INTRODUCING REAL TIME IN THE ALGEBRAIC THEORY OF FINITE AUTOMATA

Pierre Moller

September 1986
WP-86-49

INTRODUCING REAL TIME IN THE ALGEBRAIC THEORY OF FINITE AUTOMATA

Pierre Moller

September 1986
WP-86-49

FOREWORD

This paper shows how Real-Time can be introduced into the algebraic description of finite automata, to provide a tool for modelling discrete-event-systems.

Alexander B. Kurzhanski,
Chairman,
System and Decision Sciences Program

Introducing Real Time into the Algebraic Theory of Finite Automata

Pierre MOLLER
International Institute for Applied Systems Analysis.
September 1986.

Algebraic automata theory and its corollary, regular language theory, are efficient tools used in computer science for modeling logical circuits, designing compilers, evaluating complexity of algorythms and other problems.

Recently, W.M. Wonham and P.J. Ramadge (see ref. [7] to [11]) applied this algebraic framework to "discrete-event systems". This is a new class of systems, which appears in various domains, ranging from flexible-manufacturing plants to communication-protocols controllers.

The major drawback of the classical automata theory is that it does not take into account the "real time". There is only an implicit notion of "logic time", due to the precedence of events.

After a brief introduction to the algebraic theory of finite automata, we shall show how real-time can be introduced in these models.

1 Introduction to finite state automata.

A good intoduction to this theory can be found in reference [0], chapters 1 to 3 .

A finite state automaton is a mathematical model for a system, which can be in any one of a finite number of states, and which moves from one state to another according to discrete inputs, taken from a finite set of inputs, producing discrete outputs, from a finite set of possible outputs.

The state of the automaton summarizes all the past inputs and their influence on the future outputs.

This definition can be formalized in the following way:

Definition 1.1 :

A finite automaton is a 5-tuple (Q , Σ , δ , q , F) where:
- Q is a finite set of states,
- Σ is a finite input alphabet,
- q is the initial state, an element of Q
- F is the subset of Q of final states,
- δ is the transition function, mapping $Q \times \Sigma$ to Q.

The last item indicates that for any state p and any input a, $\delta(q,a)$ is defined and is a state. Thus, a can be viewed as a control.

We can generalise this definition to automata accepting strings of inputs, in the following way:

We define Σ^* as the set of all finite strings constructed with the alphabet Σ, plus the empty string which we shall denote $\varnothing$; Σ^* is usually called the free monoïd generated by Σ, because the concatenation of strings provides Σ^* with a structure of monoïd.

Concatenation of strings w and x will be denoted wx.

Since Σ^* contains Σ, we can extend the function δ to Σ^*. This is done recursively: the state associated to a string is the state obtained by aplying successively all letters of the string as inputs, read from left to right.

Definition 1.2:

The extended transition function $\delta\hat{}$ of the finite automaton (Q , Σ , δ , q , F) is the function mapping Q X Σ^* on Q defined by:

$\delta\hat{}(p, \varepsilon) = p$ for every state p and the empty string ε,

$\delta\hat{}(p, wa) = \delta(\delta\hat{}(p,w),a)$

for every state p, string w, and letter a.

From now on, we shall use only $\delta\hat{}$ and denote it δ, for sake of simplicity.

A finite automaton can be represented by its transition diagram. This is a directed graph, which has one node associated with each state, and whose arcs are labeled by the input alphabet: for every state p and every input a, there is one arc labelled by a, leaving the node p.

As an example, let us consider a finite automaton with 3 states:

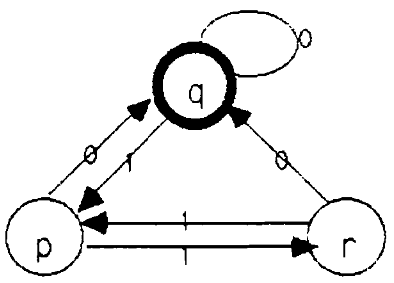
Let Q = {q,p,r} q be the initial state,

and the input alphabet be $\Sigma = \{0,1\}$.

The transition function is given by the table:

		state		
		q	p	r
input	0	q	q	q
	1	p	r	p

The transition diagram of this automaton is:



In this automaton, the input string 11101 will take the automaton from the initial state q to the state p.

