Automata and rational expressions

Jacques Sakarovitch

Laboratoire Traitement et Communication de l'Information
C.N.R.S. and Télécom ParisTech
email: `sakarovitch@telecom-paristech.fr`

2010 Mathematics Subject Classification: 68Q45

Key words: Finite automata, regular expressions. Rational sets, recognisable sets.

Contents

1 A new look at Kleene's theorem

Not very many results in computer science are recognised as being as basic and fundamental as *Kleene's theorem*. It was originally stated as the equality of two sets of objects, and is still so, even if the names of the objects have changed — see for instance Theorem I.4.11 in Chap. I of this handbook. This chapter proposes a new look at this statement, in two ways. First, we explain how Kleene's theorem can be seen as the conjunction of *two results* with distinct hypotheses and scopes. Second, we express the first of these two results as the *description of algorithms* that relate the symbolic descriptions of the objects rather than as the *equality* of two sets.

A two step Kleene's theorem In Kleene's theorem, we first distinguish a step that consists in proving that the *set of regular* (or *rational*) *languages* is equal to the *set of languages accepted by finite automata* — a set which we denote by $\mathrm{Rat}\, A^*$. This seems already to be Kleene's theorem itself and is indeed what S. C. Kleene established in [34]. But it is not, if one considers — as we shall do here — that this equality merely states the equality of the expressive power of rational expressions and that of finite labelled graphs. This is universally true. It holds independently of the structure in which the labels of the automata or the atoms of the expressions are taken, in any monoids or even in the algebra of polynomials under certain hypotheses.

By the virtue of the numerous properties of finite automata over finitely generated (f.g., for short) free monoids: being apt to determinisation for instance, the family of languages accepted by such automata is endowed with many properties as well: being closed under complementation for instance. These properties are extraneous to the definition of the languages by expressions, and then — by the former result — to the definition by automata. It is then justified, especially in view of the generalisation of expressions and automata to other monoids and even to other structures, to set up a definition of a new family of languages by a new mean, that will extend in the case of other structures, these properties of the languages over f.g. free monoids. It turns out that the adequate definition will be given in terms of *representations by matrices of finite dimension*; we shall call the languages defined in that way the *recognisable languages* and we shall denote their family by $\mathrm{Rec}\, A^*$. The second step of Kleene's theorem consists then in establishing that finite automata are equivalent to matrix representations of finite dimension under the hypothesis that the labels of automata are taken in f.g. free monoids.

These two steps correspond to two different concepts: *rationality* for the first one, and *recognisability* for the second one. This chapter focusses on rationality and on the first step, namely the equivalence of expressiveness of finite automata and rational expressions. For sake of completeness however, we sketch in Sec. 2 how one gets from rational sets to recognisable sets in the case of free monoids and in Sec. 5, we see that the same construction fails in non-free monoids and what remains true.

The languages and their representation Formal languages or, in the weighted avatar, formal power series, are potentially *infinite objects*. We are only able to compute *finite* ones; here, expressions that denote, or automata that accept, languages or series. Hopefully, these expressions and automata are faithful description of the languages or series they stand for, all the more effective that one can take advantage of this double light.

In order to prove that the family of languages accepted by finite automata coincide with the one of languages denoted by rational expressions we proceed of course by establishing a double inclusion. As sketched in Figure 1[1], given an automaton $\mathcal{A}$ that accepts a language K, we describe algorithms which compute from $\mathcal{A}$ an expression F that denotes the same language K — I call such algorithms a Φ-*map*. Conversely, given an expression E that denotes a language L, we describe algorithms that compute from E an automaton $\mathcal{B}$ that accepts the same language L — I call such algorithms a Ψ-*map*.

Most of the works devoted to the conversion between automata and expressions address the problem of the *complexity* of the computation of these Φ- and Ψ-maps. I have chosen to study here the maps for themselves, how the results of different maps applied to a same argument are related, rather than to describe the way they are actually computed. The Φ-maps are considered in Sec. 3, the Ψ-maps in Sec. 4.

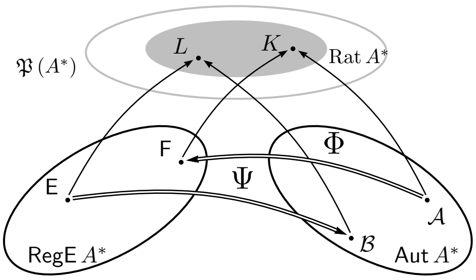


Figure 1. The Φ- and Ψ-maps

The path to generalisation The main benefit of splitting Kleene's theorem into two steps is to bring to light that the first one is a statement whose scope extends much beyond languages. It is first generalised to *subsets* of *arbitrary monoids* and then, with some precaution, to *subsets with multiplicity*, that is, to (formal power) *series*. This latter extension of the realm of Kleene's theorem is a matter for the same 'splitting' and distinction between series on arbitrary monoids and series on f.g. free monoids.

It would thus be possible to first set up the convenient and most general structure and then state and prove Kleene's theorem in that framework. My experience, however, is that many readers tend to be repelled and flee when confronted with statements outside the classical realm of *words*, *languages*, and *free monoids*. And this is where I shall stay in the first three sections of this chapter. The only difference with the classical exposition will be in the terminology and notation that will be carefully chosen or coined so that they will be ready for the generalisation to arbitrary monoids in Sec. 5 and to series in Sec. 6.

Notation and definitions given in Chap. I are used in this chapter without comment when they are refered to under the same form and with the exact same meaning.

[1] $\mathfrak{P}(A^*)$ denotes the power set of A^*, that is, the set of all languages over A^*.

2 Rationality and recognisability

We first introduce here a precise notion of *rational expression*, and revisit the definition of finite automata in order to fix our notation and to state, under the form that is studied here and eventually generalised later, what we have called above the 'first step of Kleene's theorem' and which we now refer to as the *Fundamental theorem of finite automata*. Second, we state and prove 'the second step' of Kleene's theorem in order to make the scope and essence of the first step clearer by contrast and difference.

2.1 Rational expressions

The set of *rational languages* of A^*, denoted[2] by $\mathrm{Rat}\, A^*$, is defined as in Chap. I: it is the smallest subset of $\mathfrak{P}(A^*)$ which contains the finite sets (including the empty set) and is closed under union, product, and star. The notion of *rational expression* allows to describe precisely how every element of this family can be built.

Definition 2.1. A *rational expression over* A^*, is a well-formed formula built inductively from the *constants* $\mathsf{0}$ and $\mathsf{1}$ and the letters a in A as *atomic formulas*, using two binary operators $+$ and $\cdot$ and one unary operator *: if E and F are rational expressions, so are $(\mathsf{E}+\mathsf{F})$, $(\mathsf{E}\cdot\mathsf{F})$, and (E^*). We denote by $\mathsf{RatE}\, A^*$ the set of rational expressions over A^* and often write *expression* for *rational expression*.

With every expression E in $\mathsf{RatE}\, A^*$ is associated a language of A^*, which is called *the language denoted by* E and we write[3] it as $|\mathsf{E}|$. The language $|\mathsf{E}|$ is inductively defined by $|\mathsf{0}| = \emptyset$, $|\mathsf{1}| = \{1_{A^*}\}$, $|a| = \{a\}$ for every a in A, $|(\mathsf{E}+\mathsf{F})| = |\mathsf{E}| \cup |\mathsf{F}|$, $|(\mathsf{E}\cdot\mathsf{F})| = |\mathsf{E}||\mathsf{F}|$, and $|(\mathsf{E}^*)| = \{|\mathsf{E}|\}^*$. Two expressions are *equivalent* if they denote the same language.

Proposition 2.1. *A language is rational if, and only if, it is denoted by an expression.*

Like any formula, an expression E is canonically represented by a tree, which is called *the syntactic tree* of E. Let us denote by $\ell(\mathsf{E})$ the *literal length* of the expression E (that is, the number of all occurences of letters from A in E) and by $\mathsf{d}(\mathsf{E})$ the *depth* of E which is defined as the depth — or height[4]— of the syntactic tree of the expression.

The classical precedence relation between operators: '$\,^* > \cdot > +\,$' allows to save parentheses in the writing of expressions: for instance, $\mathsf{E}+\mathsf{F}\cdot\mathsf{G}^*$ is an unambiguous writing for the expression $(\mathsf{E}+(\mathsf{F}\cdot(\mathsf{G}^*)))$. But one should be aware that, for instance, $(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G}))$ and $((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G})$ are two equivalent *but distinct* expressions. In particular, the *derivation* that we define at Sec. 4 yields different results on these two expressions.

In the sequel, any operator defined on expressions is implicitely extended additively to sets of expressions. For instance, it holds:

$$\forall X \subseteq \mathsf{RatE}\, A^* \qquad |X| = \bigcup_{\mathsf{E}\in X} |\mathsf{E}| \ .$$

[2]The empty word of A^* is denoted by 1_{A^*}.

[3]The notation $L(\mathsf{E})$ is more common, but $|\mathsf{E}|$ is lighter in computations and more appropriate when dealing with expressions over an arbitrary monoid or with weighted expressions.

[4]We rather not use *height* because of the possible confusion with the *star height*, *cf.* Sec. 4.

Definition 2.2. The *constant term* of an expression E over A^*, written $\mathsf{c}(\mathsf{E})$, is the Boolean value, effectively computable by a bottom-up traversal of the syntactic tree of E using the following equations:

$$\mathsf{c}(0) = 0\,, \quad \mathsf{c}(1) = 1\,, \quad \forall a \in A \quad \mathsf{c}(a) = 0\ ,$$
$$\mathsf{c}(\mathsf{F}+\mathsf{G}) = \mathsf{c}(\mathsf{F}) + \mathsf{c}(\mathsf{G})\,, \quad \mathsf{c}(\mathsf{F}\cdot\mathsf{G}) = \mathsf{c}(\mathsf{F})\mathsf{c}(\mathsf{G})\,, \quad \mathsf{c}(\mathsf{F}^*) = 1\ .$$

The *constant term* of a language L of A^* is the Boolean value $\mathsf{c}(L)$ that is equal to 1 if, and only if, 1_{A^*} belongs to L. By induction on $\mathsf{d}(\mathsf{E})$, $\mathsf{c}(\mathsf{E}) = \mathsf{c}(|\mathsf{E}|)$ holds.

2.2 Finite automata

We denote an *automaton over* A^* by $\mathcal{A} = \langle\, Q, A, E, I, T\,\rangle$ where, as in Chap. I, Q is the *set of states*, and is also called the *dimension* of $\mathcal{A}$, I and T are subsets of Q, and $E \subseteq Q{\times}A{\times}Q$ is the set of transitions labelled by letters of A. The automaton $\mathcal{A}$ is *finite* if E is finite; that is, A is finite, if, and only if, (the useful part of) Q is finite.

The *language accepted*[5] by $\mathcal{A}$, also called the *behaviour* of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is the set of words accepted by $\mathcal{A}$, that is, the set of labels of *successful computations*:

$$|\mathcal{A}| = \left\{ w \in A^* \;\middle|\; \exists i \in I, \exists t \in T \quad i \xrightarrow[\mathcal{A}]{w} t \right\}\ .$$

The first step of Kleene's theorem, which we call *Fundamental theorem of finite automata* then reads as follows.

Theorem 2.2. *A language of A^* is rational if, and only if, it is the behaviour of a finite automaton over A^*.*

Theorem 2.2 is proved by building connections between automata and expressions.

Proposition 2.3 (Φ-maps). *For every finite automaton $\mathcal{A}$ over A^*, there exist rational expressions over A^* which denote $|\mathcal{A}|$.*

Proposition 2.4 (Ψ-maps). *For every rational expression E over A^*, there exist finite automata over A^* whose behaviour is equal to $|\mathsf{E}|$.*

Sec. 3 describes how expressions are computed from automata, Sec. 4 how automata are associated with expressions. Before going to this matter, which is the main subject of this chapter, let us establish the second step of Kleene's theorem.

2.3 The 'second step' of Kleene's theorem

Let us first state the definition of recognisable languages, under the form that is given for recognisable subsets of arbitrary monoids (*cf.* Sec. I.5.2).

[5] We prefer not to speak of the language 'recognised' by an automaton, and would not say that a language is 'recognisable' when accepted by a finite automaton, in order to have a consistent terminology when generalising automata to arbitrary monoids

Definition 2.3. A language L of A^* is *recognised* by a morphism α from A^* into a monoid N if $L = \alpha^{-1}(\alpha(L))$. A language is *recognisable* if it is recognised by a morphism into a *finite* monoid. The set of *recognisable languages* of A^* is denoted by $\operatorname{Rec} A^*$.

Theorem 2.5 (Kleene). *If A is a finite alphabet, then* $\operatorname{Rat} A^* = \operatorname{Rec} A^*$.

The proof of this statement paves the way to further developments in this chapter. Let $\mathcal{A} = \langle Q, A, E, I, T \rangle$ be a finite automaton. The set E of transitions may be written as a $Q{\times}Q$-matrix, called the *transition matrix* of $\mathcal{A}$, also denoted by E, and whose (p, q)-entry is the set of letters that label transitions from p to q in $\mathcal{A}$. A fundamental (and well-known) lemma relates matrix multiplication and graph walking.

Lemma 2.6. *Let E be the transition matrix of the automaton $\mathcal{A}$ of finite dimension Q. Then, for every n in $\mathbb{N}$, E^n is the matrix of labels of paths of length n in $\mathcal{A}$:*

$$E^n_{p,q} = \left\{ w \in A^n \;\middle|\; p \xrightarrow[\mathcal{A}]{w} q \right\} \ .$$

The subsets I and T of Q may then be seen as Boolean vectors of dimension Q (I as a row and T as a column-vector). From the notation $E^* = \sum_{n\in\mathbb{N}} E^n$, it follows:

$$|\mathcal{A}| = I \cdot E^* \cdot T \ . \tag{2.1}$$

The next step in the preparation of the proof of Theorem 2.5 is to write the transition matrix E as $E = \sum_{a\in A} \mu(a)\, a$, where for every a in A, $\mu(a)$ is a Boolean $Q{\times}Q$-matrix and thus defines a morphism $\mu\colon A^* \to \mathbb{B}^{Q\times Q}$ (the Boolean semiring $\mathbb{B}$ has been defined at Chap. I). The second lemma, which involves the *freeness* of A^*, then reads:

Lemma 2.7. *Let $\mu\colon A^* \to \mathbb{B}^{Q\times Q}$ be a morphism and let $E = \sum_{a\in A} \mu(a)\, a$. Then, for every n in $\mathbb{N}$, $E^n = \sum_{w\in A^n} \mu(w)\, w$ and thus $E^* = \sum_{w\in A^*} \mu(w)\, w$.*

Proof of Theorem 2.5. By Theorem 2.2, a rational language L of A^* is the behaviour of a finite automaton $\mathcal{A} = \langle Q, A, E, I, T \rangle$. By (2.1) and Lemma 2.6, we write

$$L = |\mathcal{A}| = \{ w \in A^* \mid I \cdot \mu(w) \cdot T = 1 \} \ .$$

and thus $L = \mu^{-1}(S)$ where $S = \left\{ m \in \mathbb{B}^{Q\times Q} \;\middle|\; I \cdot m \cdot T = 1 \right\}$ and L is recognisable.

Conversely, let L be a recognisable language of A^*, recognised by the morphism $\alpha\colon A^* \to N$ and let $S = \alpha(L)$. Consider the automaton $\mathcal{A}_\alpha = \langle N, A, E, \{1_N\}, S \rangle$ where $E = \left\{ \big(m, a, m\,\alpha(a)\big) \;\middle|\; a \in A, m \in N \right\}$. It is immediate that

$$|\mathcal{A}_\alpha| = \left\{ w \in A^* \;\middle|\; \exists p \in S \quad 1_N \xrightarrow[\mathcal{A}]{w} p \right\} = \{ w \in A^* \mid \alpha(w) \in S \} = \alpha^{-1}(S) = L$$

and L is rational by Theorem 2.2. □

We postpone to Sec. 5 the example that shows that recognisability and rationality are indeed two distincts concepts and the description of the relationships that can be found between them. As mentionned in Chap. I, the following holds.

