Fundamenta Informaticae 69 (2006) 301–318
IOS Press

Regular Expressions for Languages over Infinite Alphabets

Michael Kaminski*
Department of Computer Science
Technion – Israel Institute of Technology
Haifa 32000, Israel
kaminski@cs.technion.ac.il

Tony Tan
Department of Computer Science
National University of Singapore
3 Science Drive 2
Singapore 117543

Abstract. In this paper we introduce a notion of a *regular expression* over *infinite alphabets* and show that a language is definable by an infinite alphabet regular expression if and only if it is accepted by *finite-state unification based automaton* – a model of computation that is tightly related to other models of automata over infinite alphabets.

Keywords: Finite state automata, infinite alphabets, regular expressions

1. Introduction

A new model of finite-state automata dealing with *infinite alphabets*, called *finite-state datalog automata* (FSDA) was introduced in [16]. These automata were intended for the abstract study of relational languages. Since the character of relational languages requires the use of infinite alphabets of names of variables, in addition to a finite set of states, FSDA are equipped with a finite set of "registers" capable of retaining a variable name (out of an infinite set of names). The equality test, which is performed in

* Address for correspondence: Department of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel

ordinary finite-state automata (FA) was replaced with *unification*, which is a crucial element of relational languages.

Later, FSDA were extended in [7] to a more general model dealing with infinite alphabets, called *finite-memory automata* (FMA). FMA were designed to accept the infinite alphabet counterpart of the ordinary regular languages. Similarly to FSDA, FMA are equipped with a finite set of registers which are either empty or contain a symbol from the infinite alphabet, but contrary to FSDA, registers in FMA cannot contain symbols currently stored in other registers. By restricting the power of the automaton to copying a symbol to a register and comparing the content of a register with an input symbol only, without the ability to perform *any* functions, the automaton is only able to "remember" a finite set of input symbols. Thus, the languages accepted by FMA possess many of the properties of regular languages.

Whereas decision of the emptiness and containment for FMA- (and, consequently, for FSDA-) languages is relatively simple, the problem of inclusion for FMA-languages is undecidable, see [11, 12].

An extension of FSDA to a general infinite alphabet called *finite-state unification based automata*, (FSUBA) was proposed in [17]. These automata are similar in many ways to FMA, but are a bit weaker, because a register of FSUBA may contain a symbol currently stored in other registers. It was shown in [17] that FSDA can be simulated by FSUBA and that the problem of inclusion for FSUBA languages is decidable.

While the study of finite automata over infinite alphabets started as purely theoretical, since the appearance of [7] and [8] it seems to have turned to more practically oriented. The key idea for the applicability is finding practical interpretations to the infinite alphabet and to the languages over it.

- In [11, 12], members of (the infinite) alphabet Σ are interpreted as records of *communication actions*, "send" and "receive" of messages during inter-process-communication. Words in a language L over this alphabet are MSCs, *message sequence charts*, capturing behaviors of the communication network.
- In [4], members of Σ are interpreted as URLs' addresses of internet sites, a word in L is interpreted as a "navigation path" in the internet, the result of some finite sequence of clicks.
- In [5] there is another internet-oriented interpretation of Σ, namely, XML mark-ups of pages in a site.

In this paper we introduce a notion of a *regular expression* for languages over infinite alphabets and show that a language is definable by an infinite alphabet regular expression if and only if it is accepted by an FSUBA.

The paper is organized as follows. In the next section we recall the definition of FSDA from [16] and in Section 3 we recall the definition of FSUBA from [17]. In Section 4 we present the main result of our paper – *unification based regular expressions* for languages over infinite alphabets, whose equivalence to FSUBA is proven in Sections 5 and 6. Section 7 contains the proof of a modification of a technical lemma from [17]. Finally, Section 8 deals with the complexity of intertranslations between FSUBA and unification based regular expressions.

2. Finite-state datalog automata

We start with examples of *relational* languages which can and cannot be defined by finite-state datalog automata (FSDA). Relational languages are languages over infinite alphabets whose symbols are of the

form $p(x_1, x_2)$, where p belongs to a finite alphabet of binary relation symbols and x_1 and x_2 come from an *infinite* alphabet of variables. For example, the relational language

$$\{p_1(x_1, x_2)p_2(x_2, x_3)p_1(x_3, x_4) \cdots p_2(x_{2n}, x_{2n+1}) : n \geq 1\}$$

generated by the Horn grammar

$$P(x, y) \to p_1(x, z)p_2(z, w)P(w, y)|p_1(x, z)p_2(z, y)$$

is definable by finite-state datalog automata, see [16, Section 2], whereas the relational language

$$\{p(x_1, x_2)p(x_2, x_3) \cdots p(x_{n-1}, x_n) \cdots p(x_n, x_{n-1}) \cdots p(x_3, x_2)p(x_2, x_1) : n \geq 2\}$$

generated by the Horn grammar

$$P(x) \to p(x, y)P(y)p(y, x)|p(x, y)p(y, x)$$

is not, because the restrictions of FSDA-languages to finite alphabets are regular, see [8, Proposition 1].

Next we recall the definition of FSDA from [16].

A *finite-state datalog automaton* or, shortly, FSDA, is a system $\boldsymbol{A} = (R, V, Q, q_0, F, r, \mu)$, where

- R and V are a finite alphabet of *binary relation symbols* and an infinite alphabet of *variables*, respectively, $R \cap V = \emptyset$ and $\# \notin V$,[1] whereas the input alphabet of $\boldsymbol{A}$ is $R \times V^2$. That is, an input symbol is a relation $p(x_1, x_2)$, where $p \in R$ is a binary relation symbol and $x_1, x_2 \in V$ are variables.
- Q, $q_0 \in Q$, and $F \subseteq Q$ are a finite set of states, the initial state, and the set of final states, respectively.
- r is the number of registers of $\boldsymbol{A}$, which are capable of either being empty or retaining a variable from V.
- $\mu \subseteq Q \times R \times \{1, 2, \ldots, r\} \times \{1, 2, \ldots, r\} \times 2^{\{1,2,\ldots,r\}} \times Q$ is the transition relation whose elements are called *transitions*. The intuitive meaning of the transition relation is as follows. If the automaton is in state q reading relation $p(x_1, x_2)$ and there is a transition $(q, p, k_1, k_2, S, q') \in \mu$ such that the register k_i either contains x_i or is empty, $i = 1, 2$, then the automaton can enter state q', copy x_i into the k_ith register, if the latter is empty, and empty (reset) the registers whose indices belong to S. The above registers k_1 and k_2 are referred to as the *transition* registers.

An actual state of an FSDA $\boldsymbol{A}$ is an element of Q together with the contents of all registers of the automaton. Thus, $\boldsymbol{A}$ has infinitely many states[2] which are pairs $(q, \boldsymbol{w})$, where $q \in Q$ and $\boldsymbol{w} \in (\Sigma \cup \{\#\})^r$. Such pairs are called *configurations* of $\boldsymbol{A}$ and are denoted Q^c. The pair $(q_0, \#^r)$, denoted q_0^c, is the *initial* configuration, and the configurations with the first component in F are called *final* configurations. The set of final configurations is denoted F^c.

The transition relation μ induces the following relation μ^c on $Q^c \times R \times V \times V \times Q^c$. Let $q, q' \in Q$ and $\boldsymbol{w} = w_1 w_2 \cdots w_r$, $\boldsymbol{w}' = w'_1 w'_2 \cdots w'_r \in (V \cup \{\#\})^r$. Then $((q, \boldsymbol{w}), p(x_1, x_2), (q', \boldsymbol{w}')) \in \mu^c$ if and only if there is a transition $(q, p, k_1, k_2, S, q') \in \mu$ such that the following four conditions are satisfied.

[1] In this paper we reserve $\#$ to denote an empty register.

[2] This is the major difference between ordinary finite-state automata and finite-state automata over infinite alphabets.

1. $w_{k_i} \in \{x_i, \#\}$, $i = 1, 2$. That is, the transition register k_i either contains x_i or is empty.
2. If $k_i \notin S$, then $w'_{k_i} = x_i$, $i = 1, 2$. That is, if the transition register k_i is not reset in the transition, its content is x_i.
3. For all $j \in S$, $w'_j = \#$.
4. For all $j \notin S \cup \{k_1, k_2\}$, $w'_j = w_j$.

