Regular Expressions for Languages over Infinite Alphabets

(Extended abstract)

Michael Kaminski[1] and Tony Tan[2]

[1] Department. of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel. kaminski@cs.technion.ac.il.
[2] School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543. tantony@comp.nus.edu.sg.

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 acceptable by *finite-state unification based automaton* – a model of computation that is tightly related to other models of automata over infinite alphabets.

1 Introduction

A new model of a finite-state automata dealing with *infinite alphabets*, called *finite-state datalog automata* (FSDA) was introduced in [7]. FSDA 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 ordinary finite-state automata (FA) was replaced with *unification*, which is a crucial element of relational languages.

Later, FSDA were extended in [3] to a more general model dealing with infinite alphabets, called *finite-memory automata* (FMA). FMA were designed to accept the counterpart of regular languages over infinite alphabets. 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 values 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 posses 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 [6].

An extension of FSDA to a general infinite alphabet called *finite-state unification based automata*, (FSUBA) was proposed in [8]. These automata are similar

in many ways to FMA, but are a bit weaker, because a register of FSUBA may contain values currently stored in other registers. It was shown in [8] 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 [3] and [4] 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 [6], 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 [1], 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 [2] 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 acceptable by an FSUBA.

The paper is organized as follows. In the next section we recall the definition of FSUBA and in Section 3 we present the main result of our paper – *unification based regular expressions* for languages over infinite alphabets, whose equivalence to FSUBA is proven in the Section 4.

2 Finite-state Unification Based Automata

Till the end of this paper Σ is an infinite alphabet not containing # that is reserved to denote an empty register. 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 $[\boldsymbol{w}] = \{w_i \neq \# : i = 1, 2, \ldots, r\}$. That is, $[\boldsymbol{w}]$ consists of all symbols of Σ which appear in $\boldsymbol{w}$.

Definition 1. *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 $u_i = \#$, then the ith register is empty.*

– $\Theta \subseteq [\boldsymbol{u}]$ *is the "read only" alphabet whose symbols cannot be copied into empty registers.*[1] *One may think of* Θ *as a set of the language constants which cannot be unified, cf [7].*
– $\mu \subseteq Q \times \{1, 2, \ldots, r\} \times 2^{\{1,2,\ldots,r\}} \times Q$ *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 the symbol* σ *and there is a transition* $(q, m, S, q') \in \mu$ *such that the* m*th register either contains* σ *or is empty, then the automaton can enter state* q'*, write* σ *in the* m*th register (if it is empty), and erase the content of the registers whose indices belong to* S*. The* m*th register will be referred to as the* transition *register.*

Like in the case of FSDA, an actual state of $\boldsymbol{A}$ is a state of Q together with the contents of all its registers. That is, $\boldsymbol{A}$ has infinitely many states which are pairs $(q, \boldsymbol{w})$, where $q \in Q$ and $\boldsymbol{w}$ is a word of length r – the content of the registers of $\boldsymbol{A}$. These pairs are called *configurations* of $\boldsymbol{A}$. The set of all configurations of $\boldsymbol{A}$ is denoted Q^c. The pair $(q_0, \boldsymbol{u})$, denoted q_0^c, is called the *initial* configuration,[2] 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 $(q, m, S, q') \in \mu$ such that the following conditions are satisfied.

– Either $w_m = \#$ (i.e., the transition register is empty in which case σ is copied into it) and $\sigma \notin \Theta$, or $w_m = \sigma$ (i.e., the transition register contains σ).
– If $m \notin S$, then $w'_m = \sigma$, 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 $j \notin S \cup \{m\}$, $w'_j = w_j$.

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.

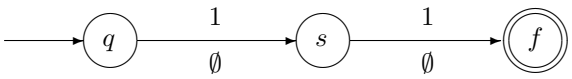
Example 1. ([8]) Let $\boldsymbol{A} = (\Sigma, \{q, s, f\}, q, \{f\}, \#, \emptyset, \mu)$ be a one-register FSUBA, where μ consists of the following two transitions:

– $(q, 1, \emptyset, s)$
– $(s, 1, \emptyset, f)$.

[1] 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.

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

Alternatively, μ can be described by the following diagram.



It can be easily seen that $L(\boldsymbol{A}) = \{\sigma_1\sigma_2 \in \Sigma^2 : \sigma_1 = \sigma_2\}$: an accepting run of $\boldsymbol{A}$ on the word $\sigma\sigma$ 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.[3] 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 m 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 $\sigma_1 \neq \sigma_2$, $w_{1,m} = \#$ and $w_{2,m} = \sigma_2$. Then, replacing $w_{2,m}$ 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 $L = L(\boldsymbol{A})$.[4]

Example 2. ([8]) 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 $(q, m, S, q') \in \mu$, $S = \emptyset$. 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. [4, Proposition 1].

