

Representing automata and languages with dioid algebra

Eduard Montgomery Meira Costa¹, Antonio Marcus Nogueira
Lima²

¹ Departamento de Engenharia Elétrica

Escola Politécnica

Universidade Federal da Bahia

Aristides Novis, 2 - Federação

40210-630 Salvador, BA, Brazil

e-mail: eduard@ig.com.br

² Departamento de Engenharia Elétrica

Universidade Federal de Campina Grande

Avenida Aprígio Veloso, 882 - Bodocongó

58109-970 Campina Grande, PB, Brazil

e-mail: amnlima@dee.ufcg.edu.br

Received: date / Revised version: date

Abstract In this paper is presented a formulation based on dioid algebra and in the incidence matrices for transition function of finite automata which have not self-loops. This formulation makes feasible the determination of a

state of a finite automaton being given as input the actual state and a string. For this procedure, any specific definitions are introduced.

1 Introduction

Automata are mathematical models used to represent state machines that recognize strings of a given alphabet [1]. The symbols are read sequentially and in this way, an automaton can be viewed as a control entity that have an internal variable that represents its state. Thus, each symbol read results in updating this variable in accordance with the transition function that associates a new state for each pair $(event, state)$ [2]. The set of all automaton states is represented by Q . The initial state of an automaton is designated by q_0 . The marked states, represented by the set $Q_m \subseteq Q$, are the states which the automaton reach when recognizes strings. The automata can be classified as finite-state, infinite-state, deterministic or non-deterministic.

In automata theory, the state changes that occurs after reading symbols are defined by the transition function δ ; the transition function is normally represented as a transition table. To determine the state reached by the automaton when a string is read the transition function is employed recursively.

This article proposes the use of dioid algebra [3, 4] and formal series [5, 6] framework as an alternative way to represent automata, its transition function as well its associated languages. In this case, the automaton is

represented by an incidence matrix and then the transition function as well its languages can be described through simple matrix operations.

The paper is organized as follows: in Section 2 the basic definitions are presented; in Section 3 the representation of the automata by incidence matrices and the definition of the automata languages using the dioid algebra are presented; in Section 4 the the definition of transition function and some examples to illustrate the use of the proposed formalism are presented; in Section 5 the main conclusions are presented.

2 Preliminaries

We recall in this section 2 some basic de notions of automata and languages.

2.1 Formal Languages and Automata

The alphabet of the formal languages is usually represented by the Greek character Σ . The concatenation of symbols of the alphabet defines a string or a word of the formal language. This operation is defined by $cat_{\Sigma} : \Sigma \times \Sigma \rightarrow \Sigma^*$, where Σ^* is the set of all strings that can be defined with symbols from Σ . The length of a string s is denoted by its cardinality $|s|$, which is equal to number of symbols that constitutes the string.

The empty string, represented by ε , is the unique string which length is zero, that is, $|\varepsilon| = 0$. In this way, $\varepsilon \notin \Sigma$ because it is a string, and not a symbol of the alphabet Σ .

The concatenation operation can be extended to strings as shown below

$cat_s : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$, where

$$\begin{aligned} cat_s(\varepsilon, s) &= cat_s(s, \varepsilon) = s, \quad s \in \Sigma^* \\ cat_s(s_1, s_2) &= s_1 s_2 = s, \quad s_1, s_2 \in \Sigma^+. \end{aligned} \tag{1}$$

It is worth noting that the empty string ε is the identity element to the operation of concatenation, that is, all string s concatenated with an empty string ε is always equal to same string.

A set of strings consisting of symbols of a given alphabet Σ , is defined as a language, that is:

Definition 1 *Given an alphabet Σ , L is a language over Σ if and only if, $L \subseteq \Sigma^*$.*

An initial portion, of arbitrary length, of a given string s is denoted as prefix of s . This can be stated formally as

Definition 2 *The prefix of a string s over an alphabet Σ is any string $u \in \Sigma^*$ that can be completed with another string $v \in \Sigma^*$ to build the string s .*

The set that includes all the strings of a given language $L \subseteq \Sigma^*$ as well as all its prefixes is denoted the prefix-closure of L .

