Part I

Mathematical foundations of the theory of automata

FINITE AUTOMATA AND RATIONAL LANGUAGES AN INTRODUCTION

Jean Berstel

LITP
Université Pierre et Marie Curie
Paris, France

Introduction

This introductory text gives a short exposition of the basic results concerning finite automata and rational languages. Its aim is to fix notation for subsequent papers, and also to give a short account of some algorithmic aspects of Kleene's theorem.

The paper is divided into four parts. The first section gives the definition of automata and recognizable languages, and considers a series of basic constructions, such as the "subset construction" used to obtain a deterministic automaton. In the second section, the construction of the minimal automaton recognizing a given language is illustrated on two algorithms.

The third section is devoted to Kleene's theorem. The classical proof is sketched, and alternatives are discussed. This section also contains the definition of the transition monoid of an automaton, and a brief discussion of the definition of recognizable and rational sets in arbitrary monoids.

The final section is slightly more technical. It describes two algorithms for computing an automaton starting from a rational expression, using derivatives. The first method, due to Brzozowski, gives a deterministic automaton whilst the second, due to Berry and Sethi, computes a nondeterministic automaton of small size.

There are numerous expository texts on finite automata and rational languages available. One of the first is by Rabin, Scott [16]. The treatise of Eilenberg [7] contributed to fix modern terminology. Among recent texts, we just mention Autebert [3] and Perrin [13].

I. Automata

1. Notation

An *automaton* over some alphabet A is composed of a set Q of *states*, a set $I \subset Q$ of *initial states*, a set $T \subset Q$ of *terminal states* and a set $\mathcal{F} \subset Q \times A \times Q$ of *edges*. The automaton is *finite* if its set of states is finite. An automaton is usually denoted by

$$\mathcal{A} = (Q, I, T)$$

or $\mathcal{A} = (Q, I, T, \mathcal{F})$ when the set of edges is emphasized. The *label* of an edge $f = (p, a, q)$ is the letter a. A *path* of length n in $\mathcal{A}$ is a sequence $c = e_1 \cdots e_n$ of n consecutive edges $e_i = (p_i, a_i, q_i)$, i.e. such that $q_i = p_{i+1}$ for $i = 1, \ldots, n-1$. The label of the path c is $|c| = a_1 \cdots a_n$. We also write

$$c : p_1 \xrightarrow{|c|} q_n$$

By convention, there is an empty path $1_q : q \to q$ with label 1 for each state q. A path $c : i \to t$ is *successful* if $i \in I$, $t \in T$. A word is *recognized* if it is the label of a successful path. The *language recognized* by the automaton $\mathcal{A}$ is the set

$$L(\mathcal{A}) = \{w \in A^* \mid \exists c : i \to t,\ i \in I,\ t \in T,\ w = |c|\}$$

A set $X \subset A^*$ is *recognizable* if there exists a finite automaton $\mathcal{A}$ such that $X = L(\mathcal{A})$. The family of all recognizable subsets of A^* is denoted by $\mathrm{Rec}(A^*)$.

The terminology introduced already sketches the usual pictorial representation of an automaton: states are represented by the nodes of a graph related by the edges. An initial state is marked by an arrow entering in it, and a final state by an arrow leaving it.

EXAMPLE 1.— The automaton given in Fig.1 below recognizes the (restricted) *Dyck language* D^* composed of those words over $A = \{a, b\}$ which correspond to "correct bracketings".

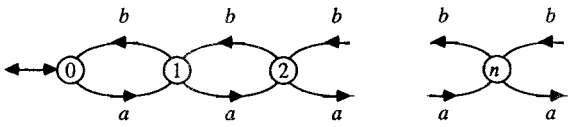

Fig. 1 An automaton for the Dyck language

EXAMPLE 2.— The automaton in Fig. 2 recognizes the set A^*aba, again with $A = \{a, b\}$.

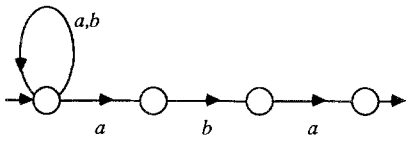

Fig. 2 An automaton for A^*aba

EXAMPLE 3.— For any given subset X of A^*, an automaton recognizing X is readily constructed as follows: the set of states is A^*, the unique initial state is the empty word 1, and X is the set of terminal states; the edges are the triples (w, a, wa) for $w \in A^*$ and $a \in A$.