Let $\boldsymbol{r} = p_1(x_1, x_2)p_2(x_3, x_4) \cdots p_n(x_{2n-1}, x_{2n})$ be a word over $R \times V^2$. A run of the automaton $\boldsymbol{A}$ on $\boldsymbol{r}$ consists of a sequence of configurations $c_0, c_1, \ldots, c_n$ such that c_0 is the initial configuration q_0^c, and $(c_{i-1}, p_i(x_{2i-1}, x_{2i}), c_i) \in \mu^c$, $i = 1, 2, \ldots, n$.

We say that $\boldsymbol{A}$ *accepts* $\boldsymbol{r}$ if there exists a run $c_0, c_1, \ldots, c_n$ of $\boldsymbol{A}$ on $\boldsymbol{r}$ such that $c_n \in F^c$. The set of all words accepted by $\boldsymbol{A}$ is denoted by $L(\boldsymbol{A})$ and is referred to as an FSDA-language. We refer the reader to [16] for additional examples of FSDA-languages and their relation to DATALOG.

3. Finite-state unification based automata

Till the end of this paper Σ is an infinite alphabet not containing $\#$. For a word $\boldsymbol{w} = w_1 w_2 \cdots w_r$ over $\Sigma \cup \{\#\}$, we define the *content* of $\boldsymbol{w}$, denoted $[\boldsymbol{w}]$, by $[\boldsymbol{w}] = \{w_j \neq \# : j = 1, 2, \ldots, r\}$. That is, $[\boldsymbol{w}]$ consists of all symbols of Σ which appear in $\boldsymbol{w}$.

Definition 3.1. ([17]) A *finite-state unification based automaton* (over Σ) or, shortly, FSUBA, is a system $\boldsymbol{A} = (\Sigma, Q, q_0, F, \boldsymbol{u}, \Theta, \mu)$, where

- Q, $q_0 \in Q$, and $F \subseteq Q$ are a finite set of states, the initial state, and the set of final states, respectively.
- $\boldsymbol{u} = u_1 u_2 \cdots u_r \in (\Sigma \cup \{\#\})^r$, $r \geq 1$, is the *initial assignment* – register initialization: the symbol in the ith register is u_i. Recall that $\#$ is reserved to denote an empty register. That is, if $u_j = \#$, then the jth register is empty.
- $\Theta \subseteq [\boldsymbol{u}]$ is the "read only" alphabet whose symbols cannot be copied into empty registers.[3] One may think of Θ as a set of the language constants which cannot be unified, cf [16].
- $\mu \subseteq Q \times \{1, 2, \ldots, r\} \times 2^{\{1,2,\ldots,r\}} \times Q$ is the transition relation whose elements are called transitions. The intuitive meaning of μ is as follows. If the automaton is in state q reading symbol σ and there is a transition $(q, k, S, q') \in \mu$ such that the kth register either contains σ or is empty, then the automaton can enter state q', write σ in the kth register (if it is empty), and erase the content of the registers whose indices belong to S. The kth register will be referred to as the *transition* register.

Like in the case of FSDA, an actual state of $\boldsymbol{A}$ is an element of Q together with the contents of all registers of the automaton. That is, $\boldsymbol{A}$ has infinitely many states which are pairs $(q, \boldsymbol{w})$, where $q \in Q$ and $\boldsymbol{w} \in (\Sigma \cup \{\#\})^r$. These pairs are called *configurations* of $\boldsymbol{A}$. The set of all configurations of $\boldsymbol{A}$ is

[3] Of course, we could let Θ be *any* subset of Σ. However, since the elements of Θ cannot be copied into empty registers, the automaton can make a move with the input from Θ only if the symbol already appears in one of the automaton registers, i.e., belongs to the initial assignment.

denoted Q^c. The pair $(q_0, \boldsymbol{u})$, denoted q_0^c, is called the *initial* configuration,[4] and the configurations with the first component in F are called *final* configurations. The set of final configurations is denoted F^c.

Transition relation μ induces the following relation μ^c on $Q^c \times \Sigma \times Q^c$.

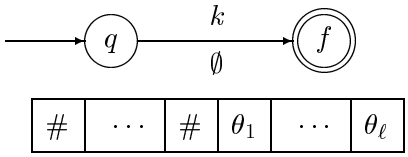
Let $q, q' \in Q$, $\boldsymbol{w} = w_1 w_2 \cdots w_r$ and $\boldsymbol{w}' = w'_1 w'_2 \cdots w'_r$. Then the triple $((q, \boldsymbol{w}), \sigma, (q', \boldsymbol{w}'))$ belongs to μ^c if and only if there is a transition (q, k, S, q') in μ such that the following conditions are satisfied.

- Either $w_k = \#$ (i.e., the transition register is empty in which case σ is copied into it) and $\sigma \notin \Theta$, or $w_k = \sigma$ (i.e., the transition register contains σ).
- If $k \notin S$, then $w'_k = \sigma$, i.e., if the transition register is not reset in the transition, its content is σ.
- For all $j \in S$, $w'_j = \#$.
- For all $j \notin S \cup \{k\}$, $w'_j = w_j$.

Let $\boldsymbol{\sigma} = \sigma_1 \sigma_2 \cdots \sigma_n$ be a word over Σ. A *run* of $\boldsymbol{A}$ on $\boldsymbol{\sigma}$ consists of a sequence of configurations $c_0, c_1, \ldots, c_n$ such that c_0 is the initial configuration q_0^c and $(c_{i-1}, \sigma_i, c_i) \in \mu^c$, $i = 1, 2, \ldots, n$.

We say that $\boldsymbol{A}$ *accepts* $\boldsymbol{\sigma}$, if there exists a run $c_0, c_1, \ldots, c_n$ of $\boldsymbol{A}$ on $\boldsymbol{\sigma}$ such that $c_n \in F^c$. The set of all words accepted by $\boldsymbol{A}$ is denoted by $L(\boldsymbol{A})$ and is referred to as an FSUBA-language.

Example 3.1. Let $\boldsymbol{A} = (\Sigma, \{q, f\}, q, \{f\}, \#^r \theta_1 \cdots \theta_\ell, \{\theta_1, \ldots, \theta_\ell\}, \mu)$ be an $(r + \ell)$-register FSUBA, where μ consists of the only one transition $(q, k, \emptyset, f)$. Alternatively, $\boldsymbol{A}$ can be described by the following diagram.



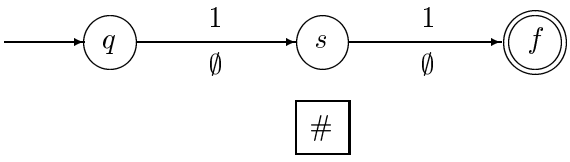
initialization

Obviously, $L(\boldsymbol{A}) = \Sigma \setminus \Theta$, if $k \leq r$, and $L(\boldsymbol{A}) = \{\theta_{k-r}\}$, otherwise.

Example 3.2. ([17]) Let $\boldsymbol{A} = (\Sigma, \{q, s, f\}, q, \{f\}, \#, \emptyset, \mu)$ be a one-register FSUBA, where μ consists of the following two transitions:

- $(q, 1, \emptyset, s)$
- $(s, 1, \emptyset, f)$,

see the diagram below.



initialization

[4] Recall that q_0 and $\boldsymbol{u}$ denote the initial state and the initial assignment, respectively.

Then $L(\boldsymbol{A}) = \{\sigma_1\sigma_2 \in \Sigma^2 : \sigma_1 = \sigma_2\}$: an accepting run of $\boldsymbol{A}$ on $\sigma\sigma$ is $(q, \#), (s, \sigma), (f, \sigma)$.

