International Journal of Foundations of Computer Science
Vol. 21, No. 5 (2010) 741–760

DOI: 10.1142/S0129054110007532

FINITE-MEMORY AUTOMATA WITH NON-DETERMINISTIC REASSIGNMENT

MICHAEL KAMINSKI*

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

DANIEL ZEITLIN

Eternix Ltd.
Azrieli Center, Round Tower
P.O.B. 65361, Tel Aviv 61653, Israel
daniel.zeitlin@eternix.co.il

Received 22 June 2008
Accepted 20 February 2010
Communicated by Erzsébet Csuhaj-Varjú and Zoltán Ésik

In this paper we extend finite-memory automata with *non-deterministic* reassignment that allows an automaton to "guess" the future content of its registers, and introduce the corresponding notion of a *regular expression* over an infinite alphabet.

Keywords: Infinite alphabets; finite-memory automata; regular expressions.

2000 Mathematics Subject Classification: 68Q10, 68Q45

1. Introduction

A new model of finite-state automata, dealing with *infinite alphabets*, called *finite-state datalog automata* (FSDA) was introduced in [10]. FSDA were intended for the abstract study of *relational* languages. Words in these languages are formed by composing basic binary relations, and have the general form

$$r_{k_1}(x_{\ell_1}, x_{m_1}) r_{k_2}(x_{\ell_2}, x_{m_2}) \cdots r_{k_n}(x_{\ell_n}, x_{m_n}),$$

$n = 1, 2, \ldots$, where r_{k_i}s belong to a finite set of binary predicate symbols, and x_{ℓ_i} and x_{m_i}, $i = 1, 2, \ldots, n$, are variables. 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 between the

*Corresponding author.

input symbol and the edge label, which is performed in ordinary finite-state automata (FA) was replaced with *unification*, which is a crucial element of relational languages.

A few years later, FSDA were extended in [3, 4] 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 membership and the emptiness problems for FMA- (and, consequently, for FSDA-) languages is relatively simple, the problem of inclusion for FMA-languages is undecidable, see [8, 9].

An extension of FSDA to a general infinite alphabet called *finite-state unification based automata* (FSUBA) was introduced in [11]. FSUBA look similar to FMA, but are much weaker, because registers of FSUBA may contain values currently stored in other registers. It was shown in [11] that FSDA can be simulated by FSUBA and that the problem of inclusion for FSUBA languages is decidable. A kind of regular expressions for describing FSUBA languages was proposed in [5, 6].[a]

Finally, in [1] FMA were extended to *pushdown* automata over infinite alphabets. This model of computation, called *infinite-alphabet infinite-stack-alphabet pushdown automata* (IIPDA), has the same input and stack infinite alphabet. It aims towards recognizing only the "natural" analog of the ordinary context-free languages. Similarly to FMA, IIPDA are equipped with a finite set of registers, but unlike FMA, IIPDA can nondeterministically change symbols stored in their registers.

In this paper, we present an extension of FMA, called *finite-memory automaton with non-deterministic reassignment* (NFMA). This model is similar to FMA, but, in addition, is able to make a *non-deterministic reassignment* by "guessing" the content of an appropriate register. We show that NFMA are more powerful than FMA, but possess all decision and closure properties of the latter. In particular, the NFMA languages are closed under *reversing*, whereas FMA languages are not. Moreover, in addition to this very important property of the ordinary regular languages, as we show in this paper, there is a kind of regular expressions for NFMA

[a] It should be noted that there is a totally different model of computation over infinite alphabets called *pebble automata*, see [8, 9]. This model allows an automaton to remember a finite set of positions by means of *pebbles* obeying the stack discipline and to perform equality tests between the symbols on these positions. Pebble automata are orthogonal to finite-memory automata and lie out of the scope of this paper. We refer the reader to [8, 9] for a comprehensive study of these automata.

languages. Also, NFMA languages are exactly those generated by *regular* grammars over infinite alphabets, see [12]. Thus, NFMA languages are "more regular" than the FMA ones.

The paper is organized as follows. In the next section we recall the definition of FMA from [3, 4]. In Section 3 we define NFMA and in Section 4 we present some of its basic properties. Then in Section 5 we introduce regular expressions for languages over infinite alphabets, whose equivalence[b] to NFMA is proven in Section 6.

2. Finite-memory Automata

In this section we recall the definitions of finite-memory automata, see [3, 4].

Let Σ be an infinite alphabet fixed through this paper and let $\# \notin \Sigma$.[c] An *assignment* is a word $\boldsymbol{w} = w_1 w_2 \cdots w_r$, where $w_k \in \Sigma \cup \{\#\}$, $k = 1, 2, \ldots, r$, such that if $w_k \neq \#$, then for any $\ell \neq k$, $w_\ell \neq w_k$. That is, an assignment is a word over $\Sigma \cup \{\#\}$, where each symbol from Σ occurs at most one time. We denote the set of all assignments of length r by $(\Sigma \cup \{\#\})^{r\neq}$.

For an assignment $\boldsymbol{w} = w_1 w_2 \cdots w_r \in (\Sigma \cup \{\#\})^{r\neq}$ we define the *content* of $\boldsymbol{w}$, denoted $[\boldsymbol{w}]$, by $[\boldsymbol{w}] = \{w_k \neq \# : k = 1, 2, \ldots, r\}$. That is, $[\boldsymbol{w}]$ consists of all symbols from Σ which occur in the word $\boldsymbol{w}$.

Throughout this paper we use the following conventions.

- Words are always denoted by boldface letters (possibly indexed or primed) and, as usual, the empty word is denoted by ϵ.
- Bold low-case Greek letters $\boldsymbol{\alpha}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\tau}$ denote words over Σ.
- Assignments and words of fixed length over Σ are denoted by $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$, etc.
- Elements of Σ are denoted by σ, τ, a, b, c, u, v, w, usually indexed.
- Symbols which occur in a word denoted by a boldface letter are always denoted by the same *non-boldface* letter with an appropriate subscript. That is, symbols which occur in $\boldsymbol{\sigma}$ are denoted by σ_i, symbols which occur in $\boldsymbol{w}$ are denoted by w_k, etc.

Definition 1. ([4, Definition 1]) *A* finite-memory automaton *(FMA) is a system* $\mathcal{A} = \langle Q, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$ *whose components are as follows.*

- Q *is a finite set of* states.
- $s_0 \in Q$ *is the* initial state.
- $\boldsymbol{u} = u_1 u_2 \cdots u_r \in (\Sigma \cup \{\#\})^{r\neq}$ *is the* initial assignment *to the* r *registers of* $\mathcal{A}$. *Namely, the content of register* k *is* u_k, $k = 1, 2, \ldots, r$.
- $\rho : Q \to \{1, 2, \ldots, r\}$ *is a partial function, called the* reassignment. *Intuitively, if* $\mathcal{A}$ *is in state* q, $\rho(q)$ *is defined, and the input symbol occurs in no*

[b] We call two models *equivalent* if they accept/generate the same class of languages.
[c] This symbol is reserved to denote an empty register.

register, then $\mathcal{A}$ "forgets" the contents of the $\rho(q)$th register and copies the input symbol into that register.

- $\mu \subseteq Q \times \{1, 2, \ldots, r\} \times Q$ *is the* transition *relation. Intuitively, if $\mathcal{A}$ is in state p, the input symbol is equal to the content of the kth register, and $(p, k, q) \in \mu$, then $\mathcal{A}$ may enter state q and pass to the next input symbol. In addition, if the input symbol occurs in no register and is placed into the kth register ($k = \rho(p)$), then in order to enter state q the transition relation must contain (p, k, q). That is, the reassignment is made prior to a transition.*
- *Finally, $F \subseteq Q$ is the set of* final states.

Similarly to the case of the ordinary classical finite automata, $\mathcal{A}$ can be represented by its initial assignment and a finite directed graph whose nodes are states. There is an edge from p to q, if for some $k = 1, 2, \ldots, r$, $(p, k, q) \in \mu$. Such an edge is labeled k. Also, if for a node q the value of ρ is defined, then q is labeled $\rho(q)$ and if $q \in F$, it is labeled as such. For graph representation of finite-memory automata see Example 2.