2. More Definitions

Let $\mathcal{A} = (Q, I, T)$ be an automaton over A. It is called

- *unambiguous* if for p, $q \in Q$ and $w \in A^*$, there exists *at most* one path $p \to q$ with label w;
- *deterministic* if $\mathrm{Card}(I) = 1$ and if for $p \in Q$ and $w \in A^*$, there is *at most* one path starting in p and labelled w;
- *complete* if for $p \in Q$ and $w \in A^*$ ther exists *at least* one path starting in p and labelled w.

For a deterministic automaton $\mathcal{A} = (Q, I, T)$, it is convenient to denote $p \cdot w$ the state reached by the path starting in p and labelled w. If there is no such path, we write $p \cdot w = \emptyset$. This defines a (partial) function $Q \times A^* \to Q$ called the *transition function* or *next state function* of the automaton. With this notation, $L(\mathcal{A}) = \{w \in A^* \mid i \cdot w \in T\}$.

The *completion* of an incomplete automaton is very easy. It suffices to add a new state s, called some times a "sink" state, and to add the edge (p, a, s) whenever there is no edge labelled a and starting in p in the original automaton.

The *determinization* of an automaton is more involved. The following proposition holds:

PROPOSITION .— *For any automaton $\mathcal{A}$, there exists an equivalent deterministic automaton $\mathcal{B}$ i.e. that recognizes the same language. If $\mathcal{A}$ is finite and has n states, then $\mathcal{B}$ can be chosen with at most 2^n states.*

The proof is by the socalled "subset construction" and goes at follows: starting with an automaton $\mathcal{A} = (Q, I, T, \mathcal{F})$, one constructs a deterministic automaton $\mathcal{B} = (\mathcal{S}, I, \mathcal{T})$ by setting $\mathcal{S} = \mathcal{P}(Q)$, $\mathcal{T} = \{S \in \mathcal{S} \mid S \cap T \neq \emptyset\}$, the next state function being defined by

$$S \cdot a = \{q \in Q \mid \exists p \in S \ : \ (p, a, q) \in \mathcal{F}\}$$

Observe that if $\mathcal{A}$ is finite and has n states, then the automaton $\mathcal{B}$ also is finite, even if it may have 2^n states.

EXAMPLE 2(continued).— The subset construction gives the following deterministic automaton for A^*aba:

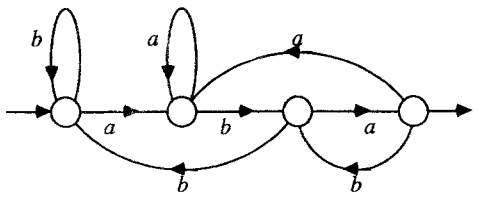

Fig. 3 A deterministic automaton for A^*aba

EXAMPLE 4.— The language $X_n = A^*aA^{n-1}$, with $A = \{a, b\}$ is recognized by the $n+1$-state automaton given below. It is easily shown that any deterministic automaton recognizing L_n has at least 2^n states.

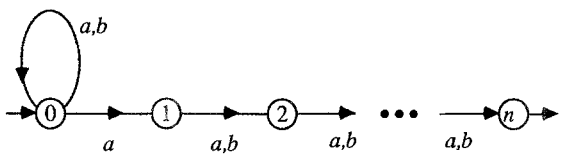

Fig. 4 An automaton for A^*aA^{n-1}

3. An Implementation

There is a well known implementation of finite automata in electronic circuitry called PLA (*plane logical array*). Such an array is composed of two parts, the "and" part, and the "or" part. There is a horizontal wire for each edge of the automaton, and two vertical wires for each state, one in the "and" part and the other one in the "or" part. Moreover, there is a vertical input wire for each letter in the "and" part.

The connexions are defined by the edges: if $e = (p, a, q)$ is an edge of the automaton, then the wire e is connected to the wires a and p in the "and" part, and to the wire q in the "or" part. Finally, each state wire in the "or" part is connected back to the corresponding wire in the "and" part. In Fig. 5, the PLA of the automaton of Example 2 is drawn.