In contrast, the language $L = \{\sigma_1\sigma_2 \in \Sigma^2 : \sigma_1 \neq \sigma_2\}$ is not an FSUBA language.[5] To prove that, assume to the contrary that for some FSUBA $\boldsymbol{A} = (\Sigma, Q, q_0, F, \boldsymbol{u}, \Theta, \mu)$, $L = L(\boldsymbol{A})$. Since Σ is infinite and $[\boldsymbol{u}]$ is finite, $\Sigma\backslash[\boldsymbol{u}]$ contains two different symbols σ_1 and σ_2. By the definition of L, it contains the word $\sigma_1\sigma_2$. Let $(q_0, \boldsymbol{u}), (q_1, \boldsymbol{w}_1), (q_2, \boldsymbol{w}_2)$, $\boldsymbol{w}_i = w_{i,1}w_{i,2}\cdots w_{i,r}$, $i = 1, 2$, be an accepting run of $\boldsymbol{A}$ on $\sigma_1\sigma_2$ and let k be the transition register between configurations $(q_1, \boldsymbol{w}_1)$ and $(q_2, \boldsymbol{w}_2)$. Since neither of σ_1 and σ_2 belongs to $\boldsymbol{u}$ and $\sigma_1 \neq \sigma_2$, $w_{1,k} = \#$ and $w_{2,k} = \sigma_2$. Then, replacing $w_{2,k}$ with σ_1 in $(q_0, \boldsymbol{u}), (q_1, \boldsymbol{w}_1), (q_2, \boldsymbol{w}_2)$ we obtain an accepting run of $\boldsymbol{A}$ on $\sigma_1\sigma_1$, which contradicts $L = L(\boldsymbol{A})$.[6]

The following example shows how FSDA can be simulated by FSUBA.

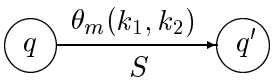
Example 3.3. ([17]) Let $\boldsymbol{A} = (R, V, Q, q_0, F, r, \mu)$, $R = \{\theta_1, \theta_2, \ldots, \theta_k\}$, be an FSDA. Consider an FSUBA $\boldsymbol{A}' = (\Sigma, Q', q_0', F', \boldsymbol{u}, \Theta, \mu')$, such that

- $\Sigma = V \cup R$,
- $Q' = Q \cup (\mu \times \{arg_1, arg_2\})$,
- $q_0' = q_0$,
- $F' = F$,
- $\boldsymbol{u} = \#^r\theta_1\theta_2\cdots\theta_k$,
- $\Theta = R$, and
- μ consists of all transitions of the form 1, 2, or 3 below

  1. $(q, r + m, \emptyset, ((q, \theta_m, k_1, k_2, S, q'), arg_1))$,
  2. $(((q, \theta_m, k_1, k_2, S, q'), arg_1), k_1, \emptyset, ((q, \theta_m, k_1, k_2, S, q'), arg_2))$, or
  3. $(((q, \theta_m, k_1, k_2, S, q'), arg_2), k_2, S, q')$,

  where $(q, \theta_m, k_1, k_2, S, q') \in \mu$.

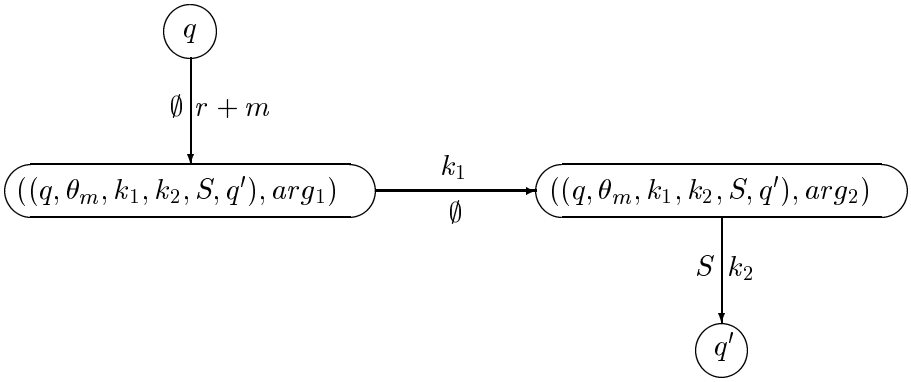
That is, we break each transition



of μ into three "consecutive" transitions

[5] It can be readily seen that L is accepted by a *finite-memory* automaton introduced in [7].

[6] The decision procedure for the inclusion of FSUBA-languages in [17] is based on a refined version of this argument.



of μ'.

A straightforward induction on the word length shows that

$$p_1(x_1, x_2)p_2(x_3, x_4)\cdots p_n(x_{2n-1}, x_{2n}) \in L(\boldsymbol{A})$$

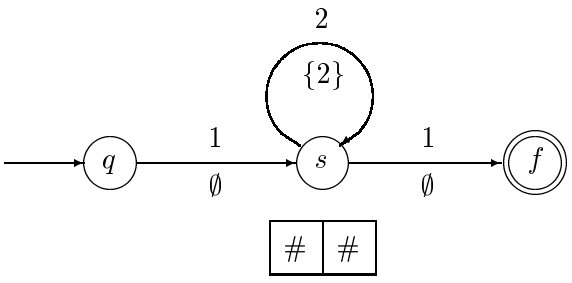
if and only if

$$p_1x_1x_2p_2x_3x_4\cdots p_nx_{2n-1}x_{2n} \in L(\boldsymbol{A}').$$

Example 3.4. Let $\boldsymbol{A} = (\Sigma, \{q, s, f\}, q, \{f\}, \#\#, \emptyset, \mu)$ be a 2-register FSUBA, where μ consists of the following three transitions:

- $(q, 1, \emptyset, s)$,
- $(s, 2, \{2\}, s)$,
- $(s, 1, \emptyset, f)$,

see the diagram below.



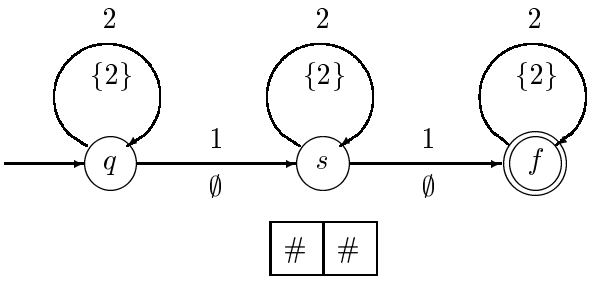
initialization

It can be easily seen that $L(\boldsymbol{A}) = \{\sigma_1\sigma_2\cdots\sigma_n \in \Sigma^* : \sigma_1 = \sigma_n\}$.

Example 3.5. ([17], cf [8, Example 1].) Let $\boldsymbol{A} = (\Sigma, \{q, s, f\}, q, \{f\}, \#\#, \emptyset, \mu)$ be an FSUBA with two registers and μ consists of the following five transitions:

- $(q, 2, \{2\}, q)$,
- $(q, 1, \emptyset, s)$,
- $(s, 2, \{2\}, s)$,
- $(s, 1, \emptyset, f)$, and
- $(f, 2, \{2\}, f)$,

see the diagram below.



initialization

It can be easily seen that

$$L(\boldsymbol{A}) = \{\sigma_1\sigma_2\cdots\sigma_n \in \Sigma^* : \text{ there exist } 1 \leq i < i' \leq n \text{ such that } \sigma_i = \sigma_{i'}\}.$$

That is, $L(\boldsymbol{A})$ consists of all words over Σ in which some symbol appears twice or more. For example, an accepting run of $\boldsymbol{A}$ on $abcbd$ is

$$(q, \#\#), (q, \#\#), (s, b\#), (s, b\#), (f, b\#), (f, b\#).$$

Example 3.6. ([17]) Let $\boldsymbol{A} = (\Sigma, Q, q_0, F, \boldsymbol{u}, \Theta, \mu)$ be an FSUBA such that $\#$ does not appear in $\boldsymbol{u}$ and for all $(q, k, S, q') \in \mu$, $S = \emptyset$. Then $L(\boldsymbol{A})$ is a regular language over $[\boldsymbol{u}]$. In general, since the restriction of a set of configurations to a finite alphabet is finite, the restrictions of FSUBA-languages to finite alphabets are regular, cf. [8, Proposition 1].

4. Regular expressions for FSUBA languages

In this section we introduce an alternative description of FSUBA languages by the so called *unification based* expressions which are the infinite alphabet counterpart of the ordinary regular expressions.

Definition 4.1. Let $X = \{x_1, \ldots, x_r\}$ be a set of variables such that $X \cap \Sigma = \emptyset$ and let Θ be a finite subset of Σ. *Unification based regular expressions over* (X, Θ), or shortly UB-expressions, if (X, Θ) is understood from the context, are defined as follows.

- $\emptyset, \epsilon$, and each element of $X \cup \Theta$ are UB-expressions.
- If α_1 and α_2 are UB-expressions, then so is $(\alpha_1 + \alpha_2)$.

