (s, s) and leaving the input unchanged. Since a word is accepted by $\mathcal{M}$ only if it is in a final state when all the reading heads have finished parsing the input, then $w \in L(\mathcal{A})$ implies $w \in L(\mathcal{M})$ and hence $L(\mathcal{A}) \subseteq L(\mathcal{M})$.

Let now w be a word accepted by $\mathcal{M}$. From the construction of the transition function f, each computational step in $\mathcal{M}$ can be translated into a computational step in $\mathcal{A}$ when we consider the input $\begin{bmatrix} w \\ w \end{bmatrix}$. Moreover after the final computational step, all 4 heads of $\mathcal{M}$ have completely read the input and the automaton is in a final state. This implies that at the same step both components of system $\mathcal{A}$ are in final states, while their reading heads from the lower and from the upper strands have completely parsed the input. So, we have $w \in L(\mathcal{A})$ and hence $L(\mathcal{M}) \subseteq L(\mathcal{A})$. □

Observation. The equivalence between Watson-Crick automata and 2-head automata is proved in [14] regardless of the structure of the complementarity relation using a similar construction as above. In their case, the second head of the 2-head automaton "guesses" the complement of the character read from the input tape, and simulates the corresponding move from the Watson-Crick automaton. However, in our proof, the injectivity of the complementarity relation plays an important role. If the complementarity relation would not be injective, then for all positions i of the input word, several reading heads would have to guess exactly the same complement but at different time steps. However, by definition this constraint cannot be imposed. Hence, the injectivity of the complementarity relation is a necessary condition in Theorem 9.

It is known from [10] that k-head automata are equivalent with parallel finite automata systems with k components and communicating by states. Moreover, it is proved in [8] that the languages accepted by multihead nondeterministic pushdown automata satisfy the semilinearity property. Hence, parallel finite automata systems communicating by states accept only semilinear languages. Since any semilinear language over an one-letter alphabet is regular, we have the following result.

Corollary 10. *Every one-letter language accepted by a parallel communicating Watson-Crick automata system with injective complementarity relation is regular.*

Recently, it was proved in [4] that on one-letter alphabets, parallel communicating Watson-Crick automata system with non-injective complementarity relation accept also some non-regular languages, e.g. $L = \{a^{n^2} \mid n \geq 2\}$.

4 Closure properties

In this section we consider some closure properties of the family of languages accepted by parallel communicating Watson-Crick automata systems. From now on, V is the input alphabet and $\# \notin V$ is a special character not included in it; let $V' = V \cup \{\#\}$. We also extend the complementarity relation by adding $(\#, \#) \in \rho$.

Theorem 11. *Let $L_1, L_2 \subseteq V^*$ be two languages accepted by some parallel communicating Watson-Crick automata systems of degrees n_1 and n_2, respectively, using the same complementarity relation. Then the language $(L_1\#) \bigcap (L_2\#)$ is also accepted by a system of degree $n_1 + n_2$.*

Proof. Let $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$, where

$$\mathcal{A}_1 = (V, \rho, A_1, \ldots, A_{n_1}, K), \text{ with } A_i = (V, \rho, Q_i, q_i, F_i, \delta_i) \text{ and}$$

$$\mathcal{A}_2 = (V, \rho, A'_1, \ldots, A'_{n_2}, K'), \text{ with } A'_i = (V, \rho, Q'_i, q'_i, F'_i, \delta'_i).$$

We construct a new system of degree $n_1 + n_2$

$$\mathcal{A} = (V', \rho, \overline{A}_1, \ldots, \overline{A}_{n_1}, \overline{A}'_1, \ldots, \overline{A}'_{n_2}, K \cup K'), \text{ where}$$

- $\overline{A}_1 = (V', \rho, Q_1 \cup \{q_1^f, q_1^v\} \cup \{v_i^{i-1} \mid 2 \leq i \leq n_1\} \cup \{K_2, \ldots, K_{n_1}\}, q_1, \{q_1^f\}, \overline{\delta}_1)$,
- $\overline{A}_i = (V', \rho, Q_i \cup \{q_i^f, q_i^v\} \cup \{v_i^j \mid 1 \leq j \leq n_1 - 1\}, q_i, \{q_i^f\}, \overline{\delta}_i)$, $2 \leq i \leq n_1$
- $\overline{A}'_1 = (V', \rho, Q'_1 \cup \{q_1'^f, q_1'^v\} \cup \{v_i'^{i-1} \mid 2 \leq i \leq n_2\} \cup \{K'_2, \ldots, K'_{n_2}\}, q'_1, \{q_1'^f\}, \overline{\delta}'_1)$,
- $\overline{A}'_i = (V', \rho, Q'_i \cup \{q_i'^f, q_i'^v\} \cup \{v_i'^j \mid 1 \leq j \leq n_2 - 1\}, q'_i, \{q_i'^f\}, \overline{\delta}'_i)$, $2 \leq i \leq n_2$.

The components $\overline{A}_i$ and $\overline{A}'_i$ are obtained from A_i and A'_i, respectively, by adding some states and some new transition rules, as follows:

(i) for all $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$: $\overline{\delta}_i(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) = \delta_i(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix})$ and $\overline{\delta}'_j(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) = \delta'_j(q, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix})$,

(ii) for all $1 \leq i \leq n_1$: $\overline{\delta}_i(s, \begin{pmatrix} \# \\ \# \end{pmatrix}) = q_i^v$, for any $s \in F_i$, $\overline{\delta}_i(q_i^f, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_i^f$,

(iii) $\overline{\delta}_1(q_1^v, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = K_2$, $\overline{\delta}_1(v_i^{i-1}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = K_{i+1}$, for all $2 \leq i \leq n_1 - 1$, and $\overline{\delta}_1(v_{n_1}^{n_1-1}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_1^f$,

(iv) for all $2 \leq i \leq n_1$: $\overline{\delta}_i(q_i^v, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = v_i^1$, $\overline{\delta}_i(v_i^j, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = v_i^{j+1}$, for $1 \leq j \leq n_1 - 2$, and $\overline{\delta}_i(v_i^{n_1-1}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_i^f$,

(v) for all $1 \leq i \leq n_2$: $\overline{\delta}'_i(s, \begin{pmatrix} \# \\ \# \end{pmatrix}) = q_i'^v$, for any $s \in F'_i$, $\overline{\delta}'_i(q_i'^f, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}) = q_i'^f$,

(vi) $\overline{\delta}'_1(q'^v_1, \binom{\lambda}{\lambda}) = K'_2$, $\overline{\delta}'_1(v'^{i-1}_i, \binom{\lambda}{\lambda}) = K'_{i+1}$, for all $2 \leq i \leq n_2 - 1$, and $\overline{\delta}'_1(v'^{n_2-1}_{n_2}, \binom{\lambda}{\lambda}) = q'^f_1$,

(vii) for all $2 \leq i \leq n_2$: $\overline{\delta}'_i(q'^v_i, \binom{\lambda}{\lambda}) = v'^1_i$, $\overline{\delta}'_i(v'^j_i, \binom{\lambda}{\lambda}) = v'^{j+1}_i$, for $1 \leq j \leq n_2 - 2$, and $\overline{\delta}'_i(v'^{n_2-1}_i, \binom{\lambda}{\lambda}) = q'^f_i$.

The system works as follows. We first check in parallel if a word w is in both languages. In order to have $w \in L_1\#$, the first n_1 components have to reach final states and read $\binom{\#}{\#}$ at exactly the same time. We use transitions of type (i) until every component A_i reaches $\binom{\#}{\#}$ in a final state, at which moment it enters a special state q^v_i. All we have to do now is to verify that all first n_1 components entered the states q^v_i at the same time. This is done by using a verification procedure composed of transitions of type (iii) and (iv). Similarly, we use transitions of type (vi) and (vii) to impose the same condition for the last n_2 components. Then, each component enters the new final states q^f_i or respectively q'^f_i and waits for all the others to finish parsing the input. Hence, the accepted language is $(L_1\#) \bigcap (L_2\#)$. □

Using a similar technique, we also obtain the following result.

Theorem 12. *Let $L_1, L_2 \subseteq V^*$ be two languages accepted by some parallel communicating Watson-Crick automata systems of degrees n_1 and n_2, respectively, using the same complementarity relation. Then the language $L_1\#L_2\#$ is also accepted by a system of degree $n_1 + n_2$.*

Proof. We construct a new system $\mathcal{A}$ of degree $n_1 + n_2$ which works as follows. The first n_1 components recognize the language $L_1\#(V \cup \{\#\})^*$ by verifying that the first $\binom{\#}{\#}$ is read by all of them at exactly the same moment and then they enter a new final state in which they finish reading the input string. Similarly, the last n_2 components recognize the language $V^*\#L_2\#$.

Since a word is accepted by $\mathcal{A}$ if and only if all components reach final states and read all the input, the language accepted by $\mathcal{A}$ is $L_1\#L_2\#$. □

Theorem 13. *Let $L \subseteq V^*$ be a language accepted by some parallel communicating Watson-Crick automata system. Then the language $(L\#)^*$ is also accepted by a system of equal degree.*

Proof. Let $L = L(\mathcal{A})$ where $\mathcal{A} = (V, \rho, A_1, \ldots, A_n, K)$ is a system of degree n with each $A_i = (V, \rho, Q_i, q_i, F_i, \delta_i)$. Starting from $\mathcal{A}$ we construct a new system $\mathcal{A}' = (V \cup \{\#\}, \rho, A'_1, \ldots, A'_n, K)$ with $A'_i = (V \cup \{\#\}, \rho, Q'_i, q^0_i, \{q^0_i, q^f_i\}, \delta'_i)$ by

adding some new states and transitions as follows. In order to recognize also the empty word, we introduce in each component a new initial and final state q_i^0. We also introduce transitions $q_i^0 \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \rightarrow \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} q_i$, where q_i is the initial state of component i in system $\mathcal{A}$. Then, the system $\mathcal{A}'$ simulates $\mathcal{A}$ on each component until we reach $\begin{pmatrix} \# \\ \# \end{pmatrix}$.

Next, we use the verification procedure described in Theorem 11 to check that all components read $\begin{pmatrix} \# \\ \# \end{pmatrix}$ at exactly the same moment in which case they each enter a new final state q_i^f. Then, by introducing transitions of the form $q_i^f \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} \rightarrow \begin{pmatrix} \lambda \\ \lambda \end{pmatrix} q_i$ in each component, we assure that the system can loop, also preserving the synchronization of components.

Thus, the system recognizes the language $\{\lambda\} \cup (L\#)^+$. □

Next, we give a representation result for recursively enumerable languages using languages accepted by parallel communicating Watson-Crick automata systems. We start by recalling a known characterization of recursively enumerable languages, see [14].

Lemma 14. *For each recursively enumerable language $L \subseteq V^*$, there exist two λ-free morphisms h_1, h_2, a regular language R, and a projection pr_V such that $L = pr_V(h_1(EQ(h_1, h_2)) \cap R)$.*

The next lemma was also proved in [14].

Lemma 15. *If $h_1, h_2 : V^* \rightarrow V^*$ are two morphisms, then $h_1(EQ(h_1, h_2))$ can be recognized by a Watson-Crick finite automaton.*

Using the previous two results as well as the closure of parallel communicating Watson-Crick automata systems under intersection, we can prove the following characterization.

Theorem 16. *For each recursively enumerable language $L \subseteq V^*$, there exists a projection pr_V such that $L = pr_V(L(\mathcal{A}))$, where $\mathcal{A}$ is a parallel communicating Watson-Crick automata system of degree 2.*

Proof. Let L be a recursively enumerable language. From Lemma 14 and Lemma 15 we have that there exists a projection pr_V such that $L = pr_V(L' \cap R)$, where L' is recognized by a Watson-Crick finite automaton and R is a regular language. Moreover, for any given complementarity relation ρ we can easily construct a Watson-Crick automaton $\mathcal{M}$ such that $L(\mathcal{M}) = R$.

From Theorem 11 we obtain that there exists a parallel communicating Watson-Crick automata system $\mathcal{A}$ of degree 2 such that $L(\mathcal{A}) = (L'\#) \cap (R\#)$. Since $\# \notin V$, we can extend the projection pr_V by setting $pr_V(\#) = \lambda$ and obtain $L = pr_V(L(\mathcal{A}))$. □

5 Conclusions

In this paper we introduced and investigated parallel communicating Watson-Crick automata systems. We prove that their accepting power is increased compared to Watson-Crick finite automata. However, every language accepted by a Watson-Crick finite automata system is context-sensitive. Moreover, one-letter languages accepted by such systems but with an injective complementarity relation prove to be regular. We also investigate some closure properties for these systems and give a representation of recursively enumerable languages.

Many questions and problems have remained open. For example it would be interesting to investigate other closure properties, e.g. under union or complementation. Also, it remains open what is the accepting power of systems using non-injective complementarity relations, when we restrict only to one-letter alphabets.

6 Acknowledgement

The authors are grateful to Prof. Arto Salomaa for valuable discussions. Also, the authors are thankful to the anonymous reviewers for their useful comments.

References

[1] S. Balan, *Distributed Processing in Automata*, Master Thesis, Department of Computer Science and Engineering, Indian Institute of Technology, (2000).

[2] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana, G. Vaszil, *Parallel Communicating Pushdown Automata Systems*, Int. J. Found. Comput. Sci. 11(4), 633-650, (2000).

[3] E. Csuhaj-Varjú, V. Mitrana, G. Vaszil, *Distributed Pushdown Automata Systems: Computational Power*, Proc. DLT 2003, LNCS, 2710, 218-229, (2003).

[4] E. Czeizler, E. Czeizler, *On the Power of Parallel Communicating Watson-Crick Automata Systems*, Theoretical Computer Science, 358: 1, 142-147, (2006). Also as TUCS Techreport 722, (2005).

[5] R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, *Watson-Crick finite automata*, Proc 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, 297-328, (1997).

[6] M. A. Harrison, *Introduction to formal language theory*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1978.

[7] J. E. Hopcroft, J. D. Ullman, *Introduction to automata theory, languages, and computation.*, Addison-Wesley, (1979).

[8] O. H. Ibarra, *A note on semilinear sets and bounded-reversal multihead pushdown automata*, Information Processing Letters, 3, 25-28, (1974).

[9] C. Martín-Vide, V. Mitrana, *Parallel communicating automata systems. A Survey*, Korean Journ. of Comp. and Appl. Math 7: 2, 237-257, (2000).

[10] C. Martín-Vide, A. Mateescu, V. Mitrana, *Parallel finite automata systems communicating by states*, Intern. Journ. of Found. of Comp. Sci. 13: 5, 733-749, (2002).

[11] C. Martín-Vide, Gh. Păun, *Normal forms for Watson-Crick finite automata*, in F. Cavoto, ed., The Complete Linguist: A Collection of Papers in Honour of Alexis Manaster Ramer: 281-296. Lincom Europa, Munich., (2000).

[12] V. Mihalache, A. Salomaa, *Lindenmayer and DNA: Watson-Crick DOL systems*, Current Trends in Theoretical Computer Science, World Sci., 740-751, (2001).

[13] A Păun, M. Păun, *State and transition complexity of Watson-Crick finite automata*, Proc. Fundamentals of Computation Theory, FCT'99, LNCS **1684**, Springer-Verlag, 409-420, (1999).

[14] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, (1998).

[15] E. Petre, *Watson-Crick ω-Automata*, J. Autom. Lang. Comb. 8(1), 59-70, (2003).

[16] G. Rozenberg, A. Salomaa (eds.) *The Handbook of Formal Languages*, Springer-Verlag, (1997).

[17] J. M. Sempere, *A Representation Theorem for Languages accepted by Watson-Crick Finite Automata*, Bulletin of the EATCS 83, 187-191, (2004).

[18] A. C. Yao, R. L. Rivest, *$k+1$ heads are better than k*, Journal of the Associaton for Computing Machinery, 25:2, 337-340, (1978).

Acta Cybernetica **17** (2006) 701–717.

A Fast Algorithm for the Constrained Multiple Sequence Alignment Problem*

Dan He† Abdullah N. Arslan† and Alan C. H. Ling†

Abstract

Given n strings $S_1, S_2, ..., S_n$, and a pattern string P, the *constrained multiple sequence alignment* ($CMSA$) problem is to find an optimal multiple alignment of $S_1, S_2, \ldots, S_n$ such that the alignment contains P, i.e. in the alignment matrix there exists a sequence of columns each entirely composed of symbol $P[k]$ for every k, where $P[k]$ is the kth symbol in P, $1 \leq k \leq |P|$, and in the sequence, a column containing $P[i]$ appears before the column containing $P[j]$ for all i, j, $i < j$. The problem is motivated from the problem of comparing multiple sequences that share a common structure, or sequence pattern. There are $O(2^n s_1 s_2 ... s_n r)$-time dynamic programming algorithms for the problem, where $s_1, s_2, ..., s_n$ and r are, respectively, the lengths of the input strings and the pattern string. Feasibility of these algorithms in practice is limited when the number of sequences is large, or the sequences are long because of the impractically long time required by these algorithms. We present a new algorithm with worst-case time complexity also $O(2^n s_1 s_2 ... s_n r)$, but the algorithm avoids redundant computations in existing dynamic programming solutions. Experiments on both randomly generated strings and real data show that this algorithm is much faster than the existing algorithms. We present an analysis that explains the speed-up obtained in our experiments by our algorithm over the naive dynamic programming algorithm for constrained multiple sequence alignment of protein sequences. The speed-up is more significant when pattern is long, or n is large. For example in the case of constrained pairwise sequence alignment (the $CMSA$ problem with $n = 2$) when the pattern is sufficiently long for strings S_1 and S_2, the asymptotic time complexity is observed to be $O(s_1 s_2)$ instead of $O(s_1 s_2 r)$. Main ideas in our algorithm can also be used in other constrained sequence alignment problems.

Keywords: constrained sequence alignment, pairwise alignment, multiple alignment, dynamic programming

*This work was supported in part by NSF Award No. CCF-0514819.

†Department of Computer Science, University of Vermont, Burlington, VT 05405, USA, E-mail: {dhe,aarslan,aling}@cs.uvm.edu

1 Introduction

Multiple sequence alignment [2] is one of the most important problems in computational biology. Detecting similarities in DNA sequences gives clues about the evolutionary relatedness of different species, and similarities in protein sequences point out similar functionality. The multiple sequence alignment problem can be defined in various ways depending on the objective function used for measuring the similarity. When *sum of pairs (SP)* scoring is used, the problem is defined as follows: Given a set of $n \geq 2$ sequences $S_1, S_2, ..., S_n$, insert gap symbols '$-$' into these sequences to obtain equal length strings, respectively, S_1^*, S_2^*, ..., S_n^* so that the global similarity score $\sum_{1 \leq i < j \leq n} score(S_i^*, S_j^*)$ is optimized where $score(S_i^*, S_j^*)$ is the similarity between S_i^* and S_j^* computed under a given scoring scheme. When $n = 2$, namely the sequence set has only two sequences S_1, S_2, the problem is the classical pairwise sequence alignment problem for which there is an $O(s_1 s_2)$-time dynamic programming algorithm [11]. This dynamic programming solution is extended to multiple sequence alignment problem with the resulting time complexity $O(2^n s_1 s_2 ... s_n)$. However, there are many heuristic algorithms to approximate the optimal solution (e.g. Clustal W [8], T-Coffee [5]). Recent progress in multiple sequence alignment is summarized in [6].

Given strings $S_1, S_2, ..., S_n$, and pattern string P, the *constrained multiple sequence alignment* ($CMSA$) problem is to find an optimal multiple alignment of $S_1, S_2, ..., S_n$ such that the alignment contains P, i.e. in the alignment matrix there exists a sequence of columns each entirely composed of symbol $P[k]$ for every k, where $P[k]$ is the kth symbol in P, $1 \leq k \leq |P|$, and in the sequence, a column containing $P[i]$ appears before column containing $P[j]$ for all i, j, $i < j$. A motivation for the problem is the alignment of RNase sequences. Such sequences are all known to contain three active residues `His(H)`, `Lyn(K)`, `His(H)` that are essential for RNA degrading. Therefore, it is natural to expect that in an alignment of RNA sequences, each of these residues should be aligned in the same column. The $CMSA$ problem when $n = 2$ is called the *constrained pairwise sequence alignment (CPSA) problem.*

For example, for $S_1 =$ `bbaba`, $S_2 =$ `abbaa`, and $P =$ `ab`, an optimal alignment that maximizes the number of matches with the constraint is shown in Figure 1.

Solutions for $CPSA$ can also be used to solve the $CMSA$ problem. One idea is to progressively align the sequences into a multiple alignment by using a mini-

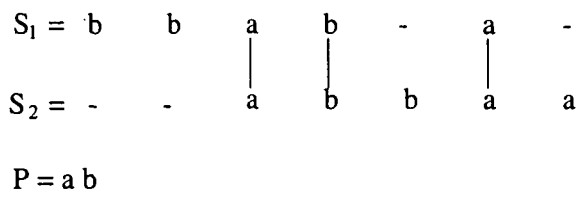


Figure 1: For $S_1 =$ `bbaba`, $S_2 =$ `abbaa`, and $P =$ `ab`, an optimal alignment which maximizes the number of matches with the constraint.

mum spanning tree obtained from a pairwise distance matrix of the sequences [7, 3]. There are many dynamic programming algorithms for the $CMSA$ and $CPSA$ problems, and their variations [7, 3, 9, 10, 1, 4]. The best known time complexity for the $CMSA$ problem is $O(2^n s_1 s_2 \dots s_n r)$ (see for example Chin et al. [3], or Tsai et al. [10]).

In this paper, we present a new dynamic programming algorithm for $CMSA$ based on the dynamic programming formulation given by Chin et al. [3], and the observation that we can use the pattern string P to avoid redundant computations in the dynamic programming matrix.

We have implemented our algorithm, and performed tests on both randomly generated data and real protein sequences. Experiments show that our algorithm is much more efficient in both time and space than a naive implementation of the algorithm presented by Chin et al. [3]. For the $CPSA$ problem the time requirement of our algorithm we observe in experiments is $O(s_1 s_2)$ when the pattern length r is large for given strings S_1 and S_2. For the $CMSA$ problem when $n > 2$, efficiency with respect to the naive algorithm we achieve with our algorithm increases significantly as the pattern length of P, or the number n of the set of sequences, $S_1, S_2, ..., S_n$ increases. The speed-up we obtain by our algorithm over the original naive dynamic programming algorithm proposed in [3] for the case of real protein sequences indicates that our algorithm is more feasible for solving the constrained multiple sequence alignment problem in practice.

The outline of this paper is as follows: in Section 2 we present our algorithm for the $CMSA$ problem. We summarize the results of our experiments in Section 3, and present mathematical analysis in Section 4 to explain the speed-up we observe in these tests using our algorithm. We include our final remarks in Section 5.

2 An Algorithm for the Constrained Multiple Sequence Alignment Problem

Our algorithm uses the dynamic programming formulation given by Chin et al. [3].

Let $D(i_1, i_2, ..., i_n, k)$ be the optimal constrained pairwise sequence alignment score of sequences $S_1[1..i_1]$,$S_2[1..i_2]$,...,$S_n[1..i_n]$ with constrained pattern sequence $P[1..r]$. Then this score can be computed by the following recurrence:

Theorem 1 ([3]). *For all* k, $1 \le k \le r$, $D(i_1, \dots, i_n, k) = \infty$ *if* $i_1 = 0$ *or* $i_2 = 0$ *or* ... *or* $i_n = 0$. $D(\{0\}^n, 0) = 0$. *For all* $i_1, i_2, \dots, i_n, k$, $0 < i_1 \le s_1, 0 < i_2 \le s_2, \dots, 0 < i_n \le s_n$, $0 \le k \le r$,

$$D(i_1, i_2, ..., i_n, k) = min \begin{cases} D(i_1 - 1, i_2 - 1, ..., i_n - 1, k - 1) \\ \quad +\delta(S_1[i_1], S_2[i_2], ..., S_n[i_n]) \\ \quad if\ (S_1[i_1] = S_2[i_2] = ... = S_n[i_n] = P[k])\ and\ k \ge 1 \\ \\ min_{e \in \{0,1\}^n} D(i_1 - e_1, i_2 - e_2, ..., i_n - e_n, k) \\ \quad +\delta(e_1 * S_1[i_1], e_2 * S_2[i_2], ..., e_n * S_n[i_n]) \end{cases}$$

where $e_j = 0$ *or* 1, $e_j * S_j[i_j]$ *with* $e_j = 0$ *represents a space character '−', and* $S_j[i_j]$ *when* $e_j = 1$, *and* $\delta(x_1, x_2, ..., x_k) = \sum_{1 \le i < j \le n} \delta(x_i, x_j)$ *(when sum-of-pairs distance is used) where* $\delta(x_i, x_j)$ *is the given minimum distance between the symbols* x_i *and* x_j.

A naive $CMSA$ algorithm for the dynamic programming solution in Theorem 1 is shown in Algorithm 1. The algorithm returns the optimal $CMSA$ score, $D(s_1, s_2, ..., s_n, r)$, in time $O(2^n s_1 s_2 \dots s_n r)$ where $s_1, s_2, ..., s_n, r$ are the lengths of the sequences S_1, S_2, ..., S_n, and P, respectively. The reason for factor 2^n is that computing $D(i_1, i_2, ..., i_n, k)$ uses $\Theta(2^n)$ neighboring entries of $(i_1, i_2, ..., i_n, k)$ in the dynamic programming matrix. When $n = 2$, the solution in Theorem 1 is a solution for the $CPSA$ problem.

Algorithm 1 The dynamic programming algorithm for the $CMSA$ problem proposed by Chin et al. [3].

```
Algorithm NaiveCMSA

1. Initialize D(0,0,...,0) = 0, D(i1,i2,...,in,k) = ∞, for all
   i1 * i2 * ... * in = 0, 0 ≤ i1 ≤ s1, 0 ≤ i2 ≤ s2, ..., 0 ≤ in ≤ sn, 1 ≤ k ≤ r
2. for k = 0 to r do
     for i1 = 0 to s1 do
       for i2 = 0 to s2 do
             ⋮
         for in = 0 to sn do
       If D(i1,i2,...,in,k) is not initialized, compute D(i1,i2,...,in,k)
       according to Theorem 1
3. return D(s1,s2,...,sn,k)
```

This algorithm computes the complete dynamic programming matrix parts of which are redundant in many cases. We observe that in an alignment matrix for S_1, S_2, ..., S_n, each $P[k]$ in P is required to appear in an entire column (we call such a column a *constraint-column* for $P[k]$) for the constraint to be satisfied. If $S_i[j_i]$ is aligned to $P[k]$ for the satisfaction of the constraint (i.e. if $S_i[j_i]$ appears in a constraint-column for $P[k]$ together with $S_1[j_1], S_2[j_2], \dots, S_{i-1}[j_{i-1}]$, $S_{i+1}[j_{i+1}], \dots, S_n[j_n]$) then $S_i[1..(j_i - 1)]$ can never be aligned with $S_p[(j_p + 1)..s_p]$ for all p, $1 \le p \le n$ and $p \ne i$. This means that we can save time by avoiding calculations in redundant regions in the dynamic programming matrix.

Our algorithm is based on the same dynamic programming formulation for computing $D(i_1, i_2, ..., i_n, k)$ given in Theorem 1. It is shown in Algorithm 2.

We first analyze Algorithm $FastCMSA$ for $CPSA$ computations. The analysis, and the results can be generalized for $CMSA$ computations which involve more than two sequences (i.e. $n > 2$). The dynamic programming algorithm here can be seen as computing $r + 1$ layers, one layer at each iteration k, where each layer is an n dimensional dynamic programming matrix. Figure 2 illustrates layers during the computations of $CPSA$ for a pattern whose length is 2.

Algorithm 2 Our algorithm for the $CMSA$ problem.

Algorithm *FastCMSA*

1. Initialize $D(0,0,...,0)=0, D(i_1,i_2,...,i_n,k)=\infty$, for all $i_1 * i_2 * i_3 * ... * i_n = 0,\ 0 \le i_1 \le s_1, 0 \le i_2 \le s_2, ..., 0 \le i_n \le s_n, 1 \le k \le r$
2. For each k, find every pair of first and last possible positions that match $P[k]$ in each string $S_1, S_2, \ldots, S_n$ in a constrained alignment:
   for $t = 1$ to n do
    for $k = 0$ to $r-1$ do
     set $S_{first}[t][k]$ = the first position f in S_t
       such that $P[1..(k+1)]$ is a subsequence of $S_t[1..f]$
     set $S_{last}[t][k]$ = the last position l in S_t
       such that $P[(k+1)..r]$ is a subsequence of $S_t[l..s_t]$
3. For each k, find a pair of start point and end point:
   $(S_{1begin}[k], S_{1last}[k]), (S_{2begin}[k], S_{2last}[k]), ..., (S_{nbegin}[k], S_{nlast}[k])$
   for k=0 to r do
    if($k == 0$){
     $S_{1begin}[0] = 0;$
     $S_{2begin}[0] = 0;$
        $\vdots$
     $S_{nbegin}[0] = 0;$
    } else {
     $S_{1begin}[k] = S_{first}[1][k-1] + 1;$
     $S_{2begin}[k] = S_{first}[2][k-1] + 1;$
        $\vdots$
     $S_{nbegin}[k] = S_{first}[n][k-1] + 1;$
    }
    if ($k == r$){
     $S_{1last}[k] = s_1;$
     $S_{2last}[k] = s_2;$
        $\vdots$
     $S_{nlast}[k] = s_k;$
    }else{
     $S_{1last}[k] = S_{last}[1][k] + 1;$
     $S_{2last}[k] = S_{last}[2][k] + 1;$
        $\vdots$
     $S_{nlast}[k] = S_{last}[n][k] + 1;$
    }
4. for $k = 0$ to r do
   for $i_1 = S_{1begin}[k]$ to $S_{1last}[k]$
    for $i_2 = S_{2begin}[k]$ to $S_{2last}[k]$
        $\vdots$
     for $i_n = S_{nbegin}[k]$ to $S_{nlast}[k]$
       compute $D(i_1, i_2, ..., i_n, k)$ using the expression in Theorem 1
5. return $D(s_1, s_2, ..., s_n, r)$

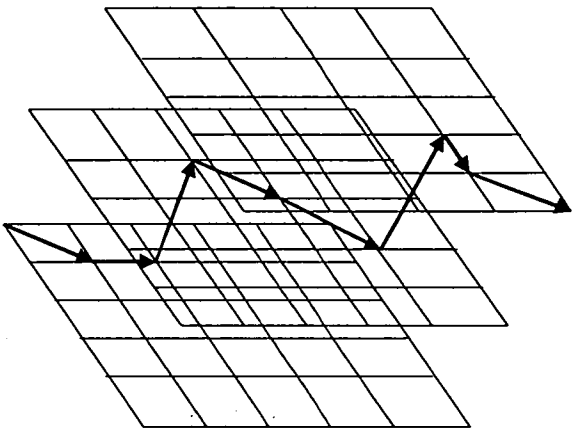


Figure 2: $CPSA$ computation for pattern string of length 2.

In the naive solution in Algorithm 1, at every iteration k (staring at $k = 0$) the whole layer is computed. On the other hand in Algorithm $FastCMSA$, when we process Layer k we compute only the subregion of the n dimensional dynamic programming matrix whose two diagonal corners, respectively, are $(S_{1begin}[k], S_{2begin}[k], ...; S_{nbegin}[k])$, $(S_{1last}[k], S_{2last}[k], ..., S_{nlast}[k])$. This is based on our observation that the area outside this region is not needed in later iterations because an optimal constrained alignment path does not pass there. For illustrative purposes, we only give an example for $CPSA$ computations in Figure 3. We only show the first two layers, and the last layer in the figure. Layers for $CMSA$ when $n > 2$ are similar, but have more dimensions. In Layer 0 we only need to compute the region whose two diagonal corners are $((S_{1begin}[0], S_{2begin}[0]), (S_{1last}[0], S_{2last}[0]))$. This is the only region required in the computations in the next layer, Layer 1. Similarly, at Layer 1, we only need to consider the region identified by two diagonal corners $((S_{1begin}[1], S_{2begin}[1])$, and $(S_{1last}[1], S_{2last}[1]))$. Computations in our algorithm proceed layer by layer in this manner.

Compared to the naive algorithm, our algorithm performs fewer operations on average for the points in the computed region of the dynamic programming matrix. For simplicity, we show this in the pairwise alignment case in Figure 4. On layer 0, we need to compute the rectangular region identified by its two diagonal corners $(S_1begin[0], S_2begin[0])$, $(S_1last[0], S_2last[0])$. In this region, the number of operations per point is the same in both algorithms. The differences are on Layer 1 and higher. For Layer 1, we need to compute the rectangular region of $(S_1begin[1], S_2begin[1])$, $(S_1last[1], S_2last[1])$. In the rectangular region of $(S_1begin[1], S_2begin[1])$, $(S_1last[0], S_2last[0])$ (in Figure 4 the rectangular region shaded with backward diagonal lines) the number of operations per point considered is still the same in both algorithms, but for the region elsewhere on Layer 1 (non-rectangular region shaded with forward diagonal lines in the figure), we do not need to consider the entries from the previous layer, Layer 0 in this case, since

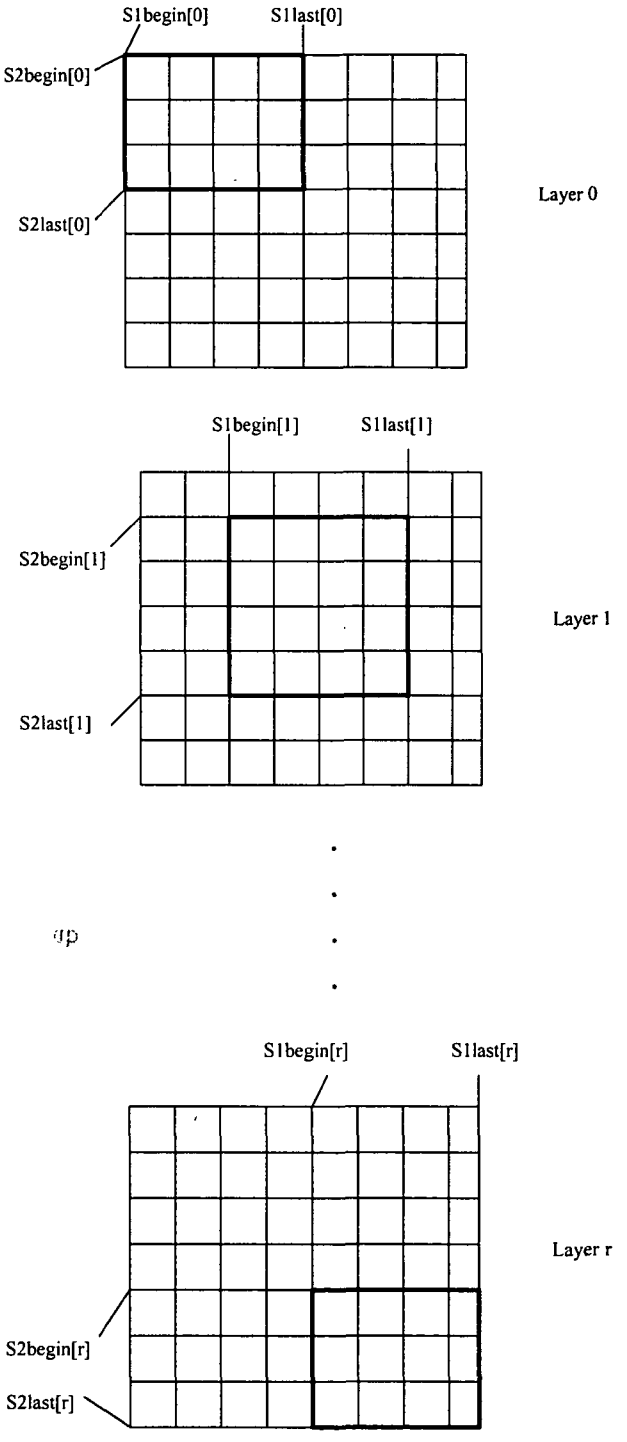


Figure 3: Regions in each layer considered in the computation of $CPSA$ with pattern string length $r > 2$.

on Layer 0, this region is not computed at all since there are no entries from last layer in this region.

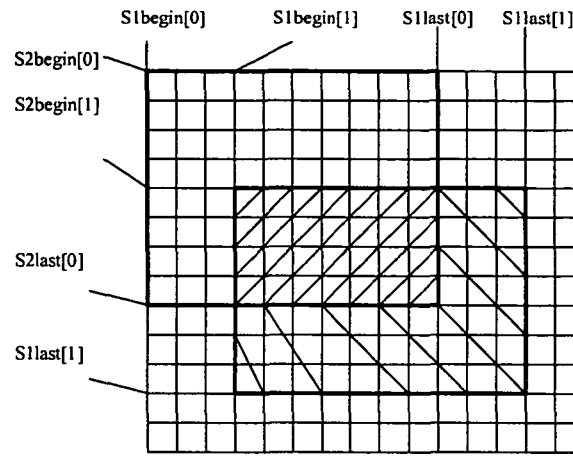


Figure 4: Illustration of the computation efficiency of our algorithm $FastCMSA$ over the naive dynamic programming algorithm.

Clearly the time complexity of our solution in Algorithm 2 is $O(s_1 s_2 r)$ for $CPSA$ computations. In our algorithm, for each layer, we only compute the region identified by $(S_{1begin}[k], S_{2begin}[k], ..., S_{nbegin}[k])$, $(S_{1last}[k], S_{2last}[k], ..., S_{nlast}[k])$. The larger the area, the longer our algorithm runs. We can create a worst case scenario as follows: For Layer 0, we try to move the last possible position which matches $P[1]$ as far as possible and the most backward position for S_i is $s_i - r$ since the length of the pattern string is r, there must be at least r symbols from this position. For the first layer, the area we need to compute is $\Omega((s_1 - r)(s_2 - r)...(s_n - r))$. For simplicity we only consider the pairwise sequence alignment case in Figure 5. For Layer 1, we try to move the first possible position which matches $P[1]$ to the beginning as much as possible, and move the last possible position which matches $P[2]$ to the end as much as possible. For similar reasons we discuss for the case of Layer 0, the smallest and largest positions, that determine the region we need to consider, in S_1, respectively, are 1 and $s_1 - r + 1$. Then we can see that the computations for Layer 1 takes $\Omega((s_1 - r)(s_2 - r))$ time. We can conclude that there is a case in which our algorithm requires $\Omega((s_1 - r)(s_2 - r)r)$ time for $CPSA$ computation. For $n > 2$ case, we can create a similar worst-case scenario for $S_1, S_2 \ldots S_n$, and P, and therefore, the worst-case computation time for $CMSA$ is $\Omega(2^n(s_1 - r)(s_2 - r)...(s_n - r)r)$. From the analysis of the worst-case scenarios, we can see that the longer the pattern string, or the higher the dimension, the better the speed-up we achieve relative to the naive $CMSA$ algorithm. We verify this by the results of our experiments.

Our discussions about the application of Algorithm $FastCMSA$ for the $CPSA$ computations can be extended to $CMSA$ computations that involve more than two

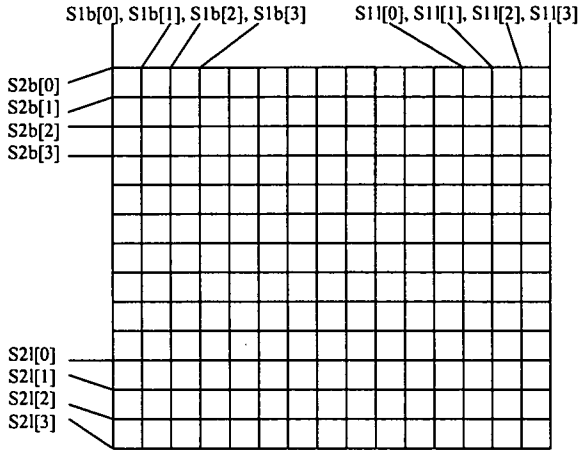


Figure 5: A worst case scenario for our algorithm $FastCMSA$ for a $CPSA$ computation with pattern string length 3. We use $S_{ib}[j]$ for $S_{ibegin}[j]$, and $S_{il}[j]$ for $S_{ilast}[j]$ to save space in the figure.

dimensions. Compared to the naive solution in Algorithm 1, our algorithm does computations for fewer points, and spends less time at each point.

3 Experiments

We first tested the performance of our algorithm $FastCMSA$ (which we call $FastCPSA$ when $n = 2$, i.e. when it is used for solving the $CPSA$ problem). We compare its performance with that of Algorithm $NaiveCMSA$ (which we call $NaiveCPSA$ when it is used for solving the $CPSA$ problem). In our tests, we randomly generate, over the alphabet of amino acids that contains 20 symbols, strings S_1 and S_2 with equal length, and pattern string P. We use 10 consecutive seeds to generate the sequences and the pattern each time, and show only the average performance. To measure time we count in the dynamic programming matrix the number of points for which the algorithms perform computations. Our algorithm is consistently faster than the naive solution in Algorithm 1. We note that when sequences S_1 and S_2 are fixed, the time requirement of our algorithm does not increase linearly with the increasing length of P. Figure 6 illustrates this. We plot pattern length *plength* versus time in the figure. In this test, we fix the sequence lengths *seqlength* as 1,000 and increase the pattern length *plength* from 4 to 35. The time requirement of the naive algorithm linearly increases with the pattern length, and for our algorithm, it increases at slower pace first, and it starts to decrease permanently after certain level of *plength*. This is because as the *plength* increases, the matching regions in the matrix on average is confined to smaller parts in the matrix and the volume computed by our algorithm is expected to be smaller

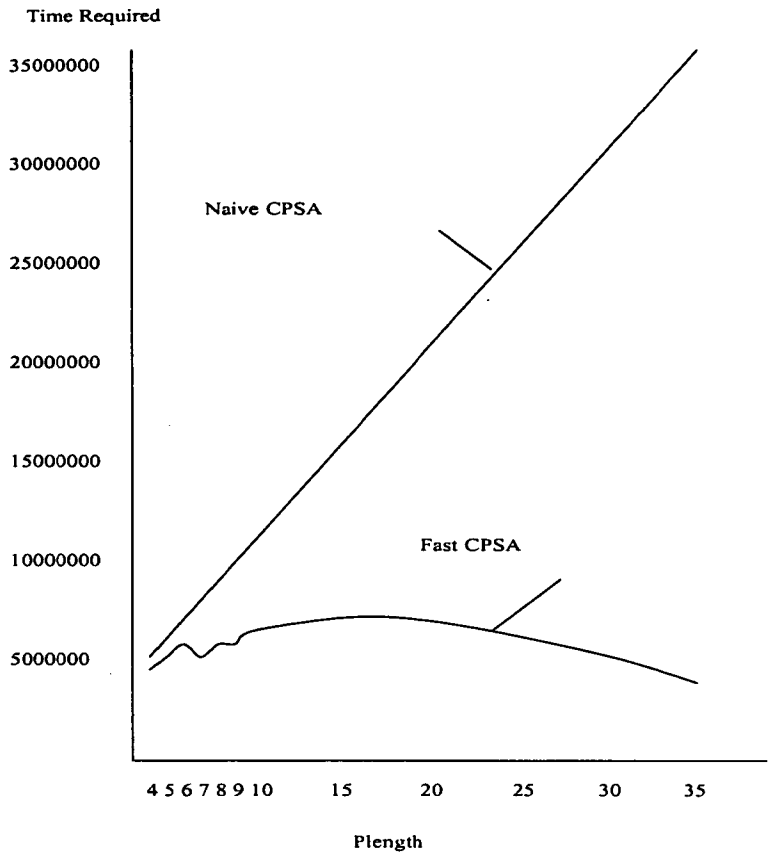


Figure 6: Time requirement of $CPSA$ computation when *seqlength* is fixed as 1,000, and *plength* is increased from 4 to 35. For each pattern length we use 10 consecutive seeds to generate the sequences and the pattern, and show only the average performance.

in ratio on average to the size of the entire matrix. We will discuss this in more detail in Section 4.

We next tested the performance of Algorithm $FastCMSA$ on randomly generated sets of 4 protein sequences with equal length, and pattern string with length 1, 2, 3, 4 separately, over alphabet of 20 amino acid symbols. For each pattern length we use 10 consecutive seeds to generate the sequences and the pattern, and show only the average performance.

We compare the number of points in the dynamic programming matrix Algorithm $FastCMSA$ needs to compute with the number of points the naive dynamic programming algorithm computes. Table 1 shows that our algorithm is consistently faster than the naive $CMSA$ algorithm, and the performance of our algorithm over

Table 1: Average number of points the two algorithms need to compute for the alignment of 4 sequences when we fix *seqlength* as 100 and increase *plength* from 1 to 4 at increments of 1, and use 10 consecutive seeds to generate the sequences and the pattern for each pattern length, and show only the average performance.

plength	FastCMSA	NaiveCMSA	*Naive/Fast*
1	$8.09e+007$	$2.12e+008$	2.62
2	$6.30e+007$	$3.18e+008$	5.05
3	$4.77e+007$	$4.25e+008$	8.90
4	$2.10e+007$	$5.31e+008$	25.28

$seqlength = 100$

Table 2: Number of points both algorithms need to compute when we fix *seqlength* as 200, *plength* as 4 and increase the number of sequences from 3 to 6. For each case, we use 10 consecutive seeds to generate the sequences and the pattern, and show only the average performance.

dimension	FastCMSA	NaiveCMSA	*Naive/Fast*
3	$9.60e+006$	$4.06e+007$	4.22
4	$1.10e+008$	$8.16e+009$	7.42
5	$1.19e+011$	$1.64e+012$	13.78
6	$1.30e+013$	$3.30e+014$	25.38

$seqlength = 200,\ plength = 4$

the naive $CMSA$ algorithm increases quickly with the increasing pattern length. This is because the larger the *plength*, the less chances there are for the worst-case scenario. Therefore, for the same sequence set, the longer the pattern string is, the more significantly our algorithm outperforms the naive $CMSA$ algorithm.

In another set of tests, we fixed the sequence lengths *seqlength* as 200 and the pattern length *plength* as 4. Then we solved $CMSA$ problems for $n = 3, 4, 5, 6$. For each n, we also show the average performance of 10 tests by 10 consecutive seeds. We summarize the results in Table 2. We observe that the performance of Algorithm $FastCMSA$ over the naive $CMSA$ algorithm nearly doubles every time we add one more sequence (increase n by one). This is because with new sequences being involved in the alignment, a larger region in the original dynamic programming matrix is avoided.

Another advantage of our algorithm is that it first computes the possible pattern occurrence positions in each sequence, if there are no such positions then our algorithm stops immediately while $NaiveCPSA$ computes the entire dynamic programming matrix.

Table 3: Experiments on constrained alignment of 5 $RNase$ sequences with pattern string HKH and $HKSH$, separately.

pattern	FastCMSA	NaiveCMSA	$Naive/Fast$
HKH	$7.343e+009$	$2.737e+011$	37.3
$HKSH$	$5.053e+009$	$3.421e+011$	67.7

number of computation points

We have also done experiments on real protein sequences. We used the set of sequences with references given in [3](Data Set 1, and Data Set 2):

$Seq1 : gi|119124|sp|p12724|ecp\_human,$
$Seq2 : gi|2500564|sp|p70709|ecp\_rat,$
$Seq3 : gi|13400006|pdb|ldyt|,$
$Seq4 : gi|20930966|ref|xp\_142859.1,$
$Seq5 : gi|20930966|ref|xp\_142859.1$

The results of the experiments are shown in Table 3. Clearly, our algorithm is much faster than the naive $CMSA$ algorithm on RNase sequences.

4 Performance analysis of our algorithm

The performance of our algorithm depends on the total size of the layers from Layer 0 to Layer r.

We note that our algorithm does not perform computations for all the points considered by the naive algorithm implementing Theorem 1, and for the points it does it spends less time than the naive algorithm. Therefore, we compare the total volume (number of points) at which our algorithm performs computations with the total size of the $(n+1)$-dimensional dynamic programming matrix the naive algorithm uses.

Size of each layer in our algorithm is determined by the first and last matches of the given pattern P in each dimension (i.e. on each sequence). Let $b_{i,k}$ be the position of $P[k]$ in the first occurrence of $P[1..k]$ in S_i, and let $e_{i,k}$ be the position of $P[k]$ in the last occurrence of $P[k..r]$ in S_i.

We assume that pattern P occurs at least once in each sequence S_i. Otherwise, our algorithm does not do any computations in the dynamic programming matrix.

Throughout our analysis we also assume that each symbol in alphabet Σ over which sequences $S_1, S_2, \ldots, S_n$ are defined appears with equal probability in each position in these sequences.

Layer 0 is identified by two extreme points $(0, 0, \ldots, 0)$ and $(e_{1,r}, e_{2,r}, \ldots, e_{n,r})$, and its size is

$$\prod_{i=1}^{n} e_{i,1} \tag{1}$$

Each Layer k, $1 < k < r$, has two extreme points $(b_{1,k}, b_{2,k}, \ldots, b_{r,k})$ and $(e_{1,k+1}, b_{2,k+1}, \ldots, b_{r,k+1})$, and its size is

$$\sum_{k=1}^{k=r-1} \prod_{i=1}^{n} (e_{i,k+1} - b_{i,k}) \tag{2}$$

Two extreme points on Layer r are $(b_{1,1}, b_{2,1}, \ldots, b_{n,1})$ and $(s_1, s_2, \ldots, s_n)$, and the size of this layer is

$$\prod_{i=1}^{n} (s_i - b_{i,r}) \tag{3}$$

We study the expected sizes of these layers and their sum.

Lemma 2. *Suppose $P = a_1 a_2 \ldots a_r$ is a pattern of length r. Let S be a sequence of length s that contains P as a subsequence. Let Σ be the alphabet for P and S. The expected position of $P[r]$ in the first occurrence of $P[1..r]$ in S is $|\Sigma|r$.*

Proof. Let $\bar{a_i} = Q \setminus \{a_i\}$ be the set of alphabet except a_i. Then, all strings contain the first occurrence of P as a subsequence must have a unique representation of the form $A = \bar{a_1}^* a_1 \bar{a_2}^* \ldots \bar{a_r}^* a_r$. One can see this because when we scan the sequence from left to right, we first seek for a_1, then a_2, and so on until we find a_r eventually. We next compute a generating function $f(x)$ that counts the number of strings in A. Here, we mean $f(x) = \sum_{a \in A} x^{len(a)}$ where $len(a)$ denotes the length of a. Based on the decomposition of A, we can easily deduce that $f(x) = (\frac{x}{1-(|\Sigma|-1)x})^r$ [12]. In order to compute the expected length of such sequences, we need to determine $\sum_{i=0}^{\infty} i f_i$ where f_i is the coefficient of x^i in the function $f(x)$. It is evident that the expected length is equal to $xf'(x)|_{x=\frac{1}{|\Sigma|}}$. Simple calculus shows that the expected length of such strings is $|\Sigma|r$. We can also calculate the expected length when the sequence length is finite. This gives us the expected position of $P[r]$ in the first occurrence of P in S given that P occurs in S at least once. In this case, for a given sequence length s, the expected length is $\sum_{i=0}^{s} i f_i = \sum_{n=0}^{s} n \frac{\binom{n-1}{r-1}(|\Sigma|-1)^{n-r}}{|\Sigma|^n}$. We calculate expected lengths for $s = 10, \ldots, 200$ in increments of 10, and in Figure 7 we plot them versus sequence length s for varying pattern lengths $r = 1, \ldots, 5$, and for a fixed alphabet size $|\Sigma| = 20$. We see that they converge to $|\Sigma|r$ quickly (before the sequence length s approaches to 200). We note that length of a protein sequence used in constrained multiple sequence alignments is typically 150 [3, 7]. □

By using Lemma 2, and observing that the expected position of the last occurrence of pattern P is the same as the expected first occurrence of the pattern P^R where P^R means the reverse of the pattern, we can reach the following corollaries:

Corollary 3. *For a given pattern P of length r, and a string S of length s that contains P as a subsequence, the expected position of $P[1]$ in the last occurrence of $P[1..r]$ approaches quickly to $s - |\Sigma|r$ if S is sufficiently long for r and $|\Sigma|$ where Σ is the alphabet for S and P.*

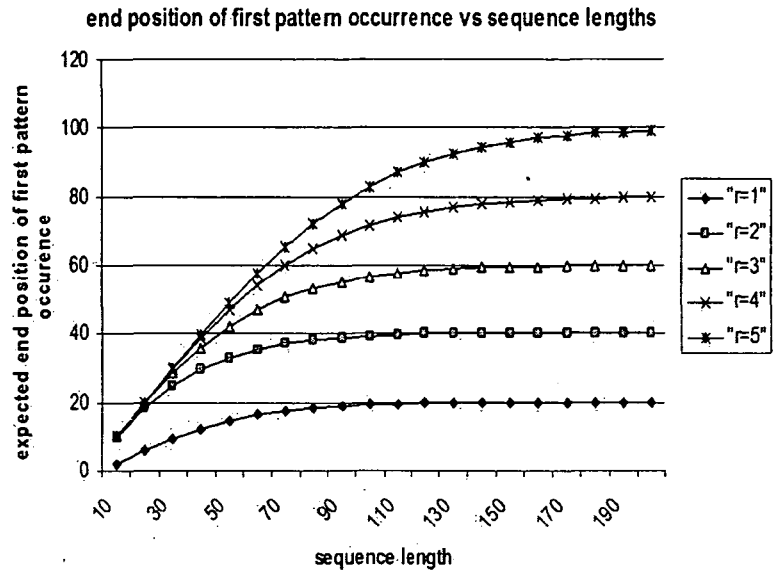


Figure 7: Expected position of $P[r]$ in the first occurrence of pattern $P[1..r]$ in string S that contains P as a subsequence versus the length s of S. Pattern length r varies from 1 to 5. The alphabet size is $|\Sigma| = 20$. The convergence is observed when s approaches to 200.

We use $x \sim V$ to denote that the value of x approaches to V.

Corollary 4. *For all i, $1 \leq i \leq n$, $E(b_{i,r}) = |\Sigma|r$, and if S_i is sufficiently long for r and $|\Sigma|$ then $E(e_{i,1}) \sim s_i - |\Sigma|r$.*

Corollary 5. *For a given pattern P of length r, and a string S of length s that contains P as a subsequence where P and S are defined over alphabet Σ, for all k, $1 < k < r$, let b_k be the position of $P[k]$ in the first occurrence of $P[1..k]$ in S, and let e_{k+1} be the position of $P[k+1]$ in the last occurrence of $P[(k+1)..r]$ in S. The expected position $E(b_k) = |\Sigma|k$, and if S is sufficiently long for r and $|\Sigma|$ then the expected position $E(e_{k+1}) \sim s - |\Sigma|(r-k)$, and therefore, the expected difference $E(e_{k+1} - b_k) = E(e_{k+1}) - E(b_k) \sim s - |\Sigma|r$.*

Corollary 6. *For all i, $1 \leq i \leq n$, and k, $1 \leq k \leq r$, if S_i is sufficiently long for r and $|\Sigma|$ then $E(e_{i,k+1} - b_{i,k}) \sim s_i - |\Sigma|r$.*

It is easy to see that $e_{i,1}$ for different S_i's are independent, and by the product rule of expectation for independent random variables, and using Equation (1) the expected size of Layer 0 is

$$E(\prod_{i=1}^{n} e_{i,1}) = \prod_{i=1}^{n} E(e_{i,1}) \tag{4}$$

If we consider $e_{i,k+1}$ and $b_{i,k}$ as random variables then $e_{i,k+1} - b_{i,k}$ are independent for different S_i's. We note that $e_{i,k+1} - b_{i,k}$ are not independent for different layer k's for the same S_i but the linearity of expectation does not require this property, and therefore, using Equation (2) the expected size of Layer k, for all $1 \leq k \leq r-1$, is

$$E(\prod_{i=1}^{n} e_{i,k+1} - b_{i,k}) = \prod_{i=1}^{n} E(e_{i,k+1} - b_{i,k}) \tag{5}$$

Since $(s_i - b_{i,r})$ are independent for different S_i's, and if we use Equation (3) we can see that the expected size of Layer r is

$$E(\prod_{i=1}^{n} (s_i - b_{i,r}) = \prod_{i=1}^{n} E(s_i - b_{i,r}) \tag{6}$$

Adding equations (4), (5), and (6), and using corollaries 4 and 6, if S_i is sufficiently long for r and $|\Sigma|$ for all i, $1 \leq i \leq n$, then the expected total volume of layers from 0 to r approaches to

$$(r+1) \prod_{i=1}^{n} (s_i - |\Sigma| r) \tag{7}$$

If we compare this volume with the total size $(r+1) \prod_{i=1}^{n} s_i$ of the dynamic programming matrix used by the naive algorithm we can see that the expected speed-up achieved by our algorithm over the naive algorithm approaches to

$$\prod_{i=1}^{n} \frac{s_i}{s_i - |\Sigma| r} .$$

Given a pattern of length r, and n sequences of lengths $s_1, s_2, \ldots, s_n$ over alphabet Σ where each S_i contains P as a subsequence, and S_i sufficiently long for r and $|\Sigma|$, and $s_i > |\Sigma| r$, let $C_i = \frac{s_i}{|\Sigma| r}$ for all i, $1 \leq i \leq n$, then we can see that the expected speed-up of our algorithm over the naive algorithm approaches to

$$\prod_{i=1}^{n} \frac{s_i}{s_i - |\Sigma| r} \geq \prod_{i=1}^{n} \frac{C_i}{C_i - 1} .$$

This expression for the speed-up explains the results we have shown in Figure 6, and tables 1, 2, and 3. The speed-up is more significant if $C_i = \frac{s_i}{|\Sigma| r} > 1$ is a small number close to 1. For example, for the $CPSA$ problem with fixed sequence lengths $s_1 = s_2 = 1000$ and with pattern length r increasing from 4 to 35, and alphabet size is 20, the speed-up accelerates with increasing r as shown in Figure 6.

The target application of this paper is the constrained multiple sequence alignment of protein sequences where the alphabet is composed of 20 amino acids, a typical protein sequence length is 150 [3, 7], and a pattern used as a constraint is typically $3-4$ character-long. In these cases all $C_i \leq 2.5$, and the expected speed-up $\sim (5/3)^n$ where n is the number of sequences compared.

5 Concluding Remarks

We present an algorithm for the constrained multiple sequence alignment problem based on the dynamic programming formulation given by Chin et al. [3]. We observe that it is redundant to compute the entire dynamic programming matrix because the alignments are constrained to include pattern string P. We can precompute a set of points that breaks the dynamic programming matrix into parts some of which are redundant for solving the problem. Although our algorithm does not improve the worst-case time-complexity of the problem, the experiments we have conducted on both syntectic data and real RNase sequences show that our algorithm is significantly faster than the original naive dynamic programming algorithm proposed by Chin et al. [3]. The speed-up we achieve is more significant when the pattern is long, and the number of sequences is large. We present mathematical analysis for the expected speed-up achieved by our algorithm. The speed-up is expected to be significant if the product of the alphabet size and the pattern length is a relatively large fraction of the sequences aligned. This is in general true in practice in constrained multiple sequence alignment of protein sequences [3, 7].

