On the Computation of Quotients and Factors of Regular Languages*

Mircea Marin[1] and Temur Kutsia[2]

[1] Graduate School of Systems and Information Engineering
University of Tsukuba
Tsukuba 305-8573, Japan
[2] Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria

Abstract. Quotients and factors are important notions in the design of various computational procedures for regular languages and for the analysis of their logical properties. We propose a new representation of regular languages, by linear systems of language equations, which is suitable for the following computations: language reversal, left quotients and factors, right quotients and factors, and factor matrix. We present algorithms for the computation of all these notions, and indicate an application of the factor matrix to the computation of solutions of a particular language reconstruction problem. The advantage of these algorithms is that they all operate only on linear systems of language equations, while the design of the same algorithms for other representations often require translations to other representations.

1 Introduction

In [3], Brzozowski introduced the notion of word derivative and proposed an elegant algorithm to turn a regular expression into a deterministic finite automaton (DFA). His insight was that the word derivatives of a regular expression serve as states of the corresponding DFA. Antimirov [1] went a step further by introducing the notion of partial word derivative and observing that partial word derivatives of regular expressions serve as states of a corresponding nondeterministic finite automaton (NFA) with a relatively small number of states. He also noticed that computations based on partial word derivatives can improve the efficiency of Brzozowski's algorithm of translating regular expressions into finite automata. A much broader analysis of the role of word derivatives in the algebra of regular expressions was carried out by Conway [4]. His notion of left (resp. right) derivate of a language coincides with the current notion of language left quotient (resp. right quotient), whereas his notion of right factor coincides with the recently proposed notion of product derivative with respect to some

* A preliminary version of this paper [7] was presented at the Sixth Asian Workshop on Foundations of Software (AWFS 2009).

language [10]. Left quotients and product derivatives are natural generalizations of the notion of word derivative, but in slightly different directions:

- The left quotient of a language E with respect to a language F is the *union* of all word derivatives of E with respect to a word from F,
- The product derivative of E with respect to F in [10], is the *intersection* of all word derivatives of E with respect to a word from F.

In the same way, the notion of right quotient coincides with the notion of product antiderivative with respect to some language, and right quotients and product antiderivatives are natural generalizations of the notion of word antiderivative in different directions.

Quotients and factors are relevant in the design of various computational procedures for regular languages and the analysis of their logical properties. In particular, the factors of a regular language can be arranged in a factor matrix which exhibits several interesting properties [4], making it an useful tool for solving problems like, e.g. the language reconstruction problem discussed in Sect. 4.

The purpose of this paper is to identify a representation of regular languages that is suitable for the computation of all these notions. There are several well known representations of regular languages: by regular expressions, finite automata, systems of linear equations, etc. The choice of a representation depends on the computation under consideration, and conversions between different representations are often required. For example, the computation of quotients is well known for representations of regular languages by finite automata [5, Thm. 3.6]. Algorithms for the computation of factors are more recent developments. In [10], the right factor of a regular language with respect to another regular language is obtained from the computation of the greatest fixed point of a continuous operator which renders the result as a finite intersection of languages represented by regular expressions.

In this paper we identify the notions of product derivative and reversal of a language to be useful in the computation of left/right factors and factor matrix of regular languages, and a new representation of regular languages, by so called linear systems of language equations (LSs for short). LSs are a special case of the more general notion of systems of linear equations as defined, e.g., in [6], which turn out to be suitable for several computations on regular languages, including those mentioned before. We propose algorithms that manipulate LSs for the computation of language reversal, product derivative and product antiderivative, left/right quotients and factors, and factor matrix, without need to perform costly translations between different representations. Thus, by introducing LSs we define a common framework for the analysis of all these computations. We also indicate an application of the factor matrix to the computation of the solutions of a language reconstruction problem.

The paper is structured as follows. In Sect. 2 we present preliminary notions and known theoretical results for regular languages and regular expressions, and propose the alternative representation of regular languages by LSs. Conversion

algorithms between LSs and regular expressions, and between LSs and nondeterministic finite automata are given in Sects. 2.3, 2.4, and 2.5. An algorithm for the computation of language reversal of a language represented by an LS is given in Sect. 2.6. Section 3 presents our algorithms for the computation of quotients, factors, product derivative, and factor matrix. Section 4 indicates an interesting language reconstruction problem with an algorithmic solution based on the notion of factor matrix. Section 5 concludes.

2 Preliminaries

From now on we assume given a finite alphabet $\mathcal{A}$ and a set of language variables $\mathcal{X}$. Elements of $\mathcal{A}$ will be denoted by a, possibly subscripted, and elements of $\mathcal{X}$ will be denoted by x, y, z, possibly subscripted. We write $\mathcal{A}^*$ for the set of words of symbols from $\mathcal{A}$, and ϵ for the empty word. The *length* of a word w is denoted by $|w|$. A *language* is an arbitrary set of words, that is, a subset of $\mathcal{A}^*$. The *sum* of two languages is their union $E \cup F$. Their *product* (or *concatenation*) $E\,F$ is the set $\{vw \mid v \in E, w \in W\}$. The *asterate* E^* is the set $\bigcup_{n\in\mathbb{N}} E^n$ where $E^0 = \{\epsilon\}$ and $E^{n+1} := E\,E^n$. These 3 operations are called *regular operations*. The set $\mathbf{Reg}(\mathcal{A})$ of *regular languages* over $\mathcal{A}$ is the set of languages obtained from languages of the set $\{\emptyset, \{\epsilon\}\} \cup \{\{a\} \mid a \in \mathcal{A}\}$ by a finite number of applications of regular operations.

The *reversal* of a language L is the language $L^{\mathtt{r}} := \{a_n \dots a_1 \mid a_1 \dots a_n \in L\}$. Regular languages are closed under intersection, complementation, and reversal, that is, $L_1 \cap L_2, \mathcal{A}^* - L, L^{\mathtt{r}} \in \mathbf{Reg}(\mathcal{A})$ whenever $L, L_1, L_2 \in \mathbf{Reg}(\mathcal{A})$.

The set $\mathcal{T}(\mathcal{A}, \mathcal{X})$ of *regular expressions* over $\mathcal{A}$ and $\mathcal{X}$ is defined recursively as follows:

- The symbols of $\mathcal{A}$, $\mathcal{X}$, and 0 and 1 are regular expressions;
- If r and s are regular expressions, then so are $r \cdot s$, $r + s$ and $s^\star$.

We write $\mathcal{T}(\mathcal{A})$ instead of $\mathcal{T}(\mathcal{A}, \emptyset)$. The *alphabetic width* of $r \in \mathcal{T}(\mathcal{A})$, denoted by $\|r\|$, is the number of occurrences of symbols from $\mathcal{A}$ in r. The *reversal* of $r \in \mathcal{T}(\mathcal{A}, \mathcal{X})$ is the regular expression $r^{\mathtt{r}}$ defined by: $r^{\mathtt{r}} := r$ if $r \in \{0, 1\} \cup \mathcal{A} \cup \mathcal{X}$; $(r + s)^{\mathtt{r}} := r^{\mathtt{r}} + s^{\mathtt{r}}$; $(r \cdot s)^{\mathtt{r}} := s^{\mathtt{r}} \cdot r^{\mathtt{r}}$; and $(r^\star)^{\mathtt{r}} := (r^{\mathtt{r}})^\star$.

In general, if M is a set and $m, n \in \mathbb{N} \setminus \{0\}$, then $\mathcal{M}_{m,n}(M)$ denotes the set of $m \times n$ matrices with elements from M. We write $\mathbf{M}_{i,j}$ for the element at position (i, j) of $\mathbf{M} \in \mathcal{M}_{m,n}(M)$.

An *assignment* for $r \in \mathcal{T}(\mathcal{A}, \mathcal{X})$ is a mapping $\sigma : \mathcal{X} \to 2^{\mathcal{A}^*}$. The interpretation $[\![r]\!]_\sigma$ of a regular expression $r \in \mathcal{T}(\mathcal{A}, \mathcal{X})$ with respect to an assignment σ is the language defined recursively by:

- $[\![0]\!]_\sigma := \emptyset$, $[\![1]\!]_\sigma := \{\epsilon\}$, $[\![a]\!]_\sigma := \{a\}$, $[\![x]\!]_\sigma := \sigma(x)$,
- $[\![r \cdot s]\!]_\sigma := [\![r]\!]_\sigma\,[\![s]\!]_\sigma$,
- $[\![r + s]\!]_\sigma := [\![r]\!]_\sigma \cup [\![s]\!]_\sigma$, and
- $[\![r^\star]\!]_\sigma := [\![r]\!]_\sigma^*$.