- If $X' \subseteq X$ and α_1 and α_2 are UB-expressions, then so are $(\alpha_1 \cdot_{X'} \alpha_2)$ and $(\alpha_1^{*_{X'}})$.

The intuition behind the above definition is as follows. Each variable in X corresponds to a register of the automaton and a "variable" assignment of symbols from $\Sigma \setminus \Theta$ to variables in X is the register assignment. Finally, subscripts X' indicate the set of registers reset by the automaton.

The definition of languages defined by UB-expressions is based on the observation that the set of all sequences of an r-register FSUBA diagram labels corresponding to its accepting runs is a regular language over $\{1, 2, \ldots, r\} \times 2^{\{1,2,\ldots,r\}}$. Thus, with a unification based regular expressions α over (X, Θ) we associate an ordinary regular expression over (finite) alphabet $X \cup \Theta \cup 2^X$, denoted $\underline{\alpha}$, that is defined by induction as follows.

- If $\alpha \in \{\emptyset, \epsilon\} \cup X \cup \Theta$, then $\underline{\alpha}$ is α.
- $\underline{(\alpha_1 + \alpha_2)}$ is $(\underline{\alpha_1} + \underline{\alpha_2})$.
- $\underline{(\alpha_1 \cdot_{X'} \alpha_2)}$ is $((\underline{\alpha_1} \cdot X') \cdot \underline{\alpha_2})$.
- Finally, $\underline{(\alpha^{*_{X'}})}$ is $((\underline{\alpha} \cdot X')^*)$.

Let $\boldsymbol{w} = w_1 w_2 \cdots w_n \in (X \cup \Theta \cup 2^X)^*$. With the ith symbol w_i of $\boldsymbol{w}$, $i = 1, 2, \ldots, n$, we associate a word $\overline{w_i} \in \Sigma \cup \{\epsilon\}$ as described below, cf. [7, Definition 3].

- If $w_i \in \Theta$, then $\overline{w_i} = w_i$.
- If $w_i = X' \subseteq X$, then $\overline{w_i} = \epsilon$.
- If $w_i = x \in X$, then $\overline{w_i}$ satisfies the following (global) conditions.
  - If for each $i' < i$ such that $w_{i'} = x$, there exists i'', $i' < i'' < i$, such that $x \in w_{i''}$,[7] then $\overline{w_i}$ can be any element of $\Sigma \setminus \Theta$.
  - Otherwise, let i' be the maximal integer less than i such that $w_{i'} = x$ and no symbol $X' \subseteq X$ that appears between the i'th and the ith positions of $\boldsymbol{w}$ contains x. Then $\overline{w_i} = \overline{w_{i'}}$.

The word $\overline{\boldsymbol{w}} = \overline{w_1}\,\overline{w_2} \cdots \overline{w_n}$, where $\overline{w_i}$ is as defined above, $i = 1, 2, \ldots, n$, is called an *instance* of $\boldsymbol{w}$. The set of all instances of $\boldsymbol{w}$ is denoted by $I(\boldsymbol{w})$.

Example 4.1. Let $\boldsymbol{w}_1, \boldsymbol{w}_2 \in (X \cup \Theta \cup 2^X)^*$. Then $I(\boldsymbol{w}_1 \emptyset \boldsymbol{w}_2) = I(\boldsymbol{w}_1 \boldsymbol{w}_2)$.[8]

Next, for a language $L \subseteq (X \cup \Theta \cup 2^X)^*$, we denote by $[L]$ the set of all instances of all elements of L. That is, $[L] = \bigcup_{\boldsymbol{w} \in L} I(\boldsymbol{w})$.

Finally, for a UB-expression α we define the language $L(\alpha)$ (over Σ) as the set of all instances of the elements of $L(\underline{\alpha})$: $L(\alpha) = [L(\underline{\alpha})]$.[9]

[7] Of course, in such case, $w_{i''}$ must be of the form $X' \subseteq X$.
[8] Note that $\boldsymbol{w}_1 \emptyset \boldsymbol{w}_2 \in (X \cup \Theta \cup 2^X)^*$.
[9] Recall that $L(\underline{\alpha})$ is a language over $X \cup \Theta \cup 2^X$.

Example 4.2. It can be readily seen that $\cdot_X$ and ^{*_X} behave like the ordinary concatenation and Kleene star, respectively. In addition, for a non-empty X', $^{*_{X'}}$ is redundant, because $L(\alpha^{*_{X'}}) = L((\alpha \cdot_{X'} \epsilon)^{*_\emptyset})$.[10]

Example 4.3. The language from Example 3.2 is $L(x \cdot_\emptyset x)$. Similarly, for a UB-expression $\alpha = x \cdot_\emptyset y^{*_{\{y\}}} \cdot_\emptyset x$ over $(\{x, y\}, \emptyset)$, $L(\alpha)$ consists of all words over Σ having the same first and last symbols. Thus, $L(y^{*_{\{y\}}} \cdot_\emptyset x \cdot_\emptyset y^{*_{\{y\}}} \cdot_\emptyset x \cdot_\emptyset y^{*_{\{y\}}})$ is the language from Example 3.5.

Example 4.4. Consider a subclass of UB-expressions, called *FSDA*-expressions, that is defined below.

- $\emptyset, \epsilon$, and UB-expressions of the form $(\theta \cdot_\emptyset (x_1 \cdot_\emptyset x_2))$, where $\theta \in \Theta$ and $x_1, x_2 \in X$ are FSDA-expressions.
- If α_1 and α_2 are FSDA-expressions, then so are $(\alpha_1 + \alpha_2)$, $(\alpha_1 \cdot_{X'} \alpha_2)$, and $(\alpha_1^{*_{X'}})$.

It easily follows from Example 3.3 and the constructions in Sections 5 and 6 that FSDA languages are defined by FSDA-expressions and vice versa, each FSDA expression defines an FSDA language.

Theorem 4.1. A language is defined by a UB-expression if and only if it is accepted by an FSUBA.

The proof of the "if" part of Theorem 4.1 is based on a tight relationship between FSUBA and the ordinary finite automata. It is presented in the next section. The proof of the "only if" part of the theorem is based on the relevant closure properties of FSUBA-languages and is quite standard. For the sake of completeness, we present it in Section 6.

We conclude this section with one more closure property of FSUBA-languages that is an immediate corollary to Theorem 4.1.

Corollary 4.1. FSUBA languages are closed under reversing.[11]

Proof:
It can be easily verified that, for a UB-expression α, $(L(\alpha))^R = L(\alpha^R)$, where α^R is defined by the following induction.

- If $\alpha \in \{\emptyset, \epsilon\} \cup X \cup \Theta$, then α^R is α.
- $(\alpha_1 + \alpha_2)^R$ is $(\alpha_1^R + \alpha_2^R)$.
- $(\alpha_1 \cdot_{X'} \alpha_2)^R$ is $(\alpha_2^R \cdot_{X'} \alpha_1^R)$.
- $((\alpha)^{*_{X'}})^R$ is $(\alpha^R)^{*_{X'}}$.

□

Remark 4.1. Using an alternative equivalent model of computation that is similar to M-automata introduced in [8], one can show that FSUBA languages are also closed under intersection.[12]

[10] Of course, ϵ is redundant as well (but, still, very useful), because $L(\emptyset^*) = \{\epsilon\}$.
[11] It should be pointed out that FMA languages are not closed under reversing, see [8, Example 8]. Therefore, it is unlikely that there is a kind of regular expressions for FMA languages.
[12] It follows from Example 3.2 that FSUBA languages are not closed under complementation.

5. Proof of the "if" part of Theorem 4.1

The proof of the "if part" of Theorem 4.1 is based on a tight relationship between FSUBA and ordinary FA. We shall use the following model of FA.

Definition 5.1. ([10]) A (non-deterministic) finite state automaton over a finite alphabet Σ' is a system $\boldsymbol{M} = (Q, q_0, F, \Delta)$, where

- Q is a finite set of states.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of final states.
- Δ is a finite subset of $Q \times \Sigma'^* \times Q$ called the *transition relation*.

A word $\boldsymbol{w}$ over alphabet Σ' is accepted by $\boldsymbol{M}$, if there is a partition $\boldsymbol{w} = w_1 \cdots w_n$ of $\boldsymbol{w}$, $w_i \in \Sigma'^*$, and a sequence of states $s_0, s_1, \ldots, s_n$ such that

- $s_0 = q_0$,
- $s_n \in F$, and
- and for each $i = 0, 1, \ldots, n-1$, $(s_i, w_{i+1}, s_{i+1}) \in \Delta$.