An interesting behavior of our algorithm is observed when it is applied to the constrained pairwise sequence alignment. In this case, our algorithm's observed asymptotic time complexity is quadratic instead of cubic when the pattern is sufficiently long for given sequences.

Our ideas on the $CMSA$ can also be used in the algorithms for the constrained longest common subsequence problems [1, 4], and similar speed-up can be achieved.

Other kinds of existing techniques for multiple sequence alignment, both heuristic and exact, can be combined with the main steps of our algorithm to increase the feasibility of the $CMSA$ problem in real-life applications.

References

[1] Arslan, A. N. and Eğecioğlu., Ö. Algorithms for the constrained longest common subsequence problems. *International Journal of Foundations of Computer Science*, (16)6:1099-1111, December 2005.

[2] Carrillo, H. and Lipman, D. J. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math*, 48(5):1073-1082, 1988.

[3] Chin, F. Y. L., Ho, N. L., Lam, T. W., Wong, P. W. H., and Chan, M. Y. Efficient constrained multiple sequence alignment with performance guarantee. *Proc. IEEE Computational Systems Bioinformatics (CSB 2003)*, pp. 337-346, 2003.

[4] Chin, F. Y. L., Santis, A. D., Ferrara, A. L., Ho, N. L., and Kim, S. K. A simple algorithm for the constrained sequence problems. *Information Processing Letters* Vol. 90, pp. 175-179, 2004.

[5] Notredame, C., Higgins, D. G., and Heringa, J. T-Coffee: A novel algorithm for multiple sequence alignment. *J.Mol.Biol.*, 302,205-217, 2000.

[6] Notredame, C. Recent progresses in multiple sequence alignment: a survey. Ashley Publications Ltd, ISSN 1462-2416, 2001.

[7] Tang, C. Y., Lu, C. L., Chang, M. D.-T., Tsai, Y.-T., Sun, Y.-J., Chao, K.-M., Chang, J.-M., Chiou, Y.-H., Wu, C.-M., Chang, H.-T., and Chou, W.-I. Constrained multiple sequence alignment tool development and its applications to RNase family alignment. *Proceeding of the 1st IEEE Computer Society Bioinformatics Conference (CSB 2002)*, pp. 127-137, 2002.

[8] Thompson, J., Higgins, D., and Gibson, T. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting position specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22,4673-4690, 1994.

[9] Tsai, Y.-T. The constrained common sequence problem. *Information Processing Letters*, 88:173–176, 2003.

[10] Tsai, Y.-T., Lu, C. L., Yu, C. T., and Huang, Y. P. MuSiC: A tool for multiple sequence alignment with constraint. *Bioinformatics*, 20(14):2309-2311, 2004.

[11] Waterman, M. S. Introduction to computational biology. *Chapman & Hall*, 1995.

[12] Wilf, H. S. Generating functionology. *Academic Press, Inc*, 1994.

Acta Cybernetica 17 (2006) 719–749.

Kleene Theorems for skew formal power series*

Werner Kuich†

Abstract

We investigate the theory of skew (formal) power series introduced by Droste, Kuske [5, 6], if the basic semiring is a Conway semiring. This yields Kleene Theorems for skew power series, whose supports contain finite and infinite words. We then develop a theory of convergence in semirings of skew power series based on the discrete convergence. As an application this yields a Kleene Theorem proved already by Droste, Kuske [5].

1 Introduction and preliminaries

The purpose of our paper is to investigate the skew formal power series introduced by Droste, Kuske [5, 6]. These skew formal power series are a clever generalization of the ordinary power series and are defined as follows.

Let A be a semiring and $\varphi : A \to A$ be an endomorphism of this semiring. Then Droste, Kuske [5] define the *φ-skew product* $r \odot_\varphi s$ of two power series $r, s \in A^{\Sigma^*}$, Σ an alphabet, by

$$(r \odot_\varphi s, w) = \sum_{uv=w} (r, u)\varphi^{|u|}(s, v)$$

for all $w \in \Sigma^*$. They denote the structure $(A^{\Sigma^*}, +, \odot_\varphi, 0, 1)$ by $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ and prove the following result.

Theorem 1 (Droste, Kuske [5]). *The structure $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is a semiring.*

They call $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ the *semiring of skew (formal) power series (over Σ^*)*.

In the sequel, we often denote $\odot_\varphi$ simply by $\cdot$ or concatenation and A, φ and Σ denote a semiring, an endomorphism $\varphi : A \to A$ and an alphabet, respectively.

The paper consists of this and four more sections. In this section we give a survey on the results achieved by this paper and then define the necessary algebraic structures: starsemirings, Conway semirings, semimodules, starsemiring-omegasemimodule pairs, Conway semiring-semimodule pairs, complete semiring-semimodule pairs and quemirings. These algebraic structures, due to Elgot [8], Bloom, Ésik [2] and Ésik, Kuich [9] give an algebraic basis for the theory of power

*Partially supported by Aktion Österreich-Ungarn, Wissenschafts- und Erziehungskooperation, Projekt 60öu12.

†Technische Universität Wien, E-mail: kuich@tuwien.ac.at

series, whose supports contain finite and infinite words. At the end of this section we refer to some examples for these algebraic structures.

In Section 2 we prove that the semiring of skew power series over a Conway semiring is again a Conway semiring. Moreover, we prove two isomorphisms of certain semirings defined in connection with Conway semirings.

In Section 3, the results of Section 2 are applied to finite automata. A Kleene Theorem over quemirings defined by skew power series over Conway semirings and the usual Kleene Theorem over Conway semirings are shown.

In Section 4, we consider a semiring-semimodule pair defined by skew power series and prove that under certain conditions this pair is complete. This gives rise to another Kleene Theorem that is then applied to a tropical semiring and yields a result already achieved by Droste, Kuske [5].

In the last section we develop a theory of convergence in semirings of skew power series based on the discrete convergence. We show that important equations, which hold in Conway semirings, are valid under certain conditions also in semirings of skew power series over an arbitrary semiring. As an application this yields then another Kleene Theorem proved already by Droste, Kuske [5].

We assume that the reader of this paper is familiar with the theory of semirings as given in Sections 1–4 of Kuich, Salomaa [14]. Familiarity with Ésik, Kuich [9, 10, 11] is desired.

Recall that a *starsemiring* is a semiring A equipped with a star operation $^* : A \to A$. The *Conway identities* are the *sum-star equation* and the *product-star equation*

$$\begin{aligned} (a+b)^* &= (a^*b)^*a^* \\ (ab)^* &= 1 + a(ba)^*b. \end{aligned}$$

A *Conway semiring* is a starsemiring satisfying the Conway equations. Note that any Conway semiring satisfies the *star fixed point equations*

$$\begin{aligned} aa^* + 1 &= a^* \\ a^*a + 1 &= a^*, \end{aligned}$$

as well as the equations

$$\begin{aligned} a(ba)^* &= (ab)^*a \\ (a+b)^* &= a^*(ba^*)^*. \end{aligned}$$

Suppose that A is a semiring and V is a commutative monoid written additively. We call V a (left) *A-semimodule* if V is equipped with a (left) action

$$\begin{aligned} A \times V &\to V \\ (s, v) &\mapsto sv \end{aligned}$$

subject to the following rules:

$$\begin{aligned} s(s'v) &= (ss')v \\ (s+s')v &= sv + s'v \\ s(v+v') &= sv + sv' \\ 1v &= v \\ 0v &= 0 \\ s0 &= 0, \end{aligned}$$

for all $s, s' \in A$ and $v, v' \in V$. When V is an A-semimodule, we call (A, V) a *semiring-semimodule pair.*

Suppose that (A, V) is a semiring-semimodule pair such that A is a starsemiring and A and V are equipped with an omega operation $^\omega : A \to V$. Then we call (A, V) a *starsemiring-omegasemimodule pair.* Following Bloom, Ésik [2], we call a starsemiring-omegasemimodule pair (A, V) a *Conway semiring-semimodule pair* if A is a Conway semiring and if the omega operation satisfies the *sum-omega equation* and the *product-omega equation:*

$$\begin{aligned} (a+b)^\omega &= (a^*b)^\omega + (a^*b)^*a^\omega \\ (ab)^\omega &= a(ba)^\omega, \end{aligned}$$

for all $a, b \in A$. It then follows that the *omega fixed-point equation* holds, i.e.,

$$aa^\omega = a^\omega,$$

for all $a \in A$.

Recall that a *complete monoid* is a commutative monoid $(M, +, 0)$ equipped with all sums $\sum_{i \in I} m_i$ such that

$$\begin{aligned} \sum_{i \in \emptyset} &= 0 \\ \sum_{j \in \{1\}} m &= m \\ \sum_{i \in \{1,2\}} m_i &= m_1 + m_2 \\ \sum_{j \in J} \sum_{i \in I_j} m_i &= \sum_{i \in \cup_{j \in J} I_j} m_i, \end{aligned}$$

where in the last equation it is assumed that the sets I_j are pairwise disjoint. A *complete semiring* is a semiring A which is also a complete monoid satisfying the distributive laws

$$\begin{aligned} s(\sum_{i \in I} s_i) &= \sum_{i \in I} ss_i \\ (\sum_{i \in I} s_i)s &= \sum_{i \in I} s_i s, \end{aligned}$$

for all $s \in A$ and for all families s_i, $i \in I$ over A. Ésik, Kuich [9] define a *complete semiring-semimodule pair* to be a semiring-semimodule pair (A, V) such that A is a complete semiring, V is a complete monoid and an *infinite product operation*

$$(s_1, s_2, \ldots) \mapsto \prod_{j \geq 1} s_j$$

is given mapping infinite sequences over A to V with

$$\begin{aligned} s(\sum_{i \in I} v_i) &= \sum_{i \in I} s v_i \\ (\sum_{i \in I} s_i) v &= \sum_{i \in I} s_i v, \end{aligned}$$

for all $s \in A$, $v \in V$, and for all families s_i, $i \in I$ over A and v_i, $i \in I$ over V and with the following three conditions:

$$\begin{aligned} \prod_{i \geq 1} s_i &= \prod_{i \geq 1} (s_{n_{i-1}+1} \cdot \cdots \cdot s_{n_i}) \\ s_1 \cdot \prod_{i \geq 1} s_{i+1} &= \prod_{i \geq 1} s_i \\ \prod_{j \geq 1} \sum_{i_j \in I_j} s_{i_j} &= \sum_{(i_1, i_2, \ldots) \in I_1 \times I_2 \times \ldots} \prod_{j \geq 1} s_{i_j}, \end{aligned}$$

where in the first equation $0 = n_0 \leq n_1 \leq n_2 \leq \ldots$ and $I_1, I_2, \ldots$ are arbitrary index sets. Suppose that (A, V) is complete. Then we define

$$\begin{aligned} s^* &= \sum_{i \geq 0} s^i \\ s^\omega &= \prod_{i \geq 1} s, \end{aligned}$$

for all $s \in A$. This turns (A, V) into a starsemiring-omegasemimodule pair. By Ésik, Kuich [9], each complete semiring-semimodule pair is a Conway semiring-semimodule pair. Observe that, if (A, V) is a complete semiring-semimodule pair, then $0^\omega = 0$.

A *star-omega semiring* is a semiring A equipped with unary operations * and $^\omega : A \to A$. A star-omega semiring A is called *complete* if (A, A) is a complete semiring-semimodule pair, i.e., if A is complete and is equipped with an infinite product operation that satisfies the three conditions stated above.

Consider a starsemiring-omegasemimodule pair (A, V). Then, following Conway [4], we define, for all $n \geq 0$, the operation $^* : A^{n \times n} \to A^{n \times n}$ by the following inductive definition. When $n = 0$, M^* is the unique 0×0-matrix, and when $n = 1$, so that $M = (a)$, for some a in A, $M^* = (a^*)$. Assuming that $n > 1$, let us write M as

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{1}$$

where a is 1×1 and d is $(n-1) \times (n-1)$. We define

$$M^* = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}, \tag{2}$$

where $\alpha = (a + bd^*c)^*$, $\beta = a^*b\delta$, $\gamma = d^*c\alpha$, $\delta = (d + ca^*b)^*$.

Following Bloom, Ésik [2], we define a matrix operation $^\omega : A^{n\times n} \to V^{n\times 1}$ on a starsemiring-omegasemimodule pair (A, V) as follows. When $n = 0$, M^ω is the unique element of V^0, and when $n = 1$, so that $M = (a)$, for some $a \in A$, $M^\omega = (a^\omega)$. Assume now that $n > 1$ and write M as in (1). Then

$$M^\omega = \begin{pmatrix} (a + bd^*c)^\omega + (a + bd^*c)^*bd^\omega \\ (d + ca^*b)^\omega + (d + ca^*b)^*ca^\omega \end{pmatrix}. \tag{3}$$

Following Ésik, Kuich [11], we define matrix operations $^{\omega_k} : A^{n\times n} \to V^{n\times 1}$, $0 \le k \le n$, as follows. Assume that $M \in A^{n\times n}$ is decomposed into blocks a, b, c, d as in (1), but with a of dimension $k \times k$ and d of dimension $(n-k) \times (n-k)$. Then

$$M^{\omega_k} = \begin{pmatrix} (a + bd^*c)^\omega \\ d^*c(a + bd^*c)^\omega \end{pmatrix} \tag{4}$$

Observe that $M^{\omega_0} = 0$ and $M^{\omega_n} = M^\omega$.

Suppose that (A, V) is a semiring-semimodule pair and consider $T = A \times V$. Define on T the operations

$$\begin{aligned} (s, u) \cdot (s', v) &= (ss', u + sv) \\ (s, u) + (s', v) &= (s + s', u + v) \end{aligned}$$

and constants $0 = (0, 0)$ and $1 = (1, 0)$. Equipped with these operations and constants, T satisfies the equations

$$\begin{aligned} (x + y) + z &= x + (y + z) && (5)\\ x + y &= y + x && (6)\\ x + 0 &= x && (7)\\ (x \cdot y) \cdot z &= x \cdot (y \cdot z) && (8)\\ x \cdot 1 &= x && (9)\\ 1 \cdot x &= x && (10)\\ (x + y) \cdot z &= (x \cdot z) + (y \cdot z) && (11)\\ 0 \cdot x &= 0. && (12) \end{aligned}$$

Elgot[8] also defined the unary operation ¶ on T: $(s, u)¶ = (s, 0)$. Thus, ¶ selects the "first component" of the pair (s, u), while multiplication with 0 on the right selects the "second component", for $(s, u) \cdot 0 = (0, u)$, for all $u \in V$. The new

operation satisfies:

$$\begin{aligned} x\P \cdot (y+z) &= (x\P \cdot y) + (x\P \cdot z) && (13)\\ x &= x\P + (x \cdot 0) && (14)\\ x\P \cdot 0 &= 0 && (15)\\ (x+y)\P &= x\P + y\P && (16)\\ (x \cdot y)\P &= x\P \cdot y\P. && (17) \end{aligned}$$

Note that when V is idempotent, also

$$x \cdot (y+z) = x \cdot y + x \cdot z$$

holds.

Elgot[8] defined a *quemiring* to be an algebraic structure T equipped with the above operations $\cdot, +, \P$ and constants $0, 1$ satisfying the equations (5)–(12) and (13)–(17). A morphism of quemirings is a function preserving the operations and constants. It follows from the axioms that $x\P\P = x\P$, for all x in a quemiring T. Moreover, $x\P = x$ iff $x \cdot 0 = 0$.

When T is a quemiring, $A = T\P = \{x\P \mid x \in T\}$ is easily seen to be a semiring. Moreover, $V = T0 = \{x \cdot 0 \mid x \in T\}$ contains 0 and is closed under $+$, and, furthermore, $sx \in V$ for all $s \in A$ and $x \in V$. Each $x \in T$ may be written in a unique way as the sum of an element of $T\P$ and a sum of an element of $T0$ as $x = x\P + x \cdot 0$. Sometimes, we will identify $A \times \{0\}$ with A and $\{0\} \times V$ with V. It is shown in Elgot [8] that T is isomorphic to the quemiring $A \times V$ determined by the semiring-semimodule pair (A, V).

Suppose now that (A, V) is a starsemiring-omegasemimodule pair. Then we define on $T = A \times V$ a *generalized star operation*:

$$(s,v)^{\otimes} = (s^*, s^\omega + s^* v) \qquad (18)$$

for all $(s, v) \in T$. Note that the star and omega operations can be recovered from the generalized star operation, since s^* is the first component of $(s,0)^{\otimes}$ and s^ω is the second component. Thus:

$$\begin{aligned} (s^*, 0) &= (s,0)^{\otimes}\P\\ (0, s^\omega) &= (s,0)^{\otimes} \cdot 0. \end{aligned}$$

Observe that, for $(s,0) \in A \times \{0\}$, $(s,0)^{\otimes} = (s^*, 0) + (0, s^\omega)$.

Suppose now that T is an (abstract) quemiring equipped with a generalized star operation $^{\otimes}$. As explained above, T as a quemiring is isomorphic to the quemiring $A \times V$ associated with the semiring-semimodule pair (A, V), where $A = T\P$ and $V = T0$, an isomorphism being the map $x \mapsto (x\P, x \cdot 0)$. It is clear that a generalized star operation $^{\otimes} : T \to T$ is determined by a star operation $^* : A \to A$ and an omega operation $^\omega : A \to V$ by (18) iff

$$\begin{aligned} x^{\otimes}\P &= (x\P)^{\otimes}\P && (19)\\ x^{\otimes} \cdot 0 &= (x\P)^{\otimes} \cdot 0 + x^{\otimes}\P \cdot x \cdot 0 && (20) \end{aligned}$$

hold. Indeed, these conditions are clearly necessary. Conversely, if (19) and (20) hold, then for any $x\P \in T\P$ we may define

$$(x\P)^* = (x\P)^{\otimes}\P \tag{21}$$
$$(x\P)^\omega = (x\P)^{\otimes} \cdot 0\,. \tag{22}$$

It follows that (18) holds. The definition of star and omega was forced.

Let us call a quemiring equipped with a generalized star operation $^{\otimes}$ a *generalized starquemiring.* Morphisms of generalized starquemirings preserve the quemiring structure and the $^{\otimes}$ operation.

We now refer to some examples for the algebraic structures defined in this section. All the following semiring-semimodule pairs are complete. Hence, they are starsemiring-omegasemimodule pairs and Conway semiring-semimodule pairs, and by (18) give rise to a generalized starquemiring.

(i) The pair $(\mathfrak{P}(\Sigma^*), \mathfrak{P}(\Sigma^\omega))$, where Σ is an alphabet and $\mathfrak{P}$ denotes the power set, is a complete semiring-semimodule pair. The first component of this pair is the set of formal languages over finite words over Σ, the second component is the set of formal languages over infinite words over Σ. (See Ésik, Kuich [10], Example 3.2.)

(ii) The pair $(\mathbb{N}^\infty\langle\langle\Sigma^*\rangle\rangle, \mathbb{N}^\infty\langle\langle\Sigma^\omega\rangle\rangle)$, where $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$ denotes the complete semiring of nonnegative integers augmented by ∞ with the usual operations, is a complete semiring-semimodule pair. The first component of this pair is the set of power series with coefficients in $\mathbb{N}^\infty$ over the finite words over Σ, the second component is the set of power series with coefficients in $\mathbb{N}^\infty$ over the infinite words over Σ. This pair is used if ambiguities of the formal languages in (i) are considered. (See Ésik, Kuich [10], Example 3.3.)

(iii) The pair $(\mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^*\rangle\rangle, \mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^\omega\rangle\rangle)$ is a complete semiring-semimodule pair. It is defined before Corollary 30.

(iv) The clock languages of Bouyer, Petit [3] give rise to a complete semiring-semimodule pair. (See Ésik, Kuich [11].)

2 Skew power series over Conway semirings

Let A be a starsemiring. Then, for $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, we define $r^* \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, called the *star of* r by

$$(r^*, \varepsilon) = (r, \varepsilon)^*\,,$$
$$(r^*, w) = (r, \varepsilon)^* \cdot \sum_{uv=w,\ u\neq\varepsilon} (r, u)\varphi^{|u|}(r^*, v)\,.$$

Moreover, we define $r^+ \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ by $r^+ = rr^*$. We prove now the result that the structure $\langle A^{\Sigma^*}, +, \odot_\varphi, *, 0, 1\rangle$, again denoted by $A_\varphi\langle\langle\Sigma^*\rangle\rangle$, is a Conway semiring if A is a Conway semiring. The proof of this result is a generalization of the proofs of Theorems 2.19, 2.20, 2.21 of Aleshnikov, Boltnev, Ésik, Ishanov, Kuich, Malachowskij [1]

Theorem 2. *Let A be a Conway semiring, $\varphi : A \to A$ be an endomorphism and Σ be an alphabet. Then the sum-star equation holds in $A_\varphi\langle\langle\Sigma^*\rangle\rangle$.*

Proof. Let $r, s \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then we prove by induction on the length of $w \in \Sigma^*$ that $((r+s)^*, w) = ((r^*s)^*r^*, w)$. The case $w = \varepsilon$ is clear. Assume now $w \neq \varepsilon$. Then we obtain $((r+s)^*, w) = ((r+s)^*, \varepsilon) \sum_{uv=w,\ u\neq\varepsilon}(r+s, u)\varphi^{|u|}((r+s)^*, v) = ((r+s)^*, \varepsilon) \sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}((r+s)^*, v) + ((r+s)^*, \varepsilon) \sum_{uv=w,\ u\neq\varepsilon}(s, u)\varphi^{|u|}((r+s)^*, v)$. We call the first and second of these terms L_1 and L_2, respectively. Moreover, we obtain

$$
\begin{aligned}
&((r^*s)^*r^*, w) = \textstyle\sum_{w_1w_2=w}((r^*s)^*, w_1)\varphi^{|w_1|}(r^*, w_2) = \\
&((r^*s)^*, \varepsilon)(r^*, w) + \textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}((r^*s)^*, w_1)\varphi^{|w_1|}(r^*, w_2) = \\
&((r^*s)^*, \varepsilon)(r^*, w) + \\
&\quad ((r^*s)^*, \varepsilon) \textstyle\sum_{w_1w_2=w} \sum_{u_1v_1=w_1,\ u_1\neq\varepsilon}(r^*s, u_1)\varphi^{|u_1|}((r^*s)^*, v_1)\varphi^{|w_1|}(r^*, w_2) = \\
&((r^*s)^*, \varepsilon)(r^*, w) + ((r^*s)^*, \varepsilon) \textstyle\sum_{w_1w_2=w} \sum_{u_1v_1=w_1, u_1\neq\varepsilon} \sum_{w_3w_4=u_1} \\
&\quad (r^*, w_3)\varphi^{|w_3|}(s, w_4)\varphi^{|u_1|}((r^*s)^*, v_1)\varphi^{|w_1|}(r^*, w_2) = \\
&((r^*s)^*, \varepsilon)(r^*, w) + ((r^*s)^*, \varepsilon) \cdot \\
&\quad \textstyle\sum_{w_1w_2=w} \sum_{u_1v_1=w_1,\ u_1\neq\varepsilon}(r^*, \varepsilon)(s, u_1)\varphi^{|u_1|}((r^*s)^*, v_1)\varphi^{|w_1|}(r^*, w_2) + \\
&\quad ((r^*s)^*, \varepsilon) \textstyle\sum_{w_1w_2=w} \sum_{u_1v_1=w_1} \sum_{w_3w_4=u_1,\ w_3\neq\varepsilon} \\
&\quad (r^*, w_3)\varphi^{|w_3|}(s, w_4)\varphi^{|u_1|}((r^*s)^*, v_1)\varphi^{|w_1|}(r^*, w_2).
\end{aligned}
$$

We call the first, second and third of these terms R_1, R_2 and R_3, respectively.

Eventually, we obtain

$$
\begin{aligned}
R_2 &= ((r+s)^*, \varepsilon) \textstyle\sum_{u_1z=w,\ u_1\neq\varepsilon}(s, u_1)\varphi^{|u_1|}((r^*s)^*r^*, z) = \\
&((r+s)^*, \varepsilon) \textstyle\sum_{u_1z=w,\ u_1\neq\varepsilon}(s, u_1)\varphi^{|u_1|}((r+s)^*, z) = L_2
\end{aligned}
$$

and

$$
\begin{aligned}
R_1 + R_3 &= ((r^*s)^*, \varepsilon)(r^*, \varepsilon) \textstyle\sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}(r^*, v) + \\
&\quad ((r^*s)^*, \varepsilon) \textstyle\sum_{w_1w_2=w} \sum_{u_1v_1=w_1} \sum_{w_3w_4=u_1}(r^*, \varepsilon) \cdot \\
&\quad \textstyle\sum_{u_2v_2=w_3,\ u_2\neq\varepsilon}(r, u_2)\varphi^{|u_2|}(r^*, v_2)\varphi^{|w_3|}(s, w_4)\varphi^{|u_1|}((r^*s)^*, v_1)\varphi^{|w_1|}(r^*, w_2) = \\
&((r+s)^*, \varepsilon) \textstyle\sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}(r^*, v) + \\
&\quad ((r+s)^*, \varepsilon) \textstyle\sum_{u_2z=w,\ u_2\neq\varepsilon}(r, u_2)\varphi^{|u_2|}((r^*s)^+r^*, z) = \\
&((r+s)^*, \varepsilon) \textstyle\sum_{u_2z=w,\ u_2\neq\varepsilon}(r, u_2)\varphi^{|u_2|}((r^*s)^*r^*, z) = \\
&((r+s)^*, \varepsilon) \textstyle\sum_{u_2z=w,\ u_2\neq\varepsilon}(r, u_2)\varphi^{|u_2|}((r+s)^*, z) = L_1 .
\end{aligned}
$$

Hence, $L_1 + L_2 = R_1 + R_2 + R_3$ and the sum-star equation holds in $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. □

Theorem 3. *Let A be a Conway semiring, $\varphi : A \to A$ be an endomorphism and Σ be an alphabet. Then, for $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, the following equation is satisfied:*

$$r^* = \varepsilon + rr^* .$$

Proof. We prove by induction on the length of $w \in \Sigma^*$ that $(r^*, w) = (\varepsilon + rr^*, w)$. The case $w = \varepsilon$ is clear. Assume now $w \neq \varepsilon$. Then we obtain

$$
\begin{aligned}
&(\varepsilon + rr^*, w) = \textstyle\sum_{w_1w_2=w}(r, w_1)\varphi^{|w_1|}(r^*, w_2) = \\
&(r, \varepsilon)(r^*, w) + \textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}(r^*, w_2) = \\
&(r, \varepsilon)(r^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}(r^*v) + \sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}(r^*, w_2) = \\
&(r^+, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}(r^*v) + \sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}(r^*, w_2) = \\
&(r^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}(r^*v) = (r^*, w)\,.
\end{aligned}
$$

□

Theorem 4. *Let A be a Conway semiring, $\varphi : A \to A$ be an endomorphism and Σ be an alphabet. Then, for $r, s \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, the following equation is satisfied:*

$$r(sr)^* = (rs)^*r\,.$$

Proof. We prove by induction on the length of $w \in \Sigma^*$ that $(r(sr)^*, w) = ((rs)^*r, w)$. The case $w = \varepsilon$ is clear. Assume now $w \neq \varepsilon$. Then we obtain

$$
\begin{aligned}
&(r(sr)^*, w) = \textstyle\sum_{w_1w_2=w}(r, w_1)\varphi^{|w_1|}((sr)^*, w_2) = \\
&(r, \varepsilon)((sr)^*, w) + \textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}((sr)^*, w_2) = \\
&(r, \varepsilon)((sr)^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(sr, u)\varphi^{|u|}((sr)^*, v) + \\
&\quad\textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}((sr)^*, w_2) = \\
&(r(sr)^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}\sum_{w_3w_4=u}(s, w_3)\varphi^{|w_3|}(r, w_4)\varphi^{|u|}((sr)^*, v) + \\
&\quad\textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}((sr)^*, w_2) = \\
&(r(sr)^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(s, \varepsilon)(r, u)\varphi^{|u|}((sr)^*, v) + \\
&\quad(r(sr)^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{w_3w_4=u,\ w_3\neq\varepsilon}(s, w_3)\varphi^{|w_3|}(r, w_4)\varphi^{|u|}((sr)^*, v) + \\
&\quad\textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}((sr)^*, w_2) = \\
&((rs)^+, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}((sr)^*, v) + \\
&\quad(r(sr)^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{w_3w_4=u,\ w_3\neq\varepsilon}(s, w_3)\varphi^{|w_3|}(r, w_4)\varphi^{|u|}((sr)^*, v) + \\
&\quad\textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}(r, w_1)\varphi^{|w_1|}((sr)^*, w_2) = \\
&((rs)^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(r, u)\varphi^{|u|}((sr)^*, v) + \\
&\quad(r(sr)^*, \varepsilon)\textstyle\sum_{w_3z=w,\ w_3\neq\varepsilon}(s, w_3)\varphi^{|w_3|}(r(sr)^*, z)
\end{aligned}
$$

and

$$
\begin{aligned}
&((rs)^*r, w) = \textstyle\sum_{w_1w_2=w}((rs)^*, w_1)\varphi^{|w_1|}(r, w_2) = \\
&((rs)^*, \varepsilon)(r, w) + \textstyle\sum_{w_1w_2=w,\ w_1\neq\varepsilon}((rs)^*, w_1)\varphi^{|w_1|}(r, w_2) = \\
&((rs)^*, \varepsilon)(r, w) + \\
&\quad\textstyle\sum_{w_1w_2=w}((rs)^*, \varepsilon)\sum_{uv=w_1,\ u\neq\varepsilon}(rs, u)\varphi^{|u|}((rs)^*, v)\varphi^{|w_1|}(r, w_2) = \\
&((rs)^*, \varepsilon)(r, w) + \textstyle\sum_{w_1w_2=w}((rs)^*, \varepsilon)\cdot \\
&\quad\textstyle\sum_{uv=w_1,\ u\neq\varepsilon}\sum_{w_3w_4=u}(r, w_3)\varphi^{|w_3|}(s, w_4)\varphi^{|u|}((rs)^*, v)\varphi^{|w_1|}(r, w_2) = \\
&((rs)^*, \varepsilon)(r, w) + \textstyle\sum_{w_1w_2=w}((rs)^*, \varepsilon)\cdot \\
&\quad\textstyle\sum_{uv=w_1,\ u\neq\varepsilon}(r, \varepsilon)(s, u)\varphi^{|u|}((rs)^*, v)\varphi^{|w_1|}(r, w_2) + \textstyle\sum_{w_1w_2=w}((rs)^*, \varepsilon)\cdot \\
&\quad\textstyle\sum_{uv=w_1}\sum_{w_3w_4=u,\ w_3\neq\varepsilon}(r, w_3)\varphi^{|w_3|}(s, w_4)\varphi^{|u|}((rs)^*, v)\varphi^{|w_1|}(r, w_2) = \\
&((rs)^*, \varepsilon)(r, w) + ((rs)^*r, \varepsilon)\textstyle\sum_{uz=w,\ u\neq\varepsilon}(s, u)\varphi^{|u|}((rs)^*r, z) + \\
&\quad((rs)^*, \varepsilon)\textstyle\sum_{w_3z=w,\ w_3\neq\varepsilon}(r, w_3)\varphi^{|w_3|}((sr)^+, z) =
\end{aligned}
$$

$$
\begin{aligned}
&((rs)^*,\varepsilon)(r,w)+((rs)^*r,\varepsilon)\textstyle\sum_{uz=w,\ u\neq\varepsilon}(s,u)\varphi^{|u|}((rs)^*r,z)+\\
&\quad((rs)^*,\varepsilon)\textstyle\sum_{w_3z=w,\ w_3\neq\varepsilon,\ z\neq\varepsilon}(r,w_3)\varphi^{|w_3|}((sr)^*,z)+\\
&\quad((rs)^*,\varepsilon)(r,w)\varphi^{|w|}((sr)^+,\varepsilon)=\\
&((rs)^*r,\varepsilon)\textstyle\sum_{uz=w,\ u\neq\varepsilon}(s,u)\varphi^{|u|}((rs)^*r,z)+\\
&\quad((rs)^*,\varepsilon)\textstyle\sum_{w_3z=w,\ w_3\neq\varepsilon}(r,w_3)\varphi^{|w_3|}((sr)^*,z)\,.
\end{aligned}
$$

Hence, $(r(sr)^*,w)=((rs)^*r,w)$. □

Corollary 5. *If A is a Conway semiring, $\varphi: A\to A$ is an endomorphism and Σ is an alphabet then $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is again a Conway semiring.*

Proof. The equations of Theorems 3 and 4 hold iff the product-star equation holds. □

Corollary 6 (Bloom, Ésik [2]). *If A is a Conway semiring and Σ is an alphabet then $A\langle\langle\Sigma^*\rangle\rangle$ is again a Conway semiring.*

In the next corollary we consider $A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$. Here $\varphi: A^{n\times n}\to A^{n\times n}$ is the pointwise extension of the endomorphism $\varphi: A\to A$. Clearly, the extended φ is again an endomorphism. Note that the set $A^{n\times n}$ of $n\times n$-matrices is equipped with the usual matrix operations addition and multiplication.

Corollary 7. *Let A be a Conway semiring, $\varphi: A\to A$ be an endomorphism, Σ be an alphabet and $n\geq 1$. Then $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ and $A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ are again Conway semirings.*

Theorem 8. *Let A be a Conway semiring, $\varphi: A\to A$ be an endomorphism, Σ be an alphabet and $n\geq 1$. Then $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ and $A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ are isomorphic starsemirings.*

Proof. We will prove that $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ and $A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ are isomorphic by the correspondence of $M\in(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ and $M'\in A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ given by $(M_{ij},w)=(M',w)_{ij}$, $w\in\Sigma^*$, $1\leq i,j\leq n$.

We prove only the compatibility of multiplication and star. Let $M_1,M_2\in(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ with corresponding $M_1',M_2'\in A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$, respectively. Then, for all $w\in\Sigma^*$ and $1\leq i,j\leq n$, we obtain

$$
\begin{aligned}
&(M_1'M_2',w)_{ij}=(\textstyle\sum_{uv=w}(M_1',u)\varphi^{|u|}(M_2',v))_{ij}=\\
&\textstyle\sum_{uv=w}\sum_{1\leq k\leq n}(M_1',u)_{ik}\varphi^{|u|}((M_2',v)_{kj})=\\
&\textstyle\sum_{1\leq k\leq n}\sum_{uv=w}((M_1)_{ik},u)\varphi^{|u|}((M_2)_{kj},v)=\\
&\textstyle\sum_{1\leq k\leq n}((M_1)_{ik}(M_2)_{kj},w)=((M_1M_2)_{ij},w)\,.
\end{aligned}
$$

Let now $M\in(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ correspond to $M'\in A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$. We assume that M is partitioned as usual into blocks

$$
M=\begin{pmatrix} a & b\\ c & d\end{pmatrix},
$$

where $a \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{1\times 1}$, $b \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{1\times(n-1)}$, $c \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{(n-1)\times 1}$, $d \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{(n-1)\times(n-1)}$. We first show by induction on the length of $w \in \Sigma^*$ that $((M^*)_{11}, w) = ((a + bd^*c)^*, w)$ and $(M'^*, w)_{11} = ((M'^*, \varepsilon)\sum_{uv=w,\ u\neq\varepsilon}(M', u) \cdot \varphi^{|u|}(M'^*, v))_{11}$ coincide. The case $w = \varepsilon$ is clear. Assume now $w \neq \varepsilon$. Then we obtain

$$
\begin{aligned}
&((M^*)_{11}, w) = (a + bd^*c, \varepsilon)^* \textstyle\sum_{uv=w,\ u\neq\varepsilon}(a + bd^*c, u)\varphi^{|u|}((a + bd^*c)^*, v) = \\
&(a + bd^*c, \varepsilon)^* \textstyle\sum_{uv=w,\ u\neq\varepsilon}(a, u)\varphi^{|u|}((a + bd^*c)^*, v) + \\
&\quad (a + bd^*c, \varepsilon)^* \textstyle\sum_{uv=w,\ u\neq\varepsilon}\sum_{z_1z_2=u}(bd^*, z_1)\varphi^{|z_1|}(c, z_2)\varphi^{|u|}((a + bd^*c)^*, v) = \\
&(a + bd^*c, \varepsilon)^* \textstyle\sum_{uv=w,\ u\neq\varepsilon}(a, u)\varphi^{|u|}((a + bd^*c)^*, v) + \\
&(a + bd^*c, \varepsilon)^* \textstyle\sum_{uv=w,\ u\neq\varepsilon}(bd^*, \varepsilon)(c, u)\varphi^{|u|}((a + bd^*c)^*, v) + \\
&(a + bd^*c, \varepsilon)^* \textstyle\sum_{uv=w}\sum_{z_1z_2=u,\ z_1\neq\varepsilon}(bd^*, z_1)\varphi^{|z_1|}(c, z_2)\varphi^{|u|}((a + bd^*c)^*, v)\,.
\end{aligned}
$$

We call the first, second and third of these terms L_1, L_2 and L_3, respectively.

Moreover, we obtain

$$
\begin{aligned}
&(M'^*, w)_{11} = \textstyle\sum_{1\leq i,j\leq n}(M'^*, \varepsilon)_{1i}\sum_{uv=w,\ u\neq\varepsilon}(M', u)_{ij}\varphi^{|u|}((M'^*, v)_{j1}) = \\
&\textstyle\sum_{1\leq i,j\leq n}((M^*)_{1i}, \varepsilon)\sum_{uv=w,\ u\neq\varepsilon}(M_{ij}, u)\varphi^{|u|}((M^*)_{j1}, v) = \\
&((a + bd^*c)^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(a, u)\varphi^{|u|}((a + bd^*c)^*, v) + \\
&((a + bd^*c)^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(b, u)\varphi^{|u|}(d^*c(a + bd^*c)^*, v) + \\
&((a + bd^*c)^*bd^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(c, u)\varphi^{|u|}((a + bd^*c)^*, v) + \\
&((a + bd^*c)^*bd^*, \varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(d, u)\varphi^{|u|}(d^*c(a + bd^*c)^*, v)\,.
\end{aligned}
$$

We call the first, second, third and fourth of these terms R_1, R_2, R_3 and R_4, respectively.

It is clear that $L_1 = R_1$ and $L_2 = R_3$. Hence, we have only to prove that $L_3 = R_2 + R_4$. We obtain

$$
\begin{aligned}
&L_3 = ((a + bd^*c)^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{z_1z_2=u,\ z_1\neq\varepsilon}\sum_{z_3z_4=z_1} \\
&\quad (b, z_3)\varphi^{|z_3|}(d^*, z_4)\varphi^{|z_1|}(c, z_2)\varphi^{|u|}((a + bd^*c)^*, v) = \\
&((a + bd^*c)^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{z_1z_2=u}\sum_{z_3z_4=z_1}(b, z_3)\varphi^{|z_3|}(d^*, \varepsilon) \cdot \\
&\quad \textstyle\sum_{u_1v_1=z_4,\ u_1\neq\varepsilon}\varphi^{|z_3|}(d, u_1)\varphi^{|z_3u_1|}(d^*, v_1)\varphi^{|z_1|}(c, z_2)\varphi^{|u|}((a + bd^*c)^*, v) = \\
&((a + bd^*c)^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{z_1z_2=u}(b, \varepsilon)(d^*, \varepsilon) \cdot \\
&\quad \textstyle\sum_{u_1v_1=z_1,\ u_1\neq\varepsilon}(d, u_1)\varphi^{|u_1|}(d^*, v_1)\varphi^{|z_1|}(c, z_2)\varphi^{|u|}((a + bd^*c)^*, v) + \\
&\quad ((a + bd^*c)^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{z_1z_2=u}\sum_{z_3z_4=z_1,\ z_3\neq\varepsilon}(b, z_3)\varphi^{|z_3|}(d^*, \varepsilon) \cdot \\
&\quad \textstyle\sum_{u_1v_1=z_4,\ u_1\neq\varepsilon}\varphi^{|z_3|}(d, u_1)\varphi^{|z_3u_1|}(d^*, v_1)\varphi^{|z_1|}(c, z_2)\varphi^{|u|}((a + bd^*c)^*, v)\,.
\end{aligned}
$$

We call the first and second of these terms L_4 and L_5, respectively.

We now obtain

$$
\begin{aligned}
&L_4 = ((a + bd^*c)^*bd^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{z_1z_2=u}\sum_{u_1v_1=z_1,\ u_1\neq\varepsilon} \\
&\quad (d, u_1)\varphi^{|u_1|}(d^*, v_1)\varphi^{|z_1|}(c, z_2)\varphi^{|u|}((a + bd^*c)^*, v) = \\
&((a + bd^*c)^*bd^*, \varepsilon)\textstyle\sum_{uv=w}\sum_{u_1z_3=u,\ u_1\neq\varepsilon} \\
&\quad (d, u_1)\varphi^{|u_1|}(d^*c, z_3)\varphi^{|u|}((a + bd^*c)^*, v) = \\
&((a + bd^*c)^*bd^*, \varepsilon)\textstyle\sum_{u_1z_4=w,\ u_1\neq\varepsilon}(d, u_1)\varphi^{|u_1|}(d^*c(a + bd^*c)^*, z_4) = R_4\,.
\end{aligned}
$$

Eventually, we obtain

$$
\begin{aligned}
&L_5 = ((a+bd^*c)^*,\varepsilon)\textstyle\sum_{uv=w}\sum_{z_1z_2=u}\sum_{z_3z_4=z_1,\ z_3\neq\varepsilon}\\
&\quad (b,z_3)\varphi^{|z_3|}(d^*,z_4)\varphi^{|z_1|}(c,z_2)\varphi^{|u|}((a+bd^*c)^*,v) =\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{uv=w}\sum_{z_3z_5=u,\ z_3\neq\varepsilon}\\
&\quad (b,z_3)\varphi^{|z_3|}(d^*c,z_5)\varphi^{|u|}((a+bd^*c)^*,v) =\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_3z_6=w,\ z_3\neq\varepsilon}(b,z_3)\varphi^{|z_3|}(d^*c(a+bd^*c)^*,z_6) = R_2\,.
\end{aligned}
$$

Hence, $L_3 = L_4 + L_5 = R_4 + R_2$.

Next, we prove by induction on the length of $w \in \Sigma^*$ that the $(1,2)$-blocks of M^* and M'^* correspond to each other: $((M^*)_{12},w) = (M'^*,w)_{12}$. Here we have $((M^*)_{12},w) = ((a+bd^*c)^*bd^*,w)$ and $(M'^*,w)_{12} = ((M'^*,\varepsilon)\sum_{uv=w,\ u\neq\varepsilon}(M',u)\cdot\varphi^{|u|}(M'^*,v))_{12}$. The case $w=\varepsilon$ is clear. Assume now $w\neq\varepsilon$. Then we obtain

$$
\begin{aligned}
&((M^*)_{12},w) = \textstyle\sum_{z_1z_2=w}((a+bd^*c)^*,z_1)\varphi^{|z_1|}(bd^*,z_2) =\\
&(a+bd^*c,\varepsilon)^*(bd^*,w) + \textstyle\sum_{z_1z_2=w}(a+bd^*c,\varepsilon)^*\sum_{uv=z_1,\ u\neq\varepsilon}\\
&\quad (a+bd^*c,u)\varphi^{|u|}((a+bd^*c)^*,v)\varphi^{|z_1|}(bd^*,z_2) =\\
&((a+bd^*c)^*,\varepsilon)(bd^*,w) +\\
&\textstyle\sum_{uvz_2=w,\ u\neq\varepsilon}((a+bd^*c)^*,\varepsilon)(a,u)\varphi^{|u|}((a+bd^*c)^*,v)\varphi^{|uv|}(bd^*,z_2) +\\
&\textstyle\sum_{uvz_2=w,\ u\neq\varepsilon}((a+bd^*c)^*,\varepsilon)(bd^*,\varepsilon)(c,u)\varphi^{|u|}((a+bd^*c)^*,v)\varphi^{|uv|}(bd^*,z_2) +\\
&\textstyle\sum_{z_1z_2=w}((a+bd^*c)^*,\varepsilon)\sum_{uv=z_1}\sum_{z_3z_4=u,\ z_3\neq\varepsilon}\\
&\quad (bd^*,z_3)\varphi^{|z_3|}(c,z_4)\varphi^{|u|}((a+bd^*c)^*,v)\varphi^{|z_1|}(bd^*,z_2)\,.
\end{aligned}
$$

We call the first, second, third and fourth of these terms L_0, L_1, L_2 and L_3, respectively.

Moreover, we obtain

$$
\begin{aligned}
&(M'^*,w)_{12} = \textstyle\sum_{1\le i,j\le n}(M'^*,\varepsilon)_{1i}\sum_{uv=w,\ u\neq\varepsilon}(M',u)_{ij}\varphi^{|u|}((M'^*,v)_{j2}) =\\
&\textstyle\sum_{1\le i,j\le n}((M^*)_{1i},\varepsilon)\sum_{uv=w,\ u\neq\varepsilon}(M_{ij},u)\varphi^{|u|}((M^*)_{j2},v) =\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(a,u)\varphi^{|u|}((a+bd^*c)^*bd^*,v) +\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(b,u)\varphi^{|u|}((d+ca^*b)^*,v) +\\
&((a+bd^*c)^*bd^*,\varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(c,u)\varphi^{|u|}((a+bd^*c)^*bd^*,v) +\\
&((a+bd^*c)^*bd^*,\varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(d,u)\varphi^{|u|}((d+ca^*b)^*,v)\,.
\end{aligned}
$$

We call the first, second, third and fourth of these terms R_1, R_2, R_3 and R_4, respectively.

It is clear that $L_1 = R_1$ and $L_2 = R_3$. Hence, we have only to prove that $L_0 + L_3 = R_2 + R_4$. We obtain

$$
\begin{aligned}
&L_0 + L_3 =\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_1z_2=w}(b,z_1)\varphi^{|z_1|}(d^*,z_2) +\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_3z_5=w,\ z_3\neq\varepsilon}(bd^*,z_3)\varphi^{|z_3|}(c(a+bd^*c)^*bd^*,z_5) =\\
&((a+bd^*c)^*,\varepsilon)(b,\varepsilon)(d^*w) +\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_1z_2=w,\ z_1\neq\varepsilon}(b,z_1)\varphi^{|z_1|}(d^*,z_2) +\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_3z_5=w,\ z_3\neq\varepsilon}(b,\varepsilon)(d^*,z_3)\varphi^{|z_3|}(c(a+bd^*c)^*bd^*,z_5) +\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_3z_5=w}\sum_{z_6z_7=z_3,\ z_6\neq\varepsilon}\\
&\quad (b,z_6)\varphi^{|z_6|}(d^*,z_7)\varphi^{|z_3|}(c(a+bd^*c)^*bd^*,z_5) =
\end{aligned}
$$

$$
\begin{aligned}
&((a+bd^*c)^*,\varepsilon)(b,\varepsilon)(d^*,\varepsilon)\textstyle\sum_{uv=w,\ u\neq\varepsilon}(d,u)\varphi^{|u|}(d^*,v)+\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_1z_2=w,\ z_1\neq\varepsilon}(b,z_1)\varphi^{|z_1|}(d^*,z_2)+\\
&((a+bd^*c)^*,\varepsilon)(b,\varepsilon)\textstyle\sum_{z_3z_5=w}(d^*,\varepsilon)\sum_{uv=z_3,u\neq\varepsilon}\\
&\quad(d,u)\varphi^{|u|}(d^*,v)\varphi^{|z_3|}(c(a+bd^*c)^*bd^*,z_5)+\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_6z_8=w,z_6\neq\varepsilon}(b,z_6)\varphi^{|z_6|}(d^*c(a+bd^*c)^*bd^*,z_8)=\\
&((a+bd^*c)^*bd^*,\varepsilon)\textstyle\sum_{uv=w,u\neq\varepsilon}(d,u)\varphi^{|u|}(d^*,v)+\\
&((a+bd^*c)^*bd^*,\varepsilon)\textstyle\sum_{uz_6=w,u\neq\varepsilon}(d,u)\varphi^{|u|}(d^*c(a+bd^*c)^*bd^*,z_6)+\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_1z_2=w,\ z_1\neq\varepsilon}(b,z_1)\varphi^{|z_1|}(d^*,z_2)+\\
&((a+bd^*c)^*,\varepsilon)\textstyle\sum_{z_6z_8=w,\ z_6\neq\varepsilon}(b,z_6)\varphi^{|z_6|}(d^*c(a+bd^*c)^*bd^*,z_8)=R_4+R_2\,.
\end{aligned}
$$

Here we have used in the last equality the equation $(d+ca^*b)^*=d^*+d^*c(a+bd^*c)^*bd^*$.

The equality of the (2,1)- and (2,2)-blocks is proved by symmetry: interchange 1 and 2, a and d, b and c. □

Corollary 9. *Let A be a Conway semiring and Σ be an alphabet. Then $(A\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ and $A^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ are isomorphic starsemirings.*

Let $\varphi,\varphi' : A\to A$ be endomorphisms. Then we define the mapping $\varphi'_\Sigma : A_\varphi\langle\langle\Sigma^*\rangle\rangle \to A_\varphi\langle\langle\Sigma^*\rangle\rangle$ by $(\varphi'_\Sigma(r),w)=\varphi'(r,w)$, $r\in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, for all $w\in\Sigma^*$. Moreover, φ and φ' are *commuting* if, for all $a\in A$, $\varphi(\varphi'(a))=\varphi'(\varphi(a))$.

The next theorem is a special case of Theorem 4.3 of Droste, Kuske [5].

Theorem 10. *Let $\varphi,\varphi' : A\to A$ be commuting endomorphisms. Then $\varphi'_\Sigma : A_\varphi\langle\langle\Sigma^*\rangle\rangle \to A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is an endomorphism.*

Proof. Clearly, $\varphi'_\Sigma(0)=0$ and $\varphi'_\Sigma(\varepsilon)=\varepsilon$. Let now $r_1,r_2\in A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then, for all $w\in\Sigma^*$, $(\varphi'_\Sigma(r_1+r_2),w)=\varphi'(r_1+r_2,w)=\varphi'(r_1,w)+\varphi'(r_2,w)=(\varphi'_\Sigma(r_1),w)+(\varphi'_\Sigma(r_2),w)$, i. e.,

$$\varphi'_\Sigma(r_1+r_2)=\varphi'_\Sigma(r_1)+\varphi'_\Sigma(r_2)\,,$$

and $(\varphi'_\Sigma(r_1\odot_\varphi r_2),w)=\varphi'(r_1\odot_\varphi r_2,w)=\varphi'(\sum_{w_1w_2=w}(r_1,w_1)\varphi^{|w_1|}(r_2,w_2))=\sum_{w_1w_2=w}\varphi'(r_1,w_1)\varphi'(\varphi^{|w_1|}(r_2,w_2))=\sum_{w_1w_2=w}\varphi'(r_1,w_1)\varphi^{|w_1|}(\varphi'(r_2,w_2))=\sum_{w_1w_2=w}(\varphi'_\Sigma(r_1),w_1)\varphi^{|w_1|}(\varphi'_\Sigma(r_2),w_2))=(\varphi'_\Sigma(r_1)\odot_\varphi\varphi'_\Sigma(r_2),w)$, i. e.,

$$\varphi'_\Sigma(r_1\odot_\varphi r_2)=\varphi'_\Sigma(r_1)\odot_\varphi\varphi'_\Sigma(r_2)\,.$$

□

Corollary 11. *Let $\varphi : A\to A$ be an endomorphism. Then $\varphi_\Sigma : A_\varphi\langle\langle\Sigma^*\rangle\rangle \to A_\varphi\langle\langle\Sigma^*\rangle\rangle$ and $\varphi_\Sigma : A\langle\langle\Sigma^*\rangle\rangle \to A\langle\langle\Sigma^*\rangle\rangle$ are endomorphisms.*

Corollary 12. *Let A be a Conway semiring, $\varphi : A\to A$ be an endomorphism, and Σ_1,Σ_2 be alphabets. Then $(A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)_{\varphi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$, $(A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)\langle\langle\Sigma_2^*\rangle\rangle$, $(A\langle\langle\Sigma_1^*\rangle\rangle)_{\varphi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$ and $(A\langle\langle\Sigma_1^*\rangle\rangle)\langle\langle\Sigma_2^*\rangle\rangle$ are again Conway semirings.*

Theorem 13. *Let A be a Conway semiring, $\varphi,\psi : A\to A$ be commuting endomorphisms and Σ_1,Σ_2 be alphabets. Then $(A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)_{\psi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$ and $(A_\psi\langle\langle\Sigma_2^*\rangle\rangle)_{\varphi_{\Sigma_2}}\langle\langle\Sigma_1^*\rangle\rangle$ are isomorphic starsemirings.*

Proof. We will prove that $(A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)_{\psi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$ and $(A_\psi\langle\langle\Sigma_2^*\rangle\rangle)_{\varphi_{\Sigma_2}}\langle\langle\Sigma_1^*\rangle\rangle$ are isomorphic by the correspondence of $r \in (A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)_{\psi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$ and $r' \in (A_\psi\langle\langle\Sigma_2^*\rangle\rangle)_{\varphi_{\Sigma_2}}\langle\langle\Sigma_1^*\rangle\rangle$ given by $((r,w_2),w_1) = ((r',w_1),w_2)$, $w_1 \in \Sigma_1^*$, $w_2 \in \Sigma_2^*$.

We prove only the compatibility of multiplication and star. Let $r_1, r_2 \in (A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)_{\psi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$ with corresponding $r_1', r_2' \in (A_\psi\langle\langle\Sigma_2^*\rangle\rangle)_{\varphi_{\Sigma_2}}\langle\langle\Sigma_1^*\rangle\rangle$, respectively. Then, for all $w_1 \in \Sigma_1^*$ and $w_2 \in \Sigma_2^*$, we obtain

$$
\begin{aligned}
&((r_1'r_2', w_1), w_2) = (\textstyle\sum_{v_1v_2=w_1}(r_1', v_1) \odot_{\varphi_{\Sigma_2}} (r_2', v_2), w_2) = \\
&\textstyle\sum_{v_1v_2=w_1}((r_1', v_1)\varphi_{\Sigma_2}^{|v_1|}(r_2', v_2), w_2) = \\
&\textstyle\sum_{v_1v_2=w_1}\sum_{u_1u_2=w_2}((r_1', v_1), u_1)\psi^{|u_1|}(\varphi_{\Sigma_2}^{|v_1|}(r_2', v_2), u_2) = \\
&\textstyle\sum_{v_1v_2=w_1}\sum_{u_1u_2=w_2}((r_1', v_1), u_1)\psi^{|u_1|}(\varphi^{|v_1|}((r_2', v_2), u_2)) = \\
&\textstyle\sum_{u_1u_2=w_2}\sum_{v_1v_2=w_1}((r_1, u_1), v_1)\varphi^{|v_1|}(\psi^{|u_1|}((r_2, u_2), v_2)) = \\
&\textstyle\sum_{u_1u_2=w_2}\sum_{v_1v_2=w_1}((r_1, u_1), v_1)\varphi^{|v_1|}(\psi_{\Sigma_1}^{|u_1|}(r_2, u_2), v_2) = \\
&\textstyle\sum_{u_1u_2=w_2}((r_1, u_1)\psi_{\Sigma_1}^{|u_1|}(r_2, u_2), w_1) = ((r_1r_2, w_2), w_1)\,.
\end{aligned}
$$

Let now $r \in (A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)_{\psi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$ correspond to $r' \in (A_\psi\langle\langle\Sigma_2^*\rangle\rangle)_{\varphi_{\Sigma_2}}\langle\langle\Sigma_1^*\rangle\rangle$. We show by induction on $|w_1| + |w_2|$, $w_1 \in \Sigma_1^*$, $w_2 \in \Sigma_2^*$, that $((r^*, w_2), w_1) = ((r'^*, w_1), w_2)$. The case $|w_1| + |w_2| = 0$, i.e., $w_1 = \varepsilon$, $w_2 = \varepsilon$, is clear. Assume now $|w_1| + |w_2| > 0$. Then we consider the three cases (i) $w_1 = \varepsilon$, $w_2 \neq \varepsilon$, (ii) $w_1 \neq \varepsilon$, $w_2 = \varepsilon$ and (iii) $w_1 \neq \varepsilon$, $w_2 \neq \varepsilon$.

(i) We obtain

$$
\begin{aligned}
&((r^*, w_2), \varepsilon) = (\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}(r^*, \varepsilon)(r, u_1)\psi_{\Sigma_1}^{|u_1|}(r^*, u_2), \varepsilon) = \\
&\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}((r^*, \varepsilon), \varepsilon)((r, u_1), \varepsilon)\psi^{|u_1|}((r^*, u_2), \varepsilon)
\end{aligned}
$$

and

$$
\begin{aligned}
&((r'^*, \varepsilon), w_2) = ((r', \varepsilon)^*, w_2) = \\
&\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}((r', \varepsilon), \varepsilon)^*((r', \varepsilon), u_1)\psi^{|u_1|}((r', \varepsilon)^*, u_2) = \\
&\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}((r^*, \varepsilon), \varepsilon)((r, u_1), \varepsilon)\psi^{|u_1|}((r^*, u_2), \varepsilon)\,.
\end{aligned}
$$

(ii) By the substitution $w_1 \leftrightarrow w_2$, $\varphi \leftrightarrow \psi$, $r \leftrightarrow r'$, $\Sigma_1 \leftrightarrow \Sigma_2$, the proof of the equality $((r^*, \varepsilon), w_1) = ((r'^*, w_1), \varepsilon)$ is symmetric to the proof of (i).

(iii) We obtain

$$
\begin{aligned}
&((r^*, w_2), w_1) = (\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}(r, \varepsilon)^*(r, u_1)\psi_{\Sigma_1}^{|u_1|}(r^*, u_2), w_1) = \\
&\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}\sum_{v_1v_2v_3=w_1} \\
&\qquad((r^*, \varepsilon), v_1)\varphi^{|v_1|}((r, u_1), v_2)\varphi^{|v_1v_2|}(\psi^{|u_1|}((r^*, u_2), v_3)) = \\
&\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}\sum_{v_1v_2v_3=w_1,\ v_1\neq\varepsilon} \\
&\qquad((r^*, \varepsilon), v_1)\varphi^{|v_1|}((r, u_1), v_2)\varphi^{|v_1v_2|}(\psi^{|u_1|}((r^*, u_2), v_3)) + \\
&\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}\sum_{v_2v_3=w_1,\ v_2\neq\varepsilon}((r^*, \varepsilon), \varepsilon)((r, u_1), v_2)\varphi^{|v_2|}(\psi^{|u_1|}((r^*, u_2), v_3)) + \\
&\textstyle\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}((r^*, \varepsilon), \varepsilon)((r, u_1), \varepsilon)\psi^{|u_1|}((r^*, u_2), w_1)\,.
\end{aligned}
$$

We call the first, second and third of these terms L_1, L_2 and L_3, respectively.