Definition 3 *The prefix-closure of L , is defined by:*

$$\overline{L} = \{u \mid \exists v \in \Sigma^* \wedge uv \in L\}. \tag{2}$$

From this definition we can easily see that $L \subseteq \overline{L}$. A language is said to be prefix-closed if and only if $L = \overline{L}$. The *Kleene*-closure of a given language is defined by

$$L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \dots \quad (3)$$

which is an idempotent operation, that is, $(L^*)^* = L^*$.

The union and intersection of two languages as well as the complement of a given language are defined as follows.

Union

$$L_1 \cup L_2 = \{s \mid s \in L_1 \vee s \in L_2\}; \quad (4)$$

Intersection

$$L_1 \cap L_2 = \{s \mid (s \in L_1) \wedge (s \in L_2)\} \quad (5)$$

Complement

$$L^c = \{s \in \Sigma^* \mid s \notin L\}. \quad (6)$$

The concatenation operation can be extended to be used with languages as $cat_L : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$, where

$$\begin{aligned} cat_L(\{\varepsilon\}, L) &= cat_L(L, \{\varepsilon\}) = L, L \subset \Sigma^* \\ cat_L(L_1, L_2) &= \{s_1 s_2 \mid s_1 \in L_1 \wedge s_2 \in L_2\}. \end{aligned} \quad (7)$$

Regular languages can also be expressed by using regular expressions. The basic regular expressions are: $\sigma^{\{n,*\}}$ representing the repetition of the symbol σ by a given number of times (n) or by an arbitrary number of times ($*$), respectively; $s^{\{n,*\}}$ representing the repetition of a string s by a given number of times (n) or by an arbitrary number of times ($*$), respectively; the

symbol $+$ denoting the logic *or* operation, indicating an option between two or more possibilities. Thus, the regular expressions are defined recursively in the following manner:

1. \emptyset is a regular expression denoting the empty set; ε is a regular expression denoting the set $\{\varepsilon\}$ and σ is a regular expression denoting the set $\{\sigma\}$ for all $\sigma \in \Sigma$;
2. If e_1 and e_2 are regular expressions, then e_1e_2 , e_1^* , e_2^* and $(e_1 + e_2)^*$ are regular expressions;
3. All other regular expression can be build by using the rules 1 and 2 a finite number of times.

The regular expressions provide a compact representation for the complex languages. The empty string ε and the empty language \emptyset , are also considered in the regular expressions. The empty string and the empty language have the following properties [7]: $\varepsilon s = s\varepsilon = s$, $\varepsilon^* = \varepsilon$, and $\emptyset + L = L$, $\emptyset L = L\emptyset = \emptyset$ and $\emptyset^* = \varepsilon$.

Formal languages can also be represented graphically by automata as shown in Fig. 1. A deterministic finite-state automaton is represented by a graph consisting of a finite set of nodes connected through arcs. The arcs are labelled with symbols of an alphabet; among all the arcs leaving a given node only one must be labelled with a given symbol.

Definition 4 *A deterministic finite-state automaton is a 5-tuple denoted by $A = (Q, \Sigma, \delta, q_0, Q_m)$ where:*

$Q = \{q_1, \dots, q_n\}$ is a finite set of states;

$\Sigma = \{\sigma_1, \dots, \sigma_m\}$ is the alphabet or symbol set;

$\delta : \Sigma \times Q \rightarrow Q$ is the transition function, where

$$\begin{aligned} \delta(\varepsilon, q) &= q & e \\ \delta(\sigma, q) &= q', \text{ para } q, q' \in Q \wedge \sigma \in \Sigma; \end{aligned} \quad (8)$$

$q_0 \in Q$ is the initial state;

$Q_m \subseteq Q$ is the set of marked states.