We shall also need the following modification of [17, Lemma 2] (see Remark 5.1 below).

Lemma 5.1. Let $\boldsymbol{A} = (\Sigma, Q, q_0, F, \boldsymbol{u}, \Theta, \mu)$ be an r-register FSUBA, where $[\boldsymbol{u}] = \{\theta_1, \theta_2, \ldots, \theta_\ell\}$, and $\Theta = \{\theta_{\ell'+1}, \theta_{\ell'+2}, \ldots, \theta_\ell\}$. Then the language $L(\boldsymbol{A})$ is accepted by an FSUBA $\boldsymbol{A}'$ of the form $(\Sigma, Q', q_0', F', \#^r\theta_1\theta_2\cdots\theta_\ell, \{\theta_1, \theta_2, \ldots, \theta_\ell\}, \mu')$ such that for each transition $(q, k, S, q') \in \mu'$, $S \subseteq \{1, 2, \ldots, r\}$.[13]

The proof of Lemma 5.1 is presented in Section 7.

Remark 5.1. The only difference between Lemma 5.1 and [17, Lemma 2] is that in the latter

1. some of the θ_ms may appear in two or more registers of the initial assignment, and
2. the indices of the registers which may be reset do not necessarily belong to $\{1, 2, \ldots, r\}$.

Let L be an FSUBA language. By Lemma 5.1, we may assume that L is accepted by an FSUBA $\boldsymbol{A} = (\Sigma, Q, q_0, F, \#^r\theta_1\theta_2\cdots\theta_\ell, \{\theta_1, \theta_2, \ldots, \theta_\ell\}, \mu)$ such that for each transition $(q, k, S, q') \in \mu$, $S \subseteq \{1, 2, \ldots, r\}$.

Let $X = \{x_1, \ldots, x_r\}$ and consider a finite state automaton $\boldsymbol{M}^{\boldsymbol{A}} = (Q, q_0, F, \Delta)$ over $X \cup \Theta \cup 2^X$, where transition relation Δ is defined as follows. For every transition $(q, k, S, q') \in \mu$, Δ contains:

- $(q, \theta_{k-r}X', q')$, if $k > r$, and
- $(q, x_k X', q')$, if $k \leq r$,

where $X' = \{x_i : i \in S\}$.

[13] That is, only the first r registers of $\boldsymbol{A}'$ may be reset.

Remark 5.2. Note that the diagrams of $\boldsymbol{A}$ and $\boldsymbol{M^A}$ differ only in the transition labels which can be recovered from each other in a straightforward manner.

The proof of the "if" part of Theorem 4.1 is based on the fact that set of the sequences of labels of the accepting paths of $\boldsymbol{M^A}$ is regular. Namely, the "if" part of Theorem 4.1 immediately follows from Theorem 5.1 below.

Theorem 5.1. Let β be a regular expression over $X \cup \Theta \cup 2^X$ such that $L(\beta) = L(\boldsymbol{M^A})$ and let α be a UB-expression that results from β in replacing each occurrence of $X' \subseteq \{x_1, x_2, \ldots, x_n\}$ with $(\epsilon \cdot_{X'} \epsilon)$, each occurrence of $\cdot$ with $\cdot_\emptyset$, and each occurrence of * with $^{*\emptyset}$.[14] Then, $L(\alpha) = L(\boldsymbol{A})$.

Proof:
It follows from Example 4.1 that $[L(\underline{\alpha})] = [L(\beta)]$. Therefore, since $L(\beta) = L(\boldsymbol{M^A})$, the proof will be completed if we show that $L(\boldsymbol{A})$ consists of all instances of the elements of $L(\boldsymbol{M^A})$.

Let $\boldsymbol{p} = e_1, \ldots, e_n$ be a path of edges in the diagram of $\boldsymbol{A}$. One can think of $\boldsymbol{p}$ as the diagram of an FSUBA, also denoted by $\boldsymbol{p}$. Then $L(\boldsymbol{p})$ consists of all words of length n over Σ which "drive $\boldsymbol{A}$ through $\boldsymbol{p}$ from its first to its last vertex (state)."

Let $\boldsymbol{P}$ denote the set of all paths $\boldsymbol{p}$ starting from the initial state and ending at a final state. Then

$$L(\boldsymbol{A}) = \bigcup_{\boldsymbol{p} \in \boldsymbol{P}} L(\boldsymbol{p}).$$

On the other hand, by Remark 5.2, $\boldsymbol{p}$ has the corresponding path in $\boldsymbol{M^A}$, also denoted by $\boldsymbol{p}$, that differs from it only in the transition labels. These labels form a word over $X \cup \Theta \cup 2^X$ that we shall denote by $\boldsymbol{w_p}$. Therefore,

$$L(\boldsymbol{M^A}) = \{\boldsymbol{w_p} : \boldsymbol{p} \in \boldsymbol{P}\}.$$

Consequently the equality $L(\boldsymbol{A}) = [L(\boldsymbol{M^A})]$ will follow if we prove that for each path $\boldsymbol{p}$

$$I(\boldsymbol{w_p}) = L(\boldsymbol{p}),$$

i.e., $L(\boldsymbol{p})$ consists of all instances of $\boldsymbol{w_p}$.

The proof is by induction on the length n of $\boldsymbol{p}$. The case of $n = 0$ is immediate because the only instance of ϵ is ϵ.

For the induction step, we assume that equality $I(\boldsymbol{w_p}) = L(\boldsymbol{p})$ holds for all paths $\boldsymbol{p}$ of length n and shall show that it holds for all paths $\boldsymbol{p'}$ of length $n+1$.

Let $\boldsymbol{p}$ be a path of length n and $\boldsymbol{p'} = \boldsymbol{p}, (q, k, S, q')$ be a path of length $n+1$. Let $\boldsymbol{w_{p'}} = \boldsymbol{w_p} w_{n+1} X_{n+1}$.

If $k > r$, then, by the definition of $\boldsymbol{M^A}$, $w_{n+1} = \theta_{k-r}$. Therefore, $L(\boldsymbol{p'}) = (L(\boldsymbol{p}))\{\theta_{k-r}\}$ and, by the definition of an instance of a word, $I(\boldsymbol{w_{p'}}) = (I(\boldsymbol{w_p}))\{\theta_{k-r}\}$. Since, by the induction hypothesis, $I(\boldsymbol{w_p}) = L(\boldsymbol{p})$, $I(\boldsymbol{w_{p'}}) = L(\boldsymbol{p'})$ follows.

If $k \leq r$, then, by the definition of $\boldsymbol{M^A}$, $w_{n+1} = x_k$.

Let i be the greatest integer less than $n+1$ such that w_i is x_k, if such integer exists, and be 0, otherwise. That is, i is the last time register k appears in $\boldsymbol{p}$. Also, let i' be the greatest integer less than

[14] Cf. Example 4.2.

$n+1$ such that $x_k \in w_{i'}$, if such integer exists, and be 0, otherwise. That is, i' is the last time register k is reset in $\boldsymbol{p}$.

We shall distinguish between the cases of $i' \geq i$ and $i' < i$.

Assume first that $i' \geq i$. Then $I(\boldsymbol{w}_{\boldsymbol{p}'}) = (I(\boldsymbol{w}_{\boldsymbol{p}}))(\Sigma \setminus \Theta)$. Similarly, the kth register of FSUBA $\boldsymbol{p}'$ is reset at the i'th move and not updated until its last move. Therefore, $L(\boldsymbol{p}') = (L(\boldsymbol{p}))(\Sigma \setminus \Theta)$. Since, by the induction hypothesis, $I(\boldsymbol{w}_{\boldsymbol{p}}) = L(\boldsymbol{p})$, the desired equality $I(\boldsymbol{w}_{\boldsymbol{p}'}) = L(\boldsymbol{p}')$ follows.