Theorem 2.8. *The equivalence of finite automata over A^* is decidable.*

Proposition 2.4 then implies:

Corollary 2.9. *The equivalence of rational expressions over A^* is decidable.*

3 From automata to expressions: the Φ-maps

For the rest of this section, $\mathcal{A} = \langle\, Q, A, E, I, T \,\rangle$ is a finite automaton over A^*, and E is viewed, depending on the context, as the *set of transitions* or as the *transition matrix* of $\mathcal{A}$. As in (2.1), the language accepted by $\mathcal{A}$ is conveniently written as

$$|\mathcal{A}| = I \cdot E^* \cdot T = \sum_{i \in I, t \in T} (E^*)_{i,t} \quad .$$

In order to prove that $|\mathcal{A}|$ is rational, it is sufficient to establish the following.

Proposition 3.1. *The entries of E^* belong to the rational closure of the entries of E.*

But we want to be more precise and describe procedures that produce for every entry of E^* a rational expression whose atoms are the entries of E (and possibly 1). There are (at least) four classical methods to proving Proposition 3.1, which can easily be viewed as algorithms serving our purpose and which we present here:

(1) Direct computation of $|\mathcal{A}|$: the *state elimination method* looks the most elementary and is indeed the easiest for both hand computation and computer implementation.
(2) Computation of $E^* \cdot T$ as a solution of a system of linear equations. Based on Arden's lemma, it also allows to consider $E^* \cdot T$ as a fixed point.
(3) Iterative computation of E^*: known as *McNaughton–Yamada algorithm* and probably the most popular among textbooks on automata theory.
(4) Recursive computation of E^*: based on Arden's lemma as well, this algorithm combines mathematical elegance and computational inefficiency.

The first three are based on an ordering of the states of the automaton. For *comparing* the results of these different algorithms, and of a given one when the ordering of states varies, we first introduce the notion of *rational identities*, together with the key Arden's lemma for establishing the correctness of the algorithms as well as the identities. The section ends with a refinement of Theorem 2.2 which, by means of the notions of *star height* and *loop complexity*, relates even more closely an automaton and the rational expressions that are computed from it.

3.1 Preparation: rational identities and Arden's lemma

By definition, all expressions which are computed from a given automaton $\mathcal{A}$ are *equivalent*. We may then ask whether, and how, this equivalence may be established *within the world of expressions itself*. We consider 'elementary equivalences' of more or less simple expressions, which we call *rational identities*, or *identities* for short, and which correspond to properties of (the semiring of) the languages denoted by the expressions. And we try to determine which of these identities, considered as *axioms*, are necessary, or sufficient, to obtain by substitution one expression from another equivalent one. It is known — and out of the scope of this chapter — that no finite sets of identities exist that allow to establish the equivalence of expressions in general (see Chap. XX). We shall see however that a *basic set of identities* is sufficient to deduce the equivalence between the expressions computed by the different Φ-maps described here.

Trivial and natural identities A first set of identities, that we call *trivial identities*, expresses the fact that $\mathsf{0}$ and $\mathsf{1}$ are interpreted as the zero and unit of a semiring:

$$\mathsf{E}+\mathsf{0} \equiv \mathsf{E}\,,\quad \mathsf{0}+\mathsf{E} \equiv \mathsf{E}\,,\quad \mathsf{E}\cdot\mathsf{0} \equiv \mathsf{0}\,,\quad \mathsf{0}\cdot\mathsf{E} \equiv \mathsf{0}\,,\quad \mathsf{E}\cdot\mathsf{1} \equiv \mathsf{E}\,,\quad \mathsf{1}\cdot\mathsf{E} \equiv \mathsf{E}\,,\quad \mathsf{0}^* \equiv \mathsf{1} \qquad (\mathbf{T})$$

An expression is said to be *reduced* if it contains no subexpressions which is a left-hand side of one of the above identities; in particular, $\mathsf{0}$ does not appear in a non-zero reduced expression. Any expression H can be rewritten in an equivalent reduced expression H'; this H' is unique and independent of the way the rewriting is conducted. From now on, all expressions are implicitely reduced, which means that *all the computations on expressions that will be defined below are performed modulo the trivial identities*.

The next set of identities expresses the fact that the operators $+$ and $\cdot$ are interpreted as the *addition* and *product* in a semiring:

$$(\mathsf{E}+\mathsf{F})+\mathsf{G} \equiv \mathsf{E}+(\mathsf{F}+\mathsf{G}) \qquad \text{and} \qquad (\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G} \equiv \mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G})\ , \qquad (\mathbf{A})$$

$$\mathsf{E}\cdot(\mathsf{F}+\mathsf{G}) \equiv \mathsf{E}\cdot\mathsf{F}+\mathsf{E}\cdot\mathsf{G} \qquad \text{and} \qquad (\mathsf{E}+\mathsf{F})\cdot\mathsf{G} \equiv \mathsf{E}\cdot\mathsf{G}+\mathsf{F}\cdot\mathsf{G}\ , \qquad (\mathbf{D})$$

$$\mathsf{E}+\mathsf{F} \equiv \mathsf{F}+\mathsf{E}\ . \qquad (\mathbf{C})$$

The set $\mathbf{A} \wedge \mathbf{D} \wedge \mathbf{C}$ is abbreviated as $(\mathbf{N})$ and called the set of *natural identities*.

Aperiodic identities The product in $\mathfrak{P}(A^*)$ is *distributive* over infinite sums; then

$$\forall K \in \mathfrak{P}(A^*) \qquad K^* = 1_{A^*} + K^*K = 1_{A^*} + K\,K^*\ , \qquad (3.1)$$

from which we deduce the identities:

$$\mathsf{E}^* \equiv \mathsf{1}+\mathsf{E}\cdot\mathsf{E}^* \qquad \text{and} \qquad \mathsf{E}^* \equiv \mathsf{1}+\mathsf{E}^*\cdot\mathsf{E}\ . \qquad (\mathbf{U})$$

From (**U**) and the gradation[6] of A^* follows Arden's lemma whose usage is ubiquitous.

Lemma 3.2 (Arden). *Let K and L be two subsets of A^*. Then K^*L is a solution of the equation $\mathrm{X} = K\,\mathrm{X} + L$. If $\mathsf{c}(K) = 0$, then K^*L is the* unique solution.

For computing *expressions*, we prefer to use Arden's lemma under the following form:

Corollary 3.3. *Let K and L be two rational expressions over A^* with $\mathsf{c}(\mathsf{K}) = 0$. Then, $|\mathsf{K}^*\mathsf{L}|$ denotes the unique solution of $\mathrm{X} = |\mathsf{K}|\,\mathrm{X} + |\mathsf{L}|$.*

The next two identities, called *aperiodic identities*, are a consequence of Lemma 3.2.

[6] That is, the elements of A^* have a *length* which is a morphism from A^* onto $\mathbb{N}$ (*cf.* Sec. 6).

Proposition 3.4. *For all rational expressions* E *and* F *over* A^*

$$(\mathsf{E}+\mathsf{F})^* \equiv \mathsf{E}^* \cdot (\mathsf{F} \cdot \mathsf{E}^*)^* \quad \textit{and} \quad (\mathsf{E}+\mathsf{F})^* \equiv (\mathsf{E}^* \cdot \mathsf{F})^* \cdot \mathsf{E}^* \ , \qquad (\mathbf{S})$$

$$(\mathsf{E} \cdot \mathsf{F})^* \equiv 1 + \mathsf{E} \cdot (\mathsf{F} \cdot \mathsf{E})^* \cdot \mathsf{F} \ . \qquad (\mathbf{P})$$

There are many other (independent) identities (*cf.* Notes). The remarquable fact is that those listed above will be sufficient for our purpose.

Identities special to $\mathfrak{P}(A^*)$ Finally, the *idempotency* of the union in $\mathfrak{P}(A^*)$ yields two further identities:

$$\mathsf{E}+\mathsf{E} \equiv \mathsf{E} \ , \qquad (\mathbf{I}) \qquad\qquad (\mathsf{E}^*)^* \equiv \mathsf{E}^* \ . \qquad (\mathbf{J})$$

In contrast with the preceding ones, these two identities (**I**) and (**J**) do not hold for expressions over arbitrary semirings of formal power series (*cf.* Sec. 6).

3.2 The state elimination method

The algorithm known as *state elimination method*, originally due to Brzozowski and McCluskey [12], works directly on the automaton $\mathcal{A} = \langle Q, A, E, I, T \rangle$. It consists in suppressing the states in $\mathcal{A}$, one after the other, while transforming the labels of the transitions so that the language accepted by the resulting automaton is unchanged (*cf.* [59, 60]).

A current step of the algorithm is represented at Figure 2. The left diagram shows the state q to be suppressed, a state p_i which is the origin of a transition whose end is q and a state r_j which is the end of a transition whose origin is q (it may be the case that $p_i = r_j$). By induction, the labels are *rational expressions*. The right diagram shows the automaton after the suppression of q, and the new label of the transition from p_i to r_j. The languages accepted by the automaton before and after the suppression of q are equal — a formal proof will follow in the next subsection.

Figure 2. One step in the state elimination method

More precisely, the state elimination method consists first in augmenting the set Q with two new states i and t, and adding transitions labelled with 1 from i to every initial state of $\mathcal{A}$ and from every final state of $\mathcal{A}$ to t. Then all states in Q are suppressed according to the procedure described above and in a certain order ω. At the end, there remain only the states i and t with a transition from i to t labelled with an expression which we denote by $\mathbf{B}_\omega(\mathcal{A})$ and which is the *result* of the algorithm. Thus it holds:

$$|\mathcal{A}| = |\mathbf{B}_\omega(\mathcal{A})| \ .$$

Figure 3 shows every step of the state elimination method on the automaton $\mathcal{D}_3$ drawn in the upper left corner and following the order $\omega = r < p < q$. It shows the result $\mathbf{B}_\omega(\mathcal{D}_3) = a^* b (b a^* b + a b^* a)^* b a^* + a^*$. One should be aware that the natural computation of $\mathbf{B}_\omega(\mathcal{A})$ may silently involve the identities **A**, **C**, and **I** as well.

The effect of the order The result of the state elimination method obviously depends on the order ω in which the states are suppressed. For instance, on the automaton $\mathcal{D}_3$ of Figure 3, the order $\omega_1 = r < p < q$ yields $\mathbf{B}_{\omega_1}(\mathcal{D}_3) = a^*b\,(b\,a^*b + a\,b^*a)^*b\,a^* + a^*$, $\omega_2 = r < q < p$ yields $\mathbf{B}_{\omega_2}(\mathcal{D}_3) = (a + b\,(a\,b^*a)^*b)^*$, and $\omega_3 = p < q < r$ yields $\mathbf{B}_{\omega_3}(\mathcal{D}_3) = a^* + a^*b\,(b\,a^*b)^*b\,a^* + a^*b\,(b\,a^*b)^*a\,(b + a\,(b\,a^*b)^*a)^*a\,(b\,a^*b)^*b\,a^*$.

All these expressions are equivalent modulo the aperiodic identities:

Theorem 3.5 (Conway [17], Krob [35]). *Let ω and ω' be two orders on the set of states of an automaton $\mathcal{A}$. Then,* $\mathbf{N} \wedge \mathbf{I} \wedge \mathbf{S} \wedge \mathbf{P} \quad \vdash \quad \mathbf{B}_{\omega}(\mathcal{A}) \quad \equiv \quad \mathbf{B}_{\omega'}(\mathcal{A})$ *holds.*

The question of *the length* of these expressions is also of interest, both from a theoretical as well as practical point of view (see Notes).

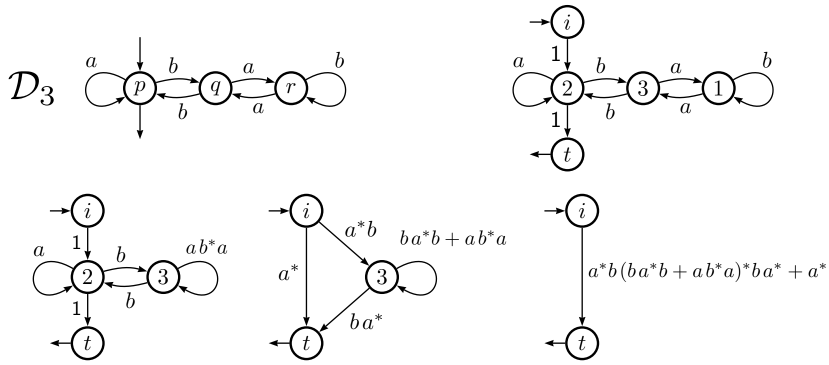


Figure 3. The state elimination method exemplified on the automaton $\mathcal{D}_3$

3.3 The system resolution method

The computation of an expression that denotes the language accepted by a finite automaton as the solution of a system of linear equations is nothing else than the state elimination method turned into a more mathematical setting.

Description of the algorithm Given $\mathcal{A} = \langle Q, A, E, I, T \rangle$, for every p in Q, we write L_p for the *set of words* which are the label of computations from p to a final state of $\mathcal{A}$: $L_p = \left\{ w \in A^* \,\middle|\, \exists t \in T \quad p \xrightarrow[\mathcal{A}]{f} t \right\}$. For a subset R of Q, we write the symbol $\delta_{p,R}$ for 1 if p is in R and 0 if not. The system of equations associated with $\mathcal{A}$ is written:

$$|\mathcal{A}| = \sum_{p \in I} L_p = \sum_{p \in Q} \delta_{p,I}\, L_p \quad (3.2) \qquad \forall p \in Q \quad L_p = \sum_{q \in Q} |\mathsf{E}_{p,q}| L_q + |\delta_{p,T}| \quad (3.3)$$

where the L_p are the 'unknowns' and the entries $E_{p,q}$, which are sums of letters of A, are considered as expressions and denoted as such. The system (3.3) may be solved by successive *elimination* of the unknowns, by means of Arden's lemma.

When all unknowns L_q have been eliminated in the ordering ω on Q, the computation yields an expression that we denote by $\mathbf{E}_\omega(\mathcal{A})$ and $|\mathcal{A}| = |\mathbf{E}_\omega(\mathcal{A})|$ holds. As for the state elimination method, the identities $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{I}$ are likely to have been involved at any step of the computation of $\mathbf{E}_\omega(\mathcal{A})$.

Comparison with the state elimination method The state elimination method and the system resolution are indeed one and the same algorithm for computing the language accepted by a finite automaton, as stated by the following.

Proposition 3.6 ([52]). *For any order ω on the states of $\mathcal{A}$, $\mathbf{B}_\omega(\mathcal{A}) = \mathbf{E}_\omega(\mathcal{A})$ holds.*

The state elimination method reproduces, in the automaton $\mathcal{A}$, the computations corresponding to the resolution of the system: the latter is a *formal proof* of the former. As another consequence of Proposition 3.6, the following corollary of Theorem 3.5 holds:

Corollary 3.7. *Let ω and ω' be two orders on the set of states of an automaton $\mathcal{A}$. Then,*

$$\mathbf{N} \wedge \mathbf{I} \wedge \mathbf{S} \wedge \mathbf{P} \quad \vdash \quad \mathbf{E}_\omega(\mathcal{A}) \quad \equiv \quad \mathbf{E}_{\omega'}(\mathcal{A}) \ .$$

3.4 The McNaughton–Yamada algorithm

Given $\mathcal{A} = \langle Q, A, E, I, T \rangle$, the McNaughton–Yamada algorithm ([42]) — called here MN-Y algorithm for short — truly computes E^*, whereas the two preceding methods rather compute $|\mathcal{A}|$ directly. Like the former methods, it relies on an ordering of Q but it is based on a different classification of computations within $\mathcal{A}$.