The PLA works as follows. When some letter a is input, its wire is activated (in some electric sense). For each activated state wire p in the "and" part, one activates those edge wires e which are connected to a *and* to p. These activated edges e now in turn activate those state wires q in the "or" part which are connected to e. It is easily seen that, starting with a set S of activated states in the "and" part and an input letter a the set of states activated in one step in the "or" part is $S \cdot a = \{q \in Q \mid \exists p \in S : (p, a, q) \in \mathcal{F}\}$. Our presentation is from [14]. In fact, the physical realization of a PLA is more involved. Interested readers are referred to Mead, Conway [11].

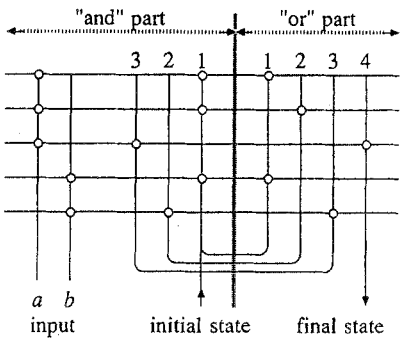

Fig. 5 A PLA for A^*aba

II. Minimal Automaton

1. Reduction

An automaton $\mathcal{A} = (Q, I, T)$ is *trim* if all its states are accessible and coaccessible, that is, if for any state $q \in Q$, there exist paths $i \to q$ and $q \to t$ for some $i \in I$ and $t \in T$. It is easy to trim an automaton: just remove all states that are not accessible or not coaccessible.

Let $\mathcal{A} = (Q, i, T)$ be a deterministic automaton recognizing some language $X \subset A^*$. The *Nerode equivalence* is an equivalence relation over Q defined by

$$p \sim q \iff L_p(\mathcal{A}) = L_q(\mathcal{A})$$

where $L_p(\mathcal{A}) = \{w \in A^* \mid p \cdot w \in T\}$. It is straightforward that this relation is right regular, i.e.

$$p \sim q \implies p \cdot w \sim q \cdot w$$

and that terminal states are saturated :

$$p \in T,\ p \sim q \implies q \in T$$

Consequently, one may define a *quotient automaton* $\mathcal{A}/\sim$ whose states are the equivalence classes of the relation, and with next state function induced by that of $\mathcal{A}$. The quotient automaton also recognizes X..

For a trim automaton $\mathcal{A}$, the quotient automaton $\mathcal{A}/\sim$ depends only on the recognized language X, and thus is unique up to a renaming of the states. Moreover, it is minimal with respect to the number of states among all automata recognizing the language X, and therefore is called the *minimal automaton* of X.

The computation of the minimal automaton of a language X can be carried out in several ways. One algorithm, called the *reduction algorithm* and due to Moore [12], starts with some automaton $\mathcal{A}$ for the language L and computes successive approximations of the Nerode equivalence. For two states p and q of $\mathcal{A}$, and any integer $k \geq 0$, define

$$p \sim_k q \iff L_p^{(k)} = L_q^{(k)}$$

where $L_p^{(k)} = \{w \in L_p \mid |w| \leq k\}$. The following facts are easily proved by induction:

(i) $p \sim_{k+1} q \iff p \sim_k q$ and $(\forall a \in A,\ p \cdot a \sim_k q \cdot a)$

(ii) if $\sim_k$ and $\sim_{k+1}$ are identical, then all $\sim_{k+n}$, for $n \geq 0$ are equal, and are the Nerode equivalence.

As a consequence, for an automaton with N states, it suffices to compute $\sim_{N-2}$ in order to get the Nerode equivalence and thus the minimal automaton. A careful implementation of this algorithm has been proposed by Hopcroft (see [1]) who proves that it can be carried out in time $O(N \log N)$ for an N-state automaton.