Definition 1.3 :

A string is said to be <u>accepted</u> by a finite automaton, if it moves the automaton from the initial state to a <u>final</u> state.

Suppose that on the previous example, r is the only final state, then the string 11101 is not accepted by this automaton, but the string 011 is.

It is easy to see that a string is accepted by this automaton if and only if:

There are 2n ($n \geq 1$) occurences of the input 1 and no 0 in the string,
Or:
There are 2n ($n \geq 1$) occurences of the input 1, following the last 0 in the string .

Definition 1.4 :

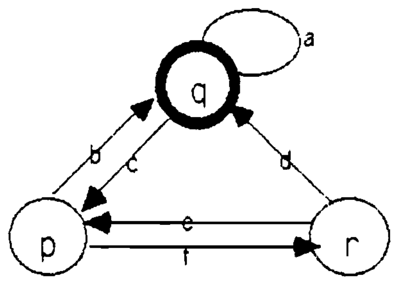
Let Σ^* be the free monoïd generated by an alphabet Σ.
A subset L of Σ^* (a language) is said to be <u>regular</u>, if there exists a finite automaton, with inputs in Σ, which accepts <u>all the strings in L</u> and <u>only the strings in L</u>.

Until now, we have associated no output to a finite automaton. To introduce outputs, we can use two type of models: <u>Mealy machines and Moore machines</u>.

2 Mealy Machines:

In the sequel, we shall suppose, for sake of simplicity, that all states are final.

Mealy machines are obtained by associating one letter, chosen in an output alphabet, to every arc in the transition diagram. In the previous example, we can re-label the arcs in the following way:



Then the input string 11101 will give the output cfebc.

This definition can be formalized in the following way:

Definition 2.1 :

A Mealy machine is a 6-tuple (Q , Σ , Γ, q , δ , η) where:

Q is a finite set of states,
Σ is a finite input alphabet,
q is the initial state, an element of Q
δ is the transition function, mapping Q X Σ into Q.
Γ is a finite output alphabet
η is a function mapping Q X Σ into Γ.

η is the output function, which associates an output in Γ, to every transition in Q X Σ. If η is not one-to-one, some moves of the automaton cannot be distinguished by an observer of the outputs.

This definition can be obviously extended to input strings, by using recursion. Thus we can consider η as mapping Q X Σ^* into Γ^*.

Restrictions:

In this first study, we shall concentrate onn tracking the transitions of the states. Thus we shall assume that for all Mealy machines, That <u>η is one-to-ne</u> . This means that two different arcs in the transition diagram always have different labels. In this case, outputs are equivalent to state-transitions. This was not the case on the example introduced previously. In the end of this paper, we shall indicate which problem arises if η is not one-to-one.

In the usual theory of finite automata, the behaviour of the automata is described by the language of all possible strings of outputs. To use numerical coefficients, we need to introduce the notion of formal and rational series.

<u>3 Formal and rational series.</u>

Definition 3.1 :

A <u>dioïd</u> is a 3-uple (D, $\oplus$, x) where
D is a set of "scalars", with two distinguished elements such that:
$\oplus$ is an associative and commutative internal operation,
x is an associative internal operation,
x is distributive over $\oplus$.
$\varepsilon \oplus a = a$ for every a in D
$\varepsilon \times a = \varepsilon$ for every a in D.
$e \times a = a$ for every a in D.

This means that (D, $\oplus$) is a commutative monoïd, (D,x) is a monoïd and, D contains two distinguished elements ε and e :

ε is the neutral element for the "addition".
e is the neutral element for "multiplication".

This structure is often called semi-ring; this word may induce some confusion, because many authors use the word "semi-ring" only if the operation $\oplus$ is regular: this is the case in the realisation of markov-chains, where the dioïd is the positive real line with usual operations. Therefore, we, shall use the word dioïd, which obviously stands for the extension of a monoïd.
Note that all rings and fields are dioïds, and that:

In any dioïd, matrix calculus can be defined by the usual formulas.

In the sequel, for simplicity of notations, the "product" $a X b$ will be denoted $a.b$. And the sum in the sense of the dioïd will be denoted $\oplus$.