Example 1 For the automaton shown in Fig. 1 the following definitions apply:

$$\Sigma = \{\alpha, \beta, \gamma\};$$

$$Q = \{0, 1, 2\};$$

$$\delta(\alpha, 0) = 1, \delta(\alpha, 1) = 2, \delta(\alpha, 2) = 2, \delta(\beta, 0) = 2, \delta(\beta, 1) = 0, \delta(\beta, 2) = 1,$$

$$\delta(\gamma, 1) = 1, \delta(\gamma, 2) = 0;$$

$$q_0 = 0 \text{ and } Q_m = \{1, 2\}.$$

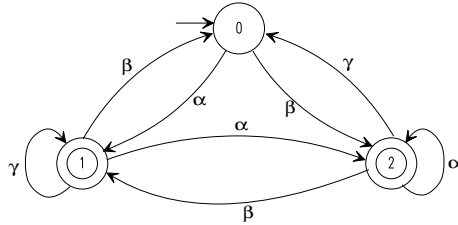


Fig. 1 Simple deterministic finite-state automaton.

By extending the transition function, we can describe how the automaton process strings.

Definition 5 For an automaton $A = (Q, \Sigma, \delta, q_0, Q_m)$ the extended transition function δ^* is defined by $\delta^* : \Sigma^* \rightarrow Q$, such that:

$$\begin{aligned} \delta^*(\varepsilon, q) &= q \\ \delta^*(s\sigma, q) &= \delta(\sigma, \delta^*(s, q)), q \in Q, s \in \Sigma^* \end{aligned} \tag{9}$$

It is a common practice to use δ instead of δ^* , since that

$$\delta^*(\sigma, q) = \delta(\sigma, \delta^*(s, q)) = \delta(\sigma, q)$$

for the case where $s = \varepsilon$.

The languages associated to a given automaton are defined as follows:

Definition 6 Given the automaton $A = (Q, \Sigma, \delta, q_0, Q_m)$, its language $L(A)$ is given by

$$L(A) = \{s | s \in \Sigma^* \wedge \delta(s, q_0) \in Q_m\}. \tag{10}$$

Definition 7 Given the automaton $A = (Q, \Sigma, \delta, q_0, Q_m)$, its marked language $L_m(A)$ is given by

$$L_m(A) = \{s | s \in \Sigma^* \wedge \delta(s, q_0) \in Q_m\}. \tag{11}$$

2.2 Diod Algebra

Definition 8 A dioid is a set D endowed with two operations denoted by \oplus (addition) and \otimes (multiplication), that satisfies the following axioms:

Axiom 1: Commutativity of \oplus : $a \oplus b = b \oplus a$

Axiom 2: Associativity of \oplus : $(a \oplus b) \oplus c = a \oplus (b \oplus c)$

Axiom 3: Associativity of \otimes : $(a \otimes b) \otimes c = a \otimes (b \otimes c)$

Axiom 4: *Distributivity of \otimes over \oplus :*

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$$

$$c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$$

Axiom 5: *Null element in \oplus : $a \oplus \epsilon = a$, $\forall a \in D$ and any $\epsilon \in D$*

Axiom 6: *Absorbing by null element in \otimes : $a \otimes \epsilon = \epsilon$*

Axiom 7: *Identity element in \otimes : $a \otimes e = a$*

Axiom 8: *Idempotency in \oplus : $a \oplus a = a$.*

A dioid is said commutative if \otimes is commutative. In dioid algebra the inverse of \oplus is not allowed, however, the inverse of \otimes can be defined.

2.3 Dioids and Formal Languages

The specification of D and its related operators \oplus and \otimes define the framework of the dioid algebra for a given application. To apply dioid algebra to deal with formal languages, we start defining that $D = \mathcal{P}(\Sigma^*)$, where $\mathcal{P}(\Sigma^*)$ is the set of all languages that can be defined with symbols of the alphabet Σ . Thus, an element of $\mathcal{P}(\Sigma^*)$ is a language, and the operations \oplus and \otimes can be defined as follows:

Definition 9 *Given a dioid (D, \oplus, \otimes) with $D = \mathcal{P}(\Sigma^*)$, the operators \oplus and \otimes , are defined as the operators of union and concatenation of languages, respectively. Given $L_1 \in D$ and $L_2 \in D$ then*

$$\begin{aligned} L_1 \otimes L_2 &= \{s_1 s_2 \mid s_1 \in L_1 \wedge s_2 \in L_2\} \\ L_1 \oplus L_2 &= \{s \mid s \in L_1 \vee s \in L_2\}. \end{aligned} \tag{12}$$

All the axioms of the Definition 8 are also valid when $D = \mathcal{P}(\Sigma^*)$. The null element ‘ ϵ ’ denotes the empty language \emptyset while the identity element ‘ e ’ denotes the language $\Sigma^0 = \{\varepsilon\}$.