Description of the algorithm The set Q ordered by ω is identified with the set of integers from 1 to $n = \mathsf{Card}(Q)$. The central idea of the algorithm is to classify the computations between any states p and q in Q according to the *highest rank* of the intermediate states. We denote by $M_{p,q}^{(k)}$ the set of labels of computations from p to q which *do not pass through intermediate states of rank greater than k*. And we shall compute expressions $\mathsf{M}_{p,q}^{(k)}$ such that $|\mathsf{M}_{p,q}^{(k)}| = M_{p,q}^{(k)}$.

A computation that does not pass through any intermediate state of rank greater than 0 reduces to a single transition. Thus $M_{p,q}^{(0)} = E_{p,q}$ and $\mathsf{M}_{p,q}^{(0)} = \mathsf{E}_{p,q}$. A computation which goes from p to q without visiting intermediate states of rank greater than k is:

(a) either a computation (from p to q) which does not visit intermediate states of rank greater than $k-1$;

(b) or the concatenation:

- of a computation from p to k without passing through states of rank greater than $k-1$;
- followed by an arbitrary number of computations which go from k to k without passing through intermediate states of rank greater than $k-1$;
- followed finally by a computation from k to q without passing through intermediate states of rank greater than $k-1$.

This decomposition implies that for all p and q in Q, for all $k \leqslant n$, it holds:

$$\mathsf{M}_{p,q}^{(k)} = \mathsf{M}_{p,q}^{(k-1)} + \mathsf{M}_{p,k}^{(k-1)} \left(\mathsf{M}_{k,k}^{(k-1)}\right)^* \mathsf{M}_{k,q}^{(k-1)} \ .$$

We write $M_{p,q}$ for $(E^*)_{p,q}$, $\mathsf{M}_{p,q}$ for an expression that denotes it. The algorithm ends with the last equation:

$$\mathsf{M}_{p,q} = \mathsf{M}^{(n)}_{p,q} \quad \text{if } p \neq q \ , \qquad \mathsf{M}_{p,q} = \mathsf{M}^{(n)}_{p,q} + 1 \quad \text{if } p = q \ .$$

For consistency with the previous sections, we write $\mathbf{M}_\omega(\mathcal{A}) = \sum_{p\in I, q\in T} \mathsf{M}_{p,q}$ and $|\mathcal{A}| = |\mathbf{M}_\omega(\mathcal{A})|$ holds.

Example 3.1. The MN-Y algorithm applied to the automaton $\mathcal{R}_1$ of Figure 4 yields the following matrices (we group together, for each k, the four $\mathsf{M}^{(k)}_{p,q}$ into a matrix $\mathsf{M}^{(k)}$):

$$\mathsf{M}^{(0)} = \begin{pmatrix} a & b \\ a & b \end{pmatrix} , \qquad \mathsf{M}^{(1)} = \begin{pmatrix} a + a(a)^*a & b + a(a)^*b \\ a + a(a)^*a & b + a(a)^*b \end{pmatrix} ,$$

$$\mathsf{M}^{(2)} = \begin{pmatrix} a + a(a)^*a + (b + a(a)^*b)(b + a(a)^*b)^*(a + a(a)^*a) & (b + a(a)^*b) + (b + a(a)^*b)(b + a(a)^*b)^*(b + a(a)^*b) \\ a + a(a)^*a + (b + a(a)^*b)(b + a(a)^*b)^*(a + a(a)^*a) & (b + a(a)^*b) + (b + a(a)^*b)(b + a(a)^*b)^*(b + a(a)^*b) \end{pmatrix} .$$

As in the first two methods, identities $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{I}$ are likely to be used at any step of the MN-Y algorithm. Moreover, identities $\mathbf{D}$ and $\mathbf{U}$ are particularly fitted for the computations involved in the MN-Y algorithm. For instance, after using these identities, the above matrices become:

$$\mathsf{M}^{(1)} = \begin{pmatrix} a^*a & a^*b \\ a^*a & a^*b \end{pmatrix} \qquad \text{and} \qquad \mathsf{M}^{(2)} = \begin{pmatrix} (a^*b)^*a^*a & (a^*b)^*a^*b \\ (a^*b)^*a^*a & (a^*b)^*a^*b \end{pmatrix} .$$

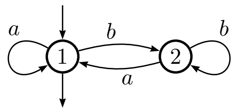


Figure 4. The automaton $\mathcal{R}_1$

Comparison with the state elimination method Comparing the MN-Y algorithm with the state elimination method amounts to relating two objects whose form and mode of construction are rather different: on one hand-side, a $Q{\times}Q$-matrix obtained by successive transformations and on the other, an expression obtained by repeated modifications of an automaton, hence of a matrix, but one whose size decreases at each step.

Proposition 3.8 ([52]). *Let* $\mathcal{A} = \langle\, Q, A, E, I, T\,\rangle$ *and* $\mathcal{A}_{p,q} = \langle\, Q, A, E, \{p\}, \{q\}\,\rangle$ *for every* p *and* q *in* Q. *For every order* ω *on* Q, $\mathbf{N} \wedge \mathbf{U} \vdash \mathbf{M}_\omega(\mathcal{A}_{p,q}) \equiv \mathbf{B}_\omega(\mathcal{A}_{p,q})$ *holds.*

As a consequence of Proposition 3.8, we have the following corollary of Theorem 3.5:

Corollary 3.9. *Let* ω *and* ω' *be two orders on the states of an automaton* $\mathcal{A}$. *Then,*

$$\mathbf{N} \wedge \mathbf{I} \wedge \mathbf{S} \wedge \mathbf{P} \quad \vdash \quad \mathbf{M}_\omega(\mathcal{A}) \quad \equiv \quad \mathbf{M}_{\omega'}(\mathcal{A}) \ .$$

3.5 The recursive algorithm

This last method is due to Conway (in [17]). It is based on computation on matrices via *bloc decomposition*. Originally, it yields a proof of Proposition 3.1. As we did above, we modify it so as to make it compute from E, a matrix of rational expressions which denotes E, a matrix E' of rational expressions which denotes the matrix E^*.

Description of the algorithm Let us write a block decomposition of E and the corresponding ones for E and E^*:

$$E = \begin{pmatrix} F & G \\ H & K \end{pmatrix} , \qquad \mathsf{E} = \begin{pmatrix} \mathsf{F} & \mathsf{G} \\ \mathsf{H} & \mathsf{K} \end{pmatrix} , \qquad E^* = \begin{pmatrix} U & V \\ W & Z \end{pmatrix} ,$$

where F and K (and thus F, K, U and Z) are *square matrices*. By (3.1)[7], it follows that

$$E^* = \begin{pmatrix} U & V \\ W & Z \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} F & G \\ H & K \end{pmatrix} \begin{pmatrix} U & V \\ W & Z \end{pmatrix} ,$$

an equation which can be decomposed into a system of four others equations:

$$U = 1 + |\mathsf{F}|\, U + |\mathsf{G}|\, W \ , \qquad\qquad Z = 1 + |\mathsf{H}|\, V + |\mathsf{K}|\, Z \ , \qquad (3.4)$$

$$V = |\mathsf{F}|\, V + |\mathsf{G}|\, Z \ , \qquad \text{and} \qquad W = |\mathsf{H}|\, U + |\mathsf{K}|\, W \ . \qquad (3.5)$$

Corollary 3.3 applies to (3.5) and then, after substitution, to (3.4). By the induction hypothesis, obviously fulfilled for matrices of dimension 1, $|\mathsf{F}|^*$ and $|\mathsf{K}|^*$ are denoted by matrices of rational expressions F' and K'. Let us write

$$\mathsf{E}' = \begin{pmatrix} (\mathsf{F} + \mathsf{G}\,\mathsf{K}'\,\mathsf{H})^* & \mathsf{F}'\,\mathsf{G}\,(\mathsf{K} + \mathsf{H}\,\mathsf{F}'\,\mathsf{G})^* \\ \mathsf{K}'\,\mathsf{H}\,(\mathsf{F} + \mathsf{G}\,\mathsf{K}'\,\mathsf{H})^* & (\mathsf{K} + \mathsf{H}\,\mathsf{F}'\,\mathsf{G})^* \end{pmatrix} .$$

and $|\mathsf{E}'| = E^*$ holds. Another application of the induction hypothesis to $|(\mathsf{F} + \mathsf{G}\,\mathsf{K}'\,\mathsf{H})|^*$ and $|(\mathsf{K} + \mathsf{H}\,\mathsf{F}'\,\mathsf{G})|^*$ shows that the entries of E', which we denote by $\mathbf{C}_\tau(\mathcal{A})$, where τ is the recursive division of Q used in the computation, are all in $\mathbb{K}\,\mathsf{RatE}\,M$.

Example 3.2. The recursive method applied to the automaton $\mathcal{R}_1$ of Example 3.1 and Figure 4 obviously — there is no choice for the recursive division — gives:

$$\mathbf{C}_\tau(\mathcal{R}_1) = \begin{pmatrix} (a + b(b)^*a)^* & a^*b(b + a(a)^*b)^* \\ b^*a(a + b(b)^*a)^* & (b + a(a)^*b)^* \end{pmatrix} .$$

Comparison with the state elimination method Both the recursive method and the MN-Y algorithm yield a matrix of expressions. Example 3.2 shows that there is no hope for a global comparison of the two matrices. We state however the following conjecture.

Conjecture 3.10. *Let $\mathcal{A} = \langle\, Q, A, E, I, T\,\rangle$ be an automaton. For every recursive division τ of Q and for every pair (p, q) of states, there exists an ordering ω of Q such that:*

$$\mathbf{N} \wedge \mathbf{I} \wedge \mathbf{U} \quad \vdash \qquad (\mathbf{C}_\tau(\mathcal{A}))_{p,q} \quad \equiv \quad \mathbf{B}_\omega(\mathcal{A}_{p,q}) \ .$$

[7] applied to matrices with entries in $\mathfrak{P}(A^*)$ rather than to elements of $\mathfrak{P}(A^*)$.

More genrally, and as a conclusion of the description of these four methods, one would conjecture that *the rational expressions computed from a same finite automaton are all equivalent modulo the natural identities and* $\mathbf{I}$, $\mathbf{S}$ *and* $\mathbf{P}$. Even if *computed from* is not formal enough, the above developments should make the general idea rather clear.

3.6 Star height and loop complexity

Among the three rational operators $+$, $\cdot$ and $*$, the operator $*$ is the one that 'gives access to the infinite', hence the idea of measuring the complexity of an expression by counting the number of nested uses of this operator, a number called *star height*. On the other hand, it is the *circuits* in a finite automaton that produce an infinite number of computations, 'all the more' that the circuits are more 'intricated'. The intuitive idea of intrication of circuits will be captured by the notion of *loop complexity*. A refinement of Theorem 2.2 relates the loop complexity of an automaton to the star height of an expression that is computed from this automaton, a result which is due originally to Eggan ([20]).

Star height of an expression Let E be an expression over A^*. The *star height* of E, denoted by $\mathsf{h}[\mathsf{E}]$, is defined by $\mathsf{h}[\mathsf{E}] = \mathsf{max}(\mathsf{h}[\mathsf{E}'], \mathsf{h}[\mathsf{E}''])$ if $\mathsf{E} = \mathsf{E}' + \mathsf{E}''$ or $\mathsf{E} = \mathsf{E}' \cdot \mathsf{E}''$, $\mathsf{h}[\mathsf{E}] = 1 + \mathsf{h}[\mathsf{F}]$ if $\mathsf{E} = \mathsf{F}^*$, from $\mathsf{h}[0] = \mathsf{h}[1] = \mathsf{h}[a] = 0$, for every a in A.

Example 3.3. (i) $\mathsf{h}[(a+b)^*] = 1$; $\mathsf{h}[a^*(b\,a^*)^*] = 2$.
(ii) The heights of the three expressions computed for the automaton $\mathcal{D}_3$ at Sec. 3.2 are: $\mathsf{h}[\mathbf{B}_{\omega_1}(\mathcal{D}_3)] = 2$, $\mathsf{h}[\mathbf{B}_{\omega_2}(\mathcal{D}_3)] = 3$, and $\mathsf{h}[\mathbf{B}_{\omega_3}(\mathcal{D}_3)] = 3$.

Two equivalent expressions may then have different star heights, and thus give rise to the *star height problem* (see Notes).

Loop complexity of an automaton We call *loop complexity* of an automaton $\mathcal{A}$ the integer $\mathsf{lc}(\mathcal{A})$ defined inductively by the following equations (where a *ball* is a non-trivial strongly connected component):

$\mathsf{lc}(\mathcal{A}) = 0$ if $\mathcal{A}$ contains no balls (in particular if $\mathcal{A}$ is empty);

$\mathsf{lc}(\mathcal{A}) = \mathsf{max}\,\{\mathsf{lc}(\mathcal{P}) \mid \mathcal{P} \text{ a ball in } \mathcal{A}\}$ if $\mathcal{A}$ is not strongly connected;

$\mathsf{lc}(\mathcal{A}) = 1 + \mathsf{min}\,\{\mathsf{lc}(\mathcal{A} \setminus \{s\}) \mid s \text{ state of } \mathcal{A}\}$ if $\mathcal{A}$ is strongly connected.

Theorem 3.11 (Eggan [20]). *The loop complexity of a trim automaton $\mathcal{A}$ is the minimum of the star height of the expressions computed on $\mathcal{A}$ by the state elimination method.*

This theorem may be proved by establishing a more precise statement which involves a refinement of the loop complexity and which we call the *loop index*.

If ω is an order on the state set of $\mathcal{A}$, we write $\overline{\omega}$ for the *greatest state* according to ω. If $\mathcal{R}$ is a subautomaton of $\mathcal{A}$, we also write ω for the trace of ω over $\mathcal{R}$ and, in such a context, $\overline{\omega}$ for the greatest state of $\mathcal{R}$ according to ω. Then, the *loop index of $\mathcal{A}$ relative to* ω, written $\mathfrak{I}_\omega(\mathcal{A})$, is the integer inductively defined by the following:

- if $\mathcal{A}$ contains no ball, or is empty, then $\mathfrak{I}_\omega(\mathcal{A}) = 0$;

- if $\mathcal{A}$ is not itself a ball, then $\mathfrak{I}_\omega(\mathcal{A}) = \mathsf{max}\left(\{\mathfrak{I}_\omega(\mathcal{P}) \mid \mathcal{P} \text{ ball in } \mathcal{A}\}\right)$;
- if $\mathcal{A}$ is a ball, then $\mathfrak{I}_\omega(\mathcal{A}) = 1 + \mathfrak{I}_\omega(\mathcal{A} \setminus \overline{\omega})$.

The difference with respect to loop complexity is that the state that we remove from a strongly connected automaton (to proceed to the induction) is fixed by the order ω rather than being the result of a minimisation. This definition immediately implies that

$$\mathsf{lc}(\mathcal{A}) = \mathsf{min}\,\{\mathfrak{I}_\omega(\mathcal{A}) \mid \ \omega \text{ is an order on } Q\ \} \ .$$

holds and Theorem 3.11 is then a consequence of the following.

Proposition 3.12 ([38]). *For any ordert ω on the states of $\mathcal{A}$, $\mathfrak{I}_\omega(\mathcal{A}) = \mathsf{h}[\mathbf{B}_\omega(\mathcal{A})]$.*

Theorem 3.11 admits a kind of a converse, stated in the following proposition.