Now assume that $i' < i$. By the definition of an instance of a word, the symbol assigned to x_k in the ith position must be assigned to it again in the $(n+1)$th position. That is, $\overline{w_1\,w_2} \cdots \overline{w_n\,w_{n+1}} \in I(\boldsymbol{w}_{\boldsymbol{p}'})$ if and only if $\overline{w_1\,w_2} \cdots \overline{w_n} \in I(\boldsymbol{w}_{\boldsymbol{p}})$ and $\overline{w_{n+1}} = \overline{w_i}$. Similarly, the kth register of FSUBA $\boldsymbol{p}'$ is used in the ith transition and is not reset till the end of the computation. Thus, a word is accepted by FSUBA $\boldsymbol{p}'$ if and only if it is accepted by $\boldsymbol{p}$ and the symbol that appears in the ith position of the word also appears in its $(n+1)$th position. Since, by the induction hypothesis, $I(\boldsymbol{w}_{\boldsymbol{p}}) = L(\boldsymbol{p})$, the equality $I(\boldsymbol{w}_{\boldsymbol{p}'}) = L(\boldsymbol{p}')$ follows in the latter case as well.

This completes the proof of the induction hypothesis and the theorem. □

6. Proof of the "only if" part of Theorem 4.1

The proof is quite standard: it is based on the relevant closure properties of FSUBA languages.[15]

Let α be a UB-expressions over (X, Θ), where $X = \{x_1, x_2, \ldots, x_r\}$ and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_\ell\}$. We shall prove by induction on the length of α that $L(\alpha)$ is accepted by an FSUBA whose initial assignment $\boldsymbol{u}$ is $\#^r\theta_1 \cdots \theta_\ell$ and whose last ℓ registers may not be reset,[16] cf. Lemma 5.1.

The basis is, actually, Example 3.1 and for the induction step we start with the case in which α is of the form $(\alpha_1 + \alpha_2)$. By the induction hypothesis, there are FSUBA $\boldsymbol{A}_1 = (\Sigma, Q_1, q_{1,0}, F_1, \boldsymbol{u}, \Theta, \mu_1)$ and $\boldsymbol{A}_2 = (\Sigma, Q_2, q_{2,0}, F_2, \boldsymbol{u}, \Theta, \mu_2)$ such that $L(\alpha_1) = L(\boldsymbol{A}_1)$ and $L(\alpha_2) = L(\boldsymbol{A}_2)$. Renaming the automaton states, if necessary, we may assume that Q_1 and Q_2 are disjoint. Consider an FSUBA $\boldsymbol{A} = (\Sigma, Q, q_0, F, \boldsymbol{u}, \Theta, \mu)$, where q_0 is a new initial state state, $Q = Q_1 \cup Q_2 \cup \{q_0\}$,

$$F = \begin{cases} F_1 \cup F_2, & \text{if } q_{1,0} \notin F_1 \text{ and } q_{2,0} \notin F_2 \\ F_1 \cup F_2 \cup \{q_0\}, & \text{otherwise} \end{cases},$$

and

$$\mu = \mu_1 \cup \mu_2 \cup \{(q_0, k, S, q) : (q_{1,0}, k, S, q) \in \mu_1 \text{ or } (q_{2,0}, k, S, q) \in \mu_2\}.$$

That is, depending on the first move, $\boldsymbol{A}$ acts either like $\boldsymbol{A}_1$ or $\boldsymbol{A}_2$. Thus,

$$L(\alpha) = L(\alpha_1 + \alpha_2) = L(\alpha_1) \cup L(\alpha_2) = L(\boldsymbol{A}_1) \cup L(\boldsymbol{A}_2) = L(\boldsymbol{A}).$$

Let α be of the form $(\alpha_1 \cdot_{X'} \alpha_2)$. By the induction hypothesis, there are FSUBA $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ such that $L(\alpha_1) = L(\boldsymbol{A}_1)$ and $L(\alpha_2) = L(\boldsymbol{A}_2)$. Let $\boldsymbol{A}_1 = (\Sigma, Q_1, q_{1,0}, F_1, \boldsymbol{u}, \Theta, \mu_1)$ and $\boldsymbol{A}_2 = (\Sigma, Q_2, q_{2,0}, F_2, \boldsymbol{u}, \Theta, \mu_2)$ Renaming the automaton states, if necessary, we may assume that Q_1 and Q_2

[15] In [17] the author deals with the inclusion problem only and does not address the closure properties of FSUBA languages at all.

[16] That is, for each transition (q, k, S, q') of the automaton, $S \subseteq \{1, 2, \ldots, r\}$.

are disjoint. Consider an FSUBA $\boldsymbol{A} = (\Sigma, Q_1 \cup Q_2, q_{1,0}, F_2, \boldsymbol{u}, \Theta, \mu)$, where

$$\begin{aligned}\mu \;=\;& \mu_1 \cup \mu_2 \cup \\ & \{(q, k, \{i : x_i \in X'\}, q_{2,0}) : \text{for some } S \subseteq \{1, 2, \ldots, r\} \\ & \qquad\qquad \text{and some } q' \in F_1,\ (q, k, S, q') \in \mu_1\}.\end{aligned}$$

That is, instead of entering a final state of $\boldsymbol{A}_1$, $\boldsymbol{A}$ may enter the initial state of $\boldsymbol{A}_2$ and reset the registers corresponding to the elements of X'. Thus,

$$L(\alpha) = L(\alpha_1 \cdot_{X'} \alpha_2) = L(\boldsymbol{A}).$$

The case in which α is of the form $(\alpha_1^{*_{X'}})$, $X' \subseteq X$, is similar to the above and is omitted.

7. Proof of Lemma 5.1

The proof basically follows the proof of [17, Lemma 2]. The idea lying behind the construction of $\boldsymbol{A}'$ is quite standard. We replace all θ_ms in the registers of $\boldsymbol{A}$ with $\#$, add to $\boldsymbol{A}$ ℓ new registers containing θ_ms, $m = 1, 2, \ldots, \ell$, and remember the registers of $\boldsymbol{A}$ containing θ_ms by a state of $\boldsymbol{A}'$. That is, FSUBA $\boldsymbol{A}' = (\Sigma, Q', q_0', F', \#^r\theta_1\theta_2\cdots\theta_\ell, \{\theta_1, \theta_2, \ldots, \theta_\ell\}, \mu')$ is defined as follows.

- $Q' = Q \times \{\#, \theta_1, \theta_2, \ldots, \theta_\ell, *\}^r$.
  The intended meaning of component $a_1a_2\cdots a_r \in \{\#, \theta_1, \theta_2, \ldots, \theta_\ell, *\}^r$ in $(q, a_1a_2\cdots a_r) \in Q'$ is that it corresponds to the register component $w_1w_2\cdots w_r$ of a configuration $(q, w_1w_2\cdots w_r)$ of $\boldsymbol{A}$ in the following sense.

  $$a_j = \begin{cases} w_j, & \text{if } w_j \in \{\#, \theta_1, \theta_2, \ldots, \theta_\ell\} \\ *, & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, r.$$

  Note that in $\boldsymbol{A}'$ only the $(r+m)$th register contains θ_m, $m = 1, 2, \ldots, \ell$.

- $q_0' = (q_0, \boldsymbol{u})$. That is, the second component of q_0' is $\boldsymbol{a} = a_1a_2\cdots a_r$, where

  $$a_j = \begin{cases} \#, & \text{if } u_j = \# \\ \theta_m, & \text{if } u_j = \theta_m \end{cases}, \quad j = 1, 2, \ldots, r.$$

- $F' = F \times \{\#, \theta_1, \theta_2, \ldots, \theta_\ell, *\}^r$.

- μ' consists of all transitions $((q, a_1a_2\cdots a_r), k', S', (q', a_1'a_2'\cdots a_r'))$ such that

  - $S' \subseteq \{1, 2, \ldots, r\}$,
  - for all $j \in S'$, $a_j' = \#$,
  - for all $j \notin S' \cup \{k'\}$, $a_j' = a_j$,

  and conditions 1,2 and 3 below are satisfied.

1. If $k' \in \{1, 2, \ldots, r\}$, then $(q, k', S', q') \in \mu$, $a_{k'} \in \{*, \#\}$, and
$$a'_{k'} = \begin{cases} *, & \text{if } k' \notin S' \\ \#, & \text{if } k' \in S' \end{cases} .$$

2. If $k' = r + m$, $m = 1, 2, \ldots, \ell'$,[17] then for some $k \in \{1, 2, \ldots, r\}$, $(q, k, S', q') \in \mu$, $a_k \in \{\#, \theta_m\}$,[18] and
$$a'_k = \begin{cases} \theta_m, & \text{if } k \notin S' \\ \#, & \text{otherwise} \end{cases} ,$$

3. If $k' = r + m$, $m = \ell' + 1, \ell' + 2, \ldots, \ell$, then for some $k \in \{1, 2, \ldots, r\}$, $(q, k, S', q') \in \mu$, $a_k = \theta_m$, and
$$a'_k = \begin{cases} \theta_m, & \text{if } k \notin S' \\ \#, & \text{otherwise} \end{cases} ,$$

see the intuition of the θ_ms in the definition of Q'.