2. A Characterization

We consider now another way to define the minimal automaton of a language. It works directly on the language, and thus is by definition unique. For words u and v, the *left quotient* is defined by

$$u^{-1}v = \begin{cases} w & \text{if } v = uw \\ \emptyset & \text{otherwise} \end{cases}$$

This notation is extended to a subset $X \subset A^*$ by

$$u^{-1}X = \bigcup \{u^{-1}x \mid x \in X\}$$

Clearly

$$(uv)^{-1}X = v^{-1}(u^{-1}X), \quad 1^{-1}X = X$$

An automaton

$$\mathcal{A}(X) = (Q, i, T)$$

is constructed by setting

$$\begin{aligned} Q &= \{u^{-1}X \mid u \in A^*,\ u^{-1}X \neq \emptyset\} \\ i &= X \\ T &= \{u^{-1}X \mid 1 \in u^{-1}X\} \end{aligned}$$

The next state function is defined by

$$Y \cdot a = a^{-1}Y \qquad (Y \in Q,\ a \in A)$$

PROPOSITION .— *The automaton $\mathcal{A}(X)$ is the minimal automaton of X.*

In the case X is recognizable, this gives the following characterization:

PROPOSITION .— *The language X is recognizable if and only if the set $\{u^{-1}X \mid u \in A^*\}$ is finite.*

EXAMPLE.— Let $X = b^*aA^*$, with $A = \{a, b\}$. Then

$$a^{-1}X = A^*, \quad b^{-1}X = X,$$
$$a^{-1}A^* = b^{-1}A^* = A^*$$

thus the minimal automaton of X has just two states, namely X and A^*.

III. Kleene's Theorem

1. Rational Languages

Let A be an alphabet. The *rational operations* over the subsets of A^* are the following:

union $X \cup Y$
product $XY = \{xy \mid x \in X,\ y \in Y\}$
star X^* = the submonoid generated by X

A family of subsets of A^* is *rationally closed* if it is closed for the three rational operations. The *rational languages* of A^* are the elements in the smallest rationally closed family of A^* containing the singletons and the empty language. This family is denoted by $\mathrm{Rat}(A^*)$.

2. Kleene's Theorem

The following result is due to Kleene [9]:

THEOREM.— *Rational and recognizable languages over a finite alphabet A coincide:* $\mathrm{Rec}(A^*) = \mathrm{Rat}(A^*)$.

The proof is in two parts. The first part consists in showing that every rational languange is recognizable, i. e. that $\mathrm{Rat}(A^*) \subset \mathrm{Rec}(A^*)$. There are several constructions to do that. We sketch the wellknown method of constructing an appropriate automaton for each rational language. For this, we call an automaton $\mathcal{A}$ *normalized* if the two following conditions hold :

(i) There is a unique initial state i, and no edge enters i;
(ii) There is a unique final state $t \neq i$, and no edge leaves t.

It is easy to construct, for any recognizable language, a normalized automaton recognizing this language up to the empty word.

Given two normalized automata $\mathcal{A}$ and $\mathcal{A}'$ over distinct sets of states recognizing the languages X and X' respectively, an automaton recognizing $X \cup X'$ is obtained in pasting together the initial states i and i', and the final states t and t' of the automata. An automaton recognizing XX' is obtained by setting $i' = t$, and by taking i and t' as the initial and the final state of the resulting automaton. Finally, an automaton recognizing X^* is derived from $\mathcal{A}$ by identifying i and t (the resulting automaton is no longer normalized). These constructions show that $\mathrm{Rec}(A^*)$ is rationally closed. Of course, singletons and the empty set are recognizable, and therefore $\mathrm{Rat}(A^*) \subset \mathrm{Rec}(A^*)$.

Among the variations of this construction, let us mention the following: instead of pasting together states, special edges labelled by the empty word are used to connect

states. Thus, in order to recognize $X \cup X'$, two additional states are introduced; a new initial state j connected to i and to i' by edges labelled by 1, and similarly a new final state. The feature of this construction is that almost two edges are leaving any state, making thus the implementation particularly easy (this is knowm as Thomson's construction, see Aho, Hopcroft, Ullman [1] for a more detailed discussion).