Proposition 3.13. *With every rational expression* E *is associated an automaton which accepts* $|\mathsf{E}|$ *and whose loop complexity is equal to the star height of* E*.*

4 From expressions to automata: the Ψ-maps

The transformation of rational expressions into finite automata establishes Proposition 2.4. It is even more interesting than the transformation in the other way, both from a theoretical point of view and for practical purposes, as there are many questions that cannot be answered directly on expressions but require first their transformation into automata.

Every expression is mapped to, or associated with, several automata, each of them being computed in different ways. We distinguish the objects themselves, that is, the computed automata, which we try to give definitions as intrinsic as possible, from the algorithms that allow to compute them. We essentially present two such automata: the *Glushkov*, or *position*, automaton and that we rather call *the standard automaton* of the expression, and the *derived term automaton*, that was first defined by Antimirov.

The standard automaton may be defined for expressions over *any monoid* whereas the derived term automaton will be defined for expressions over a *free monoid* only. In this section however, we restrict ourselves to expressions over a free monoid. We begin with the presentation of two techniques for transforming an automaton into another one, that will help us in comparing the various automata associated with a given expression.

4.1 Preparation: closure and quotient

Closure Automata have been defined (Sec. 2.2) as graphs labelled by *letters* of an alphabet. It is known that the family of languages accepted by finite automata is not increased if transitions labelled by the empty word — called *spontaneous transitions* — are allowed as well. The *backward closure* of such an automaton $\mathcal{A} = \langle\, Q, A, E, I, T\,\rangle$ is the equivalent automaton $\mathcal{B} = \langle\, Q, A, F, I, U\,\rangle$ with no spontaneous transitions defined by

$$F = \left\{(p,a,r) \,\middle|\, \exists q \in Q \quad p \xrightarrow[\mathcal{A}]{1_{A^*}} q\ , \quad (q,a,r) \in E\right\} \quad \text{and} \quad U = \left\{p \,\middle|\, \exists q \in T \quad p \xrightarrow[\mathcal{A}]{1_{A^*}} q\right\} .$$

It is effectively computable — as the determination of F and U amounts to computing the transitive closure of a finite directed graph.

Morphisms and quotient Automata are structures; a morphism is a map from an automaton into another one which is compatible with this structure.

Definition 4.1. Let $\mathcal{A} = \langle Q, A, E, I, T \rangle$ and $\mathcal{A}' = \langle Q', A, E', I', T' \rangle$ be two automata. A map $\varphi\colon Q \to Q'$ is a *morphism (of automata)* if:
(i) $\varphi(I) \subseteq I'$, (ii) $\varphi(T) \subseteq T'$, (iii) $\forall (p,a,q) \in E \quad (\varphi(p), a, \varphi(q)) \in E'$.

The automaton $\mathcal{A}'$ is *a quotient* of $\mathcal{A}$ if, moreover, φ is *surjective* and:
(iv) $\varphi(I) = I'$, (v) $\varphi^{-1}(T') = T$,
(vi) $\forall (r,a,s) \in E'$, $\forall p \in \varphi^{-1}(r)$, $\exists q \in \varphi^{-1}(s) \quad (p,a,q) \in E$.

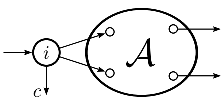
If φ is such a morphism, we write $\varphi\colon \mathcal{A} \to \mathcal{A}'$, and the inclusion $|\mathcal{A}| \subseteq |\mathcal{A}'|$ holds. If $\mathcal{A}'$ is a quotient of $\mathcal{A}$, then $|\mathcal{A}| = |\mathcal{A}'|$ holds.

Definition 4.1 generalises to arbitrary automata the classical notion of quotient for complete deterministic automata. Every automaton $\mathcal{A}$ admits a *minimal quotient*, which is a quotient of every quotient of $\mathcal{A}$. In contrast with the former case, the minimal quotient of $\mathcal{A}$ is canonically associated with $\mathcal{A}$, not with the language accepted by $\mathcal{A}$.

4.2 The standard automaton of an expression

The first automaton we associate with an expression E, which we write $\mathcal{S}_{\mathsf{E}}$ and which will play a central role in our presentation, has been first[8] defined by Glushkov (in [28]). In order to give an intrinsic description of $\mathcal{S}_{\mathsf{E}}$, we define a restricted class of automata, and then show that rational operations on sets can be lifted on the automata that accept them.

4.2.1 Operations on standard automata An automaton is *standard* if it has only one initial state, which is the end of no transition. Figure 5 shows a standard automaton, both as a sketch, and under the matrix form. The definition does not forbid the initial state i from also being final and the scalar c, equal to 0 or 1, is the *constant term* of $|\mathcal{A}|$.



$$\mathcal{A} = \left\langle \begin{pmatrix} 1 & \boxed{\;0\;} \end{pmatrix}, \begin{pmatrix} 0 & \boxed{\;J\;} \\ \boxed{0} & \boxed{\;F\;} \end{pmatrix}, \begin{pmatrix} c \\ \boxed{U} \end{pmatrix} \right\rangle .$$

Figure 5. A standard automaton

Every automaton is equivalent to a standard one. Their special form allows to define *operations* on standard automata that are parallel to the *rational operations*. Let $\mathcal{A}$ (as in Figure 5) and $\mathcal{B}$ (with obvious notation) be two standard automata; the following

[8] Independently and even earlier, McNaughton and Yamada computed directly the *determinisation* of $\mathcal{S}_{\mathsf{E}}$ in the paper [42] that we already quoted (see Notes).

standard automata are defined:

$$\mathcal{A}+\mathcal{B} = \left\langle \begin{pmatrix} 1 & \boxed{0} & \boxed{0} \end{pmatrix}, \begin{pmatrix} 0 & \boxed{J} & \boxed{K} \\ \boxed{0} & \boxed{F} & \boxed{0} \\ \boxed{0} & \boxed{0} & \boxed{G} \end{pmatrix}, \begin{pmatrix} c+d \\ \boxed{U} \\ \boxed{V} \end{pmatrix} \right\rangle , \tag{4.1}$$

$$\mathcal{A}\cdot\mathcal{B} = \left\langle \begin{pmatrix} 1 & \boxed{0} & \boxed{0} \end{pmatrix}, \begin{pmatrix} 0 & \boxed{J} & \boxed{cK} \\ \boxed{0} & \boxed{F} & \boxed{U\cdot K} \\ \boxed{0} & \boxed{0} & \boxed{G} \end{pmatrix}, \begin{pmatrix} cd \\ \boxed{Ud} \\ \boxed{V} \end{pmatrix} \right\rangle , \tag{4.2}$$

$$\mathcal{A}^* = \left\langle \begin{pmatrix} 1 & \boxed{0} \end{pmatrix}, \begin{pmatrix} 0 & \boxed{J} \\ \boxed{0} & \boxed{H} \end{pmatrix}, \begin{pmatrix} 1 \\ \boxed{U} \end{pmatrix} \right\rangle , \tag{4.3}$$

where $H = U \cdot J + F$. The use of the constants c and d allows to treat uniformly the cases when the initial states of $\mathcal{A}$ and $\mathcal{B}$ are final and when they are not. Straightforward computations show that $|(\mathcal{A}+\mathcal{B})| = |\mathcal{A}|+|\mathcal{B}|$, $|(\mathcal{A}\cdot\mathcal{B})| = |\mathcal{A}|\cdot|\mathcal{B}|$ and $|(\mathcal{A}^*)| = |\mathcal{A}|^*$.

With every rational expression E and by induction on its depth, we thus canonically associate a standard automaton, which we write $\mathcal{S}_{\mathsf{E}}$ and which we call *the* standard automaton of E. The same induction also shows that the *dimension* of $\mathcal{S}_{\mathsf{E}}$ is $\ell(\mathsf{E})+1$ and that $\mathcal{S}_{\mathsf{E}}$ answers the question, that is, the map $\mathsf{E} \mapsto \mathcal{S}_{\mathsf{E}}$ is a Ψ-map:

Proposition 4.1. *If* E *is a rational expression over* A^**, then* $|\mathcal{S}_{\mathsf{E}}| = |\mathsf{E}|$.

Example 4.1. Figure 6 shows $\mathcal{S}_{\mathsf{E}_1}$, where $\mathsf{E}_1 = (a^*b + b\,b^*a^*)^*$.

The example of $\mathsf{E} = \left(\left((a^*+b^*)^* + c^*\right)^* + d^*\right)^* \dots$ shows that the direct computation of $\mathcal{S}_{\mathsf{E}}$ by (4.1)–(4.3) leads to an algorithm whose complexity is *cubic* in $\ell(\mathsf{E})$. The quest for a better algorithm lead to a construction that is interesting *per se*.

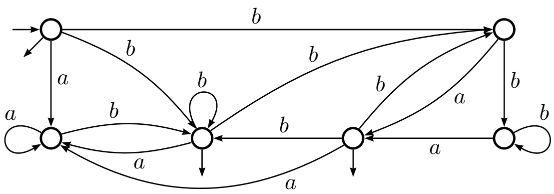


Figure 6. The automaton $\mathcal{S}_{\mathsf{E}_1}$.

4.2.2 The star normal form of an expression The *star normal form* of an expression has been defined by Brüggemann-Klein (in [10]) in order to design a quadratic algorithm for the computation of $\mathcal{S}_{\mathsf{E}}$. The interest of this notion certainly goes further.

Definition 4.2 ([10]). A rational expression E is in star normal form (SNF) if, and only if, for any F such that F^* is a subexpression of E, $\mathsf{c}(\mathsf{F}) = 0$.[9]

Two operators on expressions: $\bullet$ and $\square$, are defined by an intertwinned induction on the depth of the expressions in order to compute (and define by this computation) *the* star normal form of the expression.

$$0^{\square} = 0^{\bullet} = 0\,, \quad 1^{\square} = 1^{\bullet} = 0\,, \qquad \forall a \in A \quad a^{\square} = a^{\bullet} = a \ ,$$
$$(\mathsf{F}+\mathsf{G})^{\square} = \mathsf{F}^{\square} + \mathsf{G}^{\square}\,, \quad (\mathsf{F}^*)^{\square} = \mathsf{F}^{\square}\,, \quad (\mathsf{F}\cdot\mathsf{G})^{\square} = \begin{cases} \mathsf{F}^{\square} + \mathsf{G}^{\square} & \text{if} \quad \mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 1 \ , \\ \mathsf{F}^{\bullet}\cdot\mathsf{G}^{\bullet} & \text{otherwise} \ , \end{cases}$$
$$(\mathsf{F}+\mathsf{G})^{\bullet} = \mathsf{F}^{\bullet} + \mathsf{G}^{\bullet}\,, \quad (\mathsf{F}\cdot\mathsf{G})^{\bullet} = \mathsf{F}^{\bullet}\cdot\mathsf{G}^{\bullet}\,, \quad (\mathsf{F}^*)^{\bullet} = (\mathsf{F}^{\square})^* \ .$$

Example 4.2. If $\mathsf{E}_2 = (a^*b^*)^*$, then ${\mathsf{E}_2}^{\bullet} = (a+b)^*$.

Theorem 4.2 ([10]). *For any expression* E, $\mathsf{E}^{\bullet}$ *is in star normal form and* $\mathcal{S}_{\mathsf{E}^{\bullet}} = \mathcal{S}_{\mathsf{E}}$.

As the computation of $\mathsf{E}^{\bullet}$ is linear in $\ell(\mathsf{E})$, the goal is achieved by the following:

Theorem 4.3 ([10]). *The computation of* $\mathcal{S}_{\mathsf{E}^{\bullet}}$ *has a quadratic complexity in* $\ell(\mathsf{E})$.

4.2.3 The Thompson automaton A survey on Ψ-maps cannot miss out the method due to Thompson [56]. It was designed to be directly implementable as a program, primarily for searching rational expressions in text. It is based on the use of spontaneous transitions. Figure 7 shows the basic steps of the construction, which, by induction, associates with an expression E a unique (and well-defined) automaton $\mathcal{T}_{\mathsf{E}}$. This construction corresponds indeed to another way of defining the standard automaton:

Proposition 4.4. *The* backward closure *of* $\mathcal{T}_{\mathsf{E}}$ *is equal to* $\mathcal{S}_{\mathsf{E}}$.

4.3 The derived term automaton of an expression

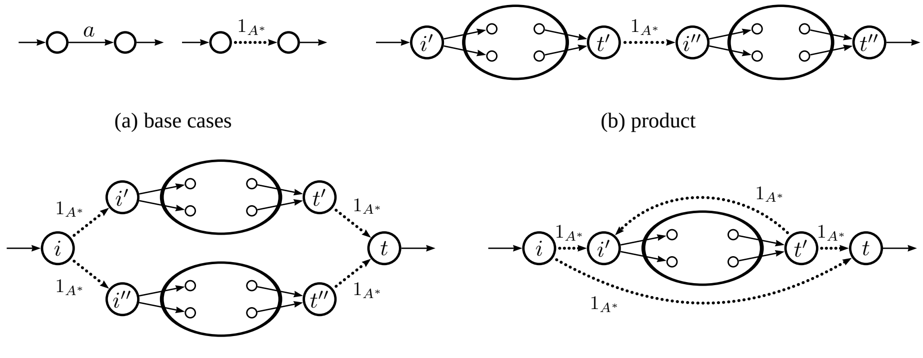
Let us first recall the *(left) quotient* operation on languages:

$$\forall L \in \mathfrak{P}(A^*)\,,\ \forall u \in A^* \qquad u^{-1}L = \{v \in A^* \mid u\,v \in L\} \ .$$

The quotient is a *(right) action* of A^* on $\mathfrak{P}(A^*)$: $(u\,v)^{-1}L = v^{-1}\left(u^{-1}L\right)$. A fundamental, and characteristic, property of rational *languages* — which is another way to express that they are recognisable — is that they have a *finite number of quotients*.

The principle of the construction we present in this section, and which we call *derivation*, is to transfer the quotient on languages to an operation on the expressions. First introduced by Brzozowski [11], the definition of the derivation of an expression E has been modified by Antimirov [4] (*cf.* Notes) and yields a non deterministic automaton $\mathcal{A}_{\mathsf{E}}$, which we propose to call the *derived term automaton* of E. This construction concerns thus expressions over *free monoids* only. In the sequel, E is a rational expression over A^*.

[9] The definition, as well as the construction, have been slightly modified from the original, for simplification.



(a) base cases (b) product

(c) union (d) star

Figure 7. Thompson's construction

Definition 4.3 (Brzozowski–Antimirov [4]). The *derivation* of E with respect to a letter a of A, denoted by $\frac{\partial}{\partial a}\mathsf{E}$, is *a set* of rational expressions over A^*, inductively defined by:

$$\frac{\partial}{\partial a}\mathsf{0} \;=\; \frac{\partial}{\partial a}\mathsf{1} \;=\; \emptyset, \qquad \forall b \in A \quad \frac{\partial}{\partial a} b \;=\; \begin{cases} \{1\} & \text{if } \; b = a\,, \\ \emptyset & \text{otherwise,} \end{cases} \quad , \tag{4.4}$$

$$\frac{\partial}{\partial a}(\mathsf{F}+\mathsf{G}) \;=\; \frac{\partial}{\partial a}\mathsf{F} \cup \frac{\partial}{\partial a}\mathsf{G}\ , \tag{4.5}$$

$$\frac{\partial}{\partial a}(\mathsf{F}\cdot\mathsf{G}) \;=\; \left(\frac{\partial}{\partial a}\mathsf{F}\right)\cdot\mathsf{G} \cup \mathsf{c}(\mathsf{F})\ \frac{\partial}{\partial a}\mathsf{G}\ , \tag{4.6}$$

$$\frac{\partial}{\partial a}(\mathsf{F}^*) \;=\; \left(\frac{\partial}{\partial a}\mathsf{F}\right)\cdot\mathsf{F}^*\ . \tag{4.7}$$

Equation (4.6) should be understood with the convention that the product xX of a set X by a Boolean value x is X if $x = 1$ and $\emptyset$ if $x = 0$. The induction involved in Equations (4.5)–(4.7) should be interpreted by extending derivation additively (as are always derivation operators) and by distributing (on the right) the $\cdot$ operator over sets as well. Finally, every operation on rational expressions is computed modulo the trivial identities ($\mathbf{T}$), but not modulo the other natural identies ($\mathbf{N}$).