Definition 3.2 :

Let Γ be a finite alphabet.
A formal series, with coefficients in a dioïd D and variables in Γ is a function S mapping Γ^*, the free monoïd generated by Γ, into D.
S can be represented formally as:

$$S = \Sigma_{\omega \in \Gamma^*} S(\omega).\omega$$

Definition 3.3 :

Let Γ be a finite alphabet.
A formal polynomial, with coefficients in a dioïd D and variables in Γ is a formal series P, such as
$P(\omega) \neq \varepsilon$ only for a finite set of strings ω in Γ^*.

Usually, the notation $D\langle\langle \Gamma \rangle\rangle$ is used for the set of formal series with coefficients in D and variables in Γ, and $D\langle \Gamma \rangle$ stands for the set of formal polynomials with coefficients in D and variables in Γ.

If $\Gamma = \{z_1, z_2, \ldots z_n\}$, one can also write $D\langle\langle z_1, z_2 \ldots z_n \rangle\rangle$ and $D\langle z_1, z_2 \ldots z_n \rangle$.

We can extend the operations defined on the dioïd to the formal series in the following way:

Definition 3.4 :

The formal sum of two series S and T is the formal series $(S \oplus T)$ defined by:
$(S \oplus T)(\omega) = S(\omega) \oplus T(\omega)$ for every string ω in Γ^*.

Definition 3.5 :

The formal product of two series s and T is the series (S.T) defined by:
$(S.T)(\omega) = \Sigma_{\alpha\beta=\omega \ \ \alpha\in\Gamma^* \ \ \beta\in\Gamma^*} S(\alpha) \,.\, T(\beta)$
Where $\alpha\beta$ is the concatenation of strings α and β.

According to these formulas, the neutral element for the sum of formal series is the function (or series) which associates ε to every coefficient; therefore it is convenient to denote this element ε.

The neutral element for the product is the same, except that the coefficient of the empty string $\emptyset$ is e; for convenience we shall denote this series e.

The dioïd D can be obviously inbeded in $D\langle\langle\Gamma\rangle\rangle$ by identifying every scalar λ to the series denoted λ ,which has all its coefficients equal to ε, except the coefficient of the empty string $\emptyset$, which is equal to λ; we just write $\lambda.\emptyset = \lambda$.

Usually, the strings with coefficients equal to ε are omitted; and coefficients equal to e are omitted too: For instance $z \oplus 3z^2$ stands for:

$$\varepsilon\emptyset \oplus ez \oplus 3z^2 \oplus \varepsilon 3^2 \oplus \varepsilon z^4 \oplus \varepsilon z^5 \oplus \varepsilon z^6 \oplus \ldots\ldots$$

Thus, the sum of series appears as an extension of the sum in the dioïd, and the product of series appears as an extension both of the product in the dioïd and of concatenation of strings.

Let us look for a series S which is solution of:

$$S = S.A \oplus B \tag{3.1}$$

Where A and B are given formal series. It is easy to compute that:

$$S = S.A^n \oplus B.(e \oplus A \oplus A^2 \oplus .... A^{n-1}) \tag{3.2}$$

Thus, if A^n "vanishes" when n tends to $+ \infty$, we get one solution of equation (3.1), which may or may not be the unique solution, depending on the dioïd:

$$S = B.A^* \tag{3.3}$$

Where A^* is defined at least for all formal series such as the coefficient of the empty string is ε.

Definition 3.6 :

If A is a formal series such as the coefficient of the empty string is ε,

$$A^* = e \oplus A \oplus A^2 \oplus .... A^n \oplus ..... \tag{3.4}$$

We can now justify the notation Γ^* for the set of all strings generated by an alphabet Γ. A language (a set of finite strings) can be viewed as a formal series with coefficients in the Boolean algebra {0,1}, and variables in Γ:

The coefficient of a string is 1 if it belongs to the language,
The coefficient of a string is 0 if it does not belong to the language.

This is true for Γ itself, and it can be easily checked that the series Γ^* is actually obtained by applying the * operation to Γ, considered as the formal sum of all its letters.

This S^* operation can be considered as the formal expansion of the quotient 1/1-S . Control theorists will recognise the importance of this operation, because it is associated to feedbacks in systems.