One way to prove the converse inclusion, namely $\mathrm{Rec}(A^*) \subset \mathrm{Rat}(A^*)$ is to use the following procedure, known as the algorithm of McNaughton and Yamada [10]. Consider a finite automaton $\mathcal{A} = (Q, I, T)$ recognizing a language X, and number the states such that $Q = \{1, \ldots, n\}$. For i, j in Q, let $X_{i,j} = \{w \mid i \xrightarrow{w} j\}$, and for $k = 0, \ldots, n$, let $X_{i,j}^{(k)}$ be the set of labels of nontrivial paths of the form

$$i \to p_1 \to \ldots \to p_s \to j, \qquad s \geq 0, \quad p_1, \ldots, p_s \leq k$$

The following formula give the proof:

$$X_{i,j}^{(0)} \subset A$$
$$X_{i,j}^{(k+1)} = X_{i,j}^{(k)} \cup X_{i,k+1}^{(k)} \left(X_{k+1,k+1}^{(k)}\right)^* X_{k+1,j}^{(k)}$$

since

$$X_{i,j} = \begin{cases} X_{i,j}^{(n)} & \text{if } i \neq j \\ 1 \cup X_{i,j}^{(n)} & \text{if } i = j \end{cases}$$
$$X = \bigcup_{\substack{i \in I \\ t \in T}} X_{i,t}$$

Since the alphabet is finite, the sets $X_{i,j}^{(0)}$ are rational, the other $X_{i,j}^{(k)}$ are in their rational closure, and therefore are rational. This achieves the proof.

A similar method is useful in hand calculations. Set $X_q = \bigcup_{t \in T} X_{q,t}$. Then $X = \bigcup_{i \in I} X_i$, and the sets X_q satisfy the system of linear equations

$$X_q = \bigcup_{p \in Q} A_{q,p} X_p \cup \Delta_{q,p} \qquad (q \in Q)$$

where $A_{q,p} = X_{i,j}^{(0)}$ and $\Delta_{q,p} = \emptyset$ if $q \neq p$, and $= \{1\}$ otherwise. This system of equations can be solved by Gaussian elimination.

3. Transition Monoid

Let $\mathcal{A} = (Q, I, T)$ be an automaton over A recognizing a language X. Each word $w \in A^*$ defines a relation $\phi(w) \subset Q \times Q$ by $(p, q) \in \phi(w)$ iff there exists a path $c : p \xrightarrow{w} q$. Since $\phi(uv) = \phi(u)\phi(v)$, the function ϕ is a morphism from A^* into the monoid $\mathcal{P}(Q \times Q)$ of relations over Q. Moreover ϕ *saturates* the language X, that is, if $u \in X$ and $\phi(u) = \phi(v)$, then $v \in X$. In other words, $X = \phi^{-1}\phi(X)$. The monoid $M(\mathcal{A}) = \phi(A^*)$ is called the *transition monoid* of the automaton $\mathcal{A}$. Clearly, the automaton $\mathcal{A}$ is

unambiguous iff $M(\mathcal{A})$ is a monoid of unambiguous relations,
deterministic iff $M(\mathcal{A})$ is a monoid of partial functions,
complete iff $M(\mathcal{A})$ is a monoid of total relations.

Of course, $M(\mathcal{A})$ is finite if X is recognizable. Conversely, a language $X \subset A^*$ is recognizable if there exists a *finite* monoid M and a morphism $\phi : A^* \to M$ such that $X = \phi^{-1}\phi(X)$.

In the case $\mathcal{A} = \mathcal{A}(X)$ is the minimal automaton of X, the monoid $M(\mathcal{A}(X))$ depends only on X. It is called the *syntactic monoid* of X, and can be defined without any reference to an automaton as the quotient monoid of A^* by the *syntactic congruence* of X defined by

$$u \equiv_X v \iff C_X(u) = C_X(v)$$

where the set $C_X(u)$ of *contexts* of u is defined as $C_X(u) = \{(w, w') \mid wuw' \in X\}$.

4. Arbitrary Monoids

Rational and recognizable sets can be defined in any monoid. Let M be a monoid. A subset X of M is *recognizable* if there exists a morphism $\phi : M \to N$, where N is a *finite* monoid, such that $X = \phi^{-1}\phi(X)$. The set of recognizable subsets of M is denoted by $\mathrm{Rec}(M)$. As in the case of a free monoid, the set $\mathrm{Rat}(M)$ of *rational subsets* of M is the smallest family of subsets of M containing the empty set and the singletons, and closed for the rational operations in $\mathcal{P}(M)$, i. e. closed under union, product of two subsets, and the star, which produces the submonoid generated by a given subset.