As a consequence of the Definition 9 the concatenation of a given language L with ‘ e ’, is $L \otimes e = e \otimes L = L$, and the concatenation of a language L with ϵ is $L \otimes \epsilon = \epsilon \otimes L = \epsilon$. Similar results are defined in the context of formal languages[1,2,7].

The \oplus operator also allows the use of regular expressions. This is done by simply replacing the $+$ by \oplus to indicate the choice between two or more paths.

Example 2 *The language $L_1 = \alpha\beta^*$ added with the language $L_2 = \alpha$ gives*

$$L_3 = L_1 \oplus L_2 = \alpha\beta^* \oplus \alpha = \alpha\beta^* + \alpha = \alpha(\beta^* + e)$$

representing the union of L_1 and L_2 .

3 Incidence Matrix

For every automaton A we can associate an incidence matrix defined as follows:

Definition 10 *For an automaton $A = (\Sigma, Q, \delta, q_0, Q_m)$ with $|Q| = N$, its incidence matrix \mathbf{A} is defined as*

$$\mathbf{A} = [a_{ij}], a_{ij} = \begin{cases} \sigma & \text{if } \exists \sigma \text{ from state } i \text{ to state } j; \\ \epsilon & \text{elsewhere,} \end{cases} \quad (13)$$

where $\sigma \in \Sigma^*$, $\sigma = \sigma^1 + \sigma^2 + \dots + \sigma^n$ is the regular expression that labels the arc connecting the node i to node j of the automaton A . The initial state is defined by the row vector $\theta_{1 \times N}(\mathbf{A})$, $\theta(\mathbf{A}) = [e \ \epsilon \ \dots \ \epsilon]$, that is, the first element is 'e' and the other are ϵ . The marked states are defined by the column vector $\phi_{N \times 1}(\mathbf{A})$,

$$\phi(\mathbf{A}) = \begin{cases} e & \text{if the row } i \text{ is marked;} \\ \epsilon & \text{elsewhere.} \end{cases} \quad (14)$$

Thus, the row i represents the actual state of the automaton and the column j , the next state (row), if $a_{i,j} \neq \epsilon$. If there is more than one event defining the change from state i to j then a_{ij} will be a regular expression. The vector θ only presents the first element different of ϵ , defining that $i = 1$ is always the initial state and, the vector ϕ presented to right of \mathbf{A} , indicates that a row is marked if $\phi_i(\mathbf{A}) = e$ [$\phi_i(\mathbf{A})$ is the element of the i -th row of the vector $\phi(\mathbf{A})$].

Remark: Since only the first element of $\theta(\mathbf{A})$ is different from ϵ this implies that the first row of \mathbf{A} always represent the initial state. The final states are represented graphically by adjoining the elements of the vector $\phi(\mathbf{A})$ as an extra column placed to the right side of \mathbf{A} .

Example 3 For the automaton shown in Fig. 2 its incidence matrix \mathbf{A} is given by

$$\mathbf{A} = \begin{bmatrix} \epsilon & \alpha + \lambda + \beta & \beta \\ \eta & \epsilon & \epsilon \\ \epsilon & \beta & \alpha + \lambda \end{bmatrix} \begin{matrix} \epsilon \\ \epsilon \\ e \end{matrix}$$

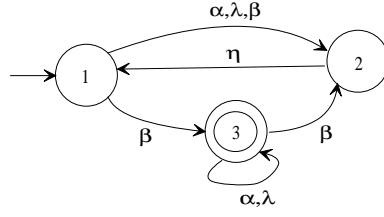


Fig. 2 Non deterministic finite-state automaton.

with row 3 representing a marked state. The elements of the vector $\phi(\mathbf{A})$ are shown in the right side of \mathbf{A} ; the ‘e’ indicates a marked state (row).