Definition 3.7 :

The <u>rational operations</u> on formal series are:
The formal sum, the formal product, the * operation.

Definition 3.8 :

The rational closure of a subset L of $D\langle\langle \Gamma \rangle\rangle$ is the smallest subset of $D\langle\langle \Gamma \rangle\rangle$ containing L which is stable under the rational operations.

Definition 3.9 :

The set of <u>rational series</u> is the rational closure of the set of polyomials.

This implies that every rational series can be written as a finite expression, using polynomials and rational operations.

Theorem 3.10 :

Every regular language is a rational series with coefficients in the Boolean algebra.

Theorem 3.11 :

The support of every rational series (the set of words which have non-ε coefficients) in $N\langle\langle \Gamma \rangle\rangle$,is a regular language.

This theorem holds in other dioids than N, the set of positive integers, with the usual operations, but it does not hold for all dioids.

Theorem 3.12 : (Realisation Theorem, Kleene-Schützenberger)

If S is a rational series with coefficients in D and variables in Γ, there exists:

an integer n

a (1,n) matrix G with coefficients in Γ,

a (n,1) matrix N with coefficients in Γ,

an application φ mapping Γ into the set of (n,n) matrixes with coefficients in D,

such that:

if we extend φ recursively to Γ^* by defining:

$\varphi\hat{}\ (\varepsilon) = e$

$\varphi\hat{}\ (\omega\alpha) = \varphi\hat{}\ (\omega).\ \varphi\ (\alpha)$ for every string ω and every letter α, where $\varphi\hat{}\ (\omega).\ \varphi\ (\alpha)$ is the matrix product of $\varphi\ (\alpha)$ and $\varphi\ (\alpha)$.

Then:

The coefficient of any word ω in S is

$S(\omega) = H.\ \varphi\hat{}\ (\omega).G$.

This theorem is obviously the extenion of the realisation theorem for finite dimensional linear systems. It is due to Kleene for regular languages, and Schützenberger for formal rational series.

<u>4 Introducing real time in automata theory.</u>

Let us consider a Mealy machine, with output alphabet Γ. We still suppose that the function η mapping f the arcs of the transition diagram on the output alphabet Γ, is one-to-one, and that all states are final.

To increase the power of this model, we are going to take into account the <u>time needed to perform a transition</u> from one state to another.

Thus, we are given a function τ mapping Γ on R+, such as:

for every output letter a, $\tau(a)$ is the time needed to produce a.

To this automata, we are going to associate a formal series with real (or infinite) coefficients such as:

> The coefficient of any string ω is the least time necessary to produce the output ω. To be precise, it will be the least time necessary until the end of theproduction of output ω.

If an output is impossible, its coefficient will be $+\infty$, which means that an infinite time is needed to perform this output.

We shall work with the following dioïd, often called max-algebra, or path-algebra, which has first been intensively studied by R. Cunningham-greene (see ref. [6]).

Definition 4.1 :

DRC is the dioïd defined by:
The set $IR \cup \{-\infty,+\infty\}$
The "addition" $a \oplus b = \max(a,b)$
The "product" $a.b = a+b$ (the product is the usual addition)
with the convention $-\infty.+\infty = -\infty = \varepsilon$.

Since we denote $-\infty$ by ε, in the calculations, we can write $+\infty = \infty$.

Formal series in one variable, with coefficients in the dioïd DRC, have been introduced and studied by Cohen and all. , in references [4] and [5].

Lemma 4.2:

In the dioîd DRC, the formal series $B.A^*$ is always defined and is the smallest solution to equation $S = S.A \oplus B$ in the sense of the partial order induced on the series by the order on the coefficients.

Demonstration of the lemma:

We first note that in this dioïd, since the "addition" is the maximum, if the formal series S, A, B satisfy

$$S = A \oplus B$$

Then $S \geq A$ and $S \geq B$ for the partial order .

Thus,

$$S = A^n.S \oplus B.(e \oplus A \oplus A^2 \oplus .... A^{n-1})$$

implies that for every string in Γ^*, its coefficient in S is greater or equal than its coefficient in $(e \oplus A \oplus A^2 \oplus .... A^{n-1})B$.