EXAMPLE .— Consider $M = \mathbb{Z}$. All finite monoids that are homomorphic images of $\mathbb{Z}$ are of the form $\mathbb{Z}/n\mathbb{Z}$. Thus, the recognizable subsets of $\mathbb{Z}$ are finite unions of infinite arithmetic progressions. This shows that $\mathrm{Rec}(\mathbb{Z})$ is strictly included in $\mathrm{Rat}(\mathbb{Z})$.

It is not difficult to show that for any finitely generated monoid M, the inclusion $\mathrm{Rec}(M) \subset \mathrm{Rat}(M)$ holds. For commutative monoids, a rather complete description of rational sets has been given by Eilenberg, Schützenberger [8].

An interesting open problem is to characterize *Kleene monoids*, that is those monoids M for which Kleene's there holds, i. e. $\mathrm{Rec}(M) = \mathrm{Rat}(M)$. Several examples of Kleene monoids have been given. The first is due to Amar and Putzelu [2]; a more general theory has been constructed in Sakarovitch [17] Pelletier, Sakarovitch [15].

IV. Rational Expressions

1. Definition

The *rational expressions* over some alphabet A are the elements of some quotient algebra $\mathcal{E}$ of the term algebra over the set $A \cup \{0, 1\}$ with function symbols $+, \cdot, *$. For instance, $a(a \cup a \cdot b)^*b$ is a rational expression.

There is a mapping L from this term algebra onto the algebra of rational subsets of A^* defined inductively as follows

$$\begin{aligned} &L(0) = \emptyset, \quad L(1) = \{1\}, \quad L(a) = \{a\}, \\ &L(e + e') = L(e) \cup L(e'), \quad L(e \cdot e') = L(e)L(e'), \\ &L(e^*) = L(e)^* \end{aligned}$$

Consider the quotient algebra $\mathcal{E}$ obtained by agreeing that $+$ is idempotent, associative and commutative, that $\cdot$ is associative, and that

$$0 + e = e = e + 0$$
$$1 \cdot e = e = e \cdot 1$$
$$0 \cdot e = 0 = e \cdot 0$$

The function L is still defined on the quotient algebra $\mathcal{E}$. There is a remarkable result of Conway [6] stating that the algebra of rational sets cannot be obtained from the algebra of rational expressions $\mathcal{E}$ by a finite set of relations.

2. Derivatives

The computation of a finite automaton recognizing the language $L(e)$ for some given rational expression e has been sketched in the previous section (in the proof of Kleene's theorem). We give here an alternative method, which is inspired by the characterization of the minimal automaton. This also gives an alternative proof to the fact that rational languages are recognizable. Set $X = L(e)$. Then the states of the minimal automaton $\mathcal{A}(X)$ are the sets $u^{-1}X$, for $u \in A^*$. The *sets* can be computed via some rules described below, depending on the rational structure of the set. On the other side, similar operations, called the *derivation* rules, can be defined on the rational expression, in such a way that the equality

$$u^{-1}L(e) = L(u^{-1}e)$$

holds for any regular expression e, and any word u. These rules are the following:

Languages (*properties*)	*Expressions* (*definition*)
$a^{-1}\emptyset = \emptyset$	$a^{-1}0 = 0$
$a^{-1}1 = \emptyset$	$a^{-1}1 = 0$
$a^{-1}a = \{1\}$	$a^{-1}a = 1$
$a^{-1}b = \emptyset \quad (b \neq a)$	$a^{-1}b = 0 \quad (b \neq a)$
$a^{-1}(X \cup Y) = a^{-1}X \cup a^{-1}Y$	$a^{-1}(e + f) = a^{-1}e + a^{-1}f$
$a^{-1}(XY) = a^{-1}XY \cup (X \cap 1)a^{-1}Y$	$a^{-1}(ef) = a^{-1}e \cdot f + \delta(e)a^{-1}f$
$a^{-1}X^* = (a^{-1}X)X^*$	$a^{-1}e^* = a^{-1}e \cdot e^*$