Note that, if $r \in \mathcal{T}(\mathcal{A})$ then $[\![r]\!]_\sigma$ does not depend on σ, and in this case we write $[\![r]\!]$ instead of $[\![r]\!]_\sigma$. We also consider the matriceal operations $+$ and $\cdot$ as being the natural extensions of the operations $+$ and $\cdot$ on $\mathcal{T}(\mathcal{A})$; and the interpretation $[\![\_]\!]_\sigma : \mathcal{M}_{m,n}(\mathcal{T}(\mathcal{A},\mathcal{X})) \to \mathcal{M}_{m,n}(\mathbf{Reg}(\mathcal{A}))$ defined by $([\![\mathbf{A}]\!]_\sigma)_{i,j} := [\![\mathbf{A}_{i,j}]\!]_\sigma$ for all assignments $\sigma : \mathcal{X} \to 2^{\mathcal{A}^*}$, $1 \leq i \leq m$ and $1 \leq j \leq n$.

Note that the equality $[\![r^{\mathtt{r}}]\!] = [\![r]\!]^{\mathtt{r}}$ holds for all $r \in \mathcal{T}(\mathcal{A})$. Two regular expressions $r, s \in \mathcal{T}(\mathcal{A},\mathcal{X})$ are *equivalent* for assignment σ, and we write $r \doteq_\sigma s$, if $[\![r]\!]_\sigma = [\![s]\!]_\sigma$. Two matrices $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{m,n}(\mathcal{T}(\mathcal{A},\mathcal{X}))$ are *equivalent* for assignment σ, and we write $\mathbf{A} \doteq_\sigma \mathbf{B}$, if $[\![\mathbf{A}]\!]_\sigma = [\![\mathbf{B}]\!]_\sigma$.

The quotients of two languages E and F are defined as follows [2]:

- the *left quotient* of E with respect to F is

$$F^{-1}E := \{w \mid \exists v.(v \in F \wedge vw \in E)\} = \bigcup_{v \in F} \{v\}^{-1}E.$$

- the *right quotient* of E with respect to F is

$$E\,F^{-1} := \{w \mid \exists v.(v \in F \wedge wv \in E)\} = \bigcup_{v \in F} E\{v\}^{-1}.$$

If F is a singleton language $\{w\}$ then we write simply $w^{-1}E$ and Ew^{-1} instead of the more cumbersome notations $\{w\}^{-1}E$ and $E\{w\}^{-1}$. We call these sets the *word derivative* and *word antiderivative* of E with respect to w. It is well known that a regular language E has only finitely many left quotients $F^{-1}E$ (even for irregular language F), and all left quotients are regular [4]. The proof of this result can be easily adapted to conclude the same property for right quotients.

2.1 Factors of Regular Languages

According to [4], an *n-subfactorization* of a language L is a tuple of languages $(L_1, \ldots, L_n)$ such that the language concatenation $L_1 \cdots L_n$ is a subset of L. The relation '$<$' defined on n-tuples of languages by

$$(L_1, \ldots, L_n) < (L'_1, \ldots, L'_n) \text{ iff } L_i \subseteq L'_i \text{ for all } 1 \leq i \leq n \text{ and } L_j \neq L'_j \text{ for some } 1 \leq j \leq n$$

is irreflexive and transitive, thus a partial order. An *n-factorization* of L is an n-subfactorization of L which is $<$-maximal. The components of such n-factorizations are called *factors*. We write $\mathtt{Facts}(L)$ for the possibly infinite set of factors of L. A *factorization* of L is an n-factorization for some $n \geq 1$. A *left* (resp. *right*) factor of L is the leftmost (resp. rightmost) term in a factorization of L. We denote the set of left factors of L by $\mathtt{LF}(L)$, and the set of right factors of L by $\mathtt{RF}(L)$.

It is important to notice that there are two popular but different ways to define language factorization. In [8, 9], an n-factorization of a regular language L is a decomposition into n languages $L_1, \ldots, L_n$ such that $L_1 \cdots L_n = L$. In this paper we have considered factorizations to be $<$-maximal subfactorizations, as suggested by Conway in [4]. Therefore, we can encounter n-factorizations $(L_1, \ldots, L_n)$ of a language L such that $L_1 \cdots L_n \subsetneq L$.

Example 1. Let $\mathcal{A} = \{a, b\}$ and $L = [\![a^\star b^\star]\!]$. There are four 2-factorizations of L: $(\emptyset, [\![(a+b)^\star]\!])$, $([\![a^\star]\!], [\![a^\star b^\star]\!])$, $([\![a^\star b^\star]\!], [\![b^\star]\!])$, and $([\![(a+b)^\star]\!], \emptyset)$. Note that $([\![(a+b)^\star]\!], \emptyset)$ is a factorization of L although $[\![(a+b)^\star]\!]\,\emptyset \neq L$. This is so because:

- $([\![(a+b)^\star]\!], \emptyset)$ is a 2-subfactorization: $[\![(a+b)^\star]\!]\,\emptyset = \emptyset \subset L$.
- $([\![(a+b)^\star]\!], \emptyset)$ is $<$-maximal: the first subfactor $[\![a+b]\!]^*$ can not be enlarged any further, and $\emptyset$ is the only value of L_2 for which $[\![a+b]\!]^*\, L_2 \subseteq [\![a^\star b^\star]\!]$.

□

For $E, F \in \mathbf{Reg}(\mathcal{A})$ we define the languages

$$E \lhd F := \max_{\subseteq}\{L \mid L\,F \subseteq E\} \quad \text{and} \quad F \rhd E := \max_{\subseteq}\{R \mid F\,R \subseteq E\}.$$

The language $F \rhd E$ is called the *product derivative* of E with respect to F, and $E \lhd F$ the *product antiderivative* of E with respect to F. Note that

$$E \lhd F = \{w \mid \forall v.\ (v \in F \to wv \in E)\} = \bigcap_{v \in F} E\{v\}^{-1},$$
$$F \rhd E = \{w \mid \forall v.\ (v \in F \to vw \in E)\} = \bigcap_{v \in F} \{v\}^{-1}E,$$

and that if (L, R) is a 2-factorization of a language E then $L = E \lhd R$ and $R = L \rhd E$. A remarkable fact [4] is that, if E is a regular language then $\texttt{Facts}(E)$ is a finite set. Moreover, if $\{L_1, \ldots, L_n\} := \texttt{LF}(E)$, $R_i := L_i \rhd E$, and $F_{i.j} := L_i \rhd L_j$ for all $1 \leq i, j \leq n$, then $\texttt{RF}(E) = \{R_1, \ldots, R_n\}$, $\texttt{Facts}(E) = \{F_{i,j} \mid 1 \leq i, j \leq n\}$, and the following properties hold:

(f1) All elements of $\texttt{Facts}(L)$ are regular languages,
(f2) $F_{i,k}\,F_{k,j} \subseteq F_{i,j}$ for all $1 \leq i, j, k \leq n$, and
(f3) There exist $l, r \in \{1, \ldots, n\}$ such that $F_{l,r} = L$ and, for any k-subfactorization $(L_1, \ldots, L_k)$ of L there exist $u_1, \ldots, u_{k+1} \in \{1, \ldots, n\}$ with $u_1 = l$ and $u_{k+1} = r$, such that $L_i \subseteq F_{u_i, u_{i+1}}$ for all $1 \leq i \leq k$. In this case, the matrix $(F_{i,j})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}}$ is called the *factor matrix* of L.

To appreciate the relevance of this result, consider the following language reconstruction problem:

Given $r \in \mathcal{T}(\mathcal{A}, \{x_1, \ldots, x_k\})$ and $L \in \mathbf{Reg}(\mathcal{A})$,
Find all $<$-maximal tuples $(L_1, \ldots, L_k)$ of languages over $\mathcal{A}$ such that $[\![r]\!]_\sigma \subseteq L$ holds for the assignment $\sigma = \{x_1 \mapsto L_1, \ldots, x_k \mapsto L_k\}$.

This problem shows up in situations when we use r as a pattern for the language produced by a combination of k input languages and we wish to compute all $<$-maximal tuples of languages $(L_1, \ldots, L_k)$ that can instantiate $(x_1, \ldots, x_k)$ such that the corresponding instance of r is valid with respect to the output specification language L. Conway noticed in [4, Ch. 6] that this problem has finitely many solutions, that the solutions consist of regular languages, and that they can be found "with patience" using a generate and test approach that makes use of the factor matrix of L. Since "with patience" is a rather vague statement, we indicate an algorithm to solve this problem in Sect. 4.

2.2 Linear Systems of Language Equations

Let $\mathcal{A}$ be a finite alphabet, and

$$\mathrm{lin}(\mathcal{A}) := \{0\} \cup \{a_1 + \ldots + a_p \mid p > 0 \text{ and } a_1, \ldots, a_p \in \mathcal{A}\}.$$

Note that $\mathrm{lin}(\mathcal{A}) \in 2^{\mathcal{T}(\mathcal{A})}$. A *linear system of language equations* (LS) over $\mathcal{A}$ is

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \mathbf{B} + \mathbf{A} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \tag{1}$$

where $\mathbf{A} \in \mathcal{M}_{n,n}(\mathrm{lin}(\mathcal{A}))$, $\mathbf{B} \in \mathcal{M}_{n,1}(\{0,1\})$, and $x_1, \ldots, x_n$ are language variables. A *solution* of (1) is an assignment $\sigma : \{x_1, \ldots, x_n\} \to 2^{\mathcal{A}^*}$ such that $\mathbf{X} \doteq_\sigma \mathbf{B} + \mathbf{A} \cdot \mathbf{X}$ where $\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$.