An *instantaneous description* of $\mathcal{A}$ is an element of $Q \times (\Sigma \cup \{\#\})^{r\neq} \times \Sigma^*$. The first component of an instantaneous description is the (current) state of the automaton, the second one is the assignment consisting of the contents of its registers (in the increasing order of their indices), and the third component is the portion of the input yet to be read.

The relation $\vdash$ (yielding in one step) between two instantaneous descriptions is defined as follows. Let $(p, \boldsymbol{v}, \sigma\boldsymbol{\sigma})$ and $(q, \boldsymbol{w}, \boldsymbol{\sigma})$, $\boldsymbol{v} = v_1 v_2 \cdots v_r$, $\boldsymbol{w} = w_1 w_2 \cdots w_r$, and $\sigma \in \Sigma$, be instantaneous descriptions. We write

$$(p, \boldsymbol{v}, \sigma\boldsymbol{\sigma}) \vdash (q, \boldsymbol{w}, \boldsymbol{\sigma}),$$

if the following holds.

- For some $k = 1, 2, \ldots, r$, $\sigma = v_k$, $\boldsymbol{w} = \boldsymbol{v}$, and $(p, k, q) \in \mu$; or
- $\sigma \notin [\boldsymbol{v}]$, $\rho(p)$ is defined, $w_{\rho(p)} = \sigma$, $w_j = v_j$, for each $j \neq \rho(p)$, and $(p, \rho(p), q) \in \mu$.

As usual, the reflexive and transitive closure of $\vdash$ is denoted by $\vdash^*$, and we say that $\mathcal{A}$ *accepts* a word $\boldsymbol{\sigma} \in \Sigma^*$, if $(s_0, \boldsymbol{u}, \boldsymbol{\sigma}) \vdash^* (f, \boldsymbol{v}, \epsilon)$ for some $\boldsymbol{v} \in (\Sigma \cup \{\#\})^{r\neq}$ and

The set of all words accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

The requirement that the contents of FMA's registers are pairwise distinct is an important feature of FMA, because it allows an automaton to perform *both* equality and inequality tests. This is in contrast with a weaker computation model – FSUBA, where only equality tests are allowed, see [5, 6, 11]. It should be noted, however, that there is a model of computation equivalent to FMA in which the contents of the registers are not necessarily pairwise distinct. In this model, called finite-memory automaton with *multiple assignment*, the next input symbol is related to *all* registers containing this symbol, see [4, Section 3].

Example 2. *Consider a 2-register FMA* $\mathcal{A} = \langle \{s, p, f\},\ s, \#\#, \rho, \mu, \{f\} \rangle$, *where*

- $\rho(s) = 1,\ \rho(p) = 2$; *and*
- $\mu = \{(s, 1, p), (p, 2, p), (p, 2, f)\}$.

Alternatively, $\mathcal{A}$ *can be described by the diagram in Fig. 1.*

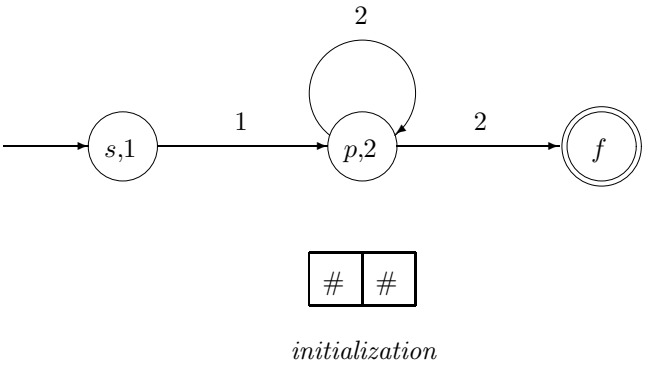

Fig. 1. The FMA $\mathcal{A}$.

Obviously, $\mathcal{A}$ *accepts all words of length greater than* 1 *whose* first *symbol differs from all the others.*

$$L(\mathcal{A}) = \{\sigma_1\sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_1,\ i = 2, 3, \ldots, n,\ n = 2, 3, \ldots\}.$$

3. Finite-memory Automata with Non-deterministic Reassignment

In this section we define the main subject of our paper – *finite-memory automata with non-deterministic reassignment* (NFMA). This model of computation is an extension of FMA and can be thought of as a "regular" counterpart of pushdown automata over infinite alphabets introduced in [1].

NFMA behave similarly to FMA, but, in addition, are able to make a *non-deterministic reassignment* by "guessing" the content of an appropriate register. They are more powerful than FMA, but still possess all decision and closure properties of the latter. In particular, the NFMA languages are closed under *reversing* (whereas the FMA languages are not). Also, there is a kind of regular expressions for NFMA languages.

Definition 3. *A* finite-memory automaton with non-deterministic reassignment *(NFMA) is a system* $\mathcal{A} = \langle Q, s_0, \boldsymbol{u}, \rho, \mu, F \rangle$ *whose components are defined as follows.*

- Q *is a finite set of* states.
- $s_0 \in Q$ *is the* initial state.

- $\boldsymbol{u} = u_1 u_2 \cdots u_r \in (\Sigma \cup \{\#\})^{r\neq}$ *is the* initial assignment *to the* r *registers of* $\mathcal{A}$.
- $\rho : \{(p,q) : (p,\epsilon,q) \in \mu\} \to \{1,2,\ldots,r\}$ *is a function called the* non-deterministic reassignment*, see the definition of* μ *in the next item. Intuitively, if* $\mathcal{A}$ *is in state* p*, there is an* ϵ*-transition from* p *to* q*, and* $\rho(p,q) = k$*, then* $\mathcal{A}$ *can* non-deterministically *replace the content of the* k*th register with an element of* Σ *not occurring in any other register and enter state* q*. Note that, unlike in Definition 1,* ρ *is defined on edges and* $\mathcal{A}$ *is allowed to "guess" the replacement.*
- $\mu \subseteq Q \times (\{1,2,\ldots,r\} \cup \{\epsilon\}) \times Q$ *is the* transition *relation. Intuitively, if* $\mathcal{A}$ *is in state* p*, the input symbol is equal to the content of the* k*th register, and* $(p,k,q) \in \mu$*, then* $\mathcal{A}$ *may enter state* q *and pass to the next input symbol. If* $(p,\epsilon,q) \in \mu$*, then* $\mathcal{A}$ *may replace the content of the* $\rho(p,q)$*th register with* any *element of* Σ *that does not occur in any other register, and enter state* q *without reading the next input symbol. That is,* $\mathcal{A}$ *is allowed to reassign* non-deterministically *the* $\rho(p,q)$*th register.*
- $F \subseteq Q$ *is the set of* final states.

Similarly to FMA, $\mathcal{A}$ can be represented by its initial assignment and a finite directed graph whose nodes are states. There is an edge from p to q, if for some $k \in \{1,2,\ldots,r\} \cup \{\epsilon\}$, $(p,k,q) \in \mu$. Such an edge is labeled k. In addition, the "ϵ-edge" (p,q) has one more label $\rho(p,q)$. Finally, the final state nodes are labeled as such. For graph representation of NFMA see the examples below.

An *instantaneous description* of $\mathcal{A}$ is an element of $Q \times (\Sigma \cup \{\#\})^{r\neq} \times \Sigma^*$. The first component of an instantaneous description is the (current) state of the automaton, the second one is the assignment consisting of the contents of its registers (in the increasing order of their indices), and the third component is the portion of the input yet to be read.

Next, we define the relation $\vdash$ (yielding in one step) between two instantaneous descriptions $(p, \boldsymbol{v}, \sigma\boldsymbol{\sigma})$ and $(q, \boldsymbol{w}, \boldsymbol{\sigma})$, $\boldsymbol{v} = v_1 v_2 \cdots v_r$, $\boldsymbol{w} = w_1 w_2 \cdots w_r$, $\sigma \in \Sigma \cup \{\epsilon\}$, and $\boldsymbol{\sigma} \in \Sigma^*$. We write

$$(p, \boldsymbol{v}, \sigma\boldsymbol{\sigma}) \vdash (q, \boldsymbol{w}, \boldsymbol{\sigma}),$$

if the following holds.