Moreover, we obtain

$$((r'^*, w_1), w_2) = \sum_{x_1x_2=w_1,\ x_1\neq\varepsilon}((r'^*, \varepsilon)(r', x_1)\varphi_{\Sigma_2}^{|x_1|}(r'^*, x_2), w_2) =$$
$$\sum_{x_1x_2=w_1,\ x_1\neq\varepsilon}\sum_{y_1y_2y_3=w_2}$$
$$((r^*, y_1), \varepsilon)\psi^{|y_1|}((r, y_2), x_1)\varphi^{|x_1|}(\psi^{|y_1y_2|}((r^*, y_3), x_2)) =$$
$$\sum_{x_1x_2=w_1,\ x_1\neq\varepsilon}\sum_{y_1y_2y_3=w_2,\ y_1\neq\varepsilon}$$
$$((r^*, y_1), \varepsilon)\psi^{|y_1|}((r, y_2), x_1)\varphi^{|x_1|}(\psi^{|y_1y_2|}((r^*, y_3), x_2)) +$$
$$\sum_{x_1x_2=w_1,\ x_1\neq\varepsilon}\sum_{y_2y_3=w_2,\ y_2\neq\varepsilon}((r^*, \varepsilon), \varepsilon)((r, y_2), x_1)\varphi^{|x_1|}(\psi^{|y_2|}((r^*, y_3), x_2)) +$$
$$\sum_{x_1x_2=w_1,\ x_1\neq\varepsilon}((r^*, \varepsilon), \varepsilon)((r, \varepsilon), x_1)\varphi^{|x_1|}((r^*, w_2), x_2).$$

We call the first, second and third of these terms R_1, R_2 and R_3, respectively. It is clear that $L_2 = R_2$. We will prove that $L_3 = R_1$ and $L_1 = R_3$. We obtain

$$L_3 = \sum_{u_1u_2=w_2,\ u_1\neq\varepsilon,\ u_2\neq\varepsilon}\sum_{t_1t_2=w_1,\ t_1\neq\varepsilon}$$
$$((r^*, \varepsilon), \varepsilon)((r, u_1), \varepsilon)\psi^{|u_1|}((r'^*, \varepsilon)(r', t_1)\varphi_{\Sigma_2}^{|t_1|}(r'^*, t_2), u_2) +$$
$$((r^*, \varepsilon), \varepsilon)((r, w_2), \varepsilon)\psi^{|w_2|}((r^*, \varepsilon), w_1) =$$
$$\sum_{u_1u_2=w_2,\ u_1\neq\varepsilon,\ u_2\neq\varepsilon}\sum_{t_1t_2=w_1,\ t_1\neq\varepsilon}\sum_{s_1s_2s_3=u_2}$$
$$((r^*, \varepsilon), \varepsilon)((r, u_1), \varepsilon)\psi^{|u_1|}(((r^*, s_1), \varepsilon)\psi^{|s_1|}((r, s_2), t_1)\varphi^{|t_1|}(\psi^{|s_1s_2|}((r^*, s_3), t_2))) +$$
$$\sum_{t_1t_2=w_1,\ t_1\neq\varepsilon}((r^*, \varepsilon), \varepsilon)((r, w_2), \varepsilon)\psi^{|w_2|}(((r^*, \varepsilon), \varepsilon)((r, \varepsilon), t_1)\varphi^{|t_1|}((r^*, \varepsilon), t_2)) =$$
$$\sum_{u_1s_1s_2s_3=w_2,\ u_1\neq\varepsilon}\sum_{t_1t_2=w_1,\ t_1\neq\varepsilon}$$
$$((r^*, \varepsilon), \varepsilon)((r, u_1), \varepsilon)\psi^{|u_1|}((r^*, s_1), \varepsilon)\psi^{|u_1s_1|}((r, s_2), t_1)\varphi^{|t_1|}(\psi^{|u_1s_1s_2|}((r^*, s_3), t_2))$$

and

$$R_1 = \sum_{x_1x_2=w_1,\ x_1\neq\varepsilon}\sum_{p_1p_2y_2y_3=w_2,\ p_1\neq\varepsilon}$$
$$((r^*, \varepsilon), \varepsilon)((r, p_1), \varepsilon)\psi^{|p_1|}((r^*, p_2), \varepsilon)\psi^{|p_1p_2|}((r, y_2), x_1)\varphi^{|x_1|}(\psi^{|p_1p_2y_2|}((r^*, y_3), x_2)).$$

Hence, $L_3 = R_1$.

We now write L_1 and R_3 in an other form, using the isomorphism of the induction hypothesis. Then we obtain

$$L_1 = \sum_{u_1u_2=w_2,\ u_1\neq\varepsilon}\sum_{v_1v_2v_3=w_1,\ v_1\neq\varepsilon}$$
$$((r'^*, v_1), \varepsilon)\varphi^{|v_1|}((r', v_2), u_1)\varphi^{|v_1v_2|}(\psi^{|u_1|}((r'^*, v_3), u_2))$$

and

$$R_3 = \sum_{x_1x_2=w_1,\ x_1\neq\varepsilon}((r'^*, \varepsilon), \varepsilon)((r', x_1), \varepsilon)\varphi^{|x_1|}((r'^*, x_2), w_2).$$

By the substitution $x_1 \leftrightarrow u_1$, $x_2 \leftrightarrow u_2$, $v_1 \leftrightarrow y_1$, $v_2 \leftrightarrow y_2$, $v_3 \leftrightarrow y_3$, $w_1 \leftrightarrow w_2$, $\phi \leftrightarrow \psi$, $r \leftrightarrow r'$, $\Sigma_1 \leftrightarrow \Sigma_2$, $L \leftrightarrow R$, the proof of the equality $L_1 = R_3$ is symmetric to the proof of the equality $R_1 = L_3$. □

Corollary 14. *Let A be a Conway semiring, $\varphi : A \to A$ be an endomorphism and Σ_1, Σ_2 be alphabets. Then $(A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)_{\varphi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$, $(A\langle\langle\Sigma_1^*\rangle\rangle)_{\varphi_{\Sigma_1}}\langle\langle\Sigma_2^*\rangle\rangle$, $(A_\varphi\langle\langle\Sigma_1^*\rangle\rangle)\langle\langle\Sigma_2^*\rangle\rangle$ and $(A\langle\langle\Sigma_1^*\rangle\rangle)\langle\langle\Sigma_2^*\rangle\rangle$, and $(A_\varphi\langle\langle\Sigma_2^*\rangle\rangle)_{\varphi_{\Sigma_2}}\langle\langle\Sigma_1^*\rangle\rangle$, $(A_\varphi\langle\langle\Sigma_2^*\rangle\rangle)\langle\langle\Sigma_1^*\rangle\rangle$, $(A\langle\langle\Sigma_2^*\rangle\rangle)_{\varphi_{\Sigma_2}}\langle\langle\Sigma_1^*\rangle\rangle$ and $(A\langle\langle\Sigma_2^*\rangle\rangle)\langle\langle\Sigma_1^*\rangle\rangle$ are isomorphic starsemirings, respectively.*

3 Finite automata and Kleene Theorems over Conway semiring-semimodule pairs

In this section we consider finite automata over semirings and quemirings and prove some Kleene Theorems.

By $\langle A_\varphi\langle\langle\Sigma^\omega\rangle\rangle, +, 0\rangle$ we denote the set of skew power series $\sum_{v\in\Sigma^\omega}(s,v)v$, $(s,v)\in A$, with pointwise addition. We define a (left) action $\otimes_\varphi : A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle \to A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$, $(r,s)\mapsto r\otimes_\varphi s$, by

$$(r\otimes_\varphi s, v) = \sum_{w\in\Sigma^*,\ u\in\Sigma^\omega,\ wu=v} (r,w)\varphi^{|w|}(s,u), \qquad v\in\Sigma^\omega .$$

Theorem 15. *Let A be a complete semiring, $\varphi : A \to A$ be an endomorphism of complete semirings and Σ be an alphabet. Then $A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ is a (left) $A_\varphi\langle\langle\Sigma^*\rangle\rangle$-semimodule.*

Throughout this section, A is a *Conway semiring*, such that $(A_\varphi\langle\langle\Sigma^*\rangle\rangle, A_\varphi\langle\langle\Sigma^\omega\rangle\rangle)$ is a starsemiring-omegasemimodule pair (see Elgot [8], Ésik, Kuich [9]). Moreover, we assume $0^\omega = 0$. Furthermore, we use the notation $A_\varphi\langle\Sigma\cup\varepsilon\rangle = \{a\varepsilon + \sum_{x\in\Sigma} a_x x \mid a, a_x \in A\}$, $A_\varphi\langle\Sigma\rangle = \{\sum_{x\in\Sigma} a_x x \mid a_x\in A\}$, $A_\varphi\langle\varepsilon\rangle = \{a\varepsilon \mid a\in A\}$.

A *finite automaton over the semiring* $A_\varphi\langle\langle\Sigma^*\rangle\rangle$

$$\mathfrak{A} = (n, I, M, P)$$

is given by

(i) a finite *set of states* $\{1,\dots,n\}$, $n\geq 1$,

(ii) a *transition matrix* $M\in(A_\varphi\langle\Sigma\cup\varepsilon\rangle)^{n\times n}$,

(iii) an *initial state vector* $I\in(A_\varphi\langle\varepsilon\rangle)^{1\times n}$,

(iv) a *final state vector* $P\in(A_\varphi\langle\varepsilon\rangle)^{n\times 1}$.

The *behavior* of $\mathfrak{A}$ is a skew power series in $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ and is defined by

$$\|\mathfrak{A}\| = IM^*P .$$

(See Conway [4], Bloom, Ésik [2], Kuich, Salomaa [14].)

A finite automaton over the quemiring $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$

$$\mathfrak{A} = (n, I, M, P, k)$$

is given by

(i) a *finite automaton* (n, I, M, P) *over* $A_\varphi\langle\langle\Sigma^*\rangle\rangle$,

(ii) a *set of repeated states* $\{1,\dots,k\}$, $0\leq k\leq n$.

The behavior of $\mathfrak{A}$ is a pair of skew power series in $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ and is defined by

$$||\mathfrak{A}|| = IM^*P + IM^{\omega_k}.$$

(See Bloom, Ésik [2], Ésik, Kuich [11].)

Observe that, if $\mathfrak{A} = (n, I, M, P)$ is a finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ and $\mathfrak{A}' = (n, I, M, P, 0)$ is a finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ without repeated states, then $||\mathfrak{A}'|| = ||\mathfrak{A}||$.

A finite automaton $\mathfrak{A} = (n, I, M, P)$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ or $\mathfrak{A}' = (n, I, M, P, k)$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ is called *ε-free* if the entries of M are in $A_\varphi\langle\Sigma\rangle$.

A subsemiring of $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is *rationally closed* if it is closed under the operations $+, \cdot, ^*$. A subquemiring of the generalized starquemiring $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ is *ω-rationally closed* if it is closed under the operations $+, \cdot, \P, ^\otimes$. By definition, $A_\varphi^{\mathrm{rat}}\langle\langle\Sigma^*\rangle\rangle$ (resp. ω-$\mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$) is the *smallest* rationally (resp. ω-rationally) closed semiring (resp. quemiring) that contains $A_\varphi\langle\Sigma \cup \varepsilon\rangle$.

Since A is a Conway semiring, we can specialize the Kleene Theorem (Theorem 3.10) of Ésik, Kuich [11].

Theorem 16. *Let $(A_\varphi\langle\langle\Sigma^*\rangle\rangle, A_\varphi\langle\langle\Sigma^\omega\rangle\rangle)$ be a starsemiring-omegasemimodule pair, where A is a Conway semiring and $0^\omega = 0$. Then the following statements are equivalent for $(r, s) \in A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$:*

(i) $(r, s) = ||\mathfrak{A}||$, *where $\mathfrak{A}$ is a finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$,*

(ii) $(r, s) \in \omega\text{-Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$,

(iii) $r \in A_\varphi^{\mathrm{rat}}\langle\langle\Sigma^*\rangle\rangle$, $s = \sum_{1\le j\le m} u_j v_j^\omega$ *with* $u_j, v_j \in A_\varphi^{\mathrm{rat}}\langle\langle\Sigma^*\rangle\rangle$.

Proof. By Theorem 3.10 of Ésik, Kuich [11] and by Corollary 5. □

Moreover, Conway [4], Bloom, Ésik [2], or Aleshnikov, Boltnev, Ésik, Ishanov, Kuich, Malachowskij [1] imply at once the following generalization of the Kleene-Schützenberger Theorem.

Theorem 17. *Let A be a Conway semiring. Then the following statements are equivalent for $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$:*

(i) $r = ||\mathfrak{A}||$, *where $\mathfrak{A}$ is a finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$,*

(ii) $r = ||\mathfrak{A}||$, *where $\mathfrak{A}$ is an ε-free finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$,*

(iii) $r \in A_\varphi^{\mathrm{rat}}\langle\langle\Sigma^*\rangle\rangle$.

Proof. Theorems 3.2 and 3.3 of Aleshnikov, Boltnev, Ésik, Ishanov, Kuich, Malachowskij [1]. □

This theorem can also be seen to be a specialization of Theorem 16 for finite automata over $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ with empty repeated states set.

4 Cycle-free finite automata and a Kleene Theorem over complete semiring-semimodule pairs

We first prove that, for a complete star-omega semiring A and an endomorphism $\varphi : A \to A$ compatible with infinite sums and products, $(A_\varphi\langle\langle\Sigma^*\rangle\rangle, A_\varphi\langle\langle\Sigma^\omega\rangle\rangle)$ is a complete semiring-semimodule pair.

Then, for a subsemiring A' of A, such that, for any cycle-free $q \in A'\langle\Sigma \cup \varepsilon\rangle$, q^ω is in $A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$, we consider cycle-free finite automata over the quemiring $A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ and prove a Kleene Theorem.

We then show that the star-omega semiring $\mathbb{R}^\infty_{\max}$ is complete. This implies then the Kleene Theorem of Droste, Kuske [5].

Assume that A is a complete star-omega semiring, i.e., there exists an infinite product subject to three conditions appearing in the definition of a complete semiring-semimodule pair. Then we define an infinite product for skew power series in the following way:

$$(r_1, r_2, \dots) \mapsto \prod_{j\geq 1}^{\varphi} r_j \in A_\varphi\langle\langle\Sigma^\omega\rangle\rangle, \qquad r_j \in A_\varphi\langle\langle\Sigma^*\rangle\rangle,\ j \geq 1\,,$$

where, for all $v \in \Sigma^\omega$,

$$(\prod_{j\geq 1}^{\varphi} r_j, v) = \sum_{v=v_1v_2\dots} \prod_{j\geq 1} \varphi^{|v_1\dots v_{j-1}|}(r_j, v_j)\,.$$

Observe that now, for $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$,

$$r^\omega = \prod_{j\geq 1}^{\varphi} r\,.$$

Theorem 18. *Let A be a complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Then $(A_\varphi\langle\langle\Sigma^*\rangle\rangle, A_\varphi\langle\langle\Sigma^\omega\rangle\rangle)$ is a complete semiring-semimodule pair satisfying $(a\varepsilon)^\omega = 0$ for $a \in A$.*

Proof. We only prove the equation

$$\prod_{j\geq 1}^{\varphi} (\sum_{i_j\in I_j} r_j) = \sum_{(i_1,i_2,\dots)\in I_1\times I_2\times\dots} \prod_{j\geq 1}^{\varphi} r_j, \qquad r_j \in A_\varphi\langle\langle\Sigma^*\rangle\rangle,\ j \geq 1\,.$$

We obtain, for $v \in \Sigma^\omega$,

$$\begin{aligned}
&(\textstyle\prod_{j\geq 1}^{\varphi}(\sum_{i_j\in I_j} r_j), v) = \\
&\textstyle\sum_{v=v_1v_2\dots} \prod_{j\geq 1} \varphi^{|v_1\dots v_{j-1}|}(\sum_{i_j\in I_j}(r_j, v_j)) = \\
&\textstyle\sum_{v=v_1v_2\dots} \sum_{(i_1,i_2,\dots)\in I_1\times I_2\times\dots} \prod_{j\geq 1} \varphi^{|v_1\dots v_{j-1}|}(r_j, v_j) = \\
&\textstyle\sum_{(i_1,i_2,\dots)\in I_1\times I_2\times\dots} \sum_{v=v_1v_2\dots} \prod_{j\geq 1} \varphi^{|v_1\dots v_{j-1}|}(r_j, v_j) = \\
&\textstyle\sum_{(i_1,i_2,\dots)\in I_1\times I_2\times\dots} (\prod_{j\geq 1}^{\varphi} r_j, v) = \\
&\textstyle(\sum_{(i_1,i_2,\dots)\in I_1\times I_2\times\dots} \prod_{j\geq 1}^{\varphi} r_j, v)\,.
\end{aligned}$$

Consider now, for $a \in A$, $v \in \Sigma^\omega$, $(\prod^{\varphi}_{j\geq 1} a\varepsilon, v) = \sum_{v=v_1v_2\ldots} \prod_{j\geq 1} \varphi^{|v_1\ldots v_{j-1}|}(a\varepsilon, v_j)$. Then infinitely many of the v_j are unequal to ε. Hence, $(a\varepsilon, v_j) = 0$ for infinitely many j and $(\prod^{\varphi}_{j\geq 1} a\varepsilon, v) = 0$. □

In the sequel, we often denote $\otimes_\varphi$ simply by $\cdot$ or concatenation.

Corollary 19. *Let A be a complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Then $(A_\varphi\langle\langle\Sigma^*\rangle\rangle, A_\varphi\langle\langle\Sigma^\omega\rangle\rangle)$ is a Conway semiring-semimodule pair satisfying $(a\varepsilon)^\omega = 0$ for $a \in A$.*

Proof. By Theorem 3.1 of Ésik, Kuich [9]. □

Corollary 20. *Let A be a complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Then, for $n \geq 1$, $((A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}, (A_\varphi\langle\langle\Sigma^\omega\rangle\rangle)^n)$ is a complete semiring-semimodule pair satisfying $(M\varepsilon)^\omega = 0$ for $M \in A^{n\times n}$.*

Proof. By Ésik, Kuich [9] and an easy proof by induction on n. □

Corollary 21. *Let A be a complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Then the following statements are equivalent for $(r, s) \in A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$:*

(i) $(r, s) = ||\mathfrak{A}||$, *where $\mathfrak{A}$ is a finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$,*

(ii) $(r, s) \in \omega\text{-Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$,

(iii) $r \in A^{\mathrm{rat}}_\varphi\langle\langle\Sigma^*\rangle\rangle$, $s = \sum_{1\leq j\leq m} u_j v_j^\omega$ *with* $u_j, v_j \in A^{\mathrm{rat}}_\varphi\langle\langle\Sigma^*\rangle\rangle$.

(iv) $(r, s) = ||\mathfrak{A}||$, *where $\mathfrak{A}$ is an ε-free finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$.*

(v) $(r, s) \in \hat{\omega}\text{-Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$,

(vi) $r \in A^{\mathrm{rat}}_\varphi\langle\langle\Sigma^*\rangle\rangle$, $s = \sum_{1\leq j\leq m} u_j v_j^\omega$ *with* $u_j, v_j \in A^{\mathrm{rat}}_\varphi\langle\langle\Sigma^*\rangle\rangle$ *where* $(u_j, \varepsilon) = 0$, $(v_j, \varepsilon) = 0$.

Proof. Since $(A_\varphi\langle\langle\Sigma^*\rangle\rangle, A_\varphi\langle\langle\Sigma^\omega\rangle\rangle)$ is a complete semiring-semimodule pair, it is also a Conway semiring-semimodule pair by Corollary 19. Moreover, $(a\varepsilon)^\omega = 0$ for $a \in A$. Hence, the corollary is implied by Theorems 16 and 17. □

A semiring A is called *zerosumfree* if, for all $a_1, a_2 \in A$, $a_1 + a_2 = 0$ implies $a_1 = 0$ and $a_2 = 0$. A semiring A is called *positive* if A is zerosumfree and if, for all $a_1, a_2 \in A$, whenever $s_1 \cdot s_2 = 0$ then $s_1 = 0$ or $s_2 = 0$ (see Eilenberg [7]). An element $a \in A$ is called *nilpotent* if there exists a $k \geq 1$ such that $a^k = 0$. The following lemma is from Ésik, Kuich [10].

Lemma 22. *(i) Let A be a complete positive semiring. Assume that*

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in A^{n\times n}, \text{ where } a \in A^{1\times 1},\ d \in A^{(n-1)\times(n-1)}.$$

If M is nilpotent then $a + bd^*c = 0$.

(ii) Let A be a zerosumfree semiring. Assume that

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in A^{n\times n}, \text{ where } a \in A^{n_1\times n_1},\ d \in A^{n_2\times n_2},\ n_1 + n_2 = n.$$

If M is nilpotent then a, d, bc and cb are nilpotent.

A skew power series $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is called *cycle-free* if there exists a $k \geq 1$ such that $(r,\varepsilon)^k = 0$, i. e., if (r,ε) is nilpotent. A finite automaton $\mathfrak{A} = (n, I, M, P)$ (resp. $\mathfrak{A} = (n, I, M, P, k)$) over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ (resp. $A_\varphi\langle\langle\Sigma^*\rangle\rangle \times A_\varphi\langle\langle\Sigma^\omega\rangle\rangle$) is called *cycle-free* if M is cycle-free.

For the rest of this section, A is a complete star-omega semiring and $\varphi : A \to A$ is an endomorphism compatible with infinite sums and products.

Theorem 23. *Let A be a positive complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Let A' be a subsemiring of A such that, for any cycle-free $q \in A'_\varphi\langle\Sigma\cup\varepsilon\rangle$, $q^\omega \in A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. Assume that $M \in (A'_\varphi\langle\Sigma\cup\varepsilon\rangle)^{n\times n}$ is cycle-free. Then $M^\omega \in (A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle)^n$.*

Proof. The proof is by induction on n. The case $n = 1$ is clear. Assume now that $n > 1$ and partition M as usual into blocks a, b, c, d, where $a \in A'_\varphi\langle\Sigma\cup\varepsilon\rangle$ and $d \in (A'_\varphi\langle\Sigma\cup\varepsilon\rangle)^{(n-1)\times(n-1)}$. Consider $(M^\omega)_1 = (a + bd^*c)^\omega + (a + bd^*c)^*bd^\omega$. By Lemma 22, $(a + bd^*c, \varepsilon) = 0$ and d is cycle-free. Hence, $(a + bd^*c)^\omega \in A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ and $d^\omega \in (A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle)^{n-1}$. Moreover, $(a + bd^*c)^* \in A'_\varphi\langle\langle\Sigma^*\rangle\rangle$. This implies that $(M^\omega)_1 \in A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. By application of the omega-permutation-equation (see Bloom, Ésik [2]) we obtain that $M^\omega \in (A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle)^n$. □

By definition, $\mathfrak{Rat}(A_\varphi\langle\Sigma\cup\varepsilon\rangle) \subseteq A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is the smallest semiring containing $A_\varphi\langle\Sigma\cup\varepsilon\rangle$ such that, for $q \in \mathfrak{Rat}(A_\varphi\langle\Sigma\cup\varepsilon\rangle)$ where $(q,\varepsilon) = 0$, q^* is again in $\mathfrak{Rat}(A_\varphi\langle\Sigma\cup\varepsilon\rangle)$.

Theorem 24. *Let A be a positive complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Let A' be a subsemiring of A such that, for any cycle-free $q \in A'_\varphi\langle\Sigma\cup\varepsilon\rangle$, $q^\omega \in A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. Assume that $M \in (A'_\varphi\langle\Sigma\cup\varepsilon\rangle)^{n\times n}$ is cycle-free. Then, for $1 \leq i \leq n$, $1 \leq j \leq m$, there exist $u_{ij}, v_{ij} \in \mathfrak{Rat}(A_\varphi\langle\Sigma\cup\varepsilon\rangle)$, where $(u_{ij},\varepsilon) = 0$, $(v_{ij},\varepsilon) = 0$, such that $(M^\omega)_i = \sum_{1\leq j\leq m} u_{ij}v_{ij}^\omega$.*

Proof. The proof is by induction on n. The case $n = 1$ is clear. Assume now that $n > 1$ and partition M as usual into blocks a, b, c, d, where $a \in A'_\varphi\langle\Sigma\cup\varepsilon\rangle$ and $d \in (A'_\varphi\langle\Sigma\cup\varepsilon\rangle)^{(n-1)\times(n-1)}$. The entries of $a + bd^*c$, $(a + bd^*c)^*b$ and d

are in $\mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$. Hence, by Lemma 22 , there exist $t \in \mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$, $u \in (\mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle))^{1\times(n-1)}$, where $(t,\varepsilon) = 0$, such that $(M^\omega)_1 = t^\omega + ud^\omega = t^\omega + u(d^k)^\omega = t^\omega + ud^k(d^k)^\omega$ for all $k \geq 1$. Here the second equality follows by Corollaries 4.3 and 4.2. Since d is cycle-free there exists a $k \geq 1$ such that $(d^k,\varepsilon) = 0$. Let now $(ud^k)_i = u_i$, $(d^k)_i^\omega = v_i$. By induction hypothesis, $v_i = \sum_{1\leq j\leq m} u'_{ij}v'^\omega_{ij}$, where $(u'_{ij},\varepsilon) = 0$, $(v'_{ij},\varepsilon) = 0$. Then $(M^\omega)_1 = t^\omega + \sum_{1\leq i\leq n}\sum_{1\leq j\leq m} u_iu'_{ij}v'^\omega_{ij}$, where $(t,\varepsilon) = 0$, $(u_i,\varepsilon) = 0$, $(u'_{ij},\varepsilon) = 0$, $(v'_{ij},\varepsilon) = 0$. The omega-permutation-equation proves the theorem for $(M^\omega)_i$, $2 \leq i \leq n$. □

Theorem 25. *Let A be a complete semiring and A' be a subsemiring of A. Let $\mathfrak{A} = (n, I, M, P)$ be a cycle-free finite automaton over the semiring $A'_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then $||\mathfrak{A}|| \in A'_\varphi\langle\langle\Sigma^*\rangle\rangle$.*

Proof. Since $\mathfrak{A}$ is cycle-free, $(M,\varepsilon)^* \in A'^{n\times n}$. Let $M_1 = \sum_{x\in\Sigma}(M,x)x$. Then, since $((M,\varepsilon)^*M_1,\varepsilon) = 0$,

$$M^* = ((M,\varepsilon)^*M_1)^*(M,\varepsilon)^* \in (A'_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}.$$

(Here we have applied already the forthcoming Theorem 38.) Hence, $||\mathfrak{A}|| \in A'_\varphi\langle\langle\Sigma^*\rangle\rangle$. □

Theorem 26. *Let A be a positive complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Let A' be a subsemiring of A such that, for any cycle-free $q \in A'_\varphi\langle\Sigma \cup \varepsilon\rangle$, $q^\omega \in A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. Let $\mathfrak{A} = (n, I, M, P, k)$ be a cycle-free finite automaton over the quemiring $A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. Then $||\mathfrak{A}|| \in A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$.*

Proof. By the proof of Theorem 25, $M^* \in (A'_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$. By Theorem 23, $M^\omega \in (A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle)^n$. Hence, $||\mathfrak{A}|| \in A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. □

Theorem 27. *Let A be a positive complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Let A' be a subsemiring of A such that, for any cycle-free $q \in A'_\varphi\langle\Sigma\cup\varepsilon\rangle$, $q^\omega \in A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. Then the behaviors of cycle-free finite automata over $A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ form a subquemiring $\hat{T}_\varphi$ of $A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ containing $A'_\varphi\langle\Sigma\cup\varepsilon\rangle$, such that for $r \in \hat{T}_\varphi$, where $(r\P,\varepsilon) = 0$, $r^\otimes$ is again in $\hat{T}_\varphi$.*

Proof. Inspection of the proofs of Theorems 3.3–3.8 of Ésik, Kuich [11] shows that all constructed finite automata are again cycle-free. This is seen by the proofs of Lemmas 3.15–3.17 of Ésik, Kuich [10]. Hence, Theorem 26 proves our theorem. □

Theorem 28. *Let A be a positive complete star-omega semiring, $\varphi : A \to A$ be an endomorphism compatible with infinite sums and products and Σ be an alphabet. Let A' be a subsemiring of A such that, for any cycle-free $q \in A'_\varphi\langle\Sigma\cup\varepsilon\rangle$, $q^\omega \in A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$. Then the following statements are equivalent for $(r,s) \in A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$:*

(i) $(r,s) = ||\mathfrak{A}||$, where $\mathfrak{A}$ is a cycle-free finite automaton over $A'_\varphi\langle\langle\Sigma^*\rangle\rangle \times A'_\varphi\langle\langle\Sigma^\omega\rangle\rangle$,

(ii) $(r,s) \in \hat{\omega}\text{-}\mathfrak{Rat}(A'_\varphi\langle\Sigma\cup\varepsilon\rangle)$,

(iii) $r \in \mathfrak{Rat}(A_\varphi\langle\Sigma\cup\varepsilon\rangle)$ *and* $s = \sum_{1\le i\le m} u_i v_i^\omega$ *with* $u_i, v_i \in \mathfrak{Rat}(A_\varphi\langle\Sigma\cup\varepsilon\rangle)$ *and* $(u_i,\varepsilon)=0$, $(v_i,\varepsilon)=0$.

Proof. (i) $\Rightarrow$ (iii): By Theorems 24 and 25.

(iii) $\Rightarrow$ (ii): Since $r \in \mathfrak{Rat}(A_\varphi\langle\Sigma\cup\varepsilon\rangle)$ and $s \in \hat{\omega}\text{-}\mathfrak{Rat}(A'_\varphi\langle\Sigma\cup\varepsilon\rangle.0$, we obtain $(r,s) \in \hat{\omega}\text{-}\mathfrak{Rat}(A'_\varphi\langle\Sigma\cup\varepsilon\rangle$.

(ii) $\Rightarrow$ (i): By Theorem 27. □

We now want to prove the Kleene Theorem of Droste, Kuske [5]. We first consider the complete semiring

$$\mathbb{R}^\infty_{\max} = \langle\{a \ge 0 \mid a \in \mathbb{R}\} \cup \{-\infty,\infty\}, \max, +, -\infty, 0\rangle .$$

Here the operations are as usual, with $-\infty+\infty=-\infty$, *infinite sums* are defined by $\sum'_{i\in I} a_i = \sup\{a_i \mid i \in I\}$ and *infinite products* are defined by $\prod'_{i\ge1} a_i = \sum_{i\ge1} a_i$. Here $\sum_{i\ge1} a_i$ denotes $\sup\{\sum_{1\le i\le n} a_i \mid n \ge 1\}$. We now show that this infinite product satisfies the three laws of a complete star-omega semiring.

(i) Let $a_i \ge 0$ and $0 = n_0 \le n_1 \le n_2 \le \dots$ and define $b_i = a_{n_{i-1}+1} \dots a_{n_i} = \sum_{n_{i-1}+1\le j\le n_i} a_j$, $i \ge 1$. We have to show that $\prod'_{i\ge1} a_i = \prod'_{i\ge1} b_i$. We obtain $\prod'_{i\ge1} b_i = \sum_{i\ge1} b_i = \sum_{i\ge1}\sum_{n_{i-1}+1\le j\le n_i} a_j = \sum_{i\ge1} a_i = \prod'_{i\ge1} a_i$.

(ii) Let $a_i \ge 0$, $i \ge 1$. Then we obtain $a_1 + \prod'_{i\ge1} a_{i+1} = a_1 + \sum_{i\ge1} a_{i+1} = \sum_{i\ge1} a_i = \prod'_{i\ge1} a_i$.

(iii) Let $a_{i_j} \ge 0$, $i_j \in I_j$, $j \ge 1$. Then we have to show that $\prod'_{j\ge1}\sum'_{i_j\in I_j} a_{i_j} = \sum'_{(i_1,i_2,\dots)\in I_1\times I_2\times\dots}\prod'_{j\ge1} a_{i_j}$. We obtain $\prod'_{j\ge1}\sum'_{i_j\in I_j} a_{i_j} = \sum_{j\ge1}\sup\{a_{i_j} \mid i_j \in I_j\} = \sup\{\sum_{j\ge1} a_{i_j} \mid (i_1,i_2,\dots)\in I_1\times I_2\times\dots\} = \sum'_{(i_1,i_2,\dots)\in I_1\times I_2\times\dots}\prod'_{j\ge1} a_{i_j}$.

Hence, we have proved the next theorem.

Theorem 29. $\mathbb{R}^\infty_{\max}$ *is a complete star-omega semiring.*

The only endomorphisms of $\mathbb{R}^\infty_{\max}$ are of the form $\varphi(a) = q\cdot a$ for some $q \in \mathbb{R}$, $q \ge 0$. (See Droste, Kuske [5], Lemma 5.1.) Denote $(\mathbb{R}^\infty_{\max})_\varphi\langle\langle\Sigma^*\rangle\rangle$ by $\mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^*\rangle\rangle$ and $(\mathbb{R}^\infty_{\max})_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ by $\mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^\omega\rangle\rangle$ if φ is defined as above, and observe that the multiplication $+_q$ in $\mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^*\rangle\rangle$ is defined by

$$(r_1 +_q r_2, w) = \max\{(r_1,w_1) + q^{|w_1|}(r_2,w_2) \mid w_1w_2 = w\},$$

where $r_1, r_2 \in \mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^*\rangle\rangle$, $w \in \Sigma^*$.

Corollary 30 (Ésik, Kuich [10]). $(\mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^*\rangle\rangle, \mathbb{R}^\infty_{\max,q}\langle\langle\Sigma^\omega\rangle\rangle)$ *is a complete semiring-semimodule pair.*

Let $\mathbb{R}_{\max}$ be the following subsemiring of $\mathbb{R}^\infty_{\max}$:

$$\mathbb{R}_{\max} = \langle\{a \ge 0 \mid a \in \mathbb{R}\} \cup \{-\infty\}, \max, +, -\infty, 0\rangle .$$

Denote $(\mathbb{R}_{\max})_\varphi\langle\langle\Sigma^*\rangle\rangle$ by $\mathbb{R}_{\max,q}\langle\langle\Sigma^*\rangle\rangle$ and $(\mathbb{R}_{\max})_\varphi\langle\langle\Sigma^\omega\rangle\rangle$ by $\mathbb{R}_{\max,q}\langle\langle\Sigma^\omega\rangle\rangle$.

Theorem 31 (Droste, Kuske [5]). *The following statements are equivalent for* $(r,s) \in \mathbb{R}_{\max,q}\langle\langle\Sigma^*\rangle\rangle \times \mathbb{R}_{\max,q}\langle\langle\Sigma^\omega\rangle\rangle$, $0 \le q < 1$*:*

(i) $(r,s) = ||\mathfrak{A}||$*, where* $\mathfrak{A}$ *is a cycle-free finite automaton over* $\mathbb{R}_{\max,q}\langle\langle\Sigma^*\rangle\rangle \times \mathbb{R}_{\max,q}\langle\langle\Sigma^\omega\rangle\rangle$,

(ii) $(r,s) \in \hat{\omega}\text{-}\mathfrak{Rat}(\mathbb{R}_{\max,q}\langle\Sigma \cup \varepsilon\rangle)$,

(iii) $r \in \mathfrak{Rat}(\mathbb{R}_{\max,q}\langle\Sigma \cup \varepsilon\rangle)$ *and* $s = \max\{u_i +_q v_i \mid 1 \le i \le m\}$ *with* $u_i, v_i \in \mathfrak{Rat}(\mathbb{R}_{\max,q}\langle\Sigma \cup \varepsilon\rangle)$ *and* $(u_i,\varepsilon) = -\infty$, $(v_i,\varepsilon) = -\infty$.

Proof. By Theorem 28. □

5 Skew power series over arbitrary semirings

We assume that the reader is familiar with the axiomatic theory of convergence considered in Section 2 of Kuich, Salomaa [14]. We also use the notations and isomorphisms used there.

In this section we define a convergence in the semiring $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. This is done mainly for the purpose to define the star of a cycle-free power series in $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. If A is a starsemiring, these considerations on a convergence are not necessary. Hence, we assume that A is *not* a starsemiring. (Or, if A is a starsemiring, we do not consider explicitly the star operation in A.) We then show variants of the sum-star-equation, the product-star-equation and the matrix-star-equation. Eventually, we prove a Kleene Theorem due to Droste, Kuske [5] by application of these equations.

By $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$ we denote the set of sequences in $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. We denote by o and η the sequences defined by $o(n) = 0$ and $\eta(n) = \varepsilon$, $n \ge 0$. For $\alpha_1, \alpha_2 \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$ we define $\alpha_1 + \alpha_2$ and $\alpha_1 \odot_\varphi \alpha_2$ in $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$ by $(\alpha_1 + \alpha_2)(n) = \alpha_1(n) + \alpha_2(n)$ and $(\alpha_1 \odot_\varphi \alpha_2)(n) = \alpha_1(n) \odot_\varphi \alpha_2(n)$, $n \ge 0$. For $\alpha \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$, $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, we define $r \odot_\varphi \alpha$ and $\alpha \odot_\varphi r$ in $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$ by $(r \odot_\varphi \alpha)(n) = r \odot_\varphi \alpha(n)$ and $(\alpha \odot_\varphi r)(n) = \alpha(n) \odot_\varphi r$, $n \ge 0$. Observe that $\langle(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}, +, \odot_\varphi, o, \eta\rangle$ is a semiring, the full Cartesian product of ω copies of the semiring $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. In the sequel, we often denote $\odot_\varphi$ by $\cdot$ or by concatenation.

Consider $\alpha \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$ and $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then $\alpha_r \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$ denotes the sequence defined by $\alpha_r(0) = r$, $\alpha_r(n+1) = \alpha(n)$, $n \ge 0$. Moreover, for a sequence $\beta \in A^{\mathbb{N}}$, $\varphi(\beta)$ is the sequence in A defined by $\varphi(\beta)(n) = \varphi(\beta(n))$, $n \ge 0$.

By $D_\varphi\langle\langle\Sigma^*\rangle\rangle \subseteq (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbb{N}}$ we denote the set of sequences $\alpha : \mathbb{N} \to A_\varphi\langle\langle\Sigma^*\rangle\rangle$ such that for all $w \in \Sigma^*$ there exists an $n_{\alpha,w} \ge 0$ with $(\alpha(n_{\alpha,w} + k), w) = (\alpha(n_{\alpha,w}), w)$ for all $k \ge 0$. Let D_d be the set of convergent sequences of the discrete convergence in A. Then $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ iff $(\alpha, w) \in D_d$ for all $w \in \Sigma^*$.

We now will show that $D_\varphi\langle\langle\Sigma^*\rangle\rangle$ is a set of convergent sequences. Hence, we have to prove that the following conditions are satisfied:

(D1) $\eta \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$,

(D2) (i) if $\alpha_1, \alpha_2 \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ then $\alpha_1 + \alpha_2 \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$,

(ii) if $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ and $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ then $r \odot_\varphi \alpha, \alpha \odot_\varphi r \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$,

(D3) if $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ and $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ then $\alpha_r \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$.

Lemma 32. *$D_\varphi\langle\langle\Sigma^*\rangle\rangle$ is a set of convergent sequences in $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{\mathbf{N}}$.*

Proof. We only prove (D2)(ii), i. e., we prove that for $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$, $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, the sequences $r \odot_\varphi \alpha$ and $\alpha \odot_\varphi r$ are again in $D_\varphi\langle\langle\Sigma^*\rangle\rangle$. We obtain, for all $w \in \Sigma^*$,

$$(r \odot_\varphi \alpha, w) = \sum_{w_1 w_2 = w} (r, w_1)\varphi^{|w_1|}(\alpha, w_2)$$

and

$$(\alpha \odot_\varphi r, w) = \sum_{w_1 w_2 = w} (\alpha, w_1)\varphi^{|w_1|}(r, w_2)\,.$$

Since $\varphi^{|w_1|}(\alpha, w_2)$ and (α, w_1) are in D_d, these sequences $r \odot_\varphi \alpha$ and $\alpha \odot_\varphi r$ are in $D_\varphi\langle\langle\Sigma^*\rangle\rangle$.

The rest of the proof is analogous to the proof of Lemma 2.10 of Kuich, Salomaa [14]. □

We now will show that the mapping $\lim : D_\varphi\langle\langle\Sigma^*\rangle\rangle \to A_\varphi\langle\langle\Sigma^*\rangle\rangle$ defined by $\lim \alpha = \sum_{w\in\Sigma^*} \lim_d(\alpha, w)w$, $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$, is a limit function on $D_\varphi\langle\langle\Sigma^*\rangle\rangle$. Here $\lim_d : D_d \to A$ is the limit function of the discrete convergence in A defined by $\lim_d \beta = \beta(n_\beta)$ if $\beta \in D_d$ with $\beta(n_\beta + k) = \beta(n_\beta)$ for all $k \geq 0$. Hence, we have to prove that the following conditions are satisfied:

(lim1) $\lim \eta = 1$,

(lim2) (i) if $\alpha_1, \alpha_2 \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ then $\lim(\alpha_1 + \alpha_2) = \lim \alpha_1 + \lim \alpha_2$,
(ii) if $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ and $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ then $\lim(r\alpha) = r \lim \alpha$ and $\lim(\alpha r) = (\lim \alpha)r$,

(lim3) if $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ and $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ then $\lim \alpha_r = \lim \alpha$.

Theorem 33. *The mapping* $\lim : D_\varphi\langle\langle\Sigma^*\rangle\rangle \to A_\varphi\langle\langle\Sigma^*\rangle\rangle$ *defined by* $\lim \alpha = \sum_{w\in\Sigma^*} \lim_d(\alpha, w)w$, *$\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$, is a limit function on $D_\varphi\langle\langle\Sigma^*\rangle\rangle$.*

Proof. We only prove (lim2)(ii). Let $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, $\alpha \in D_\varphi\langle\langle\Sigma^*\rangle\rangle$ and $w \in \Sigma^*$. Then

$$\begin{aligned}&(\lim r\alpha, w) = \lim_d(r\alpha, w) = \lim_d(\textstyle\sum_{w_1w_2=w}(r, w_1)\varphi^{|w_1|}(\alpha, w_2)) = \\ &\textstyle\sum_{w_1w_2=w}(r, w_1)\lim_d \varphi^{|w_1|}(\alpha, w_2) = \sum_{w_1w_2=w}(r, w_1)\varphi^{|w_1|}(\lim_d(\alpha, w_2)) = \\ &\textstyle\sum_{w_1w_2=w}(r, w_1)\varphi^{|w_1|}(\lim \alpha, w_2) = (r \lim \alpha, w)\end{aligned}$$

and

$$\begin{aligned}&(\lim \alpha r, w) = \lim_d(\alpha r, w) = \lim_d(\textstyle\sum_{w_1w_2=w}(\alpha, w_1)\varphi^{|w_1|}(r, w_2)) = \\ &\textstyle\sum_{w_1w_2=w}\lim_d(\alpha, w_1)\varphi^{|w_1|}(r, w_2) = \\ &\textstyle\sum_{w_1w_2=w}(\lim \alpha, w_1)\varphi^{|w_1|}(r, w_2) = ((\lim \alpha)r, w)\,.\end{aligned}$$

We now obtain

$$\lim(r\alpha) = \sum_{w\in\Sigma^*} \lim_d(r\alpha, w)w = \sum_{w\in\Sigma^*} (r \lim \alpha, w)w = r \lim \alpha$$

and

$$\lim(\alpha r) = \sum_{w\in\Sigma^*} \lim_d(\alpha r, w)w = \sum_{w\in\Sigma^*} ((\lim \alpha)r, w)w = (\lim \alpha)r\,.$$

The rest of the proof is analogous to the proof of Lemma 2.11 of Kuich, Salomaa [14]. □

We make now the following conventions throughout the rest of this paper: In A we use always the discrete convergence; in $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ we use always the convergence defined in Theorem 33; in $A^{n\times n}$ we use always the discrete convergence; and in $A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ (and isomorphically in $(A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$) we use always the convergence defined in Theorem 33.

If, for $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ the sequence $(\sum_{j=0}^n r^j)$ is in $D_\varphi\langle\langle\Sigma^*\rangle\rangle$ then we write $\lim_{n\to\infty}\sum_{j=0}^n r^j = r^*$ and call r^* the *star* of r.

Clearly, a skew power series $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is cycle-free iff $\lim_{n\to\infty}((r,\varepsilon),\varepsilon)^n = 0$. A proof analogous to the proof of Theorem 3.8 of Kuich, Salomaa [14] yields the next theorem.

Theorem 34. *If $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is cycle-free then there exists a $k \geq 1$ such that*

$$(r^{(n+1)k+j}, w) = 0$$

for all $w \in \Sigma^*$, $|w| = n$, and $j \geq 0$. Furthermore, r^* exists and

$$(r^*, w) = \sum_{j=0}^{(n+1)k-1} (r^j, w), \quad w \in \Sigma^*\,.$$

Corollary 35. *If $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is cycle-free then $\lim_{n\to\infty} r^n = 0$ and r^* exists. Moreover,*

$$r^* = \varepsilon + rr^* = \varepsilon + r^*r\,.$$

Proof. The second statement follows from Kuich, Salomaa [14], Theorem 2.3. □

Theorem 36. *Let $r, s \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then rs is cycle-free iff sr is cycle-free and, in this case,*

$$s(rs)^* = (sr)^*s\,.$$

Proof. If rs is cycle-free there exists a $k \geq 1$ such that $((rs)^k, \varepsilon) = 0$. This implies that $((sr)^{k+1}, \varepsilon) = (s(rs)^k r, \varepsilon) = 0$. Hence, rs is cycle-free iff sr is cycle-free. Now apply Theorem 2.7 of Kuich, Salomaa [14]. □

Recall that, in case of a Conway semiring A, for $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$, r^* is defined by a formula given in Section 1. In case of a cycle-free skew power series we can prove the validity of that formula in arbitrary semirings.

Theorem 37. *If $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is cycle-free then*

$$(r^*, \varepsilon) = (r, \varepsilon)^*$$

and, for all $w \in \Sigma^*$, $w \neq \varepsilon$,

$$(r^*, w) = \sum_{uv=w,\ u\neq\varepsilon} (r^*, \varepsilon)(r, u)(r^*, v)\,.$$

Proof. Analogous to the proofs of Lemmas 3.3, 3.4 and Theorem 3.5 of Kuich, Salomaa [14]. □

The next theorem shows that the sum-star-equation and the product-star-equation are valid for certain skew power series.

Theorem 38. *Let $r, s \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$. If r is cycle-free and $(s, \varepsilon) = 0$, or $(r, \varepsilon) = 0$ and s is cycle-free then*

$$(r + s)^* = (r^* s)^* r^*\,.$$

If rs or sr is cycle-free then

$$(rs)^* = \varepsilon + r(sr)^* s\,.$$

Proof. If r is cycle-free (resp. $(r, \varepsilon) = 0$) and $(s, \varepsilon) = 0$ (resp. s is cycle-free) then $r + s$ is cycle-free. Hence, $\lim_{n\to\infty}(r + s)^n = 0$ and $(r + s)^*$ exists by Corollary 35. Moreover, $(r^* s, \varepsilon) = 0$ (resp. $(r^* s, \varepsilon) = (s, \varepsilon)$). Hence, $r^* s$ is cycle-free and $(r^* s)^*$ exists by Theorem 34. Eventually, r^* exists, again by Theorem 34. Now, Theorems 2.8 and 2.7 of Kuich, Salomaa [14] prove the first statement of our theorem.

By Corollary 36, $s(rs)^* = (sr)^* s$. Hence, $\varepsilon + rs(rs)^* = \varepsilon + r(sr)^* s$. By Corollary 35, we obtain the equality $(rs)^* = \varepsilon + rs(rs)^*$. □

Corollary 39. *Let $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ be cycle-free and $r_0 = (r, \varepsilon)\varepsilon$, $r_1 = \sum_{w\in\Sigma^*,\ w\neq\varepsilon}(r, w)w$. Then*

$$r^* = (r_0 + r_1)^* = (r_0^* r_1)^* r_0^*\,.$$

We now turn to matrices $M \in A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$. In Theorem 40 and Corollary 41, we partition M and M^* into blocks

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \quad \text{and} \quad M^* = \begin{pmatrix} M^*(n_1, n_1) & M^*(n_1, n_2) \\ M^*(n_2, n_1) & M^*(n_2, n_2) \end{pmatrix},$$

where $n_1 + n_2 = n$, $M_{11}, M^*(n_1, n_1) \in A_\varphi^{n_1\times n_1}\langle\langle\Sigma^*\rangle\rangle$ and $M_{22}, M^*(n_2, n_2) \in A_\varphi^{n_2\times n_2}\langle\langle\Sigma^*\rangle\rangle$. The next theorem shows that, under certain conditions, the matrix-star-equation is valid.

Theorem 40. *Let $M \in A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ and assume that M_{11} and M_{22} are cycle-free and $(M_{21},\varepsilon)=0$. Then M is cycle-free and*

$$\begin{aligned} M^*(n_1,n_1) &= (M_{11}+M_{12}M_{22}^*M_{21})^*\,,\\ M^*(n_1,n_2) &= (M_{11}+M_{12}M_{22}^*M_{21})^*M_{12}M_{22}^*\,,\\ M^*(n_2,n_1) &= (M_{22}+M_{21}M_{11}^*M_{12})^*M_{21}M_{11}^*\,,\\ M^*(n_2,n_2) &= (M_{22}+M_{21}M_{11}^*M_{12})^*\,. \end{aligned}$$

Proof. In the proof of Theorem 4.22 of Kuich, Salomaa [14] it is shown that, for $j\geq 1$,

$$(M,\varepsilon)^j=\begin{pmatrix} (M_{11},\varepsilon)^j & \sum_{j_1+j_2=j-1}(M_{11},\varepsilon)^{j_1}(M_{12},\varepsilon)(M_{22},\varepsilon)^{j_2}\\ 0 & (M_{22},\varepsilon)^j \end{pmatrix}.$$

Since M_{11} and M_{22} are cycle-free there exist $k_1,k_2\geq 1$ such that $(M_{11},\varepsilon)^{k_1}=0$ and $(M_{22},\varepsilon)^{k_2}=0$. Hence, $(M,\varepsilon)^{k_1+k_2+1}=0$ and M is cycle-free.

Let now

$$a_1=\begin{pmatrix} M_{11} & 0\\ 0 & M_{22}\end{pmatrix} \quad\text{and}\quad a_2=\begin{pmatrix} 0 & M_{12}\\ M_{21} & 0\end{pmatrix}$$

and consider the matrix

$$\begin{aligned}(a_1+a_2a_1^*a_2,\varepsilon)=&\begin{pmatrix}(M_{11},\varepsilon) & 0\\ 0 & (M_{22},\varepsilon)\end{pmatrix}+\\ &\begin{pmatrix}0 & (M_{12},\varepsilon)\\ (M_{21},\varepsilon) & 0\end{pmatrix}\begin{pmatrix}(M_{11}^*,\varepsilon) & 0\\ 0 & (M_{22}^*,\varepsilon)\end{pmatrix}\begin{pmatrix}0 & (M_{12},\varepsilon)\\ (M_{21},\varepsilon) & 0\end{pmatrix}.\end{aligned}$$

Since $(M_{21},\varepsilon)=0$ this matrix equals (a_1,ε). Since $a_1+a_2=M$, and a_1 and $a_1+a_2a_1^*a_2$ are cycle-free, we can apply Theorem 2.9 of Kuich, Salomaa [14]:

$$(a_1+a_2)^*=(a_1+a_2a_1^*a_2)^*(1+a_2a_1^*)\,.$$

Computation of the right side of this equality yields the equations of our theorem. □

Corollary 41. *Let $M\in A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ and assume that M_{11} and M_{22} are cycle-free and $M_{21}=0$. Then M is cycle-fee and*

$$M^*=\begin{pmatrix} M_{11}^* & M_{11}^*M_{12}M_{22}^*\\ 0 & M_{22}^*\end{pmatrix}.$$

Corollary 42. *Let $M\in A_\varphi^{n\times n}\langle\langle\Sigma^*\rangle\rangle$ be of the form*

$$M=\begin{pmatrix} M_{11} & M_{12} & M_{13}\\ 0 & M_{22} & M_{23}\\ 0 & 0 & M_{33}\end{pmatrix},$$

where M_{11}, M_{22} and M_{33} are square blocks and assume that these blocks are cycle-free matrices. Then M is cycle-free and

$$M^* = \begin{pmatrix} M_{11}^* & M_{11}^* M_{12} M_{22}^* & M_{11}^* M_{12} M_{22}^* M_{23} M_{33}^* + M_{11}^* M_{13} M_{33}^* \\ 0 & M_{22}^* & M_{22}^* M_{23} M_{33}^* \\ 0 & 0 & M_{33}^* \end{pmatrix}.$$

Theorem 43. *Let $M \in (A_\varphi \langle\langle \Sigma^* \rangle\rangle)^{n_1 \times n_2}$ and $M' \in (A_\varphi \langle\langle \Sigma^* \rangle\rangle)^{n_2 \times n_1}$. Then MM' is cycle-free iff $M'M$ is cycle-free and, in this case,*

$$(MM')^* M = M(M'M)^*.$$

Proof. If MM' is cycle-free there exists a $k \geq 1$ such that $((MM')^k, \varepsilon) = 0$. This implies that $((M'M)^{k+1}, \varepsilon) = (M'(MM')^k M, \varepsilon) = 0$. Hence MM' is cycle-free iff $M'M$ is cycle-free.

We now distinguish three cases: $n_1 = n_2$, $n_1 > n_2$ and $n_1 < n_2$.

(i) If $n_1 = n_2$ then Theorem 36 proves our theorem.

(ii) If $n_1 > n_2$, write $M = \begin{pmatrix} a \\ b \end{pmatrix}$, $M' = (a' \; c')$, where $a, a' \in (A_\varphi \langle\langle \Sigma^* \rangle\rangle)^{n_2 \times n_2}$. Denote $M_0 = \begin{pmatrix} a & 0 \\ b & 0 \end{pmatrix}$, $M_0' = \begin{pmatrix} a' & c' \\ 0 & 0 \end{pmatrix}$ and observe that $M_0 M_0' = MM'$ and $M_0' M_0 = \begin{pmatrix} M'M & 0 \\ 0 & 0 \end{pmatrix}$. Moreover, by Corollary 41, $(M_0' M_0)^* = \begin{pmatrix} (M'M)^* & 0 \\ 0 & E \end{pmatrix}$. We now apply Theorem 36 and obtain, by $(M_0 M_0')^* M_0 = M_0 (M_0' M_0)^*$, the equation $(MM')^* M = M(M'M)^*$.

(iii) If $n_2 > n_1$, write $M = (a \; c)$, $M' = \begin{pmatrix} a' \\ b' \end{pmatrix}$, where $a, a' \in (A_\varphi \langle\langle \Sigma^* \rangle\rangle)^{n_1 \times n_2}$. Denote $M_0 = \begin{pmatrix} a & c \\ 0 & 0 \end{pmatrix}$, $M_0' = \begin{pmatrix} a' & 0 \\ b' & 0 \end{pmatrix}$ and observe that $M_0 M_0' = \begin{pmatrix} MM' & 0 \\ 0 & 0 \end{pmatrix}$ and $M_0' M_0 = M'M$. Moreover, by Corollary 41, $(M_0 M_0')^* = \begin{pmatrix} (MM')^* & 0 \\ 0 & E \end{pmatrix}$. We now apply Theorem 36 and obtain, by $(M_0 M_0')^* M_0 = M_0 (M_0' M_0)^*$, the equation $(MM')^* M = M(M'M)^*$. □

We now show part of the Kleene Theorem of Droste, Kuske [5], Theorem 3.6. Before, some auxiliary results are necessary.

A finite automaton $\mathfrak{A} = (n, I, M, P)$ over $A_\varphi \langle\langle \Sigma^* \rangle\rangle$ is called *normalized* if $n \geq 2$ and

(i) $I_1 = \varepsilon$, $I_i = 0$, $2 \leq i \leq n$;

(ii) $P_n = \varepsilon$, $P_i = 0$, $1 \leq i \leq n-1$;

(iii) $M_{i1} = M_{ni} = 0$, $1 \leq i \leq n$.

Theorem 44. *Let $\mathfrak{A}$ be a cycle-free finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then there exists a normalized cycle-free finite automaton $\mathfrak{A}'$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ with $||\mathfrak{A}'|| = ||\mathfrak{A}||$.*

Proof. Let $\mathfrak{A} = (n, I, M, P)$. Define

$$\mathfrak{A}' = (1+n+1, \begin{pmatrix} 0 & I & 0 \\ 0 & M & P \\ 0 & 0 & 0 \end{pmatrix}, (\varepsilon\ 0\ 0), \begin{pmatrix} 0 \\ 0 \\ \varepsilon \end{pmatrix}).$$

Then $\mathfrak{A}'$ is normalized. Moreover, by Corollary 42, $\mathfrak{A}'$ is cycle-free. Applying Corollary 42 yields the proof that $||\mathfrak{A}'|| = ||\mathfrak{A}||$. □

Theorem 45. *Let $\mathfrak{A}_1$ and $\mathfrak{A}_2$ be cycle-free finite automata over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then there exist cycle-free finite automata $\mathfrak{A}_1 + \mathfrak{A}_2$ and $\mathfrak{A}_1\mathfrak{A}_2$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ with $||\mathfrak{A}_1 + \mathfrak{A}_2|| = ||\mathfrak{A}_1|| + ||\mathfrak{A}_2||$ and $||\mathfrak{A}_1\mathfrak{A}_2|| = ||\mathfrak{A}_1||\,||\mathfrak{A}_2||$.*

Proof. Let $\mathfrak{A}_i = (n_i, I_i, M_i, P_i)$, $i = 1, 2$. Define

$$\begin{aligned} \mathfrak{A}_1 + \mathfrak{A}_2 &= (n_1+n_2, \begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix}, (I_1\ I_2), \begin{pmatrix} P_1 \\ P_2 \end{pmatrix}, \\ \mathfrak{A}_1\mathfrak{A}_2 &= (n_1+n_2, \begin{pmatrix} M_1 & P_1I_2 \\ 0 & M_2 \end{pmatrix}, (I_1\ 0), \begin{pmatrix} 0 \\ P_2 \end{pmatrix}. \end{aligned}$$

Then, by Corollary 41, $\mathfrak{A}_1 + \mathfrak{A}_2$ and $\mathfrak{A}_1\mathfrak{A}_2$ are cycle-free. Applying Corollary 41 yields the proof that $||\mathfrak{A}_1 + \mathfrak{A}_2|| = ||\mathfrak{A}_1|| + ||\mathfrak{A}_2||$ and $||\mathfrak{A}_1\mathfrak{A}_2|| = ||\mathfrak{A}_1||\,||\mathfrak{A}_2||$. □

A finite automaton $\mathfrak{A} = (n, I, M, P)$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is called *ε-free* if $(M, \varepsilon) = 0$.