A system of type (1) is a special case of the more general notion of system of linear equations as defined, e.g., in [6], where the elements of $\mathbf{A}$ and $\mathbf{B}$ are arbitrary regular expressions. It is well known known that (1) has the solution σ defined by $\sigma(x_i) = [\![\mathbf{A}^\star \cdot \mathbf{B}]\!]_{i,1}$ for all $1 \leq i \leq n$, where $\mathbf{A}^\star \in \mathcal{M}_{n,n}(\mathcal{T}(\mathcal{A}))$ is the *asterate* of $\mathbf{A}$. Note that the elements of $\mathbf{A}$ denote ϵ-free languages, therefore the solution of (1) is unique.

We call (1) an *LS representation* of a language L if $[\![\mathbf{A}^\star \cdot \mathbf{B}]\!]_{1,1} = L$. For $x_i \in \mathcal{X}$ and $a \in \mathcal{A}$ we also define

- $\mathtt{rhs}(x_i) := \mathbf{B}_{i,1} + \sum_{j=1}^{n} \mathbf{A}_{i,j} \cdot x_j$, and
- $\mathtt{rhs}_a(x_i) := \{x_j \in \mathcal{X} \mid a \cdot x_j \text{ occurs in the equation of } x_i \text{ in LS (1)}\}$.

2.3 Converting LSs to Regular Expressions

If (1) is an LS representation of L then $L = [\![r]\!]$ where r is the regular expression $(\mathbf{A}^\star \cdot \mathbf{B})_{1,1}$. Thus, languages represented by an LS are regular, and the conversion of the LS representation to an equivalent regular expression amounts to the computation of $(\mathbf{A}^\star \cdot \mathbf{B})_{1,1}$.

2.4 Converting Regular Expressions to LSs

We consider the problem of computing an LS representation for $[\![r]\!]$ when $r \in \mathcal{T}(\mathcal{A})$. We can solve this problem by using Antimirov's improvement of Brzozowski algorithm [1, Sect. 4.2] which computes, for every $r \in \mathcal{T}(\mathcal{A})$, the following:

- A finite set $\partial_{\mathcal{A}^*}(r) := \{r_1, \ldots, r_n\} \subseteq \mathcal{T}(\mathcal{A})$, called the *partial word derivatives* of r,
- $\mathbf{A} \in \mathcal{M}_{n,n}(\mathrm{lin}(\mathcal{A}))$ and $\mathbf{B} \in \mathcal{M}_{n,1}(\{0,1\})$

such that $n \leq \|r\| + 1$ and

$$\begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} \doteq \mathbf{B} + \mathbf{A} \cdot \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}. \tag{2}$$

We recall from [1] that $\partial : (\mathcal{A}^* \cup 2^{\mathcal{A}^*}) \times \mathcal{T}(\mathcal{A}) \to 2^{\mathcal{T}(\mathcal{A})}$ satisfies the conditions:

- $\partial_\epsilon(s) := s$, $\partial_{aw}(s) := \bigcup_{s' \in \partial_a(s)} \partial_w(s')$, $\partial_W(s) := \bigcup_{w \in W} \partial_w(s)$, and
- $w^{-1}[\![s]\!] = \bigcup_{s' \in \partial_w(s)} [\![s']\!]$

for all $a \in \mathcal{A}$, $w \in \mathcal{A}^*$, $W \subseteq \mathcal{A}^*$, and $s \in \mathcal{T}(\mathcal{A})$. Therefore we can assume without loss of generality that $r_1 = r$, and learn from (2) that $\mathbf{X} = \mathbf{B} + \mathbf{A} \cdot \mathbf{X}$ with $\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ is an LS representation of $[\![r]\!]$.
Antimirov's algorithm identifies regular expressions modulo fine similarity, which is an equivalence relation coarser than $\doteq$. Therefore, (2) may not be minimal in the number of equations among the LS representations of $[\![r]\!]$. We can minimize the system if we identify the partial word derivatives of r modulo $\doteq$.

2.5 Converting between LSs and Finite Automata

The translations between the representation of a regular language by an LS and a representation by a nondeterministic finite automaton are straightforward:

- Given the LS (1), we construct the automaton $(Q, \mathcal{A}, \Delta, x_1, F)$ with $Q := \{x_1, \ldots, x_n\}$, $F = \{x_i \mid \mathbf{B}_{i,1} = 1\}$, and $\Delta = \{(x_i, a, x_j) \mid x_j \in \mathtt{rhs}_a(x_i)\}$.
- Given a finite automaton $(Q, \mathcal{A}, \Delta, x_1, F)$ with $Q = \{x_1, \ldots, x_n\}$, we construct the LS (1) with $\mathbf{A}_{i,j} := \sum_{(x_i, a, x_j) \in \Delta} a$, and $\mathbf{B}_{i,1} := 1$ if $x_i \in F$.

2.6 LS Computation for the Reversal of a Regular Language

Suppose L is a regular language represented by the LS

$$S : \begin{cases} x_1 = b_1 + \ell_{1,1} \cdot x_1 + \ldots + \ell_{1,n} \cdot x_n \\ \vdots \\ x_n = b_n + \ell_{n,1} \cdot x_1 + \ldots + \ell_{n,n} \cdot x_n \end{cases}$$

with $b_i \in \{0, 1\}$ and $\ell_{i,j} \in \mathrm{lin}(\mathcal{A})$. In order to compute an LS for $L^{\mathtt{r}}$ we can proceed as follows. Then L coincides with the language accepted by the nondeterministic finite automaton $N = (Q, \mathcal{A}, \Delta, x_1, F)$ where $Q := \{x_1, \ldots, x_n\}$, $F := \{x_i \mid b_i = 1\}$, and $\Delta := \{(x_i, a, x_j) \mid a \text{ occurs as summand in } \ell_{i,j}\}$. By reverting the transitions of N, adding a fresh new state x_0 with ϵ-transitions to all final states of N, and making x_1 final state, we obtain an NFA with ϵ-transitions

for $L^{\mathtt{r}}$, namely $N_\epsilon^{\mathtt{r}} := (Q \cup \{x_0\}, \mathcal{A}, \Delta^{\mathtt{r}} \cup \Delta_\epsilon, x_0)$ where $\Delta_\epsilon := \{x_0 \to x_i \mid x_i \in F\}$ and $\Delta^{\mathtt{r}} := \{(x_j, a, x_i) \mid (x_i, a, x_j) \in \Delta\}$.

Let $N^{\mathtt{r}}$ be the NFA produced from $N^{\mathtt{r}}$ by the elimination of ϵ-transitions. It is not hard to see that an LS representation of the language accepted by $N^{\mathtt{r}}$ is

$$S^{\mathtt{r}} : \begin{cases} x_0 = c_0 + \ell_{0,1} \cdot x_1 + \ldots + \ell_{0,n} \cdot x_n \\ x_1 = c_1 + \ell_{1,1} \cdot x_1 + \ldots + \ell_{n,1} \cdot x_n \\ \quad \vdots \\ x_n = c_n + \ell_{1,n} \cdot x_1 + \ldots + \ell_{n,n} \cdot x_n \end{cases}$$

where $c_0 = b_1$, $c_1 = 1$, $c_j = 0$ for $2 \leq j \leq n$, and $l_{0,k} := \sum_{i \in \{j \mid b_j = 1\}} \ell_{i,k}$ for $1 \leq k \leq n$. We claim that the computational complexity of $S^{\mathtt{r}}$ from S is linear in the size of S. This is so because:

- The elements of the linear matrix of $S^{\mathtt{r}}$ at positions $(1,1), \ldots, (1,n)$ are $\ell_{0,1}$, $\ldots$, $\ell_{0,n}$, and the computation of every such element has complexity $O(n)$. Thus, the overall computation of these n elements has complexity $O(n^2)$.
- The values of all other elements that characterize $S^{\mathtt{r}}$ are either constants 0 or 1, or can be picked up from the LSH S. There are $n \cdot (n+1)$ such elements, and the computation of every such element is $O(1)$. Thus, the overall computation of the remaining elements has complexity $O(n^2)$.
- The size of S is $\Omega(n^2)$. Thus, the complexity of computing $S^{\mathtt{r}}$ is linear in the size of S.

3 Computational Methods for Quotients and Factors

In this section we initiate the study of LSs as a common representational framework for several computations in the algebra of regular languages, by avoiding translations to other equivalent representations. We have already seen in Sect. 2.6 how to compute an LS for the reversal of a language L directly from an LS representation of L. We are especially interested in algorithms that take as input LS representations of two regular languages E and F, and produce

1. LS representations for the right and left quotient of E with respect to F.
2. LS representations for the right and left factors of E.
3. LS representations for $F \rhd E$ and $E \lhd F$.
4. LS representations for the elements of the factor matrix of E.