- For some $k = 1,2,\ldots,r$, $\sigma = v_k$, $\boldsymbol{w} = \boldsymbol{v}$, and $(p,k,q) \in \mu$; or
- $\sigma = \epsilon$, $w_{\rho(p,q)} \in \Sigma \setminus \{v_1, \ldots, v_{\rho(p,q)-1}, v_{\rho(p,q)+1}, \ldots, v_r\}$ and for each $j \neq \rho(p,q)$, $w_j = v_j$.

We say that $\mathcal{A}$ *accepts* a word $\boldsymbol{\sigma} \in \Sigma^*$, if $(s_0, \boldsymbol{u}, \boldsymbol{\sigma}) \vdash^* (f, \boldsymbol{v}, \epsilon)$ for some $\boldsymbol{v} \in (\Sigma \cup \{\#\})^{r\neq}$ and $f \in F$, where, as usual, $\vdash^*$ is the reflexive and transitive closure of $\vdash$.

The set of all words accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

We proceed with examples which illustrate the difference and similarity between NFMA and FMA.

Example 4. (Cf. [4, Example 4].) *Consider a 2-register NFMA* $\mathcal{A}_1 = \langle \{s, p, f\}, s, \#\#, \rho, \mu, \{f\} \rangle$, *where*

- $\rho(s, p) = 1$ *and* $\rho(p, p) = 2$*; and*
- $\mu = \{(s, \epsilon, p), (p, \epsilon, p), (p, 2, p), (p, 1, f)\}$.

Alternatively, $\mathcal{A}_1$ *can be described by the diagram in Fig. 2.*

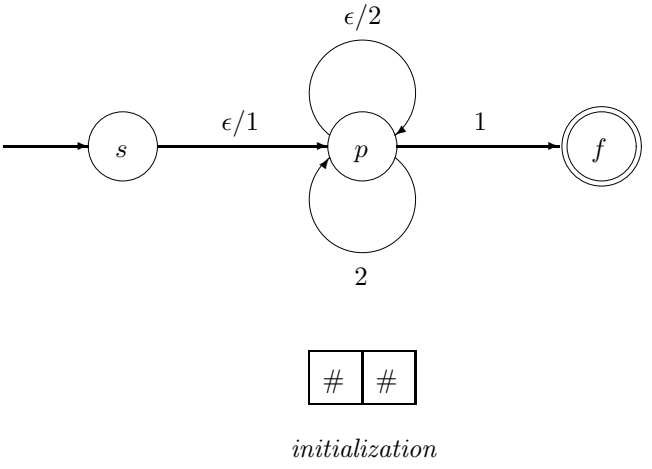

Fig. 2. The NFMA $\mathcal{A}_1$.

It can be easily seen that $L(\mathcal{A}_1)$ *consists exactly of all words whose* last *symbol differs from all the others. That is,*

$$L(\mathcal{A}_1) = \{\sigma_1\sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_n,\ i = 1, 2, \ldots, n-1,\ n = 2, 3, \ldots\}.^{\text{d}}$$

Remark 5. *By [4, Example 4],* $L(\mathcal{A}_1)$ *is not an FMA language.*[e] *This, together with Theorem 7 in the next section, shows that NFMA are more powerful than FMA.*

Example 6. (Cf. [4, Example 1].) *Consider a 2-register NFMA* $\mathcal{A}_2 = \langle \{s, p, q, f\}, s, \#\#, \rho, \mu, \{f\} \rangle$, *where*

- $\rho(p, p) = \rho(q, q) = \rho(f, f) = 1$ *and* $\rho(s, p) = 2$*; and*
- $\mu = \{(s, \epsilon, p), (p, \epsilon, p), (q, \epsilon, q), (f, \epsilon, f)\}$
  $\cup\ \{(p, 1, p), (p, 2, q), (q, 1, q), (q, 2, f), (f, 1, f), (f, 2, f)\}$.

[d] Note that $L(\mathcal{A}_1)$ is the reversal of $L(\mathcal{A})$ from Example 2.
[e] A much more involved argument in [2] shows that the pushdown automata with non-deterministic reassignment introduced in [1] are more powerful than their deterministic reassignment counterpart.

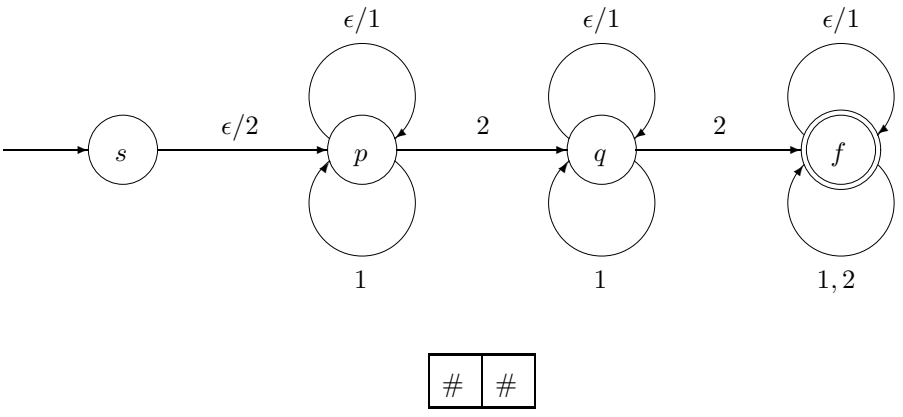

Fig. 3. The NFMA $\mathcal{A}_2$.

Alternatively, $\mathcal{A}_2$ can be described by the diagram in Fig. 3.

One can easily verify that $L(\mathcal{A}_2)$ consists exactly of those words in which some element of Σ occurs twice or more. That is,

$$L(\mathcal{A}_2) = \{\sigma_1\sigma_2\cdots\sigma_n : \textit{there exist } i \textit{ and } j,\ 1 \le i < j \le n,\ \textit{such that } \sigma_i = \sigma_j\}.$$

The behavior of $\mathcal{A}_2$ on such words is as follows.

- *Being in the initial state s, the automaton guesses a symbol that occurs twice or more, stores it in the second register, and enters the state p.*
- *In the state p, $\mathcal{A}_2$ guesses the next input symbol, if it differs from that stored in the second register, stores it in the first register, and then reads it from the input.*
- *When reading the symbol guessed in the beginning of the computation (this symbol should occur twice and is stored in the second register), the automaton enters the state q.*
- *From the state q the automaton proceeds as follows.*
  - *It guesses the next input symbol, stores it in the first register, and then reads it form the input, or*
  - *if the input is the symbol stored in the second register, the automaton changes the state to the final state f, where it stays until the end of the computation.*

For example, $\boldsymbol{\sigma} = abcbd \in L$*, because* b *occurs twice in* $\boldsymbol{\sigma}$*. An accepting run of* $\mathcal{A}_2$ *on* $\boldsymbol{\sigma}$ *is*

$$(s, \#\#, abcbd) \vdash (p, \#b, abcbd) \vdash (p, ab, abcbd) \vdash (p, ab, bcbd) \vdash$$
$$(q, ab, cbd) \vdash (q, cb, cbd) \vdash (q, cb, bd) \vdash (f, cb, d) \vdash (f, db, d) \vdash (f, db, \epsilon).$$

We shall see in the sequel (Example 12) that the complement $\overline{L(\mathcal{A}_2)}$ of $L(\mathcal{A}_2)$ to Σ^* is not an NFMA language.

4. Basic Properties of NFMA

This section contains some basic properties of NFMA. First we show that NFMA are, at least, as powerful as FMA.