Theorem 46. *Let $\mathfrak{A}$ be a cycle-free finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then there exists an ε-free finite automaton $\mathfrak{A}'$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ with $||\mathfrak{A}'|| = ||\mathfrak{A}||$.*

Proof. Let $\mathfrak{A} = (n, I, M, P)$. Define

$$\mathfrak{A}' = (n, I, M_0^*M_1, M_0^*P),$$

where $M_0 = (M, \varepsilon)$ and $M_1 = \sum_{x\in\Sigma}(M, x)x$. Then $\mathfrak{A}'$ is ε-free. We now apply the sum-star-equation of Corollary 39: $||\mathfrak{A}'|| = I(M_0^*M_1)^*M_0^*P = I(M_0 + M_1)^*P = IM^*P = ||\mathfrak{A}||$. □

Theorem 47. *Let $\mathfrak{A}$ be an ε-free finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$. Then there exists a cycle-free finite automaton $\mathfrak{A}^*$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ with $||\mathfrak{A}^*|| = ||\mathfrak{A}||^*$.*

Proof. Let $\mathfrak{A} = (n, I, M, P)$. Define

$$\mathfrak{A}^+ = (n, I, M + PI, P).$$

Since $\mathfrak{A}$ is ε-free, we obtain $IP = 0$. Hence, $(PI)^2 = 0$ and $\mathfrak{A}^+$ is cycle-free. We now apply Theorems 38 and 43: $||\mathfrak{A}^+|| = I(M + PI)^*P = I(M^*PI)^*M^*P = IM^*P(IM^*P)^*$.

Consider now the ε-free finite automata $\mathfrak{A}_\varepsilon = (1, \varepsilon, 0, \varepsilon)$ and $\mathfrak{A}^* = \mathfrak{A}_\varepsilon + \mathfrak{A}^+$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ with $||\mathfrak{A}_\varepsilon|| = \varepsilon$ and $||\mathfrak{A}^*|| = ||\mathfrak{A}||^*$. Here the second equality is obtained by Theorem 45 and Corollary 35. □

Theorem 48. *Given $r \in A_\varphi\langle\Sigma \cup \varepsilon\rangle$, there exists a cycle-free finite automaton $\mathfrak{A}$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ with $||\mathfrak{A}|| = r$.*

Proof. For $a \in A$, the finite automaton $\mathfrak{A}_a = (1, a\varepsilon, 0, \varepsilon)$ has behavior $||\mathfrak{A}_a|| = a\varepsilon$. For $x \in \Sigma$, the finite automaton

$$\mathfrak{A}_x = (2, (\varepsilon\ 0), \begin{pmatrix} 0 & x \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \varepsilon \end{pmatrix})$$

has behavior $||\mathfrak{A}_x|| = x$.

Since each $r \in A_\varphi\langle\Sigma \cup \varepsilon\rangle$ is generated from $a\varepsilon$, $a \in A$, and x, $x \in \Sigma$, by addition and multiplication, Theorem 45 proves our theorem. □

Corollary 49. *If $r \in \mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$ then there exists a cycle-free finite automaton $\mathfrak{A}$ over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ such that $||\mathfrak{A}|| = r$.*

Theorem 50. *Let $M \in (A_\varphi\langle\langle\Sigma^*\rangle\rangle)^{n\times n}$ with $(M, \varepsilon) = 0$. Then $M^* \in (\mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle))^{n\times n}$.*

Proof. An easy proof by induction on n using the matrix-star-equation of Theorem 40 proves our theorem (see Theorem 8.1 of Kuich, Salomaa [14]). □

Theorem 51 (Droste, Kuske [5]). *Let A be a semiring, $\varphi : A \to A$ be an endomorphism and Σ be an alphabet. Then the following statements are equivalent for $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$:*

(i) $r = ||\mathfrak{A}||$, where $\mathfrak{A}$ is a cycle-free finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$,

(ii) $r = ||\mathfrak{A}||$, where $\mathfrak{A}$ is an ε-free finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$,

(iii) $r \in \mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle)$.

Proof. (i) ⇒ (ii): By Theorem 46. (ii) ⇒ (iii): By Theorem 50. (iii) ⇒ (i): By Corollary 49. □

Droste, Kuske [5] introduce generalized weighted automata. This model of a finite automaton is captured by our next definition.

A *generalized finite automaton* $\mathfrak{A} = (n, I, M, P)$ *over* $A_\varphi\langle\langle\Sigma^*\rangle\rangle$ is defined as a finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$, except that $M \in (\mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle))^{n\times n}$. If $M \in (\mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle))^{n\times n}$ with $(M, \varepsilon) = 0$, then we obtain by an easy proof by induction on n using the matrix-star-equation of Theorem 40 that $M^* \in (\mathfrak{Rat}(A_\varphi\langle\Sigma \cup \varepsilon\rangle))^{n\times n}$ (see Theorem 8.1 of Kuich, Salomaa [14]). This together with a generalized version of Theorem 46 yields the following result, due to Droste, Kuske [5].

Theorem 52 (Droste, Kuske [5]). *Let A be a semiring, $\varphi : A \to A$ be an endomorphism and Σ be an alphabet. Then the following statements on $r \in A_\varphi\langle\langle\Sigma^*\rangle\rangle$ are equivalent to the statements of Theorem 51:*

(iv) $r = ||\mathfrak{A}||$, where $\mathfrak{A}$ is a cycle-free generalized finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$,

(v) $r = ||\mathfrak{A}||$, where $\mathfrak{A}$ is an ε-free generalized finite automaton over $A_\varphi\langle\langle\Sigma^*\rangle\rangle$.

References

[1] Aleshnikov, S., Boltnev, J., Ésik, Z., Ishanov, S., Kuich, W., Malachowskij, N.: Formaljnyje jasyki i awtomaty I: Polukoljza Konweja i konetschnyje awtomaty. (Formal languages and automata I: Conway semirings and finite automata.) Westnik Kaliningradskogo Gosudarstwennogo Universiteta, Wyp. 3. Ser. Informatika i telekommunikazii (2003) 7–38.

[2] Bloom, S. L., Ésik, Z.: Iteration Theories. EATCS Monographs on Theoretical Computer Science. Springer, 1993.

[3] Bouyer, P, Petit, A.: A Kleene/Büchi-like theorem for clock languages. J. of Automata, Languages and Combinatorics 7(2002) 167–186.

[4] Conway, J. H.: Regular Algebra and Finite Machines. Chapman & Hall, 1971.

[5] Droste, M., Kuske, D.: Skew and infinitary formal power series. Technical Report 2002–38, Department of Mathematics and Computer Science, University of Leicester.

[6] Droste, M., Kuske, D.: Skew and infinitary formal power series. ICALP 2003, LNCS 2719(2003) 426–438.

[7] Eilenberg, S.: Automata, Languages and Machines. Vol. A. Academic Press, 1974.

[8] Elgot, C.: Matricial theories. J. Algebra 42(1976) 391–422.

[9] Ésik, Z., Kuich, W.: On iteration semiring-semimodule pairs. To appear.

[10] Ésik, Z., Kuich, W.: A semiring-semimodule generalization of ω-regular languages I. J. of Automata, Languages and Combinatorics, to appear.

[11] Ésik, Z., Kuich, W.: A semiring-semimodule generalization of ω-regular languages II. J. of Automata, Languages and Combinatorics, to appear.

[12] Ésik, Z., Kuich, W.: A semiring-semimodule generalization of ω-context-free languages. LNCS 3113(2004) 68–80.

[13] Kuich, W.: Conway semirings and skew formal power series. Proceedings of the 11th International Conference on Automata and Formal Languages 2005 (Z. Ésik, Z. Fülöp, eds.), Institute of Informatics, University of Szeged, 2005, pp. 164–177.

[14] Kuich, W., Salomaa, A.: Semirings, Automata, Languages. EATCS Monographs on Theoretical Computer Science, Vol. 5. Springer, 1986.

Acta Cybernetica 17 (2006) 751–763.

A regular viewpoint on processes and algebra*

Kamal Lodaya†

Abstract

While different algebraic structures have been proposed for the treatment of concurrency, finding solutions for equations over these structures needs to be worked on further. This article is a survey of process algebra from a very narrow viewpoint, that of finite automata and regular languages. What have automata theorists learnt from process algebra about finite state concurrency? The title is stolen from [31]. There is a recent survey article [7] on finite state processes which deals extensively with rational expressions. The aim of the present article is different. How do standard notions such as Petri nets, Mazurkiewicz trace languages and Zielonka automata fare in the world of process algebra? This article has no original results, and the attempt is to raise questions rather than answer them.[1]

1 Formal languages

Formal language theory begins with the monoid of words $(\Sigma^*, \cdot, 1)$ over a finite alphabet Σ. A language is a set of words, and the algebraic structure of a set can be added to form an idempotent semiring $(\wp(\Sigma^*), \cdot, 1, +, 0)$. The identification of the semiring as a relevant algebraic structure is due to Conway [14] and Eilenberg [18].

Definition 1. *A* ***semiring*** *is a set S with an associative, commutative binary operation $+$ on S with identity 0; an associative binary operation $\cdot$ on S with identity 1 and absorbing element 0; and $\cdot$ distributing over $+$. The semiring is said to be* ***idempotent*** *if $+$ is idempotent.*

If we restrict ourselves to a regular language, recognized by a finite automaton, this amounts to saying that some equations hold in addition to those derived from the axioms of an idempotent semiring. Myhill and Nerode showed that recognizable languages, those saturated by finite-index congruences over the word monoid, are exactly the regular languages.

*This article is based on the talk "Looking back at process algebra" given at the AFL '05 conference in Dobogókő. I take this opportunity to thank the organizers of the conference, Zoltán Ésik and Zoltán Fülöp, for their invitation and hospitality. I also thank Zoltán Ésik for his encouragement over the years.

†The Institute of Mathematical Sciences, CIT Campus, Chennai 600 113, India.

[1] For some related questions in the world of process calculi, see [2].

Kleene showed that the regular languages can be modelled by rational expressions, formed by adding to the signature an additional unary star operation forming the (Kleene) starred (idempotent) semiring $(\wp(\Sigma^*), \cdot, 1, +, 0, ^*)$. We will henceforth assume idempotence of $+$ in our algebraic structures. As is usual, we will omit $\cdot$ when writing expressions.

Chomsky's type 3 grammars are another formalism to describe regular languages, where one works with a system of tail-recursive equations over the semiring $S[V]$ with a set of variables V. The equations can be put in Greibach form and solved using Arden's rule [3] which says that, with the proviso $a \neq 1 + a$, the equation $x = ax + b$ has the solution $\mu x.(ax + b) = a^*b$, where $\mu : V \times S[V] \to S$ is a partial function giving a unique solution $\mu x.e$ to the equation $x = e$ when it exists. Formally we are in a (Chomsky) μ-semiring [20] $(\wp(\Sigma^*)[V], \cdot, 1, +, 0, \mu)$.

This solution procedure is the basis of the axiomatization of equality of rational expressions by Aanderaa [1] and Salomaa [46], using the "no empty word property" (NEWP), a syntactically checkable condition equivalent to $a \neq 1 + a$ over the semiring of regular languages. Here is Salomaa's axiomatization:

Axiom system S for starred semirings

(Assoc)	$(a + b) + c = a + (b + c)$; $(ab)c = a(bc)$
(Ident)	$a + 0 = a$; $a1 = 1a = a$
(Comm)	$a + b = b + a$
(Idem)	$a + a = a$
(Absorp)	$a0 = 0a = 0$
(Distr)	$(a + b)c = ac + bc$; $a(b + c) = ab + ac$
(Guard)	$a^* = (1 + a)^*$
(Fixpt)	$a^* = 1 + aa^*$; $a^* = 1 + a^*a$
(GuardInd)	$\frac{x = ax + b}{x = a^*b}$; $\frac{x = xa + b}{x = ba^*}$ (provided a has NEWP)

Kozen gives an equational treatment using axioms and inference rules [28] and identifies Kleene algebras (which we will not describe here) as the basic structure. The main property used, inspired by Conway [14], is that matrices over a Kleene algebra form a Kleene algebra. These matrices can be used to encode automata and constructions over them. Completeness is proved by reducing to isomorphism over the minimal deterministic finite automaton.

1.1 Concurrency

Definition 2 (Petri [45]). *A **Petri net** is a bipartite directed graph $\mathcal{N} = (P, T, F)$ where P and T are disjoint finite sets of places and transitions, and $F \subseteq (P \times T) \cup (T \times P)$ a flow relation. For a place or transition y, its pre-set $\{x \mid xFy\}$ is conventionally denoted ${}^\bullet y$ and its post-set $\{z \mid yFz\}$ is denoted $y^\bullet$. F satisfies the condition that for each transition t, ${}^\bullet t$ and $t^\bullet$ are nonempty, and for each place p, either ${}^\bullet p$ or $p^\bullet$ is nonempty.*

A ***marking*** *is a multiset of places. A transition t is* ***enabled*** *at marking M if ${}^\bullet t \subseteq M$. A transition t enabled at M "fires" taking M to $(M - {}^\bullet t) + t^\bullet$. Given an initial marking M_0, the net system $(\mathcal{N}, M_0)$ is said to be* ***1-safe*** *if every reachable marking is a* set *(hence multisets are not required).*

The "firing sequences" of nets (words over the alphabet T) have been investigated thoroughly from the formal language viewpoint. For instance, since we have not introduced any notion of a final marking, the language accepted by a net system is prefix-closed. In the firing sequence view, nets are seen as no more than a representation of automata which have concurrent behaviour. The marking graph of a 1-safe net system, with vertices the reachable markings and edges representing the firing relation, is in fact a finite automaton. Concurrency is modelled as the shuffle or interleaving of two languages, for which rational expressions are sufficient since rationality is preserved by the shuffle.

But rational expressions are certainly not succinct for concurrent behaviour. The shuffle expression $a||b||c$ has equivalent rational expression $abc + acb + bac + bca + cab + cba$ (this is an instance of Milner's expansion axiom from CCS [34]), which shows that a shuffle can be exponentially succinct. A net for this language is exponentially succinct compared to the corresponding automaton.

The operating systems community continually had to deal with concurrent behaviour and were alive to this problem. They developed `cobegin-coend` [17], path expressions [13], and the languages COSY and CSP (fully described in the later books [27, 26]). The signature of rational expressions was expanded by binary shuffle operations $\{||_C \mid C \subseteq \Sigma\}$, with intersection over the letters in C. The intersection comes in handy to represent synchronization between concurrent processes.

Definition 3 (Grabowski [23]). *The* ***series-rational expressions*** *over an alphabet Σ consist of the atomic actions $a \in \Sigma$ and the constants 1 and 0, closed under the binary operations $\cdot, +, ||$ and the unary operation *.*

The shuffle operator $e_1||e_2$ is now redefined to additionally act as intersection whenever an action is shared between e_1 and e_2. In the term algebra generated from Σ, these expressions still describe languages over starred semirings, since a Milner-like expansion axiom can be used to eliminate the shuffle operations.

There is a translation from series-rational expressions to 1-safe net systems which preserves succinctness. The later work of Garg and Ragunath [21], when restricted to 1-safe nets, provides a method of going from net systems to these expressions (with the notable addition of renaming functions) when a distribution of places of a net is provided.

Grabowski [23] provided an interpretation of series-rational expressions over labelled posets (or "pomsets" as Pratt called them). A net can be seen as accepting a poset language, and Grabowski provided a two-way translation between 1-safe nets and series-rational expressions with renaming (including crucially renaming to the empty poset), representing regular poset languages. But posets are difficult to put into an algebraic framework. A popular representation of these posets which is closer to usual formal language theory is as Mazurkiewicz traces, to which we now turn.

2 Trace languages

Let I be an irreflexive symmetric relation over Σ, called independence, and let its reflexive transitive closure be $\sim_I$, called trace congruence. For instance, if aIb then $wabx \sim_I wbax$ (a and b commute). Sometimes it is convenient to consider the complementary symmetric dependence relation instead of independence.

Definition 4 (Mazurkiewicz [32]). *A trace over the concurrency alphabet (Σ, I) is a word over the partially commutative monoid $(\Sigma^*/\sim_I, \cdot, 1)$. Trace concatenation $\cdot$ works on the congruence classes. A trace language is a set of traces.*

Trace languages form the trace semiring $(\wp(\Sigma^*/\sim_I), \cdot, 1, +, 0)$ where the commutativity equations $ab = ba$ are added for every pair a, b in the independence relation. Hence only one representative of a trace needs to be described, the others being inferred, and we regain succinctness. We need not restrict ourselves to the term algebra, and the shuffle operations are not needed.

A 1-safe Petri net has a natural independence relation on its transitions: they are independent if their neighbourhoods are disjoint. This is a necessary condition for concurrent behaviour but not sufficient. The firing traces of a finite 1-safe net system are defined by quotienting the firing sequences with this independence relation. The set of firing traces form a recognizable trace language; that is, it is saturated by a finite-index congruence over the partially commutative monoid defined by (Σ, I). Again, because of the lack of final markings, the language will be prefix-closed.

Extend the independence relation to words: for nonzero w and x, let wIx iff every letter in w is independent with every letter in x. w and x are said to be connected if they are not independent. This syntactically checkable condition can be inductively lifted to rational expressions. Assuming that a and b are independent, we can derive $ab = ba = ab + ba$ by using idempotence. $a^*b^* = 1 + aa^*b^* + a^*bb^* = 1 + (a+b)a^*b^* = (a+b)^*$. The axiom system S is used in the first step and again in the last step, which is an application of the (GuardInd) rule.

Ochmański realized that it is sufficient to take the trace closure $[e^*]$ of the usual Kleene e^* over connected expressions e—that is, e and every starred subterm of e is connected [43]. If e does not satisfy this condition, Ochmański defined the concurrent star as described by the axiom below.

Now the trace languages (not just the prefix-closed ones) form an (Ochmański) starred trace (idempotent) semiring $(\wp(\Sigma^*/\sim_I), \cdot, 1, +, 0, ^*)$. An axiomatization for equality of recognizable trace languages was recently provided by the author [30].

Axioms TS for starred trace semirings	
(S)	All valid equalities for starred semirings
(Comm)	$ab = ba$, provided a and b are independent
(CStar)	$(ab + c)^* \stackrel{\text{def}}{=} (a + b + c)^*$, if a and b are independent

Assume that a, b and c are independent. By iterating the derivation $((a+b)^*c+d)^* = (a^*b^*c+d)^* \stackrel{\text{def}}{=} (a^* + b^* + c + d)^* = (a + b + c + d)^*$, where the first step was derived above and the last step uses the S system, the Ochmański star can be reduced to the Kleene star over connected expressions.

Question 5. *Is equality of trace languages over a given concurrency alphabet, described by rational expressions, recursively enumerable?*

Question 6. *Is there a complete axiomatization for rational trace languages over a concurrency alphabet?*

Here is a proof attempt which gets stuck.

Fix a total order over the letters of the alphabet and extend it lexicographically to words. Each trace can be represented by its lexicographically minimal word. Let Lex be the set of lexicographically minimal words. For a rational trace language TL, $Lex(TL) = Lex \cap (\bigcup TL)$ is a rational word language.

Suppose expressions a and b denote the same rational trace language $TL(a) = TL(b)$. By another theorem of Ochmański [16], there are connected rational expressions e and f whose word languages $WL(e) = Lex(TL(a))$ and $WL(f) = Lex(TL(b))$ are the same, and the trace closures are $[WL(e)] = TL(a)$ and $[WL(f)] = TL(b)$. By completeness of Salomaa's axiomatization, the equality $e = f$ is provable in S, and hence in TS. If we could show for a connected rational expression e that if e describes $Lex(TL(a))$, then $e = a$ is provable in TS, we could prove $a = b$ in TS and obtain its completeness. We do not have such an argument.

2.1 Distributed automata

A suitable automäton model which matches recognizability was defined by Zielonka [48]. Let Loc be a finite set of "locations", and $loc : \Sigma \to \wp(Loc)$ map each action to the locations required for executing it. Thus the alphabet Σ is distributed across the locations; if an action requires more than one location, we think of it as a synchronization between the distributed locations. A word language is said to be Loc-consistent if it is closed under commutation, where the actions a and b commute ($wabx \sim_{Loc} wbax$) if they are not shared by more than one location in Loc. Trace languages and Loc-consistent word languages are essentially the same thing.

Definition 7 (Zielonka [48]). *Let Q be a set of states distributed by the function $dist : Q \to Loc$. For $L \subseteq Loc$, let $\Pi_L Q$ be the functions $f : L \to Q$ such that $dist(f(i)) = i$. A **Zielonka automaton** over the distributed alphabet (Σ, Loc)*

is given by $(Q, dist, q_0, \rightarrow, F)$, *where* $q_0 \in \Pi_{Loc}Q$ *is a distributed initial state and* $F \subseteq \Pi_{Loc}Q$ *a set of distributed final states, and* $\rightarrow = \bigcup_{a \in \Sigma} \{\rightarrow_a \subseteq \Pi_{loc(a)}Q \times \Pi_{loc(a)}Q\}$ *is a transition relation.*

Zielonka automata are automata distributed over locations. The states are local, the transitions act on exactly those locations which an action is declared to require, and the final states are global. A run of a Zielonka automaton is defined over global states, every action transforming the states of the locations it affects, the other states remaining fixed. Zielonka [48] showed that the regular trace languages, those accepted by his automata, match the recognizable trace languages. Our notation for the automata follows Mukund and Sohoni [39], who provided an alternate proof of Zielonka's theorem by defining a gossip framework which explicitly represents state information shared across locations.

Thus trace theory [16] neatly generalizes formal language theory with regular trace languages playing a pivotal role. Mohalik and Ramanujam [38] provide a framework for *Loc*-consistent regular languages and a variant of series-rational expressions using special labelling functions, which provide a local presentation of distributed automata.

Question 8. *Is there an equational treatment of distributed automata in a Kleene algebra-like framework?*

3 Process calculi

We now turn to what Pnueli called the viewpoint of "reactive" systems: viewing automata in a concurrent environment not just as language generators but as processes. The classic vending machine example [26] shows that processes describe branching behaviour, and hence the left-distributivity axiom $a(b+c) = ab + ac$ for language equivalence fails. Some of the early models include failure sets, testing equivalences, synchronization trees and bisimulation [12, 15, 34, 44].

Definition 9 (Benson and Tiuryn [6]). *A* ***grove*** *is a set* G *with an associative, commutative binary operation* $+$ *with identity* 0, *an associative binary operation* $\cdot$ *with* 0 *a left zero, and where* $\cdot$ *right-distributes over* $+$, *that is,* $(a+b)\cdot c = a\cdot c + b\cdot c$. *A grove is* ***idempotent*** *if* $+$ *is idempotent. A* ***μ-grove*** $(G[V], \cdot, +, 0, \mu)$ *with a set of variables* V *has a partial solution function* $\mu : V \times G[V] \rightarrow G$ *analogous to a* μ*-semiring [20].*

A grove is defined by dropping the monoid identity, the right-absorption of 0 for multiplication and the left-distributivity of multiplication over addition from the axioms of a semiring. We will use μ-groves $G_\Sigma[V]$ generated from an alphabet Σ and a set of variables V as our basic model. (As before, we assume idempotence of $+$ in our structures.) Idempotent μ-groves are closely related to the axiomatization of bisimulation equivalence. Bloom and Ésik's monograph [10] provides a detailed description.

The first process calculus, Robin Milner's CCS, was published in 1980 [34]. Of course, CCS was based on a lot of earlier work, and Milner himself had been developing the idea for a few years, but *LNCS* 92 is the first fully developed treatment.

Milner proved a striking early result in process algebra [35], showing that tail-recursive equations (or guarded μ-expressions in his terminology) interpreted over μ-groves are sufficient to describe branching behavior of finite automata, whereas rational expressions over Kleene starred groves are not.

Axiom system M for μ-groves

(Assoc)	$(a+b)+c = a+(b+c)$; $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
(Comm)	$a+b = b+a$
(Idem)	$a+a = a$
(Ident)	$a+0 = a$
(LeftAbs)	$0 \cdot a = 0$
(RightDistr)	$(a+b) \cdot c = (a \cdot c)+(b \cdot c)$
(Guard)	$\mu x.e = \mu x.(x+e)$
(Fixpt)	$\mu x.e = e[\mu x.e/x]$
(GuardInd)	$\dfrac{f = e[f/x]}{f = \mu x.e}$ (provided x guarded in e)

The existence of unique solutions over certain groves was proved by Bergstra and Klop [8]. They also extended the positive result to automata with silent transitions [9], which was later developed by Milner in [36]. Since a finite system of tail-recursive equations implicitly defines a finite-index congruence on a finitely generated free grove, the negative result led to various kinds of extended star operations to restore the syntactic treatment known for rational languages. They are described in the survey article [7] mentioned in the introduction.[2]

3.1 Concurrency

Representing concurrency as interleaving of atomic actions, the shuffle operators can be added on since the expansion axioms are sound over groves. This yields the framework of process calculi [5]—PA and ACP for shuffles without and with synchronization respectively. Within a term model the shuffles can again be eliminated.

Bravetti and Gorrieri [11] extended Milner's axiomatization of regular behaviour to strongly guarded μ-expressions over Σ with shuffle, that is, those which are in Greibach form and do not allow a shuffle operation inside a recursion. The following question is still open:

Question 10. *Is there a direct way of going from finite 1-safe Petri nets to strongly guarded μ-expressions with shuffle, without incurring an exponential blowup?*

[2] The paper [4] provides a recent update on Milner's results and questions.

One approach may be to work with a "concurrent" bisimulation, as for example in [41]. Van Glabbeek and Vaandrager [22] proposed to axiomatize such a bisimulation by dropping the expansion axiom while retaining some desirable properties of the shuffle such as commutativity and associativity. That is, they expand groves with a shuffle operator $(G_\Sigma[V], \cdot, +, 0, ||, \mu)$. The shuffle is not reducible to the other operators.

Axiom system SM for μ-groves with shuffle	
(M)	All axioms of M
(Assoc||)	$(a\|\|b)\|\|c = a\|\|(b\|\|c)$
(Comm||)	$a\|\|b = b\|\|a$
(Ident||)	$a\|\|0 = a$
(Distr||)	$(a+b)\|\|c = (a\|\|c) + (b\|\|c)$
(StGuardInd)	$\dfrac{f = e[f/x]}{f = \mu x.e}$ (provided x strongly guarded in e)

Question 11. *Is there a complete axiomatization of concurrent bisimulation over finite state processes?*

3.2 Mobility

Process theory research seems to be moving more in the direction of value-passing [24] and mobile processes [19, 37], which are described by π-expressions upto a value-passing bisimulation, which comes in "early" and "late" variants to model eager and lazy forms of evaluation. We do not provide details of the syntax here.

Finite-control mobile systems model a state as an edge-labelled graph, where the nodes ("agents") have local storage to save some values and the edges ("links") communicate these values between the agents. Further, the values communicated are the link names themselves. Hence the atomic actions are of the form $c!v$ and $c?x$, sending a value v on a link c or receiving it in a variable x. To describe these systems, we allow tail-recursion in π-expressions, but disallow the replication operator which is sufficiently powerful to model general recursion. Effectively the syntax reduces to guarded μ-expressions *with parameters and a calling mechanism* built over an alphabet of atomic expressions with constants and variables (and with a shuffle, which is eliminable in a term algebra).

Milner's axiomatization has been extended to the value-passing bisimulations by Hennessy, Lin and Rathke [25] for finite-control systems described by tail-recursive π-expressions. However the underlying algebraic structure is far from clear. It appears to be some kind of combinatory grove, as illustrated by the communication axiom, which is based on the β-rule of λ-calculus:

$$(c!v \cdot P)||(c?x \cdot Q) = P||(Q[v/x]).$$

Question 12. *Can one describe the algebraic structure of mobile systems?*

3.3 Event structures

Definition 13 (Nielsen, Plotkin and Winskel [40]). *A (Σ-labelled) event structure $(E, \leq, \#, \ell)$ is a (Σ-labelled) poset $(E, \leq, \ell)$ with an irreflexive symmetric conflict relation $\#$ which is "inherited"; that is, if two events $e_1, e_2 \in E$ are in conflict, all events $e_1' \geq e_1$ and $e_2' \geq e_2$ above them are also in conflict. A configuration of an event structure is a downward-closed conflict-free set of events.*

Event structures are a generalization of traces or labelled posets to include branching behaviour. Events can be related by causality ($\leq$ or $\geq$), conflict ($\#$), or by neither causality nor conflict, in which case we say they are concurrent.

Configurations are a notion of "state" in an event structure. For the purposes of finite state behaviour, it is sufficient to restrict oneself to event structures which are finitary, where each event has a finite number of events below it, and have bounded enabling, that is, each configuration can be extended by a bounded number of immediately enabled successor events. In particular, this will mean that all configurations of interest are finite sets of events, and the conflict relation will be generated from an immediate conflict relation. We henceforth assume our event structures satisfy these properties.

We now lift some definitions from infinite trees.

Definition 14 (Thiagarajan [47]). *The residue of a configuration in an event structure is those events strictly above it. Two configurations are said to be right invariant if their residues are isomorphic as event structures. An event structure is recognizable if the right invariance relation on its configurations is of finite index.*

Although configurations are finite, residues can very well be infinite. The concurrent branching behaviour of a 1-safe Petri net can be defined by "unfolding" it; Thiagarajan proves that this yields a special kind of event structure.

Call an event structure deterministic if at any of its configurations, for any letter of the alphabet, at most one event labelled by that letter is enabled.

Definition 15 (Thiagarajan [47]). *A deterministic Σ-labelled event structure is said to be a **trace event structure** if there is an (irreflexive symmetric) independence relation over Σ such that the labels of concurrent events are independent, and the labels of neighbouring events (related by the immediate successor relation or immediate conflict relation) are dependent.*

Theorem 16 (Thiagarajan [47]). *An event structure is the unfolding of a 1-safe Petri net if and only if it is a recognizable trace event structure.*

The proof of the right-to-left direction goes via Zielonka's theorem.

Petri nets as we have defined them are not sufficiently abstract, since their behaviour is described in terms of the transitions T. Even a finite language like $\{a, aa\}$ is not representable. Hence one should start with a *labelled* 1-safe Petri net (P, T, F, ℓ), $\ell : T \to \Sigma$. Unfolding such a net certainly yields a recognizable labelled event structure, but it may no longer be deterministic.

Question 17. *Is the converse also true? Is a recognizable labelled event structure the unfolding of a labelled 1-safe Petri net?*

Thiagarajan [47] conjectured that the answer is yes. The conjecture has been proved for conflict-free event structures [42], where the conflict relation is empty; sequential event structures, which have no concurrency [42]; and deterministic event structures [29]. The general case is still open.

The reliance on determinism amounts, in the algebraic setting, to left-distributivity. So the basic algebraic structure is that of a semiring, or a trace semiring in the case of a trace event structure. Like posets, event structures are not well suited for algebra, and groves might be better to work with. Thiagarajan's conjecture leads one to ask the following:

Question 18. *Given a finite-index congruence over an idempotent grove with shuffle, is there a direct way of constructing a finite 1-safe Petri net which satisfies this particular behaviour?*

A categorical structure suitable for Petri nets has been proposed by Meseguer and Montanari [33]. A similar question can be raised in that setting.

References

[1] S. Aanderaa. On the algebra of regular expressions, in *Appl. Math. (course notes)*, Harvard, Jan 1965, 1–18.

[2] L. Aceto. Some of my favourite results in classic process algebra, *Bull. EATCS* 81, Oct 2003, 89–108.

[3] D.N. Arden. Delayed logic and finite state machines, in *Theory of computing machine design (course notes)*, U. Mich., Ann Arbor, 1960, 1–35.

[4] J.C.M. Baeten and F. Corradini. Regular expressions in process algebra, *Proc. LICS*, Chicago, IEEE, 2005, 12–19.

[5] J.C.M. Baeten and W.P. Weijland. *Process algebra*, CUP, 1990.

[6] D.B. Benson and J. Tiuryn. Fixed points in free process algebras I, *TCS* 63(3), 1989, 275–294.

[7] J. Bergstra, W. Fokkink and A. Ponse. Process algebra with recursive operations, in *Handbook of process algebra* (J. Bergstra, A. Ponse and S.A. Smolka, eds.), Elsevier, 2001, 333–389.

[8] J. Bergstra and J.W. Klop. Fixed point semantics in process algebra, *Report* IW 206/82, Centre for Mathematics and Computer Science, Amsterdam, 1982.

[9] J. Bergstra and J.W. Klop. A complete inference system for regular processes with silent moves, *Proc. Logic Colloquium*, Hull (F. Drake and J. Truss, eds.), North-Holland, 1986, 21–81.

[10] S. Bloom and Z.Ésik. *Iteration theories: the equational logic of iterative processes*, Springer, 1993.

[11] M. Bravetti and R. Gorrieri. Deciding and axiomatizing weak ST bisimulation for a process algebra with recursion and action refinement, *ACM TOCL* 3(4), 2002, 465–520.

[12] S.D. Brookes, C.A.R. Hoare and A.W. Roscoe. A theory of communicating sequential processes, *JACM* 31(3), 1984, 560–599.

[13] R.H. Campbell and A.N. Habermann. The specification of process synchronization by path expressions, in *Proc. Operating Systems conference* (E. Gelenbe and C. Kaiser, eds.), LNCS 16, 1974, 89–102.

[14] J.H. Conway. *Regular algebra and finite machines*, Chapman and Hall, 1971.

[15] R. De Nicola and M. Hennessy. Testing equivalences for processes, *TCS* 34, 1984, 83–133.

[16] V. Diekert and G. Rozenberg, eds. *The book of traces*, World Scientific, 1995.

[17] E.W. Dijkstra. Cooperating sequential processes, in *Programming languages* (F. Genuys, ed.), Academic Press, 1968.

[18] S. Eilenberg. *Automata, languages and machines* A, Academic Press, 1974.

[19] U.H. Engberg and M. Nielsen. A calculus of communicating systems with label-passing, Report DAIMI PB-208, Aarhus University, 1986.

[20] Z. Ésik and H. Leiß. Algebraically complete semirings and Greibach normal form, *Ann. Pure Appl. Logic* 133, 2005, 173–203.

[21] V.K. Garg and M.T. Ragunath. Concurrent regular expressions and their relationship to Petri nets, *TCS* 96(2), 1992, 285–304.

[22] R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency, *Proc. PARLE* 2, Eindhoven (J.W. de Bakker, A.J. Nijman and P.C. Treleaven, eds.), LNCS 259, 1987, 224–242.

[23] J. Grabowski. On partial languages, *Fund. Inform.* IV(2), 1981, 427–498.

[24] M. Hennessy and A. Ingólfsdóttir. A theory of communicating processes with value-passing, *Inf. Comput.* 107(2), 1993, 202–236.

[25] M. Hennessy, H. Lin and J. Rathke. Unique fixpoint induction for message-passing process calculi, *Sci. Comput. Program.* 41(3), 2001, 241–275.

[26] C.A.R. Hoare. *Communicating sequential processes*, Prentice-Hall, 1985.

[27] R. Janicki and P.E. Lauer. *Specification and analysis of concurrent systems: the COSY approach*, Springer, 1992.

[28] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events, *Inf. Comput.* 110(2), 1994, 366–390.

[29] K. Lodaya. Petri nets, event structures and algebra, in *Formal models, languages and applications* (K.G. Subramanian, K. Rangarajan and M. Mukund, eds.), World Scientific, 2006, 246–259.

[30] K. Lodaya. Product automata and process algebra, *Proc. SEFM*, Pune (D.V. Hung and P. Pandya, eds.), 2006, 128–136.

[31] D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes, *TCS* 274(1–2), 2002, 89–115.

[32] A. Mazurkiewicz. Concurrent program schemes and their interpretations, Report DAIMI PB-78, Aarhus University, 1977.

[33] J. Meseguer and U. Montanari. Petri nets are monoids, *Inf. Comput.* 88 (1990) 105–155.

[34] R. Milner. *A calculus of communicating systems*, LNCS 92, 1980.

[35] R. Milner. A complete inference system for a class of regular behaviours, *JCSS* 28(3), 1984, 439–466.

[36] R. Milner. A complete axiomatisation for observation congruence of finite-state behaviours, *Inf. Comput.* 81(2), 1989, 227–247.

[37] R. Milner, J. Parrow and D. Walker. A calculus of mobile processes I and II, *Inf. Comput.* 100(1), 1992, 1–77.

[38] S. Mohalik and R. Ramanujam. Distributed automata in an assumption-commitment framework, *Sādhanā* 27, Part 2, 2002, 209–250.

[39] M. Mukund and M. Sohoni. Keeping track of the latest gossip in a distributed system, *Distr. Comp.* 10(3), 1997, 117–127.

[40] M. Nielsen, G. Plotkin and G. Winskel. Petri nets, event structures and domains I, *TCS* 13 (1980) 86–108.

[41] M. Nielsen and P.S. Thiagarajan. Degrees of nondeterminism and concurrency: a Petri net view, *Proc. FSTTCS*, Bangalore (M. Joseph and R.K. Shyamasundar, eds.), LNCS 181, 1984, 89–117.

[42] M. Nielsen and P.S. Thiagarajan. Regular event structures and finite Petri nets: the conflict-free case, *Proc. ICATPN*, Adelaide (J. Esparza and C. Lakos, eds.), LNCS 2360, 2002, 335–351.

[43] E. Ochmański. Regular behaviour of concurrent systems, *Bull. EATCS* 27, 1985, 56–67.

[44] D. Park. Concurrency and automata on infinite sequences, *Proc. 5th GI conference*, Karlsruhe (P. Deussen, ed.), LNCS 104, 1981, 167–183.

[45] C.-A. Petri. Fundamentals of a theory of asynchronous information flow, *Proc. IFIP*, Munich (C.M. Popplewell, ed.), North-Holland, 1962, 386–390.

[46] A. Salomaa. Two complete axiom systems for the algebra of regular events, *JACM* 13(1), 1966, 158–169.

[47] P.S. Thiagarajan. Regular trace event structures, BRICS Research Abstracts RS-96-32, 1996.

[48] W. Zielonka. Notes on finite asynchronous automata, *RAIRO Inf. Th. Appl.* 21(2), 1987, 99–135.

Acta Cybernetica 17 (2006) 765–797.

Automata on Infinite Biposets*

Zoltán L. Németh†

Abstract

Bisemigroups are algebras equipped with two independent associative operations. Labeled finite sp-biposets may serve as a possible representation of the elements of the free bisemigroups. For finite sp-biposets, an accepting device, called parenthesizing automaton, was introduced in [6], and it was proved that its expressive power is equivalent to both algebraic recognizability and monadic second order definability. In this paper, we show, how this concept of parenthesizing automaton can be generalized for infinite biposets in a way that the equivalence of regularity (defined by acceptance with automata), recognizability (defined by homomorphisms and finite ω-bisemigroups) and MSO-definability remains true.

1 Introduction

The importance of automata and Büchi-automata is unquestionable in theoretical computer science from both theoretical and practical point of view. Its widespread applicability is mainly due to the fact that finite and infinite words can serve as models of a wide range of sequential systems. But, of course, there are many other computational models using more complex structures than words, such as trees, traces, posets, message sequence charts, graphs, etc. These models were introduced to capture other computational aspects, as timing or concurrency.

Besides the varying concept of automata and regularity, there is the more general notion of algebraic recognizability (by homomorphisms into finite algebras) and the concept of (counting) monadic second order logical definability. In many important cases these three notions can be suitably defined and they are known to be equivalent. In particular, this holds for finite trees, traces, message sequence charts, series-parallel posets of bounded width. See [23] for a recent survey on this topic. But sometimes we are confronted with serious difficulties. It is not always clear how to choose an appropriate algebraic or logic framework, and for graphs,

*An extended abstract of this paper appeared in the proceedings of AFL 2005 [19].

†Institute of Informatics, University of Szeged, P.O.B. 652, 6701 Szeged, Hungary, E-mail: zlnemeth@inf.u-szeged.hu

for posets, and even for sp-posets in general, a concept of automaton that matches algebraic recognizability is not known.

However, one of the most obvious generalizations of the case of words is the situation when we consider more than one, say n, associative operations. This naturally leads to the concept of n-semigroups and n-ω-semigroups. Accordingly, n-semigroups are sets equipped with n independent associative operations, and n-ω-semigroups are generalizations of the ω-semigroups of Perrin and Pin [20], where the formation of infinite (more precisely ω-ary) products is also allowed.

A description of the free n-semigroups by labeled finite n-posets was given by Ésik [5]. A Σ-labeled n-poset is a set P equipped with n patrial orders and a labeling function $P \to \Sigma$. One of the main results of [7] is a similar description of the free n-ω-semigroups by, so called, constructible n-posets. We say that a (finite or infinite) n-poset is constructible if it can be constructed from the singleton n-posets by the binary and the ω-ary product operations.

For simplicity, we only deal with the case when $n = 2$, i.e., we study bisemigroups and biposets only, although all of our notions and results can be generalized to n-semigroups and n-posets for any integer n greater than 2, without any difficulty.

In [6], an accepting device, called parenthesizing automaton, was introduced, and it was proved that for finite sp-biposets the recognizable, regular and MSO-definable languages coincide. Here we generalize the result mentioned above for infinite biposets. First, we show, with the help of a suitably defined notion of parenthesizing Büchi-automaton, that the class of regular languages of infinite constructible biposets coincides with the class of recognizable languages. We also demonstrate that, contrary to the word case, automata for infinite biposets must differ from automata for finite ones.

The equivalence of regular and recognizable sets implies that all MSO-definable languages are regular. Finally, we prove the converse inclusion, namely that every regular constructible biposet language is MSO-definable. (This verifies a conjecture of the preliminary version [19] of the present article.)

There are several branches of research that are in close connection with our investigations. Here we only briefly enumerate them, and refer to [6] where a whole section is devoted to a more detailed comparison. First of all, automata on series-parallel posets were studied by Lodaya and Weil in [15, 16, 17]. Their work was extended into two directions by Kuske [14], to automata on infinite posets and to (first- and second-order) logical definability. On text languages see the papers of Hoogeboom and ten Pas [12, 13]. On picture languages we refer to Giammarresi and Restivo [8] in general, and to Dolinka [1] in connection with sp-biposets. Finally, automata and languages over free bisemigroups (more precisely, free bisemigroups with identity, called binoids) have also been studied by Hashiguchi et al. [10, 11].

2 Basic concepts

2.1 Biposets and bisemigroups

In this paper, n always denotes a positive integer and Σ a finite alphabet. The empty word is denoted by ε. Let us call an algebra equipped with n associative operations *n-semigroup*. A *bisemigroup* is an n-semigroup for $n = 2$. It is proved in [5] that the elements of the free n-semigroups freely generated by some set Σ can be represented by finite Σ-labeled series-parallel n-posets defined as follows.

A *Σ-labeled n-poset*, or n-poset, for short, is a (finite or countably infinite) nonempty set P of vertices equipped with n (irreflexive) partial orders $<_i$ for $i = 1, \ldots, n$, and a labeling function $\lambda : P \to \Sigma$. We denote an n-poset by $P{=}(P, <_1, <_2, \ldots, <_n, \lambda)$, so we do not distinguish between the name of the biposet and the name of its vertex set. A *Σ-labeled biposet*, or biposet, is a Σ-labeled n-poset for $n = 2$.

The two partial orders of a biposet $(P, <_1, <_2, \lambda)$ are called the *horizontal* and the *vertical order*. Accordingly, instead of $<_1$ and $<_2$, we write $<_h$ and $<_v$, or $<_h^P$ and $<_v^P$ if we want to emphasize that these orderings belong to biposet P.

A *morphism* between biposets P and Q is a function on the vertices that preserves the partial orders and the labeling. An *isomorphism* is a bijective morphism whose inverse is also a morphism. Below we will identify isomorphic biposets.

Suppose that $P = (P, <_h^P, <_v^P, \lambda_P)$ and $Q = (Q, <_h^Q, <_v^Q, \lambda_Q)$ are Σ-labeled biposets. Without loss of generality, assume that P and Q are disjoint. We define their *horizontal product* as $P \bullet Q := (P \cup Q, <_h^{P \bullet Q}, <_v^{P \bullet Q}, \lambda_{P \bullet Q})$, and their *vertical product* as $P \circ Q := (P \cup Q, <_h^{P \circ Q}, <_v^{P \circ Q}, \lambda_{P \circ Q})$, where

$$\begin{array}{llll} <_h^{P\bullet Q} := <_h^P \cup <_h^Q \cup (P \times Q), & <_h^{P\circ Q} := <_h^P \cup <_h^Q, \\ <_v^{P\bullet Q} := <_v^P \cup <_v^Q, & <_v^{P\circ Q} := <_v^P \cup <_v^Q \cup (P \times Q), \end{array}$$

and the labelings are $\lambda_{P \bullet Q} = \lambda_{P \circ Q} := \lambda_P \cup \lambda_Q$.

We say that a finite or infinite biposet P is *horizontal* if there are biposets P_1 and P_2 such that $P = P_1 \bullet P_2$, otherwise P is called *$\bullet$-irreducible* or *horizontally irreducible*. Similarly, P is *vertical* if it can be written as $P = P_1 \circ P_2$, and P is said to be *$\circ$-irreducible* or *vertically irreducible* if no such decomposition exists. The fact that P is a horizontal (vertical) biposet will be abbreviated as $\mathrm{Type}(P) = \bullet$ ($\mathrm{Type}(P) = \circ$, resp.) If P is a horizontal (vertical) biposet, then any factorization $P = P_1 \bullet P_2 \bullet \ldots \bullet P_m$ ($P = P_1 \circ P_2 \circ \ldots \circ P_m$), where $m \geq 2$, is called a *horizontal (vertical, resp.) decomposition* of P. A horizontal (vertical) decomposition is said to be *maximal* if every factor is horizontally (vertically, resp.) irreducible.

It is obvious that both product operations are associative. Each letter $\sigma \in \Sigma$ may be identified with the singleton biposet labeled σ. Let $\mathrm{SPB}(\Sigma)$ denote the collection of biposets that can be generated from the singletons corresponding to the letters in Σ by the two product operations in a finite number of steps. Clearly, these biposets are finite. The biposets in $\mathrm{SPB}(\Sigma)$ are called *series-parallel biposets*, or *sp-biposets*, for short. It is known that series-parallel biposets have a graph-theoretic characterization, which is an appropriate generalization of the "N-free"

condition for posets, cf. [5, 9, 22]. We say that an arbitrary biposet P is *complete* if every two vertices of P are related either horizontally or vertically, but not by both order relations. It is obvious that every sp-biposet is complete.

Proposition 1 ([5]). *A finite biposet $(P, <_h, <_v, \lambda)$ is in* $\mathrm{SPB}(\Sigma)$ *if and only if P is complete and both posets $(P, <_h)$ and $(P, <_v)$ are N-free.*

Proposition 2 ([5]). $\mathrm{SPB}(\Sigma)$ *is freely generated by Σ in the variety of bisemigroups.*

2.2 Term and tree representation of sp-biposets

The most evident way of representing sp-biposets is describing them by terms. For this reason, we extend the alphabet with operation symbols and parentheses. Let $\widehat{\Sigma} := \Sigma \cup \{\bullet, \circ, \langle, \rangle\}$. As usual, we should put parentheses around the horizontal biposets that appear as vertical factors, and symmetrically, around the vertical biposets that appear as horizontal factors. The precise definition is the following.

Definition 3. *If $P \in \mathrm{SPB}(\Sigma)$, let P^{tm} denote the* term representation *of P. It is a word over the alphabet $\widehat{\Sigma}$, defined inductively as follows.*

(i) If $P = \sigma$ is a singleton biposet, then $P^{tm} := \sigma$.

(ii) If $P = P_1 \bullet P_2$, then $P^{tm} := \mathrm{Hform}(P_1) \bullet \mathrm{Hform}(P_2)$.

(iii) If $P = P_1 \circ P_2$, then $P^{tm} := \mathrm{Vform}(P_1) \circ \mathrm{Vform}(P_2)$.

Here $\mathrm{Hform}(P)$ *denotes the* horizontal form *of the sp-biposet P, defined as:*

$$\mathrm{Hform}(P) := \begin{cases} P^{tm} & \text{if } P \text{ is a singleton or horizontal biposet,} \\ \langle\, P^{tm} \,\rangle & \text{if } P \text{ is a vertical biposet.} \end{cases}$$

In (iii), $\mathrm{Vform}(P)$, *the* vertical form *of P, is defined symmetrically.*

It should be noted that in cases *(ii)* and *(iii)* above, the definition of P^{tm} does not depend on the choice of the factorization, since $\bullet$, $\circ$ and the concatenation of words are all associative operations.

We will also use finite ordered trees to represent sp-biposets. In that case, leaves are labeled from Σ, and the inner nodes are labeled by $\bullet$ or $\circ$.

Definition 4. *If P is an sp-biposet, its* tree form *P^{tr}, is defined as follows.*

(i) If $P = \sigma$ is a singleton, then P^{tr} is a tree consisting of a single vertex labeled by σ.

(ii) If P is horizontal, then consider the maximal horizontal decomposition $P = P_1 \bullet P_2 \bullet \ldots \bullet P_m$, $(m \geq 2)$. Now P^{tr} is the tree whose root is labeled $\bullet$ and this root connects the subtrees $P_1^{tr}, P_2^{tr}, \ldots, P_m^{tr}$ (in that order).

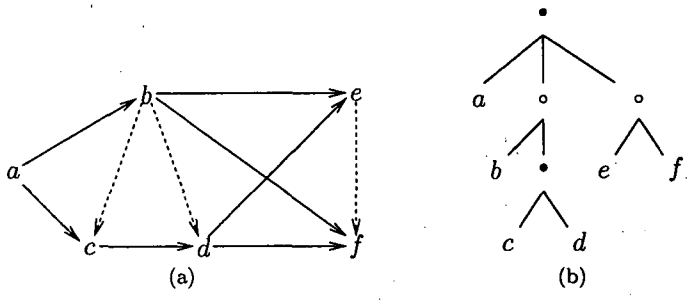


Figure 1: The biposet P of Example 5 (a), and its tree representation P^{tr} (b).

(iii) If P is vertical, then consider the maximal vertical decomposition $P = P_1 \circ P_2 \circ \ldots \circ P_m$, $(m \geq 2)$. Now P^{tr} is the tree whose root is labeled $\circ$ and this root connects the subtrees $P_1^{tr}, P_2^{tr}, \ldots, P_m^{tr}$ (in that order).

Example 5. Consider the sp-biposet $P = (\{1, 2, \ldots, 6\}, <_h, <_v, \lambda)$, where $<_h$ and $<_v$ are the transitive closures of the relations $1 <_h 2$, $1 <_h 3$, $3 <_h 4$, $2 <_h 5$, $2 <_h 6$, $4 <_h 5$, $4 <_h 6$, and $2 <_v 3, 2 <_v 4, 5 <_v 6$, respectively. Moreover, $\lambda(1) = a$, $\lambda(2) = b$, $\lambda(3) = c$, $\lambda(4) = d$, $\lambda(5) = e$, $\lambda(6) = f$. Now $P^{tm} = a \bullet \langle b \circ \langle c \bullet d \rangle \rangle \bullet \langle e \circ f \rangle$, and the graphical representation of P and the tree representation P^{tr} are depicted in Figure 1. In the figure, horizontal and vertical relations are indicated by solid arrows and dashed arrows, respectively.

It is obvious that for any leaf node in P^{tr} there is a corresponding vertex in P. Hence, we may and will identify the leaves of P^{tr} with the corresponding vertices of P. This allows us to speak about elements and subsets of P as those of P^{tr}. Similarly, we can identify vertices of P with the corresponding letters in the term representation P^{tm}.

2.3 Infinite biposets and ω-bisemigroups

In this subsection, we briefly summarize the main results of [7] regarding infinite biposets. First, we introduce two types of operations that construct infinite biposets from finite ones: the ω-product and the ω-power.

Suppose that $P_1, P_2, \ldots$ are pairwise disjoint finite biposets. Their *horizontal ω-product* is defined as

$$\omega_\bullet(P_1, P_2, \ldots) := (P_1 \cup P_2 \cup \ldots, <_h, <_v, \lambda)$$

where

$$<_h := \bigcup_{i=1}^{\infty} <_h^{P_i} \cup \bigcup_{i<j} (P_i \times P_j), \qquad <_v := \bigcup_{i=1}^{\infty} <_v^{P_i}$$

and

$$\lambda := \lambda_{P_1} \cup \lambda_{P_2} \cup \dots$$

The *vertical ω-product* $\omega_\circ(P_1, P_2, \dots)$ is defined symmetrically. We will also refer to horizontal and vertical ω-products as $P_1 \bullet P_2 \bullet \dots$ and $P_1 \circ P_2 \circ \dots$, respectively. The two ω-product operations naturally induce a horizontal and a vertical power operation: $P^{\omega\bullet} := P \bullet P \bullet P \bullet \dots$, and $P^{\omega\circ} := P \circ P \circ P \circ \dots$

Note that the definition of the product operations applies to both finite and infinite operands. Nevertheless, in order to avoid constructing biposets which have chains not contained in ω, we will restrict the product operations $P \bullet Q$ and $P \circ Q$ to a finite biposet P only. The biposet Q may be finite or infinite. The ω-product and ω-power operations are applied only to finite biposets. These restrictions seem to be necessary for the proofs later.

All the restrictions just described imply that we should use two-sorted algebras as our algebraic framework making a difference between the finite and the infinite elements. Fortunately, this can be done in complete analogy to the case of finite and infinite words cf. [20]. But, as a minor difference from op. cit., we assume the binary product operations to be appropriately polymorphic, i.e., we use the same notation for the product of two finite biposets and for the product of a finite and an infinite biposet.

Accordingly, call an algebra $\mathcal{B} = (B_F, B_I, \bullet, \circ, \omega_\bullet, \omega_\circ)$ an *ω-bisemigroup* if it satisfies the following identities

$$\begin{aligned} x * (y * u) &= (x * y) * u, \\ x * \omega_*(x_1, x_2, \dots) &= \omega_*(x, x_1, x_2, \dots), \\ \omega_*(x_1 * \dots * x_{k_1-1},\, x_{k_1} * \dots * x_{k_2-1}, \dots) &= \omega_*(x_1, \dots, x_{k_1-1},\, x_{k_1}, \\ &\qquad x_{k_1+1}, \dots, x_{k_2-1}, \dots), \end{aligned}$$

for all $x, y, x_1, x_2, \dots \in B_F$, $u \in B_F \cup B_I$, $* \in \{\bullet, \circ\}$, and for all increasing sequences of positive integers $k_1 < k_2 < \dots$

A *morphism* of ω-bisemigroups $\mathcal{C} = (C_F, C_I, \bullet, \circ, \omega_\bullet, \omega_\circ) \to \mathcal{D} = (D_F, D_I, \bullet', \circ', \omega'_\bullet, \omega'_\circ)$ is a pair of functions $h = (h_F : C_F \to D_F, h_I : C_I \to D_I)$ that jointly preserve the operations.

We call a Σ-labeled biposet *constructible* if it can be generated from the singleton Σ-labeled biposets by the (restricted) binary product operations $\bullet$ and $\circ$, and by the ω-ary product operations $\omega_\bullet$ and $\omega_\circ$.

Note that $\mathrm{SPB}(\Sigma)$ is exactly the set of those constructible biposets which are finite. Let $\mathrm{ISPB}(\Sigma)$ denote the set of infinite constructible biposets, and let

$$\omega\mathrm{SPB}(\Sigma) := (\mathrm{SPB}(\Sigma), \mathrm{ISPB}(\Sigma), \bullet, \circ, \omega_\bullet, \omega_\circ)$$

stand for the two-sorted algebra of all constructible biposets over Σ. It is clear that this is an ω-bisemigroup. Now, it is easily seen that the set of all finite and countably infinite biposets also form an ω-bisemigroup, and $\omega\mathrm{SPB}(\Sigma)$ is the smallest subalgebra of this ω-bisemigroup that contains Σ. The infinite counterpart of Proposition 2 is the following.

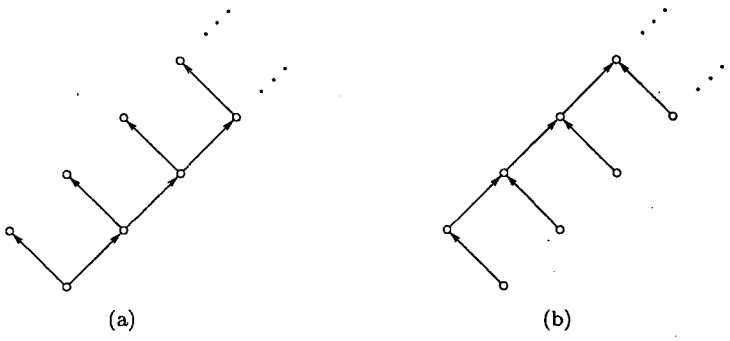


Figure 2: An upward comb (a) and a downward comb (b).

Proposition 6 ([7]). *The algebra* $\omega\mathrm{SPB}(\Sigma)$ *is freely generated by* Σ *in the variety of* ω*-bisemigroups.*

A graph-theoretic characterization of sp-biposets is also given in [7]. This, of course, is a suitable generalization of the "generalized N-free" condition of the finite case.

Proposition 7 ([7]). *An infinite biposet* $(P, <_h, <_v, \lambda)$ *is in* $\mathrm{ISPB}(\Sigma)$ *if and only if* P *is complete, and both posets* $(P, <_h)$ *and* $(P, <_v)$

(i) are N-free,

(ii) are free of "upward combs",

(iii) are free of "downward combs", and

(iv) have only finite principal ideals,

where the "upward comb" and "downward comb" posets are depicted in Figure 2.

See [7] for precise definitions.

In order to simplify the notations, in the sequel, we use $*$ to indicate any of the $\bullet$ and $\circ$ operations. Sometimes, we also give subscripts to the $*$-s, but in any formula all $*$ symbols, without subscript or with the same subscript, always denote the same operation.

A decomposition of P into an ω-product of infinitely many biposets $P = P_1 * P_2 * \ldots$ is said to be *maximal* if every P_i is $*$-irreducible. If P is an infinite constructible biposet, we say that P is *primitive* if it can be written as $P_1 * P_2 * \ldots$ for some finite sp-biposets $P_1, P_2, \ldots$ Now each infinite constructible biposet can be generated from the primitive biposets by multiplication with finite sp-biposets from the left. We define the *rank* of an infinite constructible biposet P as the least number of left multiplications with finite sp-biposets needed to construct P from the primitive infinite biposets. The rank of P is denoted by $\mathrm{Rank}(P)$.