3.1 LS Computation for Right and Left Quotients

First, we consider the following problem:

Given languages $E, F \in \mathbf{Reg}(\mathcal{A})$ and the LSs $\mathbf{X} = \mathbf{B} + \mathbf{A} \cdot \mathbf{X}$ and $\mathbf{Y} = \mathbf{D} + \mathbf{C} \cdot \mathbf{Y}$ with $\mathbf{A} \in \mathcal{M}_{m,m}(\mathrm{lin}(\mathcal{A}))$, $\mathbf{C} \in \mathcal{M}_{n,n}(\mathrm{lin}(\mathcal{A}))$, $\mathbf{B} \in \mathcal{M}_{m,1}(\{0,1\})$, $\mathbf{D} \in \mathcal{M}_{n,1}(\{0,1\})$, such that $[\![\mathbf{A}^\star \cdot \mathbf{B}]\!]_{1,1} = E$ and $[\![\mathbf{C}^\star \cdot \mathbf{D}]\!]_{1,1} = F$
Compute an LS representation for the right quotient $E\,F^{-1}$.

Suppose σ and μ are the unique solutions of the LSs $\mathbf{X} = \mathbf{B} + \mathbf{A} \cdot \mathbf{X}$ and $\mathbf{Y} = \mathbf{D} + \mathbf{C} \cdot \mathbf{Y}$. Let $\begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} := \mathbf{X}$, $\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} := \mathbf{Y}$, $E_i := \sigma(x_i)$ for $1 \leq i \leq m$, and $F_j := \tau(y_j)$ for $1 \leq j \leq n$. Then $E_1 = E$, $F_1 = F$, and

$$E_i\, F^{-1} = [\![x_i]\!]_\sigma\, [\![y_1]\!]_\mu^{-1} = [\![\mathbf{B}_i + \sum_{j=1}^{m} \mathbf{A}_{i,j} \cdot x_j]\!]_\sigma\, [\![y_1]\!]_\mu^{-1} = [\![k_{i,1} + \sum_{j=1}^{m} \mathbf{A}_{i,j} \cdot z_j]\!]_\tau$$

for all $1 \leq i \leq m$, where

- $k_{i,j} = 1$ if $\epsilon \in [\![x_i]\!]_\sigma\, [\![y_j]\!]_\mu^{-1}$ and $k_{i,j} = 0$ otherwise.
- $\tau : \{z_1, \ldots, z_m\} \to 2^{\mathcal{A}^*}$ with $\tau(z_i) := [\![x_i]\!]_\sigma\, [\![y_1]\!]_\mu^{-1}$ for $1 \leq i \leq m$.

Thus $\begin{pmatrix} z_1 \\ \vdots \\ z_m \end{pmatrix} = \begin{pmatrix} k_{1,1} \\ \vdots \\ k_{m,1} \end{pmatrix} + \mathbf{A} \cdot \begin{pmatrix} z_1 \\ \vdots \\ z_m \end{pmatrix}$ is an LS representation of $E\, F^{-1}$, where we still have to compute the components $k_{1,1}, \ldots, k_{m,1}$ of the constant vector of the LS. For this purpose, we define the monotone operator

$$\mathtt{F} : 2^{\{x_1,\ldots,x_m\}\times\{y_1,\ldots,y_n\}} \to 2^{\{x_1,\ldots,x_m\}\times\{y_1,\ldots,y_n\}}$$

represented compactly by the following collection of inference rules

$$\frac{[\mathbf{B}_{i,1} = 1 \wedge \mathbf{D}_{j,1} = 1]}{\langle x_i, y_j\rangle} \qquad \frac{\langle x_k, y_l\rangle \quad [a \in \mathcal{A} \wedge x_k \in \mathtt{rhs}_a(x_i) \wedge y_l \in \mathtt{rhs}_a(y_j)]}{\langle x_i, y_j\rangle}$$

Each rule states that if the pair above the bar is in the input set of $\mathtt{F}$ and the conditions within square brackets hold, then the pair below is in the output set of $\mathtt{F}$. In particular:

1. The first rule states that if $\mathbf{B}_{i,1} = 1$ and $\mathbf{D}_{j,1} = 1$ then $\langle x_i, y_j\rangle \in \mathtt{F}(S)$ for any $S \in 2^{\{x_1,\ldots,x_m\}\times\{y_1,\ldots,y_n\}}$.
2. The second rule states that if $S \in 2^{\{x_1,\ldots,x_m\}\times\{y_1,\ldots,y_n\}}$, $\langle x_k, y_l\rangle \in S$, $a \in \mathcal{A}$, $x_k \in \mathtt{rhs}_a(x_i)$, and $y_l \in \mathtt{rhs}_a(y_j)$ then $\langle x_i, x_j\rangle \in \mathtt{F}(S)$.

These rules are designed to infer $\langle x_i, y_j\rangle$ if and only if x_i and y_j have a common word in their denotations, that is, iff $[\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu \neq \emptyset$, or equivalently, iff $\epsilon \in [\![x_i]\!]_\sigma [\![y_j]\!]_\mu^{-1}$. A proof of this fact will be given shortly. First, we note that the operator $\mathtt{F}$ defined on the finite cpo $(2^{\{x_1,\ldots,x_m\}\times\{y_1,\ldots,y_n\}}, \subseteq)$ is monotone. Therefore, the least fixed point $\mu\mathtt{F}$ of $\mathtt{F}$ is a finite computable set.

Lemma 1. *The least fixed point $\mu\mathtt{F}$ of $\mathtt{F}$ is the set $\{\langle x_i, y_j\rangle \mid \epsilon \in [\![x_i]\!]_\sigma [\![y_j]\!]_\mu^{-1})\}$.*

Proof. Let $M = \{\langle x_i, y_j\rangle \mid \epsilon \in [\![x_i]\!]_\sigma [\![y_j]\!]_\mu^{-1})\}$. We prove that $\langle x_i, y_j\rangle \in \mu\mathtt{F} \Rightarrow \langle x_i, y_j\rangle \in M$ by induction of the length of the inference derivation that witnesses that $\langle x_i, y_j\rangle \in \mu\mathtt{F}$. If $\langle x_i, y_j\rangle \in \mu\mathtt{F}$ was deduced by the inference rule

$$\frac{[\mathbf{B}_{i,1} = 1 \wedge \mathbf{D}_{j,1} = 1]}{\langle x_i, y_j\rangle}$$

then $\epsilon \in [\![\mathbf{B}_{i,1} + \sum_{k=1}^{m} \mathbf{A}_{i,k} \cdot x_k]\!]_\sigma \cap [\![\mathbf{D}_{j,1} + \sum_{k=1}^{n} \mathbf{C}_{i,k} \cdot y_k]\!]_\mu = [\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu$, thus $\epsilon \in [\![x_i]\!]_\sigma [\![y_j]\!]_\mu^{-1}$, which shows that $\langle x_i, y_j \rangle \in M$.

If $\langle x_i, y_j \rangle \in \mu\mathtt{F}$ was deduced by a derivation with the last inference rule

$$\frac{\langle x_k, y_l \rangle \quad [a \in \mathcal{A} \wedge x_k \in \mathtt{rhs}_a(x_i) \wedge y_l \in \mathtt{rhs}_a(y_j)]}{\langle x_i, y_j \rangle}$$

then $\langle x_k, y_l \rangle \in \mu\mathtt{F}$, $[\![a \cdot x_k]\!]_\sigma \subseteq [\![x_i]\!]_\sigma$ and $[\![a \cdot y_l]\!]_\mu \subseteq [\![y_j]\!]_\mu$. By induction hypothesis we have $\epsilon \in [\![x_k]\!]_\sigma [\![y_l]\!]_\mu^{-1}$. But $\epsilon \in [\![x_k]\!]_\sigma [\![y_l]\!]_\mu^{-1}$ iff $[\![x_k]\!]_\sigma \cap [\![y_l]\!]_\mu \neq \emptyset$ iff $[\![a \cdot x_k]\!]_\sigma \cap [\![a \cdot y_l]\!]_\mu \neq \emptyset$. Since $[\![a \cdot x_k]\!]_\sigma \cap [\![a \cdot y_l]\!]_\mu \subseteq [\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu$, we conclude $[\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu \neq \emptyset$, thus $\epsilon \in [\![x_i]\!]_\sigma [\![y_j]\!]_\mu^{-1}$, that is, $\langle x_i, y_j \rangle \in M$.

Next, we prove $M \subseteq \mu\mathtt{F}$. Note that $\langle x_i, y_j \rangle \in M$ iff there exists a word $w \in [\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu$, and thus we can define the complexity measure $|\langle x_i, y_j \rangle|$ of every $\langle x_i, y_j \rangle \in M$ as the minimal length of a word of $[\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu$. We prove $M \subseteq \mu\mathtt{F}$ by induction on the complexity measure of elements of M.