3.0.1 Automata Languages The incidence matrix \mathbf{A} of the automaton A can be used to define the languages $L(\mathbf{A}) = L(A)$ and $L_m(\mathbf{A}) = L_m(A)$. To determine these languages it is required to introduce the following definition:

Definition 11 The matrix $\mathbf{A}^n = \mathbf{A} \otimes \mathbf{A} \otimes \dots \otimes \mathbf{A}$ (n times) is defined as the path matrix. Each element a_{ij}^n , represents one or more paths of length n from state i to state j . The initial state vector and final state vector of \mathbf{A}^n are the same of \mathbf{A} .

If there is not a path between state i and state j then $a_{ij}^n = \epsilon$.

Example 4 For the automaton A shown in Fig. 3 the incidence matrix is given

$$\mathbf{A} = \begin{bmatrix} \epsilon & \alpha & \epsilon \\ \beta & \epsilon & \mu \\ \epsilon & \beta & \alpha \end{bmatrix} \begin{matrix} \epsilon \\ \epsilon \\ e \end{matrix}$$

The path matrix \mathbf{A}^2 for this automaton is

$$\mathbf{A}^2 = \mathbf{A} \otimes \mathbf{A} = \begin{bmatrix} \alpha\beta & \epsilon & \alpha\mu \\ \epsilon & \mu\beta + \beta\alpha & \mu\alpha \\ \beta\beta & \alpha\beta & \alpha\alpha + \beta\mu \end{bmatrix} \begin{matrix} \epsilon \\ \epsilon \\ e \end{matrix}$$

and the path matrix \mathbf{A}^3 is

$$\mathbf{A}^3 = \begin{bmatrix} \epsilon & \alpha\beta\alpha + \alpha\mu\beta & \alpha\mu\alpha \\ \mu\beta\beta + \beta\alpha\beta & \alpha\mu\beta & \mu\alpha\alpha \\ \alpha\beta\beta & \beta\beta\alpha + \alpha\alpha\beta + \beta\mu\beta & \alpha\beta\mu + \alpha\alpha\alpha + \beta\mu\alpha \end{bmatrix} \begin{matrix} \epsilon \\ \epsilon \\ e \end{matrix}.$$

In these matrices, each element (i, j) represents a string of length 2 and 3, respectively. These sequences are defined by the extended transition function of the automaton A , that represent the changes from state i to state j . In \mathbf{A}^2 , we see that from initial state there exists the string $\alpha\beta$. The strings $a_{13}^2 = \alpha\mu$ and $a_{13}^3 = \alpha\mu\alpha$ (row 1, column 3) are recognized by this automaton because these strings determine the changing from the initial state to a marked state 3 ($\phi_3(\mathbf{A}) = e$).

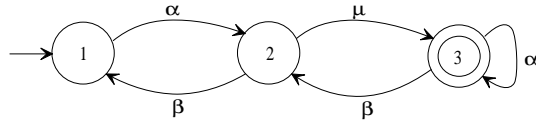


Fig. 3 Simple deterministic finite-state automaton used to illustrate the construction of the path matrix.

Based on the definition provided for \mathbf{A}^n we can formalize $L(\mathbf{A})$ and $L_m(\mathbf{A})$.

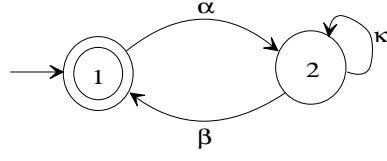


Fig. 4 Simple example of deterministic finite-state automaton.

Definition 12 Given an incidence matrix \mathbf{A} , its language is defined by

$$L(\mathbf{A}) = \bigoplus_i (\theta(\mathbf{A}) \otimes \mathbf{A}^i) = \bigoplus_i \bigoplus_{j=1}^n (\theta_1(\mathbf{A}) \otimes a_{1j}^i), \quad (15)$$

where a_{1j}^i is the element of row 1, column j of the path matrix \mathbf{A}^i .