Here $\delta(e)$ is equal to 0 or 1 according to $1 \notin L(e)$ or $1 \in L(e)$. Observe that this can also be defined just on the structure of the expressions e. In order to compute the minimal automaton from the sets $u^{-1}X$, one must decide whether two sets $u^{-1}X$ and $v^{-1}X$ are equal. This cannot be done easily. On the other hand, it is easy to decide whether two *expressions* $u^{-1}e$ and $v^{-1}e$ are equal in the algebra $\mathcal{E}$. The fact which turns the computation of derivatives of expressions into an algorithm for computing an automaton is the following theorem:

THEOREM (Brzozowski[5]).— *For any rational expression e, the set $\mathcal{D}(e) = \{u^{-1}e \mid u \in A^*\}$ of derivatives of e is finite.*

Proof.— For $w \in A^*$, one has

$$w^{-1}(ef) = (w^{-1}e)f + \sum_{\substack{w=uv \\ v\neq 1}} \delta(u^{-1}e)v^{-1}f$$

and for $w \in A^+$,

$$w^{-1}(e^*) = \sum_{\substack{w=uv \\ v\neq 1}} \delta(u,w)(v^{-1}e)e^*$$

where $\delta(u,w)$ is 0 or 1. These formula show that the set of rational expressions e for which the set $\mathcal{D}(e)$ is finite is closed under product and star. Since it is closed under sum and contains the basic expressions, the result follows. ∎

The set $\mathcal{D}(e)$ is turned into an automaton $\mathcal{A}(e)$ recognizing $L(e)$ by taking e as an initial state, those $u^{-1}e$ with $\delta(u^{-1}e) = 1$ as final states, and by defining the next state fuction as $f \cdot a = a^{-1}f$. This automaton is deterministic, but is not necessarily minimal.

3. Linear Expressions

There is a variation of the preceding algorithm, due to Berry and Sethi [4] that constructs a nondeterministic automaton which is "small".

A rational expression e is *linear* if all letters occurring in it are distinct. For instance, $(ab + c)^*dg$ is linear. Any expression can be linearized just by renaming all its letters by distinct symbols. The language denoted by the original expression is a morphic image of the new expression. If an automaton for the new language is found, a (usually nondeterministic) automaton for the old language is obtained by identifying the corresponding letters.

Given a letter a and any expression e, a *continuation* of a in e is any expression $(wa)^{-1}e \neq 0$. Assume $(wa)^{-1}e$ is a continuation. Then $(wa)^{-1}e = a^{-1}(w^{-1}e) = (w^{-1}e) \cdot a$, and consequently there is an edge $w^{-1}e \xrightarrow{a} (wa)^{-1}e$ in the automaton $\mathcal{A}(e)$.

PROPOSITION (Berry, Sethi[4]).— *If e is a linear expression, then any two continuations $(wa)^{-1}e$ and $(w'a)^{-1}e$ are equal.*

This proposition has an easy interpretation. It means that, in the associated automaton $\mathcal{A}(e)$, all edges labelled by the letter a have the same final vertex. Thus there is a bijection between the states of $\mathcal{A}(e)$ (excepted the initial state) and the alphabet of letters occurring in e. In particular, $\mathcal{A}(e)$ has exactly $1 + n$ letters, where n is the number of letters occurring in e.

There is a systematic way to compute the set of edges if $\mathcal{A}(e)$. For this we observe first the following

LEMMA .— *Let $X \subset A^*$. The following conditions are equivalent:*
(i) X is a local rational language;
(ii) For all $a \in A$, the set $\{(ua)^{-1}X \mid u \in A^*,\ (ua)^{-1}X \neq \emptyset\}$ is a singleton or is empty.