If $\langle x_i, y_j \rangle \in M$ and $|\langle x_i, y_j \rangle| = 0$ then $\epsilon \in [\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu$. This can happen only if $\mathbf{B}_{i,1} = 1$ and $\mathbf{D}_{j,1} = 1$, and in this case we can perform the derivation

$$\frac{[\mathbf{B}_{i,1} = 1 \wedge \mathbf{D}_{j,1} = 1]}{\langle x_i, y_j \rangle}$$

to deduce that $\langle x_i, y_j \rangle \in \mu\mathtt{F}$. If $\langle x_i, y_j \rangle \in M$ and $|\langle x_i, y_j \rangle| = p > 0$ then $\epsilon \notin [\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu$ and there exists $w = aw_1 \in [\![x_i]\!]_\sigma \cap [\![y_j]\!]_\mu$ of length p with $a \in \mathcal{A}$. This implies $w_1 \in [\![x_k]\!]$ for some $x_k \in \mathtt{rhs}_a(x_i)$ and $w_1 \in y_l$ for some $y_l \in \mathtt{rhs}_a(y_j)$. Thus $w_1 \in [\![x_k]\!]_\sigma \cap [\![y_l]\!]_\mu$, and we learn that $\langle x_k, y_l \rangle \in M$. Since w_1 has length $p-1$, we learn that $|\langle x_k, y_l \rangle| \leq p-1 < p$ and we can apply the induction hypothesis to conclude that $\langle x_k, y_l \rangle \in \mu\mathtt{F}$. Finally, we can use the facts that $x_k \in \mathtt{rhs}_a(x_i)$ and $y_l \in \mathtt{rhs}_a(y_j)$, and apply the inference step $\dfrac{\langle x_k, y_l \rangle}{\langle x_i, y_j \rangle}$ to conclude $\langle x_i, y_j \rangle \in \mu\mathtt{F}$. $\square$

Example 2. Consider the regular languages E and F represented by the LSs

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} a+b & a \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ and } \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 & b \\ 0 & a \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \text{ respectively.}$$

Note that the solutions of these LSs are $\sigma = \{x_1 \mapsto [\![(a+b)^\star \cdot a]\!], x_2 \mapsto [\![1]\!]\}$, $\tau = \{y_1 \mapsto [\![b \cdot a^\star]\!], y_2 \mapsto [\![a^\star]\!]\}$, thus $E = [\![(a+b)^\star \cdot a]\!]$ and $F = [\![b \cdot a^\star]\!]$.

According to our algorithm, an LS representation of $E\,F^{-1}$ is

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} k_{1,1} \\ k_{2,1} \end{pmatrix} + \begin{pmatrix} a+b & a \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

where $k_{i,j} \in \{0,1\}$ and $k_{i,j} = 1$ iff $\langle x_i, y_j \rangle$ belongs to the least fixed point of the monotone operator $\mathtt{F}$ on $2^{\{x_1,x_2\} \times \{y_1,y_2\}}$ defined by the inference rules

$$\frac{\langle x_1, y_2 \rangle}{\langle x_1, y_1 \rangle} \quad \frac{\langle x_1, y_2 \rangle}{\langle x_1, y_2 \rangle} \quad \frac{\langle x_2, y_2 \rangle}{\langle x_1, y_2 \rangle} \quad \frac{}{\langle x_2, y_2 \rangle}.$$

Since $\mu\mathtt{F} = \{\langle x_2, y_2\rangle, \langle x_1, y_2\rangle, \langle x_1, y_1\rangle\}$, we get $k_{1,1} = 1$, $k_{2,1} = 0$ and learn that

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \mathbf{A} \cdot \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \text{ with } \mathbf{A} = \begin{pmatrix} a+b & a \\ 0 & 0 \end{pmatrix}$$

is an LS representation of $E\,F^{-1}$. Since $\mathbf{A}^\star = \begin{pmatrix} (a+b)^\star & (a+b)^\star \cdot a \\ 0 & 1 \end{pmatrix}$, we learn that $E\,F^{-1} = \left[\!\!\left[\mathbf{A}^\star \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right]\!\!\right]_{1,1} = [\![(a+b)^\star]\!] = \{a, b\}^*$. □

Computing Right and Left Quotients: Given two languages E and F in an LS representation as it is specified at the beginning of this section, we would like to compute LS representations for the right quotient EF^{-1} and for the left quotient $F^{-1}E$. For EF^{-1}, we use the algorithm described in this section. For $F^{-1}E$, since $F^{-1}E = (E^{\mathtt{r}}(F^{\mathtt{r}})^{-1})^{\mathtt{r}}$, we proceed as follows:

1. Compute an LS representation of $E^{\mathtt{r}}$ from the LS representation of E, and an LS representation of $F^{\mathtt{r}}$ from the LS representation of F.
2. Compute an LS representation of $E^{\mathtt{r}}(F^{\mathtt{r}})^{-1}$ from the LSs computed before, by using the algorithm presented in this section.
3. Compute an LS representation of $(E^{\mathtt{r}}(F^{\mathtt{r}})^{-1})^{\mathtt{r}}$ from the LS representation of $E^{\mathtt{r}}(F^{\mathtt{r}})^{-1}$.

3.2 LS Computations for Right and Left Factors

We start with the following problem:

Given an LS representation $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \mathbf{B} + \mathbf{A} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ of a regular language E with $\mathbf{A} \in \mathcal{M}_{n,n}(\mathrm{lin}(\mathcal{A}))$,
Compute LS representations of all elements of $\mathtt{RF}(E)$.

First we introduce some auxiliary notions. Let $\mathcal{X} := \{x_1, \ldots, x_n\}$. For every $X \subseteq \mathcal{X}$ and $w \in \mathcal{A}^*$ we define the operation $D_w : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ by induction on the length of w: $D_\epsilon(X) := X$; and $D_{aw}(X) := D_w(\bigcup_{x \in \mathcal{X}} \mathtt{rhs}_a(x))$. If $W \subseteq \mathcal{A}^*$ then we define $D_W(X) := \{D_w(X) \mid w \in W\}$. For $\mathfrak{N} \subseteq 2^{\mathcal{X}}$ we denote by $\min_\subseteq(\mathfrak{N})$ the set of its $\subseteq$-minimal elements. Let σ be the unique solution of the given LS representation of E.

The following result is straightforward.

Lemma 2. *For any $w \in \mathcal{A}^*$ and $X \subseteq \mathcal{X}$ we have*

$$w^{-1} \left[\!\!\left[\sum_{x \in X} x \right]\!\!\right]_\sigma = \left[\!\!\left[\sum_{x \in D_w(X)} x \right]\!\!\right]_\sigma .$$

Let $\mathcal{G}$ be the directed graph with set of nodes $2^{\mathcal{X}}$ and set of edges $\{X \to D_a(X) \mid X \subseteq \mathcal{X}, a \in \mathcal{A}\}$. Then $D_{\mathcal{A}^*}(X)$ coincides with the set of all nodes of $\mathcal{G}$ reachable from X. Since $\mathcal{G}$ is finite, $D_{\mathcal{A}^*}(X)$ is computable for any X.

F is a right factor of E if and only if there exists a 2-subfactorization (L, F) of E where F is $\subseteq$-maximal; that is to say, if and only if $F = L \rhd E = \bigcap_{w \in L} w^{-1}E$ for some $L \subseteq \mathcal{A}^*$. This shows that the set of right factors of E is

$$\mathtt{RF}(E) = \left\{ \bigcap_{w \in L} w^{-1}[\![x_1]\!]_\sigma \;\middle|\; L \subseteq \mathcal{A}^* \right\} = \left\{ \bigcap_{w \in L} [\![\sum_{x \in D_w(\{x_1\})} x]\!]_\sigma \;\middle|\; L \subseteq \mathcal{A}^* \right\} =$$

$$\left\{ \bigcap_{w \in L} \left(\bigcup_{x \in D_w(\{x_1\})} [\![x]\!]_\sigma \right) \;\middle|\; L \subseteq \mathcal{A}^* \right\} = \left\{ \bigcap_{N \in \mathfrak{N}} \left(\bigcup_{x \in N} [\![x]\!]_\sigma \right) \;\middle|\; \mathfrak{N} \in 2^{D_{\mathcal{A}^*}(\{x_1\})} \right\}.$$

If we apply distributivity of intersection over union, we can identify for any $\mathfrak{N} \in 2^{D_{\mathcal{A}^*}(\{x_1\})}$ a set $\delta(\mathfrak{N})$ such that $\bigcap_{N \in \mathfrak{N}} \bigcup_{x \in N} [\![x]\!]_\sigma = \bigcup_{M \in \delta(\mathfrak{N})} \bigcap_{x \in M} [\![x]\!]_\sigma$.