Theorem 7. *Each FMA language is accepted by NFMA.*

Proof. Given an FMA $\mathcal{A} = \langle Q, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$ we construct an NFMA $\mathcal{A}'$ such that $L(\mathcal{A}') = L(\mathcal{A})$. Let $\tilde{Q} = \{\tilde{p} : \rho(p) \text{ is defined}\}$ be a set of states disjoint with Q. Then $\mathcal{A}' = \langle Q \cup \tilde{Q}, s_0, \boldsymbol{u}, \rho', \mu', F\rangle$, where ρ' and μ' are defined as follows.

- $\rho'(p, \tilde{p}) = \rho(p)$; and
- $\mu' = \mu_1 \cup \mu_2 \cup \mu_3$, where
  - $\mu_1 = \{(p, k, q) : (p, k, q) \in \mu \text{ and } \rho(p) \neq k\}$,

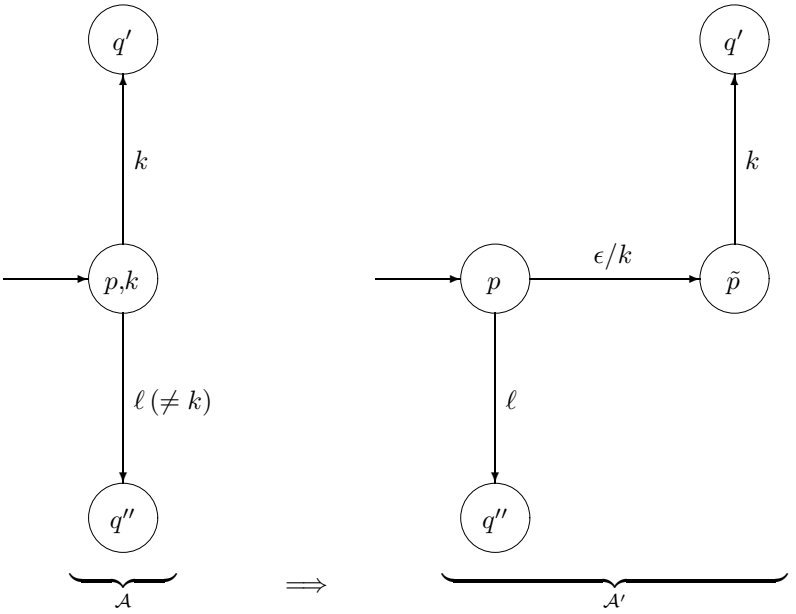

Fig. 4. The FMA to NFMA translation scheme.

- $\mu_2 = \{(p, \epsilon, \tilde{p}) : \tilde{p} \in \tilde{Q}\}$, and
- $\mu_3 = \{(\tilde{p}, k, q) : (p, k, q) \in \mu \text{ and } \rho(p) = k\}$.

That is, we use the translation scheme from the FMA $\mathcal{A}$ to the NFMA $\mathcal{A}'$ as described in Fig. 4.

Now, each run

$$(q_0, \boldsymbol{w}_0, \sigma_1\sigma_2\cdots\sigma_n), (q_1, \boldsymbol{w}_1, \sigma_2\cdots\sigma_n), \ldots, (q_n, \boldsymbol{w}_n, \epsilon)$$

of $\mathcal{A}$ becomes a run of $\mathcal{A}'$ after inserting after each instantaneous description $(q_i, \boldsymbol{w}_i, \sigma_i\sigma_{i+1}\cdots\sigma_n)$ such that $\sigma_i \notin [\boldsymbol{w}_i]$ the instantaneous description $(\tilde{q}_i, \boldsymbol{w}_{i+1}, \sigma_i\sigma_{i+1}\cdots\sigma_n)$. Recall that, by definition, $\boldsymbol{w}_{i+1} = w_{i+1,1}w_{i+1,2}\cdots w_{i+1,r}$ is as follows. Let $\boldsymbol{w}_i = w_{i,1}w_{i,2}\cdots w_{i,r}$. Then

$$w_{i+1,k} = \begin{cases} w_{i,k}, \text{ if } k \neq \rho(q_i) \quad (\text{i.e., } k \neq \rho'(q_i, \tilde{q}_i)) \\ \sigma_i, \quad \text{if } k = \rho(q_i) \quad (\text{i.e., } k = \rho'(q_i, \tilde{q}_i)) \end{cases}, \qquad k = 1, 2, \ldots, r.$$

Conversely, each run of $\mathcal{A}'$ transforms into a run of $\mathcal{A}$ by deleting from it all instantaneous descriptions whose state component belongs to $\tilde{Q}$. Thus, $L(\mathcal{A}) = L(\mathcal{A}')$. □

Corollary 8. *The inclusion problem for NFMA languages is undecidable.*

Proof. The proof follows from Theorem 7 and the fact that the inclusion problem for FMA languages is undecidable, see [9, Section 5]. □

As we have already noted in Remark 5, Theorem 7 and Example 4 show that NFMA are more powerful than FMA. Nevertheless, the former possesses important properties of the latter. We list these properties below. The proofs are almost identical to the proofs of the corresponding properties in [4] and, therefore, are omitted.

Proposition 9. (Cf. [4, Proposition 1].) *Let $\mathcal{A}$ be an NFMA and let Σ' be a finite subset of Σ. Then $\mathcal{A}$ can be effectively converted into a finite automaton $\mathcal{A}'$ over Σ' such that $L(\mathcal{A}') = L(\mathcal{A}) \cap \Sigma'^*$.*

Proposition 10. (Closure under automorphisms, cf. [4, Proposition 2].) *Let $\mathcal{A} = \langle S, s, \boldsymbol{u}, \rho, \mu, F\rangle$ be an NFMA. Then for each automorphism $\iota : \Sigma \to \Sigma$ that is an identity on $[\boldsymbol{u}]$, $\iota(L(\mathcal{A})) = L(\mathcal{A})$.*

Proposition 11 below reflects the fact that an NFMA can only sense the input symbols which occur in the most recent reassignment. It cannot, however, distinguish between different "new" reassignments. This property is a useful tool for proving that a language is not accepted by an NFMA, see Example 12. Note that an occurrence of an input symbol that belonged to some register and was "forgotten" in a later reassignment is also "new."

Proposition 11. (Indistinguishability property of NFMA, cf. [4, Proposition 3].) *Let $\mathcal{A}$ be an r-register NFMA. If $\boldsymbol{\sigma\tau} \in L(\mathcal{A})$, then there exists a subset Σ' of $[\boldsymbol{\sigma}]$ containing at most r elements for which the following holds. For any $\sigma \notin \Sigma'$ and any $\tau \notin [\boldsymbol{\tau}] \cup \Sigma'$, the word $\boldsymbol{\sigma}(\boldsymbol{\tau}(\sigma/\tau))$ obtained from $\boldsymbol{\sigma\tau}$ by the substitution of τ for each occurrence of σ in $\boldsymbol{\tau}$ is in $L(\mathcal{A})$.*

Example 12. (Cf. [4, Example 4].) *Let $\mathcal{A}_2$ be the NFMA from Example 6. Then $\overline{L(\mathcal{A}_2)}$ consists of all words in which each symbol occurs at most one time. Exactly like in [4, Example 4], it can be shown that $\overline{L(\mathcal{A}_2)}$ is not an NFMA language.*

Remark 13. *It follows from Examples 6 and 12 that NFMA languages are not closed under complementation.*

Proposition 14. (Cf. [4, Proposition 4].) *If an r register NFMA accepts a word of length n, then it accepts a word of length n that contains at most r pairwise different symbols.*

Corollary 15. (Cf. [4, Theorem 1].) *The emptiness problem for NFMA languages is decidable.*[f]

We conclude this section with closure properties of NFMA.

Theorem 16. (Cf. [4, Theorem 3].) *The NFMA languages are closed under union, intersection, concatenation, and iteration (Kleene star).*

The proof of Theorem 16 is very similar to that for FMA. It is based on an equivalent model of computation similar to *M-automata* introduced in [4, Section 3], see [12, Section 2.5] for details. As we shall see in the sequel, the NFMA languages are also closed under reversal. This closure property is an immediate corollary to our description of NFMA languages by means of *regular expressions* over infinite alphabets, see Corollary 31 in the beginning of Section 6.