Definition 4.4. The *derivation* of E with respect to a non-empty word v of A^*, denoted by $\frac{\partial}{\partial v}\mathsf{E}$, is *the set of rational expressions over* A^*, defined by (4.5)–(4.7) for letters in A and by induction on the length of v by:

$$\forall u \in A^+\,,\ \forall a \in A \qquad \frac{\partial}{\partial ua}\mathsf{E} = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{E}\right)\ . \tag{4.8}$$

The derivation of expressions is parallel to the quotient of languages and we have:

$$\forall \mathsf{E} \in \mathsf{RatE}\,A^*\,,\ \forall u \in A^+ \qquad \left|\frac{\partial}{\partial u}\mathsf{E}\right| = u^{-1}|\mathsf{E}|\ . \tag{4.9}$$

Example 4.3. The derivation of $\mathsf{E}_1 = (a^*b + b\,b^*a)^*$ (*cf.* Example 4.1) yields:

$$\frac{\partial}{\partial a}\,\mathsf{E}_1 = \frac{\partial}{\partial aa}\,\mathsf{E}_1 = \{a^*b\,\mathsf{E}_1\}\,, \quad \frac{\partial}{\partial b}(\mathsf{E}_1)^* = \{\mathsf{E}_1, b^*a\,\mathsf{E}_1\}\,,$$
$$\frac{\partial}{\partial b}\,a^*b\,\mathsf{E}_1 = \{\mathsf{E}_1\}\,, \quad \frac{\partial}{\partial a}(b^*a\,\mathsf{E}_1)^* = \{\mathsf{E}_1\}\,, \quad \frac{\partial}{\partial b}(b^*a\,\mathsf{E}_1)^* = \{b^*a\,\mathsf{E}_1\}\,.$$

4.3.1 The derived term automaton Derivation thus associates a pair of an expression and a word with a set of expressions. We now turn this map into an automaton.

Definition 4.5. We call *true derived term* of E every expression that belongs to $\frac{\partial}{\partial w}\,\mathsf{E}$ for some word w of A^+; we write $\mathrm{TD}\,(\mathsf{E})$ for the set of true derived terms of E:

$$\mathrm{TD}\,(\mathsf{E}) = \bigcup_{w \in A^+} \frac{\partial}{\partial w}\,\mathsf{E}\ . \tag{4.10}$$

The set $\mathrm{D}\,(\mathsf{E}) = \mathrm{TD}\,(\mathsf{E}) \cup \{\mathsf{E}\}$ is the set of *derived terms* of E.

Example 4.4 (Example 4.3 cont.). $\mathrm{D}\,(\mathsf{E}_1) = \{\mathsf{E}_1, a^*b\,\mathsf{E}_1, b^*a\,\mathsf{E}_1\}\,.$

The sets of derived terms and the rational operations are related by the following equations, from which most of the subsequent properties will be derived.

Proposition 4.5. *Let* F *and* G *be two expressions. Then,* $\mathrm{TD}\,(\mathsf{F}+\mathsf{G}) = \mathrm{TD}\,(\mathsf{F}) \cup \mathrm{TD}\,(\mathsf{G})$, $\mathrm{TD}\,(\mathsf{F}\cdot\mathsf{G}) = (\mathrm{TD}\,(\mathsf{F}))\cdot\mathsf{G} \cup \mathrm{TD}\,(\mathsf{G})$, *and* $\mathrm{TD}\,(\mathsf{F}^*) = (\mathrm{TD}\,(\mathsf{F}))\cdot\mathsf{F}^*$ *hold.*

Starting from $\mathrm{TD}\,(0) = \mathrm{TD}\,(1) = \emptyset$ and $\mathrm{TD}\,(a) = \{1\}$ for every a in A, $\mathrm{TD}\,(\mathsf{E})$ *can be computed* from Proposition 4.5 by induction on $\mathsf{d}(\mathsf{E})$ and *without reference to the derivation operation* (*cf.* the *prebases* in [43] and Definition 6.2 below). It follows in particular that $\mathsf{Card}\,(\mathrm{TD}\,(\mathsf{E})) \leqslant \ell\,(\mathsf{E})$ and thus:

Corollary 4.6. $\mathsf{Card}\,(\mathrm{D}\,(\mathsf{E})) \leqslant \ell\,(\mathsf{E}) + 1\,.$

The computation of $\mathrm{D}\,(\mathsf{E})$ is a Ψ-map, as expressed by the following.

Definition 4.6 (Antimirov [4]). The *derived term automaton* of E is the automaton $\mathcal{A}_{\mathsf{E}}$ whose set of states is $\mathrm{D}\,(\mathsf{E})$ and whose transitions are defined by:

(i) if K and K' are derived terms of E and a a letter of A, then $(\mathsf{K}, a, \mathsf{K}')$ is a transition if, and only if, K' belongs to $\frac{\partial}{\partial a}\,\mathsf{K}$;

(ii) the initial state is E;

(iii) a derived term K is final if, and only if, $\mathsf{c}(\mathsf{K}) = 1$.

Theorem 4.7 ([4]). *For any rational expression* E, $|\mathsf{E}| = |\mathcal{A}_{\mathsf{E}}|$.

Example 4.5 (Example 4.4 cont.). The automaton $\mathcal{A}_{\mathsf{E}_1}$ is shown at Figure 8.

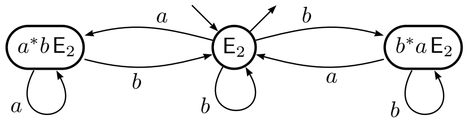


Figure 8. The automaton $\mathcal{A}_{\mathsf{E}_1}$.

4.3.2 Relationship with the standard automaton The constructions of the standard and derived term automata of an expression are of different nature. But both arise from the same inner structure of the expression by two inductive processes, and the two automata have a structural likeness which yields another proof of Corollary 4.6:

Theorem 4.8 ([16]). *For any rational expression* E, $\mathcal{A}_{\mathsf{E}}$ *is a quotient of* $\mathcal{S}_{\mathsf{E}}$.

4.3.3 Derivation and bracketing The derivation operator is sensitive to the bracketing of expressions; on the other hand, it does commute to the associativity identity (**A**).

Example 4.6. Let $a\,b\,(c\,(a\,b))^*$ be an expression which is not completely bracketed. The derivation of the two expressions obtained by different bracketings yields:

$$\mathrm{D}\left(a\,(b\,(c\,(a\,b))^*)\right) = \{a\,(b\,(c\,(a\,b))^*)\,,\; b\,(c\,(a\,b))^*, (c\,(a\,b))^*\,,\; (a\,b)\,(c\,(a\,b))^*\}\;.$$
$$\mathrm{D}\left((a\,b)\,(c\,(a\,b))^*\right) = \{(a\,b)\,(c\,(a\,b))^*\,,\; b\,(c\,(a\,b))^*, (c\,(a\,b))^*\}\;.$$

More precisely, we have the following.

Proposition 4.9 ([3]). *Let* E, F *and* G *be three rational expressions. Then:*

$$\mathsf{Card}\left(\mathrm{D}\left((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G}\right)\right) \leqslant \mathsf{Card}\left(\mathrm{D}\left(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G})\right)\right) \quad \textit{and} \quad \mathbf{A} \vdash \quad \mathrm{D}\left((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G}\right) \equiv \mathrm{D}\left(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G})\right)\;.$$

5 Changing the monoid

Most of what has been presented so far extends without problems from languages to subsets of arbitrary monoids, from expressions over a free monoid to expressions over such monoids. We run over definitions and statements to transform them accordingly. The main difference will be that rational and recognisable sets do not coincide anymore, making the link between finite automata and rational expressions even tighter, and ruling out quotient and derivation that refer to the recognisable 'side' of rational languages.

Non-free monoids of interest in the field of computer science and automata theory are — among others — direct products of free monoids (for relations between words), among which free commutative monoids for counting purpose, partially commutative, or trace, monoids (for modellisation of concurrent or parallel computations), free groups and polycyclic monoids (in relation with pushdown automata).

In the sequel, M is a monoid, and 1_M its identity element.

5.1 Rationality

Rational sets and expressions Product and star are defined in $\mathfrak{P}(M)$ as in $\mathfrak{P}(A^*)$ and the set of *rational subsets* of M, denoted by $\mathrm{Rat}\, M$, is the smallest subset of $\mathfrak{P}(M)$ which contains the finite sets and which is closed under union, product, and star.

Rational expressions over M are defined as those over A^*, with the only difference that the *atoms* are the elements of M; their set is denoted by $\mathsf{RatE}\, M$. We also write $|\mathsf{E}|$ for the subset *denoted* by an expression E. Two expressions are equivalent if they denote the same subset and we have the same statement as Proposition 2.1:

Proposition 5.1. *A subset of* M *is rational if, and only if, it is denoted by a rational expression over* M.

A subset G of M is a *generating set* if $G^* = M$. The direct part of Proposition 5.1 may be restated with more precision as: *any rational subset of* M *is denoted by a rational expression whose atoms are taken in any generating set.* It follows from the converse part that a rational subset of M is contained in a finitely generated submonoid.

Finite automata An *automaton over* M, denoted by $\mathcal{A} = \langle Q, M, E, I, T \rangle$, is defined like an automaton over A^*, with the only difference that the transitions are labelled by elements of M: $E \subseteq Q \times M \times Q$. Then, $\mathcal{A}$ is *finite* if E is finite.

The *subset accepted* by $\mathcal{A}$, called the *behaviour* of $\mathcal{A}$ and denoted by $|\mathcal{A}|$ as above, is the set of labels of *successful computations*: $|\mathcal{A}| = \left\{ m \in M \;\middle|\; \exists i \in I, \exists t \in T \quad i \xrightarrow[\mathcal{A}]{m} t \right\}$.

The fundamental theorem of finite automata In this setting, the statement appears more clearly different from Kleene's theorem. Its first appearance [10] seems to be in Elgot and Mezei's paper on rational relations.

Theorem 5.2 ([22]). *A subset of a monoid* M *is rational if, and only if, it is the behaviour of a finite automaton over* M *whose labels are taken in any generating set of* M.

There is not much to change in Propositions 2.3 and 2.4 to establish Theorem 5.2.

Proposition 5.3 (Φ-maps). *For every finite automaton* $\mathcal{A}$ *over* M, *there exist rational expressions over* M *which denote* $|\mathcal{A}|$.

All four methods described in Sec. 3 apply for arbitrary M, even if their formal proof may be slightly different (Arden's lemma does not hold anymore).

Proposition 5.4 (Ψ-maps). *For every rational expression* E *over* M, *there exist finite automata over* M *whose behaviour is equal to* $|\mathsf{E}|$.

Here again, the algorithms and results described in Sec. 4.2 for the construction of the standard automaton, Thompson automaton, *etc.* pass over to expressions over M. On the contrary, quotients in M define recognisable subsets of M and not rational ones (see below) and derivation of expressions over M does not make sense anymore.

[10] Hidden in a footnote!

5.2 Recognisability

Definition 2.3 may be rephrased *verbatim* for arbitrary monoids. *A subset P of M is said to be* recognised *by a morphism $\alpha\colon M \rightarrow N$ if $P = \alpha^{-1}(\alpha(P))$. A subset of M is* recognisable *if it is recognised by a morphism from M into a* finite *monoid. The set of* recognisable subsets *of M is denoted by* $\operatorname{Rec} M$.

Recognisable and rational subsets We can then reproduce almost *verbatim* the converse part of the proof of Theorem 2.5. Let P be in $\operatorname{Rec} M$, recognised by a morphism α. We replace the alphabet A by any generating set G of M in the construction of the automaton $\mathcal{A}_\alpha$. If M is finitely generated, G is finite, so is $\mathcal{A}_\alpha$ and P is rational by Theorem 5.2:

Proposition 5.5 (McKnight [41]). *If M is finitely generated, then* $\operatorname{Rec} M \subseteq \operatorname{Rat} M$.

On the other hand, the first part of the quoted proof does not generalise to non-free monoids and the inclusion in Proposition 5.5 is strict in general. For instance, the set $(a,c)^* = \big((a,1)(1,c)\big)^*$ is a rational subset of $a^* \times c^*$. It is accepted by a two state automaton which induces a map μ from the generating set of $a^* \times c^*$ into $\mathbb{B}^{2\times 2}$:

$$\mu\big((a,1)\big) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad \text{and} \qquad \mu\big((1,c)\big) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} .$$

But this map does not define a *morphism* from $a^* \times c^*$ into $\mathbb{B}^{2\times 2}$.

Decision problems for rational sets In general, $\operatorname{Rat} M$ is not a Boolean algebra. This is also accompanied with undecidability results. The undecidability of Post Correspondence Problem, easily expressed in terms of monoid morphisms implies for instance:

Theorem 5.6 (Rabin–Scott [46]). *It is undecidable whether the intersection of two rational sets of $\{a,b\}^* \times \{c,d\}^*$ is empty or not.*

From which one deduce:

Theorem 5.7 (Fischer–Rosenberg [23]). *The equivalence of finite automata, and hence of rational expressions, over $\{a,b\}^* \times \{c,d\}^*$ is undecidable.*

In contrast, the cases where $\operatorname{Rat} M$ is an effective Boolean algebra — such as when M is a (finitely generated) *free commutative monoid* [27] or *free group* [25] — play a key role in model checking issues which involve counters or pushdown automata.

6 Introducing weights

Most of the statements about automata and expressions established in the previous sections extend again without much difficulties in the *weighted case*, as we have taken care to formulate them adequately. There are two questions though that should be settled first in order to set up the framework of this generalisation. First, the definition of the *star operator* requires some mathematical apparatus to be meaningful. Second, the definition of *weighted expressions* has to be tuned in such a way former computations such as the *derivation* remain valid.[11]

6.1 Weighted languages, automata, and expressions

6.1.1 The series semiring The *weights*, with which we enrich the languages or subsets of monoids are taken in a *semiring*, so as to give the set of *series* we build the desired structure. We are interested in weights as they actually appear in the modelisation of phenomena that we want to be able to describe (and not because they fullfil some axioms). These are the classical numerical semirings $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, *etc.*, the less classical $\langle\, \mathbb{Z} \cup +\infty, \mathsf{min}, + \,\rangle$, *etc.* None of them are *Conway semirings* (*cf.* Chap. XX), $\mathbb{N}$ is a *quasi-Conway semiring* but not the others. In the sequel, $\mathbb{K}$ is a semiring. The unweighted case corresponds to $\mathbb{K} = \mathbb{B}$ and will be refer to as *the Boolean case*.

As in the Boolean case, free monoids give rise to results which do not hold in non-free ones (the Kleene–Schützenberger theorem). But not all non-free monoids allow to easily define series with weights in arbitrary semirings. We restrain ourselves to *graded monoids*, that is, which are equipped with a *length function*. They behave exactly like the free monoids as far as the construction of series is concerned, they cover many monoids that are considered in computer science, and they are sufficient to make clear the difference between the free and non-free cases as far as rationality is concerned. In the sequel, M is a finitely generated graded monoid.