It is easy to prove that if an infinite constructible biposet P is not primitive, than it can be uniquely written as $P = P' * P''$, where P'' is $*$-irreducible and $\mathrm{Rank}(P'') < \mathrm{Rank}(P)$. A direct consequence of this fact is that every infinite constructible biposet has the form

$$P_1 *_1 (P_2 *_2 (P_3 *_3 \ldots P_k *_k (Q_1 *_{k+1} Q_2 *_{k+1} Q_3 *_{k+1} \ldots))), \tag{1}$$

where all P_i and Q_i are finite biposets in $\mathrm{SPB}(\Sigma)$, and $*_1, *_2, *_3 \ldots$ is an alternating sequence of the $\bullet$ and $\circ$ operations. Moreover, this form is unique provided that every Q_i is $*_{k+1}$-irreducible. In this case, we call it the *normal form* of P. Note that if (1) is the normal form of P, then $\mathrm{Type}(P) = *_1$ and $\mathrm{Rank}(P) = k$.

2.4 Tree and term representations of infinite constructible biposets

Here we outline the changes to be made if one intends to represent infinite constructible biposets by terms and trees.

The only thing we need to describe is how to handle infinite products as $P = P_1 \bullet P_2 \bullet \ldots$ The definition of P^{tr} is straightforward if we allow ω-branching in trees. The tree P^{tr} has a root labeled by $\bullet$, and this root has ω branches connecting the tree representations of all the horizontally irreducible components of the P_i-s $(i \geq 1)$.

There are only slight changes also in the term representation. The term representations of a biposet in $\mathrm{ISPB}(\Sigma)$ is an ω-word over the extended alphabet $\widehat{\Sigma}' := \Sigma \cup \{\langle, \rangle, [\}$. We should add two more cases to Definition 3:

(iv) *If* $P = P_1 \bullet P_2 \bullet \ldots$, *then* $P^{tm} := \mathrm{Hform}(P_1) \bullet \mathrm{Hform}(P_2) \bullet \ldots$

(v) *If* $P = P_1 \circ P_2 \circ \ldots$, *then* $P^{tm} := \mathrm{Vform}(P_1) \circ \mathrm{Vform}(P_2) \circ \ldots$

The definitions of the horizontal and vertical forms are also extended appropriately. In the representation of a product of a finite biposet with an infinite one, we use the [symbol if the type of the product differs from the type of the infinite factor. We only give the definition of the horizontal form:

$$\mathrm{Hform}(P) := \begin{cases} P^{tm} & \text{if } P \text{ is a singleton or a horizontal biposet,} \\ \langle\, P^{tm} \,\rangle & \text{if } P \text{ is a finite vertical biposet,} \\ [\, P^{tm} & \text{if } P \text{ is an infinite vertical biposet.} \end{cases}$$

2.5 Recognizability

A language consisting of finite sp-biposets is said to be *recognizable* if it is recognized by a homomorphism into a finite bisemigroup, i.e., $L \subseteq \mathrm{SPB}(\Sigma)$ is recognizable if and only if $L = \varphi^{-1}(F)$, for some bisemigroup homomorphism $\varphi : \mathrm{SPB}(\Sigma) \to B$, where B is a finite bisemigroup, and $F \subseteq B$.

Similarly, for a language that contains both finite and infinite biposets, $L = (L_F, L_I) \subseteq \omega\mathrm{SPB}(\Sigma)$, is recognizable if and only if there is a finite ω-bisemigroup $\mathcal{B} = (B_F, B_I)$, a subset of it, $T = (T_F, T_I) \subseteq (B_F, F_I)$, and a morphism $\varphi = (\varphi_F, \varphi_I) : \omega\mathrm{SPB}(\Sigma) \to \mathcal{B}$ such that $L = \varphi^{-1}(T)$. Here $(T_F, T_I) \subseteq (B_F, F_I)$ means $T_F \subseteq B_F$ and $T_I \subseteq F_I$, moreover, $L = \varphi^{-1}(T)$ stands for $L_F = \varphi_F^{-1}(T_F)$ and $L_I = \varphi_I^{-1}(T_I)$.

Example 8. Let $\Sigma = \{a, b, c\}$, and consider the following language $L \subseteq \mathrm{ISPB}(\Sigma)$ of infinite biposets

$$L = \{\, c^{\omega\bullet},\ a \bullet (b \circ (c^{\omega\bullet})),\ a \bullet (b \circ (a \bullet (b \circ (c^{\omega\bullet})))), \ldots \}.$$

L is the least solution of the fixed point equation $a \bullet (b \circ X) + c^{\omega\bullet} = X$. It is not hard to show that L is recognizable. Indeed, consider the finite bisemigroup $B = (B_F, B_I)$, where $B_F = \{\, d_a, d_b, d_c, 0 \,\}$, and $B_I = \{\, t_1, t_2, t_3, \underline{0} \,\}$. The binary product operations are given by $d_c \bullet d_c = d_c$, and all other binary products of two finite elements are equal to 0, moreover, $d_c \bullet t_1 = t_1$, $d_b \circ t_1 = t_2$, $d_a \bullet t_2 = t_3$, $d_b \circ t_3 = t_2$, and all other products of a finite element with an infinite one are equal to $\underline{0}$. Finally, the ω-product operations are given by $d_c^{\omega\bullet} = t_1$, and all other ω-products are equal to $\underline{0}$. Now, if we take the homomorphism $\varphi : \omega\mathrm{SPB}(\Sigma) \to \mathcal{B}$ that is induced by the mapping $a \mapsto d_a$, $b \mapsto d_b$, $c \mapsto d_c$, then $L = \varphi_I^{-1}(\{\, t_1, t_3 \,\})$. This shows that L is recognizable.

2.6 Logical definability

By considering biposets as relational structures, there is a usual way of defining languages by logical formulas. Let $\mathcal{V} = \{x, y, \ldots\}$ denote a fixed countable set of first-order variables, and $\mathcal{W} = \{X, Y, \ldots\}$ a fixed countable set of monadic second-order variables.

Now we define monadic second order (MSO) formulas. An *atomic formula* (over Σ, $\mathcal{V}$ and $\mathcal{W}$) is of the form $Q_a(x)$, $X(x)$, $x <_h y$ or $x <_v y$, where $a \in \Sigma$, $x, y \in \mathcal{V}$, and $X \in \mathcal{W}$. MSO-*formulas* are composed from atomic formulas by the boolean connectives $\vee$ and $\neg$ and first- and second-order existential quantifiers $\exists x$ and $\exists X$, where $x \in \mathcal{V}$ and $X \in \mathcal{W}$.

We interpret formulas over both finite and infinite constructible biposets. Suppose that P is in $\mathrm{SPB}(\Sigma)$ or in $\mathrm{ISPB}(\Sigma)$. First order variables are interpreted to be vertices (also called positions) in P, whereas second order variables are interpreted to be sets of positions in P. Now, $Q_a(x)$ means that vertex x is labeled by a and $X(x)$ means that x belongs to X. The atomic formulas $x <_h y$ and $x <_v y$ have their expected meanings. The fact that a *closed formula (sentence) φ holds in, or is satisfied by P* is defined in the usual way, and it is denoted $P \models \varphi$. The *language defined by φ* is $L_\varphi := \{\, P \in \mathrm{(I)SPB}(\Sigma) \mid P \models \varphi \,\}$.

Definition 9. *We say that a language $L \subseteq$ (I)SPB(Σ) is* MSO-definable *if there is sentence φ with $L = L_\varphi$.*

3 Automata and regularity

In this section, we will define parenthesizing automata operating on finite constructible biposets (i.e., on sp-biposets), and parenthesizing Büchi-automata operating on infinite constructible biposets.

3.1 Parenthesizing automata

An accepting device, called parenthesizing automaton, was introduced in [6] to define the class of regular languages of sp-biposets. Its definition below involves a finite set Ω of parentheses. *We assume that Ω is partitioned into opening and closing parentheses that are in a bijective correspondence.* We usually denote the corresponding pairs by $\langle_1, \rangle_1$ and $\langle_2, \rangle_2$, etc.

Definition 10. *A (nondeterministic)* parenthesizing automaton *is a 9-tuple* $\mathcal{A} := (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$, *where S is a nonempty, finite set of* states, *H and V are the sets of* horizontal *and* vertical states, *which give a disjoint partition of S, Σ is the* input alphabet, *Ω is a finite set of parentheses, moreover,*

- $\delta \subseteq (H \times \Sigma \times H) \cup (V \times \Sigma \times V)$ *is the* labeled transition relation,
- $\gamma \subseteq (H \times \Omega \times V) \cup (V \times \Omega \times H)$ *is the* parenthesizing transition relation,
- $I, F \subseteq S$ *are the sets of* initial *and* final states, *respectively.*

Let $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ be a parenthesizing automaton. If $t = (p, x, q)$ is a labeled or parenthesizing transition of $\mathcal{A}$, i.e., $t \in \delta \cup \gamma$, the *starting* and the *ending state* of t is denoted by $\mathrm{start}(t) := p$ and $\mathrm{end}(t) := q$, respectively. Moreover, if $\mathbf{r} = t_1 t_2 \dots t_n \in (\delta \cup \gamma)^*$ is a sequence of transitions, then let $\mathrm{start}(\mathbf{r}) := \mathrm{start}(t_1)$ and $\mathrm{end}(\mathbf{r}) := \mathrm{end}(t_n)$. We say that two parenthesizing transitions $t_1 = (p, \omega_1, q)$ and $t_2 = (s, \omega_2, t) \in \gamma$ form a *parenthesizing transition pair* if ω_1 is an opening parenthesis and ω_2 is its closing partner.

Definition 11. *Let $\mathcal{A}$ be a parenthesizing automaton. The set of its* runs, $\mathrm{Runs}(\mathcal{A})$, *is the least set of transition sequences that contains*

(i) (p, σ, q) *for every* $(p, \sigma, q) \in \delta$;

(ii) $\mathbf{r_1 r_2}$ *for every* $\mathbf{r_1}, \mathbf{r_2} \in \mathrm{Runs}(\mathcal{A})$ *provided that* $\mathrm{end}(\mathbf{r_1}) = \mathrm{start}(\mathbf{r_2})$;

(iii) $t_1 \mathbf{r} t_2$ *for every t_1 and t_2 parenthesizing transition pair such that* $\mathrm{end}(t_1) = \mathrm{start}(\mathbf{r})$, $\mathrm{end}(\mathbf{r}) = \mathrm{start}(t_2)$, *and for every* $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$ *such that* $\mathbf{r}$ *is of the form* $\mathbf{r} = \mathbf{r_1 r_2}$, *where* $\mathbf{r_1}, \mathbf{r_2} \in \mathrm{Runs}(\mathcal{A})$.

In case (i), the run is called singleton run, *in case (ii), it is called* direct run, *in case (iii) the run is called* indirect run.

Let $\mathcal{A}$ be a parenthesizing automaton, $\mathbf{r} = t_1 \dots t_n \in \mathrm{Runs}(\mathcal{A})$. A parenthesizing transition pair t_i, t_j, $(i < j)$ is said to be a *matching parenthesizing transition pair in* $\mathbf{r}$ if $t_i \dots t_j$ is an indirect run of $\mathcal{A}$. It is obvious that every run of $\mathcal{A}$ is of the form

$$\mathbf{r} = t_1 t_2 t_3 \dots t_n = (p_0, \omega_1, p_1)(p_1, \omega_2, p_2)(p_2, \omega_3, p_3) \dots (p_{n-1}, \omega_n, p_n),$$

where $p_i \in S$ and $\omega_i \in \Sigma \cup \Omega$ for all $i = 1, \dots, n$. If $\mathbf{r}$ is an indirect run, then t_1 and t_n is a matching parenthesizing transition pair, and $t_2 \dots t_{n-1}$ is a direct run of $\mathcal{A}$. Moreover, if $\mathbf{r}$ is a direct run, then it has a unique decomposition into subruns $\mathbf{r} = \mathbf{r}_1 \mathbf{r}_2 \dots \mathbf{r}_k$, where each $\mathbf{r}_i$ is either a singleton run or an indirect run for $i = 1, \dots, k$, and $k \geq 2$.

Definition 12. *Suppose that $\mathcal{A}$ is a parenthesizing automaton and $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$. The* biposet *of $\mathbf{r}$ is an element of* $\mathrm{SPB}(\Sigma)$ *defined inductively as follows:*

(i) If $\mathbf{r} = (p, \sigma, q)$, then $\mathrm{Biposet}(\mathbf{r}) := \sigma$.

(ii) If $\mathbf{r}$ is a direct run, and $\mathbf{r} = \mathbf{r_1} \mathbf{r_2}$ for some $\mathbf{r_1}, \mathbf{r_2} \in \mathrm{Runs}(\mathcal{A})$, then

- *if* $\mathrm{end}(\mathbf{r_1}) \in H$, *then* $\mathrm{Biposet}(\mathbf{r}) := \mathrm{Biposet}(\mathbf{r_1}) \bullet \mathrm{Biposet}(\mathbf{r_2})$;
- *if* $\mathrm{end}(\mathbf{r_1}) \in V$, *then* $\mathrm{Biposet}(\mathbf{r}) := \mathrm{Biposet}(\mathbf{r_1}) \circ \mathrm{Biposet}(\mathbf{r_2})$.

(iii) If $\mathbf{r}$ is an indirect run $\mathbf{r} = t_1 \mathbf{r}' t_2$, then $\mathrm{Biposet}(\mathbf{r}) := \mathrm{Biposet}(\mathbf{r}')$.

As in Definition 3, the definition of $\mathrm{Biposet}(\mathbf{r})$ is also independent of the choice of factorization in case *(ii)* above.

If $\mathbf{r} = (p_0, \omega_1, p_1)(p_1, \omega_2, p_2) \dots (p_{n-1}, \omega_n, p_n)$ is a run of $\mathcal{A}$, we define the *word* of $\mathbf{r}$ as

$$\mathrm{Word}(\mathbf{r}) := \omega_1' \omega_2' \dots \omega_n',$$

where

$$\omega_i' := \begin{cases} \omega_i & \text{if } \omega_i \in \Sigma, \\ (& \text{if } \omega_i \in \Omega \text{ is an opening parenthesis, and} \\ ) & \text{if } \omega_i \in \Omega \text{ is a closing parenthesis.} \end{cases}$$

The relationship between the term representation of a biposet and the word of a run on that biposet, is given by the following lemma. This is a straightforward consequence of Definition 3, Definition 11 and Definition 12. In the sequel, we write $\mathrm{Type}(q) = \bullet$ if q is a horizontal state, and $\mathrm{Type}(q) = \circ$ if q is a vertical state of an automaton $\mathcal{A}$.

Lemma 13. *Suppose that $\mathcal{A}$ is a parenthesizing automaton, $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$, and $P = \mathrm{Biposet}(\mathbf{r})$.*

(i) $\mathbf{r}$ is singleton or direct run $\Leftrightarrow \mathrm{Type}(\mathrm{start}(\mathbf{r})) = \mathrm{Type}(P)$
$\Leftrightarrow \mathrm{Word}(\mathbf{r}) = P^{tm}$.

(ii) $\mathbf{r}$ is indirect run $\Leftrightarrow \mathrm{Type}(\mathrm{start}(\mathbf{r})) \neq \mathrm{Type}(P) \Leftrightarrow \mathrm{Word}(\mathbf{r}) = \langle P^{tm} \rangle$.

Definition 14. *Suppose that* $P \in \mathrm{SPB}(\Sigma)$ *and* $p, q \in S$. *We say that* $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ has a run on P from p to q, *denoted* $[p, P, q]_{\mathcal{A}}$, *if there is a run* $\mathrm{r} \in \mathrm{Runs}(\mathcal{A})$ *with* $\mathrm{start}(\mathrm{r}) = p$, $\mathrm{end}(\mathrm{r}) = q$, *and* $\mathrm{Biposet}(\mathrm{r}) = P$.

Definition 15. *The sp-biposet* language $L(\mathcal{A})$ accepted by the automaton $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *is defined as*

$$L(\mathcal{A}) \quad := \quad \{\, P \in \mathrm{SPB}(\Sigma) \mid [i, P, f]_{\mathcal{A}} \text{ for some } i \in I \text{ and } f \in F \,\}.$$

An sp-biposet language $L \subseteq \mathrm{SPB}(\Sigma)$ *is called* regular *if there exists a parenthesizing automaton* $\mathcal{A}$ *that accepts it, i.e.,* $L = L(\mathcal{A})$. *Two automata are said to be* equivalent *if they accept the same language.*

The following lemma is a straightforward consequence of Definition 11 and Definition 12.

Lemma 16. *Let* $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *be a parenthesizing automaton, and let* P *be a horizontal sp-biposet, with maximal horizontal decomposition* $P = P_1 \bullet \ldots \bullet P_n$, $(n \geq 2)$.

If $p, q \in H$, *then*

$$[p, P, q]_{\mathcal{A}} \Leftrightarrow \exists r_1, \ldots, r_{n-1} \in H,\ r_0 = p,\ r_n = q \,:\, [r_{i-1}, P_i, r_i]_{\mathcal{A}}\ \forall i = 1, \ldots, n.$$

If $p, q \in V$, *then*

$$[p, P, q]_{\mathcal{A}} \Leftrightarrow \exists\, \langle_k, \rangle_k \in \Omega,\ \exists p', q' \in H \,:\, (p, \langle_k, p'), (q', \rangle_k, q) \in \gamma,\ [p', P, q']_{\mathcal{A}}.$$

Obviously, for vertical sp-biposets there are two analogous statements.

Corollary 17. *If* $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *is a parenthesizing automaton, and* P *is a horizontal biposet, then*

$$\begin{array}{lll} P \in L(A) \Leftrightarrow & \textit{either} & \textit{i)}\ [i, P, f]_{\mathcal{A}},\ \textit{where } i \in I \cap H,\ \textit{and } f \in F \cap H; \\ & \textit{or} & \textit{ii)}\ [r, P, s]_{\mathcal{A}}\ \textit{where } r, s \in H,\ \textit{and } (i, \langle, r), (s, \rangle, f) \in \gamma, \\ & & \quad i \in I \cap V,\ f \in F \cap V,\ \langle, \rangle \in \Omega. \end{array}$$

Again, an analogous statement holds for vertical sp-biposets.

We do not give examples of parenthesizing automata here, but several examples can be found in [6]. The main result concerning sp-biposet languages is the following.

Theorem 18 ([6]). *An sp-biposet language* $L \subseteq \mathrm{SPB}(\Sigma)$ *is recognizable if and only if it is regular if and only if it is* MSO*-definable.*

3.2 Parenthesizing Büchi-automata

Our next task is to define parenthesizing Büchi-automaton so that a language is recognizable if and only if it can be accepted by such an automaton. A straightforward approach would be to use the same accepting device and only extend the notion of run appropriately for the acceptance of languages of infinite biposets, but, as we shall see, this cannot be achieved. Thus, our definition is the following.

Definition 19. *A* parenthesizing Büchi-automaton *is a tuple* $\mathcal{A} := (S, H, V, \Sigma, \Omega, [, \delta, \beta, \gamma, I, F)$, *where* $\mathcal{A}' := (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *is a parenthesizing automaton, called the* underlying parenthesizing automaton *of* $\mathcal{A}$. *And the new components are the following:*

- $[\notin (\Sigma \cup \Omega)$ *is the* separating parenthesis, *and*
- $\beta \subseteq (H \times \{ [\} \times V) \cup (V \times \{ [\} \times H)$ *is the* separating transition relation.

For the sake of simplicity, we will write $[p, P, q]_{\mathcal{A}}$ instead of $[p, P, q]_{\mathcal{A}'}$ if P is an sp-biposet, and $\mathcal{A}'$ is the underlying parenthesizing automaton of the parenthesizing Büchi-automaton $\mathcal{A}$.

Remark 20. It was proved in [18] that if we would like to accept all regular sp-biposet languages, we cannot give a universal upper bound for the number of parentheses used in parenthesizing automata. On the other hand, as we need not to close parentheses of the separating transitions, a single symbol in itself is enough for changing the type of the state at the borders of "finite-infinite" products.

Next, we define when a parenthesizing automaton $\mathcal{A}$ accepts an infinite biposet P from a given state p. For this, we choose Büchi's approach: for acceptance a run must contain a final state r (in certain "outer" positions) infinitely many times. Let $[p, P, r]^{\infty}_{\mathcal{A}}$ denote this fact. Its definition distinguishes two cases and uses induction on the rank of P. Recall that we write $\mathrm{Type}(p) = \bullet$ if p is a horizontal state, and $\mathrm{Type}(p) = \circ$ if p is a vertical state of $\mathcal{A}$. Similarly, $\mathrm{Type}(P) = \bullet$ ($\mathrm{Type}(P) = \circ$) indicates that P is a horizontal (vertical, resp.) biposet.

Definition 21. *Suppose that* $\mathcal{A} = (S, H, V, \Sigma, \Omega, [, \delta, \gamma, \beta, I, F)$ *is a parenthesizing Büchi-automaton,* p *and* r *are in* S, *and* P *is an infinite constructible biposet. We write* $[p, P, r]^{\infty}_{\mathcal{A}}$ *in the following cases.*

i) $\mathrm{Type}(p) = \mathrm{Type}(P)$, *and either*

- *α)* P *can be written as* $P = P_0 * P_1 * P_2 * \ldots$, *where each* P_i *is a finite (not necessarily* $*$*-irreducible) sp-biposet such that* $[p, P_0, r]_{\mathcal{A}}$ *and* $[r, P_i, r]_{\mathcal{A}}$ *for* $i > 0$; *or*
- *β)* $P = P' * P''$, *where* $\mathrm{Rank}(P'') < \mathrm{Rank}(P)$, *and there is a state* $q \in S$ *such that* $[p, P', q]_{\mathcal{A}}$ *and* $[q, P'', r]^{\infty}_{\mathcal{A}}$, *the latter is defined inductively.*

ii) $\mathrm{Type}(p) \neq \mathrm{Type}(P)$, *and there exists a state* $p' \in S$ *such that* $\mathcal{A}$ *has a separating transition* $(p, [, p') \in \beta$, *and* $[p', P, r]^{\infty}_{\mathcal{A}}$ *holds according to case i) above.*

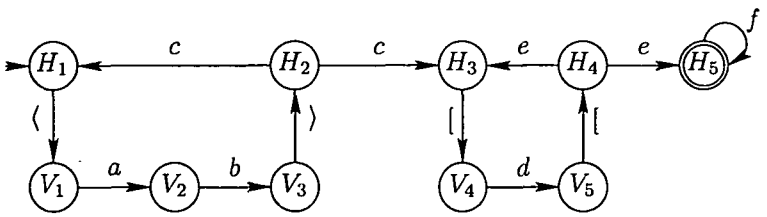


Figure 3: A parenthesizing Büchi-automaton

Definition 22. *A parenthesizing Büchi-automaton $\mathcal{A}$ accepts the following language*

$$L(\mathcal{A}) := \{ P \in \mathrm{ISPB}(\Sigma) \mid [i, P, f]^{\infty}_{\mathcal{A}} \text{ for some } i \in I \text{ and } f \in F \}.$$

Again, a language $L \subseteq \mathrm{ISPB}(\Sigma)$ is regular *if there is a parenthesizing Büchi-automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.*

Similarly to Definition 11, one could also define infinite runs formally. In this case, runs are ω-words over the union of the sets of labeled, parenthesizing and separating transitions. Later we will use the same notations as in the finite case: $\mathrm{Runs}(\mathcal{A})$, $\mathrm{Biposet}(\mathbf{r})$, etc.

Example 23. Figure 3 shows a parenthesizing Büchi-automaton. The horizontal states are those labeled H_i and the vertical states are those labeled V_j for some i and j. There is a single initial state H_1 and a single final state H_5. The angle brackets indicate parenthesizing transitions, while the square brackets represent separating transitions. It is easy to check that this automaton accepts exactly the constructible biposets of the form

$$(a \circ b) \bullet c \bullet (a \circ b) \bullet c \bullet \ldots \bullet (a \circ b) \bullet c \bullet (d \circ (e \bullet (d \circ (e \bullet \ldots (d \circ (e \bullet f \bullet f \bullet f \ldots)) \ldots))))$$

Remark 24. As we mentioned above, we cannot use original parenthesizing automata for acceptance of infinite biposets. First, there is no meaningful definition of closing a parenthesis after the acceptance of an infinite subbiposet. E.g., suppose that $P = Q \bullet R$ is an infinite constructible biposet, where Q is a finite sp-biposet (either horizontal or vertical), and R is an infinite vertical biposet. Now, if there is a finite run $[p, Q, q]_{\mathcal{A}}$ for some horizontal states p and q, we must open a parenthesis by some transition $(q, \langle, r)$ in order to arrive at a vertical state r, from where the acceptance of the infinite vertical biposet R can be started. However, as R is infinite, we have no possibility to close this parenthesis. Second, it can be proved that we must distinguish between the normal and these "non-closable" parenthesizing transitions, otherwise there would be ω-recognizable languages that cannot be accepted by Büchi-automaton. To check this, consider the language $L := \{a \bullet (b \circ c) \bullet d^{\omega\bullet}, a \bullet (b \circ (a \bullet (b \circ c))) \bullet d^{\omega\bullet}, a \bullet (b \circ (a \bullet (b \circ (a \bullet (b \circ c))))) \bullet d^{\omega\bullet}, \ldots\}$.

4 From regularity to recognizability

The fact the regularity implies recognizability is easily follows from the finite-state property of automata.

Theorem 25. *Every regular language of infinite constructible biposets is recognizable.*

Proof. Let L denote a language containing only infinite constructible biposets, i.e., $L \subseteq \mathrm{ISPB}(\Sigma)$. We show how to transform a parenthesizing Büchi-automaton $\mathcal{A} = (S, H, V, \Sigma, \Omega, [, \delta, \gamma, \beta, I, F)$ accepting L into a finite ω-bisemigroup recognizing L analogously to [20].

Recall that $[p, P, q]_{\mathcal{A}}$ means that the automaton $\mathcal{A}$ has a run on the sp-biposet P from state p to state q. Moreover, we write $\mathrm{Type}(p) = \bullet$ if p is a horizontal state, and $\mathrm{Type}(p) = \circ$ if p is a vertical state of $\mathcal{A}$. Suppose that $\mathrm{Type}(p) = \mathrm{Type}(q) = *$, and consider an sp-biposet P. If P is $*$-irreducible, then take $P_1 = P$ and $m = 1$, otherwise let the maximal $*$-decomposition of P be $P = P_1 * P_2 * \ldots * P_m$ for some $m \geq 2$. According to Lemma 16, there are states $p_0 = p, p_1, \ldots, p_m = q$ of type $*$ such that $[p_0, P_1, p_1]_{\mathcal{A}}, [p_1, P_2, p_2]_{\mathcal{A}}, \ldots, [p_{m-1}, P_m, p_m]_{\mathcal{A}}$ hold. Let us write $[p, P, q]_{\mathcal{A}_+}$ to indicate the existence of such states for which $\{p_0, \ldots, p_m\} \cap F \neq \emptyset$. Thus, $[p, P, q]_{\mathcal{A}_+}$ if and only if there is at least one final state among the "outer" states of a possible run between p and q on P.

Next, we define for any $P, Q \in \mathrm{SPB}(\Sigma)$ and $P', Q' \in \mathrm{ISPB}(\Sigma)$

$$P \sim_F Q \text{ iff } \forall p, q \in S : \big([p, P, q]_{\mathcal{A}} \Leftrightarrow [p, Q, q]_{\mathcal{A}} \text{ and } [p, P, q]_{\mathcal{A}_+} \Leftrightarrow [p, Q, q]_{\mathcal{A}_+}\big),$$
$$P' \sim_I Q' \text{ iff } \forall p \in S : \big(\exists r \in F : \ [p, P', r]^{\infty}_{\mathcal{A}} \Leftrightarrow \exists r' \in F : \ [p, Q', r']^{\infty}_{\mathcal{A}}\big).$$

Now one can check that $\sim_F$ and $\sim_I$ are equivalence relations with finitely many equivalence classes. Furthermore, they satisfy the following equalities. If $P_i, Q_i \in \mathrm{SPB}(\Sigma)$, for $i = 1, 2, \ldots$, and $P', Q' \in \mathrm{ISPB}(\Sigma)$, $* \in \{\bullet, \circ\}$, then

$$\begin{aligned}
&P_1 \sim_F Q_1, \ P_2 \sim_F Q_2 && \Rightarrow && P_1 * P_2 \sim_F Q_1 * Q_2, \\
&P_i \sim_F Q_i \text{ for } i > 0 && \Rightarrow && P_1 * P_2 * \ldots \sim_I Q_1 * Q_2 * \ldots, \text{ and} \\
&P_1 \sim_F Q_1, \ P' \sim_I Q' && \Rightarrow && P_1 * P' \sim_I Q_1 * Q'.
\end{aligned}$$

Hence the quotient can be equipped with the structure of an ω-bisemigroup. Finally, the canonical epimorphism of $\omega\mathrm{SPB}(\Sigma)$ onto this quotient accepts $L(\mathcal{A})$. □

5 From recognizability to regularity

In this section, our aim is to prove that every recognizable infinite constructible biposet language is regular, i.e., can be accepted by a parenthesizing Büchi-automaton. First, we observe that every parenthesizing automaton is equivalent to one in normal form, i.e., with a single initial and a single final state.

In the sequel, we assume that no automaton has two opening or closing parenthesizing transitions with the same label. This can easily be achieved by replacing the multiple occurrences of the same parenthesizing transition pair with new transitions using different symbols. We start with the definition of the substitution product of parenthesizing automata.

Definition 26. *Suppose that* $\mathcal{A}_1 = (S_1, H_1, V_1, \Sigma, \Omega, \delta_1, \gamma_1, I_1, F_1)$ *and* $\mathcal{A}_2 = (S_2, H_2, V_2, \Sigma, \Omega, \delta_2, \gamma_2, I_2, F_2)$ *are parenthesizing automata, and either* $p, q \in H_1$ *and* $R, S \subseteq H_2$*; or* $p, q \in V_1$ *and* $R, S \subseteq V_2$*. We assume that* S_1 *and* S_2 *are disjoint. We define the* substitution product *of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *with respect to* p*,* q*,* R *and* S*, as*

$$\mathcal{A}_1 *_{[p \to R, S \to q]} \mathcal{A}_2 := (S_3, H_3, V_3, \Sigma, \Omega_3, \delta_3, \gamma_3, I_1, F_1),$$

where

$$\begin{aligned}
S_3 &:= S_1 \cup S_2, \quad H_3 := H_1 \cup H_2, \quad V_3 := V_1 \cup V_2, \\
\Omega_3 &:= \Omega \cup \{ \langle^{\text{first}}, \rangle^{\text{first}}, \langle^{\text{last}}, \rangle^{\text{last}} \mid \langle, \rangle \in \Omega \}, \\
\delta_3 &:= \delta_1 \cup \delta_2 \\
&\quad \cup \{ (p, a, x) \mid a \in \Sigma, x \in S_2, \exists r \in R : (r, a, x) \in \delta_2 \} \\
&\quad \cup \{ (y, b, q) \mid y \in S_2, b \in \Sigma, \exists s \in S : (y, b, s) \in \delta_2 \}, \\
\gamma_3 &:= \gamma_1 \cup \gamma_2 \\
&\quad \cup \{ (p, \langle^{\text{first}}, x), (y, \rangle^{\text{first}}, z) \mid x, y, z \in S_2, \exists r \in R : (r, \langle, x), (y, \rangle, z) \in \gamma_2 \} \\
&\quad \cup \{ (x, \langle^{\text{last}}, y), (z, \rangle^{\text{last}}, q) \mid x, y, z \in S_2, \exists s \in S : (x, \langle, y), (z, \rangle, s) \in \gamma_2 \}.
\end{aligned}$$

The construction is illustrated in Figure 4. If both R and S are singletons, say $R = \{r\}$ and $S = \{s\}$, then we will write $\mathcal{A}_1 *_{[p \to r, s \to q]} \mathcal{A}_2$ instead of $\mathcal{A}_1 *_{[p \to \{r\}, \{s\} \to q]} \mathcal{A}_2$. The next lemma formulates a key property of the substitution product.

Lemma 27. *Suppose that* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are parenthesizing automata as above,* $p, q \in H_1$*,* $R, S \subseteq H_2$ *and* $\mathcal{A}_3 = \mathcal{A}_1 *_{[p \to R, S \to q]} \mathcal{A}_2$*. Moreover,* $p \neq q$*, and no transition of* $\mathcal{A}_1$ *arrives at* p *or starts from* q*. Then, for every* $P \in \mathrm{SPB}(\Sigma)$

$$[p, P, q]_{\mathcal{A}_3} \Leftrightarrow \quad \begin{array}{ll} \textit{either} & \textit{i)}\ [p, P, q]_{\mathcal{A}_1}, \\ \textit{or} & \textit{ii)}\ \exists r \in R, \exists s \in S\ :\ [r, P, s]_{\mathcal{A}_2} \textit{ and } P \textit{ is horizontal.} \end{array}$$

Proof. $[p, P, q]_{\mathcal{A}_3}$ implies that $P = \mathrm{Biposet}(\mathbf{r})$ for a run $\mathbf{r} = t_1 \dots t_m \in \mathrm{Runs}(\mathcal{A}_3)$ with $\mathrm{start}(t_1) = p$ and $\mathrm{end}(t_m) = q$.

If $\mathrm{end}(t_1) \in S_1$, then $\mathbf{r} \in \mathrm{Runs}(\mathcal{A}_1)$ also holds. This follows from the definition of $\mathcal{A}_3$ and from the fact that no transition arrives at p. Thus, case (i) is true.

If $\mathrm{end}(t_1) \in S_2$, then we can modify $\mathbf{r}$ to obtain a run $\mathbf{r}' := t'_1 \dots t'_m \in \mathrm{Runs}(\mathcal{A}_2)$ with $\mathrm{Biposet}(\mathbf{r}') = P$ and $\mathrm{start}(t'_1) \in R$, and $\mathrm{end}(t'_m) \in S$. Indeed, if t_1 is of the form $t_1 = (p, a, x)$, $a \in \Sigma$, then there is an $r \in R$ such that $t'_1 := (r, a, x) \in \delta_2$. Similarly, if $t_m = (y, b, q)$, $b \in \Sigma$, then there is an $s \in S$ such that $t'_m := (y, b, s) \in \delta_2$. On the other hand, if t_1 or t_m involves parenthesis, e.g., $t_1 = (p, \langle^{\text{first}}, x)$, then

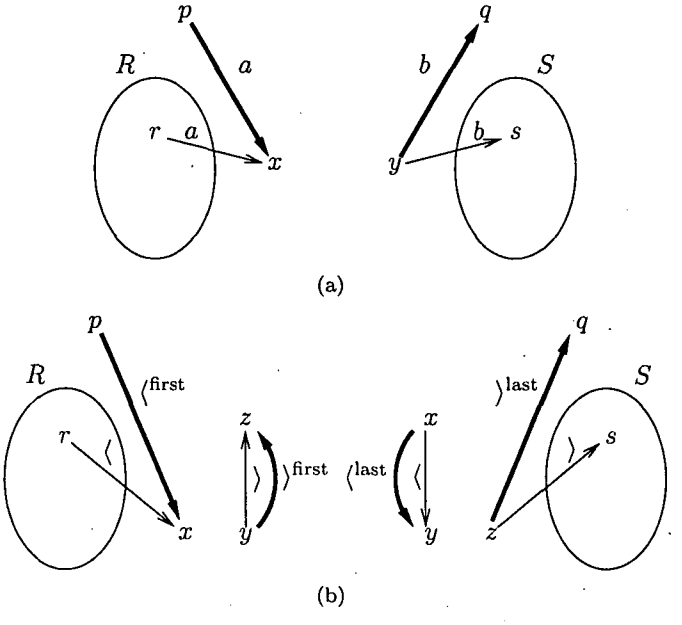


Figure 4: The labeling (a) and the parenthesizing (b) transitions used in Definition 26. The thin arrows represent the original transitions, and the thick arrows the new ones.

there is a closing transition partner of t_1, say $t_i = (y, \rangle^{\text{first}}, z)$, where $i < m$, and $x, y, z \in S_2$. Moreover, by definition, there is an $r \in R$ such that $t'_1 := (r, \langle, x)$, $t'_i := (y, \rangle, z) \in \gamma_2$. Similarly, if $t_m = (z, \rangle^{\text{last}}, q)$, then there is an index $j > 1$ such that $t_j = (x, \langle^{\text{last}}, y)$. So, we can set $t'_j := (x, \langle, y)$ and $t'_m := (z, \rangle, s) \in \gamma_2$ for a suitable $s \in S$. So far we have defined t'_k for at most four k-s. Let $t'_k := t_k$ for all other k-s (note that $t_k \in \delta_2 \cup \gamma_2$ in these cases). Now $\text{Biposet}(\mathbf{r}') = P$, $\text{start}(t'_1) = r \in R$, and $\text{end}(t'_m) = s \in S$ implies $[r, P, s]_{\mathcal{A}_2}$. Since $\langle^{\text{first}}$ and $\rangle^{\text{last}}$ do not match, t_1 and t_m cannot be a matching parenthesizing transition pair. Hence, $\mathbf{r}$ is a direct run, and so is $\mathbf{r}'$. Consequently, by Lemma 13, $r, s \in H$ implies that P is horizontal. Thus, (ii) holds.

For the converse direction, it is obvious that $[p, P, q]_{\mathcal{A}_1}$ implies $[p, P, q]_{\mathcal{A}_3}$. Assume that $ii)$ holds, so $P = \text{Biposet}(\mathbf{r})$ for $\mathbf{r} = t_1 \dots t_m \in \text{Runs}(\mathcal{A}_2)$ with $\text{start}(t_1) = r \in R$ and $\text{end}(t_m) = s \in S$. By Lemma 13, as P is horizontal and r and s are in H, $\mathbf{r}$ is a direct run. Hence, t_1 and t_m is not a matching parenthesizing transition pair. Thus, it is possible to replace both $\langle$ and $\rangle$ with $\langle^{\text{first}}$ and $\rangle^{\text{last}}$, in the first and in the last transitions, if necessary. We can also substitute their closing and opening partners by $\rangle^{\text{first}}$ and $\langle^{\text{last}}$, if needed. Therefore, the construction of $\mathcal{A}_3$ ensures that $[p, P, q]_{\mathcal{A}_3}$ holds. □

Definition 28. *We say that a parenthesizing automaton is in* horizontal normal form *if it has a single initial state i_h, and a single final state f_h, and both i_h and f_h are horizontal states, moreover, there is no transition into i_h or from f_h. Automata in* vertical normal form *can be defined accordingly.*

Lemma 29. *For every parenthesizing automaton $\mathcal{A}$, there exists an equivalent parenthesizing automaton $\mathcal{A}_h$ in horizontal normal form and an equivalent parenthesizing automaton $\mathcal{A}_v$ in vertical normal form.*

Proof. First we prove that for every parenthesizing automaton $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ there exists a parenthesizing automaton $\mathcal{A}^{\cap\mathcal{H}}$ in horizontal normal form that accepts exactly the horizontal biposets accepted by $\mathcal{A}$, i.e.,

$$L(\mathcal{A}^{\cap\mathcal{H}}) = L(\mathcal{A}) \cap \mathcal{H},$$

where $\mathcal{H}$ denotes the set of all horizontal biposets. Indeed, let

$$T := \{\, (s,t) \mid \exists i \in I \cap V, \exists f \in F \cap V, \exists \langle, \rangle \in \Omega \;:\; (i, \langle, s), \; (t, \rangle, f) \in \gamma \,\},$$

and assume that $T = \{\, (s_1,t_1), (s_2,t_2), \ldots, (s_n,t_n) \,\}$. Moreover, let $\mathcal{A}_0$ be the automaton without transitions, with two states only, an initial horizontal state i_h, and a final horizontal state f_h.

Now, with the help of the substitution product, we define

$$\begin{aligned} \mathcal{A}_1 &:= \mathcal{A}_0 *_{[i_h \to I\cap H, F\cap H \to f_h]} \mathcal{A}, \\ \mathcal{A}_{k+1} &:= \mathcal{A}_k *_{[i_h \to s_k, t_k \to f_h]} \mathcal{A} \quad \text{for } k = 1, \ldots, n, \\ \mathcal{A}^{\cap\mathcal{H}} &:= \mathcal{A}_{n+1}. \end{aligned}$$

Using Lemma 27 and Corollary 17, it is straightforward to check that $L(\mathcal{A}^{\cap\mathcal{H}}) = L(\mathcal{A}) \cap \mathcal{H}$, as claimed.

Similarly, there is an automaton $\mathcal{A}^{\cap\mathcal{V}}$ in vertical normal form which accepts exactly the vertical biposets accepted by $\mathcal{A}$. Let i_v and f_v denote the (single) initial and final vertical states of $\mathcal{A}^{\cap\mathcal{V}}$.

Now, we can construct $\mathcal{A}_h$ by taking the disjoint union of $\mathcal{A}^{\cap\mathcal{H}}$ and $\mathcal{A}^{\cap\mathcal{V}}$ and adding two new parenthesizing transitions, $(i_h, \{, i_v)$ and $(f_v, \}, f_h)$, where $\{$ and $\}$ is a new pair of parentheses. Of course, we do not regard i_v and f_v as initial and final states any longer. In order to accept the singleton biposets, we also define (i_h, σ, f_h) for each singleton biposet $\sigma \in L(\mathcal{A})$. As $\mathcal{A}^{\cap\mathcal{H}}$ accepts all horizontal, and $\mathcal{A}^{\cap\mathcal{V}}$ all vertical biposets of $L(\mathcal{A})$, the resulting automaton is equivalent to $\mathcal{A}$. Again, $\mathcal{A}_v$ can be defined symmetrically. □

Now, we are ready to prove the converse of Theorem 25.

Theorem 30. *Every recognizable language of infinite constructible biposets is regular.*

Proof. Suppose that a language $L \subseteq \mathrm{ISPB}(\Sigma)$ of infinite constructible biposets is recognized by a morphism $\varphi : \omega\mathrm{SPB}(\Sigma) \to \mathcal{B}$, where $\mathcal{B} = (B_F, B_I)$ is a finite ω-bisemigroup, and $L = \varphi^{-1}(F)$ for some $F \subseteq B_I$.

Let us call an element $e \in B_F$ *horizontally idempotent* if it is idempotent with respect to the horizontal product, i.e., $e \bullet e = e$. Similarly, e is said to be *vertically idempotent* if $e \circ e = e$. This notion is important for the fact that every primitive biposet $P_0 * P_1 * \ldots$ can be written in the form $P'_0 * P'_1 * \ldots$, where $\varphi(P'_0) = b$ and $\varphi(P'_i) = e$ for all $i > 0$, where e is a $*$-idempotent element of B_F. This follows from an application of the Ramsey-theorem, cf. [20]. We can even assume that $b = b * e$, but we do not need this now.

Thus, if we omit P'_0 from the above biposet, then the remaining primitive biposet is $P'_1 * P'_2 * \ldots$, where $\varphi(P'_i) = e$ for all $i > 0$. Let us call those biposets that can be written in such a form *e-$*$-primitive*.

For a given e and $*$, the set of all e-$*$-primitive biposets is easy to accept by a parenthesizing Büchi-automaton $\mathcal{A}_{e,*}$ constructed as follows. Since $\varphi^{-1}(e)$ is a recognizable set of finite sp-biposets, it is also regular by Theorem 18. So there is a parenthesizing (finite) automaton $\mathcal{A}$ accepting $\varphi^{-1}(e)$. Moreover, it can be assumed that $\mathcal{A}$ is in $*$-normal form (i.e. in horizontal normal form if $* = \bullet$, or in vertical normal form if $* = \circ$). Thus, $\mathcal{A}$ has a single initial state i and a single finite state f, both of them are of type $*$. We can transform $\mathcal{A}$ into $\mathcal{A}_{e,*}$ just by merging i and f. We will refer to this fused state as the *basic state* of $\mathcal{A}_{e,*}$. Now, it is obvious that if we regard $\mathcal{A}_{e,*}$ as a Büchi-automaton with its basic state as the only initial and final state, it accepts exactly the e-$*$-primitive biposets.

Recall that according to (1) the normal form of a constructible biposet is

$$P_1 *_1 (P_2 *_2 \ldots (P_k *_k (Q_1 *_{k+1} Q_2 *_{k+1} \ldots))).$$

We can assume that except for a finite factor Q', the biposet $Q_1 *_{k+1} Q_2 *_{k+1} \cdots$ is e-$*_{k+1}$-primitive for some $*_{k+1}$-idempotent e. Thus, we only need to build our automaton in a way that it can also process the finite "introductory slice" $P_1 *_1 (P_2 *_2 (\ldots P_k *_k (Q' *_{k+1}$ before the e-$*_{k+1}$-primitive tail.

Assume that $B_I = \{t_1, t_2, \ldots, t_m\}$. We start to construct a Büchi-automaton $\mathcal{A}$ from the horizontal states H_0, vertical states V_0, with separating transitions β, where

$$\begin{aligned} H_0 &:= \{t_1^\bullet, t_2^\bullet, \ldots, t_m^\bullet\}, \\ V_0 &:= \{t_1^\circ, t_2^\circ, \ldots, t_m^\circ\}, \text{ and} \\ \beta &:= \{(t_i^\bullet, [, t_i^\circ), (t_i^\circ, [, t_i^\bullet) \mid i = 1, \ldots, m\}. \end{aligned}$$

For all $b \in B_F$, there is a parenthesizing (finite) automaton $\mathcal{A}_b$ recognizing $\varphi^{-1}(b)$. Similarly as before, we will incorporate these finite automata into $\mathcal{A}$.

More precisely, if p and q are states of $\mathcal{A}$ of the same type, say $*$, then one can take a copy of $\mathcal{A}_b$ in $*$-normal form and merge its initial state i and final state f with the states p and q of $\mathcal{A}$, respectively. In the sequel, we refer to this construction as *extension* of $\mathcal{A}$ (between p and q) by $\varphi^{-1}(b)$. Let us denote it by

$$p \overset{\varphi^{-1}(b)}{\Longrightarrow} q.$$

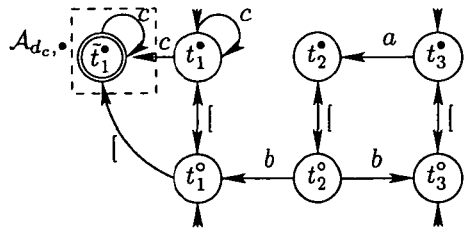


Figure 5: A Büchi-automaton accepting the recognizable language of Example 8

We need to add the following extensions to $\mathcal{A}$:

$$t_i^* \overset{\varphi^{-1}(b)}{\Longrightarrow} t_j^* \quad \text{for all } t_i = b * t_j, \quad b \in B_F,\ t_i, t_j \in B_I, * \in \{\bullet, \circ\}.$$

Now we obviously have

$$[t_i^*, P, t_j^*]_{\mathcal{A}} \Leftrightarrow t_i = \varphi(P) * t_j \text{ for any } P \in \mathrm{SPB}(\Sigma).$$

Furthermore,

$$[t_i^*, P_1 *_1 (P_2 *_2 (\ldots P_{k-1} * (P_k *_k, t_j^*]_{\mathcal{A}} \Leftrightarrow t_i = \varphi(P_1) *_1 (\varphi(P_2) *_2 (\ldots \varphi(P_k) *_k t_j)),$$

where the left hand side abbreviates the first part of an infinite run of $\mathcal{A}$ (as a Büchi-automaton) on an infinite biposet beginning with $P_1 *_1 (P_2 *_2 \ldots (P_k *_k$.

Next, add an instance of $\mathcal{A}_{e,*}$ for each $*$-idempotent element e in B_F to $\mathcal{A}$, and assure the reachability of the new components by adding some new transitions. In more detail, for $\mathcal{A}_{e,*}$, consider $t := e^{\omega *}$ and duplicate each transition arriving at t^* using the same source and label, but with the target of the basic state of $\mathcal{A}_{e,*}$ instead of t^*.

Our last task is to settle the initial and the final states. Let the initial states be the states $t^\bullet$ and t° for each $t \in F$, and set the basic states of the components $\mathcal{A}_{e,*}$-s as final states.

Finally, it can be argued by induction on the rank of the biposets that in fact this automaton accepts $L = \varphi^{-1}(F)$. We omit the formal proof. □

Example 31. Figure 5 shows a parenthesizing Büchi-automaton that was constructed according to the proof of Theorem 30 from the morphism φ, the ω-bisemigroup B, and the set $F \subseteq B$ of Example 8. We omitted the two shrink states $\underline{0}^\bullet, \underline{0}^\circ$ that correspond to the infinite zero element, and also the transitions pointing to them. We should admit that this example represents a somewhat special case, since, for every $x \in B_F$ the extension by $\varphi^{-1}(x)$ is a single transition, and there is only one idempotent in B_F. Of course, in the general case the constructed automaton can have a more complex structure.

6 From regularity to MSO-definability

In this section, we prove the equivalence of regularity and MSO-definability. First of all, it is not hard to demonstrate that MSO-definability implies recoginzability, and hence regularity. This can be shown by formula induction using the closure properties of the recognizable sets, more precisely, the closure under Boolean operations and direct letter-to-letter morphisms. See Chapter III.1 of Straubing [21] for a similar argument. Thus, we have the following theorem.

Theorem 32. *Every* MSO*-definable language of infinite constructible biposets is regular.*

The rest of the paper is devoted to the converse of the previous theorem:

Theorem 33. *Every regular language of infinite constructible biposets is* MSO*-definable.*

Before the proof, let us introduce a few definitions and lemmas.

The notion of clan is one of our key definition, that can be easily adapted from the theory of 2-structures [2] and texts [4, 13]. If $(P, <_h, <_v, \lambda)$ is a finite or infinite constructible sp-biposet, a subset X of P is said to be a *clan* of P if for all $x, y \in X$, $z \in P \setminus X$ and for each relation $\rho \in \{<_h, <_v, >_h, >_v\}$

$$x \rho z \Leftrightarrow y \rho z.$$

Two clans X and Y *overlap* if $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$ and $Y \setminus X \neq \emptyset$. A clan is called *prime clan* if it does not overlap with any other clan. A clan of P is a *proper clan* if it is neither a singleton, nor equal to P.

Example 34. In the biposet of Example 5, the clans of P are the following: the singletons, P, $\{1,2,3,4\}$, $\{2,3,4\}$, $\{3,4\}$, $\{2,3,4,5,6\}$ and $\{5,6\}$. Since, only $\{1,2,3,4\}$ and $\{2,3,4,5,6\}$ overlap, the other clans are prime clans as well. Thus, the proper prime clans are $\{2,3,4\}$, $\{3,4\}$ and $\{5,6\}$. As we will see later in Lemma 38, these are the sets which are surrounded by parentheses in the term representation of P.

The proof of the following lemma is trivial and is left to the reader.

Lemma 35. *The property of being a clan, a prime clan or a proper prime clan can be expressed in* MSO *logic.*

Lemma 36. *If $P = (P, <_h, <_v, \lambda)$ is a (finite or infinite) constructible biposet, then $<_h \cup <_v$ is a linear order on P.*

Proof. By induction on the construction of P. □

In the sequel, let $<$ denote the $<_h \cup <_v$ relation, and we interpret the functions $+$ and $-$ also according to this relation.

By definition, clans form sections (or intervals) with respect to $<$. That is, if X is a clan then $x \in X$, $y \in X$ and $x < z < y$ imply $z \in X$. Thus, we can talk about *prefix* and *suffix* relations among the clans of P. Formally,

$$\begin{aligned} \mathrm{Prefix}(X,Y) &:= X \subsetneq Y \wedge \forall x \forall y \big(y < x \wedge X(x) \wedge Y(y) \rightarrow X(y)\big); \\ \mathrm{Suffix}(X,Y) &:= X \subsetneq Y \wedge \forall x \forall y \big(y > x \wedge X(x) \wedge Y(y) \rightarrow X(y)\big), \end{aligned}$$

where $X \subsetneq Y$ means that X is a proper subset of Y. Thus, under prefix and suffix relations we always mean proper prefix and suffix.

Recall that P^{tr} denotes the tree representation of P. Two (or more) subtrees of a tree are said to be *sibling subtrees* if their roots have the same parent.

Lemma 37. *Suppose that P is a (finite or infinite) constructible biposet and X is a subset of P, then*

(i) X is a clan of P if and only if there are consecutive sibling subtrees in P^{tr} such that X is exactly the union of the sets of leaves of these subtrees;

(ii) X is a prime clan of P if and only if X is the set of leaves of a single subtree of P^{tr}.

Proof. We start with the proof of case (i). The necessity of the condition is based on the following observation. Suppose that x and y are vertices of P. As we mentioned earlier, we can regard them as two leaves in the tree representation P^{tr}. The (horizontal or vertical) type of the order relation between x and y is solely determined by the label of their lowest common ancestor node. For this reason, let u denote the lowest common ancestor of x and y. If the label of u in P^{tr} is $\bullet$, then $x <_h y$ or $y <_h x$; if the label is $\circ$, then x and y are ordered vertically. We can also easily decide whether x is less or greater than y. Consider u_x and u_y, the children of u that are ancestors of x and y, respectively. Now, x is less than y if and only if u_x is less than u_y according to the order of the children nodes at u. It follows that if a subset X of P satisfies the condition of (i), then it also fulfills the requirements of being a clan. Indeed, the order relation between a vertex x from X and a vertex y outside X is independent of the choice of x from X.

For the converse direction, suppose, on the contrary, that X is a clan, but the condition does not hold. First let v denote the lowest common ancestor node of the vertices of X. Now consider those children of v that have leaves in X, and then take the subtrees generated by them. The condition can be violated in two ways. Either these subtrees are not consecutive or there is a subtree that has leaves both from X and $P \setminus X$. In the first case, it is straightforward to show that X is not a clan. In the second case, consider a child u of v that has a leaf x in X and has a leaf z in $P \setminus X$ as well. We can even assume that x and z are descendant of different children of u, so their lowest common ancestor is u. Moreover, there must also be a vertex y in X whose lowest common ancestor with x is v, otherwise the lowest common ancestor of the set X could not be v. But, x and z are related by an order of type determined by the label of u, while y and z are related by an order of type determined by the label of v. As u is a child of v, their labels in P^{tr} are different.

Consequently, the types of the order relations between x and z and between y and z are also different. This contradicts to our assumption that X is a clan.

Now case (ii) easily follows from case (i), since if X consists of the leaves of two or more (but not all) subtrees of a given node, then overlapping clans can be constructed by (i) showing that X is not prime. □

The following lemma is important for a later proof. It connects the various representations of biposets with the use of parentheses of automata. Recall that $\widehat{\Sigma} = \Sigma \cup \{ \bullet, \circ, \langle, \rangle \}$.

Lemma 38. *For any $P \in \mathrm{SPB}(\Sigma)$, $X \subseteq P$, parenthesizing automaton $\mathcal{A}$, and $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$ with $\mathrm{Biposet}(\mathbf{r}) = P$, the following statements are equivalent:*

(i) X is a proper prime clan of P.

(ii) X is the set of leaves of a proper subtree of P^{tr}.

(iii) P^{tm} can be written as $P^{tm} = u\langle X^{tm}\rangle v$, where $u, v \in \widehat{\Sigma}^*$, and the subword X^{tm} above corresponds to those vertices of P that are in X.[1]

(iv) $\mathbf{r}$ is of the form $\mathbf{r} = \mathbf{r_1} t_1 \mathbf{r_x} t_2 \mathbf{r_2}$, where $\mathbf{r_1 r_2} \neq \varepsilon$, t_1 and t_2 form a matching parenthesizing transition pair in $\mathbf{r}$, and $\mathbf{r_x}$ denotes the direct subrun of $\mathbf{r}$ on the vertices of X.

Proof. The equivalence of (i) and (ii) follows from Lemma 37. The equivalence of (ii) and (iii) is obvious, it expresses a usual correspondence between the term and the tree representations. Finally, the equivalence of (iii) and (iv) is a consequence of Lemma 13. □

Now we are ready to start the proof of the main theorem of this section.

Proof of Theorem 33. For sp-biposet languages, the equivalence of MSO-definability and recognizability (and hence regularity) directly follows from an analogous equivalence result on text languages shown by Hoogeboom and ten Pas [13]. See also [6] about the relationship between texts and biposets. Even though, here we outline an alternative proof of this fact, since it will serve as the base for the proof of the infinite case. Our argument does not rely on the equivalence of recognizability and MSO-definability of finite binary trees, but shows how the operations of parenthesizing automata can be described by logical formulas. We start with the finite case, i.e., with the case of sp-biposets, and explain the necessary changes for the infinite case later.

Let $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ be a parenthesizing automaton accepting an sp-biposet language L. Our aim is to construct an MSO-formula φ for which $L_\varphi = L$.

The proof of Lemma 29 implies that we may assume that $\mathcal{A}$ accepts via direct and singleton runs only. Therefore, it is sufficient to construct a formula $\varphi_{i,f}$ which expresses the fact that $\mathcal{A}$ has a direct run from an initial state i to a final state f.

[1] Note that the subword X^{tm} can also appear at other places in the word P^{tm}.

We use second order variables for storing information about the states of the runs. In more detail, two types of monadic second order variables are used. First, let X_s be a variable for each state s in S, and let $Z_{\langle j \rangle_j}$ denote a variable for each pair of parentheses in Ω. Formally,

$$\mathrm{Var}_{\mathcal{A}} := \{ X_s \mid s \in S \} \cup \{ Z_{\langle j \rangle_j} \mid \langle_j, \rangle_j \in \Omega \}.$$

The general form of $\varphi_{i,f}$ is the following

$$\varphi_{i,f} := \exists X_{s_1} \exists X_{s_2} \ldots \exists X_{s_m} \exists Z_{\langle 1 \rangle_1} \exists Z_{\langle 2 \rangle_2} \ldots \exists Z_{\langle n \rangle_n} \psi_{i,f},$$

where, $\psi_{i,f}$ expresses the fact that the values of our variables in fact encode a direct run of $\mathcal{A}$ from i to f.

We need to check three conditions. First, the run must start from i. Second, it must end in f. Third, we need to verify that $\mathcal{A}$ has correct transitions everywhere between the states indicated by the variables. We handle the labeled and the parenthesizing transitions of the run separately.

For labeled transitions, the usual technique (cf. [21]) can be applied, that is, the intended meaning of $x \in X_s$ is that $\mathcal{A}$ reads position x in state s. The storage of parenthesizing transitions is more involved. Fortunately, by Lemma 38 the use of parentheses is always around proper prime clans. But we also need to arrange a unique position in the biposet for each matching parenthesizing transition pair used during the run.