5. Regular Expressions for NFMA Languages

In this section we introduce an alternative description of NFMA languages by the so called *infinite alphabet regular expressions* which are the infinite alphabet counterpart (with respect to NFMA) of the ordinary regular expressions over a finite alphabet.

Definition 17. (Cf. [6, Definition 4.1].) *Let $X = \{x_1, x_2, \ldots, x_r\}$ be a set of variables such that $X \cap \Sigma = \emptyset$ and let Θ be a finite subset of Σ.* Infinite alphabet regular expressions *over (X, Θ), or just infinite alphabet regular expressions (IARE), if (X, Θ) is understood from the context, are defined as follows.*

[f] It easily follows from the definition that the membership problem for NFMA languages is decidable as well.

- $\emptyset, \epsilon$, *and each element of* $X \cup \Theta$ *are IARE.*[g]
- *If* α_1 *and* α_2 *are IARE, then so are* $(\alpha_1 + \alpha_2)$ *and* $(\alpha_1 \cdot \alpha_2)$.
- *If* α *is an IARE, then so is* $(\alpha)^*$.
- *If* $x \in X$, α_1 *and* α_2 *are IARE, then so is* $(\alpha_1 \cdot_x \alpha_2)$.
- *If* $x \in X$ *and* α *is an IARE, then so is* $(\alpha)^{*_x}$.

The intuition underlying the above definition is as follows. Each variable in X corresponds to a register of the NFMA and an assignment of symbols from $(\Sigma \cup \{\#\}) \setminus \Theta$ to the variables corresponds to the content of the automaton registers. The subscripts of '$\cdot$' and '$*$' indicate the registers reassigned by the automaton.

For what follows we observe that an IARE α over (X, Θ) can be thought of as a regular expression $\widehat{\alpha}$ over the (finite) alphabet

$$\widehat{\Sigma} = X \cup \Theta \cup \{\cdot_x : x \in X\}.$$

This *ordinary* regular expression is obtained from α by replacing each its subexpression of the form $(\alpha_1 \cdot_x \alpha_2)$ with $(\alpha_1 \cdot \cdot_x \cdot \alpha_2)$ and each its subexpression of the form $(\alpha)^{*_x}$ with $(\alpha \cdot \cdot_x)^*$. Ordinary regular expressions over $\widehat{\Sigma} = X \cup \Theta \cup \{\cdot_x : x \in X\}$ are transformed into IARE by the converse construction, see the formal definition below.

Definition 18. *With an IARE* α *over* (X, Θ), *we associate the regular expression* $\widehat{\alpha}$ *over* $\widehat{\Sigma} = X \cup \Theta \cup \{\cdot_x : x \in X\}$ *that is recursively defined as follows.*

- *If* $\alpha \in \{\emptyset, \epsilon\} \cup \Theta \cup X$, *then* $\widehat{\alpha} = \alpha$.
- *If* $\alpha = \alpha_1 \otimes \alpha_2$, $\otimes \in \{+, \cdot\}$, *then* $\widehat{\alpha} = \widehat{\alpha_1} \otimes \widehat{\alpha_2}$.
- *If* $\alpha = \alpha_1 \cdot_x \alpha_2$, $x \in X$, *then* $\widehat{\alpha} = \widehat{\alpha_1} \cdot \cdot_x \cdot \widehat{\alpha_2}$. *(Recall that* $\cdot_x \in \widehat{\Sigma}$.*)*
- *If* $\alpha = (\alpha_1)^*$, *then* $\widehat{\alpha} = (\widehat{\alpha_1})^*$.
- *If* $\alpha = (\alpha_1)^{*_x}$, *then* $\widehat{\alpha} = (\widehat{\alpha_1} \cdot \cdot_x)^*$.

Conversely, with a regular expression $\widehat{\alpha}$ *over* $\widehat{\Sigma}$ *we associate the IARE* α *over* (X, Θ) *defined by the recursion below.*

- *If* $\widehat{\alpha} \in \{\emptyset, \epsilon\} \cup \Theta \cup X$, *then* $\alpha = \widehat{\alpha}$.
- *If* $\widehat{\alpha} = \cdot_x$, $x \in X$, *then* $\alpha = \epsilon \cdot_x \epsilon$.
- *If* $\widehat{\alpha} = \widehat{\alpha_1} \otimes \widehat{\alpha_2}$, $\otimes \in \{+, \cdot\}$, *then* $\alpha = \alpha_1 \otimes \alpha_2$.
- *If* $\widehat{\alpha} = (\widehat{\alpha_1})^*$, *then* $\alpha = (\alpha_1)^*$.

Example 19. *Let* $\alpha_1 = y^{*_y} \cdot x$ *be an IARE over* $(\{x, y\}, \emptyset)$. *Then* $\widehat{\alpha_1} = (y \cdot \cdot_y)^* \cdot x$.

Example 20. *Let* $\alpha_2 = y^{*_y} \cdot x \cdot y^{*_y} \cdot x \cdot_x y^{*_y}$ *be an IARE over* $(\{x, y\}, \emptyset)$. *Then* $\widehat{\alpha_2} = (y \cdot \cdot_y)^* \cdot x \cdot (y \cdot \cdot_y)^* \cdot x \cdot \cdot_x \cdot (y \cdot \cdot_y)^*$.

Definition 21. *Let* $\boldsymbol{w} = w_1 w_2 \cdots w_n \in \widehat{\Sigma}^*$, $i = 1, 2, \ldots, n$, *and let* $w_i = x \in X$. *Let* j *be the maximal integer less than* i *such that* $w_j = \cdot_x$, *if exists, and be zero, otherwise. Let* k *be the minimal integer greater than* i *such that* $w_k = \cdot_x$, *if exists,*

[g] Of course ϵ is redundant (but still very useful), because $L(\emptyset^*) = \{\epsilon\}$.

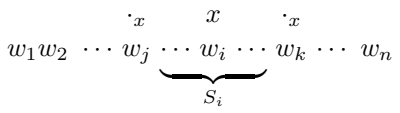

Fig. 5. The scope of ith position in $\boldsymbol{w}$.

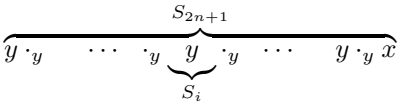

Fig. 6. The scopes of positions in $\boldsymbol{w}_n$.

and be $n+1$*, otherwise. The set of positive integers* $\{j+1, j+2, \ldots, k-1\}$ *is called the* scope *of (the position)* i *and is denoted by* S_i*, see Fig. 5.*

Example 22. *Let* $\boldsymbol{w}_n = (y\cdot_y)^n x \in \widehat{\Sigma}^*$*. Then* $S_{2n+1} = \{1, 2, \ldots, 2n+1\}$*, and, for* $i = 1, 3, \ldots, 2n-1$*,* $S_i = \{i\}$*, see Fig. 6.*

Example 23. *Let* $\boldsymbol{w}_{n_1,n_2,n_3} = (y\cdot_y)^{n_1} x (y\cdot_y)^{n_2} x \cdot_x (y\cdot_y)^{n_3} \in \widehat{\Sigma}^*$*. Then* $S_{2n_1+1} = S_{2n_1+2n_2+2} = \{1, 2, \ldots, 2n_1 + 2n_2 + 2\}$*, and if the symbol at the* i*th position is* y*, then*

$$S_i = \begin{cases} \{2n_1+1, 2n_1+2\}, & \text{if } i = 2n_1+2 \\ \{2n_1+2n_2+2, 2n_1+2n_2+3, 2n_1+2n_2+4\}, & \text{if } i = 2n_1+2n_2+4 \\ \{i\}, & \text{otherwise} \end{cases}$$

see Fig. 7.

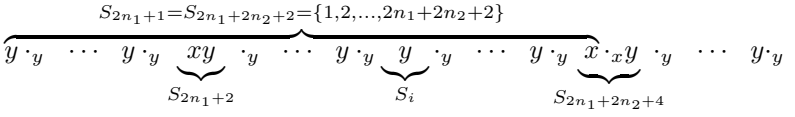

Fig. 7. The scopes of positions in $\boldsymbol{w}_{n_1,n_2,n_3}$.