Let $\boldsymbol{\sigma} = \sigma_1\sigma_2 \cdots \sigma_n \in \Sigma^*$. We shall prove by induction on n that there is a run $\boldsymbol{c} = c_0, c_1, \ldots, c_n$ of $\boldsymbol{A}$ on $\boldsymbol{\sigma}$, $c_i = (q_i, \boldsymbol{w}_i)$, $i = 0, 1, \ldots, n$, if and only if there is a run $\boldsymbol{c}' = c'_0, c'_1, \ldots, c'_n$ of $\boldsymbol{A}'$ on $\boldsymbol{\sigma}$, $c'_i = ((q_i, \boldsymbol{a}_i), \boldsymbol{v}_i\theta_1\theta_2 \cdots \theta_\ell)$, $\boldsymbol{a}_i = a_{i,1}a_{i,2} \cdots a_{i,r} \in \{\#, \theta_1, \theta_2, \ldots, \theta_\ell, *\}^r$ and $\boldsymbol{v}_i = v_{i,1}v_{i,2} \cdots v_{i,r} \in (\Sigma \setminus \{\theta_1, \theta_2, \ldots, \theta_\ell\})^r$, $i = 0, 1, \ldots, n$, such that for all $i = 0, 1, \ldots, n$, conditions 1,2, and 3 below are satisfied.

1.
$$a_{i,j} = \begin{cases} w_{i,j}, & \text{if } w_{i,j} \in \{\#, \theta_1, \theta_2, \ldots, \theta_\ell\} \\ *, & \text{otherwise} \end{cases} , \quad j = 1, 2, \ldots, r.$$

2.
$$v_{i,j} = \begin{cases} w_{i,j}, & \text{if } w_{i,j} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\} \\ \#, & \text{otherwise} \end{cases} , \quad j = 1, 2, \ldots, r.$$

3. Let k and k' be the transition registers between configurations c_{i-1} and c_i and configurations c'_{i-1} and c'_i, respectively, $i = 1, 2, \ldots, n$. Then
$$k' = \begin{cases} r + m, & \text{if } \sigma_i = \theta_m, \ m = 1, 2, \ldots, \ell \\ k, & \text{otherwise} \end{cases} .$$

The lemma will follow from the above equivalence and the definition of $F' = F \times \{*, \theta_1, \theta_2, \ldots, \theta_\ell, \#\}^r$. *Basis*: $n = 0$ implies that $\boldsymbol{c} = c_0 = (q_0, \boldsymbol{u})$ and $\boldsymbol{c}' = c'_0 = (q'_0, \#^r\theta_1\theta_2 \cdots \theta_\ell) = ((q_0, \boldsymbol{u}), \#^r\theta_1\theta_2 \cdots \theta_\ell)$ are runs of $\boldsymbol{A}$ and $\boldsymbol{A}'$ on $\boldsymbol{\sigma}$ $(= \epsilon)$, respectively. Then conditions 1 and 2 follow from the definition of the initial assignment of $\boldsymbol{A}'$, and condition 3 is satisfied, because there are no transitions on the empty word.

[17] Recall that $[\boldsymbol{u}] = \{\theta_1, \theta_2, \ldots, \theta_\ell\}$ and $\Theta = \{\theta_{\ell'+1}, \theta_{\ell'+2}, \ldots, \theta_\ell\}$, see the statement of Lemma 5.1.
[18] That is, the kth register of $\boldsymbol{A}$ either contains θ_m or is empty.

Induction step: Assume that the induction hypothesis is true for all words of length n and prove it for a word $\boldsymbol{\sigma} = \sigma_1\sigma_2\cdots\sigma_n\sigma_{n+1} \in \Sigma^*$. We start with the proof of the "only if" direction.

Let $\boldsymbol{c} = c_0, c_1, \ldots, c_n, c_{n+1}$, where $c_i = (q_i, \boldsymbol{w}_i)$, $i = 0, 1, \ldots, n$, be a run of $\boldsymbol{A}$ on $\boldsymbol{\sigma}$ and let (q_n, k, S, q_{n+1}) be the transition of $\boldsymbol{A}$ on σ_{n+1} between configurations c_n and c_{n+1}. By the induction hypothesis, there is a run $\boldsymbol{c}' = c'_0, c'_1, \ldots, c'_n$ of $\boldsymbol{A}'$ on $\sigma_1\sigma_2\cdots\sigma_n$, $c'_i = ((q_i, \boldsymbol{a}_i), \boldsymbol{v}_i)$, $i = 0, 1, \ldots, n$, such that for all $i = 0, 1, \ldots, n$, conditions 1,2, and 3 above are satisfied. Let

$$a_{n+1,j} = \begin{cases} w_{n+1,j}, & \text{if } w_{n+1,j} \in \{\#, \theta_1, \theta_2, \ldots, \theta_\ell\} \\ *, & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, r, \tag{1}$$

$q'_{n+1} = (q_{n+1}, \boldsymbol{a}_{n+1})$,

$$v_{n+1,j} = \begin{cases} w_{n+1,j}, & \text{if } w_{n+1,j} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\} \\ \#, & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, r, \tag{2}$$

and let

$$k' = \begin{cases} r + m, & \text{if } \sigma_{n+1} = \theta_m, \ m = 1, 2, \ldots, \ell \\ k, & \text{otherwise} \end{cases}.$$

Recall that k is the transition register between configurations c_n and c_{n+1}.

The proof will be completed if we show that $((q'_n, \boldsymbol{v}_n), \sigma_{n+1}, (q'_{n+1}, \boldsymbol{v}_{n+1}))$ is in μ'^c and that k' is the transition register between configurations c'_n and c'_{n+1}. We shall distinguish between the cases of $\sigma_{n+1} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\}$, $\sigma_{n+1} \in \{\theta_1, \theta_2, \ldots, \theta_{\ell'}\}$, and $\sigma_{n+1} \in \{\theta_{\ell'+1}, \theta_{\ell'+2}, \ldots, \theta_\ell\}$.

Let $\sigma_{n+1} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\}$. Then $k' = k \in \{1, 2, \ldots, r\}$. First we show that the transition $((q_n, \boldsymbol{a}_n), k, S, (q_{n+1}, \boldsymbol{a}_{n+1}))$ is in μ':

- By definition, $(q_n, k, S, q_{n+1}) \in \mu$.
- Since $\sigma_{n+1} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\}$, $w_{n,k} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\}$ either. Thus, by the induction hypothesis, $a_{n,k} \in \{*, \#\}$.
- Since (q_n, k, S, q_{n+1}) is the transition between configurations $\boldsymbol{c}_n$ and $\boldsymbol{c}_{n+1}$, $w_{n+1,k} \in \{\sigma_{n+1}, \#\}$. Thus, by (1), $a_{n+1,k} \in \{*, \#\}$.
- If $j \in S$, then $w_{n+1,j} = \#$. Consequently, by (1), $a_{n+1,j} = \#$.
- If $j \notin S \cup \{k\}$, then $w_{n+1,j} = w_{n,j}$, and, by (1) and the induction hypothesis, $a_{n+1,j} = a_{n,j}$.

That is, condition 1 of the definition of μ' is satisfied.

We continue to prove that $((q_n, \boldsymbol{a}_n), k, S, (q_{n+1}, \boldsymbol{a}_{n+1}))$ is a transition of $\boldsymbol{A}'$ on σ_{n+1} between configurations c'_n and c'_{n+1}.

- Since either $w_{n,k} = \#$ (and $\sigma_{n+1} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\}$) or $w_{n,k} = \sigma_{n+1}$, by condition 2 of the induction hypothesis, either $v_{n,k} = \#$ or $v_{n,k} = \sigma_{n+1}$.
- If $k \notin S$, then $w_{n+1,k} = \sigma_{n+1}$, and, by (2), $v_{n+1,k} = \sigma_{n+1}$.
- Let $j \in S$. Then $w_{n+1,j} = \#$ and, by (2), $v_{n+1,j} = \#$.