Furthermore, the coefficient of every string in this partial sum defines monotonous non-decreasing sequence when n tends to $+\infty$, and is convergent if we allow infinite coefficients.

So we can define the sum

$$(e \oplus A \oplus A^2 \oplus .... A^{n-1} \oplus ...)$$

without intoducing a topology, and

$$B.(e \oplus A \oplus A^2 \oplus .... A^{n-1})$$

is defined and minorates every solution of the equation $S = S.A \oplus B$ for the partial order.

B.A* is obviously is a solution, thus it is the smallest solution.

Now we can state the most important result.

Theorem 4.3 :

Let Γ be the output alphabet of a Mealy machine,
Let S be the formal series, with coefficients in DRC and variables in Γ defined in the following way:

If a string w is a possible output of the machine,
the coefficient ω is the least time necessary to produce ω .

if a string w cannot be produced bi the machine,
its coefficient is $+\infty$.

Then:

S is rational in the sense of the dioïd DRC and can be computed by the following formula:

$$S = (P \oplus Q) . (e \oplus T \oplus R)^*$$

Where P, Q and R are the following polynomials:

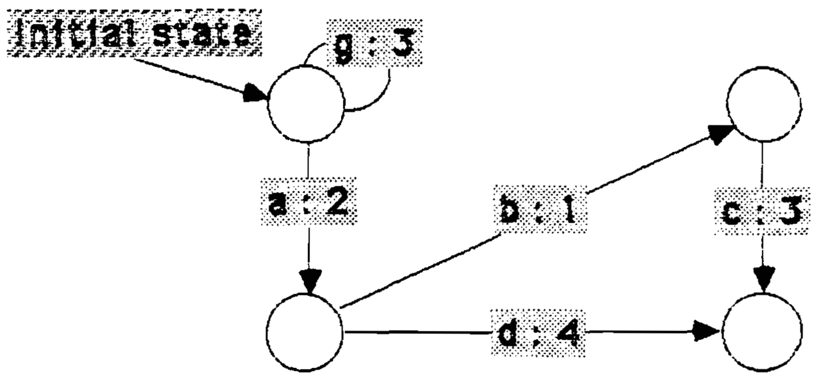
P is the sum of all <u>single outputs</u> (letters)which are possible from the <u>initial state</u>, multiplied by their duration.

Q is the sum of all <u>single outputs</u> which are impossible from the <u>initial state.</u> multiplied by $+\infty$.

T is the sum of all single outputs (all letters in Γ) multiplied by their duration.

R is the sum of all strings of 2 letters, which cannot be produced from any state, thus cannot appear in any output string, multiplied by $+\infty$.

Before giving the demonstration of the theorem, we shall study an example to clarify this statement. Consider the following Mealy machine. We have omitted the inputs to simplify the graph.



The polynomials are :

$$P = 3.g \oplus 2.a$$
$$Q = \infty.b \oplus \infty.c \oplus \infty.d$$
$$T = 3.g \oplus 2.a \oplus 1.b \oplus 3.c \oplus 4.d$$
$$R = \infty.(\ a(g \oplus a \oplus c) \oplus b(g \oplus a \oplus b \oplus d) \oplus (c \oplus d)(g \oplus a \oplus b \oplus d)\ )$$

If we compute the formal series associated to this automaton:

$$(P \oplus Q)(\ e \oplus T \oplus R\ )^*$$

we find that the only strings with finite coefficients are the following:

g^n with coefficient $3n$ (in the usual algebra, or 3^n in the dioïd),
$g^n a$ with coefficient $3n + 2$,
$g^n ab$ with coefficient $3n + 3$
$g^n abc$ with coefficient $3n + 6$
$g^n ad$ with coefficient $3n + 6$

Demonstration of the theorem 4.3:

We have defined S in the following way:

For any string ω, $S(\omega)$ is the least time needed to produce output w, if this output is possible, otherwise $S(\omega)$ is $+\infty$.

Note that by definition of S, all its coefficients are greater or equal to e=0 thus are never equal to ε.