Next we define the language $L(\alpha)$ generated by an IARE α. The definition is based on the notion of an (NFMA) *instance* of a word over $\widehat{\Sigma}$. It is similar to that for *UB-expressions*, see [6, Section 4].

Definition 24. *Let* $\boldsymbol{w} = w_1 w_2 \cdots w_n \in \widehat{\Sigma}^*$*. With the* i*th symbol* w_i *of* $\boldsymbol{w}$*,* $i = 1, 2, \ldots, n$*, we associate a word* $\underline{w_i} \in \Sigma \cup \{\epsilon\}$ *in the following manner.*

- *If* w_i *is of the form* $\cdot_x$*, where* $x \in X$*, then* $\underline{w_i} = \epsilon$*.*
- *If* $w_i \in \Theta$*, then* $\underline{w_i} = w_i$*.*

- *If $w_i = x \in X$, then $\underline{w_i}$ can be any element of $\Sigma \setminus \Theta$ such that the following condition is satisfied.*
  - *For any $j = 1, 2, \ldots, n$ such that the scope S_j overlaps the scope S_i,*[h] *$\underline{w_i} = \underline{w_j}$ if and only if $w_i = w_j$.*

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

Example 25. *For $\boldsymbol{w}_n = (y\cdot_y)^n x$ from Example 22,*

$$I(\boldsymbol{w}_n) = \{\sigma_1\sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_n,\ i = 1, 2, \ldots, n-1\}.$$

Example 26. *For $\boldsymbol{w}_{n_1,n_2,n_3} = (y\cdot_y)^{n_1} x (y\cdot_y)^{n_2} x \cdot_x (y\cdot_y)^{n_3}$ from Example 23,*

$$I(\boldsymbol{w}_{n_1,n_2,n_3}) = \{\sigma_1\sigma_2 \cdots \sigma_{n_1+n_2+n_3+2} : \sigma_{n_1+1} = \sigma_{n_1+n_2+2} \text{ and } \sigma_{n_1+1} \neq \sigma_i,\ i = 1, 2, \ldots, n_1, n_1+2, \ldots, n_1+n_2+1\}.$$[i]

At last we have arrived at the definition of the language $L(\alpha)$ generated by an IARE α.

Definition 27. *Let α be an IARE over (X, Θ). The language $L(\alpha)$ generated by α is defined by*

$$L(\alpha) = I(L_{\widehat{\Sigma}}(\widehat{\alpha})) = \bigcup_{\boldsymbol{w} \in L_{\widehat{\Sigma}}(\widehat{\alpha})} I(\boldsymbol{w}).$$[j]

That is, the language $L(\alpha)$ is a set of all instances of all elements of $L_{\widehat{\Sigma}}(\widehat{\alpha})$.

Note the similarity between IARE and *UB-expressions* introduced in [6]. The only difference is that in the former '$\cdot$' and '$*$' are subscribed with *elements* of X, whereas in the latter they are subscribed with *subsets* of X. However, breaking a *simultaneous* reset $\cdot_{X'}$, $X' = \{x_1, x_2, \ldots, x_n\}$, into an equivalent sequence of resets $\cdot_{x_1} \epsilon \cdot_{x_2} \cdots \epsilon \cdot_{x_{n-1}} \epsilon \cdot_{x_n}$ (and similarly for '$*$') we may assume that subsets of X in the definition of UB-expressions contain exactly one element. Thus, the difference between the languages generated by IARE and UB-expressions is in the definition of instances of words over $\widehat{\Sigma}$.

Example 28. (Cf. Example 4.) *Let $\alpha_1 = y^{*_y} \cdot x$ be the IARE from Example 19. It follows from Examples 19, 22, and 25 that $I(\alpha_1)$ consists of all words over Σ whose last symbol differs from all the others.*

[h] Of course, by "the scope S_j overlaps the scope S_i" we mean that $S_i \cap S_j \neq \emptyset$.

[i] That is, σ_{n_1+1} and $\sigma_{n_1+n_2+2}$ are the first two occurrences of σ_{n_1+1} $(= \sigma_{n_1+n_2+2})$ in $\sigma_1\sigma_2 \cdots \sigma_{n_1+n_2+n_3+2}$.

[j] Here and hereafter, $L_{\widehat{\Sigma}}(\widehat{\alpha})$ denotes the language over the finite alphabet $\widehat{\Sigma}$ defined by $\widehat{\alpha}$, see Definition 18.

Example 29. (Cf. Example 6.) *Let* $\alpha_2 = y^{*_y} \cdot x \cdot y^{*_y} \cdot x \cdot_x y^{*_y}$ *be the IARE from Example 20. It follows from Examples 20, 23, and 26 that* $I(\alpha_2)$ *consists of all words over* Σ *in which some element of* Σ *occurs twice or more.*

6. Equivalence of IARE and NFMA

This section contains the main result of our paper.

Theorem 30. *A language is generated by an IARE if and only if it is accepted by an NFMA.*

We precede the proof of Theorem 30 the following corollary.

Corollary 31. (Cf. [7, Theorem 1].) *NFMA languages are closed under reversing.*

Proof. By Theorem 30, it suffices to show that the languages defined by IARE are closed under reversing. Let α be an IARE. It can be easily verified that $(L(\alpha))^R = L(\alpha^R)$, where the IARE α^R is defined by the following recursion.

- If $\alpha \in \{\emptyset, \epsilon\} \cup \Theta \cup X$, then $\alpha^R = \alpha$.
- $(\alpha_1 + \alpha_2)^R$ is $(\alpha_1^R + \alpha_2^R)$.
- $(\alpha_1 \cdot \alpha_2)^R$ is $(\alpha_2^R \cdot \alpha_1^R)$.
- $(\alpha_1 \cdot_x \alpha_2)^R$ is $(\alpha_2^R \cdot_x \alpha_1^R)$.
- $((\alpha_1)^*)^R$ is $(\alpha_1^R)^*$.
- $((\alpha_1)^{*_x})^R$ is $(\alpha_1^R)^{*_x}$. □

For the proof of Theorem 30 we shall use the following model of ordinary FA.

Definition 32. *A (non-deterministic) finite state automaton (over* $\widehat{\Sigma}$*) is a system* $\boldsymbol{M} = \langle Q, \widehat{\Sigma}, s_0, \Delta, F \rangle$*, where*

- Q *is a finite set of states;*
- $s_0 \in Q$ *is the initial state;*
- Δ *is a subset of* $Q \times \widehat{\Sigma} \times Q$ *called the* transition *relation; and*
- $F \subseteq Q$ *is the set of final states.*

As usual, for $\boldsymbol{\sigma} \in \widehat{\Sigma}^*$, $\sigma \in \widehat{\Sigma}$, *and* $p, q \in Q$, *we write* $(p, \sigma\boldsymbol{\sigma}) \vdash (q, \boldsymbol{\sigma})$ *if and only if* $(p, \sigma, q) \in \Delta$*; and we say that a word* $\boldsymbol{\sigma} \in \widehat{\Sigma}^*$ *is* accepted *by* $\boldsymbol{M}$ *if and only if* $(s_0, \boldsymbol{\sigma}) \vdash^* (f, \epsilon)$*, for some* $f \in F$.

6.1. *Proof of the "only if" part of Theorem 30*

The proof of the "only if" part of the theorem is based on the equivalence of ordinary regular expression to FA and the translation scheme in Fig. 8.

Let α be an IARE over (X, Θ), $X = \{x_1, x_2, \ldots, x_{r_v}\}$ and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{r_c}\}$. Let $\boldsymbol{M} = \langle Q, \widehat{\Sigma}, s_0, \Delta, F \rangle$ be an FA over $\widehat{\Sigma} = X \cup \Theta \cup \{\cdot_x : x \in X\}$ such that

$$\text{IARE } \alpha \implies \text{RE } \widehat{\alpha} \implies \text{FA } \boldsymbol{M} \implies \text{NFMA } \mathcal{A}_{\boldsymbol{M}}$$

Fig. 8. The IARE to NFMA translation scheme.