Let $\mathtt{MS} := \{\min_{\subseteq}(\delta(\mathfrak{N})) \mid \mathfrak{N} \in 2^{D_{\mathcal{A}^*}(\{x_1\})}\}$. Note that the equality

$$\bigcup_{M \in \mathfrak{M}} \bigcap_{x \in M} [\![x]\!]_\sigma = \bigcup_{M \in \min_{\subseteq}(\mathfrak{M})} \bigcap_{x \in M} [\![x]\!]_\sigma$$

holds for all $\mathfrak{M} \subseteq 2^{\mathcal{X}}$. Therefore, we conclude that

$$\mathtt{RF}(E) = \left\{ \bigcup_{M \in \mathfrak{M}} \bigcap_{x \in M} [\![x]\!]_\sigma \;\middle|\; \mathfrak{M} \in \mathtt{MS} \right\}.$$

Let $\mathcal{Z} := \{z_M \mid M \subseteq \mathcal{X}\}$ be a set of fresh variables, and the assignment μ for $\mathcal{Z}$ defined by $\mu(z_M) := \bigcap_{x \in M} [\![x]\!]_\sigma$.

Suppose $\mathfrak{M} \in \mathtt{MS}$. We start producing an LS representation for the right factor $\bigcup_{M \in \mathfrak{M}} \bigcap_{x \in M} [\![x]\!]_\sigma$ as follows:

1. For every $M = \{x_{k_1}, \ldots, x_{k_p}\} \in \mathfrak{M}$ and $1 \leq i \leq p$ we have

$$x_{k_i} \doteq_\sigma \mathbf{B}_{k_i,1} + \sum_{j=1}^{n} \mathbf{A}_{k_i,j} \cdot x_j.$$

   By intersecting these p equations we obtain

$$z_M \doteq_\mu \min\{\mathbf{B}_{k_i,1} \mid 1 \leq i \leq p\} + \sum_{a \in \mathcal{A}} \sum_{x_{l_1} \in \mathtt{rhs}_a(x_{k_1})} \cdots \sum_{x_{l_p} \in \mathtt{rhs}_a(x_{k_p})} a \cdot z_{\{l_i \mid 1 \leq i \leq p\}}.$$

2. Like in step 1, we continue producing equations for every $M \subseteq \{x_1 \ldots, x_n\}$ such that z_M occurs in the right side of a produced equation. This process will eventually terminate because the set $\mathcal{Z}$ is finite. In this way we generate an LS with at least $|\mathfrak{M}|$ equations and at most 2^n equations.

3. Finally, we construct the first equation of the LS representation of the language $\bigcup_{M\in S}\bigcap_{x\in M}[\![x]\!]_\sigma$:

$$z = \sum_{M\in\mathfrak{M}} \mathtt{rhs}(z_M)$$

with fresh variable z.

Example 3. Let $E = [\![r]\!]$ where $r = (a^\star \cdot b + b \cdot b^\star \cdot a)^\star$. Antimirov's algorithm yields the following system of characteristic equations for E:

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} \doteq \begin{pmatrix} 0 & a & b & 0 \\ b & a & 0 & 0 \\ 0 & 0 & b & a \\ 0 & a & b & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

where $r_1 := r$, $r_2 := a^\star \cdot b \cdot r$, $r_3 := b^\star \cdot a \cdot r + r$, $r_4 := a^\star \cdot b \cdot r + r$. Note that $r_1 \doteq r_4$, therefore we can conclude that

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \doteq \begin{pmatrix} 0 & a & b \\ b & a & 0 \\ a & 0 & b \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

is an LS representation of E with solution $\sigma = \{x_1 \mapsto E, x_2 \mapsto [\![r_2]\!], x_3 \mapsto [\![r_3]\!]\}$.

Let $\mathcal{X} := \{x_1, x_2, x_3\}$. To compute LSs for the elements of $\mathtt{RF}(E)$ we proceed as follows:

1. Consider the graph $\mathcal{G}$ with set of nodes $2^{\mathcal{X}}$ and set of edges $\{X \to D_\ell(X) \mid \ell \in \{a,b\}\}$. The only edges starting from singleton nodes are $\{x_1\} \to \{x_2\}$, $\{x_1\} \to \{x_3\}$, $\{x_2\} \to \{x_1\}$, $\{x_2\} \to \{x_2\}$, $\{x_3\} \to \{x_1\}$, $\{x_3\} \to \{x_3\}$. It follows that the set of nodes reachable from $\{x_1\}$ is $\{\{x_1\},\{x_2\},\{x_3\}\}$, thus $D_{\{a,b\}^*}(\{x_1\}) = \{\{x_1\},\{x_2\},\{x_3\}\}$. Therefore

$$\mathtt{RF}(E) = \left\{ \bigcap_{N\in\mathfrak{N}} \bigcup_{x\in N} [\![x]\!]_\sigma \mid \mathfrak{N} \in 2^{\{\{x_1\},\{x_2\},\{x_3\}\}} \right\} = \left\{ \bigcup_{M\in\mathfrak{M}} \bigcap_{x\in M} [\![x]\!]_\sigma \mid \mathfrak{M} \in \mathtt{MS} \right\}$$

where $\mathtt{MS} = \{\{\emptyset\},\{\{x_1\}\},\{\{x_2\}\},\{\{x_3\}\},\{\{x_1,x_2\}\},\{\{x_2,x_3\}\},\{\{x_1,x_3\}\},\{\{x_1,x_2,x_3\}\}\}$.
2. At this stage, we can start computing LS representations of the elements of $\mathtt{RF}(E)$. We illustrate only the computation of an LS for the right factors $\bigcup_{M\in\{\emptyset\}}\bigcap_{x\in M}[\![x]\!]_\sigma$ and $\bigcup_{M\in\{\{x_1,x_2\}\}}\bigcap_{x\in M}[\![x]\!]_\sigma$.
   - The right factor $\bigcup_{M\in\{\emptyset\}}\bigcap_{x\in M}[\![x]\!]_\sigma$ is a degenerate factor, produced by the computation of an empty intersection of languages. By convention, this intersection is the regular language $\{a,b\}^*$. The computation of an LS representation for this factor amounts to intersecting 0 equations. This degenerate case produces the trivial equation $z_\emptyset = 1 + (a+b)\cdot z_\emptyset$. We end up with the following LS for $\{a,b\}^*$

$$\begin{aligned} z &= 1 + (a+b)\cdot z_\emptyset \\ z_\emptyset &= 1 + (a+b)\cdot z_\emptyset. \end{aligned}$$

– The intersection of the equations for x_1 and x_2 produces the equation

$$z_{\{1,2\}} = 0 + a \cdot z_{\{2\}} + b \cdot z_{\{1,3\}}.$$

– The variables introduced by the previous equation are $z_{\{2\}}$ and $z_{\{1,3\}}$. We produce an equation for $z_{\{2\}}$ by intersecting the equation of x_2 with itself, and get

$$z_{\{2\}} = 0 + b \cdot z_{\{1\}} + a \cdot z_{\{2\}}.$$

The equation for $z_{\{1,3\}}$ is

$$z_{\{1,3\}} = 1 + a \cdot z_{\{1,2\}} + b \cdot z_{\{3\}}.$$

We continue producing equations for the newly introduced variables:

$$\begin{aligned} z_{\{1\}} &= 1 + a \cdot z_{\{2\}} + b \cdot z_{\{3\}} \\ z_{\{3\}} &= 1 + a \cdot z_{\{1\}} + b \cdot z_{\{3\}}. \end{aligned}$$

– Finally, we add the initial equation for $\bigcup_{M \in \{\{x_1,x_2\}\}} \bigcap_{x \in M} [\![x]\!]_\sigma$, and obtain the LS representation

$$\begin{aligned} z &= 0 + a \cdot z_{\{2\}} + b \cdot z_{\{1,3\}} \\ z_{\{1\}} &= 1 + a \cdot z_{\{2\}} + b \cdot z_{\{3\}} \\ z_{\{2\}} &= 0 + b \cdot z_{\{1\}} + a \cdot z_{\{2\}} \\ z_{\{3\}} &= 1 + a \cdot z_{\{1\}} + b \cdot z_{\{3\}} \\ z_{\{1,2\}} &= 0 + a \cdot z_{\{2\}} + b \cdot z_{\{1,3\}} \\ z_{\{1,3\}} &= 1 + a \cdot z_{\{1,2\}} + b \cdot z_{\{3\}}. \end{aligned}$$

Computing Right and Left Factors. Given an LS representation of a regular language E, we would like to compute LS representations of all elements of $\mathtt{RF}(E)$ and $\mathtt{LF}(E)$. For $\mathtt{RF}(E)$, we use the algorithm described in this section. For $\mathtt{LF}(E)$, since $\mathtt{LF}(E) = \{R^{\mathtt{r}} \mid R \in \mathtt{RF}(E^{\mathtt{r}})\}$, we proceed as follows:

1. Compute an LS representation of $E^{\mathtt{r}}$ from the LS representation of E.
2. Compute the LS representations of the right factors of $E^{\mathtt{r}}$ with the algorithm of this section.
3. For every LS computed in the previous step, do:
   If the LS represents the language R, compute an LS for the language $R^{\mathtt{r}}$.

3.3 LS Computation for Product Derivative and Antiderivative

We first address the following problem:

Given LS representations of the languages E and F,
Compute an LS representation of the product derivative $F \rhd E$ of E with respect to F.