We need to prove that S is equal to X° , which we define as the smallest solution, for the partial order, of equation:

$$X = (P \oplus Q) \oplus X.(e \oplus T \oplus R) \qquad (4\text{-}1)$$

Where P, Q, R, T are as defined in the theorem.

We know from the lemma 4.2 that:

$$X^\circ = (P \oplus Q).(e \oplus T \oplus R)^*$$

First part of the demonstration:

We first show that S is larger or equal, for the partial order, than X°, by showing that S is a solution of equation 4-1, that means:

$$S = (P \oplus Q) \oplus S.(e \oplus T \oplus R) = S \oplus P \oplus Q \oplus S.T \oplus S.R$$

Obviously $S \leq (P \oplus Q) \oplus S.(e \oplus T \oplus R)$ for the partial order,

Thus we need only to show that

$$S \geq P \oplus Q \oplus S.T \oplus S.R$$

This can be done by showing that any term appearing on the right hand side is dominated by a term on the left.

By definition of P, its terms are the single outputs which are possible from the initial state, multiplied by their duration. These terms appear also in S, by definition of S.

By definition of Q, its terms are the single outputs which are impossible from the initial state, multiplied by $+\infty$. These terms appear also in S,.

A term in the product S.T has the form $S(\omega).T(\kappa)\ \omega\kappa$, where $S(\omega).T(\kappa)$ is the dioïd notation for the usual sum of $S(\omega)$ and $T(\kappa)$.

> If the output ω is impossible, $S(\omega) = +\infty$ and the output $\omega\kappa$ is impossible too; thus we have $S(\omega).T(\kappa) \leq S(\omega\kappa) = +\infty$ as required.
>
> If the output ω is possible, but $\omega\kappa$ is impossible, we still have $S(\omega).T(\kappa) \leq S(\omega\kappa) = +\infty$ as required.
>
> If the output ω is possible, and $\omega\kappa$ is possible, then the duration needed to produce $\omega\kappa$ is at least the sum of the needed durations, so we have $S(\omega).T(\kappa) \leq S(\omega\kappa)$ as required.

A term in the product S.R has the form $+\infty\ \omega\kappa$, where ω is a string in S and κ is a string in R. This means that the output κ is impossible from any initial state, thus the output $\omega\kappa$ is impossible for the automaton, thus its coefficient in S is $+\infty$ as required.

In conclusion of this part of the demonstration, S is a solution of equation 4-1, thus it is greater or equal than the smallest solution X°.

Second part of the demonstration:

To complete the proof, we need to show that X° is larger than S for the partial order.

To do this, we are going to show, by induction on n, the following assertion:

> <u>For any string ω of length smaller or equal than n, its coefficient in S is smaller or equal that its coefficient in X°.</u>

This is true for n=0, when the only possible string is the empty string, which has coefficient e in both cases.

<u>We now suppose the assertion is true for a fixed n, let's consider a string ω of length n+1.</u>

If n=0, the string is a single letter and the result is obvious: from equation 3-1, we deduce that X° is larger than $P \oplus Q$, which is, by definition, equal to the sum of the terms of degree 1 in S. So we now suppose that $n \geq 2$.

If the output w is possible, we can split w in $\upsilon\kappa$ where υ is a possible output of length n-1, and κ is a letter which appears in T.

From the recursion hypothesis, we know that $S(\upsilon) \leq X^\circ(\upsilon)$.

Then $S(\upsilon\kappa) \leq S(\upsilon).T(\varpi)$ because $S(\upsilon\kappa)$ is the smallest delay needed to produce $\upsilon\kappa$, and the decomposition υ followed by κ is only one of the possible decompositions.

Since X° is solution of equation 4-1, which implies that X° is larger than $X^\circ.T$, we deduce that:

$X^\circ(\upsilon\kappa) \geq X^\circ(\upsilon).T(\kappa)$, thus we obtain $S(\upsilon\kappa) \leq X^\circ(uv)$.

If the output $\upsilon\kappa$ is impossible, its coefficient in S is $+\infty$, thus we need to show that its coefficient in X° is $+\infty$ too.

If ω is impossible, ω must contain a substring u of length 2, made of two outputs which cannot be produced successively. Thus the term $\infty.\omega$ appears in R.