- Let $j \notin S \cup \{k\}$. Then $w_{n+1,j} = w_{n,j}$ and, by (2) and the induction hypothesis, $v_{n+1,j} = v_{n,j}$.

Thus, $((q_n, \boldsymbol{a}_n), k', S, (q_{n+1}, \boldsymbol{a}_{n+1}))$ is a transition of $\boldsymbol{A}'$ on σ_{n+1} between configurations c'_n and c'_{n+1}.

The treatment of the cases of $\sigma_{n+1} \in \{\theta_1, \theta_2, \ldots, \theta_{\ell'}\}$ and $\sigma_{n+1} \in \{\theta_{\ell'+1}, \theta_{\ell'+2}, \ldots, \theta_\ell\}$ (which correspond to conditions 2 and 3 of the definition of μ') is similar to the above and is omitted.

For the proof of the "if" direction let $\boldsymbol{c}' = c'_0, c'_1, \ldots, c_n, c'_{n+1}$, be a run of $\boldsymbol{A}'$ on $\boldsymbol{\sigma}$, where $c'_i = ((q_i, \boldsymbol{a}_i), \boldsymbol{v}_i)$, $i = 0, 1, \ldots, n$, and let $((q_n, \boldsymbol{a}_n), k', S', (q_{n+1}, \boldsymbol{a}_{n+1})$ be the transition of $\boldsymbol{A}'$ on σ_{n+1} between configurations c'_n and c'_{n+1}. By the induction hypothesis, there is a run $\boldsymbol{c} = c_0, c_1, \ldots, c_n$, $c_i = (q_i, \boldsymbol{w}_i)$, $i = 0, 1, \ldots, n$, of $\boldsymbol{A}$ on $\boldsymbol{\sigma}$ such that for all $i = 0, 1, \ldots, n$, conditions 1,2 and 3 of the induction hypothesis are satisfied. Let

$$w_{n+1,j} = \begin{cases} v_{n+1,j}, & \text{if } a_{n+1,j} = * \\ a_{n+1,j}, & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, r. \tag{3}$$

The proof will be completed if we show that $((q_n, \boldsymbol{w}_n), \sigma_{n+1}, (q_{n+1}, \boldsymbol{w}_{n+1}))$ is in μ^c. Like in the case of the "only if" direction, we shall distinguish between the cases of $\sigma_{n+1} \notin \{\theta_1, \theta_2, \ldots, \theta_\ell\}$, $\sigma_{n+1} \in \{\theta_1, \theta_2, \ldots, \theta_{\ell'}\}$, and $\sigma_{n+1} \in \{\theta_{\ell'+1}, \theta_{\ell'+2}, \ldots, \theta_\ell\}$. These cases are treated similarly each to other, and we shall consider only the case of $\sigma_{n+1} \in \{\theta_1, \theta_2, \ldots, \theta_{\ell'}\}$.

So, let $\sigma_{n+1} = \theta_m \in \{\theta_1, \theta_2, \ldots, \theta_{\ell'}\}$. Then $k' = r + m$ and, by the definition of μ', for some $k = 1, 2, \ldots, r$, $(q_n, k, S', q_{n+1}) \in \mu$. We contend that (q_n, k, S', q_{n+1}) is a transition of $\boldsymbol{A}'$ on $\sigma_{n+1}\,(= \theta_m)$ between configurations $(q_n, \boldsymbol{w}_n)$ and $(q_{n+1}, \boldsymbol{w}_{n+1})$.

- By the definition of μ', $a_{n,k} \in \{\#, \theta\}$. Thus, by condition 1 of the induction hypothesis, $w_{n,k} \in \{\#, \theta\}$.
- If $k \notin S'$, then $a_{n+1,k} = \theta_m$. Thus, by condition 2 of the induction hypothesis, $w_{n+1,k} = \theta_m$.
- Let $j \in S'$. Then $a_{n+1,j} = \#$ and, by (3), $w_{n+1,j} = \#$.
- Let $j \notin S' \cup \{k\}$. Then $a_{n+1,j} = a_{n,j}$ and $v_{n+1,j} = v_{n,j}$. Therefore, by condition 2 of the induction hypothesis and (3), $w_{n+1,j} = w_{n,j}$.

That is, (q_n, k, S', q_{n+1}) is indeed a transition of $\boldsymbol{A}'$ on σ_{n+1} between configurations $(q_n, \boldsymbol{w}_n)$ and $(q_{n+1}, \boldsymbol{w}_{n+1})$, which completes the proof.

8. Concluding remarks

We conclude the paper with a discussion of the complexity of intertranslations between FSUBA and UB-expressions.

It follows from [1, Theorem 5.5, p. 198] that the complexity of the construction of a UB-expression from an FSUBA given by Lemma 5.1 is cubic in the number of the FSUBA states, whereas the complexity of the construction of an FSUBA from a UB-expression is linear in the UB-expression length, cf. [1, Theorem 9.2, p. 322]. That is, the complexity of these constructions is the same as of the corresponding classical ones.

However, in order to construct a UB-expression from a general FSUBA, we first convert it into a FSUBA given by Lemma 5.1, which makes the complexity exponential in the number of the FSUBA registers.

References

[1] Aho, A., Hopcroft, J., Ullman, J.: *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, USA, 1981.

[2] Autebert, J.-M., Beauquir, J., Boasson, L.: Langages des Alphabets Infinis, *Discrete Applied Mathematics*, **2**, 1980, 1–20.

[3] Autebert, J.-M., Beauquir, J., Boasson, L.: Formes de langages et de grammaries, *Acta Informatica*, **17**, 1982, 193–213.

[4] Bielecki, M., Hidders, J., Paredaens, J., Tyszkiewicz, J., den Bussch, J. V.: Navigating with a browser, *Proceedings of the 29th International Colloquium on Automata, Languages and Programming – ICALP 2002* (P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, R. Conejo, Eds.), Springer, Berlin, 2002, 764–775, Lecture Notes in Computer Science 2380.

[5] Bolling, B., Leucker, M., Noll, T.: *Regular MSA Languages*, Technical report, Department of Computer Science, Aachen University of Technology, 2001.

[6] Itd, J.: Automates a pile sur des alphabets infinis, *Proceedings of the Symposium of Theoretical Aspects of Computer Science*, Springer-Verlag, Berlin, 1984, 260–273, Lecture Notes in Computer Science 166.

[7] Kaminski, M., Francez, N.: Finite-Memory Automata, *Proceedings of the 31th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1990, 683–688.

[8] Kaminski, M., Francez, N.: Finite-memory automata, *Theoretical Computer Science A*, **138**, 1994, 329–363.

[9] Kleene, S.: Representation of Events by Nerve Nets and Finite Automata, in: *Automata Studies* (C. Shannon, J. McCarthy, Eds.), Princeton University Press, Princeton NJ, USA, 1956, 3–42.

[10] Lewis, H., Papadimitriou, C.: *Elements of the Theory of Computation*, Prentice-Hall, Inc., Englewood Cliffs, NJ, USA, 1981.

[11] Neven, F., Schwentick, T., Vianu, V.: Towards Regular Languages over Infinite Alphabets, *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science* (J. Sgall, A. Pultr, P. Kolman, Eds.), Springer, Berlin, 2001, 560–572, Lecture Notes in Computer Science 2136.

[12] Neven, F., Schwentick, T., Vianu, V.: Finite State Machines for Strings over Infinite Alphabets, *ACM Transactions on Computational Logic*, **5**, 2004, 403–435.

[13] Otto, F.: Classes of regular and context-free languages over countably infinite alphabets, *Discrete Applied Mathematics*, **12**, 1985, 41–56.

[14] Rabin, M., Scott, D.: Finite Automata and Their Decision Problems, *IBM Journal of Research and Development*, **3**, 1959, 114–125.

[15] Sakamoto, H., Ikeda, D.: Intractability of decision problems for finite-memory automata, *Theoretical Computer Science A*, **231**, 2000, 297–308.

[16] Shemesh, Y., Francez, N.: Finite-State Unification Automata and Relational Languages, *Information and Computation*, **114**, 1994, 192–213.

[17] Tal, A.: *Decidability of Inclusion for Unification Based Automata*, M.Sc. thesis, Department of Computer Science, Technion – Israel Institute of Technology, 1999.