Series Any *map* s from M to $\mathbb{K}$ is a *formal power series* (*series* for short) over M with coefficients in $\mathbb{K}$. The image by s of an element m in M is written $\langle s, m \rangle$ and is called the *coefficient of* m *in* s. The set of these series, written $\mathbb{K}\langle\langle M \rangle\rangle$, is equipped with the (left and right) *'exterior' multiplications*, the pointwise *addition*, and the *(Cauchy) product*: for every m in M, $\langle st, m \rangle = \sum_{uv=m} \langle s, u \rangle \langle t, v \rangle$. As M is graded, the product is well-defined, and the three operations make $\mathbb{K}\langle\langle M \rangle\rangle$ a semiring (*cf.* Chap. IV).

The *support* of a series s is the subset of elements of M whose coefficient in s is not $0_{\mathbb{K}}$. A series with finite support is a *polynomial*; the set of polynomials over M with coefficients in $\mathbb{K}$ is written $\mathbb{K}\langle M \rangle$.

Topology The definition to come of the star as an *infinite sum* calls for the definition of a *topology* on $\mathbb{K}\langle\langle M \rangle\rangle$. The semirings $\mathbb{K}$ we consider are equipped with a topology defined by a *distance*, whether it is a *discrete topology* ($\mathbb{N}$, $\mathbb{Z}$, $\langle\, \mathbb{Z} \cup +\infty, \mathsf{min}, + \,\rangle$, *etc.*) or a more classical one ($\mathbb{Q}$, $\mathbb{R}$, another $\mathbb{L}\langle\langle N \rangle\rangle$, *etc.*). Since M is graded (and finitely generated) it is

[11] The definition of the *behaviour of weighted automata* also conceals a problem due to the existence of *spontaneous* or ε-transitions). This is out of the scope of this chapter where we focus on the relationships between automata and expressions. All usual definitions eventually allow to establish that every automaton whose behaviour is defined is equivalent to a *proper* automaton. This is the definition we take for a weighted automaton and where we begin our presentation. We thus save a significant amount of foundation results. On this subject, we refer to other chapters of this handbook (Chapters IV and XX) and other works ([55, 8, 37, 52, 19]).

easy to derive a distance which defines on $\mathbb{K}\langle\langle M\rangle\rangle$ the *simple convergence topology*:

s_n *converges to* s if, and only if, for all m in M, $\langle s_n, m\rangle$ *converges to* $\langle s, m\rangle$.

Along the same line, a family of series $\{s_i\}_{i\in I}$ is *summable* if for every m in M the family $\{\langle s_i, m\rangle\}_{i\in I}$ is summable (in $\mathbb{K}$). An obvious case of summability is when for every m in M there is only a finite number of indices i such that $\langle s_i, m\rangle$ is different from $0_{\mathbb{K}}$, in which case the family $\{s_i\}_{i\in I}$ is said to be *locally finite*.

All quoted semirings that we consider are *topological semirings*, that is, not only equipped with a topology, but their semiring operations are *continuous*. We also use silently in the sequel the following identification: if Q is a finite set, $\mathbb{K}\langle\langle M\rangle\rangle^{Q\times Q}$, the semiring of $Q\times Q$-matrices with entries in $\mathbb{K}\langle\langle M\rangle\rangle$ *is isomorphic* to $\mathbb{K}^{Q\times Q}\langle\langle M\rangle\rangle$, the semiring of series on M with coefficients in $\mathbb{K}^{Q\times Q}$.

Star The star, denoted t^*, of an element t in an arbitrary topological semiring $\mathbb{T}$ (not only in a semiring of series) is defined if the family $\{t^n\}_{n\in\mathbb{N}}$ is summable and in this case, $t^* = \sum_{n\in\mathbb{N}} t^n$. If t^* is defined, then $t^* = 1_{\mathbb{T}} + t\,t^* = 1_{\mathbb{T}} + t^*t$ hold. If moreover $\mathbb{T}$ is a *ring*, this can be written $(1-t)\,t^* = t^*(1-t) = 1$ and t^* is the *inverse* of $1-t$. Coming back to general semirings, it means that *computing the star can be considered as a substitute of taking the inverse* in poor structure that has no inverse. Hence the name of *rational* given to objects that can be computed with the star.

The *constant term* of a series s is the coefficient of the identity of M: $\mathsf{c}(s) = \langle s, 1_M\rangle$. A series is *proper* if its constant term is zero. If s is proper, the family $\{s^n\}_{n\in\mathbb{N}}$ is locally finite *since* M *is graded* and the star of a proper series of $\mathbb{K}\langle\langle M\rangle\rangle$ is thus always defined.

Lemma 6.1. *Let s and t be two series in $\mathbb{K}\langle\langle M\rangle\rangle$. If s^* is defined, then s^*t is the* unique *solution of the equation* $\mathrm{X} = s\,\mathrm{X} + s$.

6.1.2 Rational series and expressions The *rational operations* on $\mathbb{K}\langle\langle M\rangle\rangle$ are: the two *exterior multiplications* by elements of $\mathbb{K}$, the *addition*, the *product*, and the *star* which is not defined everywhere. A subset $\mathcal{E}$ of $\mathbb{K}\langle\langle M\rangle\rangle$ is *closed under star* if for every s in $\mathcal{E}$ such that s^* is defined then s^* belongs to $\mathcal{E}$. The *rational closure* of a set $\mathcal{E}$, written $\mathbb{K}\mathrm{Rat}\,\mathcal{E}$, is the *smallest* subset of $\mathbb{K}\langle\langle M\rangle\rangle$ closed under the rational operations and which contains $\mathcal{E}$. The set of $\mathbb{K}$-*rational series*, written $\mathbb{K}\mathrm{Rat}\,M$, is the rational closure of $\mathbb{K}\langle M\rangle$.

Rational weighted expressions A rational expression on M with weight in $\mathbb{K}$ — a *weighted expression* — is defined by completing Definition 2.1 with *two* operations *for every k in $\mathbb{K}$*: if E is an expression, then so are $(k\,\mathsf{E})$ and $(\mathsf{E}\,k)$. The set of weighted rational expressions is written $\mathbb{K}\,\mathsf{RatE}\,M$. As for the languages, we write $|\mathsf{E}|$ for the *series denoted by* E, with the supplementary equations: $|(k\,\mathsf{E})| = k|\mathsf{E}|$ and $|(\mathsf{E}\,k)| = |\mathsf{E}|k$.

The *constant term* $\mathsf{c}(\mathsf{E})$ is defined as in Definition 2.2 but for the last equation ($\mathsf{c}(\mathsf{F}^*) = 1$) which is replaced by: '$\mathsf{c}(\mathsf{F}^*) = \mathsf{c}(\mathsf{F})^*$ *if the latter is defined*'. An expression is *valid* if its constant term is defined. As M is graded, $\mathsf{c}(\mathsf{E}) = \langle|\mathsf{E}|, 1_M\rangle$ holds for every valid weighted rational expression E. Finally, the following holds:

Proposition 6.2. *A series of $\mathbb{K}\langle\langle M\rangle\rangle$ is $\mathbb{K}$-rational if, and only if, it is denoted by a valid rational $\mathbb{K}$-expression over M.*

In this framework, we reformulate Lemma 6.1 as:

Corollary 6.3. *Let* U *and* V *be two expressions in* $\mathbb{K}\,\mathsf{RatE}\,M$. *If* $\mathsf{c}(\mathsf{U})$ *is starable, then* $\mathsf{U}^*\,\mathsf{V}$ *denotes the unique solution of the equation* $\mathrm{X} = |\mathsf{U}|\mathrm{X} + |\mathsf{V}|$.

6.1.3 Weighted automata and the fundamental theorem An automaton $\mathcal{A}$ over M with weight in $\mathbb{K}$, a $\mathbb{K}$-automaton for short, still written $\mathcal{A} = \langle\, Q, M, E, I, T\,\rangle$, is an automaton where the *sets* of initial and final states are replaced with *maps* from Q to $\mathbb{K}$, that is, every state has an initial and a final weight, and where the set E of transitions is contained in $Q\times\mathbb{K}\times(M \setminus 1_M)\times Q$, that is, every transition is labelled with a *monomial* in $\mathbb{K}\langle M\rangle$, different from a constant term. The automaton $\mathcal{A}$ is *finite* if E is finite.

Alternatively, the same automaton is (more often) written $\mathcal{A} = \langle\, I, E, T\,\rangle$, with the convention taken at Sec. 2: E is the *transition matrix* of $\mathcal{A}$, a $Q\times Q$-matrix whose (p,q)-entry is the sum of the labels of all transitions from p to q, and I and T are vectors in $\mathbb{K}^{\,Q}$. In this setting, $\mathcal{A}$ is finite if every entry of E is a polynomial of $\mathbb{K}\langle M\rangle$.

The *label* of a computation in $\mathcal{A}$ is, as above, the product of the labels of the transitions that form the computation, multiplied (on the left) by the initial weight of the origin and (on the right) by the final weight of the end of the computation. With the definition we have taken for automata (no transition labelled with a constant term), and because M is graded, the family of labels of all transitions of $\mathcal{A}$ is *summable* and the *series accepted* by $\mathcal{A}$, also called *behaviour* of $\mathcal{A}$ and written $|\mathcal{A}|$, is its sum. The fundamental theorem of automata then reads:

Theorem 6.4. *Let* M *be a graded monoid. A series of* $\mathbb{K}\langle\!\langle M\rangle\!\rangle$ *is rational if, and only if, it is the behaviour of a finite* $\mathbb{K}$*-automaton over* M.

6.2 From automata to expressions: the Φ-maps

With the definition taken for a $\mathbb{K}$-automaton $\mathcal{A} = \langle\, I, E, T\,\rangle$, every entry of E is a *proper polynomial* of $\mathbb{K}\langle M\rangle$, E is in $\mathbb{K}\langle M\rangle^{Q\times Q}$, hence a proper polynomial of $\mathbb{K}^{Q\times Q}\langle M\rangle$, and E^* is well-defined. Lemma 2.6 generalises to $\mathbb{K}$-automata and $|\mathcal{A}| = I\cdot E^*\cdot T$ holds.

In every respect, the weighted case is similar to the Boolean one. The direct part of Theorem 6.4 follows from the generalised statement of Proposition 3.1:

Proposition 6.5. *The entries of the star of a proper matrix* E *of* $\mathbb{K}\langle\!\langle M\rangle\!\rangle^{Q\times Q}$ *belong to the rational closure of the entries of* E.

The same algorithms as those presented at Sec. 3: the state elimination and system resolution methods, the McNaughton–Yamada and recursive algorithms, will establish the weighted version of Proposition 2.3:

Proposition 6.6. *Let* M *be a graded monoid. For every finite* $\mathbb{K}$*-automaton* $\mathcal{A}$ *over* M, *there exist rational* $\mathbb{K}$*-expressions over* M *which denote* $|\mathcal{A}|$.

If the algoritms are the same, one has to establish nevertheless their correctness in this new and more complex framework. We develop the case of the system resolution method, the other ones could be treated in the same way. To begin with, we have to enrich the set of *trivial identities* in order to set up the definition of *reduced weighted expressions,* which in turn is necessary to define computations on expressions. The set $\mathbf{T}$ as defined at Sec. 3.1 is now denoted as $\mathbf{T}_{\mathsf{u}}$:

$$\mathsf{E}+0 \equiv \mathsf{E}\,, \quad 0+\mathsf{E} \equiv \mathsf{E}\,, \quad \mathsf{E}\cdot 0 \equiv 0\,, \quad 0\cdot\mathsf{E} \equiv 0\,, \quad \mathsf{E}\cdot 1 \equiv \mathsf{E}\,, \quad 1\cdot\mathsf{E} \equiv \mathsf{E}\,, \quad 0^* \equiv 1 \qquad (\mathbf{T}_{\mathsf{u}})$$

and augmented with three other sets of identities:

$$0_{\mathbb{K}}\,\mathsf{E} \equiv 0\,, \quad \mathsf{E}\,0_{\mathbb{K}} \equiv 0\,, \quad k\,0 \equiv 0\,, \quad 0\,k \equiv 0\,, \quad 1_{\mathbb{K}}\,\mathsf{E} \equiv \mathsf{E}\,, \quad \mathsf{E}\,1_{\mathbb{K}} \equiv \mathsf{E} \qquad (\mathbf{T}_{\mathbb{K}})$$

$$k\,(h\,\mathsf{E}) \equiv kh\,\mathsf{E}\,, \quad (\mathsf{E}\,k)\,h \equiv \mathsf{E}\,kh\,, \quad (k\,\mathsf{E})\,h \equiv k\,(\mathsf{E}\,h) \qquad (\mathbf{A}_{\mathbb{K}})$$

$$1\,k \equiv k\,1\,, \quad \mathsf{E}\cdot(k\,1) \equiv \mathsf{E}\,k\,, \quad (k\,1)\cdot\mathsf{E} \equiv k\,\mathsf{E} \qquad (\mathbf{U}_{\mathbb{K}})$$

From now on, all computations on weighted expressions are performed modulo the *(trivial identities* $\mathbf{T} = \mathbf{T}_{\mathsf{u}} \wedge \mathbf{T}_{\mathbb{K}} \wedge \mathbf{A}_{\mathbb{K}} \wedge \mathbf{U}_{\mathbb{K}}$. Besides the trivial identities, the *natural identities* $\mathbf{N} = \mathbf{A} \wedge \mathbf{D} \wedge \mathbf{C}$ hold on the expressions of $\mathbb{K}\,\mathsf{RatE}\,M$ for any $\mathbb{K}$ and (graded) M, and, in contrast, the identities $\mathbf{I}$ and $\mathbf{J}$ that are special to $\mathfrak{P}\,(M)$ do not hold anymore.

The system resolution method starts from a proper automaton $\mathcal{A} = \langle\, I, E, T\,\rangle$ of dimension Q whose behaviour is $|\mathcal{A}| = I \cdot V$ where $V = E^* \cdot T$ is a vector in $\mathbb{K}\langle\!\langle M\rangle\!\rangle^{Q}$. Lemma 6.1 easily generalises and as E is proper (in $\mathbb{K}^{Q\times Q}\langle\!\langle M\rangle\!\rangle$), V is the unique solution of the equation $\mathrm{X} = E\,\mathrm{X} + T$ which we rewrite as a system of $\mathsf{Card}\,(Q)$ equations:

$$\forall p \in Q \qquad V_p = \textstyle\sum_{q\in Q} |\mathsf{E}_{p,q}| V_q + |T_p\,1| \tag{6.1}$$

where the V_p are the 'unknowns' and the entries $E_{p,q}$, which are linear combinations of letters of A, are considered as expressions and denoted as such. The system (6.1) may be solved by successive *elimination* of the unknowns, by means of Corollary 6.3. When all unknowns V_q have been eliminated in an ordering ω on Q, the computation yields an expression that we denote by $\mathbf{E}_{\omega}(\mathcal{A})$, as in Sec. 3.3, and $|\mathcal{A}| = |\mathbf{E}_{\omega}(\mathcal{A})|$ holds.

The parallel with the Boolean case can be carried on: given a $\mathbb{K}$-automaton $\mathcal{A}$ of dimension Q, an ordering ω, and a recursive division τ on Q, the expressions $\mathbf{B}_{\omega}(\mathcal{A})$, $\mathbf{M}_{\omega}(\mathcal{A})$, and $\mathbf{C}_{\tau}(\mathcal{A})$ that all denote $|\mathcal{A}|$ are computed by the state elimination method, the McNaughton–Yamada and recursive algorithms respectively. The results on the comparison between these expressions also extend to the weighted case.