For this purpose, the following rule can be applied. If a proper prime clan is a prefix of an other proper prime clan, then let the *designated position* be its last position, otherwise let the designated position be its first position. Thus, the statement that z is the designated position of a proper prime clan X, can be expressed as:

$$\begin{aligned} \mathrm{Dp}(z, X) \;:=\; & \big[\neg \exists Y \big(\mathrm{PPC}(Y) \wedge \mathrm{Prefix}(X, Y) \big) \wedge \mathrm{First}(z, X) \big] \\ & \vee \big[\exists Y \big(\mathrm{PPC}(Y) \wedge \mathrm{Prefix}(X, Y) \big) \wedge \mathrm{Last}(z, X) \big], \end{aligned}$$

where $\mathrm{PPC}(Y)$ states that Y is a proper prime clan, and $\mathrm{First}(z, X)$ ($\mathrm{Last}(z, X)$) is true if and only is z is the first (last, resp.) position of the clan X.

Now, it can be verified that the prime property implies that the designated positions of any two proper prime clans do not coincide.

Lemma 39. *Different proper prime clans have different designated positions.*

Proof. For a contradiction, assume that X and Y are different proper prime clans, but z is their common designated position. If z is the first position of both X and Y, then either X is a prefix of Y, or Y is a prefix of X, in contradiction with the definition of designated position. If z is the first position of one clan and the last position of the other clan, then X and Y overlap, leading to a contradiction again. Finally, assume that z is the last position of both X and Y, and $X \subseteq Y$. By definition, there is a proper prime clan X' such that X is a prefix of X'. Therefore, in this case, Y and X' overlap, again a contradiction. □

Proof of Theorem 33, continued. If x is a position, then the intended meaning of $Z_{\langle_j,\rangle_j}(x)$ is that $\mathcal{A}$ uses parentheses $\langle_j,\rangle_j$ (more precisely the unique pair of transitions labeled $\langle_j$ and $\rangle_j$) before and after processing the proper prime clan whose designated position is x.

As usual, we require that every position belongs to exactly one X_s:

$$\psi_1 := \forall x\Big[\bigvee_{s\in S} X_s(x) \wedge \bigwedge_{\substack{q_1,q_2\in S,\\ q_1\neq q_2}} \big(\neg X_{q_1}(x)\vee\neg X_{q_2}(x)\big)\Big].$$

Moreover, the designated positions of proper prime clans must also belong to a unique set $Z_{\langle_i\rangle_i}$:

$$\psi_2 := \forall x\Big[\exists X\big(\mathrm{PPC}(X)\wedge \mathrm{Dp}(x,X)\big) \rightarrow \bigvee_{\langle_j,\rangle_j\in\Omega} Z_{\langle_j\rangle_j}(x) \wedge \bigwedge_{\substack{\langle_j,\rangle_j\in\Omega,\\ \langle_k,\rangle_k\in\Omega,\\ j\neq k}} \big(\neg Z_{\langle_j\rangle_j}(x)\vee\neg Z_{\langle_k\rangle_k}(x)\big)\Big].$$

We know that for all positions x, the state of $\mathcal{A}$ before processing this position is indicated by a unique state p such that $x\in X_p$. Moreover, q, *the state after reading position* x, can be computed as follows. First we observe whether P has a proper prime clan that ends at x. If so, then we determine the smallest such clan, and q is the starting state of the closing parenthesizing transition of that clan. Of course, this state can be determined by observing the designated position of the clan. If there is no proper prime clan that ends at x, then q is indicated at the position $x+1$ or at the greatest prime clan that starts at $x+1$. Besides, if x is the last position, then q must be the final state of the run. Finally, we can check whether $\mathcal{A}$ in fact has a labeled transition with the label of x between p and q. We should perform this verification for every position x. The precise algorithm of this computation and the way of converting it to an MSO-formula are presented in the appendix.

We can also check the correctness of the parenthesizing transitions by a similar procedure. For all proper prime clans, we compute four states of the encoded run: the states before and after the opening, and before and after the closing parenthesizing transitions around the clan. In the pseudocode presented in the appendix, these states are denoted by *ob*, *oe*, *cb* and *ce*. Then, it is straightforward to check whether $\mathcal{A}$ has a parenthesizing transition pair between the computed states, and whether the labels of these transitions are indicated at the designated position of the clan. It is also a nontrivial computation, since we must take into consideration various inclusion relations of the clans. For more detail, see the appendix again.

Finally, note that the algorithm of verification can be transformed into an MSO-formula ψ_3. Hence we can write $\psi_{i,f}$ as $\psi_{i,f} := \psi_1\wedge\psi_2\wedge\psi_3$. This completes the proof for finite constructible biposets.

We now turn to a brief discussion of the infinite case. Here we only describe the necessary changes compared to the finite case. First, the adaptation of Lemma 38 for infinite constructible biposets is the following. Note that we must distinguish between finite and infinite clans, but this distinction is in parallel to the use of the parenthesizing and separating parentheses.

Lemma 40. *For any $P \in \mathrm{ISPB}(\Sigma)$, $X \subseteq P$, parenthesizing Büchi-automaton $\mathcal{A}$, infinite run $\mathbf{r} \in \mathit{Runs}(\mathcal{A})$ with $\mathrm{Biposet}(\mathbf{r}) = P$, the following statements are equivalent:*

(i) X is a finite proper prime clan of P.

(ii) X is the set of leaves of a finite proper subtree of P^{tr}.

(iii) P^{tm} can be written as $P^{tm} = u\langle X^{tm}\rangle v$, where $u, v \in \widehat{\Sigma}'^*$, and the subword X^{tm} above corresponds to those vertices of P that are in X.

(iv) $\mathbf{r}$ is of the form $\mathbf{r} = \mathbf{r}_1 t_1 \mathbf{r}_\mathbf{x} t_2 \mathbf{r}_2$, where t_1 and t_2 is a matching parenthesizing transition pair, and $\mathbf{r}_\mathbf{x}$ denotes the direct subrun of $\mathbf{r}$ on the vertices of X.

Moreover, the following statements are also equivalent.

(i') X is an infinite proper prime clan of P.

(ii') X is the set of leaves of an infinite proper subtree of P^{tr}.

(iii') P^{tm} can be written as $P^{tm} = u[X^{tm}$, where $u \in \widehat{\Sigma}'^*$, and the subword X^{tm} above corresponds to those vertices of P that are in X.

(iv') $\mathbf{r}$ is of the form $\mathbf{r} = \mathbf{r}_1 t \mathbf{r}_\mathbf{x}$, where $\mathbf{r}_1 \neq \varepsilon$, t is a separating transition of $\mathcal{A}$, and $\mathbf{r}_\mathbf{x}$ denotes the direct subrun of $\mathbf{r}$ on the vertices of X.

Proof of Theorem 33, completed. It is trivial that we can express the finiteness of clans, as

$$\mathrm{Finite}(X) \quad := \neg\exists z\, \mathrm{Last}(z, X).$$

Hence, we can easily locate the separating transitions and check their correctness, as well. Furthermore, we have no trouble formulating the acceptance condition: a finite state has to appear infinitely often as outer state of the encoded run. We leave the reader to verify the correctness of the formulas bellow.

$$\psi_{\mathrm{acc}} := \bigvee_{f \in F} \forall z\, \exists X \Big[\mathrm{MaxFiniteClan}(X) \wedge \mathrm{OuterState}_f(X) \wedge \forall x \big(\mathrm{Last}(x, X) \rightarrow (z < x) \big) \Big];$$

$$\mathrm{MaxFiniteClan}(X) := \mathrm{Finite}(X) \wedge \mathrm{Clan}(X) \wedge \neg\exists Y \big(\mathrm{Finite}(Y) \wedge \mathrm{Clan}(Y) \wedge X \subsetneq Y \big);$$

$$\begin{aligned}\mathrm{OuterState}_f(X) := {} & \big[\,\mathrm{Singleton}(X) \wedge \exists x\,\big(X(x) \wedge X_f(x)\big)\,\big] \\ & \vee \big[\,\mathrm{PPC}(X) \wedge \bigvee_{\substack{(f,\langle_k,p)\in\gamma \\ \langle_k\in\Omega,p\in S}} \exists z\,\big(\mathrm{Dp}(z,X) \wedge Z_{\langle_k\rangle_k}(z)\big)\,\big].\end{aligned}$$

Of course, here the formulas $\mathrm{Finite}(X)$, $\mathrm{Clan}(X)$ and $\mathrm{Singleton}(X)$ have their expected meanings. □

Finally, we summarize the main results of the paper.

Theorem 41. *Let $L \subseteq \mathrm{ISPB}(\Sigma)$. Then L is recognizable if and only if L is regular if and only if L is* MSO*-definable.*

Acknowledgement. The author would like to thank the anonymous referees for their numerous valuable comments and suggestions.

Appendix

In this appendix, we give the detailed algorithm of verification of the correctness of encoded runs. And we briefly describe how to build formula ψ_3 that realizes the algorithm.

Suppose that $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ is a parenthesizing automaton, $i \in I$ and $f \in F$. Recall that $\mathrm{Var}_{\mathcal{A}} = \{ X_{s_i} \mid s_i \in S \} \cup \{ Z_{\langle_i\rangle_i} \mid \langle_i,\rangle_i \in \Omega \}$. Let $P = (P, <_h, <_v, \lambda) \in \mathrm{SPB}(\Sigma)$ denote an sp-biposet, and assume that η is an evaluation of the monadic second order variables, i.e.,

$$\eta : \mathrm{Var}_{\mathcal{A}} \to \mathcal{P}(P),$$

where $\mathcal{P}(P)$ denotes the power-set of P. Moreover, assume that P with η satisfies formulas ψ_1 and ψ_2 on page 789.

The following algorithm decides whether η encodes a direct run of $\mathcal{A}$ on P that starts from i and ends in f. For the sake of simplicity, we write X_j instead of $\eta(X_j)$. Moreover, in the names of the procedure calls below, "Clan" always means a proper prime clan of P.

Unfortunately, in the definition of function NEXTSTATE a difficulty arises. As $\mathcal{A}$ is nondeterministic, for a given position x and $s \in S$, there can be more than one t such that $(s, \lambda(x), t) \in \delta$ holds. But when we convert our algorithm into an MSO-formula, we only need to test whether $\mathrm{NEXTSTATE}(x) = t$ holds, which resolves the problem.

The pseudocode in Lines 1–10 verifies that the run starts from i and ends in f. The code in Lines 11–22 checks the correctness of the labeled transitions, while Lines 23–49 verify the parenthesizing transitions. The proof of correctness of the algorithm is omitted, but Figures 6–9 should help the reader to establish it.

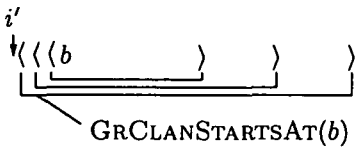


Figure 6: The computation of i' in Line 3.

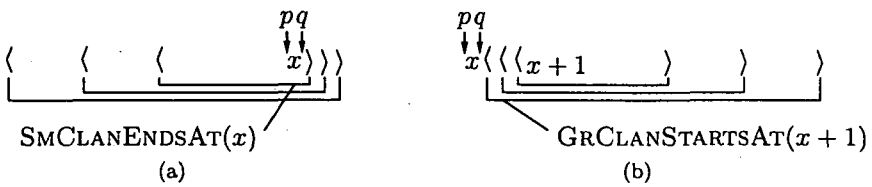


Figure 7: The computation of q in Line 14 (a) and in Line 18 (b).

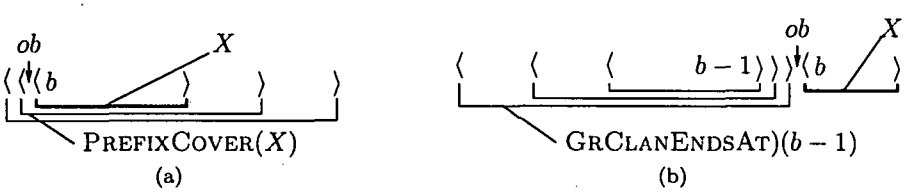


Figure 8: The computation of *ob* in Line 27 (a) and in Line 31 (b).

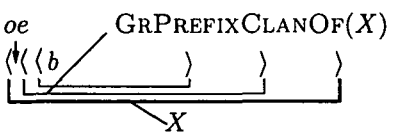


Figure 9: The computation of *oe* in Line 34.

Algorithm CORRECT-RUN$(\mathcal{A}, i, f, P, \eta)$
1 $b \leftarrow$ FIRSTOF(P)
2 **if** ISCLANSTARTSAT(b)
3 **then** $i' \leftarrow$ STARTOFOPPAR(GRCLANSTARTSAT(b))
4 **else** $i' \leftarrow$ STATE(b)
5 $e \leftarrow$ LASTOF(P)

```
6  if IsClanEndsAt(e)
7    then f' ← EndOfClPar(GrClanEndsAt(e))
8    else f' ← NextState(e)
9  if i ≠ i' or f ≠ f'
10   then return 'no'
11 for all x ∈ P
12   do p ← State(x)
13     if IsClanEndsAt(x)
14       then q ← StartOfClPar(SmClanEndsAt(x))
15       else if IsLastPosition(x)
16              then q ← f
17              else if IsClanStartsAt(x + 1)
18                     then q ← StartOfOpPar(GrClanStartsAt(x + 1))
19                     else q ← State(x + 1)
20     σ ← λ(x)
21     if not (p, σ, q) ∈ δ
22       then return 'no'
23 for all proper prime clans X ⊆ P
24   do b ← FirstOf(X)
25     e ← LastOf(X)
26     if IsPrefixOfClan(X)
27       then ob ← EndOfOpPar(PrefixCover(X))
28       else if IsFirstPosition(b)
29              then ob ← i
30              else if IsClanEndsAt(b − 1)
31                     then ob ← EndOfClPar(GrClanEndsAt(b − 1))
32                     else ob ← NextState(b − 1)
33     if IsPrefixClanIn(X)
34       then oe ← StartOfOpPar(GrPrefixClanOf(X))
35       else oe ← State(b)
36     if IsSuffixClanIn(X)
37       then cb ← EndOfClPar(GrSuffixClanOf(X))
38       else cb ← NextState(e)
39     if IsSuffixOfClan(X)
40       then ce ← StartOfClPar(SuffixCover(X))
41       else if IsLastPosition(e)
42              then ce ← f
43              else if IsClanStartsAt(e + 1)
44                     then ce ← StartOfOpPar(GrClanStartsAt(e + 1))
45                     else ce ← State(e + 1)
46     k ← IndexOfParUsedAround(X)
47     if not(ob, ⟨k, oe), (cb, ⟩k, ce) ∈ γ
48       then return 'no'
49 return 'yes'
```

The input-output specifications of the predicates and functions used in the algorithm are the following:

IsFirstPosition(x) / IsLastPosition(x)
input: a position $x \in P$;
output: 'yes' if x is the first/last position of P;
'no' otherwise.

IsClanStartsAt(x) / IsClanEndsAt(x)
input: a position $x \in P$;
output: 'yes' if there is a proper prime clan $X \subseteq P$ whose first/last position is x;
'no' otherwise.

IsPrefixOfClan(X) / IsSuffixOfClan(X)
input: a proper prime clan $X \subseteq P$;
output: 'yes' if there is a proper prime clan Y such that X is a prefix/suffix of Y;
'no' otherwise.

IsPrefixClanIn(X) / IsSuffixClanIn(X)
input: a proper prime clan $X \subseteq P$;
output: 'yes' if there is a proper prime clan Z such that Z is a prefix/suffix of X;
'no' otherwise.

State(x)
input: a position $x \in P$;
output: a state $s \in S$ in which $\mathcal{A}$ reads position x, i.e., $x \in X_s$.

NextState(x)
input: a position $x \in P$;
output: a state $t \in S$ at which $\mathcal{A}$ arrives after reading the position x, i.e., $x \in X_s$ and $(s, \lambda(x), t) \in \delta$.

FirstOf(X) / LastOf(X)
input: a proper prime clan $X \subseteq P$;
output: the first/last position of X.

StartOfOpPar(X) / EndOfOpPar(X)
input: a proper prime clan $X \subseteq P$;
output: a state $s \in S$ such that the s is the source/target of an opening parenthesizing transition $(s, \langle_j, t)$ / $(r, \langle_j, s) \in \gamma$, and this transition was used immediately before X, i.e., the designated position of X is in $Z_{\langle_j\rangle_j}$.

StartOfClPar(X) / EndOfClPar(X)
input: a proper prime clan $X \subseteq P$;
output: a state $s \in S$ such that the s is the source/target of a closing parenthesizing transition $(s, \rangle_j, t)$ / $(r, \rangle_j, s) \in \gamma$, and this transition was used immediately after X, i.e., the designated position of X is in $Z_{\langle_j\rangle_j}$.

SMCLANENDSAT(x) / GRCLANENDSAT(x)
input: a position $x \in P$;
output: the smallest/greatest proper prime clan of P that ends at position x.

GRCLANSTARTSAT(x)
input: a position $x \in P$;
output: the greatest proper prime clan of P that starts at position x.

GRPREFIXCLANOF(X) / GRSUFFIXCLANOF(X)
input: a proper prime clan $X \subseteq P$;
output: the greatest proper prime clan $Y \subseteq X$ that is a proper prefix/suffix of X.

PREFIXCOVER(X) / SUFFIXCOVER(X)
input: a proper prime clan $X \subseteq P$;
output: the smallest proper prime clan Y such that is X is a proper prefix/suffix of Y.

INDEXOFPARUSEDAROUND(X)
input: a proper prime clan X;
output: an index k such that the parentheses $(_k,)_k$ were used before and after X in the encoded run, i.e., the designated position of X is in $Z_{(_k)_k}$.

Finally, we outline the transformation of the algorithm into formula ψ_3. The following observations lead to this transformation.

1. All predicates of the algorithm can be expressed by MSO-formulas. For example, ISPREFIXOFCLAN(X) can be formulated as
$$\exists Y \big(\mathrm{PPC}(Y) \wedge \mathrm{Prefix}(X, Y) \big)$$

2. For any function $f(x_1, \ldots, x_l)$ of the algorithm and for any element c in the range of f, the fact $f(x_1, \ldots, x_l) = c$ can also be expressed by an MSO-formula. For example, for any state s in S, STARTOFOPPAR(X) $= s$ can be written as
$$\bigvee_{j \in J} \exists z \big(\mathrm{Dp}(z, X) \wedge Z_{(_j)_j}(z) \big),$$
where $J = \{ j \mid \exists t \in S, (s, (_j, t) \in \gamma \}$ is a finite set.

3. The variables whose values are not positions or sets of positions of P, all take their values from a finite set. Namely, i, f, i', f', p, q, ob, oe, cb, ce take values from S, σ from Σ, and k is an index of a parenthesis in Ω.

4. The composition of functions can be handled with the help of auxiliary variables. For example, STARTOFOPPAR(GRCLANSTARTSAT(b)) $= s$ can be expressed as
$$\exists Z \big(\textsc{GrClanStartsAt}(b) = Z \wedge \textsc{StartOfOpPar}(Z) = s \big)$$

5. Assignments like $y \leftarrow f(x_1, \ldots, x_l)$ can be treated as follows. We can consider all possible values c in the range of y in advance, and at the points of the assignments we can test whether $f(x_1, \ldots, x_l) = c$ holds. If the range of y is P or the power-set $\mathcal{P}(P)$, i.e., y is a 'standard' first or second order variables, then existential quantification can be used. On the other hand, if y is not 'standard', then it has a finite range by point 3. Hence we can use disjunction over this finite range. For example, we can start the formula realizing Lines 12–22 as

$$\bigvee_{p \in S} \bigvee_{q \in S} \bigvee_{\sigma \in \Sigma} \cdots$$

6. The control flow of the algorithm is easily expressible in the logic framework. For the sequential executions conjunctions, for "for all" loops universal quantifications, and for the conditional statements implications and negations can be used.

References

[1] I. Dolinka. A note on identities of two-dimensional languages. *Disc. Appl. Math.*, 146(2005), 43–50.

[2] A. Ehrenfeucht and G. Rozenberg. Theory of 2-structures, Part 1: clans, basic subclasses, and morphisms. Part 2: representation through labeled tree families. *Theoret. Comput. Sci.*, 70(1990), 277–303, 305–342.

[3] A. Ehrenfeucht and G. Rozenberg. Angular 2-structures. *Theoret. Comput. Sci.*, 92(1992), 227–248.

[4] A. Ehrenfeucht and G. Rozenberg. T-structures, T-functions and texts. *Theoret. Comput. Sci.*, 116(1993), 227–290.

[5] Z. Ésik. Free algebras for generalized automata and language theory. *RIMS Kokyuroku 1166*, Kyoto University, Kyoto, 2000, 52–58.

[6] Z. Ésik and Z. L. Németh. Higher dimensional automata, *J. of Autom. Lang. Comb.*, 9(2004), 3–29.

[7] Z. Ésik and Z. L. Németh. Algebraic and graph-theoretic properties of infinite n-poests. *Theoret. Informatics Appl.*, 39(2005), 305–322.

[8] D. Giammarresi and A. Restivo. Two-dimensional languages. In: *Handbook of Formal Languages*, Vol. 3, Springer-Verlag, Berlin, 1997, 215–267.

[9] J. Grabowski. On partial languages. *Fund. Inform.*, 4(1981), 427–498.

[10] K. Hashiguchi, S. Ichihara and S. Jimbo. Formal languages over free binoids. *J. Autom. Lang. Comb.*, 5(2000), 219–234.

[11] K. Hashiguchi, Y. Sakakibara and S. Jimbo. Equivalence of regular binoid expressions and regular expressions denoting binoid languages over free binoids. *Theoret. Comput. Sci.*, 312(2004), 251–266.

[12] H. J. Hoogeboom and P. ten Pas. Text languages in an algebraic framework. *Fund. Inform.*, 25(1996), 353–380.

[13] H. J. Hoogeboom and P. ten Pas. Monadic second-order definable text languages. *Theory Comput. Syst.*, 30(1997), 335–354.

[14] D. Kuske. Towards a language theory for infinite N-free pomsets. *Theoret. Comput. Sci.*, 299(2003), 347–386.

[15] K. Lodaya and P. Weil. Kleene iteration for parallelism. In: proc. *FST & TCS'98*, LNCS 1530, Springer-Verlag, 1998, 355–366.

[16] K. Lodaya and P. Weil. Series-parallel languages and the bounded-width property. *Theoret. Comput. Sci.*, 237(2000), 347–380.

[17] K. Lodaya and P. Weil. Rationality in algebras with series operation. *Inform. and Comput.*, 171(2001), 269–293.

[18] Z. L. Németh. A Hierachy Theorem for Regular Languages over Free Bisemigroups. *Acta Cybern.*, 16(2004), 567–577.

[19] Z. L. Németh. Automata on Infinite Biposets. In: *Automata and Formal Languages, 11th Int. Conf., AFL 2005, Dobogókő, Hungary, May 17–20, Proceedings*, Inst. of Informatics, Univ. of Szeged, Szeged, 2005, 213–226.

[20] D. Perrin and J.-E. Pin. *Infinite Words.* Pure and Applied Mathematics, Vol. 141, Elsevier, 2004.

[21] H. Straubing. *Automata, Formal Logic and Circuit Complexity.* Birkhuser, Boston, 1994.

[22] J. Valdes, R. E. Tarjan and E. L. Lawler. The recognition of series-parallel digraphs. *SIAM J. Comput.*, 11(1982), 298–313.

[23] P. Weil. Algebraic recognizability of languages. In: proc. *MFCS 2004*, LNCS 3153, Springer-Verlag, 2004, 149–175.

[24] Th. Wilke. An algebraic theory for regular languages of finite and infinite words. *Internat. J. Algebra Comput.*, 3(1993), 447–489.

Acta Cybernetica 17 (2006) 799–810.

Rotational tree structures on binary trees and triangulations*

Jean Marcel Pallo†

Abstract

A rotation in a binary tree is a simple and local restructuring technique commonly used in computer science. We propose in this paper three restrictions on the general rotation operation. We study the case when only leftmost rotations are permitted, which corresponds to a natural flipping on polygon triangulations. The resulting combinatorial structure is a tree structure with the root as the greatest element. We exhibit an efficient algorithm for computing the join of two trees and the minimum number of leftmost rotations necessary to transform one tree into the other.

Keywords: Binary trees; Rotation; Distance; Lattice; Algorithms

1 Introduction

Rotation is one of the most common operations for restructuring binary trees. It has the advantage of altering the depths of some of the nodes in the tree while preserving the symmetric order of all the nodes. Thus rotation is commonly used in a variety of algorithms for maintaining binary search trees with a good amortized behavior [10, 24, 28].

The combinatorial properties of binary trees under the rotation operation have been studied for thirty years [27]. In [17] we have shown that a directed version of the rotation graph of binary trees with n nodes is a lattice, known as the nth Tamari lattice. This corresponds to the case when only left rotations are permitted in the binary tree transformation. Over the last ten years, Tamari lattices have often been used as examples to illustrate algebraic theories [1, 3, 16, 25].

Initially, Tamari lattices were orderings of parenthesizations of words. But nowadays they can be described in other ways via the well-known bijections between families of Catalan combinatorial objects. A system that is isomorphic to Tamari lattices is that of triangulations of a convex polygon related by the diagonal flip operation. This is the transformation that converts one triangulation into another

*The results reported in this paper were presented at the 11*th* International Conference AFL 2005 (Automata and Formal Languages) held at Dobogókő, Hungary, May 17–20.

†LE2I, UMR 5158, Université de Bourgogne, BP 47870, F21078 DIJON-Cedex, France, E-mail: pallo@u-bourgogne.fr

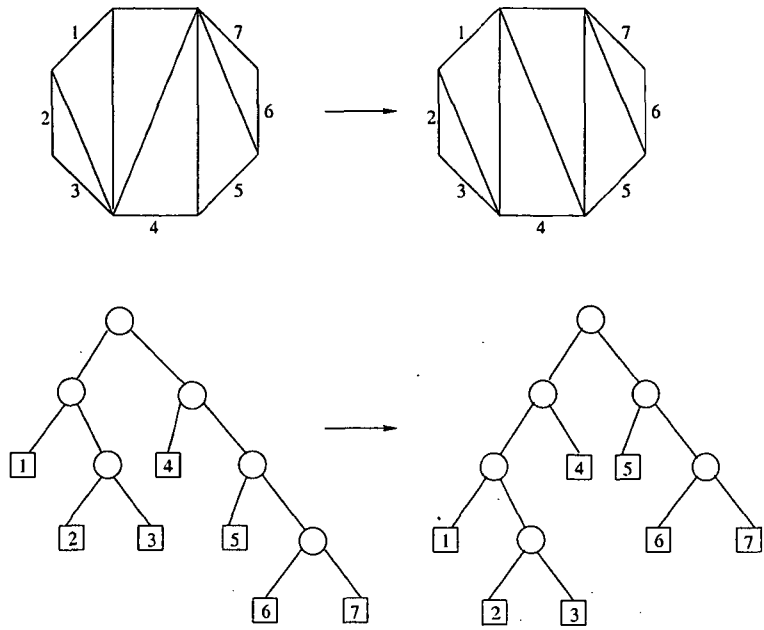


Figure 1: A triangulation diagonal flip and its corresponding binary tree rotation

by removing a diagonal in the triangulation and adding the diagonal that subdivides the resulting quadrilateral on the opposite way [8, 9, 26] (see Fig. 1).

In 1982, Culik and Wood defined the rotation distance between two binary trees with the same number of leaves as the minimum number of rotations necessary to transform one tree into the other [4]. Using the classical bijection between binary trees with n internal nodes and triangulations of $(n+2)$- gons, the previous distance is equivalent to the minimum number of diagonal-flip transformations needed to convert one triangulation of a polygon into another. There remains today an open problem whether the rotation distance can be computed in polynomial time.

Therefore it seems natural to consider special instances of rotation transformations in order to obtain simpler operations [12, 24]. In [2] the rotation operation is limited to the case where the leftmost subtree is constrained to be a leaf. In [5, 6, 11, 22] the authors only allow rotations at nodes along the right arm of a tree.

The current paper belongs to this appoach. We consider the problem by limiting the general rotation operation to the restricted version where only leftmost rotations on trees are allowed. We obtain a tree structure which is a join-semilattice with the root as the greatest element. An efficient algorithm computes the corresponding restricted rotation distance. This algorithm is constructive: it builds a sequence of leftmost rotations transforming one tree into the other.

Clearly, the restricted rotation distance defined above is bounded below by the usual rotation distance for which no efficient algorithm is known to compute it

exactly. However, this restricted rotation distance is a weak approximation of the usual rotation distance. A better approximation can be found in [21, 23]. This new metric can be considered as a way of measuring the difference in shape between two binary trees.

2 Definitions and terminology

Let us denote by ○ (respectively □) internal nodes (respectively leaves) of a binary tree. Let T_L (respectively T_R) denote the left (respectively right) subtree of a binary tree T (the order is significant). Thus we can write $T = \bigcirc T_L T_R$ in Polish notation, i.e. by traversing T in preorder (visit the root and then the left and right subtrees recursively). The weight $|T|$ of a binary tree T is the number of leaves of T. Let B_n denote the set of binary trees with n internal nodes (and thus with $n+1$ leaves). The leaves of $T \in B_n$ are numbered from 1 to $n+1$ by a preorder traversal of T (i.e. from left to right). The left (respectively right) arm of $T \in B_n$ is the path from the root of T to its first (respectively $(n+1)$th) leaf. The mirror image $\widetilde{T}$ of T is recursively defined by $\widetilde{T} = \bigcirc \widetilde{T_R}\widetilde{T_L}$ and $\widetilde{\square} = \square$. Let us define $\mathbf{0}_n = (\bigcirc\square)^n\square$ (respectively $\mathbf{1}_n = \bigcirc^n \square^{n+1}$) the tree of B_n where every internal node has a leaf as a left (respectively right) child.

In this paper we use the representation of binary trees via weight sequences introduced in [17]. This coding is defined as follows. Given $T \in B_n$, the weight sequence of T is the integer sequence $w_T = (w_T(1), \ldots, w_T(n))$ where $w_T(i)$ is the weight of the largest subtree of T whose last leaf is the ith leaf (see Fig. 2). The usual left-rotation $\rightharpoonup$ on B_n is defined as follows. A tree $T \in B_n$ being given, it associates a tree T' obtained by replacing some subtree $\bigcirc T_1 \bigcirc T_2 T_3$ of T by the

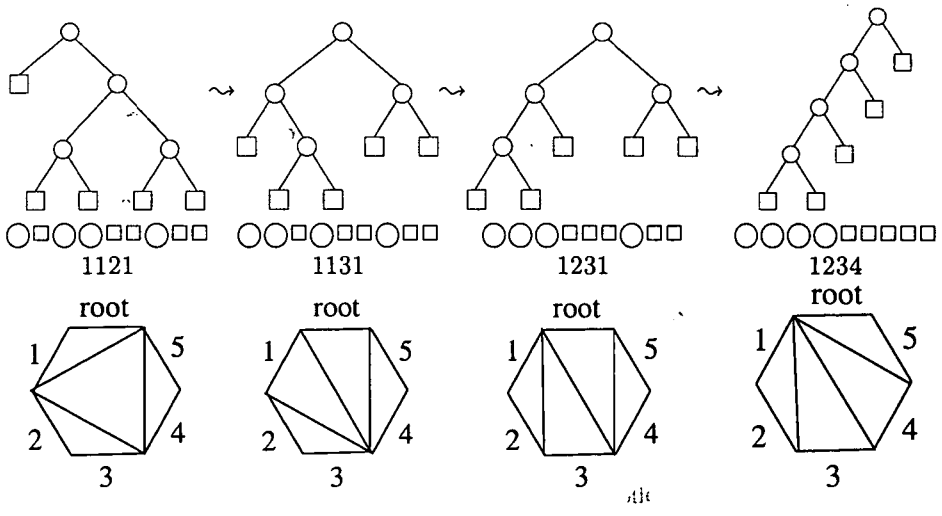


Figure 2: Three lefmost left-rotations in B_4 and the corresponding flips in T_6

subtree $\bigcirc\bigcirc T_1T_2T_3$. Let $\overset{-1}{\rightarrow}$ denote the right-rotation and let $\overset{*}{\rightarrow}$ denote the reflexive transitive closure of $\rightarrow$. The usual rotation distance between T and $T' \in B_n$ is the fewest number of left- and right-rotations required to convert T into T'. We have proved in [17] the following characterization: given $T, T' \in B_n$, we have $T \overset{*}{\rightarrow} T'$ iff for all $i \in [1,n]$: $w_T(i) \leq w_{T'}(i)$.

Let us consider $(n+2)$-gons, i.e. convex polygons with $n+2$ sides and with a distinguished side as the top. We label the other sides from 1 to $n+1$ counterclockwise. Any triangulation of the $(n+2)$-gon has n triangles and $n-1$ non-crossing diagonals. Let T_{n+2} denote the set of triangulations of the $(n+2)$-gon. There is an explicit bijection τ between B_n and T_{n+2} [23,26]. The top of the $(n+2)$-gon $\tau(T)$ corresponds to the root of the tree T. The ith side of $\tau(T)$ corresponds to the ith leaf of T. Diagonals corresponds to internal nodes recursively as follows. If j is the last leaf of the left subtree T_L of T, then T_L corresponds to the $(j+1)$-gon having edge set $\{1,\ldots,j\}$ and the right subtree T_R corresponds to the $(n-j+2)$-gon having edge set $\{j+1,\ldots,n+1\}$ (see Fig. 1 and 2).

Given some $T \in B_n$ with $T \neq \mathbf{1}_n$, according to the Polish notation of T, consider the leftmost $\square$ followed by a $\bigcirc$ which respectively are the last leaf of a subtree T_1 and the root of a subtree $\bigcirc T_2T_3$. Thus the root of $\bigcirc T_1 \bigcirc T_2T_3$ is located on the left arm of T. Then define as the leftmost left-rotation on B_n the transformation $T \rightsquigarrow T'$ which consists in converting the leftmost subtree $\bigcirc T_1 \bigcirc T_2T_3$ of T into $\bigcirc\bigcirc T_1T_2T_3$ (see Fig. 2). Given $T \in B_n$, the leftmost left-rotation transformation is uniquely defined.

Let us describe the transformation on $\tau(T) \in T_{n+2}$ which corresponds to the leftmost left-rotation on $T \in B_n$ via the classical bijection τ between B_n and T_{n+2}. This transformation is the unique operation on $\tau(T)$ which consists in removing some diagonal and adding a new diagonal an end of which coincides with the vertex located between the root side and the side labelled 1 (see Fig. 2 and 3). This alternative formulation may seem more natural and intuitive. But the weight sequences of binary trees are more appropriate for calculations.

Let $\overset{*}{\rightsquigarrow}$ denote the reflexive transitive closure of $\rightsquigarrow$. The leftmost rotation graph LG_n is the directed graph which has a node for each tree of B_n. Two nodes are adjacent when their corresponding trees differ by a single leftmost left-rotation. Since the leftmost left-rotation operation on T is uniquely defined, LG_n enjoys a tree structure. The leftmost rotation distance $d(T,T')$ between T and $T' \in B_n$ is the length of the unique path between T and T' in the directed graph LG_n. LG_n is a subgraph of the graph G_n according to the usual rotation. Algebraic properties of G_n can be found in [1, 3, 16, 20, 25, 26].

3 Tree structure B_n

Given $T \in B_n$ with $T \neq \mathbf{1}_n$ and w_T, we obtain the weight sequence of the unique T' such that $T \rightsquigarrow T'$ in the following way. Let $i \geq 2$ be the smallest integer such that $w_T(i) = 1$. Let $j = max\{m \in [i,n] | i = m - w_T(m) + 1\}$, i.e. the greatest integer m such that the largest subtree with last leaf m has i as the first leaf. Then

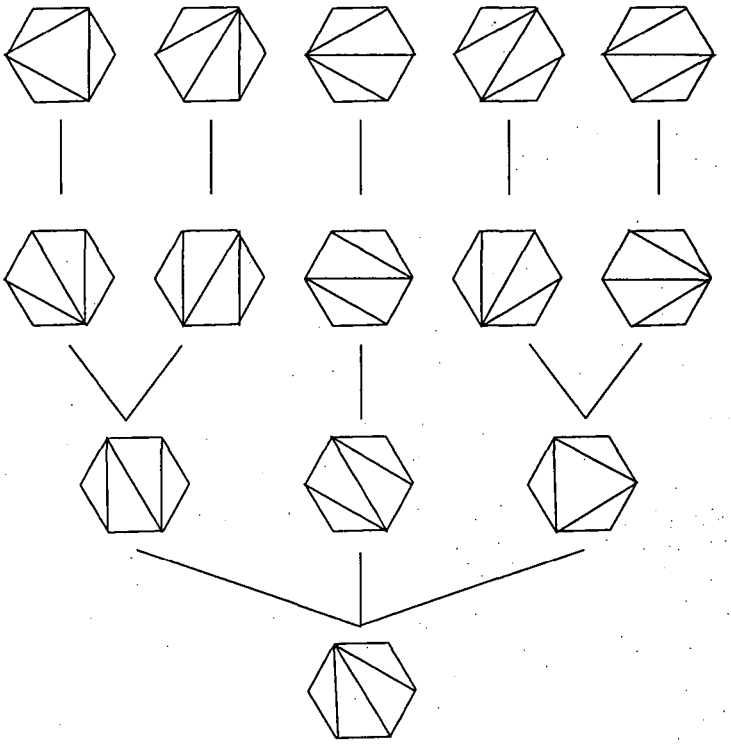


Figure 3: The flipping tree structure T_6

$w_{T'} = w_T$ except for the integer j: $w_{T'}(j) = j$. It is worth noting that this integer j cannot be modified further since we have $1 \leq w_T(k) \leq k$ for all $T \in B_n$ and $k \in [1, n]$.

The poset $(B_n, \overset{*}{\rightsquigarrow})$ enjoys some properties which can be easily obtained. $(B_n, \overset{*}{\rightsquigarrow})$ is a poset with greatest element $\mathbf{1}_n$ for which $w_{\mathbf{1}_n} = (1, 2, 3, \ldots, n)$. This poset has a tree structure (with the greatest element $\mathbf{1}_n$ as root) and thus is a join-semilattice (see Fig. 4 and 5). The poset $(B_n, \overset{*}{\rightsquigarrow})$ is graded, i.e. there exists an integer-valued function r defined on B_n by $r(T) = card\{i \in [1, n] | w_T(i) = i\}$ such that $T \overset{*}{\rightsquigarrow} T'$ and $r(T') = 1 + r(T)$ iff $T \rightsquigarrow T'$. $r(T)$ is equal to the number of internal nodes that are on the left arm of T. We have $r(\mathbf{1}_n) = n - 1$.

Let us remark that B_n is isomorphic to two subtrees of B_{n+1}. One is obtained by sustituting $\bigcirc\square\square$ for the last leaf $\square$ in all the B_n trees. If $(w_1, \ldots, w_n) \in B_n$, then $(w_1, \ldots, w_n, 1)$ is the weight sequence of a tree in the corresponding subtree of B_{n+1}. The other is obtained by sustituting $\bigcirc\square\square$ for the one before last leaf $\square$ in all the B_n trees. If $(w_1, \ldots, w_n) \in B_n$, then $(w_1, \ldots, w_{n-1}, 1, 1 + w_n)$ is the weight sequence of a tree in the corresponding subtree of B_{n+1}. For example in Fig. 5, the left and right subtrees of B_5 are isomorphic to B_4 (Fig. 4).

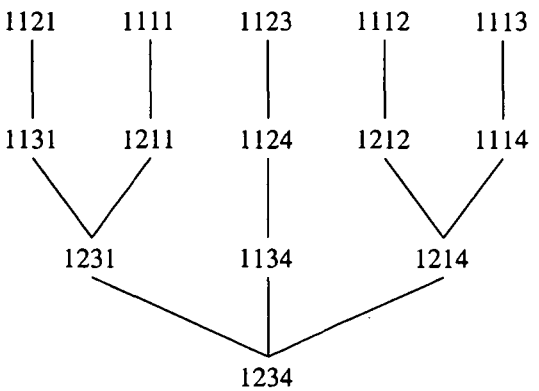


Figure 4: The leftmost tree structure B_4

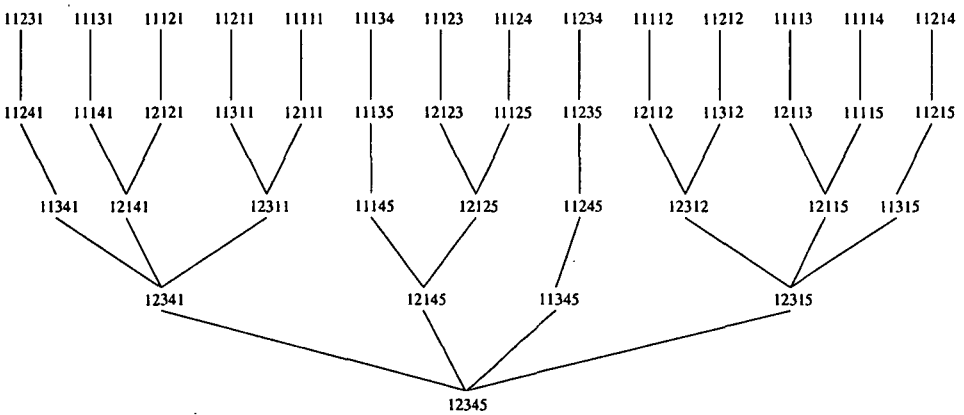


Figure 5: The leftmost tree structure B_5

The leftmost rotation distance between T and T' can be computed by the formula $d(T, T') = 2r(T \vee T') - r(T) - r(T')$. Thus we are led to compute the join $T \vee T'$ of any couple of trees T and T'.

4 Computing joins and leftmost rotation distance

We already have observed that in applying the leftmost rotation $T \rightsquigarrow T'$ the unique integer which has been transformed reaches its maximal possible value and thus cannot increase. Now, for every $T \in B_n$, compute from w_T an ordered array a_T which keeps track of the sequence of all the integer transformations for designing the unique path between T and 1_n.

Algorithm (Computation of a_T from w_T)
Given $T \in B_n$ and its weight sequence w_T
$k := 1$
for $i := 1$ **to** n **do**
  if $w_T(i) = 1$ **then**
    for $j := n$ **downto** i **do**
      if $i = j - w_T(j) + 1$ **then** $a_T(k) := j; k := k + 1$ **endif**
    enddo
  endif
enddo

This algorithm requires $O(n^2)$ time in the worst case and $O(n)$ space.

The join $T \vee T'$ of T and T' is located at the intersection of the two paths connecting T and T' to $\mathbf{1}_n$. Thus we compute $w_{T \vee T'}$ in the following way.

Let us consider the greatest suffix which is common to a_T and $a_{T'}$ (if it exists). The corresponding prefixes of a_T and $a_{T'}$ contain the same integers i (possibly in different order) for which $w_{T \vee T'}(i) = i$. The remaining integers j verify $w_{T \vee T'}(j) = w_T(j) = w_{T'}(j)$. Therefore it is easy to compute $w_{T \vee T'}$, and then $r(T)$, $r(T')$, $d(T, T') = 2r(T \vee T') - r(T) - r(T')$ using the rank function $r(T) = card\{i \in [1, n] | w_T(i) = i\}$. See some examples in Table 1 where suffixes are shown in bold type.

Table 1:

w_T	$w_{T'}$	a_T	$a_{T'}$	$w_{T \vee T'}$	$d(T, T')$
11112	11315	12354	53124	12315	4
11234	11345	15432	54312	11345	3
11214	11215	15324	51324	11215	1
11111	12345	12345	54321	12345	4
11111	11114	12345	15234	12345	8
11315	12112	53124	21354	12315	3
11212	11114	13254	15234	12315	6

a_T (respectively $a_{T'}$) allows to build the unique path between T and $T \vee T'$ (respectively T' and $T \vee T'$). Thus we obtain the unique path $(T, T \vee T', T')$ between T and T'.

5 Mirror leftmost rotation distance

Let us define the mirror leftmost rotation $\hookrightarrow$ on B_n by $T \hookrightarrow T'$ iff $\widetilde{T'} \rightsquigarrow \widetilde{T}$. Then $(B_n, \overset{*}{\hookrightarrow})$ is a poset with least element $\mathbf{0}_n$ for which $w_{\mathbf{0}_n} = (1, 1, 1, \ldots, 1)$. This poset has a tree structure (with the least element $\mathbf{0}_n$ as root) and thus is a meet-

semilattice. This poset $(B_n, \overset{*}{\hookrightarrow})$ is ranked by the rank function $\rho(T) = n - k_T + 1$ where k_T is the number of internal nodes on the right arm of $T \in B_n$. We have $\rho(\mathbf{0}_n) = 1$ The following algorithm computes $\rho(T)$ using the weight sequence of T:

Rank algorithm (Computation of $\rho(T)$ from w_T)
Given $T \in B_n$ and its weight sequence w_T;
$k_T := 1$; $i := n$;
while $i > 1$ **do**
  if $w_T(i) = 1$ **then** $k_T := k_T + 1$;$i := i - 1$
  else $i := i - w_T(i) + 1$ **endif**
enddo
$\rho(T) = n - k_T + 1$

See $(B_4, \overset{*}{\hookrightarrow})$ in Fig. 6. Observe that $\hookrightarrow$ is a particular case of the right-arm rotation transformation defined in [22]. As illustration, compare for example Fig. 3 of [22, p. 176] and Fig. 6 of this paper. The edge which links 1112 and 1212 in Fig. 3 of [22] has disappeared in Fig. 6. The graph drawn in Fig. 3 of [22] does not enjoy the tree structure property.

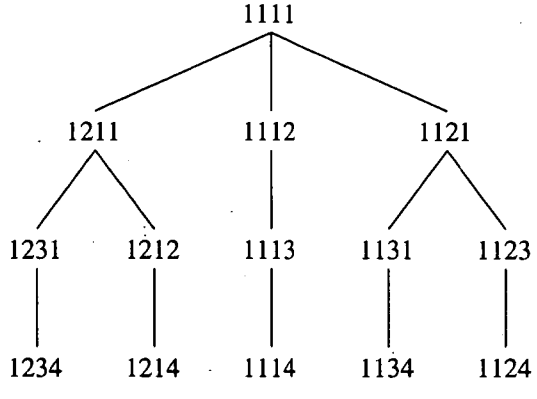


Figure 6: The mirror image of B_4

Let us define the mirror leftmost rotation distance $\widetilde{d}(T, T')$ between T and $T' \in B_n$ as the length of the unique path between T and T' in the graph of $(B_n, \overset{*}{\hookrightarrow})$. Therefore we have: $\widetilde{d}(T, T') = d(\widetilde{T}, \widetilde{T'})$.

Since $w_{\widetilde{T}}$ can be easily computed recursively from w_T, the mirror leftmost rotation distance $\widetilde{d}(T, T') = d(\widetilde{T}, \widetilde{T'})$ is computed using Section 4. Then $\delta(T, T') = min(d(T, T'), d(\widetilde{T}, \widetilde{T'}))$ is bounded below by the usual rotation distance for which no polynomial time algorithm is known to compute it exactly today. See some examples in B_8 (Table 2).

Table 2:

w_T	$w_{T'}$	$d(T,T')$	$w_{\widetilde{T}}$	$w_{\widetilde{T'}}$	$d(\widetilde{T},\widetilde{T'})$	$\delta(T,T')$
11121511	12123611	8	12312148	12311141	9	8
11121518	11234112	11	11212147	11341118	11	11
11312312	11214111	13	11311612	12341218	6	6
11115123	12311312	11	11141234	11312611	11	11
11111123	11231237	14	11145678	11114118	8	8
11235112	12115111	11	11341114	12341231	9	9
11211612	11111312	5	11312315	11312678	9	5
11311245	11341678	6	11113612	12112678	9	6

6 Open problems

We propose below two other new definitions of restricted rotations which lead to computing open problems.

First we can restrict the general definition of the rotation transformation by choosing $\bigcirc T_1 \bigcirc T_2 T_3$ as the rightmost subtree in the Polish notation of T. More precisely, let us consider in the Polish notation of T the rightmost pattern $\square\bigcirc$ made up of a $\square$ followed by a $\bigcirc$. This $\bigcirc$ is the root of a subtree denoted by $\bigcirc T_2 T_3$, and thus T_3 is always equal to a leaf $\square$. Let us denote by T_1 the largest subtree of T whose last leaf is the leaf $\square$ involved in the previous pattern $\square\bigcirc$. The uniquely defined rotation which transforms $\bigcirc T_1 \bigcirc T_2 \square$ of T into $\bigcirc \bigcirc T_1 T_2 \square$ is called rightmost left-rotation on the tree T. B_n endowed with this transformation has a tree structure (with the root as the greatest element $\mathbf{1}_n$) and thus is a join-semilattice (see Fig. 7). Despite this tree structure, the direct computation of the joins of two trees seems to be more arduous. The definition of an efficient algorithm for computing the corresponding rightmost rotation distance d' seems difficult, too. However, we can easily exhibit the unique paths connecting T and T' with $\mathbf{1}_n$. The weight sequence of the unique tree $succ(T)$ obtained from T by a rightmost rotation is such that $w_{succ(T)} = w_T$ except for the integer $i = max\{k \in [j,n] | w_T(k) = k - j + 1\}$ where $j = max\{l \in [1,n] | w_T(l) = 1\}$. For this integer i, we have $w_{succ(T)}(i) = w_T(i) + w_T(i - w_T(i))$. The join $T \vee T'$ of T and T' is located at the intersection of the two paths $(T, \mathbf{1}_n)$ and $(T', \mathbf{1}_n)$. Unfortunately, this rough construction requires $O(n^3)$ time and $O(n^2)$ space.

It is worth noting that leftmost d and rightmost d' rotation distances cannot be compared. For example: $d(1112,1114) = 3 < d'(1112,1114) = 4$ and $d'(1113,1121) = 2 < d(1113,1121) = 6$ (see Fig. 4 and 7).

Second, we have limited in [2] the rotation operation to the case where the leftmost subtree T_1 of the subtree $\bigcirc T_1 \bigcirc T_2 T_3$ is always constrained to be a leaf $\square$. This transformation $\bigcirc\square\bigcirc T_2 T_3 \xrightarrow{L} \bigcirc\bigcirc\square T_2 T_3$ induces a graded lower semimodular meet-semilattice structure on B_n. We can define a new restricted rotation by compelling, this time, the central subtree T_2 of the subtree $\bigcirc T_1 \bigcirc T_2 T_3$ to be always

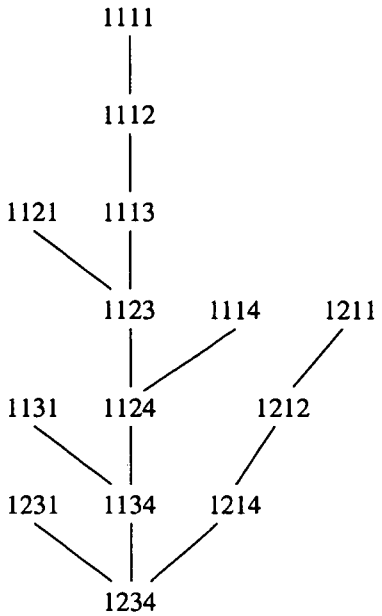


Figure 7: The rightmost tree structure B_4

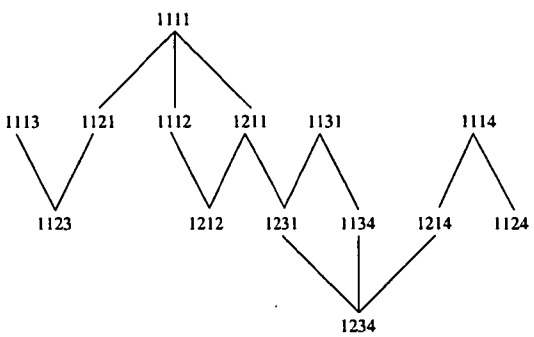


Figure 8: The central poset B_4

equal to a leaf $\square$. This transformation $\bigcirc T_1 \bigcirc \square T_3 \xrightarrow{C} \bigcirc\bigcirc T_1 \square T_3$ induces a graded poset structure on B_n, but does not have as good algebraic properties as before. However, this "central" rotation operation $\xrightarrow{C}$ has a nice characterization: $T \xrightarrow{C} T'$ iff $w_T = w_{T'}$ except for an integer i such that $w_T(i) = 1 < w_{T'}(i)$ (see Fig. 8). The rank of $T \in B_n$ is easily computed by $r(T) = n + 1 - card\{i \in [1,n] | w_T(i) = 1\}$. Here too, it seems difficult to exhibit an efficient algorithm for computing the corresponding central rotation distance.

Acknowledgement

I would like to thank the anonymous referees for their constructive remarks and recommendations which have greatly helped to improve this paper.

References

[1] M.K. Bennet, G. Birkhoff, Two families of Newman lattices, Alg. Universalis **32**(1994), 115-144.

[2] A. Bonnin, J. Pallo, A shortest path metric on unlabeled binary trees, Pattern Recognition Lett. **13**(1992), 411-415.

[3] C. Chameni-Nembua, B. Monjardet, Les treillis pseudocomplments finis, European J. Combin. **13**(1992), 89-107.

[4] K. Cukik, D. Wood, A note on some tree similarity measures, Inform. Process. Lett. **15**(1982), 39-42.

[5] S. Cleary, Restricted rotation distance between binary trees, Inform. Process. Lett. **84**(2002), 333-338.

[6] S. Cleary, J. Taback, Bounding restricted rotation distance, Inform. Process. Lett. **88**(2003), 251-256.

[7] R.D. Dutton, R.O. Rogers, Properties of the rotation graph of binary trees, Congr. Numer. **109**(1995), 51-63.

[8] S. Hanke, T. Ottmann, S. Schuierer, The edge-flipping distance of triangulations, J. Universal Comput. Sci. **2**(1996), 570-579.

[9] F. Hurtado, M. Noy, J. Urrutia, Flipping edges in triangulations, Discrete Comput. Geom. **22**(1999), 333-346.

[10] K.S. Larsen, E. Soisalon-Soininen, P. Widmayer, Relaxed balance using standard rotations, Algorithmica **31**(2001), 501-512.

[11] J.M. Lucas, A direct algorithm for restricted rotation distance, Inform. Process. Lett. **90**(2004), 129-134.

[12] J.M. Lucas, Localized rotation distance in binary trees, Congr. Numer. **169**(2004), 161-178.

[13] J.M. Lucas, Untangling binary trees via rotations, Comput. J. **47**(2004), 259-269.

[14] F. Luccio, L. Pagli, On the upper bound of the rotation distance of binary trees, Inform. Process. Lett. **31**(1989), 57-60.

[15] E. Mäkinen, On the rotation distance of binary trees, Inform. Process. Lett. **26**(1987/88), 271-272.

[16] G. Markowski, Primes, irreducibles and extremal lattices, Order **9**(1992), 265-290.

[17] J.M. Pallo, Enumerating, ranking and unranking binary trees, Comput. J. **29**(1986), 171-175.

[18] J.M. Pallo, On the rotation distance in the lattice of binary trees, Inform. Process. Lett. **25**(1987), 369-373.

[19] J.M. Pallo, Some properties of the rotation lattice of binary trees, Comput. J. **31**(1988), 564-565.

[20] J.M. Pallo, An algorithm to compute the Mbius function of the rotation lattice of binary trees, RAIRO Theoret. Inform. Appl. **27**(1993), 341-348.

[21] J.M. Pallo, An efficient upper bound of the rotation distance of binary trees, Inform. Process. Lett. **73**(2000), 87-92.

[22] J.M. Pallo, Right-arm rotation distance between binary trees, Inform. Process. Lett. **87**(2003), 173-177.

[23] R.O. Rogers, On finding shortest paths in the rotation graph of binary trees, Congr. Numer. **137**(1999), 75-95.

[24] A.A. Ruiz, F. Luccio, A.M. Enriquez, L. Pagli, k-restricted rotation with an application to search tree rebalancing, $9th$ WADS, Lecture Notes in Computer Science, Springer, vol. **3608**(2005), 2-13.

[25] B.E. Sagan, A generalization of Rota's NBC theorem, Adv. Math. **111**(1995), 195-207.

[26] D.D. Sleator, R.E. Tarjan, W. Thurston, Rotation distance, triangulations and hyperbolic geometry, J. Amer. Math. Soc. **1**(1988), 647-681.

[27] D. Tamari, Monodes prordonns et chanes de Malcev, Bull. Soc. Math. France **82**(1954), 53-96.

[28] R. Wilber, Lower bounds for accessing binary search trees with rotations, SIAM J. Comput. **18**(1989), 56-67.

Acta Cybernetica **17** (2006) 811–823.

Regular tree languages and quasi orders

Tatjana Petković*

Abstract

Regular languages were characterized as sets closed with respect to monotone well-quasi orders. A similar result is proved here for tree languages. Moreover, families of quasi orders that correspond to positive varieties of tree languages and varieties of finite ordered algebras are characterized.

1 Introduction

Regular languages are characterized by the well-known Myhill–Nerode theorem as those that can be saturated by a congruence, or a right congruence, of finite index defined on the free semigroup over the same alphabet over which the language is defined. A generalization of this result, proved by Ehrenfeucht, Haussler and Rozenberg in [3], characterizes regular languages as closed sets with respect to monotone well-quasi orders. A result analogous to Myhill-Nerode's theorem exists for tree languages, whereas we are going to prove here a characterization of regular tree languages similar to the generalized Myhill–Nerode's theorem from [3].

On the other hand, variety theory establishes correspondences between families of languages, algebras, semigroups and relations. The elementary result of this type is Eilenberg's Variety theorem [4] which was motivated by characterizations of several families of string languages by syntactic monoids or semigroups (see [4, 10]), such as Schützenberger's theorem [12] connecting star-free languages and aperiodic monoids. Eilenberg's theorem has been extended in various directions. We are going to mention here only those that are of the greatest interest for this work. Thérien [16] extended the Eilenberg's correspondence to varieties of congruences on free monoids. Concerning trees and algebras, similar correspondences were established by Steinby [13, 14, 15], Almeida [1], Ésik [5], Ésik and Weil [6]. On the other hand, a correspondence between positive varieties of string languages and varieties of ordered semigroups was established by Pin in [11], and similar results were proved for trees by Ésik [5], and Petković and Salehi in [9]. Motivated by this, and a characterization of regular tree languages established in the first part of the paper, we involve in the correspondence suitable families of quasi orders on term algebras.

*Nokia, Joensuunkatu 7, 24100 Salo, Finland, E-mail: tatjana.petkovic@nokia.com

The paper consists of three parts. In Section 2 concepts are introduced and preliminary results given. In Section 3 regular tree languages are characterized by well-quasi orders. In Section 4 varieties of quasi orders are defined and a correspondence between positive varieties of tree languages, varieties of ordered algebras and varieties of quasi orders is established.