We conclude this section with the observation that FSUBA languages are closed under union, intersection, concatenation, and the Kleene star.[5] The proof of the closure under union and intersection[6] is based on an alternative equivalent model of computation called FSUBA *with multiple assignment* that is similar to M-automata introduced in [4], and for the proof of the closure under concatenation and Kleene star we show that FSUBA with ϵ-transitions can be simulated by ordinary FSUBA.

3 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.

[3] It can be readily seen that L is accepted by a *finite-memory* automaton introduced in [3].

[4] The decision procedure for the inclusion of FSUBA-languages in [8] is based on a refined version of this argument.

[5] [8] deals with the inclusion problem only and does not address the closure properties of FSUBA languages at all.

[6] It follows from Example 1 that FSUBA languages are not closed under complementation.

Definition 2. *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.*

- $\emptyset, \epsilon$,[7] *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 is $(\alpha_1 \cdot_{X'} \alpha_2)$.*
- *If α_1 is a UB-expression, then so is $(\alpha_1)^*$.*

Remark 1. A unification based regular expressions α over (X, Θ) can be thought of as an ordinary regular expression over (finite) alphabet $X \cup \Theta \cup \{\cdot_{X'} : X' \subseteq X\}$, that we shall also denote by α. That is, the *ordinary* regular expression α is obtained from the corresponding UB-expression by replacing its each subexpression of the form $\alpha_1 \cdot_{X'} \alpha_2$ with $\alpha_1 \cdot \cdot_{X'} \cdot \alpha_2$. We leave the formal definition to the reader.

Next we define the language generated by a UB-expression. Let α be a UB-expression over (X, Θ) and let $\boldsymbol{w} = w_1 w_2 \cdots w_n \in L_{X \cup \Theta \cup \{\cdot_{X'} : X' \subseteq X\}}(\alpha)$.[8] With the ith symbol w_i of $\boldsymbol{w}$, $i = 1, 2, \ldots, n$, we associate a word $\overline{w_i} \in \Sigma \cup \{\epsilon\}$ in the following manner.

- If $w_i \in \Theta$, then $\overline{w_i} = w_i$.
- If w_i is of the form $\cdot_{X'}$, where $X' \subseteq X$, then $\overline{w_i} = \epsilon$.
- If $w_i \in X$, then $\overline{w_i}$ can be any element of $\Sigma \setminus \Theta$ such that the following condition is satisfied.
  Let $w_i = x \in X$ and let j be the minimal integer greater than i such that $w_j = x$. If there is no k, $i < k < j$, such that w_k is of the form $\cdot_{X'}$ and $x \in X'$,[9] then $\overline{w_i} = \overline{w_j}$.

The word $\overline{\boldsymbol{w}} = \overline{w_1 w_2} \cdots \overline{w_n}$, where $\overline{w_i}$ is as defined above, $i = 1, 2, \ldots, n$, is called an *instance* of $\boldsymbol{w}$. We denote by $L(\alpha)$ the set of all instances of all elements of $L_{X \cup \Theta \cup \{\cdot_{X'} : X' \subseteq X\}}(\alpha)$.

Example 3. Consider a UB-expression $\alpha = x \cdot_\emptyset (y \cdot_{\{y\}} \epsilon)^* \cdot_\emptyset x$ over $(\{x, y\}, \emptyset)$. It can be easily seen that $L(\alpha)$ consists of all words over Σ having the same first and last symbols.

Similarly, for a UB-expression $x \cdot_\emptyset x$ over $(\{x\}, \emptyset)$, $L(x \cdot_\emptyset x)$ is the language from Example 1.

Theorem 1. *A language is defined by a UB-expression if and only if it is accepted by an FSUBA.*

---

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

[8] Here and hereafter, $L_{X \cup \Theta \cup \{\cdot_{X'} : X' \subseteq X\}}(\alpha)$ denotes the language over $X \cup \Theta \cup \{\cdot_{X'} : X' \subseteq X\}$ defined by α, see Remark 1.

[9] That is, no subscript X' that appears between the ith and the jth positions of $\boldsymbol{w}$ contains x.

The proof of the "only if" part of the theorem is based on the closure properties of FSUBA-languages and is quite standard. The proof of the "if" part of Theorem 1 is presented in the next section.

We conclude this section with one more closure property of FSUBA-languages that is an immediate corollary to Theorem 1.

Corollary 1. *FSUBA languages are closed under reversing.*[10]

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 $(\alpha_2^R \cdot_{X'} \alpha_1^R)$.
- $((\alpha)^*)^R$ is $(\alpha^R)^*$.

4 Proof of the "if" Part of Theorem 1

The proof of the "if part" of Theorem 1 is based on a tight relationship between FSUBA and ordinary FA. We shall use the following model of FA.