Its worth noting that $L(\mathbf{A}) = \overline{L(\mathbf{A})}$.

Definition 13 Given an incidence matrix \mathbf{A} , its marked language is defined by

$$\begin{aligned} L_m(\mathbf{A}) &= \bigoplus_i (\theta(\mathbf{A}) \otimes \mathbf{A}^i \otimes \phi(\mathbf{A})) \\ &= \bigoplus_i \bigoplus_{j=1}^n (\theta_1(\mathbf{A}) \otimes a_{1j}^i \otimes \phi_j(\mathbf{A})), \end{aligned} \quad (16)$$

where a_{1j}^i is the element of the row 1, marked column j of the path matrix \mathbf{A}^i .

This definition is similar to Definition 12. However, in this case, we only consider the strings that start at the initial state and end in a marked column.

Example 5 Consider the automaton shown in Fig. 4 for which

$$\mathbf{A} = \begin{bmatrix} \epsilon & \alpha \\ \beta & \kappa \end{bmatrix} \begin{matrix} e \\ \epsilon \end{matrix}.$$

To determine its language, we compute

$$\mathbf{A}^2 = \begin{bmatrix} \alpha\beta & \alpha\kappa \\ \kappa\beta & \beta\alpha + \kappa\kappa \end{bmatrix} \begin{matrix} e \\ \epsilon \end{matrix},$$

$$\mathbf{A}^3 = \begin{bmatrix} \alpha\kappa\beta & \alpha\beta\alpha + \alpha\kappa\kappa \\ \beta\alpha\beta + \kappa\kappa\beta & \kappa\beta\alpha + \beta\alpha\kappa + \kappa\kappa\kappa \end{bmatrix} \begin{matrix} e \\ \epsilon \end{matrix}, \dots$$

whose elements of the first row are strings of length 2, 3, ..., of the language of \mathbf{A} . It can be seen that

$$L(\mathbf{A}) = \overline{L(\mathbf{A})} = \{\varepsilon, \alpha, \alpha\beta, \alpha\kappa, \alpha\kappa\beta, \alpha\beta\alpha, \alpha\kappa\kappa, \dots\}.$$

Observe that the empty string $\varepsilon \in L(\mathbf{A})$, and that the strings of length i , are found in row 1 of \mathbf{A}^i . The marked language is $L_m(\mathbf{A}) = \{\varepsilon, \alpha\beta, \alpha\kappa\beta, \dots\}$ and is determined by the strings of the row 1, column 1 of \mathbf{A}^i , since $\phi_1(\mathbf{A}) = e$.

4 Transition Function Formulation

Employing the Definition 12 we can determine the set of all strings of length $|s| = i$ by

$$L(\mathbf{A})_i = \theta(\mathbf{A}) \otimes \mathbf{A}^i = \bigoplus_{j=1}^n (\theta_1(\mathbf{A}) \otimes a_{1j}^i) \quad (17)$$

and the set of all strings of length $|s| = i$ that makes the state to change from row 1 to row k by

$$L(\mathbf{A})_i^k = \theta(\mathbf{A}) \otimes \mathbf{A}^i \otimes \pi = \theta_1(\mathbf{A}) \otimes a_{1k}^i. \quad (18)$$

Through these equations it is possible to derive an expression for the transition function of any automaton (without self-loops). As a first step for this derivation we need to state the following definition:

Definition 14 *Given $\Sigma = \{\sigma_1, \dots, \sigma_n\}$ and consider that the symbol σ_i determines the transition from a state q to a state q' . Now lets define σ_i^{-1} as the inverse symbol that determines from which state the state q' has been reached.*

With this definition, we understand the following: if in an automaton exists σ departing from state q to state q' , then σ^{-1} can be viewed like an arc that departs from state q' to state q . Thus, this definition can be extended to deal with strings in the following way:

Definition 15 *Given $\Sigma = \{\sigma_1, \dots, \sigma_n\}$ and Σ^* representing the set of all possible strings s formed by the symbols of Σ . Consider a string s_i that determines the transition from the state q to a state q' . Now lets define s_i^{-1} as the inverse string that determines from which state the state q' has been reached.*

Remark Similarly to σ^{-1} , if in an automaton exists the string s that changes the state of the automaton from state q to state q' , following the path $qq_1q_2\dots q_nq'$, then s^{-1} can be viewed like an string that departs from state q' to state q , following the inverse path $q'q_nq_{n-1}\dots q_1q$.