In particular, this shows that for a linear expression e, the language $L(e)$ is local. It therefore suffices to compute the following three sets:

$$\begin{aligned} \mathit{first}(e) &= \{a \in A \mid aA^* \cap L(e) \neq \emptyset\} \\ \mathit{last}(e) &= \{a \in A \mid A^*a \cap L(e) \neq \emptyset\} \\ \mathit{follow}(e) &= \{(a,b) \in A \times A \mid A^*abA^* \cap L(e) \neq \emptyset\} \end{aligned}$$

The automaton then has final states $\mathit{last}(e)$, and the edges

$$\begin{aligned} 1 &\xrightarrow{a} a \quad \text{for} \quad a \in \mathit{first}(e) \\ a &\xrightarrow{b} b \quad \text{for} \quad (a,b) \in \mathit{follow}(e) \end{aligned}$$

The computation of $\mathit{first}(e)$ and $\mathit{last}(e)$ is easy. For the set $\mathit{follow}(e)$, we introduce and auxiliary function ($\mathcal{E}$ is the algebra of expressions)

$$\Phi : \mathcal{E} \times \mathcal{E} \to \mathcal{P}(A \times A)$$

by

$$\begin{aligned} &\Phi(0,f) = \Phi(1,f) = \emptyset \\ &\Phi(a,f) = \{a\} \times \mathit{first}(f) \\ &\Phi(e+e',f) = \Phi(e,f) \cup \Phi(e',f) \\ &\Phi(e \cdot e',f) = \Phi(e,e' \cdot f) \cup \Phi(e',f) \\ &\Phi(e^*,f) = \Phi(e,e+f) \end{aligned}$$

It is easily shown that

$$\Phi(e,f) = \mathit{follow}(e) \cup \mathit{last}(e) \times \mathit{first}(f)$$

and consequently $\mathit{follow}(e) = \Phi(e,0)$. The interest of this computation rule is that it can be realized directly on the tree representing the rational expression by a tree traversal: The attribute of the root of the tree being $\Phi(e,0)$, its evaluation requires the evaluation of the attributes of its children; these are synthesized with an inherited value of the second argument.

References

[1] A. AHO, J. HOPCROFT, J. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] V. AMAR, G. PUTZOLU, Generalizations of regular events, *Inform. Contr.* **8** (1965), 56–63.

[3] J. M. AUTEBERT, *Langages algébriques*, Masson, 1987.

[4] G. BERRY, R. SETHI, From regular expressions to deterministic automata, *Theoret. Comput. Sci.* **48** (1986), 117–126.

[5] J. BRZOZOWSKI, Derivatives of regular expressions, *J. Assoc. Comput. Mach.* **11** (1964), 481–494.

[6] J. CONWAY, *Regular Algebra and Finite Machines*, Chapman and Hall, 1971.

[7] S. EILENBERG, *Automata, Languages and Machines*, Vol. A, Academic Press, 1974.

[8] S. EILENBERG, M. P. SCHÜTZENBERGER, Rational sets in commutative monoids, *J. Algebra* **13** (1969), 173–191.

[9] S. C. KLEENE, Representation of events in nerve nets and finite automata, *in:* C. Shannon, J. McCarthy (eds), *Automata Studies*, Princeton University Press, 1956, 3–41.

[10] R. MCNAUGTHON, H. YAMADA, Regular expressions and state graphs for automata, *IRE Trans. on Electronic Computers* **EC-9:1**, 1960, 39–47.

[11] C. MEAD, L. CONWAY, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[12] E. F. MOORE, Gedanken experiments on sequential machines, *in:* C. Shannon, J. McCarthy (eds), *Automata Studies*, Princeton Univ. Press, 1956, 129–153.

[13] D. PERRIN, Finite automata, *Handbook of Theoretical Computer Science*, North-Holland, (in press).

[14] D. PERRIN, J. SAKAROVITCH, Automates finis, Journées scientifiques et prix UAP, Edition scientifiques de l'UAP, vol. 1, Paris, 1988.

[15] M. PELLETIER, J. SAKAROVITCH, Easy multiplications II. Extensions of rational semigroups, Techn. Report 88-63, LITP, Paris.

[16] M. O. RABIN, D. SCOTT, Finite automata and their decision problem, *IBM J. Res. Devel.* **3**, 1959, 114–125.

[17] J. SAKAROVITCH, Easy multiplications I. The realm of Kleene's theorem, *Inf. Comput.* **74**, 1987, 173–197.