Suppose the LS representations of E and F are

$$\begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \mathbf{B} + \mathbf{A} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \text{ and } \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \mathbf{D} + \mathbf{C} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix},$$

and let σ and μ be their unique solutions. Let also $\mathcal{X} := \{x_1, \ldots, x_m\}$ and $\mathcal{Y} := \{y_1, \ldots, y_n\}$. It is not hard to see that

$$(F \rhd E) = \bigcap_{w \in F} w^{-1}E = \bigcap_{n \in \mathbb{N}} \bigcap_{\substack{w \in F \\ |w|=n}} w^{-1}E.$$

We consider the ternary relation $y \rhd M \rightsquigarrow N$ with $y \in \{y_1, \ldots, y_n\}$ and $M, N \in 2^{\{x_1,\ldots,x_m\}}$, defined inductively by

$$\frac{\mathbf{D}_{i,1} = 1}{y \rhd M \rightsquigarrow M} \quad \frac{y' \rhd D_a(M) \rightsquigarrow N \quad [y' \in \mathtt{rhs}_a(y)]}{y \rhd M \rightsquigarrow N}.$$

Intuitively, the relation $y \rhd M \rightsquigarrow N$ holds iff there exists $w \in [\![y]\!]_\mu$ such that $N = D_w(M)$. This ternary relation is decidable because it is defined inductively on finite sets.

Lemma 3. *If $y \in \mathcal{Y}$ and $M \subseteq \mathcal{X}$ then* $\left([\![y]\!]_\mu \rhd \bigcup_{x \in M} [\![x]\!]_\sigma \right) = \bigcap_{y \rhd M \rightsquigarrow N} \bigcup_{x \in N} [\![x]\!]_\sigma$.

Proof. For every $p \in \mathbb{N}$ we define the ternary relation

$$y \rhd M \rightsquigarrow_p N :\Leftrightarrow \text{there is a derivation of length } p \text{ of } y \rhd M \rightsquigarrow N$$

and note that

$$\left([\![y]\!]_\mu \rhd \bigcup_{x \in M} [\![x]\!]_\sigma \right) = \bigcap_{p \in \mathbb{N}} \bigcap_{\substack{w \in [\![y]\!]_\mu \\ |w|=p}} w^{-1} \left(\bigcup_{x \in M} [\![x]\!]_\sigma \right),$$

$$\bigcap_{y \rhd M \rightsquigarrow N} \left(\bigcup_{x \in N} [\![x]\!]_\sigma \right) = \bigcap_{p \in \mathbb{N}} \bigcap_{y \rhd M \rightsquigarrow_p N} \left(\bigcup_{x \in N} [\![x]\!]_\sigma \right).$$

Thus it is sufficient to prove that

$$\left\{ w^{-1} \left(\bigcup_{x \in M} [\![x]\!]_\sigma \right) \,\middle|\, w \in [\![y]\!]_\mu \wedge |w| = p \right\} = \left\{ \bigcup_{x \in N} [\![x]\!]_\sigma \,\middle|\, y \rhd M \rightsquigarrow_p N \right\}$$

holds for every $y \in \mathcal{Y}$, $M \subseteq \mathcal{X}$, and $p \in \mathbb{N}$. This fact can be proved easily by induction on p. □

We end up with the following algorithm for the computation of an LS representation of $F \rhd E$:

1. Compute the finite set $\mathfrak{N} := \{N \mid y_1 \rhd \{x_1\} \rightsquigarrow N\}$. By Lemma 3, we have $F \rhd E = [\![y_1]\!]_\mu \rhd [\![x_1]\!]_\sigma = \bigcap_{y_1 \rhd \{x_1\} \rightsquigarrow N} (\bigcup_{x \in N} [\![x]\!]_\sigma) = \bigcap_{N \in \mathfrak{N}} \bigcup_{x \in N} [\![x]\!]_\sigma$.
2. Apply distributivity of intersection over union and identify a set $\mathfrak{M}_0 \subseteq 2^{\mathcal{X}}$ such that $\bigcap_{N \in \mathfrak{N}} \bigcup_{x \in N} [\![x]\!]_\sigma = \bigcup_{M \in \mathfrak{M}_0} \bigcap_{x \in M} [\![x]\!]_\sigma$.
3. Compute $\mathfrak{M} := \min_{\subseteq}(\mathfrak{M}_0)$ and note that $F \rhd E = \bigcup_{M \in \mathfrak{M}} \bigcap_{x \in M} [\![x]\!]_\sigma$.
4. Compute an LS representation of $\bigcup_{M \in \mathfrak{M}} \bigcap_{x \in M} [\![x]\!]_\sigma$ as shown in Sect. 3.2.

Computing Product Derivatives and Antiderivatives: Given LS representations of the languages E and F, we would like to compute an LS representation of $F \rhd E$ and $E \lhd F$. For $F \rhd E$, we use the algorithm described in this section. For $E \lhd F$, since $E \lhd F = (F^{\mathtt{r}} \rhd E^{\mathtt{r}})^{\mathtt{r}}$, we proceed as follows:

1. Compute LS representations for $F^{\mathtt{r}}$ and $E^{\mathtt{r}}$.
2. Compute an LS representation for $F^{\mathtt{r}} \rhd E^{\mathtt{r}}$ by the algorithm presented in this section.
3. Compute an LS representation for $(F^{\mathtt{r}} \rhd E^{\mathtt{r}})^{\mathtt{r}}$.

3.4 LS Computations for the Factor Matrix

If E is a regular language with $\mathtt{LF}(E) = \{L_1, \ldots, L_n\}$ then the factor matrix of E is the $n \times n$ matrix $\mathbf{F}$ whose element at position (i, j) is $\mathbf{F}_{i,j} := L_i \rhd L_j$.

Given an LS representation of E, we can compute LS representations of the elements of the factors matrix of E as follows:

1. Compute the LS representations of the elements of $\mathtt{LF}(E)$, as indicated at the end of Sect. 3.2.
2. Compute LS representations of $L_i \rhd L_j$ for all $i, j \in \{1, \ldots, n\}$.

4 A Regular Language Reconstruction Algorithm

In this section we consider the problem mentioned in Sect. 2.1:

Given $L \in \mathbf{Reg}(\mathcal{A})$ and $s \in \mathcal{T}(\mathcal{A}, \{x_1, \ldots, x_k\})$,
Find all $<$-maximal tuples $(L_1, \ldots, L_k)$ of languages over $\mathcal{A}$ such that $[\![s]\!]_\sigma \subseteq L$ holds for the assignment $\sigma = \{x_1 \mapsto L_1, \ldots, x_k \mapsto L_k\}$.

We show that the solutions of this problem are made of regular languages, and they are finitely many. Moreover, we propose an algorithm that finds all solutions of this problem by exploiting the properties of the factor matrix of L. Let's denote this problem by $s \ll L$ and let

$$Sol(s \ll L) := \{(L_1, \ldots, L_k) \mid (L_1, \ldots, L_k) \text{ is solution of } s \ll L\}.$$

Our algorithm for solving $s \ll L$ proceeds in 3 stages.

In stage 1 we compute the factor matrix $\mathbf{F}$ of L, and the constants l, r characterized by property (f3). Suppose $\mathbf{F}$ is a matrix of dimension $n \times n$.

In stage 2 we make use of a function

$$A : \mathbf{Reg}(\{x_1, \ldots, x_k\}) \times \mathtt{Facts}(L) \to 2^{(\mathcal{A}^\star)^k}$$

such that $Sol(s \ll \mathbf{F}_{i,j}) \subseteq A(s, \mathbf{F}_{i,j})$. The function A is defined as follows:

1. $A(0, \mathbf{F}_{i,j}) := \{(\underbrace{\mathcal{A}^*, \ldots, \mathcal{A}^*}_{k \text{ times}})\}$.
2. $A(1, \mathbf{F}_{i,j}) := \begin{cases} A(0, \mathbf{F}_{i,j}) & \text{if } \epsilon \in \mathbf{F}_{i,j}, \\ \emptyset & \text{otherwise.} \end{cases}$
3. $A(x_p, \mathbf{F}_{i,j}) := \{(P_1, \ldots, P_k)\}$ where $P_l := \mathcal{A}^*$ if $l \neq p$ and $P_p := \mathbf{F}_{i,j}$.
4. $A(s_1 + s_2, \mathbf{F}_{i,j}) := A(s_1, \mathbf{F}_{i,j}) \cup A(s_2, \mathbf{F}_{i,j})$.
5. $A(s_1 \cdot s_2, \mathbf{F}_{i,j}) := \bigcup_{p=1}^{n} (A(s_1, \mathbf{F}_{i,p}) \sqcap A(s_2, \mathbf{F}_{p,j}))$.
6. $A(s^\star, \mathbf{F}_{i,j}) := A(1, \mathbf{F}_{i,j}) \sqcap A(s, \mathbf{F}_{i,j}) \sqcap$
   $\left(\bigcup_{p=1}^{n} (A(s, \mathbf{F}_{i,p}) \sqcap A(s, \mathbf{F}_{p,p}) \sqcap A(s, \mathbf{F}_{p,j}))\right)$

where $E \sqcap F := \{(E_1 \cap F_1, \ldots, E_k \cap F_k) \mid (E_1, \ldots, E_k) \in E, (F_1, \ldots, F_k) \in F\}$.