By using these two definitions one can derive a representation of the transition function of an automaton (without self-loops).

With equation (18) it is possible to determine the set of all strings that starting from the initial state have length $|s| = i$ and lead the automaton to state k . Working with that equation one can find all the strings of

length $|s| = i$ that lead the automaton from state k to state h . This can be expressed by

$$L^h(\mathbf{A})_i^k = \xi \otimes \mathbf{A}^i \otimes \pi = \xi_h \otimes a_{1k}^i, \quad (19)$$

where ξ is a row vector $1 \times n$, where only its h -th element is equal to e while all the others elements are equal to ϵ .

Given equation (19) the strings can be determined if: i) the incidence matrix \mathbf{A} of the automaton is known and ii) the initial and final states where the automaton initiates and ends its processing, respectively, are known.

Lets consider, initially, that $L^h(\mathbf{A})_i^k$ contains only one string s .

$$s = \xi \otimes \mathbf{A}^i \otimes \pi. \quad (20)$$

Working with equation (20), we find that

$$s = \xi \otimes \mathbf{A}^i \otimes \pi, \quad (21)$$

$$\xi^T \otimes s = \mathbf{A}^i \otimes \pi, \quad (22)$$

$$(\mathbf{A}^i)^{-1} \otimes \xi^T \otimes s = \pi, \quad (23)$$

$$\pi = (\mathbf{A}^i)^{-1} \otimes \xi^T \otimes s, \quad (24)$$

where $(\mathbf{A}^i)^{-1}$ is the inverse path matrix (all its elements are inverse strings), such that

$$(\mathbf{A}^i)^{-1} \otimes \mathbf{A}^i = \mathbf{I}.$$

The column vector π has only one element different from ϵ ; $\pi_k = e$ is this element and the subscript k represents the state reached when the string s is processed.

Now, let us consider the row vector

$$\tau | \tau_j = j, j = 1, 2, \dots, n.$$

Vector τ can be used to determine what is the reached state k as given by

$$k = \pi \otimes \tau = \left[(\mathbf{A}^i)^{-1} \otimes \xi^T \otimes s \right] \otimes \tau. \quad (25)$$

Equation (25) is the indeed the transition function for an automaton that does have self-loops in its structure; by using this expression, given the string s one can compute the reached state from any state. The example presented in the following will illustrate the use of this equation.

Example 6 Consider the following incidence matrix

$$\mathbf{A} = \begin{bmatrix} \epsilon & \beta \\ \kappa & \epsilon \end{bmatrix}$$

that represents the automaton of the Fig. 5. Given $s = \kappa$ and the state

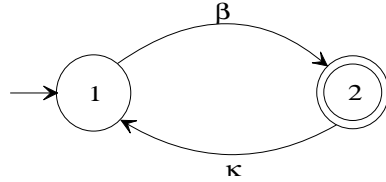


Fig. 5 Simple determinist finite-state automaton.

$q = 2$, the reached state is

$$k = \left[\begin{bmatrix} \epsilon & \kappa^{-1} \\ \beta^{-1} & \epsilon \end{bmatrix} \otimes \begin{bmatrix} \epsilon \\ e \end{bmatrix} \otimes \kappa \right] \otimes \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$k = \left[\begin{bmatrix} \kappa^{-1} \\ \epsilon \end{bmatrix} \otimes \kappa \right] \otimes \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$k = \begin{bmatrix} e \\ \epsilon \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \end{bmatrix} = 1.$$