2 Preliminaries

A finite set of function symbols is called a *ranked alphabet*. The ranked alphabet Σ will be fixed throughout the paper, and the set of m-ary function symbols from Σ is denoted by Σ_m ($m \geq 0$). A *Σ-algebra* is a structure $\mathcal{A} = (A, \Sigma)$ where A is a set and operations of Σ are interpreted in A, i.e., any $c \in \Sigma_0$ is interpreted by an element $c^{\mathcal{A}} \in A$ and any $f \in \Sigma_m$ ($m > 0$) is interpreted by an m-ary function $f^{\mathcal{A}} : A^m \to A$. Congruences, morphisms, subalgebras, direct products, etc., are defined as usual for algebras (see e.g. [2, 15]).

For a ranked alphabet Σ and a *leaf alphabet* X, the set of *ΣX-trees* $\mathrm{T}_\Sigma(X)$ is the smallest set satisfying

(1) $\Sigma_0 \cup X \subseteq \mathrm{T}_\Sigma(X)$, and

(2) $f(t_1, \ldots, t_m) \in \mathrm{T}_\Sigma(X)$ for all $m > 0$, $f \in \Sigma_m$, $t_1, \ldots, t_m \in \mathrm{T}_\Sigma(X)$.

The *ΣX-term algebra* $\mathcal{T}_\Sigma(X) = (\mathrm{T}_\Sigma(X), \Sigma)$ is determined by

(1) $c^{\mathcal{T}_\Sigma(X)} = c$ for $c \in \Sigma_0$,

(2) $f^{\mathcal{T}_\Sigma(X)}(t_1, \ldots, t_m) = f(t_1, \ldots, t_m)$ for all $m > 0$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in \mathrm{T}_\Sigma(X)$.

A *ΣX-tree language* is any subset of the ΣX-term algebra. An algebra $\mathcal{A} = (A, \Sigma)$ *recognizes* a tree language $T \subseteq \mathrm{T}_\Sigma(X)$ if there is a morphism $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ and a subset $F \subseteq A$ such that $T = F\phi^{-1}$. In the case a tree language can be recognized by a finite algebra, it is *regular* or *recognizable*. It is known that a tree language is regular if and only if it is saturated by a congruence of finite index.

Let ξ be a symbol which does not appear in any other alphabet considered here. The set of *ΣX-contexts*, denoted by $\mathrm{C}_\Sigma(X)$, consists of the $\Sigma(X \cup \{\xi\})$-trees in which ξ appears exactly once. For $P, Q \in \mathrm{C}_\Sigma(X)$ and $t \in \mathrm{T}_\Sigma(X)$ the context PQ, the composition of P and Q, is obtained by replacing the special leaf ξ in P with Q, and the term $P(t)$ results from P by replacing ξ with t. Note that $\mathrm{C}_\Sigma(X)$ is a monoid with the composition operation and that $(PQ)(t) = P(Q(t))$ holds for all $P, Q \in \mathrm{C}_\Sigma(X)$, $t \in \mathrm{T}_\Sigma(X)$.

For an algebra $\mathcal{A} = (A, \Sigma)$, an m-ary function symbol $f \in \Sigma_m$ ($m > 0$) and elements $a_1, \ldots, a_m \in A$, the term $f^{\mathcal{A}}(a_1, \ldots, \xi, \ldots, a_m)$ where the new symbol ξ sits in the i-th position, for some $i \leq m$, determines a unary function $A \to A$ defined by $a \mapsto f^{\mathcal{A}}(a_1, \ldots, a, \ldots, a_m)$ which is an *elementary translation* of $\mathcal{A}$. The set of translations of $\mathcal{A}$, denoted by $\mathrm{Tr}(\mathcal{A})$, is the smallest set that contains the identity mapping and elementary translations and is closed under composition of unary

functions. The set $\mathrm{Tr}(\mathcal{A})$ equipped with the composition operation is a monoid, called the *translation monoid* of $\mathcal{A}$.

Lemma 1 ([14]). *Let $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Sigma)$ be two algebras, and $\varphi : \mathcal{A} \to \mathcal{B}$ be a morphism. The mapping φ induces a monoid morphism $\mathrm{Tr}(\mathcal{A}) \to \mathrm{Tr}(\mathcal{B})$, $p \mapsto p_\varphi$ such that $p(a)\varphi = p_\varphi(a\varphi)$ for any $a \in A$. Moreover, if φ is an epimorphism then the induced mapping is a monoid epimorphism.*

There is a bijective correspondence between the set of ΣX-contexts $\mathrm{C}_\Sigma(X)$ and translations of term algebra $\mathrm{Tr}(\mathcal{T}_\Sigma(X))$ in a natural way: an elementary context $P = f(t_1, \ldots, \xi, \ldots, t_m)$ corresponds to the translation $P^{\mathcal{T}_\Sigma(X)} = f^{\mathcal{T}_\Sigma(X)}(t_1, \ldots, \xi, \ldots, t_m)$, and the composition of contexts corresponds to the composition of translations.

Let us recall that for a relation ρ defined on a set A, by ρ^{-1} the inverse relation of ρ is denoted, i.e., $a\,\rho^{-1}\,b \Leftrightarrow b\,\rho\,a$ for any $a, b \in A$. Let ρ be a quasi order, i.e., a reflexive and transitive relation, on a set A. Then the relation $\equiv_\rho = \rho \cap \rho^{-1}$ is an equivalence on A and the relation $\leq_\rho$ defined on the factor set $A/\equiv_\rho$ by

$$a/\equiv_\rho \leq_\rho b/\equiv_\rho \Leftrightarrow\ a\,\rho\,b$$

is an order. The ordered set $(A/\equiv_\rho, \leq_\rho)$ is denoted by A/ρ.

Let $\preceq$ be a quasi order on an algebra $\mathcal{A} = (A, \Sigma)$, i.e., $\preceq$ is a quasi order on A. Then $\preceq$ is *compatible with* Σ if $a_1 \preceq b_1, \ldots, a_m \preceq b_m$ implies $f^{\mathcal{A}}(a_1, \ldots, a_m) \preceq f^{\mathcal{A}}(b_1, \ldots, b_m)$ for any $f \in \Sigma_m$ $(m > 0)$ and $a_1, \ldots, a_m, b_1, \ldots, b_m \in A$. In case when it is not necessary to emphasize the alphabet Σ, we say that $\preceq$ is a *compatible quasi order* on $\mathcal{A}$.

An *ordered Σ-algebra* is a structure $\mathcal{A} = (A, \Sigma, \leqslant)$ where (A, Σ) is a Σ-algebra and $\leqslant$ is an order on A compatible with Σ. Moreover, if a quasi order ρ defined on an algebra $\mathcal{A} = (A, \Sigma, \leqslant)$ is compatible, then $\equiv_\rho$ is a congruence on (A, Σ) and the order factor algebra is $\mathcal{A}/\rho = (A/\equiv_\rho, \Sigma, \leq_\rho)$. Compatible quasi orders containing the order of the algebra play on ordered algebras the role of congruences on ordinary algebras. We note that any algebra (A, Σ) in the classical sense is an ordered algebra (A, Σ, Δ_A) in which the order relation is equality.

For a tree language $T \subseteq \mathrm{T}_\Sigma(X)$ the relation (see [9])

$$t \preceq_T s \Leftrightarrow (\forall P \in \mathrm{C}_\Sigma(X))\ (P(s) \in T \Rightarrow P(t) \in T)$$

is a compatible quasi order on $\mathcal{T}_\Sigma(X)$. The corresponding equivalence relation is the well-known *syntactic congruence* of T, denoted by θ_T, and the corresponding order is $\leq_T$. The corresponding factor algebra is the *syntactic ordered algebra* of T, in notation $\mathrm{SOA}(T) = \mathcal{T}_\Sigma(X)/\preceq_T$. It is known that a tree language is regular if and only if its syntactic congruence has finite index, i.e., the algebra $\mathrm{SOA}(T)$ is finite. On the other hand, the compatible quasi order $\precsim_T$ is defined on $\mathrm{C}_\Sigma(X)$ by (see [9])

$$P \precsim_T Q \Leftrightarrow (\forall t \in \mathrm{T}_\Sigma(X))\ (\forall R \in \mathrm{C}_\Sigma(X))\ (RQ(t) \in T \Rightarrow RP(t) \in T)$$

and the corresponding equivalence is the m-congruence of T, in notation μ_T, ([15], definition 10.1) defined on $C_\Sigma(X)$ by

$$P\mu_T Q \Leftrightarrow (\forall t \in T_\Sigma(X))\ (\forall R \in C_\Sigma(X))\ (RQ(t) \in T \Leftrightarrow RP(t) \in T).$$

3 Regular tree languages and well-quasi orders

We are going to characterize regular tree languages in terms of well-quasi orders. Motivation for this comes from [3], where a similar result for string languages was given. There are several equivalent ways to define well-quasi orders (see [8]), but we list here only those that we are going to use. A quasi order $\preceq$ defined on a set A is a *well-quasi order* if either of the following conditions is satisfied:

(1) for each infinite sequence $\{x_i\}_{i\in\mathbb{N}}$ of elements of A there exist i and j with $i < j$ such that $x_i \preceq x_j$;
(2) each infinite sequence $\{x_i\}_{i\in\mathbb{N}}$ of elements of A contains an infinite ascending subsequence;
(3) every sequence of $\preceq$-closed subsets of A which is strictly ascending under inclusion is finite.

Recall that a subset H is *$\preceq$-closed* if $a \preceq b$ and $a \in H$ imply $b \in H$.

The following lemma contains some simple properties of well-quasi orders. Parts (a) and (b) are from [3].

Lemma 2.

(a) *If $\rho_1 \subseteq \rho_2$, ρ_1 is a well-quasi order and ρ_2 is a quasi order on A, then ρ_2 is a well-quasi order, too.*
(b) *Let ρ_1 and ρ_2 be well-quasi orders on A_1 and A_2 respectively. Then the transitive closure of $\rho_1 \cup \rho_2$ is a well-quasi order on $A_1 \cup A_2$ and $\rho_1 \times \rho_2$ is a well-quasi order on $A_1 \times A_2$.*
(c) *If ρ_1 and ρ_2 are well-quasi orders on A, then $\rho_1 \cap \rho_2$ is a well-quasi order on A, too.*

Recall that $\rho_1 \times \rho_2$ is defined on $A_1 \times A_2$ by

$$(a_1, a_2)\, \rho_1 \times \rho_2\, (b_1, b_2) \Leftrightarrow a_1\, \rho_1\, b_1 \text{ and } a_2\, \rho_2\, b_2,$$

for $a_1, b_1 \in A_1$ and $a_2, b_2 \in A_2$.

Let ρ be a quasi order on $T_\Sigma(X)$. Then the relation ρ^C defined on $C_\Sigma(X)$ by

$$P\rho^C Q \Leftrightarrow (\forall t \in T_\Sigma(X))\ P(t)\, \rho\, Q(t)$$

is a quasi order *induced* by quasi order ρ. For example, for a tree language $T \subseteq T_\Sigma(X)$ and the relations defined in Section 2, it can be proved that $\preceq_T^C = \precsim_T$ and $\theta_T^C = \mu_T$.

Theorem 3. *If θ is a congruence on $T_\Sigma(X)$, then $C_\Sigma(X)/\theta^C \cong \mathrm{Tr}(T_\Sigma(X)/\theta)$.*

Proof. Let $\pi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(X)/\theta$ be the natural epimorphism. According to Lemma 1, there is an epimorphism from $\mathrm{C}_\Sigma(X) = \mathrm{Tr}(\mathcal{T}_\Sigma(X))$ to $\mathrm{Tr}(\mathcal{T}_\Sigma(X)/\theta)$ where $P \mapsto P_\pi$ and $P_\pi(t\pi) = (P(t))\pi$ holds for all $P \in \mathrm{C}_\Sigma(X)$ and $t \in \mathrm{T}_\Sigma(X)$. Thus it suffices to prove that the kernel of this epimorphism is θ^C, i.e., that $P_\pi = Q_\pi$ if and only if $P\,\theta^C\,Q$, for any $P, Q \in \mathrm{C}_\Sigma(X)$. Indeed, assume that $P_\pi = Q_\pi$ for some $P, Q \in \mathrm{C}_\Sigma(X)$. Then $P_\pi(t\pi) = Q_\pi(t\pi)$ for every $t\pi \in \mathcal{T}_\Sigma(X)/\theta$, which is equivalent to $(P(t))\pi = (Q(t))\pi$ for every $t \in \mathcal{T}_\Sigma(X)$. This means that $P(t)\,\theta\,Q(t)$ for every $t \in \mathcal{T}_\Sigma(X)$, and so $P\,\theta^C\,Q$. □

A quasi order ρ defined on a set A is of *finite index* if $\equiv_\rho$ is of finite index, i.e., if the set A/ρ is finite. Clearly, such quasi orders are well-quasi orders.

Corollary 4. *If ρ is a compatible quasi order on $\mathcal{T}_\Sigma(X)$ of finite index, then ρ^C is of finite index as well.*

Proof. According to Theorem 3, $\mathrm{C}_\Sigma(X)/\equiv_{\rho^C}$ has as many elements as $\mathrm{Tr}(\mathcal{T}_\Sigma(X)/\equiv_\rho)$ which is finite since $\mathcal{T}_\Sigma(X)/\equiv_\rho$ is finite. □

We are ready now to prove a tree version of the generalized Myhill–Nerode's theorem (Theorem 3.3 [3]).

Theorem 5. *For a tree language $T \subseteq \mathrm{T}_\Sigma(X)$ the following conditions are equivalent:*

(i) *T is regular;*

(ii) *T is ρ-closed where ρ is a compatible well-quasi order and ρ^C is a well-quasi order too;*

(iii) *T is ρ-closed where ρ is a compatible well-quasi order on $\mathcal{T}_\Sigma(X)$ and there exists a well-quasi order on $\mathrm{C}_\Sigma(X)$ contained in ρ^C.*

Proof. (i)⇒(ii). Since T is regular, the relation θ_T is a congruence of finite index, and hence a compatible well-quasi order. The fact that T is saturated by θ_T implies that T is θ_T-closed. According to Corollary 4 it follows that θ_T^C is of finite index, and so a well-quasi order.

(ii)⇒(iii). This is obvious since ρ^C satisfies the condition.

(iii)⇒(i). Suppose that T is not regular. Then θ_T is not of finite index, and hence there exists an infinite sequence $\{t_i\}_{i\in\mathbf{N}}$ such that $t_i/\theta_T \neq t_j/\theta_T$ whenever $i \neq j$. Since ρ is a well-quasi order there exists an infinite ρ-ascending subsequence of $\{t_i\}_{i\in\mathbf{N}}$. Without losing generality we can assume that $\{t_i\}_{i\in\mathbf{N}}$ itself is ascending, i.e., $t_i\,\rho\,t_j$ whenever $i \leq j$. Using compatibility we get $P(t_i)\rho P(t_j)$ for all $P \in \mathrm{C}_\Sigma(X)$ and $i \leq j$. If $P(t_i) \in T$ then $P(t_j) \in T$ since T is ρ-closed. If we denote by $T.t^{-1}$ the set

$$T.t^{-1} = \{P \in \mathrm{C}_\Sigma(X) \mid P(t) \in T\}$$

then we get $P \in T.t_i^{-1}$ implies $P \in T.t_j^{-1}$, i.e., $T.t_i^{-1} \subseteq T.t_j^{-1}$ when $i \leq j$. Moreover, $t_i/\theta_T \neq t_j/\theta_T$ implies that $T.t_i^{-1} \subset T.t_j^{-1}$ for $i < j$. Therefore the sequence $\{T.t_i^{-1}\}_{i\in\mathbf{N}}$ is infinite.

Let ν be a well-quasi order on $C_\Sigma(X)$ contained in ρ^C. We are going to prove that the set $T.t^{-1}$ is ν-closed for any $t \in T_\Sigma(X)$. Assume that $P \nu Q$. Since $\nu \subseteq \rho^C$, then $P(t) \rho Q(t)$ for any $t \in T$. If $P \in T.t^{-1}$ then $P(t) \in T$ and since T is ρ-closed, it follows that $Q(t) \in T$, and so $Q \in T.t^{-1}$.

Finally, we have proved that $\{T.t_i^{-1}\}_{i \in \mathbf{N}}$ is an infinite ascending sequence of ν-closed sets, which contradicts the fact that ν is a well-quasi order. Therefore, T must be regular. □

For a language $T \subseteq T_\Sigma(X)$ the relation $\preceq_T^{-1}$ is the greatest compatible well-quasi order on $T_\Sigma(X)$ such that T is $\preceq_T^{-1}$-closed. Indeed, if T is ρ-closed for a compatible well-quasi order ρ on $T_\Sigma(X)$, then from $t_1 \rho t_2$ follows that $P(t_1) \rho P(t_2)$ for any $P \in C_\Sigma(X)$ and so $P(t_1) \in T$ implies $P(t_2) \in T$, i.e., $t_1 \preceq_T^{-1} t_2$, for any $t_1, t_2 \in T_\Sigma(X)$. Moreover, in case T is a regular language, $\preceq_T^{-1}$ is of finite index and, according to Corollary 4, $(\preceq_T^{-1})^C$ is of finite index too, and thus it is a well-quasi order. Hence, $\preceq_T^{-1}$ is the greatest well-quasi order on $T_\Sigma(X)$ satisfying condition (ii) of Theorem 5.

Example 6. For a tree $t \in T_\Sigma(X)$, let $\bar{t} \in (\Sigma \cup X)^*$ be the string obtained by reading symbols as they appear in t, i.e., in right Polish notation. Denote by $\leq_e$ the embedding order relation on the free monoid $(\Sigma \cup X)^*$, i.e., the relation defined by $u \leq_e v \Leftrightarrow u = u_1 u_2 \cdots u_n$, $v = v_0 u_1 v_1 u_2 \cdots v_{n-1} u_n v_n$ for $u_1, \ldots, u_n, v_0, v_1, \ldots, v_n \in (\Sigma \cup X)^*$. It is a well order. Let ρ be the relation defined on $T_\Sigma(X)$ by $t_1 \rho t_2 \Leftrightarrow \overline{t_1} \leq_e \overline{t_2}$. It can be proved that ρ is a compatible well-quasi order and ρ^C is a well-quasi order. Thus, every ρ-closed ΣX-language is regular.

4 Varieties of quasi orders

A correspondence between positive varieties of tree languages and varieties of finite ordered algebras has been given in [9]. It is known that in the case of ordinary varieties of (tree) languages and varieties of algebras the corresponding families of relations are varieties of congruences of finite index (see [14]). Results from the previous section, as well as from [9], suggest that families of relations corresponding to positive varieties of languages and varieties of ordered algebras consist of compatible well-quasi orders for which the induced relations on contexts are well-quasi orders. Moreover, the fact that we are dealing only with finite algebras restricts our attention to compatible quasi orders of finite index. According to Corollary 4, their induced quasi orders on contexts are of finite index, too.

Let us recall first necessary concepts and the Positive Variety Theorem from [9].

Let $\mathcal{A} = (A, \Sigma, \leqslant_\mathcal{A})$ and $\mathcal{B} = (B, \Sigma, \leqslant_\mathcal{B})$ be two ordered algebras. The structure $\mathcal{B}$ is an *order subalgebra* of $\mathcal{A}$ if (B, Σ) is a subalgebra of (A, Σ) and $\leqslant_\mathcal{B}$ is the restriction of $\leqslant_\mathcal{A}$ on B. A mapping $\varphi : A \to B$ is an *order morphism* if it is a Σ-morphism, i.e., if $c^\mathcal{A}\varphi = c^\mathcal{B}$ and $f^\mathcal{A}(a_1, \ldots, a_m)\varphi = f^\mathcal{B}(a_1\varphi, \ldots, a_m\varphi)$ for any $c \in \Sigma_0, f \in \Sigma_m$ $(m > 0)$ and $a_1, \ldots, a_m \in A$, and preserves the order, i.e., for any $a, b \in A$ if $a \leqslant_\mathcal{A} b$ then $a\varphi \leqslant_\mathcal{B} b\varphi$. The order morphism φ is an *order epimorphism* if

it is surjective, and then $\mathcal{B}$ is an *order image of* $\mathcal{A}$. When φ is bijective and its inverse is also an order morphism, then it is an *order isomorphism*, and $\mathcal{A} \cong \mathcal{B}$ denotes that $\mathcal{A}$ and $\mathcal{B}$ are order isomorphic. The structure $\mathcal{A} \times \mathcal{B} = (A \times B, \Sigma, \leqslant_{\mathcal{A}} \times \leqslant_{\mathcal{B}})$, where $(A \times B, \Sigma)$ is the product of the algebras (A, Σ) and (B, Σ), is the *direct product* of $\mathcal{A}$ and $\mathcal{B}$. A *variety of finite ordered algebras* is a class of finite ordered algebras closed under order subalgebras, order images and direct products.

Let A and B be arbitrary sets. For a mapping $\phi : A \to B$ and a relation ρ on B the relation $\phi \circ \rho \circ \phi^{-1}$ is defined on A by

$$(a, b) \in \phi \circ \rho \circ \phi^{-1} \Leftrightarrow (a\phi, b\phi) \in \rho.$$

Lemma 7. *For ordered algebras* $\mathcal{A} = (A, \Sigma, \leqslant_{\mathcal{A}})$ *and* $\mathcal{B} = (B, \Sigma, \leqslant_{\mathcal{B}})$ *and order morphism* $\varphi : \mathcal{A} \to \mathcal{B}$, *if* $\preccurlyeq$ *is a compatible quasi order on* $\mathcal{B}$ *containing* $\leqslant_{\mathcal{B}}$, *then the relation* $\varphi \circ \preccurlyeq \circ \varphi^{-1}$ *is a compatible quasi order on* $\mathcal{A}$ *containing* $\leqslant_{\mathcal{A}}$. *Moreover, if* φ *is an order epimorphism then* $\mathcal{A}/(\varphi \circ \preccurlyeq \circ \varphi^{-1}) \cong \mathcal{B}/\preccurlyeq$.

Let us recall that for a tree language $T \subseteq \mathrm{T}_\Sigma(X)$, a context $P \in \mathrm{C}_\Sigma(X)$, and a Σ-morphism $\varphi : \mathcal{T}_\Sigma(Y) \to \mathcal{T}_\Sigma(X)$, the inverse translation of T under P is $P^{-1}(T) = \{t \in \mathrm{T}_\Sigma(X) \mid P(t) \in T\}$, and the inverse morphism of T under φ is $T\varphi^{-1} = \{t \in \mathrm{T}_\Sigma(Y) \mid t\varphi \in T\}$ (cf. [14]). An indexed family of recognizable tree languages $\mathscr{V} = \{\mathscr{V}(X)\}$ is a *positive variety of tree languages* if it is closed under positive Boolean operations (intersection and union), inverse translations and inverse morphisms.

Theorem 8 (Positive Variety Theorem [9]**).** *For a positive variety of tree languages* $\mathscr{V}$, *let* $\mathscr{V}^{\mathrm{a}}$ *be the variety of finite ordered algebras generated by syntactic ordered algebras of tree languages in* $\mathscr{V}$. *For a variety of finite ordered algebras* $\mathscr{K}$ *let the indexed family* $\mathscr{K}^{\mathrm{t}} = \{\mathscr{K}^{\mathrm{t}}(X)\}$ *be defined by* $\mathscr{K}^{\mathrm{t}}(X) = \{T \subseteq \mathrm{T}_\Sigma(X) \mid \mathrm{SOA}(T) \in \mathscr{K}\}$. *The mappings* $\mathscr{K} \mapsto \mathscr{K}^{\mathrm{t}}$ *and* $\mathscr{V} \mapsto \mathscr{V}^{\mathrm{a}}$ *are mutually inverse lattice isomorphisms between the class of all varieties of finite ordered algebras and the class of all positive varieties of recognizable tree languages.*

Let us denote by $\mathrm{FQ}(X)$ the set of all compatible quasi orders of finite index defined on $\mathrm{T}_\Sigma(X)$.

Lemma 9. *Let* $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$ *be a morphism.*

(a) *If* $\rho \in \mathrm{FQ}(Y)$ *then* $\phi \circ \rho \circ \phi^{-1} \in \mathrm{FQ}(X)$.

(b) *If* $T \subseteq \mathcal{T}_\Sigma(Y)$ *then*

$$\bigcap_{P \in \mathrm{C}_\Sigma(Y)} \preceq^{-1}_{P^{-1}(T)\phi^{-1}} \subseteq \phi \circ \preceq^{-1}_T \circ \phi^{-1}.$$

Moreover, if T *is regular then the intersection can be taken over a finite subset of* $\mathrm{C}_\Sigma(Y)$.

Proof. (a) Clearly $\phi \circ \rho \circ \phi^{-1}$ is reflexive and transitive. Let us prove that it is compatible. Assume $t_1(\phi \circ \rho \circ \phi^{-1})t_2$, i.e., $(t_1\phi)\,\rho\,(t_2\phi)$. Compatibility of ρ implies

that $Q(t_1\phi)\,\rho\,Q(t_2\phi)$ for any $Q \in \mathrm{C}_\Sigma(Y)$. In particular, for any $P \in \mathrm{C}_\Sigma(X)$ we have $P_\phi(t_1\phi)\,\rho\,P_\phi(t_2\phi)$, and so $P(t_1)\,(\phi \circ \rho \circ \phi^{-1})P(t_2)$.

It remains to prove that $\phi \circ \rho \circ \phi^{-1}$ has a finite index. It is easy to prove that $\equiv_{\phi\circ\rho\circ\phi^{-1}} = \phi\circ \equiv_\rho \circ\phi^{-1}$. Therefore the mapping $t/\equiv_{\phi\circ\rho\circ\phi^{-1}} \mapsto t\phi/\equiv_\rho$ is a well-defined one-to-one mapping. Moreover, it is a bijection onto $\mathcal{T}_\Sigma(X)\phi/\equiv_\rho$. Therefore, $|\mathcal{T}_\Sigma(X)/\equiv_{\phi\circ\rho\circ\phi^{-1}}| = |\mathcal{T}_\Sigma(X)\phi/\equiv_\rho| \le |\mathcal{T}_\Sigma(Y)/\equiv_\rho|$ and this number is finite.

(b) The following proves the claim:

$$\begin{aligned}
(t_1,t_2) \;&\in\; \textstyle\bigcap_{P\in \mathrm{C}_\Sigma(Y)} \preceq^{-1}_{P^{-1}(T)\phi^{-1}} \Leftrightarrow \\
&\Leftrightarrow\; (\forall P \in \mathrm{C}_\Sigma(Y))\; t_1 \preceq^{-1}_{P^{-1}(T)\phi^{-1}} t_2 \\
&\Leftrightarrow\; (\forall P \in \mathrm{C}_\Sigma(Y))\,(\forall Q \in \mathrm{C}_\Sigma(X)) \\
&\qquad (Q(t_1) \in P^{-1}(T)\phi^{-1} \Rightarrow Q(t_2) \in P^{-1}(T)\phi^{-1}) \\
&\Rightarrow\; (\forall P \in \mathrm{C}_\Sigma(Y))\; (t_1 \in P^{-1}(T)\phi^{-1} \Rightarrow t_2 \in P^{-1}(T)\phi^{-1}) \\
&\Leftrightarrow\; (\forall P \in \mathrm{C}_\Sigma(Y))\; (t_1\phi \in P^{-1}(T) \Rightarrow t_2\phi \in P^{-1}(T)) \\
&\Leftrightarrow\; (\forall P \in \mathrm{C}_\Sigma(Y))\; (P(t_1\phi) \in T \Rightarrow P(t_2\phi) \in T) \\
&\Leftrightarrow\; (t_1\phi) \preceq^{-1}_T (t_2\phi) \\
&\Leftrightarrow\; t_1(\phi\circ \preceq^{-1}_T \circ\phi^{-1})t_2
\end{aligned}$$

Let us define a relation ν on $\mathrm{C}_\Sigma(Y)$ by $P\,\nu\,Q \Leftrightarrow P^{-1}(T) = Q^{-1}(T)$. Clearly, ν is an equivalence and $\mu_T \subseteq \nu$. In case T is regular μ_T has finite index, and hence ν has finite index. Therefore, there can be only finitely many different sets of the form $P^{-1}(T)$. □

A family $\mathscr{R} = \{\mathscr{R}(X)\}$, where $\mathscr{R}(X)$ is a set of compatible quasi orders on $\mathcal{T}_\Sigma(X)$ of finite index, is a *variety of quasi orders* if

(1) $\rho_1, \rho_2 \in \mathscr{R}(X)$ then $\rho_1 \cap \rho_2 \in \mathscr{R}(X)$ for any X;
(2) $\rho_1 \subseteq \rho_2$ and $\rho_1 \in \mathscr{R}(X)$ then $\rho_2 \in \mathscr{R}(X)$ for any X;
(3) $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$ is a morphism and $\rho \in \mathscr{R}(Y)$ then $\phi \circ \rho \circ \phi^{-1} \in \mathscr{R}(X)$.

In other words, $\mathscr{R}(X)$ is a filter of the lattice $\mathrm{FQ}(X)$ satisfying condition (3).

Lemma 10. *Let $\mathscr{V} = \{\mathscr{V}(X)\}$ be a positive variety of tree languages. Let $\mathscr{V}^{\mathrm{r}}(X)$ be the filter in the lattice $\mathrm{FQ}(X)$ generated by the set $\{\preceq^{-1}_T \mid T \in \mathscr{V}(X)\}$. Then $\mathscr{V}^{\mathrm{r}} = \{\mathscr{V}^{\mathrm{r}}(X)\}$ is a variety of quasi orders.*

Proof. Conditions (1) and (2) from the definition of varieties of quasi orders are fulfilled by the way $\mathscr{V}^{\mathrm{r}}$ is defined. Assume that $\rho \in \mathscr{V}^{\mathrm{r}}(Y)$ and $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$ is a morphism. Since $\rho \in \mathscr{V}^{\mathrm{r}}(Y)$ there are languages $T_1, \ldots, T_n \in \mathscr{V}(Y)$, $n \in \mathbb{N}$, such that $\cap_{k=1}^n \preceq^{-1}_{T_k} \subseteq \rho$. For a language $T_k \in \mathscr{V}(Y)$ and any $P \in \mathrm{C}_\Sigma(Y)$ we have that $P^{-1}(T_k) \in \mathscr{V}(Y)$, and then $P^{-1}(T_k)\phi^{-1} \in \mathscr{V}(X)$. This implies that $\preceq^{-1}_{P^{-1}(T_k)\phi^{-1}} \in \mathscr{V}^{\mathrm{r}}(X)$. Since T_k is regular, the family $\{P^{-1}(T_k)\phi^{-1} \in \mathscr{V}(X) \mid P \in \mathrm{C}_\Sigma(Y)\}$ is finite. Therefore, $\phi\circ \preceq^{-1}_{T_k} \circ\phi^{-1} \in \mathscr{V}^{\mathrm{r}}(X)$ according to Lemma 9. Now from $\cap_{k=1}^n \preceq^{-1}_{T_k} \subseteq \rho$ follows that $\cap_{k=1}^n(\phi\circ \preceq^{-1}_{T_k} \circ\phi^{-1}) \subseteq \phi \circ \rho \circ \phi^{-1}$, and so $\phi \circ \rho \circ \phi^{-1} \in \mathscr{V}^{\mathrm{r}}(X)$. □

Lemma 11. *Let $\mathscr{R} = \{\mathscr{R}(X)\}$ be a variety of quasi orders. Let us denote $\mathscr{R}^{\mathrm{t}}(X) = \{T \subseteq \mathrm{T}_\Sigma(X) \mid \preceq_T^{-1} \in \mathscr{R}(X)\}$. Then $\mathscr{R}^{\mathrm{t}} = \{\mathscr{R}^{\mathrm{t}}(X)\}$ is a positive variety of tree languages.*

Proof. According to Theorem 5 it follows that languages belonging to the family are regular. From $\preceq_{T_1}^{-1} \cap \preceq_{T_2}^{-1} \subseteq \preceq_{T_1 \cap T_2}^{-1}$ and $\preceq_{T_1}^{-1} \cap \preceq_{T_2}^{-1} \subseteq \preceq_{T_1 \cup T_2}^{-1}$ it follows that $\mathscr{R}^{\mathrm{t}}(X)$ is closed for positive Boolean operations. Similarly, $\preceq_T^{-1} \subseteq \preceq_{P^{-1}(T)}^{-1}$ implies closure for quotients. Finally, if $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$ is a morphism and $T \in \mathscr{R}^{\mathrm{t}}(Y)$ then $\preceq_T^{-1} \in \mathscr{R}(Y)$, and so $\phi \circ \preceq_T^{-1} \circ \phi^{-1} \in \mathscr{R}(X)$. It is easy to prove that $\phi \circ \preceq_T^{-1} \circ \phi^{-1} \subseteq \preceq_{T\phi^{-1}}^{-1}$, which further implies $\preceq_{T\phi^{-1}}^{-1} \in \mathscr{R}(X)$, and hence $T\phi^{-1} \in \mathscr{R}^{\mathrm{t}}(X)$. □

Lemma 12. *For positive varieties of tree languages $\mathscr{V} = \{\mathscr{V}(X)\}$, $\mathscr{V}_1 = \{\mathscr{V}_1(X)\}$ and $\mathscr{V}_2 = \{\mathscr{V}_2(X)\}$, and varieties of quasi orders $\mathscr{R} = \{\mathscr{R}(X)\}$, $\mathscr{R}_1 = \{\mathscr{R}_1(X)\}$ and $\mathscr{R}_2 = \{\mathscr{R}_2(X)\}$, the following hold:*

(a) $\mathscr{V} = \mathscr{V}^{\mathrm{rt}}$;
(b) $\mathscr{R} = \mathscr{R}^{\mathrm{tr}}$;
(c) $\mathscr{V}_1 \subseteq \mathscr{V}_2$ *implies* $\mathscr{V}_1^{\mathrm{r}} \subseteq \mathscr{V}_2^{\mathrm{r}}$;
(d) $\mathscr{R}_1 \subseteq \mathscr{R}_2$ *implies* $\mathscr{R}_1^{\mathrm{t}} \subseteq \mathscr{R}_2^{\mathrm{t}}$.

Proof. (a) The inclusion $\mathscr{V} \subseteq \mathscr{V}^{\mathrm{rt}}$ is obvious. Assume now that $T \in \mathscr{V}^{\mathrm{rt}}(X)$. Then $\preceq_T^{-1} \in \mathscr{V}^{\mathrm{r}}(X)$. This means that there are languages $T_1, \ldots, T_n \in \mathscr{V}(X)$, $n \in \mathbb{N}$, such that $\cap_{k=1}^n \preceq_{T_k}^{-1} \subseteq \preceq_T^{-1}$, which implies that $\mathrm{SOA}(T)$ is an order image of an order subalgebra of $\mathrm{SOA}(T_1) \times \cdots \times \mathrm{SOA}(T_n)$. Now $\mathrm{SOA}(T_1), \ldots, \mathrm{SOA}(T_n) \in \mathscr{V}^{\mathrm{a}}$ and $\mathscr{V}^{\mathrm{a}}$ is a variety of ordered algebras, which implies that $\mathrm{SOA}(T) \in \mathscr{V}^{\mathrm{a}}$, and hence $T \in \mathscr{V}^{\mathrm{at}}(X) = \mathscr{V}(X)$, according to Theorem 8.

(b) It is easy to check that $\mathscr{R}^{\mathrm{tr}} \subseteq \mathscr{R}$. Consider now $\rho \in \mathscr{R}(X)$. Since ρ has finite index, there are finitely many ρ-closed sets. Let $T_1, \ldots, T_n$, $n \in \mathbb{N}$, be all of them. We are going to prove that $\cap_{k=1}^n \preceq_{T_k}^{-1} \subseteq \rho$. Assume that $t, s \in \mathrm{T}_\Sigma(X)$ are such that $t \rho s$ does not hold. Then the set $\{t' \in \mathrm{T}_\Sigma(X) \mid t \rho t'\}$ is ρ-closed and hence equal to some T_i, and so $t \not\preceq_{T_i}^{-1} s$ does not hold, i.e., $(t, s) \notin \cap_{k=1}^n \preceq_{T_k}^{-1}$. On the other hand, $\rho \subseteq \preceq_{T_k}^{-1}$ for every $k \in \{1, \ldots, n\}$ since T_k is ρ-closed and $\preceq_{T_k}^{-1}$ is the greatest such well-quasi order. Therefore, $\preceq_{T_k}^{-1} \in \mathscr{R}(X)$ which implies $T_k \in \mathscr{R}^{\mathrm{t}}(X)$, this further gives $\preceq_{T_k}^{-1} \in \mathscr{R}^{\mathrm{tr}}(X)$, which finally, together with $\cap_{k=1}^n \preceq_{T_k}^{-1} \subseteq \rho$, implies $\rho \in \mathscr{R}^{\mathrm{tr}}(X)$.

(c) and (d) are obvious. □

Summing up the results from Lemmas 10, 11, 12 we get the following variety theorem.

Theorem 13. *For a positive variety of tree languages $\mathscr{V} = \{\mathscr{V}(X)\}$, let $\mathscr{V}^{\mathrm{r}}(X)$ be the filter of the lattice $\mathrm{FQ}(X)$ generated by the set*

$$\{\preceq_T^{-1} \mid T \in \mathscr{V}(X)\}.$$

On the other hand, for a variety of quasi orders $\mathscr{R} = \{\mathscr{R}(X)\}$, let us denote

$$\mathscr{R}^{\mathrm{t}}(X) = \{T \subseteq \mathrm{T}_\Sigma(X) \mid \preceq_T^{-1} \in \mathscr{R}(X)\}.$$

The mappings $\mathcal{V} \mapsto \mathcal{V}^{\mathrm{r}} = \{\mathcal{V}^{\mathrm{r}}(X)\}$ and $\mathcal{R} \mapsto \mathcal{R}^{\mathrm{t}} = \{\mathcal{R}^{\mathrm{t}}(X)\}$ are mutually inverse lattice isomorphisms between the lattices of all positive varieties of tree languages and all varieties of quasi orders.

The next theorem establishes a similar result for varieties of finite ordered algebras and varieties of quasi orders. First we need to prove several lemmas.

Lemma 14. *Let $\mathcal{K}$ be a variety of finite ordered Σ-algebras. Let $\mathcal{K}^{\mathrm{r}}(X) = \{\rho \in \mathrm{FQ}(X) \mid \mathcal{T}_\Sigma(X)/\rho \in \mathcal{K}\}$. Then $\mathcal{K}^{\mathrm{r}} = \{\mathcal{K}^{\mathrm{r}}(X)\}$ is a variety of quasi orders.*

Proof. Let $\rho_1, \rho_2 \in \mathcal{K}^{\mathrm{r}}(X)$. Then $\mathcal{T}_\Sigma(X)/(\rho_1 \cap \rho_2)$ is an order image of an order subalgebra of $\mathcal{T}_\Sigma(X)/\rho_1 \times \mathcal{T}_\Sigma(X)/\rho_2$, and hence $\mathcal{T}_\Sigma(X)/\rho_1, \mathcal{T}_\Sigma(X)/\rho_2 \in \mathcal{K}$ imply $\mathcal{T}_\Sigma(X)/(\rho_1 \cap \rho_2) \in \mathcal{K}$, what means $\rho_1 \cap \rho_2 \in \mathcal{V}^{\mathrm{r}}(X)$. Similarly, if $\rho_1 \in \mathcal{K}^{\mathrm{r}}(X)$ and $\rho_1 \subseteq \rho_2$ then $\mathcal{T}_\Sigma(X)/\rho_2$ is an order image of $\mathcal{T}_\Sigma(X)/\rho_1 \in \mathcal{K}$, and so $\mathcal{T}_\Sigma(X)/\rho_2 \in \mathcal{K}$, which implies $\rho_2 \in \mathcal{K}^{\mathrm{r}}(X)$.

Consider now $\rho \in \mathcal{K}^{\mathrm{r}}(Y)$ and a morphism $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$. The mapping $\psi : \mathcal{T}_\Sigma(X)/(\phi \circ \rho \circ \phi^{-1}) \to \mathcal{T}_\Sigma(Y)/\rho$ defined by $t/(\phi \circ \equiv_\rho \circ \phi^{-1}) \mapsto (t\phi)/\equiv_\rho$ is an order isomorphism from $\mathcal{T}_\Sigma(X)/(\phi \circ \rho \circ \phi^{-1})$ to $\mathcal{T}_\Sigma(X)\phi/\rho$, which is an order subalgebra of $\mathcal{T}_\Sigma(Y)/\rho$. Therefore, $\mathcal{T}_\Sigma(Y)/\rho \in \mathcal{K}$ implies $\mathcal{T}_\Sigma(X)/(\phi \circ \rho \circ \phi^{-1}) \in \mathcal{K}$, and so $\phi \circ \rho \circ \phi^{-1} \in \mathcal{K}^{\mathrm{r}}(X)$. □

Lemma 15. *Let $\mathcal{R} = \{\mathcal{R}(X)\}$ be a variety of quasi orders. Let $\mathcal{R}^{\mathrm{a}}$ be the set of all ordered Σ-algebras $\mathcal{A}$ such that $\mathcal{A} \cong \mathcal{T}_\Sigma(X)/\rho$ for some X and $\rho \in \mathcal{R}(X)$. Then $\mathcal{R}^{\mathrm{a}}$ is a variety of finite ordered algebras.*

Proof. Let us notice first that for any order algebra $\mathcal{A} \cong \mathcal{T}_\Sigma(X)/\rho$ for some alphabet X and a compatible quasi order ρ, there exists an epimorphism $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ such that $\rho = \phi \circ \leq_{\mathcal{A}} \circ \phi^{-1}$, where $\leq_{\mathcal{A}}$ is the order of $\mathcal{A}$. Indeed, if $\pi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(X)/\rho$ is the natural epimorphism defined by $t \mapsto t/\equiv_\rho$, and $\psi : \mathcal{T}_\Sigma(X)/\rho \to \mathcal{A}$ is an order isomorphism, then $\pi\psi : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ is an epimorphism and $\rho = (\pi\psi) \circ \leq_{\mathcal{A}} \circ (\pi\psi)^{-1}$.

Consider now $\mathcal{A} \in \mathcal{R}^{\mathrm{a}}$. Then there exists an alphabet X and $\rho \in \mathcal{R}(X)$ such that $\mathcal{A} \cong \mathcal{T}_\Sigma(X)/\rho$, and let $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ be an order epimorphism such that $\rho = \phi \circ \leq_{\mathcal{A}} \circ \phi^{-1}$.

Let $\mathcal{B}$ be an order subalgebra of $\mathcal{A}$. Then there exists a finitely generated order subalgebra $\mathcal{C}$ of $\mathcal{T}_\Sigma(X)$ such that $\mathcal{B}$ is the order image of $\mathcal{C}$ under epimorphism ϕ. Let Y be a finite alphabet such that there exists an order epimorphism $\psi : \mathcal{T}_\Sigma(Y) \to \mathcal{C}$. Therefore, the mapping $\psi\phi : \mathcal{T}_\Sigma(Y) \to \mathcal{B}$ is an order epimorphism and $\mathcal{B} \cong \mathcal{T}_\Sigma(Y)/((\psi\phi) \circ (\leq_{\mathcal{B}}) \circ (\psi\phi)^{-1})$ where $\leq_{\mathcal{B}}$ is the restriction of $\leq_{\mathcal{A}}$ on B. It is easy to check that $\mathcal{B} \cong \mathcal{T}_\Sigma(Y)/((\psi\phi) \circ (\leq_{\mathcal{B}}) \circ (\psi\phi)^{-1}) = \mathcal{T}_\Sigma(Y)/((\psi\phi) \circ \leq_{\mathcal{A}} \circ (\psi\phi)^{-1})$. Now $\mathcal{A} \in \mathcal{R}^{\mathrm{a}}$ implies $\phi \circ \leq_{\mathcal{A}} \circ \phi^{-1} = \rho \in \mathcal{R}(X)$, what further implies $(\psi\phi) \circ \leq_{\mathcal{A}} \circ (\psi\phi)^{-1} = \psi \circ (\phi \circ \leq_{\mathcal{A}} \circ \phi^{-1}) \circ \psi^{-1} \in \mathcal{R}(Y)$. Therefore, $\mathcal{B} \cong \mathcal{T}_\Sigma(Y)/((\psi\phi) \circ \leq_{\mathcal{B}} \circ (\psi\phi)^{-1}) \in \mathcal{R}^{\mathrm{a}}$.

Assume now that $\mathcal{B}$ is an order image of $\mathcal{A}$ and let $\psi : \mathcal{A} \to \mathcal{B}$ be the order epimorphism. Then $\phi\psi : \mathcal{T}_\Sigma(X) \to \mathcal{B}$ is an order epimorphism. If $\leq_{\mathcal{B}}$ is the order of $\mathcal{B}$, then $\mathcal{B} \cong \mathcal{T}_\Sigma(X)/((\phi\psi) \circ \leq_{\mathcal{B}} \circ (\phi\psi)^{-1})$. From the fact that ψ is an order

morphism, it follows that $\leq_{\mathcal{A}} \subseteq \psi \circ \leq_{\mathcal{B}} \circ\psi^{-1}$. This further implies $\rho = \phi \circ \leq_{\mathcal{A}} \circ\phi^{-1} \subseteq \phi \circ \psi \circ \leq_{\mathcal{B}} \circ\psi^{-1} \circ \phi^{-1} \in \mathscr{R}(X)$, and so $\mathcal{B} \in \mathscr{R}^{\mathrm{a}}$.

Consider now two ordered algebras $\mathcal{A}_1, \mathcal{A}_2 \in \mathscr{R}^{\mathrm{a}}$. Let $\leq_1, \leq_2$ be their orders respectively, and X_1 and X_2 alphabets for which there are quasi orders $\rho_1 \in \mathscr{R}(X_1)$ and $\rho_2 \in \mathscr{R}(X_2)$ such that $\mathcal{A}_1 \cong \mathcal{T}_\Sigma(X_1)/\rho_1$ and $\mathcal{A}_2 \cong \mathcal{T}_\Sigma(X_2)/\rho_2$, respectively. Denote by $\pi_1 : \mathcal{T}_\Sigma(X_1) \to \mathcal{A}_1$ and $\pi_2 : \mathcal{T}_\Sigma(X_2) \to \mathcal{A}_2$, respectively, order epimorphisms such that $\rho_1 = \pi_1 \circ \leq_1 \circ\pi_1^{-1}$ and $\rho_2 = \pi_2 \circ \leq_2 \circ\pi_2^{-1}$. Let Y be a finite alphabet such that there is an epimorphism $\psi : \mathcal{T}_\Sigma(Y) \to \mathcal{T}_\Sigma(X_1) \times \mathcal{T}_\Sigma(X_2)$, and let $\psi_1 : \mathcal{T}_\Sigma(Y) \to \mathcal{T}_\Sigma(X_1)$ and $\psi_2 : \mathcal{T}_\Sigma(Y) \to \mathcal{T}_\Sigma(X_2)$ be the projection mappings of ψ. Then the mapping $\Phi : \mathcal{T}_\Sigma(Y) \to \mathcal{A}_1 \times \mathcal{A}_2$ whose projection mappings are $\Phi_1 = \psi_1\pi_1$ and $\Phi_2 = \psi_2\pi_2$ is an order epimorphism and $\mathcal{A}_1 \times \mathcal{A}_2 \cong \mathcal{T}_\Sigma(Y)/(\Phi \circ (\leq_1 \times \leq_2) \circ \Phi^{-1})$. It can be easily checked that $\Phi \circ (\leq_1 \times \leq_2) \circ \Phi^{-1} = (\Phi_1 \circ \leq_1 \circ\Phi_1^{-1}) \cap (\Phi_2 \circ \leq_2 \circ\Phi_2^{-1})$. Now $\Phi_1 \circ \leq_1 \circ\Phi_1^{-1} = \psi_1 \circ \pi_1 \circ \leq_1 \circ\pi_1^{-1} \circ \psi_1^{-1} = \psi_1 \circ \rho_1 \circ \psi_1^{-1} \in \mathscr{R}(Y)$ since $\rho_1 \in \mathscr{R}(X_1)$. Similarly, $\Phi_2 \circ \leq_2 \circ\Phi_2^{-1} \in \mathscr{R}(Y)$, and hence $\Phi \circ (\leq_1 \times \leq_2) \circ \Phi^{-1} \in \mathscr{R}(Y)$ what implies $\mathcal{A} \times \mathcal{B} \in \mathscr{R}^{\mathrm{a}}$.

Therefore, $\mathscr{R}^{\mathrm{a}}$ is a variety of finite ordered algebras. □

Lemma 16. *For varieties of finite ordered algebras $\mathscr{K}$, $\mathscr{K}_1$ and $\mathscr{K}_2$, and varieties of quasi orders $\mathscr{R} = \{\mathscr{R}(X)\}$, $\mathscr{R}_1 = \{\mathscr{R}_1(X)\}$ and $\mathscr{R}_2 = \{\mathscr{R}_2(X)\}$, the following hold:*

(a) $\mathscr{K} = \mathscr{K}^{\mathrm{ra}}$;
(b) $\mathscr{R} = \mathscr{R}^{\mathrm{ar}}$;
(c) $\mathscr{K}_1 \subseteq \mathscr{K}_2$ *implies* $\mathscr{K}_1^{\mathrm{r}} \subseteq \mathscr{K}_2^{\mathrm{r}}$;
(d) $\mathscr{R}_1 \subseteq \mathscr{R}_2$ *implies* $\mathscr{R}_1^{\mathrm{a}} \subseteq \mathscr{R}_2^{\mathrm{a}}$.

Proof. It is easy to check (a), (c), (d) and the inclusion $\mathscr{R}(X) \subseteq \mathscr{R}^{\mathrm{ar}}(X)$ for any X.

Consider $\rho \in \mathscr{R}^{\mathrm{ar}}(X)$. Then $\mathcal{A} = \mathcal{T}_\Sigma(X)/\rho \in \mathscr{R}^{\mathrm{a}}$, which further implies that $\mathcal{A} \cong \mathcal{T}_\Sigma(Y)/\mu$ for some alphabet Y and $\mu \in \mathscr{R}(Y)$. Let $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ and $\psi : \mathcal{T}_\Sigma(Y) \to \mathcal{A}$ be order epimorphisms such that $\rho = \phi \circ \leq_{\mathcal{A}} \circ\phi^{-1}$ and $\mu = \psi \circ \leq_{\mathcal{A}} \circ\psi^{-1}$, where $\leq_{\mathcal{A}}$ is the order of $\mathcal{A}$. Let us define the morphism $\Phi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$ so that $x\Phi \in x\phi\psi^{-1}$ for any $x \in X$. Then $\phi = \Phi\psi$ and so $\phi \circ \leq_{\mathcal{A}} \circ\phi^{-1} = (\Phi\psi) \circ \leq_{\mathcal{A}} \circ(\Phi\psi)^{-1}$, i.e., $\rho = \Phi \circ \mu \circ \Phi^{-1} \in \mathscr{R}(X)$ since $\mu \in \mathscr{R}(Y)$. □

As a corollary of Lemmas 14, 15, 16 we get the following variety theorem for algebras and relations.

Theorem 17. *For a variety of finite ordered Σ-algebras $\mathscr{K}$, let us define*

$$\mathscr{K}^{\mathrm{r}}(X) = \{\rho \in \mathrm{FQ}(X) \mid \mathcal{T}_\Sigma(X)/\rho \in \mathscr{K}\}.$$

For a variety of quasi orders $\mathscr{R} = \{\mathscr{R}(X)\}$, let $\mathscr{R}^{\mathrm{a}}$ be the set of all ordered Σ-algebras $\mathcal{A}$ such that $\mathcal{A} \cong \mathcal{T}_\Sigma(X)/\rho$ for some alphabet X and $\rho \in \mathscr{R}(X)$.
The mappings $\mathscr{K} \mapsto \mathscr{K}^{\mathrm{r}} = \{\mathscr{K}^{\mathrm{r}}(X)\}$ and $\mathscr{R} \mapsto \mathscr{R}^{\mathrm{a}}$ are mutually inverse lattice isomorphisms between the lattices of all varieties of finite ordered algebras and all varieties of quasi orders.

The correspondences established here are similar to those used in [14] between varieties of tree languages, varieties of finite algebras and varieties of finite congruences. However, in [14] the variety of algebras assigned to a variety of finite congruences was generated by a family which resembles our family $\mathscr{R}^{\mathrm{a}}$, and it has been shown here that the family already forms a variety of finite ordered algebras.

Example 18. Ordered nilpotent algebras and cofinite tree language were introduced in [9]. Namely, an ordered algebra $\mathcal{A} = (A, \Sigma, \leqslant)$ is *ordered n-nilpotent*, $n \in \mathbb{N}$, if $p_1 \cdots p_n(a) \leqslant b$ holds for all $a, b \in A$ and non-trivial translations $p_1, \ldots, p_n$ of $\mathcal{A}$, and it is *ordered nilpotent* if it is ordered n-nilpotent for some $n \in \mathbb{N}$. A non-empty tree language $T \subseteq \mathrm{T}_\Sigma(X)$ is *cofinite* if its complement $\mathrm{T}_\Sigma(X) \setminus T$ is finite. The family of cofinite tree languages for all leaf alphabets X is a positive variety of tree languages and finite ordered nilpotent algebras form the corresponding variety of finite ordered algebras. Let ρ_n, $n \in \mathbb{N}$, be the relation on $\mathrm{T}_\Sigma(X)$ defined by

$$t \,\rho_n s \Leftrightarrow \mathrm{hg}(s) \geq n \text{ or } t = s$$

where $\mathrm{hg}(s)$ is the height of s. It is easy to show that ρ_n is a compatible quasi order of finite index for every $n \in \mathbb{N}$, and a tree language T is cofinite if and only if $\rho_n \subseteq \preceq_T^{-1}$ for some $n \in \mathbb{N}$. Therefore, the corresponding variety of quasi orders is $\mathscr{R} = \{\mathscr{R}(X)\}$, where $\mathscr{R}(X)$ is the filter of $\mathrm{FQ}(X)$ generated by $\{\rho_n \mid n \in \mathbb{N}\}$.

Example 19. Symbolic algebras and symbolic tree languages were introduced in [9]. An algebra $\mathcal{A} = (A, \Sigma, \leq_{\mathcal{A}})$ is *symbolic* if it satisfies the following: for every $f, g \in \Sigma$ and $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, a \in A$, where boldface letters stand for appropriately long sequences of elements from A:

$$f^{\mathcal{A}}(\mathbf{a}, f^{\mathcal{A}}(\mathbf{a}, a, \mathbf{b}), \mathbf{b}) = f^{\mathcal{A}}(\mathbf{a}, a, \mathbf{b});$$
$$f^{\mathcal{A}}(\mathbf{a}, g^{\mathcal{A}}(\mathbf{c}, a, \mathbf{d}), \mathbf{b}) = g^{\mathcal{A}}(\mathbf{c}, f^{\mathcal{A}}(\mathbf{a}, a, \mathbf{b}), \mathbf{d});$$
$$f^{\mathcal{A}}(\mathbf{a}, a, \mathbf{b}) \leq_{\mathcal{A}} a.$$

For a tree $t \in \mathrm{T}_\Sigma(X)$, the *contents* $\mathrm{c}(t)$ of t is the set of symbols from $\Sigma \cup X$ which appear in t. For a subset $Z \subseteq \Sigma \cup X$, the tree language $T(Z)$ consists of all trees which contain at least one appearance of each symbol from Z. A tree language $T \subseteq \mathrm{T}_\Sigma(X)$ is *symbolic* if it is a union of tree languages of the form $T(Z)$ for some subsets $Z \subseteq \Sigma \cup X$. It was shown in [9] that symbolic tree languages form a positive variety of tree languages, symbolic algebras form a variety of finite ordered algebras and that the positive variety of symbolic tree languages corresponds to this variety of ordered algebras. It can be easily proved that the relation ρ defined on $\mathrm{T}_\Sigma(X)$ by

$$t \,\rho\, s \Leftrightarrow c(t) \subseteq c(s)$$

is a compatible quasi order of finite index, and a tree language T is symbolic if and only if $\rho \subseteq \preceq_T^{-1}$. Therefore, the variety of quasi orders corresponding to the classes of symbolic tree languages and symbolic algebras consists of filters of $\mathrm{FQ}(X)$ generated by ρ, i.e., $\mathscr{R}(X) = \{\sigma \in \mathrm{FQ}(X) \mid \rho \subseteq \sigma\}$.

References

[1] J. Almeida, On pseudovarieties, varieties of languages, filters of congruences, pseudoidentities and related topics, *Algebra Universalis* **27** (1990), 333–350.

[2] S. Burris and H. P. Sankappanavar, *A course in universal algebra*, Springer-Verlag, New York, 1981.

[3] A. Ehrenfeucht, D. Haussler, G. Rozenberg, On regularity of context-free languages, *Theoretical Computer Science* **27** (1983), 311–332.

[4] S. Eilenberg, *Automata, Languages, and Machines*, Vol. **B.** Pure and Applied Mathematics, Vol. 59, Academic Press, New York – London (1976).

[5] Z. Ésik, A variety theorem for trees and theories, in: *Automata and formal languages* VIII (Salgótarján, 1996), *Publ. Math. Debrecen* **54** (1999), 711–762.

[6] Z. Ésik and P. Weil, Algebraic recognizability of regular tree languages, *Theoretical Computer Science* **340** (2005), 291–321.

[7] F. Gécseg, B. Imreh: On Monotone Automata and Monotone Languages, *Journal of Automata, Languages and Combinatorics* **7** (1) (2002), 71–82.

[8] G. Higman, Ordering by divisibility in abstract algebras, *Proc. London Math. Soc.* **3** (2) (1952), 326–336.

[9] T. Petković and S. Salehi, Positive Varieties of Tree Languages, *Theoretical Computer Science* **347**/1-2 (2005), 1-35.

[10] J. E. Pin, *Varieties of formal languages*, Foundations of Computer Science, Plenum Publishing Corp., New York, 1986.

[11] J. E. Pin, A variety theorem without complementation, *Izvestiya VUZ Matematika* **39** (1995), 80–90. English version, *Russian Mathem. (Iz. VUZ)* **39** (1995), 74–63.

[12] M. P. Schützenberger, On finite monoids having only trivial subgroups, *Information and Control* **8** (1965), 190–194.

[13] M. Steinby, Syntactic algebras and varieties of recognizbale sets, in: *Proc. CAAP'79*, University of Lille (1979), 226–240.

[14] M. Steinby, A theory of tree language varieties, in: Nivat M. & Podelski A. (ed.) *Tree Automata and Languages*, Elsevier-Amsterdam (1992), 57–81.

[15] M. Steinby, General varieties of tree languages, *Theoretical Computer Science* **205** (1998), 1–43.

[16] D. Thérien, Recognizable languages and congruences, *Semigroup Forum* **23** (1981), 371–373.

Acta Cybernetica **17** (2006) 825–841.