If we split ω in $\upsilon.\kappa.\lambda_1.\ \lambda_2.\ ....\ \lambda_p.$, where the λ_i are letters, we can deduce from equation 4-1 that:

$X^\circ(\omega) \geq X^\circ(\upsilon).R(\kappa).\ T(\lambda_1).\ T(\lambda_2).\ ....\ T(\lambda_p)$

By definition of T, none of the terms $T(\lambda_1)\ \ T(\lambda_2).\ ....\ T(\lambda_p)$ is equal to ε. The recursion hypothesis implies that $X^\circ(\upsilon) \geq 0$.

Because R(u) is $+\infty$, we finally have $X^{\circ}(w) \geq +\infty$.

This completes the proof.

5 The case of Moore machines.

Another class of automata with outputs is given by Moore machines: in this model, outputs are genereted when the automaton enters a new state:

Definition 5.1 :

A Moore machine is a 6-uple (Q , Σ , Γ, q , δ , λ) where:

Q is a finite set of states,
Σ is a finite input alphabet,
q is the initial state, an element of Q
δ is the transition function, mapping Q X Σ into Q.
Γ is a finite output alphabet
λ is a function mapping Q on Γ.

The main difference with mealy machines is that the initial output appears first in any possible output string.

It can be proven that Mealy and Moore machines are equivalent, in the sense that:

Every system modelled by a Mealy machine can be modelled by a Moore machine and vice-versa.

Moore machines are less convenient for our purpose, because they generate output strings which are one unit longer than those generated by a Mealy machine. We shall briefly indicate how to apply our results to Moore machines.

In Moore machines, outputs are associated to the states and not to transitions. Therefore, it is natural to temporize states instead of transitions:

To each state, we shall associate a minimal stay of the token , before it becomes available for another transition.
We shall suppose that the token is initially available at time 0, in state q.

τ is still a mapping from Γ onto the dioïd D.

If we suppose that the mapping λ of states on the output alphabet is one-to-one, we can identify the states with the outputs, and talk of the duration of a state.

Theorem 5.1 :

Let Γ be the output alphabet of a Moore machine (Q , Σ , Γ, q , δ , λ), where λ is one-to-one. Let S be the formal series, with coefficients in DRC and variables in Γ such that the coefficient of any string ω is the least time necessary to produce ω . Then:

S is rational in the sense of DRC and can be computed by the formula:

$$S = (\lambda(q) \oplus U)\,(e \oplus (P \oplus Q) \,.\, (e \oplus T \oplus R)^*)$$

Where q is the initial state and P,Q and R are the following polynomials:

P is the sum of all states which are reachable from the initial state, multiplied by their minimal duration.

Q is the sum of all single outputs which are impossible from the initial state, multiplied by $+\infty$.

T is the sum of all states multiplied by their duration,

R is the sum of all strings of 2 states, which cannot be succesively reached , multiplied by $+\infty$.

U is the sum of all ouputs different from $\lambda(q)$, which is the output of the initial state, multiplied $+\infty$.

6 Remarks on the restrictions and the algebra.

What happens if an output letter could be produced by different state transitions?

If those transitions have different durations, a minimization problem occurs: two strings of outputs may be produced with different durations; and we are looking for the minimal time needed to perform outputs. This would imply the use of the min operator, so question may be reformulated in another way:

Why use (max,+) algebra and not (min,+)?

There are three reasons for this:

$+\infty$ needs to be an absorbing element for the $\approx$ of the dioïd, because the finite coefficients have to be cancelled if a string is impossible. Unfortunatly, $+\infty$ is the neutral element for the min.

To compute the series associated to the Mealy machine, we introduce an implicit equation in the (max,+) algebra; the series we are looking for is the minimal solution of this equation ; this is essential for the proof, because it shows that we can use the resolvent formula to compute the series. If we were working with (min,+), the resolvent formula would give us a maximal solution.

The (max,+) algebra has been used by Cohen et al. to describe discrete-event systems with only synchronisations problems. Using the same algebra gives us the hope to link both models to apply this calculus to more general models.

To generalize this result to automata with several transitions producing the same outputs, we shall have to assume that one outout is performed in a fixed duration, whatever state transition has produced it,.