Proposition 6.7. *For every order ω on Q, $\mathbf{B}_{\omega}(\mathcal{A}) = \mathbf{E}_{\omega}(\mathcal{A})$ holds.*

Proposition 6.8. *For every order ω on Q, $\mathbf{N} \wedge \mathbf{U} \vdash \mathbf{M}_{\omega}(\mathcal{A}_{p,q}) \equiv \mathbf{B}_{\omega}(\mathcal{A}_{p,q})$ holds.*

Theorem 3.5 also extend to the weighted case, with the important change that the identity $\mathbf{I}$ does not come into play anymore.

Theorem 6.9. *Let ω and ω' be two orders on the set of states of a $\mathbb{K}$-automaton $\mathcal{A}$. Then, $\mathbf{N} \wedge \mathbf{S} \wedge \mathbf{P} \vdash \mathbf{B}_{\omega}(\mathcal{A}) \equiv \mathbf{B}_{\omega'}(\mathcal{A})$ holds.*

6.3 From expressions to automata: the Ψ-maps

6.3.1 The standard automaton of a weighted expression The definition of a *standard* weighted automaton is the same as the one of a standard Boolean automaton: a unique intial state on which the initial map takes the value $1_{\mathbb{K}}$ and which is not the end of any transition. Such an automaton may thus be represented as in Figure 5 and every weighted automaton is equivalent to, and may be turned into, a standard one.

As in the Boolean case, operations are defined on standard weighted automata that are parallel to the rational weighted operators. With the notation of Figure 5, the operators $\mathcal{A}+\mathcal{B}$ and $\mathcal{A}\cdot\mathcal{B}$ are given by (4.1) and (4.2), $k\,\mathcal{A}$ and $\mathcal{A}\,k$ by

$$k\,\mathcal{A} = \left\langle \begin{pmatrix} 1 & \boxed{0} \end{pmatrix}, \begin{pmatrix} 0 & \boxed{k\,J} \\ \boxed{0} & \boxed{F} \end{pmatrix}, \begin{pmatrix} k\,c \\ \boxed{U} \end{pmatrix} \right\rangle, \quad \mathcal{A}\,k = \left\langle \begin{pmatrix} 1 & \boxed{0} \end{pmatrix}, \begin{pmatrix} 0 & \boxed{J} \\ \boxed{0} & \boxed{F} \end{pmatrix}, \begin{pmatrix} c\,k \\ \boxed{U\,k} \end{pmatrix} \right\rangle,$$

and $\mathcal{A}^*$, which is defined when c^* is defined, by the following modification of (4.3):

$$\mathcal{A}^* = \left\langle \begin{pmatrix} 1 & \boxed{0} \end{pmatrix}, \begin{pmatrix} 0 & \boxed{c^*\,J} \\ \boxed{0} & \boxed{H} \end{pmatrix}, \begin{pmatrix} c^* \\ \boxed{U\,c^*} \end{pmatrix} \right\rangle , \qquad (4.3')$$

where $H = U\cdot c^*\,J + F$. As in Sec. 4.2, these operations allow to associate with every weighted expression E and by induction on its depth, a standard weighted automaton $\mathcal{S}_{\mathsf{E}}$ which we call *the* standard automaton of E. Straightforward computations show that $|(k\,\mathcal{A})| = k|\mathcal{A}|$, $|(\mathcal{A}\,k)| = |\mathcal{A}|k$, $|(\mathcal{A}+\mathcal{B})| = |\mathcal{A}|+|\mathcal{B}|$, $|(\mathcal{A}\cdot\mathcal{B})| = |\mathcal{A}|\cdot|\mathcal{B}|$ and $|(\mathcal{A}^*)| = |\mathcal{A}|^*$. From which one concludes that the construction of $\mathcal{S}_{\mathsf{E}}$ is a Ψ-map:

Proposition 6.10 ([13, 39]). *If* E *is a weighted expression over* A^**, then* $|\mathcal{S}_{\mathsf{E}}| = |\mathsf{E}|$.

The automaton $\mathcal{S}_{\mathsf{E}}$ has $\ell(\mathsf{E})+1$ states. Computing $\mathcal{S}_{\mathsf{E}}$ from (4.1), (4.2) and (4.3') is *cubic* in $\ell(\mathsf{E})$ and a *star normal form* for weighted expressions is something that does not seem to exist in the general case. Figure 9 shows the standard $\mathbb{Q}$-automaton $\mathcal{S}_{\mathsf{E}_3}$ associated with $\mathsf{E}_3 = (\frac{1}{6}\,a^* + \frac{1}{3}\,b^*)^*$ and $\mathbb{Z}$-automaton $\mathcal{S}_{\mathsf{E}_4}$ associated with $\mathsf{E}_4 = (1-a)\,a^*$.

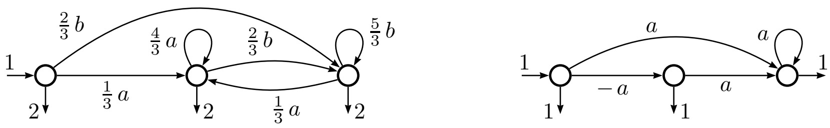


Figure 9. The $\mathbb{Q}$-automaton $\mathcal{S}_{\mathsf{E}_3}$ and the $\mathbb{Z}$-automaton $\mathcal{S}_{\mathsf{E}_4}$

It is the necessary definition of $k\,\mathcal{A}$ and $\mathcal{A}\,k$ that rules out the equivalence $k\,m \equiv m\,k$, with m in M, from the set of trivial identities.

6.3.2 The derived term automaton of a weighted expression The *(left) quotient* operation also extends from languages to series: for every s in $\mathbb{K}\langle\langle A^*\rangle\rangle$, and every u in A^*, $u^{-1}s$ is defined by $\langle u^{-1}s, v\rangle = \langle s, u\,v\rangle$ for every v in A^*. The quotient is a *(right) action* of A^* on $\mathbb{K}\langle\langle A^*\rangle\rangle$: $(u\,v)^{-1}s = v^{-1}\left(u^{-1}s\right)$.

In contrast with the Boolean case, a series in $\mathbb{K}\mathrm{Rat}\,A^*$ may have an infinite number of distinct quotients. However, the quotient operation allows to express a characteristic property of rational series. Let us call *stable* a subset U of $\mathbb{K}\langle\langle A^*\rangle\rangle$ closed under quotient. Then, a characterisation due to Jacob reads: *a series of* $\mathbb{K}\langle\langle A^*\rangle\rangle$ *is* $\mathbb{K}$*-rational if, and only if, it is contained in a finitely generated stable submodule of* $\mathbb{K}\langle\langle A^*\rangle\rangle$, *cf.* [8, 54].

$\mathbb{K}$-derivation The *derivation* of rational $\mathbb{K}$-expressions implements the lifting of the quotient of series to the level of expressions. It yields an effective version of the characterisation quoted above.

In the sequel, addition in $\mathbb{K}$ is written $\oplus$ to distinguish it from the $+$ operator in expressions. The set of *(left) linear combinations* of $\mathbb{K}$-expressions with coefficients in $\mathbb{K}$ is denoted, by abuse, by $\mathbb{K}\langle\mathbb{K}\,\mathsf{RatE}\,A^*\rangle$. In the following, $[k\,\mathsf{E}]$ or $k\,\mathsf{E}$ is a monomial in $\mathbb{K}\langle\mathbb{K}\,\mathsf{RatE}\,A^*\rangle$ whereas $(k\,\mathsf{E})$ is an expression in $\mathbb{K}\,\mathsf{RatE}\,A^*$. An external right multiplication on $\mathbb{K}\langle\mathbb{K}\,\mathsf{RatE}\,A^*\rangle$ by an expression and by a scalar is needed in the sequel. It is first defined on monomials by $([k\,\mathsf{E}]\cdot\mathsf{F})\equiv k\,(\mathsf{E}\cdot\mathsf{F})$ and $([k\,\mathsf{E}]\,k')\equiv k\,(\mathsf{E}\,k')$ and then extended to $\mathbb{K}\langle\mathbb{K}\,\mathsf{RatE}\,A^*\rangle$ by linearity.

Definition 6.1 ([39]). The $\mathbb{K}$-*derivation* of E in $\mathbb{K}\,\mathsf{RatE}\,A^*$ with respect to a in A, denoted by $\frac{\partial}{\partial a}\,\mathsf{E}$, is a linear combination of expressions in $\mathbb{K}\,\mathsf{RatE}\,A^*$ defined by (4.4) for the base cases and inductively by the following formulas.

$$\frac{\partial}{\partial a}(k\,\mathsf{E}) = k\,\frac{\partial}{\partial a}\,\mathsf{E}\,,\quad \frac{\partial}{\partial a}(\mathsf{E}\,k) = \left(\left[\frac{\partial}{\partial a}\,\mathsf{E}\right]\,k\right),\quad \frac{\partial}{\partial a}(\mathsf{E}+\mathsf{F}) = \frac{\partial}{\partial a}\,\mathsf{E}\oplus\frac{\partial}{\partial a}\,\mathsf{F}\ ,$$

$$\frac{\partial}{\partial a}(\mathsf{E}\cdot\mathsf{F}) = \left(\left[\frac{\partial}{\partial a}\,\mathsf{E}\right]\cdot\mathsf{F}\right)\oplus\mathsf{c}(\mathsf{E})\,\frac{\partial}{\partial a}\,\mathsf{F}\,,\quad\text{and}\quad \frac{\partial}{\partial a}(\mathsf{E}^*) = \mathsf{c}(\mathsf{E})^*\,\left(\left[\frac{\partial}{\partial a}\,\mathsf{E}\right]\cdot(\mathsf{E}^*)\right)\ .$$

The last equation is defined only if (E^*) is a *valid* expression. The $\mathbb{K}$-derivation of an expression with respect to a *word* u is defined by induction on the length of u: for every u in A^* and every a in A, $\frac{\partial}{\partial ua}\,\mathsf{E} = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\,\mathsf{E}\right)$ and the definition of $\mathbb{K}$-derivation is consistent with that of quotient of series since for every u in A^* $\left|\frac{\partial}{\partial u}(\mathsf{E})\right| = u^{-1}|\mathsf{E}|$ holds.

The derived term automaton At Sec. 4.3.1, we have defined the *derived terms* of a (Boolean) expression as the expressions that occur in a derivation of that expression. Proposition 4.5 then established properties that allow to compute these derived terms, without derivation. For the weighted case, we take the same properties as the definition.

Definition 6.2 ([39]). The set $\mathrm{TD}\,(\mathsf{E})$ of *true derived terms* of E in $\mathbb{K}\,\mathsf{RatE}\,A^*$ is inductively defined by: $\mathrm{TD}\,(k\,\mathsf{E}) = \mathrm{TD}\,(\mathsf{E})$, $\mathrm{TD}\,(\mathsf{E}\,k) = (\mathrm{TD}\,(\mathsf{E})\;k)$, $\mathrm{TD}\,(\mathsf{E}+\mathsf{F}) = \mathrm{TD}\,(\mathsf{E})\,\cup\,\mathrm{TD}\,(\mathsf{F})$, $\mathrm{TD}\,(\mathsf{E}\cdot\mathsf{F}) = (\mathrm{TD}\,(\mathsf{E}))\cdot\mathsf{F}\cup\mathrm{TD}\,(\mathsf{F})$, $\mathrm{TD}\,(\mathsf{E}^*) = (\mathrm{TD}\,(\mathsf{E}))\cdot\mathsf{E}^*$, starting from the base cases $\mathrm{TD}\,(0) = \mathrm{TD}\,(1) = \emptyset$, and $\mathrm{TD}\,(a) = 1$ for every a in A.

$\mathrm{TD}\,(\mathsf{E})$ is a set of unitary monomials of $\mathbb{K}\langle\mathbb{K}\,\mathsf{RatE}\,A^*\rangle$, with $\mathsf{Card}\,(\mathrm{TD}\,(\mathsf{E}))\leqslant\ell(\mathsf{E})$. The set of *derived terms* of E is $\mathrm{D}\,(\mathsf{E}) = \mathrm{TD}\,(\mathsf{E})\cup\{\mathsf{E}\}$. Theorem 6.11 insures consistency between Definitions 6.1 and 6.2; the usefulness of the latter follows from Theorem 6.12.

Theorem 6.11 ([49, 39]). *Let* E *be in* $\mathbb{K}\,\mathsf{RatE}\,A^*$ *and* $\mathrm{D}(\mathsf{E}) = \{\mathsf{K}_1, ..., \mathsf{K}_n\}$. *For every* a *in* A, *there exist an* $n \times n$*-matrix* $\mu(a)$ *with entries in* $\mathbb{K}$ *such that*

$$\forall i \in [n] \qquad \frac{\partial}{\partial a}\,\mathsf{K}_i = \bigoplus_{j\in[n]} \mu(a)_{i,j}\,\mathsf{K}_j\ .$$

The $\mathbb{K}$-derivation of an expression E in $\mathbb{K}\,\mathsf{RatE}\,A^*$ with respect to every word in A^* is thus a linear combination of derived terms of E. Hence the derived terms of an expression denote *the generators of a stable submodule* that contains the series denoted by the expression. Theorem 6.11 yields the *the derived term automaton* of E, $\mathcal{A}_{\mathsf{E}} = \langle\, I, X, T\,\rangle$, of dimension $\mathrm{D}(\mathsf{E})$, with $I = 1_{\mathbb{K}}$ if $\mathsf{K}_i = \mathsf{E}$ and $0_{\mathbb{K}}$ otherwise, $X = \bigoplus_{a\in A} \mu(a)\,a$, and $T_j = \mathsf{c}(\mathsf{K}_j)$. The $\mathbb{K}$-derivation is another Ψ-map since $|\mathcal{A}_{\mathsf{E}}| = |\mathsf{E}|$ holds.

Morphisms and quotients of (Boolean) automata are generalised to $\mathbb{K}$*-coverings* and $\mathbb{K}$*-quotients* of $\mathbb{K}$-automata (*cf.* [54]). Theorem 4.8 is then extended to the weighted case.

Theorem 6.12 ([39]). *Let* E *be in* $\mathbb{K}\,\mathsf{RatE}\,A^*$. *Then* $\mathcal{A}_{\mathsf{E}}$ *is a* $\mathbb{K}$*-quotient of* $\mathcal{S}_{\mathsf{E}}$.

Remark 6.13. This statement is a justification for Definition 4.5. The monomials that appear in the $\mathbb{K}$-derivations of an expression E are in $\mathrm{D}(\mathsf{E})$. The converse is not necessarily true when $\mathbb{K}$ is not a positive semiring: some derived terms may never appear in a $\mathbb{K}$-derivation, as it can be observed for instance on the $\mathbb{Z}$-expression $\mathsf{E}_4 = (1-a)\,a^*$ (*cf.* Figure 10). With a definition of derived terms based on $\mathbb{K}$-derivation only, Theorem 6.12 would not hold anymore.

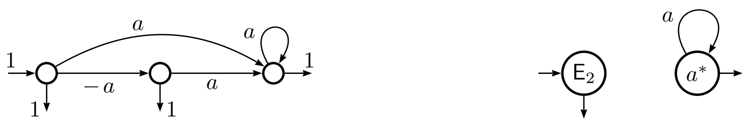


Figure 10. The $\mathbb{Z}$-automaton $\mathcal{S}_{\mathsf{E}_4}$ and its $\mathbb{Z}$-quotient $\mathcal{A}_{\mathsf{E}_4}$

6.4 Recognisable series

The distinction between *rational* and *recognisable* carries over from subsets of a monoid M to series over M. The equivalence between automata over free monoids and matrix representation (*cf.* Sec. 2.3) paves the way to the definition of recognisability.