$L_{\widehat{\Sigma}}(\widehat{\alpha}) = L(\boldsymbol{M})$.[k]

Consider the NFMA $\mathcal{A}_{\boldsymbol{M}} = \langle Q, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$, such that

- $\boldsymbol{u} = \#^{r_v}\theta_1\theta_2\cdots\theta_{r_c}$;
- $\rho(p,q) = k$ if and only if $(p, \cdot_{x_k}, q) \in \Delta$; and
- $\mu = \mu_1 \cup \mu_2 \cup \mu_3$, where
  - $\mu_1 = \{(p,k,q) : (p, x_k, q) \in \Delta\}$,
  - $\mu_2 = \{(p, r_v + k, q) : (p, \theta_k, q) \in \Delta\}$, and
  - $\mu_3 = \{(p, \epsilon, q) : (p, \cdot_{x_k}, q) \in \Delta\}$.

Remark 33. *Note that the diagrams of $\boldsymbol{M}$ and $\mathcal{A}_{\boldsymbol{M}}$ differ only in the transition labels which can be recovered each from other in a straightforward manner.*

Since $L_{\widehat{\Sigma}}(\widehat{\alpha}) = L(\boldsymbol{M})$, for the proof of the "only if" part of Theorem 30 it suffices to show that $L(\mathcal{A}_{\boldsymbol{M}})$ consists of all instances of all elements of $L(\boldsymbol{M})$.

Let $\boldsymbol{p} = e_1, e_2, \ldots, e_n$ be a path of edges in the diagram of $\mathcal{A}_{\boldsymbol{M}}$. One can think of $\boldsymbol{p}$ as the diagram of the appropriate NFMA, also denoted by $\boldsymbol{p}$. Then $L(\boldsymbol{p})$ consists of all words over Σ which "drive $\mathcal{A}_{\boldsymbol{M}}$ through $\boldsymbol{p}$ from its first vertex (state) to its last vertex (state)."

Let $\boldsymbol{P}$ denote the set of all paths $\boldsymbol{p}$ starting from the initial state s_0 and ending at a final state. Then

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

By Remark 33, $\boldsymbol{p}$ has the corresponding path in $\boldsymbol{M}$ that differs from it only in the transition labels. The sequence of the labels of this corresponding path is a word over $\widehat{\Sigma}$ that we shall denote by $\boldsymbol{w}_{\boldsymbol{p}}$. Consequently,

$$L(\boldsymbol{M}) = \{\boldsymbol{w}_{\boldsymbol{p}} : \boldsymbol{p} \in \boldsymbol{P}\}.$$

Therefore, the equality of $L(\mathcal{A}_{\boldsymbol{M}})$ to the set of all instances of all elements of $L(\boldsymbol{M})$ will follow, if we prove that for each path $\boldsymbol{p}$ (not necessarily in $\boldsymbol{P}$),

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

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

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

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

[k] Recall that $\widehat{\alpha}$ is an ordinary regular expression associated with α, see Definition 18.

Let $\boldsymbol{p}'$ be a path of length $n+1$. Then $\boldsymbol{p}' = \boldsymbol{p}, (p, k, q)$, where $\boldsymbol{p}$ is a path of length n. Let $\boldsymbol{w_p} = w_1 w_2 \cdots w_n$, and let $\boldsymbol{w_{p'}} = \boldsymbol{w_p} \cdot w_{n+1}$. We shall distinguish between the cases of $k = \epsilon$ and $k \neq \epsilon$.

Let $k = \epsilon$. Then $L(\boldsymbol{p}') = L(\boldsymbol{p})$. Also, by the definition of $\mathcal{A}_{\boldsymbol{M}}$, $w_{n+1} = \cdot_{x_k}$, for some $k = 1, 2, \ldots, r_v$. Therefore, by the definition of an instance of a word over $\widehat{\Sigma}$ (Definition 24), $I(\boldsymbol{w_{p'}}) = I(\boldsymbol{w_p})\{\epsilon\} = I(\boldsymbol{w_p})$. Since, by the induction hypothesis, $I(\boldsymbol{w_p}) = L(\boldsymbol{p})$, $I(\boldsymbol{w_{p'}}) = L(\boldsymbol{p}')$ follows.

Let $k \neq \epsilon$. If $r_v < k \leq r_v + r_c$, then, by the definition of $\boldsymbol{u}$, $w_{n+1} = \theta_{k-r_v}$. Therefore, $L(\boldsymbol{p}') = L(\boldsymbol{p})\{\theta_{k-r_v}\}$. Also, by the definition of an instance of a word over $\widehat{\Sigma}$, $I(\boldsymbol{w_{p'}}) = I(\boldsymbol{w_p})\{\theta_{k-r_v}\}$. Since, by the induction hypothesis, $I(\boldsymbol{w_p}) = L(\boldsymbol{p})$, $I(\boldsymbol{w_{p'}}) = L(\boldsymbol{p}')$ follows.

If $1 \leq k \leq r_v$, then, by the definition of $\mathcal{A}_{\boldsymbol{M}}$, $w_{n+1} = x_k$. Let $\ell \leq n$ to be the greatest index such that w_ℓ is x_k, if such index exists, and be 0, otherwise. That is, ℓ is the last time the kth register occurs in the path $\boldsymbol{p}$. Also, let $m \leq n$ be the greatest index such that $w_m = \cdot_{x_k}$, if such index exists, and be 0, otherwise. That is, m is the last time the kth register is (non-deterministically) reassigned by $\boldsymbol{p}$. We shall distinguish between the cases of $\ell > m$ and $\ell \leq m$.

Assume first that $\ell > m$. By the definition of an instance of a word over $\widehat{\Sigma}$, the symbol assigned to x_k at the ℓth position (whose scope is $\{m+1, m+2, \ldots, n+1\}$) must be assigned to it again at the $(n+1)$th position, because the scopes S_ℓ and S_{n+1} overlap.[l] That is, $\underline{w_1}\,\underline{w_2} \cdots \underline{w_n}\,\underline{w_{n+1}} \in I(\boldsymbol{w_{p'}})$ if and only if $\underline{w_1}\,\underline{w_2} \cdots \underline{w_n} \in I(\boldsymbol{w_p})$ and $\underline{w_{n+1}} = \underline{w_\ell}$.

Similarly, the kth register of the NFMA $\boldsymbol{p}'$ is used in the ℓth transition and is not reassigned anymore in the computation. Thus, a word is accepted by the NFMA $\boldsymbol{p}'$ if and only if it is accepted by the NFMA $\boldsymbol{p}$ and the symbol that occurs in the ℓth position of the word also occurs in its $(n+1)$th position. Since, by the induction hypothesis, $I(\boldsymbol{w_p}) = L(\boldsymbol{p})$, $I(\boldsymbol{w_{p'}}) = L(\boldsymbol{p}')$ follows.

Assume now that $\ell \leq m$. Then $\underline{w_1}\,\underline{w_2} \cdots \underline{w_n}\,\underline{w_{n+1}} \in I(\boldsymbol{w_{p'}})$ if and only if $\underline{w_1}\,\underline{w_2} \cdots \underline{w_n} \in I(\boldsymbol{w_p})$ and for each $i = 1, 2, \ldots, n$ the following holds.

- If $w_i \in X$, $w_i \neq x_k$, and S_i overlaps S_{n+1}, then $\underline{w_{n+1}} \neq \underline{w_i}$.

Similarly, the kth register of the NFMA $\boldsymbol{p}'$ was reassigned in the mth move for the last time. Therefore, $\sigma_1\sigma_2 \cdots \sigma_n\sigma_{n+1} \in L(\boldsymbol{p}')$ if and only if $\sigma_1\sigma_2 \cdots \sigma_n \in L(\boldsymbol{p})$ and for each $i = 1, 2, \ldots, n$ the following holds.[m]

- If $\sigma_i \in \Sigma$, σ_i is not stored in the kth register, and the register containing σ_i is not reassigned before the mth move of $\boldsymbol{p}'$, then $\sigma_{n+1} \neq \sigma_i$.