Another point worth noticing, is that all these results still hold when the <u>real time is discrete</u>:
$(\mathbb{R} \cup \{+\infty,-\infty\}, \max, +)$ is replaced by $(Z \cup \{+\infty,-\infty\}, \max, +)$

7 Conclusions and new directions of research.

We have shown that the so-called "max-algebra" can be applied to the temporal description of finite-state of automata. The tools used, rational series, is an extension of regular languages, which was used by Wonham and Ramadge to solve some control problems in refrences 7 to 11.

These models, finite automata, are only a sub-class of all possible discrete-even systems: no synchronisation problems are modelled. The interesting point is that the same algebra has been used by Cohen et al. in references 2 to 5, to model discrete-event systems where only synchronisation problems occur, which is the other "extreme case".

This gives us the hope that this mathematical theory, rational series in the max-algebra, can be applied to a broader class of discrete-event systems, containing the two special cases we mentioned.

Introducing these series raises a new problem: the study the structure of rational series in in the max-algebra, with several variables. The case of one variable has been completely studied by Cohen et al., but all results do not seem to extend to the case of several variables, because these variables do not commute.

Another issue is to link timed-outputs to timed inputs. This was done by Cohen and All, who introduced the notion of transfer function for the class of systems they were able to model. Doing this will probably be much more difficult in the case of finite-state automata, since the output alphabet is different from the input alphabet.

References

[0] J.E. HOPCROFT, J.D. ULLMANN,
Introduction to Automata theory, languages and computation,
Addison-Wesley, Reading, Massachusetts, 1979.

[1] G. COHEN, D. DUBOIS, J.P. QUADRAT, M. VIOT,
Analyse du comportement périodique des sytstèmes de production par la théorie des dioïdes.
Rapport I.N.R.I.A N□ 191 ,Le Chesnay , 1983.

[2] G. COHEN, D. DUBOIS, J.P. QUADRAT, M. VIOT,
A linear-System-Theoretic view of Discrete-Event Processes.
22nd IEEE Conf. on Decision and Control ,
San Antonio , Texas , 1983.

[3] G. COHEN, D. DUBOIS, J.P. QUADRAT, M. VIOT,
A linear-System-Theoretic view of Discrete-Event Processes ands its use for Performance evaluation in Manufacturing.
IEEE Trans. on Automatic Control , 1985.

[4] G. COHEN, P. MOLLER, J.P. QUADRAT, M. VIOT,
Une théorie linéaire des systèmes à évènements discrets.
Rapport I.N.R.I.A N□ 362 ,Le Chesnay , Janvier 1985.

[5] G. COHEN, P. MOLLER, J.P. QUADRAT, M. VIOT,
Linear System Theory for Discrete Event Systems.
23rd IEEE conf. on Decision and Control ,
Las Vegas, Nevada, December 1984.

[6] R.A. CUNINGHAME-GREEN,
Minimax Agebra.
Lecture Notes in Economics and Mathématical Systems,
vol. 166, Springer Verlag, 1979.

[7] P.J. RAMADGE, W.M. WONHAM,
Supervisory control of a class of discrete-event systems.
To appear in the SIAM journal on Control and optimisation.

[8] P.J. RAMADGE, W.M. WONHAM,
Modular feedback logic for discrete event systems,
Information sciences and Systems Report N□ 48,
Department of Electrical Engineering,
Princeton University, Princeton N. J.

[9] P.J. RAMADGE, W.M. WONHAM,
On the supremal controlable sublangage of a given language.
Linear System Theory for Discrete Event Systems.
23rd IEEE conf. on Decision and Control ,
Las Vegas, Nevada, December 1984.

[10] P.J. RAMADGE, W.M. WONHAM,
On modular synthesis of supervisory control for discrete-event processes.
International conference on computers, Systems & signal processing, Bangalore, India, December 1984.

[10] P.J. RAMADGE, W.M. WONHAM,
Proceedings of the Seventh international conference on Analysis and optimisation of systems, Antibes, France,1986.
Published in Lecture Notes in Control and information sciences, Springer-Verlag, 1986.

[11] P.J. RAMADGE,
Control and supervision of Discrete event processes.
Ph. D. Thesis, Department of electrical engineering, University of Toronto, 1983.