A $\mathbb{K}$*-representation* of M of dimension Q is a triple (λ, μ, ν) where $\mu\colon M \to \mathbb{K}^{Q\times Q}$ is a morphism, and λ and ν are two vectors of $\mathbb{K}^Q$. The representation (λ, μ, ν) realises the series $s = \sum_{m\in M} (\lambda \cdot \mu(m) \cdot \nu)\, m$; a series in $\mathbb{K}\langle\langle M\rangle\rangle$ is *recognisable* if it is realised by a representation and the set of recognisable series is denoted by $\mathbb{K}\mathrm{Rec}\,M$. The family of rational and of recognisable series are distinct in general, and, a proof which is very similar to the one given at Sec. 2.3, and *independent from* $\mathbb{K}$, yields the following.

Theorem 6.14 (Kleene–Schützenberger). *If* A *is finite, then* $\mathbb{K}\mathrm{Rat}\,A^* = \mathbb{K}\mathrm{Rec}\,A^*$.

7 Notes

Most of the material presented in this chapter has appeared in previous work of the author [52, 53, 54]. A detailed version of this chapter is to be found at `http://arxiv.org/abs/xxx.yyy`

Sec. 1. New look at Kleene's theorem A detailled history of the development of ideas at the beginning of the theory of automata is given in [45]. Berstel [6] attributes to Eilenberg the idea of distinguishing the family of recognisables from that of rationals.

Besides the already quoted Elgot and Mezei's paper [22], other authors have certainly noticed the equality of expressiveness of automata and expressions beyond free monoids. It is part of Walljasper's thesis [57]; it can be found in Eilenberg's treatise [21]. The splitting of Kleene's theorem has been proposed in [51].

Sec. 3. From automata to expressions First note that this section is mostly of theoretical interest: for which practical purpose would one exchange an automaton for an expression?

Identities. As mentioned, the axiomatisation of rational expressions, even hinting at bibliographic references, is out of the scope of this chapter. Conway showed that besides the identities $\mathbf{S}$ and $\mathbf{P}$ (that are at the basis of the definition of the so-called 'Conway semirings', *cf.* Chap. XX), each finite simple *group* gives rise to an identity that is independant from the others [17]. Krob, who showed that this set of identities is complete, coined $\mathbf{S}$ and $\mathbf{P}$ the *aperiodic identities* [35].

State elimination method. The example $\mathcal{D}_3$ of Figure 3 is easily generalised so as to find an exponential gap between the length of expressions for two distinct orders. Finding an order on states that yields a shortest expression is PSPACE-complete *cf.* [32]. The search for short expressions is performed by heuristics; as reported in [29], the naive one, modified as in [18], appears to be good.

McNaughton–Yamada algorithm is the implementation in the semiring of languages of the comtemporary Floyd–Roy–Warshall algorithms (in the Boolean or tropical semirings) [26, 47, 58].

Star height. The star height of a rational language L is *the minimum of the star heights of the rational expressions that denote* L. Whether the star height of a language is effectively computable has been a long standing open problem until it was positively solved first by K. Hashiguchi [30] and then by D. Kirsten [33].

Sec. 4. From expressions to automata The presentation of the standard automaton given here is not the classical one, and not only for the chosen name. The recursive definition, also used in [24] for instance, avoids the definition of `First`, `Last`, and `Follow` functions that are built in most papers on the subject. Based on these functions, other automata may be defined: *e.g.* in [42] they are used to compute directly the *determinisation* of $\mathcal{S}_{\mathsf{E}}$, in [31] positions with the same image by `Follow` are merged, giving rise to a possibly smaller automaton, called *follow automaton*.

Attributing derivation to Brzozowski and Antimirov together is an unusual but sensible foreshortening. Original Brzozowski's *derivatives* [11] are obtained by replacing '$\cup$' by a '$+$' in (4.5) and (4.6). Derivatives are then *expressions*, and there is a finite number of them, modulo the $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{I}$ identities. By replacing the '$+$' by a '$\cup$' in Brzozowski's definition, Antimirov [4] changed the derivatives into a *set of expressions*, which he called *partial derivatives*, as they are 'parts' of derivatives. As they are applied to union of sets, and not to expressions, the $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{I}$ identities come for free, and are no longer necessary to insure the finiteness of derived terms.

A common technique for defining Ψ-maps has been the *linearisation* $\overline{\mathsf{E}}$ of the expression E, that is, making all letters in E distinct by indexing them by their position in E (*e.g.* [42, 31]). Berry–Sethi [5] showed that the (Brzozowski) derivatives of $\overline{\mathsf{E}}$ coincide with the states of $\mathcal{S}_{\mathsf{E}}$, whereas Berstel–Pin [7] observed that $|\mathsf{E}|$ is a *local* language $\overline{L}$ and interpreted Berry-Sethi's result as the construction of *the* deterministic automaton canonically associated with $\overline{L}$.

The similarity between Mirkin's prebases [43] and Antimirov's derived terms was noted by Champarnaud–Ziadi [15], who called *equation automaton* the derived term automaton. Allauzen–Mohri have generalised Proposition 4.4 and Theorem 4.8 and computed $\mathcal{A}_{\mathsf{E}}$ and the follow automaton of E from $\mathcal{T}_{\mathsf{E}}$ by quotient and elimination of marked spontaneous transitions [2].

In [14], an algorithm is given which is a kind of converse of a Ψ-map: it recognises if an automaton is the standard automaton $\mathcal{S}_{\mathsf{E}}$ of an expression E and, in this case, computes such an E in *star normal form*. The problem of inverting a Φ-map has been given a partial answer in [40]: it is possible to compute $\mathcal{A}$ from $\mathbf{B}_{\omega}(\mathcal{A})$ for certain $\mathcal{A}$ (and any ω); this has lead to the definition of a variant of the derivation: the *broken derivation*, that has been further studied in [3].

Sec. 5. Changing the monoid Proposition 5.5 leads naturally to consider monoids M in which $\operatorname{Rat} M = \operatorname{Rec} M$ holds, and which one could call *Kleene monoids*. In [50], was defined the family of *rational monoids* which contains all previously known examples of Kleene monoids; still the inclusion is strict [44]. *Commutative* Kleene monoids, as well as finitely generated submonoids of $\operatorname{Rat} a^*$ are rational monoids [48, 1].

Sec. 6. Introducing weights If the definition of rational (and algebraic) series in non-commuting variables as generalisation of rational (and context-free) languages on one hand-side, as well as the formalisation of rational expressions on the other, date back to the beginning of automata theory, the formalisation of *weighted* rational expressions seem to have appeared in various papers in the years 2000 only [13, 49, 39]. A satisfactory definition of trivial identities for weighted expressions proves to be tricky and has evolved in the publications of the author.

By replacing quotient and derivation by *co-induction*, Rutten formulated the equivalent of Theorem 6.11 [49].

Krob [36] and Berstel–Reutenauer [9] have considered 'weighted rational expression' slightly different from those dealt with in this chapter. With their *differentiation* and *derivation*, they have tackled different problems than the construction of Ψ-maps.

Acknowledgements The author is grateful to Z. Ésik and to J. Brzozowski who read a first draft of this chapter and made numerous helpful remarks. The unbounded patience of Jean-Éric Pin is also acknowledged.

References

[1] S. Afonin and E. Khazova. On the structure of finitely generated semigroups of unary regular languages. *Int. J. Foundations Computer Sci.*, 21:689–704, 2010. 32

[2] C. Allauzen and M. Mohri. A unified construction of the Glushkov, Follow, and Antimirov automata. In R. Kralovic and P. Urzyczyn, editors, *MFCS 2006*, number 4162 in Lect. Notes in Comput. Sci., pages 110–121, 2006. 32

[3] P.-Y. Angrand, S. Lombardy, and J. Sakarovitch. On the number of broken derived terms of a rational expression. *J. Automata, Languages, and Combinatorics*, 15:27–51, 2010. 21, 32

[4] V. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Computer Sci.*, 155:291–319, 1996. 18, 19, 20, 31

[5] G. Berry and R. Sethi. From regular expressions to deterministic automata. *Theoret. Computer Sci.*, 48:117–126, 1986. 31

[6] J. Berstel. *Transductions and Context-Free Languages*. Teubner, 1979. 31

[7] J. Berstel and J.-E. Pin. Local languages and the Berry-Sethi algorithm. *Theoret. Computer Sci.*, 155:439–446, 1996. 31

[8] J. Berstel and C. Reutenauer. *Les séries rationnelles et leurs langages*. Masson, 1984. Translation: *Rational Series and Their Languages.* Springer, 1988. 24, 29

[9] J. Berstel and C. Reutenauer. Extension of Brzozowski's derivation calculus of rational expressions to series over the free partially commutative monoids. *Theoret. Computer Sci.*, 400(1-3):144–158, 2008. 32

[10] A. Brügemann-Klein. Regular expressions into finite automata. *Theoret. Computer Sci.*, 120:197–213, 1993. 17, 18

[11] J. A. Brzozowski. Derivatives of regular expressions. *J. Assoc. Comput. Mach.*, 11:481–494, 1964. 18, 31

[12] J. A. Brzozowski and E. J. McCluskey. Signal flow graph techniques for sequential circuit state diagrams. *IEEE Trans. Electronic Computers*, 12:67–76, 1963. 9

[13] P. Caron and M. Flouret. Glushkov construction for multiplicities. In A. Paun and S. Yu, editors, *CIAA 2000*, number 2088 in Lect. Notes in Comput. Sci., pages 67–79, 2001. 28, 32

[14] P. Caron and D. Ziadi. Characterization of Glushkov automata. *Theoret. Computer Sci.*, 233:75–90, 2000. 32

[15] J.-M. Champarnaud and D. Ziadi. From Mirkin's prebases to Antimirov's word partial derivatives. *Fundam. Inform.*, 45(3):195–205, 2001. 32

[16] J.-M. Champarnaud and D. Ziadi. Canonical derivatives, partial derivatives and finite automaton constructions. *Theoret. Computer Sci.*, 289:137–163, 2002. 21

[17] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971. 10, 13, 31

[18] M. Delgado and J. Morais. Approximation to the smallest regular expression for a given regular language. In M. Domaratzki, A. Okhotin, K. Salomaa, and S. Yu, editors, *CIAA 2004*, volume 3317 of *Lect. Notes in Comput. Sci.*, pages 312–314, 2004. 31

[19] M. Droste, W. Kuich, and H. Vogler. (Ed.), *Handbook of Weighted Automata*, Springer, 2009. 24

[20] L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10:385–397, 1963. 14

[21] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974. 31

[22] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. and Develop.*, 9:47–68, 1965. 22, 31

[23] P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *J. Computer System Sci.*, 2:88–101, 1968. 23

[24] S. Fischer, F. Huch, and T. Wilke. A play on regular expressions: functional pearl. In P. Hudak and S. Weirich, editors, *ICFP 2010*, pages 357–368, 2010. 31

[25] M. Fliess. Deux applications de la représentation matricielle d'une série non commutative. *J. Algebra*, 19:344–353, 1971. 23

[26] R. W. Floyd. Algorithm 97. *Comm. Assoc. Comput. Mach.*, 5:345, 1962. 31

[27] S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas and languages. *Pacif. J. Math.*, 16:285–296, 1966. 23

[28] V. M. Glushkov. The abstract theory of automata. *Russian Math. Surveys*, 16:1–53, 1961. 16

[29] H. Gruber, M. Holzer, and M. Tautschnig. Short regular expressions from finite automata: Empirical results. In S. Maneth, editor, *CIAA 2009*, volume 5642 of *Lect. Notes in Comput. Sci.*, pages 188–197, 2009. 31

[30] K. Hashiguchi. Algorithms for determining relative star height and star height. *Inform. and Comput.*, 78:124–169, 1988. 31

[31] L. Ilie and S. Yu. Follow automata. *Inform. and Comput.*, 186(1):140–162, 2003. 31

[32] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993. 31

[33] D. Kirsten. Distance desert automata and the star height problem. *RAIRO Theor. Informatics and Appl.*, 39(3):455–509, 2005. 31

[34] S. C. Kleene. Representation of events in nerve nets and finite automata. in C. Shannon and J. McCarthy, editors, *Automata Studies*, Princeton Univ. Press, pages 3–41, 1956. 2

[35] D. Krob. Complete systems of B-rational identities. *Theoret. Computer Sci.*, 89:207–343, 1991. 10, 31

[36] D. Krob. Differentiation of K-rational expressions. *Int. J. of Algebra and Computation*, 2:57–87, 1992. 32

[37] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. Springer, 1986. 24

[38] S. Lombardy and J. Sakarovitch. On the star height of rational languages. In M. Ito, editor, *Words, Languages and Combinatorics III*. World Scientific, 2003. 15

[39] S. Lombardy and J. Sakarovitch. Derivation of rational expressions with multiplicity. *Theoret. Computer Sci.*, 332:141–177, 2005. 28, 29, 30, 32

[40] S. Lombardy and J. Sakarovitch. How expressions can code for automata. *RAIRO Theor. Informatics and Appl.*, 39:217–237, 2005. Corrigendum. 44:339–362, 2010. 32

[41] J. McKnight. Kleene's quotient theorems. *Pacific J. Math.*, 14:43–52, 1964. 23

[42] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Trans. Electronic Computers*, 9:39–47, 1960. 11, 16, 31

[43] B. G. Mirkin. An algorithm for constructing a base in a language of regular expressions. *Engineering Cybernetics*, 5:51–57, 1966. 20, 32

[44] M. Pelletier and J. Sakarovitch. Easy multiplications II. Extensions of rational semigroups. *Inform. and Comput.*, 88:18–59, 1990. 32

[45] D. Perrin. Les débuts de la théorie des automates. *Technique et Science Informatique*, 14:409–443, 1995. 31

[46] M. O. Rabin and D. Scott. Finite automata and their decision problems. *I.B.M. J. Res. Develop.*, 3:125–144, 1959. Reprinted in *Sequential Machines : Selected Papers* (E. Moore, ed.), Addison-Wesley, 1965. 23

[47] B. Roy. Transitivité et connexité. *C. R. Acad. Sci. Paris Sér. A*, 249:216–218, 1959. 31

[48] C. P. Rupert. On commutative Kleene monoids. *Semigroup Forum*, 43:163–177, 1991. 32

[49] J. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoret. Computer Sci.*, 308:1–53, 2003. 30, 32

[50] J. Sakarovitch. Easy multiplications I. The realm of Kleene's theorem. *Inform. and Comput.*, 74:173–197, 1987. 32

[51] J. Sakarovitch. Kleene's Theorem revisited. In A. Kelemenova and K. Kelemen, editors, *Trends, Techniques and Problems in Theoretical Computer Science*, number 281 in Lect. Notes in Comput. Sci., pages 39–50, 1987. 31

[52] J. Sakarovitch. *Eléments de théorie des automates*. Vuibert, 2003. Corrected English translation: *Elements of Automata Theory*, Cambridge University Press, 2009. 11, 12, 24, 31

[53] J. Sakarovitch. The Language, the Expression and the (small) Automaton. In J. Farré, I. Litovsky, and S. Schmitz, editors, *CIAA 2005*, number 3845 in Lect. Notes in Comput. Sci., pages 15–30, 2005. 31

[54] J. Sakarovitch. Rational and recognisable power series, 2009. in M. Droste et al., editors, *Handbook of Weighted Automata*, Springer, pages 105–174. 29, 30, 31

[55] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer, 1977. 24

[56] K. Thompson. Regular expression search algorithm. *Comm. Assoc. Comput. Mach.*, 11:419–422, 1968. 18

[57] S. J. Walljasper. *Non-Deterministic Automata and Effective Languages*. PhD thesis, Univ. Iowa, 1970. 31

[58] S. Warshall. A theorem on Boolean matrices. *J. Assoc. Comput. Mach.*, 9:11–12, 1962. 31

[59] D. Wood. *Theory of Computation*. John Wiley, 1987. 9

[60] S. Yu. Regular languages. in G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, vol. 1, Elsevier, pages 41–111, 1997. 9