[l] In fact, they are the same.
[m] Of course, we assume that the ith transition of the NFMA $\boldsymbol{p}'$ is on $\sigma_i \in \Sigma \cup \{\epsilon\}$, $i = 1, 2, \ldots, n+1$.

By the definition of $\boldsymbol{w}_{\boldsymbol{p}'}$, σ_i is not stored in the kth register if and only if $w_i \neq x_k$; and the register containing σ_i is not reassigned before the mth move of $\boldsymbol{p}'$ if and only if S_i overlaps S_{n+1}. Also, by the induction hypothesis, $I(\boldsymbol{w}_{\boldsymbol{p}}) = L(\boldsymbol{p})$. Therefore, $I(\boldsymbol{w}_{\boldsymbol{p}'}) = L(\boldsymbol{p}')$ follows.

This completes the proof of the induction hypothesis and the "only if" part of the theorem.

6.2. *Proof of the "if" part of Theorem 30*

The proof of the "if" part of the theorem involves a kind of a "normalization" of NFMA given by Lemma 34 below.

Lemma 34. ([12, Lemma 1, p. 15]) *Let $\mathcal{A} = \langle Q, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$ be an r-register NFMA. Then $L(\mathcal{A})$ is accepted by an NFMA $\mathcal{A}' = \langle Q', s_0', \boldsymbol{u}', \rho', \mu', F'\rangle$ such that $\boldsymbol{u}' \in \{\#^r\}\Sigma^{r\neq}$ and ρ' is a function into $\{1, 2, \ldots, r\}$.*[n]

The proof of Lemma 34 can be found in [12] or obtained by a straightforward modification from the proof of [6, Lemma 5.1], and, therefore, is omitted.

The proof of the "if" part of the theorem is similar to the proof of its "only if" part. It is based on the translation scheme in Fig. 9.[o]

$$\text{NFMA } \mathcal{A} \implies \text{FA } \boldsymbol{M}_{\mathcal{A}} \implies \text{RE } \widehat{\alpha} \implies \text{IARE } \alpha.$$

Fig. 9. The NFMA to IARE translation scheme.

So, given an NFMA $\mathcal{A} = \langle Q, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$, by Lemma 34, we may assume that $\boldsymbol{u} = \#^{r_v}\theta_1\theta_2\cdots\theta_{r_c}$. Let $X = \{x_1, x_2, \ldots, x_{r_v}\}$ be a set of variables such that $X \cap \Sigma = \emptyset$ and let $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{r_c}\}$.

Consider an FA $\boldsymbol{M}_{\mathcal{A}} = \langle Q, \widehat{\Sigma}, s_0, \Delta, F\rangle$, $\widehat{\Sigma} = X \cup \Theta \cup \{\cdot_x : x \in X\}$, such that $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3$, where

- $\Delta_1 = \{(p, x_k, q) : (p, k, q) \in \mu \text{ and } 1 \leq k \leq r_v\}$;
- $\Delta_2 = \{(p, \theta_{k-r_v}, q) : (p, k, q) \in \mu \text{ and } r_v < k \leq r\}$; and
- $\Delta_3 = \{(p, \cdot_{x_k}, q) : (p, \epsilon, q) \in \mu \text{ and } \rho(p, q) = k\}$.

Note that the diagrams of $\mathcal{A}$ and $\boldsymbol{M}_{\mathcal{A}}$ differ only in the transition labels which can be recovered each from other in a straightforward manner, cf. Remark 33.

Let $\widehat{\alpha}$ be an ordinary regular expression over $\widehat{\Sigma}$ such that $L_{\widehat{\Sigma}}(\widehat{\alpha}) = L(\boldsymbol{M}_{\mathcal{A}})$, and let α be the IARE over (X, Θ) associated with $\widehat{\alpha}$ (see Definition 18).

[n] That is, the first r registers of $\mathcal{A}'$ are empty, and its last *non-empty* r registers cannot be reassigned. Note that the NFMA $\mathcal{A}_{\boldsymbol{M}}$ constructed in the proof of the "only if" part of the theorem is of such a form.

[o] This scheme is the reversal of the scheme in Fig. 8.

By definition,

$$L(\alpha) = I(L_{\widehat{\Sigma}}(\widehat{\alpha})) = I(L(\boldsymbol{M}_{\mathcal{A}})).$$

Therefore, the proof will be complete if we show that

$$I(L(\boldsymbol{M}_{\mathcal{A}})) = L(\mathcal{A}). \tag{21}$$

It follows from the proof of the "only if" part of the theorem that

$$I(L(\boldsymbol{M}_{\mathcal{A}})) = L(\mathcal{A}_{\boldsymbol{M}_{\mathcal{A}}}), \tag{22}$$

and, by the definition of the NFMA $\mathcal{A}_{\boldsymbol{M}}$ in the proof of the "only if" part of the theorem and the definition of $\boldsymbol{M}_{\mathcal{A}}$ above,

$$\mathcal{A}_{\boldsymbol{M}_{\mathcal{A}}} = \mathcal{A}. \tag{23}$$

Thus, (21) follows from (22) and (23), and the proof is complete.

Acknowledgments

The authors are grateful to Tony Tan for reading the final version of the paper.

This research was supported by the S. and N. Grand Research Fund, by the fund for the promotion of research at the Technion, and by a grant from the Software Technology Laboratory (STL) in the Department of Computer Science, the Technion.

References

[1] E. Y. C. Cheng and M. Kaminski, "Context-free languages over infinite alphabets", *Acta Informatica* **35** (1998) 245–267.

[2] Y. Dubov and M. Kaminski, "The power of non-deterministic reassignment in infinite-alphabet pushdown automata", in *Languages: from Formal to Natural. Essays dedicated to Nissim Francez on his 65th Birthday*, eds. O. Grumberg, M. Kaminski, S. Katz and S. Wintner (Springer, Berlin, 2009), volume 5533 of *Lecture Notes in Computer Science*, pp. 107–127.

[3] M. Kaminski and N. Francez, "Finite-memory automata", in *Proceedings of the 31th Annual IEEE Symposium on Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1990), pp. 683–688.

[4] M. Kaminski and N. Francez, "Finite-memory automata", *Theoretical Computer Science* **134** (1994) 329–363.

[5] M. Kaminski and T. Tan, "Regular expressions for languages over infinite alphabets", in *Computing and Combinatorics, 10th Annual International Conference, COCOON 2004*, eds. K.-Y. Chwa and J. I. Munro (Springer, Berlin, 2004), volume 3106 of *Lecture Notes in Computer Science*, pp. 171–178.

[6] M. Kaminski and T. Tan, "Regular expressions for languages over infinite alphabets", *Fundamenta Informaticae* **69** (2006) 301–318.

[7] M. Kaminski and T. Tan, "Tree automata over infinite alphabets", in *Pillars of Computer Science – Essays Dedicated to Boris (Boaz) Trakhtenbrot*, eds. A. Avron, N. Dershowitz and A. Rabinovich (Springer, Berlin, 2008), volume 4800 of *Lecture Notes in Computer Science*, pp. 386–423.

[8] F. Neven, T. Schwentick and V. Vianu, "Towards regular languages over infinite alphabets", in *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001*, eds. J. Sgall, A. Pultr and P. Kolman (Springer, Berlin, 2001), volume 2136 of *Lecture Notes in Computer Science*, pp. 560–572.

[9] F. Neven, T. Schwentick and V. Vianu, "Finite state machines for strings over infinite alphabets", *ACM Transactions on Computational Logic* **5** (2004) 403–435.

[10] Y. Shemesh and N. Francez, "Finite-state unification automata and relational languages", *Information and Computation* **114** (1994) 192–213.

[11] A. Tal, "Decidability of inclusion for unification based automata", Master's thesis, Department of Computer Science, Technion - Israel Institute of Technology, 1999.

[12] D. Zeitlin, "Look-ahead finite-memory automata", Master's thesis, Department of Computer Science, Technion - Israel Institute of Technology, 2006.