Note that $A(s, \mathbf{F}_{i,j})$ is defined recursively on the structure of s and that it yields a set of tuples made of finite intersections of $\mathcal{A}^*$ and factors of $\mathbf{F}_{i,j}$. Since regular languages are closed under finite intersection, it follows that A yields finite sets of k-tuples of regular languages.

The fact that $Sol(s \ll \mathbf{F}_{i,j}) \subseteq A(s, \mathbf{F}_{i,j})$ can be proved easily by induction on the structure of s and using properties (f1)–(f3) of the factor matrix. Here we explain only the proof case $Sol(s^\star \ll \mathbf{F}_{i,j}) \subseteq A(s^\star, \mathbf{F}_{i,j})$, which is the most interesting. Suppose $\mathbf{L} = (L_1, \ldots, L_k) \in Sol(s^\star \ll \mathbf{F}_{i,j})$ and let $\sigma := \{x_1 \mapsto L_1, \ldots, x_k \mapsto L_k\}$ and $F := [\![s]\!]_\sigma$. Then $F^* \subseteq \mathbf{F}_{i,j}$, therefore $\{\epsilon\} \subseteq \mathbf{F}_{i,j}$, $F \subseteq \mathbf{F}_{i,j}$, and $F^* = (F^*)^n \subseteq \mathbf{F}_{i,j}$. ¿From $F \subseteq \mathbf{F}_{i,j}$ we learn the existence of $\mathbf{L}_1 \in Sol(s \leq \mathbf{F}_{i,j})$ with $\mathbf{L} \leq \mathbf{L}_1$. From $F^* = (F^*)^n \subseteq \mathbf{F}_{i,j}$ we learn by (f3) that there exist $u_1, \ldots, u_{n+1} \in \{1, \ldots, n\}$ such that $u_1 = i, u_{n+1} = j$, and $F^* \subseteq \mathbf{F}_{u_i, u_{i+1}}$ for all $i \in \{1, \ldots, n\}$. Since $\{u_1, \ldots, u_{n+1}\} \subseteq \{1, \ldots, n\}$, there exist $1 \leq l < m \leq n+1$ and $1 \leq p \leq n$ such that $u_l = u_m = p$. Thus

$$[\![s]\!]_\sigma = F \subseteq (F^*)^{m-1} \subseteq \mathbf{F}_{u_1,u_2} \cdots \mathbf{F}_{u_{m-1},u_m} \subseteq \mathbf{F}_{i,p},$$
$$[\![s]\!]_\sigma = F \subseteq (F^*)^{m-l} \subseteq \mathbf{F}_{u_l,u_{l+1}} \cdots \mathbf{F}_{u_{m-1},u_m} \subseteq \mathbf{F}_{p,p},$$
$$[\![s]\!]_\sigma = F \subseteq (F^*)^{n-l} \subseteq \mathbf{F}_{u_l,u_{l+1}} \cdots \mathbf{F}_{u_n,u_{n+1}} \subseteq \mathbf{F}_{p,j},$$

hence there exist $\mathbf{L}_2 \in Sol(s \ll \mathbf{F}_{i,p})$, $\mathbf{L}_3 \in Sol(s \ll \mathbf{F}_{p,p})$, and $\mathbf{L}_4 \in Sol(s \ll \mathbf{F}_{p,j})$ such that $\mathbf{L} \leq \mathbf{L}_v$ for $2 \leq v \leq 4$. Let $\mathbf{L}' = (L'_1, \ldots, L'_k) := \mathbf{L}_1 \sqcap \mathbf{L}_2 \sqcap \mathbf{L}_3 \sqcap \mathbf{L}_4$ and $\sigma' := \{x_1 \mapsto L'_1, \ldots, x_k \mapsto L'_k\}$. Then $\mathbf{L} \leq \mathbf{L}'$ because $\mathbf{L} \leq \mathbf{L}_v$ for $1 \leq v \leq 4$. Also $[\![s^\star]\!]_{\sigma'} \subseteq \{\epsilon\} \cup [\![s]\!]_{\sigma'} \cup [\![s]\!]_{\sigma'} [\![s]\!]^*_{\sigma'} [\![s]\!]_{\sigma'} \subseteq \{\epsilon\} \cup \mathbf{F}_{i,j} \cup \mathbf{F}_{i,p} \mathbf{F}^*_{p,p} \mathbf{F}_{p,j} \subseteq \mathbf{F}_{i,j}$ where the last inclusion follows from (f2) and the observation that $\{\epsilon\} \subseteq \mathbf{F}_{i,j}$. Since $\mathbf{L}$ is $<$-maximal such that $[\![s^\star]\!]_\sigma \subseteq \mathbf{F}_{i,j}$, we learn that $\sigma = \sigma'$, hence $\mathbf{L} = \mathbf{L}'$.

By induction hypothesis, we have $\mathbf{L}_1 \in A(s, \mathbf{F}_{i,j})$, $\mathbf{L}_2 \in A(s, \mathbf{F}_{i,p})$, $\mathbf{L}_3 \in A(s, \mathbf{F}_{p,p})$, and $\mathbf{L}_4 \in A(s, \mathbf{F}_{p,j})$. Since $\epsilon \in \mathbf{F}_{i,j}$ we have $\mathbf{L}_0 := (\underbrace{\mathcal{A}^*, \ldots, \mathcal{A}^*}_{k \text{ times}}) \in$

$A(1, \mathbf{F}_{i,j})$ and therefore $\mathbf{L} = \mathbf{L}' = \mathbf{L}_0 \sqcap \mathbf{L}_1 \sqcap \mathbf{L}_2 \sqcap \mathbf{L}_3 \sqcap \mathbf{L}_4$. It follows that $\mathbf{L} \in A(1, \mathbf{F}_{i,j}) \sqcap A(s, \mathbf{F}_{i,j}) \sqcap A(s, \mathbf{F}_{i,p}) \sqcap A(s, \mathbf{F}_{p,p}) \sqcap A(s, \mathbf{F}_{p,j}) \subseteq A(s^\star, \mathbf{F}_{i,j})$.

In stage 3 we filter out the elements of $A(s, \mathbf{F}_{l,r})$ which are not $<$-maximal. Since $Sol(s \ll \mathbf{F}_{l,r}) \subseteq A(s, \mathbf{F}_{l,r})$ and $Sol(s \ll \mathbf{F}_{l,r})$ consists of $<$-maximal elements, the outcome of this step is $Sol(s \ll \mathbf{F}_{l,r})$.

5 Conclusion

We proposed linear systems of equations as a new representation of regular languages. The main advantage of this representation is that it can be used for various operations over regular languages without converting to other representations (e.g. automata, grammars, or regular expressions). We demonstrated this advantage by designing algorithms for computing language reversal, left and right quotients, left and right factors, product derivatives and antiderivatives, and factor matrix. All these algorithms work on LS representations and produce LS representations. We also demonstrated an application in the language reconstruction problem.

Acknowledgments

Mircea Marin was supported by JSPS Grant-in-Aid no. 20500025 for Scientific Research (C). Temur Kutsia was supported by the European Commission Framework 6 Programme for Integrated Infrastructures Initiatives under the project SCIEnce—Symbolic Computation Infrastructure for Europe (Contract No. 026133).

References

1. V. M. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoretical Computer Science*, 155:291–319, 1996.
2. J. Berstel. *Transductions and Context-Free Languages*. B.G. Teubner Stuttgart, 1979.
3. J. A. Brzozowski. Derivatives of regular expressions. *Journal of the Association for Computing Machinery*, 11(4):481–494, October 1964.
4. J. H. Conway. *Regular Algebra and Finite Machines*. Mathematics series. Chapman and Hall, 1971.
5. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Series in Computer Science. Addison-Wesley Publishing Company, Inc., 1979.
6. D. C. Kozen. *Automata and Computability*. Undergraduate Texts in Computer Science. Springer-Verlag New York, Inc., 1997.
7. M. Marin and T. Kutsia. On the computation of quotients and factors of regular languages. In Z. Hu and J. Zhang, editors, *Proceedings of the Sixth Asian Workshop on Foundations of Software*, GRACE Technical Reports, pages 67–78, Tokyo, Japan, 2009. National Institute of Informatics. GRACE TR 2009-01.

8. A. Mateescu, A. Salomaa, and S. Yu. On the decomposition of finite languages. In G. Rozenberg and W. Thomas, editors, *Developments in Language Theory: Foundations, Applications, Perspectives*, pages 22–31. World Scientific, 2000.
9. A. Mateescu, A. Salomaa, and S. Yu. Factorizations of languages and commutativity conditions. *Acta Cybernetica*, 15(3):339–351, 2002.
10. T. Suzuki and S. Okui. Product derivatives of regular expressions. *ISPJ Online Transactions*, 1:53–65, July 2008.