Definition 3. *([5]) 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 $\boldsymbol{w} = u_1 \cdots u_n$ of $\boldsymbol{w}$, $u_i \in \Sigma'^*$, 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 $i = 0, 1, \ldots, n-1$, $(s_i, u_{i+1}, s_{i+1}) \in \Delta$.

Let $\boldsymbol{A} = (\Sigma, Q, q_0, F, \boldsymbol{u}, \Theta, \mu)$ be an FSUBA with r registers. By [8, Claim 2], we may assume that $[\boldsymbol{u}] = \Theta$. Also, changing the order of registers, if necessary, we may assume that $\boldsymbol{u} = \theta_1 \cdots \theta_m \#^n$, where $n + m = r$. Consider a finite state automaton $\boldsymbol{M}^{\boldsymbol{A}} = (Q, q_0, F, \Delta)$ over $\{x_1, \ldots, x_n\} \cup \Theta \cup \{\cdot_{X'}\epsilon : X' \subseteq \{x_1, \ldots, x_n\}\}$, where transition relation Δ is defined as follows. For every transition $(q, k, S, q') \in \mu$, Δ contains:

- $(q, \theta_k \cdot_{X'} \epsilon, q')$, if $1 \leq k \leq m$, or
- $(q, x_{(k-m)} \cdot_{X'} \epsilon, q')$, if $m + 1 \leq k \leq r$,

[10] It should be pointed out that FMA languages are not closed under reversing, see [4, Example 8]. The proof of the corollary shows that this constitutes rather serious obstacle to find a kind of regular expressions for FMA languages.

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

Remark 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 1 is based on the fact that set of the sequences of labels of the accepting paths $\boldsymbol{M^A}$ is regular. Namely, the "if" part of Theorem 1 immediately follows from Theorem 2 below.

Theorem 2. *Let $\boldsymbol{A}$ and $\boldsymbol{M^A}$ be as above and let α be an ordinary regular expression that defines $L(\boldsymbol{M^A})$.*[11] *Then, $L(\alpha) = L(\boldsymbol{A})$.*

Proof. (Sketch) Since $L_{\{\theta_1,\ldots,\theta_m,x_1,\ldots,x_n\}\cup\{\cdot_{X'}\in X'\subseteq\{x_1,\ldots,x_n\}\}}(\alpha) = L(\boldsymbol{M^A})$, it suffices to 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 graph representation 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 the 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 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.

The labels of $\boldsymbol{p}$ in the graph representation of $\boldsymbol{M^A}$ form a UB-expression that we shall denote by $\boldsymbol{w_p}$. Therefore,

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

Consequently the equality of $L(\boldsymbol{A})$ to the set of all instances of the elements of $L(\boldsymbol{M^A})$ will follow if we prove that for each path $\boldsymbol{p}$

$$L(\boldsymbol{w_p}) = L(\boldsymbol{p}), \tag{1}$$

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

The proof of (1) is by induction on the length n of $\boldsymbol{p}$ and is omitted.

References

1. M. Bielecki, J. Hidders, J. Paredaens, J. Tyszkiewicz, and J. Van den Bussch. Navigating with a browser. In P. Widmayer, F. Triguero, R. Morales, M. Hennessyd, S. Eidenbenz, and R. Conejo, editors, *Proceedings of the 29th International Colloquium on Automata, Languages and Programming – ICALP 2002*, pages 764–775, Berlin, 2002. Springer. Lecture Notes in Computer Science 2380.
2. B. Bolling, M. Leucker, and T. Noll. Generalized regular MSA languages. Technical report, Department of Computer Science, Aachen University of Technology, 2002.

[11] Recall that, by Remark 1, α is a UB-expression as well.

3. M. Kaminski and N. Francez. Finite-memory automata. In *Proceedings of the 31th Annual IEEE Symposium on Foundations of Computer Science*, pages 683–688, Los Alamitis, CA, 1990. IEEE Computer Society Press.
4. M. Kaminski and N. Francez. Finite-memory automata. *Theoretical Computer Science A*, 138:329–363, 1994.
5. H.R. Lewis and C.H. Papadimitriou. *Elements of the theory of computation.* Prentice-Hall, Inc., Englewood Cliffs, NJ, USA, 1981.
6. F. Neven, T. Schwentik, and V. Vianu. Towards regular languages over infinite alphabets. In J. Sgall, A. Pultr, and P. Kolman, editors, *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*, pages 560–572, Berlin, 2001. Springer. Lecture Notes in Computer Science 2136.
7. Y. Shemesh and N. Francez. Finite-state unification automata and relational languages. *Information and Computation*, 114:192–213, 1994.
8. A. Tal. Decidability of inclusion for unification based automata. M.Sc. thesis, Department of Computer Science, Technion – Israel Institute of Technology, 1999.