Now consider that $s = \beta\kappa\beta\kappa\beta$ and the state $q = 1$. The reached state is

$$k = \left[\begin{bmatrix} \epsilon & (\kappa\beta\kappa\beta\kappa)^{-1} \\ (\beta\kappa\beta\kappa\beta)^{-1} & \epsilon \end{bmatrix} \otimes \begin{bmatrix} e \\ \epsilon \end{bmatrix} \otimes \beta\kappa\beta\kappa\beta \right] \otimes \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$k = \left[\begin{bmatrix} (\beta\kappa\beta\kappa\beta)^{-1} \\ \epsilon \end{bmatrix} \otimes \beta\kappa\beta\kappa\beta \right] \otimes \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$k = \begin{bmatrix} e \\ \epsilon \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \end{bmatrix} = 1.$$

Whenever a string s is not feasible to occur from state q , the utilization of the equation (25) does not yield valid result. To solve this problem, we must extend the operator \otimes in the following manner:

Definition 16 For $D = \mathcal{P}(\Sigma^*)$ and given two languages L and L' where L' is an inverse language, the operator \otimes is defined as

$$L \otimes L' = \begin{cases} ss' & \text{if } s \in L \wedge s' \in L^{-1} \wedge s' = s^{-1} \\ \epsilon & \text{elsewhere.} \end{cases} \quad (26)$$

According to this definition, when the string s is not feasible to occur in state q the result of the operation is ϵ . Then, equation (25) remains valid but now yields the correct result.

Example 7 *Considering again Example 6 with $s = \beta\kappa\beta\kappa\beta$ and $q = 2$. In this case*

$$\begin{aligned}
 k &= \left[\begin{bmatrix} \epsilon & (\kappa\beta\kappa\beta\kappa)^{-1} \\ (\beta\kappa\beta\kappa\beta)^{-1} & \epsilon \end{bmatrix} \otimes \begin{bmatrix} \epsilon \\ e \end{bmatrix} \otimes \beta\kappa\beta\kappa\beta \right] \otimes \begin{bmatrix} 1 & 2 \end{bmatrix} \\
 k &= \left[\begin{bmatrix} (\kappa\beta\kappa\beta\kappa)^{-1} \\ \epsilon \end{bmatrix} \otimes \beta\kappa\beta\kappa\beta \right] \otimes \begin{bmatrix} 1 & 2 \end{bmatrix} \\
 k &= \begin{bmatrix} \epsilon \\ \epsilon \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \end{bmatrix} = \epsilon,
 \end{aligned}$$

showing that the automaton does not accept the string $s = \beta\kappa\beta\kappa\beta$ from state $q = 2$.

5 Conclusions

This paper presents a generic formulation to equation of the transition function of the finite automata utilizing the incidence matrices and dioid algebra applied to languages. For this formalization we define the inversion of symbols and strings, which are allowed in formalizing of the dioid algebra, as good an extension in the definition of the operator \otimes applied to languages. The described formalism allows find the reached state q' in an automaton A from a given state q and a string s . The equation of the transition function is utilized to automata that do not present self-loops in their structure. The computing of the inverse incidence matrix is done by using the same algebraic procedures to computing any inverse matrix. With the examples shown, we see that the formalized equation solve the procedure of reached

state computing in an automaton without the recurrence to the table usually defined to the automata transition function.

References

1. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, USA, 1979)
2. J.E. Hopcroft and J.D. Ullman, *Formal Languages and Their Relation to Automata* (Addison-Wesley Publishing Company, 1969)
3. S. Gaubert, *Théorie des Systèmes Linéaires dans les Dioïdes* (Thèse de Doctorat, École Nationale Supérieure des Mines de Paris, 1992)
4. J.A. Beachy, *Abstract Algebra II* (Waveland Press, Inc., 1996)
5. J. Berstel and C. Reutenauer, *Rational Series and their Languages* (Springer, 1988)
6. I. Klimann, *Langages, Séries et Contrôle de Trajectoires* (Thèse de Doctorat, l'Université Denis Diderot - Paris 7, 1999)
7. W.M. Wonham, *SED Notes* (Course notes, 1999)