Small Conjunctive Varieties of Regular Languages*

Libor Polák†

Abstract

The author's modification of Eilenberg theorem relates the so-called conjunctive varieties of regular languages with pseudovarieties of idempotent semirings. Recent results by Pastijn and his co-authors lead to the description of the lattice of all (pseudo)varieties of idempotent semirings with idempotent multiplication. We describe here the corresponding 78 varieties of languages.

Keywords: varieties of languages, pseudovarieties of idempotent semirings

1 Introduction

Certain significant classes of regular languages can by characterized by properties of syntactic semigroups/monoids of their members. The underlining framework is the so-called Eilenberg correspondence. The books by Pin [9] (see also [10]) and Almeida [1] present the background and numerous both simple and sophisticated examples. Varieties of languages corresponding to pseudovarieties of idempotent monoids are described by Neto and Sezinando in [6] and in their previous papers.

The author introduced syntactic semirings and proved an Eilenberg-type theorem in [11]. In Section 2 we reformulate the main result of Pastijn and his collaborators [7, 3, 8] giving a description of the lattice of all varieties of idempotent semirings with idempotent multiplication. We solve the identity problems in all those varieties, we show that all of them have a finite basis of identities. Further we recall one of the main results by Kuřil and author [4] relating the above varieties with certain operators on relatively free semigroups. In Section 3 we recall the author's modification of Eilenberg theorem, we find which classes of languages correspond to pseudovarieties of idempotent semirings in term of the closure operators mentioned above. Then we formulate it concretely for all 78 varieties. We complete this section by a relationship with the so-called shuffle closed languages. The last part of our contributions shows which of our varieties of languages are positive ones; we generate by each of our variety a positive one. We end with a simple example giving a language with idempotent syntactic semigroup having a syntactic semiring with a non-idempotent multiplication.

*Supported by the Ministry of Education of the Czech Republic under the project MSM 0021622409 and partially also by Project AKTION Österreich - Tschechische Republik

†Department of Mathematics, Masaryk University, Janáčkovo nám 2a, 662 95 Brno, Czech Republic, E-mail: polak@math.muni.cz, Url: http://www.math.muni.cz/~polak

2 Varieties of idempotent semirings

A *semigroup* is a non-empty set equipped with an associative operation. Let A^+ and $A^* = A^+ \cup \{1\}$ be the free semigroup and the free monoid, respectively, over a non-empty set A. An *ordered semigroup* is a triple $(S, \cdot\ \leq)$ where $(S, \cdot)$ is a semigroup and $\leq$ is a (partial) order on S such that

$$(\ \forall\ a, b, c \in S\ )(\ a \leq b \text{ implies both } ac \leq bc \text{ and } ca \leq cb\ )\ .$$

Homomorphisms of ordered semigroups are isotone semigroup homomorphisms. An *idempotent semiring* is a structure $(S, \cdot, \vee)$ where $(S, \cdot)$ is a semigroup, $(S, \vee)$ is a semilattice, and

$$(\ \forall\ a, b, c \in S\ )(\ a(b \vee c) = ab \vee ac \text{ and } (a \vee b)c = ac \vee bc\ )$$

(we do not postulate here the existence of the neutral element for the operation $\cdot$ nor for the operation $\vee$). Such a structure becomes an ordered semigroup with respect to the relation $\leq$ defined by

$$a \leq b \iff a \vee b = b,\ a, b \in S\ .$$

Let $A^\square$ denote the set of all non-empty finite subsets of A^+. Note that this set with the operations $U \cdot V = \{\, uv \mid u \in U,\ v \in V \,\}$ and the usual union forms a free idempotent semiring over the set A.

A class of semigroups is a *variety* if it is closed with respect to the forming of homomorphic images, substructures and products. A class of finite semigroups is a *pseudovariety* if it is closed with respect to the forming of homomorphic images, substructures and finite products. Similarly for ordered semigroups and idempotent semirings.

Let $X = \{x_1, x_2, \dots\}$ be the set of *variables* and let $X_n = \{x_1, \dots, x_n\}$ for $n \in \mathbb{N}$. For a variety $\mathcal{V}$ of semigroups we put

$$\rho_{\mathcal{V}} = \{\ (u, v) \in X^+ \times X^+ \mid \text{ all members of } \mathcal{V} \text{ satisfy the identity } u = v\ \}\ .$$

As well-known, the assignment $\mathcal{V} \mapsto \rho_{\mathcal{V}}$ is an isomorphism of the lattice of all varieties of semigroups onto the set $\mathsf{Fic}\, X^+$ of all the so-called fully invariant congruences on the semigroup X^+ ordered by the opposite inclusion. We put $\rho_{\mathcal{V},n} = \rho_{\mathcal{V}} \cap (X_n^+ \times X_n^+)$. Then $X^+/\rho_{\mathcal{V}}$ is a free semigroup in $\mathcal{V}$ over X and $X_n^+/\rho_{\mathcal{V},n}$ is a free semigroup in $\mathcal{V}$ over X_n, $n \in \mathbb{N}$. Similarly, for a variety of idempotent semirings $\mathcal{X}$, we put

$$\sigma_{\mathcal{X}} = \{\ (\{u_1, \dots, u_k\}, \{v_1, \dots, v_l\}\ ) \in X^\square \times X^\square \mid$$

$$\text{all members of } \mathcal{X} \text{ satisfy the identity } u_1 \vee \cdots \vee u_k = v_1 \vee \cdots \vee v_l\ \}\ .$$

Again, $\mathcal{X} \mapsto \sigma_{\mathcal{X}}$ is an isomorphism of the lattice of all varieties of idempotent semirings onto the set $\mathsf{Fic}\, X^\square$ of all fully invariant congruences on the semiring $X^\square$ ordered by the opposite inclusion. We put $\sigma_{\mathcal{X},n} = \sigma_{\mathcal{X}} \cap (X_n^\square \times X_n^\square)$. Then $X^\square/\sigma_{\mathcal{X}}$ is

a free idempotent semiring in $\mathcal{X}$ over X and $X_n^{\square}/\sigma_{\mathcal{X},n}$ is a free idempotent semiring in $\mathcal{X}$ over X_n, $n \in \mathbb{N}$.

First of all we have to recall basics on varieties of idempotent semigroups. The class of all semigroups satisfying a set Σ of identities is denoted by $\mathsf{Mod}\,\Sigma$. We denote :

- $\mathcal{T} = \mathsf{Mod}\,(x = y)$ – the class of all trivial semigroups,
- $\mathcal{LZ} = \mathsf{Mod}\,(xy = x)$ – the class of all semigroups of left zeros,
- $\mathcal{RZ} = \mathsf{Mod}\,(xy = y)$ – the class of all semigroups of right zeros,
- $\mathcal{Sl} = \mathsf{Mod}\,(x^2 = x,\ xy = yx)$ – the class of all semilattices,
- $\mathcal{LNB} = \mathsf{Mod}\,(x^2 = x,\ xyz = xzy)$ – the class of all left normal bands,
- $\mathcal{RNB} = \mathsf{Mod}\,(x^2 = x,\ xyz = yxz)$ – the class of all right normal bands,
- $\mathcal{ReB} = \mathsf{Mod}\,(x^2 = x,\ xyx = x)$ – the class of all rectangular bands,
- $\mathcal{LRB} = \mathsf{Mod}\,(x^2 = x,\ xy = xyx)$ – the class of all left regular bands,
- $\mathcal{RRB} = \mathsf{Mod}\,(x^2 = x,\ xy = yxy)$ – the class of all right regular bands,
- $\mathcal{NB} = \mathsf{Mod}\,(x^2 = x,\ xyzx = xzyx)$ – the class of all normal bands,
- $\mathcal{LQNB} = \mathsf{Mod}\,(x^2 = x,\ xyz = xyxz)$ – the class of all left quasinormal bands,
- $\mathcal{RQNB} = \mathsf{Mod}\,(x^2 = x,\ xyz = xzyz)$ – the class of all right quasinormal bands,
- $\mathcal{RB} = \mathsf{Mod}\,(x^2 = x,\ xyzx = xyxzx)$ – the class of all regular bands.

Note that the pairs $\mathcal{LZ}$ and $\mathcal{RZ}$, $\mathcal{LNB}$ and $\mathcal{RNB}$, $\mathcal{LRB}$ and $\mathcal{RRB}$, $\mathcal{LQNB}$ and $\mathcal{RQNB}$ consist of pairwise dual semigroups.

We need to introduce several operators on words from X^* :

- $\mathsf{c}(u)$ is the set of all variables in u,
- $\mathsf{h}(u)$ is the first variable of $u \in X^+$, $\mathsf{h}(1) = 1$,
- $\mathsf{t}(u)$ is the last variable of $u \in X^+$, $\mathsf{t}(1) = 1$ (it is dual to h),
- $\mathsf{l}(u)$ is the word resulting from $u \in X^+$ leaving only the first occurrence of each variable from the left, $\mathsf{l}(1) = 1$,
- $\mathsf{r}(u)$ is the word resulting from $u \in X^+$ leaving only the first occurrence of each variable from the right, $\mathsf{r}(1) = 1$ (it is dual to l),

- u_Y, for $Y \subseteq X$, is the word resulting from u by substituting 1 for each occurrence of each variable from Y.

Next we formulate how to solve the identity problem (i.e., to describe the congruences $\rho_{\mathcal{X}}$) in varieties mentioned above.

Result 1 (see for instance [13]). *The lattice of all varieties of regular bands consists of 13 varieties introduced above; the order by the inclusion is given by the diagram below.*

Further, for $u, v \in X^+$ we have

(i) $u \, \rho_{\mathcal{T}} \, v$ *for all* u, v; *thus* $X^+/\rho_{\mathcal{T}} \cong (\{x_1\}, \circ)$,

(ii) $u \, \rho_{\mathcal{LZ}} \, v$ *iff* $\mathsf{h}(u) = \mathsf{h}(v)$; *thus* $X^+/\rho_{\mathcal{LZ}} \cong (X, \circ)$ *where* $x \circ y = x$,

(iii) $u \, \rho_{\mathcal{SL}} \, v$ *iff* $\mathsf{c}(u) = \mathsf{c}(v)$; *thus* $X^+/\rho_{\mathcal{SL}} \cong$

$$(\{ Y \mid Y \text{ is a non-empty finite subset of } X \}, \circ) \text{ where } Y \circ Z = Y \cup Z \ ,$$

(iv) $u \, \rho_{\mathcal{LNB}} \, v$ *iff* $\mathsf{c}(u) = \mathsf{c}(v)$, $\mathsf{h}(u) = \mathsf{h}(v)$; *thus* $X^+/\rho_{\mathcal{LNB}} \cong$

$$(\{ (y, Y) \mid Y \text{ is a non-empty finite subset of } X,\ y \in Y \}, \circ)$$

$$\text{where } (y, Y) \circ (z, Z) = (y, Y \cup Z),$$

(v) $u \, \rho_{\mathcal{ReB}} \, v$ *iff* $\mathsf{h}(u) = \mathsf{h}(v)$, $\mathsf{t}(u) = \mathsf{t}(v)$; *thus* $X^+/\rho_{\mathcal{ReB}} \cong$

$$(X \times X, \circ) \text{ where } (x, y) \circ (z, t) = (x, t) \ ,$$

(vi) $u \, \rho_{\mathcal{LRB}} \, v$ *iff* $\mathsf{l}(u) = \mathsf{l}(v)$; *thus* $X^+/\rho_{\mathcal{LRB}} \cong$

$$(\{ u \in X^+ \mid u \text{ has pairwise different variables} \}, \circ) \text{ where } u \circ v = \mathsf{l}(uv) \ ,$$

(vii) $u \, \rho_{\mathcal{NB}} \, v$ *iff* $\mathsf{c}(u) = \mathsf{c}(v)$, $\mathsf{h}(u) = \mathsf{h}(v)$, $\mathsf{t}(u) = \mathsf{t}(v)$; *thus* $X^+/\rho_{\mathcal{NB}} \cong$

$$(\{ (x, Y, y) \mid Y \text{ is a non-empty finite subset of } X,\ x, y \in Y \}, \circ)$$

$$\text{where } (x, Y, y) \circ (z, Z, t) = (x, Y \cup Z, t) \ ,$$

(viii) $u \, \rho_{\mathcal{LQNB}} \, v$ *iff* $\mathsf{l}(u) = \mathsf{l}(v)$, $\mathsf{t}(u) = \mathsf{t}(v)$; *thus* $X^+/\rho_{\mathcal{LQNB}} \cong$

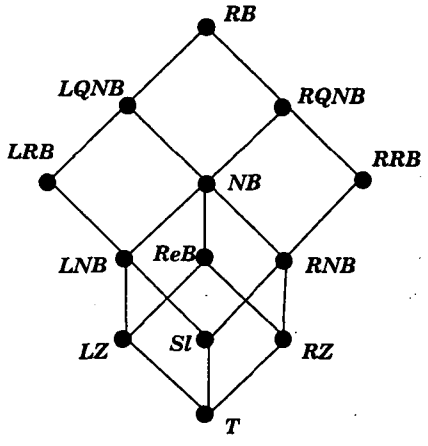
$$(\{ (u, y) \in X^+ \mid u \text{ has pairwise different variables},\ y \in \mathsf{c}(u) \}, \circ)$$

$$\text{where } (u, y) \circ (v, z) = (\mathsf{l}(uv), z) \ ,$$

(ix) $u \, \rho_{\mathcal{RB}} \, v$ *iff* $\mathsf{l}(u) = \mathsf{l}(v)$, $\mathsf{r}(u) = \mathsf{r}(v)$; *thus* $X^+/\rho_{\mathcal{RB}} \cong$

$$(\{ (u, v) \in X^+ \times X^+ \mid \mathsf{c}(u) = \mathsf{c}(v), \text{ each of } u, v \text{ has pairwise diff. variables} \}, \circ)$$

$$\text{where } (u, v) \circ (u', v') = (\mathsf{l}(uu'), \mathsf{r}(vv')) \ .$$



Now we introduce several important finite idempotent semirings :

- L is the left zero semigroup with elements a and b ordered by $a < b$,
- R is the right zero semigroup with elements a and b ordered by $a < b$,
- D is the distributive lattice with elements a and b ordered by $a < b$ (multiplication is the meet),
- M has the elements a, b and both operations equal to the join with respect to the order $a < b$,
- B is the left zero semigroup with elements a and b and with an extra neutral element 1 ordered by $a < 1 < b$,
- C is the right zero semigroup with elements a and b and with an extra neutral element 1 ordered by $a < 1 < b$.

For any idempotent semiring S, we denote by S^0 the semiring obtained from S by adding an extra element 0 and where $0 \cdot a = a \cdot 0 = 0$, $0 \vee a = a \vee 0 = a$, for every $a \in S$.

Result 2 ([4] **Thm.2.9,** [7] **Thm.2.3**). *Each idempotent semiring with an idempotent multiplication satisfies the identity $xyxzx = xyzx$; that is, its multiplicative reduct is a regular band.*

The following varieties play here a crucial role :

- $\mathcal{TS}$ – the class of all trivial (i.e., one element) semirings,
- $\mathcal{L} = < L >$ – the class of all idempotent semirings whose multiplicative reducts are left zero semigroups,
- $\mathcal{R} = < R >$ – the class of all idempotent semirings whose multiplicative reducts are right zero semigroups,

- $\mathcal{D} = < D >$ – the class of all distributive lattices,
- $\mathcal{M} = < M >$ – the class of all monobisemilattices,
- $\mathcal{S} = < M^0 >$ – the class of all bisemilattices,
- $\mathcal{B} = < B >$, $\mathcal{C} = < C >$, $\mathcal{L}^0 = < L^0 >$, $\mathcal{R}^0 = < R^0 >$, $\mathcal{B}^0 = < B^0 >$, $\mathcal{C}^0 = < C^0 >$,
- $\mathcal{I}$ – the class of all idempotent semirings whose multiplicative reducts are idempotent.

Notice that the pairs $\mathcal{L}$ and $\mathcal{R}$, $\mathcal{B}$ and $\mathcal{C}$, $\mathcal{B}^0$ and $\mathcal{C}^0$ consist of pairwise dual semirings. Next we will solve the identity problems (i.e., to describe the congruences $\sigma_{\mathcal{X}}$) for the varieties $\mathcal{L}$, $\mathcal{D}$, $\mathcal{M}$, $\mathcal{B}$, $\mathcal{L}^0$, $\mathcal{S}$ and $\mathcal{B}^0$. These results can be extracted from [5, 7, 3, 8]. We present here simple and transparent proofs. Notice that each set of identities is equivalent to the inequalities of the form

$$u \leq u_1 \vee \cdots \vee u_k \ . \tag{$*$}$$

Result 3.

(i) L *satisfies* $(*)$ *iff* $\mathsf{h}(u) \in \{ \mathsf{h}(u_1), \ldots, \mathsf{h}(u_k) \}$,

(ii) D *satisfies* $(*)$ *iff there exists* $i \in \{1, \ldots, k\}$ *such that* $\mathsf{c}(u) \supseteq \mathsf{c}(u_i)$,

(iii) M *satisfies* $(*)$ *iff* $\mathsf{c}(u) \subseteq \mathsf{c}(u_1) \cup \cdots \cup \mathsf{c}(u_k)$,

(iv) B *satisfies* $(*)$ *iff for each* $Y \subseteq X$ $\mathsf{h}(u_Y) \in \{ \mathsf{h}((u_1)_Y), \ldots, \mathsf{h}((u_k)_Y) \}$,

(v) S^0 *satisfies* $(*)$ *iff* S *satisfies* $u \leq \bigvee \{ u_i \mid i \in \{1, \ldots, k\},\ \mathsf{c}(u) \supseteq \mathsf{c}(u_i) \}$.

Proof. (i) L does not satisfy $(*)$ iff we can find a substitution $\xi : X \to L$ such that $\xi(\mathsf{h}(u))$ is b and of $\xi(\mathsf{h}(u_1)) = \cdots = \xi(\mathsf{h}(u_k)) = a$.

(ii) D does not satisfy $(*)$ iff we can find a substitution $\xi : X \to D$ such that $\xi(u) = b$ and $\xi(u_1) = \cdots = \xi(u_k) = a$. This is equivalent to

$$\forall\, i \in \{1, \ldots, k\}\ \exists\, x \in \mathsf{c}(u_i) \setminus \mathsf{c}(u) \ .$$

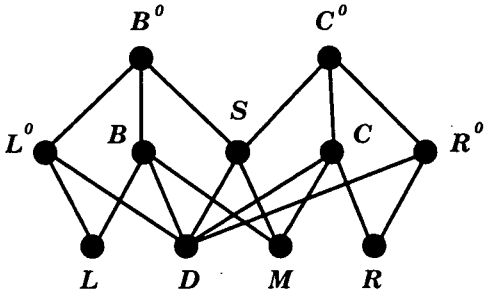
(iii) M does not satisfy $(*)$ iff we can find a substitution $\xi : X \to M$ such that $\xi(u) = b$ and $\xi(u_1) = \cdots = \xi(u_k) = a$. This is equivalent to the existence of $x \in \mathsf{c}(u) \setminus \{ \mathsf{c}(u_i) \cup \cdots \cup \mathsf{c}(u_k) \}$.

(iv) $(*)$ is valid in B for all substitutions $X \to B$ where exactly all variables from Y go to 1 iff L satisfies $u_Y \leq (u_1)_Y \vee \cdots \vee (u_k)_Y$.

(v) If we substitute 0 for a variable from $\mathsf{c}(u)$ the inequality $(*)$ holds trivially. So substitute for all of them elements from S. The worst case is to substitute for all other variables the element 0. □

We denote by $\mathscr{V}(\mathcal{X})$ the lattice of all subvarieties of a variety $\mathcal{X}$. By McKenzie and Romanowska [5], all non-trivial varieties of idempotent semirings with commutative and idempotent multiplication are exactly : $\mathcal{D}$, $\mathcal{M}$, $\mathcal{D} \vee \mathcal{M}$ and $\mathcal{S}$. Later Ghosh, Pastijn and Zhao in [3] found a description of the lattice of all varieties of idempotent semirings whose multiplicative reducts are normal bands (35 varieties). They use combination of semantical methods (congruences, Green relations,...) with syntactical ones (calculating with identities,...). The result was previously announced by the authors of [4] : they used purely syntactical approach (operators on relatively free semigroups) - see Result 5. In [8] Pastijn accomplished the task of the description of the lattice $\mathscr{V}(\mathcal{I})$. Up to now we are not able to get it purely syntactically. We formulate this deep result next in a modified form. Recall that a subset B of an ordered set $(A, \leq)$ is *hereditary* if $b \in B$, $a \in A$, $a \leq b$ implies $a \in B$.

Result 4 (extracted from [7, 3, 8]). *The lattice of all varieties of idempotent semirings with an idempotent multiplication is distributive. Its non-trivial join-irreducible elements are exactly the eleven varieties mentioned above. They form the partially ordered set $(\mathfrak{O}, \leq)$ depicted below. Consequently, the varieties of idempotent semirings with an idempotent multiplication correspond to the 78 hereditary subsets of $(\mathfrak{O}, \leq)$; more precisely, they are exactly joins of hereditary sets and joins of different hereditary sets are different.*

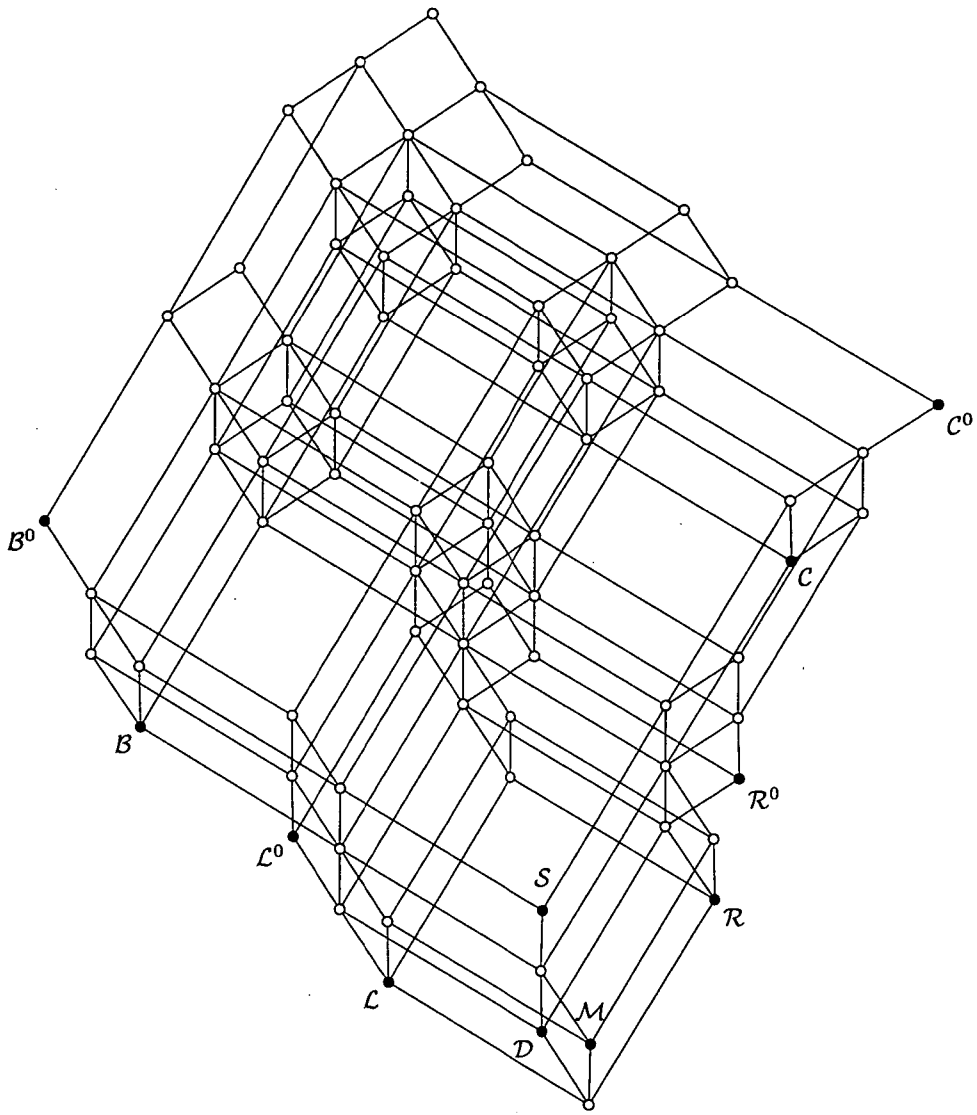


Proof. By Theorem 3.4. of [3], the kernel of the mapping

$$\phi : \mathscr{V}(\mathcal{I}) \to \mathscr{V}(\mathcal{S}),\ \mathcal{X} \mapsto \mathcal{X} \cap \mathcal{S}$$

decomposes $\mathscr{V}(\mathcal{I})$ into five intervals with the lower ends $\mathcal{TS}$, $\mathcal{D}$, $\mathcal{M}$, $\mathcal{D} \vee \mathcal{M}$, $\mathcal{S}$, respectively. By Result 4.1, the first interval is $[\mathcal{TS}, \mathcal{L} \vee \mathcal{R}]$ and it consists exactly of $\mathcal{TS}$, $\mathcal{L}$, $\mathcal{R}$ and $\mathcal{L} \vee \mathcal{R}$. Since $\mathcal{R} \not\subseteq \mathcal{B}^0$ by Result 3, this interval intersects $\mathscr{L}(\mathcal{B}^0)$ in $\{\mathcal{TS}, \mathcal{L}\}$. Similarly, by Theorems 4.5. and 4.7., the second and the third interval intersect $\mathscr{L}(\mathcal{B}^0)$ in $\{\mathcal{D}, \mathcal{D} \vee \mathcal{L}, \mathcal{L}^0\}$ and $\{\mathcal{M}, \mathcal{M} \vee \mathcal{L}\}$, respectively.

Further by Corollary 2.5 and 3.4. of [8], the fourth and the fifth interval intersect $\mathscr{L}(\mathcal{B}^0)$ in $\{\mathcal{D} \vee \mathcal{M}, \mathcal{D} \vee \mathcal{M} \vee \mathcal{L}, \mathcal{D} \vee \mathcal{M} \vee \mathcal{L}^0, \mathcal{B}, \mathcal{B} \vee \mathcal{L}^0\}$ and $\{\mathcal{S}, \mathcal{S} \vee \mathcal{L}, \mathcal{S} \vee \mathcal{L}^0, \mathcal{S} \vee \mathcal{B}, \mathcal{B}^0\}$, respectively.



Finally, by Theorem 4.1, the mapping

$$\psi : \{\ (\mathcal{X},\mathcal{Y}) \in [\mathcal{TS},\mathcal{B}^0] \times [\mathcal{TS},\mathcal{C}^0] \mid \mathcal{X} \cap \mathcal{S} = \mathcal{Y} \cap \mathcal{S}\ \} \to \mathscr{L}(\mathcal{I}),\ (\mathcal{X},\mathcal{Y}) \mapsto \mathcal{X} \vee \mathcal{Y}$$

is a bijection. □

One of the main results of [4] is recalled below. For our purposes it is not necessary to put here the 10 axioms defining the so-called $\mathcal{V}$-admissible closure operators from subsets of $X^+/\rho_{\mathcal{V}}$ to subsets of $X^+/\rho_{\mathcal{V}}$. Notice only that one of the

axioms is to be of finite character (all closures are determined by closures of finite sets).

Result 5 ([4], Theorem 4.7). *Varieties of idempotent semirings corres-pond to the pairs $(\mathcal{V}, [\ ])$ where $\mathcal{V}$ is a variety of semigroups and $[\ ]$ is a $\mathcal{V}$-admissible closure operator on $X^+/\rho_{\mathcal{V}}$; we write $\mathcal{X} \mapsto (\underline{\mathcal{X}}, [\ ]_{\mathcal{X}})$. Moreover,*

$$\{u_1, \ldots, u_k\}\ \sigma_{\mathcal{X}}\ \{v_1, \ldots, v_l\} \text{ iff}$$

$$[\,\{u_1\rho_{\underline{\mathcal{X}}}, \ldots, u_k\rho_{\underline{\mathcal{X}}}\}\,]_{\mathcal{X}} = [\,\{v_1\rho_{\underline{\mathcal{X}}}, \ldots, v_l\rho_{\underline{\mathcal{X}}}\}\,]_{\mathcal{X}}\ .$$

Conversely, given a variety $\mathcal{X}$, we get $\underline{\mathcal{X}}$ and $[\ ]_{\mathcal{X}}$ by

$$u\ \rho_{\underline{\mathcal{X}}}\ v \text{ if and only if } \{u\}\ \sigma_{\mathcal{X}}\ \{v\}, \text{ and}$$

$$u\rho_{\underline{\mathcal{X}}} \in [\,(u_1)\rho_{\underline{\mathcal{X}}}, \ldots, (u_k)\rho_{\underline{\mathcal{X}}}\,] \text{ iff } \{u_1, \ldots, u_k, u\}\ \sigma_{\mathcal{X}}\ \{u_1, \ldots, u_k\}\ ,$$

which is also equivalent to the fact that $\mathcal{X}$ satisfies $u \leq u_1 \vee \cdots \vee u_k$.

The situation above leads also to a closure operator on subsets of $X_n^+/\rho_{\mathcal{V},n}$ defined by

$$[\,u_1\rho_{\mathcal{V},n}, \ldots, u_k\rho_{\mathcal{V},n}\,]_{\mathcal{X},n} = [\,u_1\rho_{\mathcal{V}}, \ldots, u_k\rho_{\mathcal{V}}\,]_{\mathcal{X}} \cap (X_n^+/\rho_{\mathcal{V},n} \times X_n^+/\rho_{\mathcal{V},n})\ .$$

It follows how to get $(\underline{\mathcal{X}}, [\ ]_{\mathcal{X}})$ for all varieties knowing the data for the join-irreducible ones.

Result 6 ([4], Theorem 4.9). *Let $\mathcal{X}$, $\mathcal{Y}$ be varieties of idempotent semirings with idempotent multiplication. Then $\underline{\mathcal{X} \vee \mathcal{Y}} = \underline{\mathcal{X}} \vee \underline{\mathcal{Y}}$ and*

$$u(\rho_{\underline{\mathcal{X}}} \cap \rho_{\underline{\mathcal{Y}}}) \in [\,\{u_1(\rho_{\underline{\mathcal{X}}} \cap \rho_{\underline{\mathcal{Y}}}), \ldots, u_k(\rho_{\underline{\mathcal{X}}} \cap \rho_{\underline{\mathcal{Y}}})\}\,]_{\mathcal{X}\vee\mathcal{Y}} \text{ if and only if}$$

$$u\rho_{\underline{\mathcal{X}}} \in [\,u_1\rho_{\underline{\mathcal{X}}}, \ldots, u_k\rho_{\underline{\mathcal{X}}}\,]_{\mathcal{X}} \text{ and } u\rho_{\underline{\mathcal{Y}}} \in [\,u_1\rho_{\underline{\mathcal{Y}}}, \ldots, u_k\rho_{\underline{\mathcal{Y}}}\,]_{\mathcal{Y}}\ .$$

We can extract from [3, 8] that each variety from $\mathscr{V}(\mathcal{I})$ is finitely based. We formulate and prove it now in a transparent way.

Theorem 7. *For each variety $\mathcal{X} \in \mathfrak{D}$ there exists a variety $\mathcal{X}^{\mathsf{c}} \in \mathscr{V}(\mathcal{I})$ such that*

$$(\ \forall\ \mathcal{Y} \in \mathscr{V}(\mathcal{I})\ )\ \mathcal{X} \not\subseteq \mathcal{Y} \text{ if and only if } \mathcal{Y} \subseteq \mathcal{X}^{\mathsf{c}}\ .$$

More precisely,

- $\mathcal{L}^{\mathsf{c}} = \mathsf{Mod}\,(xy \leq y \vee yx)$,
- $\mathcal{D}^{\mathsf{c}} = \mathsf{Mod}\,(x \leq xyx)$,
- $\mathcal{M}^{\mathsf{c}} = \mathsf{Mod}\,(xyz \leq x \vee z)$,
- $\mathcal{S}^{\mathsf{c}} = \mathsf{Mod}\,(xyx \leq x \vee yzy)$,
- $\mathcal{B}^{\mathsf{c}} = \mathsf{Mod}\,(xyz \leq x \vee zyz)$,

- $(\mathcal{L}^0)^c = \mathsf{Mod}\,(xy \leq xz \vee yxy)$,
- $(\mathcal{B}^0)^c = \mathsf{Mod}\,(xyz \leq xz \vee zyz \vee xyt)$.

Consequently,

$$(\,\forall\, \mathcal{Y} \in \mathcal{V}(\mathcal{I})\,)\ \mathcal{Y} = \bigcap \{\, \mathcal{X}^c \mid \mathcal{X} \in \mathfrak{D},\ \mathcal{X} \not\subseteq \mathcal{Y} \,\}\,.$$

It follows that each $\mathcal{Y} \in \mathcal{V}(\mathcal{I})$ is finitely based.

Proof. We see that

- L does not satisfy $xy \leq y \vee yx$ but C^0 does,
- D does not satisfy $x \leq xyx$ but $L,\ M,\ R$ do,
- M does not satisfy $xyz \leq x \vee z$ but $L^0,\ R^0$ do,
- S does not satisfy $xyx \leq x \vee yzy$ but $L^0,\ B,\ C,\ R^0$ do,
- B does not satisfy $xyz \leq x \vee zyz$ but $L^0,\ C^0$ does,
- L^0 does not satisfy $xy \leq xz \vee yxy$ but $B,\ C^0$ do,
- B^0 does not satisfy $xyz \leq xz \vee zyz \vee xyt$ but $L^0,\ B,\ C^0$ do.

Let $\mathcal{Y} \in \mathcal{V}(\mathcal{I})$, $\{\mathcal{X}_1, \ldots, \mathcal{X}_k\} = \{\,\mathcal{X} \in \mathfrak{D} \mid \mathcal{X} \not\subseteq \mathcal{Y}\,\}$ and $\{\mathcal{X}_{k+1}, \ldots, \mathcal{X}_{11}\} = \{\,\mathcal{X} \in \mathfrak{D} \mid \mathcal{X} \subseteq \mathcal{Y}\,\}$. Then $\mathcal{X}_{k+1} \vee \cdots \vee \mathcal{X}_{11} = \mathcal{Y} \subseteq \mathcal{X}_1^c \cap \cdots \cap \mathcal{X}_k^c$. The last inclusion is, in fact, an equality since there is no $\mathcal{X}_i$, $i \in \{1, \ldots, k\}$, such that $\mathcal{X}_i \subseteq \mathcal{X}_1^c \cap \cdots \cap \mathcal{X}_k^c$. □

3 Varieties of languages

A language $L \subseteq A^+$ defines the *syntactic congruence* $\sim_L$ on $(A^\square, \cdot, \cup)$ by

$$\{u_1, \ldots, u_k\} \sim_L \{v_1, \ldots, v_l\} \text{ if and only if}$$

$$(\,\forall\, p, q \in A^*\,)\ (\,pu_1q, \ldots, pu_kq \in L \iff pv_1q, \ldots, pv_lq \in L\,)\,.$$

The factor-structure $(A^\square, \cdot, \cup)/\!\sim_L$ is called the *syntactic semiring* of L; we denote it by $(\mathsf{S}(L), \cdot, \vee)$. This structure is finite if and only if the language L is regular.

For non-empty finite sets A and B, a semiring homomorphism

$$f : (B^\square, \cdot, \cup) \to (A^\square, \cdot, \cup)$$

and $K \subseteq A^+$, we define

$$f^{[-1]}(K) = \{\, v \in B^+ \mid f(\{v\}) \subseteq K \,\}\,.$$

Similarly, for a semigroup homomorphism $g : (B^+, \cdot) \to (A^+, \cdot)$ and $K \subseteq A^+$ we put

$$g^{(-1)}(K) = \{ v \in B^+ \mid g(v) \in K \} .$$

A *class* of (regular) languages is an operator $\mathscr{L}$ assigning to every non-empty finite set A a set $\mathscr{L}(A)$ of regular languages over the alphabet A. Such a class is a *conjunctive variety* if

(i) each $\mathscr{L}(A)$ contains both $\emptyset$ and A^+,

(ii) each $\mathscr{L}(A)$ is closed with respect to finite intersections and quotients, and

(iii) for each finite sets A and B and a semiring homomorphism $f : B^\square \to A^\square$, $K \in \mathscr{L}(A)$ implies $f^{[-1]}(K) \in \mathscr{L}(B)$.

Similarly, it is a *positive* variety if (i) holds,
(ii') each $\mathscr{L}(A)$ is closed with respect to finite intersections, finite unions and quotients, and
(iii') for each finite sets A and B and a semigroup homomorphism $g : B^+ \to A^+$, $K \in \mathscr{L}(A)$ implies $f^{(-1)}(K) \in \mathscr{L}(B)$.

Adding to (ii') the closeness with respect to complements, we get the notion of a *boolean* variety.

We can assign to any class of languages $\mathscr{L}$ the pseudovariety

$$\mathsf{S}(\mathscr{L}) = [\ \{\ (\mathsf{S}(L), \cdot, \vee) \mid A \text{ is a non-empty finite set, } L \in \mathscr{L}(A)\ \}\ ]$$

of idempotent semirings generated by all syntactic semirings of members of all $\mathscr{L}(A)$'s. Conversely, for a class $\mathcal{X}$ of idempotent semirings and a non-empty finite set A, we put

$$(\mathsf{L}(\mathcal{X}))(A) = \{ L \subseteq A^* \mid (\mathsf{S}(L), \cdot, \vee) \in \mathcal{X} \} .$$

Result 8 ([11], Theorem 14). *The assignments $\mathscr{L} \mapsto \mathsf{S}(\mathscr{L})$ and $\mathcal{X} \mapsto \mathsf{L}(\mathcal{X})$ are mutually inverse bijections between the conjunctive varieties of languages and pseudovarieties of finite idempotent semirings.*

Similarly, by the classical Eilenberg theorem, the boolean varieties of languages correspond to pseudovarieties of semigroups, and the positive varieties of languages correspond to pseudovarieties of ordered semigroups (see Pin [10]).

Theorem 9. *The pseudovarieties of finite idempotent semirings with idempotent multiplication are exactly the classes $\mathsf{Fin}\,\mathcal{X}$ consisting of finite members of a variety $\mathcal{X}$ of idempotent semirings with idempotent multiplication. Finite members of different varieties form different pseudovarieties.*

Proof. Since the lattice $\mathscr{V}(\mathcal{I})$ is finite, the pseudovarieties are exactly of the form $\mathsf{Fin}\,\mathcal{X}$, $\mathcal{X} \in \mathscr{V}(\mathcal{I})$ (see [1], Proposition 3.2.4). Since all varieties are generated by finite members, the mapping Fin is injective. □

Our key result follows.

Theorem 10. *Let $\mathcal{X}$ be a variety of idempotent semirings with idempotent multiplication. For each $n \in \mathbb{N}$, the set $(\mathsf{L}(\mathsf{Fin}\,\mathcal{X}))(X_n)$ consists exactly of unions of $[\ ]_{\mathcal{X},n}$-closed subsets of $X_n^+/\rho_{\underline{\mathcal{X}},n}$.*

Proof. Let $L \subseteq X_n^+$, $u, u_1, \ldots, u_k \in X_m^+$. We show that

$$(X_n^{\square}, \cdot, \cup)/\sim_L \text{ satisfies } u \le u_1 \vee \cdots \vee u_k \qquad (\dagger)$$

if and only if ($\forall\, w_1, \ldots, w_m \in X_n^+$, $p, q \in X_n^*$)

$$(\, pu_1(w_1, \ldots, w_m)q, \ldots, pu_k(w_1, \ldots, w_m)q \in L \;\Rightarrow\; pu(w_1, \ldots, w_m)q \in L\,)\,. \qquad (\ddagger)$$

Indeed, (†) means that ($\forall\, w_1, \ldots, w_m \in X_n^+$)

$$(\, u(w_1 \sim_L, \ldots, w_k \sim_L) \le u_1(w_1 \sim_L, \ldots, w_k \sim_L) \vee \cdots \vee u_k(w_1 \sim_L, \ldots, w_k \sim_L)\,,$$

which is equivalent to ($\forall\, w_1, \ldots, w_m \in X_n^+$)

$$u(w_1, \ldots, w_k) \sim_L \le (\, u_1(w_1, \ldots, w_k) \vee \cdots \vee u_k(w_1, \ldots, w_k)\,) \sim_L\,,$$

and this is equivalent to (‡).

Now let $u_1, \ldots, u_k \in L$, $u \in X_n^+$ and let $\mathcal{X}$ satisfy $u \le u_1 \vee \ldots u_k$. Put $p = q = 1$ and $w_1 = x_1, \ldots, w_n = x_n$ (we have $m = n$) in (‡).

Conversely, let L be a union of $[\ ]_{\mathcal{X}}$-closed set of classes. Let $u, u_1, \ldots, u_k \in X_m^+$, $u \le u_1 \vee \cdots \vee u_k$ in $\mathcal{X}$. Let $w_1, \ldots, w_m \in X_n^+$, $p, q \in X_n^*$. Then also $pu(w_1, \ldots, w_m)q \le pu_1(w_1, \ldots, w_m)q \vee \cdots \vee pu_k(w_1, \ldots, w_m)q$ in $\mathcal{X}$, which gives (‡) and therefore also (†). Moreover, L is a regular language since it is recognized by a finite semigroup $X_n^+/\rho_{\underline{\mathcal{X}},n}$. □

Similar results for semigroups and ordered semigroups are almost obvious. Recall that the operator L there uses syntactic semigroups and ordered semigroups instead of semirings; we write $\mathsf{L}^{\cdot}$ and $\mathsf{L}^{\le}$. In the ordered case $\rho_{\mathcal{V}}$ consists of all pairs (u, v) such that $u \le v$ in $\mathcal{V}$. Let $\tau_{\mathcal{V}} = \rho_{\mathcal{V}} \cap (\rho_{\mathcal{V}})^{-1}$. Then $X_n^+/\tau_{\mathcal{V}}$ is ordered by $u\tau_{\mathcal{V}} \le v\tau_{\mathcal{V}}$ iff $u\ \rho_{\mathcal{V}}\ v$.

Result 11.

(i) Let $\mathcal{V}$ be a variety of semigroups. For each $n \in \mathbb{N}$, the set $(\mathsf{L}^{\cdot}(\mathsf{Fin}\,\mathcal{V}))(X_n)$ consists exactly of unions of classes of $X_n^+/\rho_{\mathcal{V},n}$ which are regular languages.

(ii) Let $\mathcal{V}$ be a variety of ordered semigroups. For each $n \in \mathbb{N}$, the set $(\mathsf{L}^{\le}(\mathsf{Fin}\,\mathcal{V}))(X_n)$ consists exactly of unions of hereditary sets of classes of $(X_n^+/\tau_{\mathcal{V},n}, \le)$ which are regular languages.

Now we describe conjunctive varieties of languages corresponding to our pseudovarieties.

Theorem 12.

(i) $\underline{\mathcal{D}} = \mathcal{S}l$,

$$\text{and } Y \in [\{Y_1, \ldots, Y_k\}]_{\mathcal{D}} \text{ iff } (\ \exists\, i \in \{1, \ldots, k\}\ )\ Y \supseteq Y_i\ .$$

Consequently,

$$L \in (\mathsf{L}(\mathcal{D}))(X_n) \text{ iff } (\ \exists\, Y_1, \ldots, Y_k \subseteq X_n\ )$$

$$\text{such that } L = \{\ u \in X_n^+ \mid (\ \exists\, i \in \{1, \ldots, k\}\ )\ \mathsf{c}(u) \supseteq Y_i\ \}\ ,$$

or equivalently L *is a union of languages of the form*

$$\{\, u \in X_n^+ \mid \mathsf{c}(u) \supseteq Y \,\},\ Y \subseteq X\ .$$

(ii) $\underline{\mathcal{M}} = \mathcal{S}l$,

$$\text{and } Y \in [\{Y_1, \ldots, Y_k\}]_{\mathcal{D}} \text{ iff } Y \subseteq Y_1 \cup \cdots \cup Y_k\ .$$

Consequently,

$$L \in (\mathsf{L}(\mathcal{M}))(X_n) \text{ iff } (\ \exists\, Y \subseteq X_n\ )\ L = Y^+\ .$$

(iii) $\underline{\mathcal{S}} = \mathcal{S}l$,

$$\text{and } Y \in [\{Y_1, \ldots, Y_k\}]_{\mathcal{S}} \text{ iff } (\ \exists\, i_1, \ldots, i_l \in \{1, \ldots, k\}\ )\ Y = Y_{i_1} \cup \cdots \cup Y_{i_l}\ .$$

Consequently,

$$L \in (\mathsf{L}(\mathcal{S}))(X_n) \text{ iff } (\ \exists\, Y_1, \ldots, Y_k \subseteq X_n\ )$$

$$\text{with } L = \{\ u \in X_n^+ \mid (\ \exists\, i_1, \ldots, i_l \in \{1, \ldots, k\}\ )\ \mathsf{c}(u) = Y_{i_1} \cup \cdots \cup Y_{i_l}\ \}\ .$$

(iv) $\underline{\mathcal{L}} = \mathcal{LZ}$, *and* $[\{y_1, \ldots, y_k\}]_{\mathcal{L}} = \{y_1, \ldots, y_k\}$. *Consequently,*

$$L \in (\mathsf{L}(\mathcal{L}))(X_n) \text{ iff } (\ \exists\, Y \subseteq X_n\ )\ L = \{\, u \in X_n^+ \mid \mathsf{h}(u) \in Y \,\} = YX_n^*\ .$$

(v) $\underline{\mathcal{L}^0} = \mathcal{LNB}$, *and*

$$(y, Y) \in [\{\, (y_1, Y_1), \ldots, (y_k, Y_k)\,\}]_{\mathcal{L}^0}$$

$$\text{iff } (\ \exists\, i \in \{1, \ldots, k\}\ )\ (\ y = y_i,\ Y \supseteq Y_i\ )\ .$$

Consequently,

$$L \in (\mathsf{L}(\mathcal{L}^0))(X_n) \text{ iff } (\ \exists\, Y_1, \ldots, Y_k \subseteq X_n, y_1 \in Y_1, \ldots, y_k \in Y_k\ )$$

$$\text{such that } L = \{\ u \in X_n^+ \mid (\ \exists\, i \in \{1, \ldots, k\}\ )\ \mathsf{h}(u) = y_i \text{ and } \mathsf{c}(u) \supseteq Y_i\ \}\ ,$$

or equivalently L *is a union of languages of the form*

$$\{\, u \in X_n^+ \mid \mathsf{h}(u) = y,\ \mathsf{c}(u) \supseteq Y \,\},\ y \in Y \subseteq X\ .$$

(vi) $\underline{\mathcal{B}} = \mathcal{LRB}$, *and*

$$u \in [\{u_1, \dots, u_k\}]_{\mathcal{B}} \text{ iff } (\forall\, Y \subseteq X\ )\ \mathsf{h}(u_Y) \in \{\, \mathsf{h}((u_1)_Y), \dots, \mathsf{h}((u_k)_Y)\,\}\ .$$

Consequently,

$$L \in (\mathsf{L}(\mathcal{B}))(X_n) \text{ iff } (\exists\, u_1, \dots, u_k \in X_n^+\ ) \text{ such that}$$

$$L = \{\ u \in X_n^+ \mid (\forall\, Y \subseteq X_n\ )\ \mathsf{h}(u_Y) \in \{\, \mathsf{h}((u_1)_Y), \dots, \mathsf{h}((u_k)_Y)\,\}\ \}\ .$$

(vii) $\underline{\mathcal{B}^0} = \mathcal{LRB}$,

$$\text{and } u \in [\{u_1, \dots, u_k\}]_{\mathcal{B}^0} \text{ iff } u \in [\{\, u_i \mid i = 1, \dots, k,\ \mathsf{c}(u_i) \subseteq \mathsf{c}(u)\,\}]_{\mathcal{B}}$$

Consequently,

$$L \in (\mathsf{L}(\mathcal{B}^0))(X_n) \text{ iff } (\exists\, u_1, \dots, u_k \in X_n^+\ ) \text{ such that } L =$$

$$\{\ u \in X_n^+ \mid (\forall\, Y \subseteq X_n\ )\ \mathsf{h}(u_Y) \in \{\, \mathsf{h}((u_i)_Y) \mid i = 1, \dots, k,\ \mathsf{c}(u_i) \subseteq \mathsf{c}(u)\,\}\ .$$

Proof. We find the values of $\underline{\mathcal{X}}$ first. It would follow from the observations below. We will use Result 3.

- The identity $x^2 = x$ holds in B^0; by duality also in C^0 and therefore in all eleven varieties from $\mathfrak{D}$.
- The identity $xy = yx$ holds in M^0 but not in L.
- The identity $xy = x$ holds in L but not in D, M, R.
- The identity $xyz = xzy$ holds in L^0 but not in M, R.
- The identity $xy = xyx$ holds in B^0 but not in R.

The descriptions of the operators $[\ ]_{\mathcal{X}}$ follows immediately from Result 3. Use Theorem 10 for the formulas for the corresponding languages. □

The corresponding results for the varieties $\mathcal{R}$, $\mathcal{R}^0$, $\mathcal{C}$ and $\mathcal{C}^0$ we get by duality. We can describe the joins of irreducible varieties of languages by Result 6 or we can use the following simple construction.

Theorem 13. *For conjunctive varieties of languages $\mathscr{K}$ and $\mathscr{L}$ and a non-empty finite set A, we have*

$$(\mathscr{K} \vee \mathscr{L})(A) = \{\, K \cap L \mid K \in \mathscr{K}(A) \text{ and } L \in \mathscr{L}(A)\,\}\ .$$

Proof. Obvious. □

A language $L \subseteq A^+$ is *closed* if $u, v \in L$ implies $uv \in L$. Recall that the *shuffle* of words $u, v \in A^+$ is the set $u \sqcup v =$

$$\{ u_1 v_1 \dots u_k v_k \mid k \in \mathbb{N},\ u = u_1 \dots u_k,\ v = v_1 \dots v_k,\ u_1, \dots, u_k, v_1, \dots, v_k \in A^* \}.$$

Thus the following system of identities characterizes languages all quotients of which are shuffle-closed

$$x_1 y_1 \dots x_k y_k \le x_1 \dots x_k \vee y_1 \dots y_k,\ x_1 y_1 \dots x_k y_k x_{k+1} \le x_1 \dots x_{k+1} \vee y_1 \dots y_k\ ,$$

$k \in \mathbb{N}$. Now the following is straightforward.

Theorem 14.

(i) All quotients of a language whose syntactic semiring has idempotent multiplication are shuffle closed.

(ii) Each language with idempotent syntactic semigroup with all quotients being closed has syntactic semiring with idempotent multiplication.

□

4 Conjunctive versus positive varieties of languages

For a class $\mathcal{V}$ of semigroups, we put

$$\mathcal{V}^{\le} = \{\ (S, \cdot, \le) \text{ is an ordered semigroup } \mid (S, \cdot) \in \mathcal{V}\ \}\ ,$$

and for a class $\mathcal{W}$ of ordered semigroups, we set

$$\mathcal{W}_+ = \{\ (S, \cdot, \le) \in \mathcal{W} \mid (S, \cdot, \le) \text{ satisfies } xyx \le x\ \} \text{ and}$$

$$\mathcal{W}_- = \{\ (S, \cdot, \le) \in \mathcal{W} \mid (S, \cdot, \le) \text{ satisfies } x \le xyx\ \}\ .$$

Result 15 ([2]). *The lattice of all varieties of ordered normal bands consists of 8 varieties of the form $\mathcal{V}^{\le}$ where $\mathcal{V}$ is a variety of normal bands and 8 varieties of the forms $\mathcal{W}_+$, $\mathcal{W}_-$ where $\mathcal{W} \in \{\ \mathcal{Sl}^{\le}, \mathcal{LNB}^{\le}, \mathcal{RNB}^{\le}, \mathcal{NB}^{\le}\ \}$.*

The operator Fin *is a bijection of this lattice onto the lattice of all pseudovarieties of ordered normal bands.*

As announced in [2] and also proved by the authors of [4] (unpublished) any other variety of ordered bands is of the form $\mathcal{V}^{\le}$ for a variety $\mathcal{V}$ of bands. Therefore we have exactly 21 varieties of ordered regular bands and (since they are generated by their finite members) exactly 21 pseudovarieties of ordered regular bands.

Theorem 16. *Let $\mathcal{L}$ be a conjunctive variety of languages. Then the smallest positive variety of languages is of the form $\mathcal{L}^{\cup}$, where for each non-empty finite set A :*

$$\mathcal{L}^{\cup}(A) = \{\ L_1 \cup \dots \cup L_k \mid k \in \mathbb{N},\ L_1, \dots, L_k \in \mathcal{L}(A)\ \}\ .$$

Proof. The result is obvious. □

Theorem 17.

(i) For $\mathcal{X} \in \mathcal{V}(\mathcal{I})$ with $\underline{\mathcal{X}} = \{\mathcal{T}, \mathcal{S}l\}$, exactly $\mathsf{L}(\mathcal{TS}) = \mathsf{L}^{\leq}(\mathcal{T})$ and $\mathsf{L}(\mathcal{D}) = \mathsf{L}^{\leq}(\mathcal{S}l_{+}^{\leq})$ are positive varieties of languages. Moreover,

$$(\mathsf{L}(\mathcal{M}))^{\cup} = \mathsf{L}^{\leq}(\mathcal{S}l_{-}^{\leq}) \text{ and } (\mathsf{L}(\mathcal{D} \vee \mathcal{M}))^{\cup} = (\mathsf{L}(\mathcal{S}))^{\cup} = \mathsf{L}^{\leq}(\mathcal{S}l^{\leq}) .$$

(ii) For $\mathcal{X} \in \mathcal{V}(\mathcal{I})$ with $\underline{\mathcal{X}} = \{\mathcal{LZ}, \mathcal{LNB}\}$, exactly $\mathsf{L}(\mathcal{L}) = \mathsf{L}^{\leq}(\mathcal{LZ})$ and $\mathsf{L}(\mathcal{L}^0) = \mathsf{L}^{\leq}(\mathcal{LNB}_{+}^{\leq})$ are positive varieties of languages. Moreover,

$$(\mathsf{L}(\mathcal{L} \vee \mathcal{M}))^{\cup} = \mathsf{L}^{\leq}(\mathcal{LNB}_{-}^{\leq}),\ (\mathsf{L}(\mathcal{L} \vee \mathcal{D}))^{\cup} = \mathsf{L}(\mathcal{L}^0), \quad \text{and}$$

$$(\mathsf{L}(\mathcal{L} \vee \mathcal{D} \vee \mathcal{M}))^{\cup} = \cdots = \mathsf{L}^{\leq}(\mathcal{LNB}^{\leq}) .$$

(iii) For $\mathcal{X} \in \mathcal{V}(\mathcal{I})$ with $\underline{\mathcal{X}} = \{\mathcal{ReB}, \mathcal{NB}\}$, exactly $\mathsf{L}(\mathcal{L} \vee \mathcal{R}) = \mathsf{L}^{\leq}(\mathcal{ReB})$ and $\mathsf{L}(\mathcal{L}^0 \vee \mathcal{R}^0) = \mathsf{L}^{\leq}(\mathcal{NB}_{+}^{\leq})$ are positive varieties of languages. Moreover,

$$(\mathsf{L}(\mathcal{L} \vee \mathcal{M} \vee \mathcal{R}))^{\cup} = \mathsf{L}^{\leq}(\mathcal{NB}_{-}^{\leq}),\ (\mathsf{L}(\mathcal{L} \vee \mathcal{D} \vee \mathcal{R}))^{\cup} = \mathsf{L}(\mathcal{L}^0 \vee \mathcal{R}^0), \quad \text{and}$$

$$(\mathsf{L}(\mathcal{L} \vee \mathcal{D} \vee \mathcal{M} \vee \mathcal{R}))^{\cup} = \cdots = \mathsf{L}^{\leq}(\mathcal{NB}^{\leq}) .$$

(iv) For all other $\mathcal{X} \in \mathcal{V}(\mathcal{I})$, $\mathsf{L}(\mathcal{X})$ is not a positive variety and

$$(\mathsf{L}(\mathcal{X}))^{\cup} = \mathsf{L}^{\leq}(\underline{\mathcal{X}}^{\leq}) .$$

Proof. All follows from simple calculations. □

Example. Let $A = \{a, b\}$, $L = a^+ \cup b^+$. Then the (ordered) syntactic semigroup is idempotent, but the syntactic semiring is not. Indeed, using the notation from [12] we have $D = \{ a^+ \cup b^+, a^*, b^*, \emptyset \}$ and the transformation semigroup consists of transformations given by a, b, ab having the presentation $a^2 = a$, $b^2 = b$, $ab = ba = 0$. Further, $\overline{D} = D \cup \{a^+, b^+, 1\}$ and there is a new transformation given by $\{a, b\}$. This element is not an idempotent.

We can derive the result from Theorem 14 : $a^2, b \in L$ but $aba \in a^2 \sqcup b$, $aba \notin L$.

References

[1] J. Almeida, *Finite Semigroups and Universal Algebra*, World Scientific, 1994.

[2] S.J. Emery, Varieties and pseudovarieties of ordered normal bands, Semigroup Forum 58 (3) (1999), 348–366.

[3] S. Ghosh, F. Pastijn and X. Zhao, Varieties generated by ordered bands I, Order, accepted.

[4] M. Kuřil and L. Polák, On varieties of semilattice-ordered semigroups, Semigroup Forum, to appear.

[5] R. McKenzie and A. Romanowska, Varieties of ·-distributive bisemilattices, in Contribution to General Algebra (Proc. Klagenfurt Conference 1978), Heyn, Klagenfurt 1979, pages 213–218.

[6] O. Neto and H. Sezinando, Band monoid languages revisited, Semigroup Forum 61 (1) (2000), 32–45.

[7] F. Pastijn and X. Zhao, Varieties of idempotent semirings with commutative addition, Algebra Universalis, to appear.

[8] F. Pastijn, Varieties generated by ordered bands II, Order, accepted.

[9] J.-E. Pin, *Varieties of Formal Languages*, Plenum, 1986.

[10] J.-E. Pin, Syntactic semigroups, Chapter 10 in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa eds, Springer, 1997.

[11] L. Polák, A classification of rational languages by semilattice-ordered monoids, Archivum Mathematicum (Brno) 40 (2004), 395-406.

[12] L. Polák, Syntactic semiring and language equations, in *Proc. of the Seventh International Conference on Implementation and Application of Automata, Tours 2002*, Springer Lecture Notes in Computer Science, Vol. 2608, pages 182-193 (2003).

[13] M. Petrich and N.R. Reilly, *Completely Regular Semigroups*, Canadian Mathematical Society Series of Monographs and Advanced Texts. 23. Chichester: Wiley, 1999.

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János