Automata and rational expressions

Jacques Sakarovitch

Automata and rational expressions

Jacques Sakarovitch

LTCI, CNRS and Télécom ParisTech

2010 Mathematics Subject Classification: 68Q45

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

This text is an extended version of the chapter 'Automata and rational expressions' in the *AutoMathA Handbook* [47] that will appear soon, published by the European Science Foundation and edited by Jean-Éric Pin.

It contains not only proofs, examples, and remarks that had been discarded due to the severe space constraints induced by the edition of a handbook of very large scope, but also developments that were not included as they did not seem to belong to the main stream of the subject. For that reason, the numbering of theorems, propositions, defintions, *etc.* may differ in the two versions, even if the general outline is the same.

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 1.4.11 in Chapter 1 of [47]. 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 directed 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 new means, 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 Section 2 how one gets from rational sets to recognisable sets in the case of free monoids and in Section 5, we see that the same construction fails in non-free monoids and explore what remains true.

The languages and their representation Formal languages or, in the weighted variant, 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 view.

In order to prove that the family of languages accepted by finite automata coincide with that of the languages denoted by rational expressions we proceed 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 Γ-*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 Δ-*map*.

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

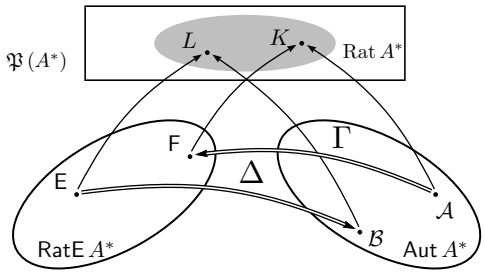


Figure 1. The Γ- and Δ-maps

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*. This is where I 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 Section 5 and to series in Section 6.

Notation and definitions given in Chapter 1 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 by $\mathrm{Rat}\, A^*$ 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. A precise structure-revealing specification for building elements of this family can be given by *rational expressions*.

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 $\mathsf{Rat}\mathsf{E}\, A^*$ the set of rational expressions over A^* and often write *expression* for *rational expression*. (As in [54], 'rational expression' is preferred to the more traditional *regular expression* for several reasons and in particular as it will be used in the weighted case as well, see Section 6.)

With every expression E in $\mathsf{Rat}\mathsf{E}\, A^*$ is associated a language of A^*, which is called *the language denoted by* E and we write[2] it as $|\mathsf{E}|$. The language $|\mathsf{E}|$ is inductively defined by[3] $|\mathsf{0}| = \emptyset$, $|\mathsf{1}| = \{1_{A^*}\}$, $|a| = \{a\}$ 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 Section 4 yields different results on these two expressions.

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

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

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

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

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

Definition 2.2. The *constant term* of an expression E over A^*, written $\mathsf{c}(\mathsf{E})$, is the Boolean value, inductively defined and computed 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 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, hence, if A is finite, if and only if (the useful part of) Q is finite.

A *computation* in $\mathcal{A}$ from state p to state q with label w is denoted by $p \xrightarrow[\mathcal{A}]{w} q$. 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 (Γ-maps). *For every finite automaton $\mathcal{A}$ over A^*, there exist rational expressions over A^* which denote $|\mathcal{A}|$.*

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

Section 3 describes how expressions are computed from automata, Section 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.

[5] I prefer not to speak of the language 'recognised' by an automaton, and I 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.

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.* Section 1.5.2).

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 $\mathrm{Rec}\, A^*$.

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

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 (the Boolean sum) of letters that label the 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 the 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 a formal sum $E = \sum_{a \in A} \mu(a)\, a$, where for every a in A, $\mu(a)$ is a Boolean $Q \times Q$-matrix. These matrices $\mu(a)$ define a morphism $\mu \colon A^* \to \mathbb{B}^{Q \times Q}$. The second lemma involves the *freeness* of A^* and 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 $E = \left\{ \big(n, a, n\,\alpha(a)\big) \;\middle|\; a \in A, n \in N \right\}$. 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 Section 5 the example that shows that recognisability and rationality are indeed two distinct concepts and the description of the relationships that can be found between them. As mentioned in Chapter 1, 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 Γ-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

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

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 denote the behaviour of 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 Chapter 20). We shall see however that a *basic set of identities* is sufficient to deduce the equivalence between the expressions computed by the different Γ-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 with their associativity, distributivity and commutativity properties:

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

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

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

The conjunction $\mathbf{A} \wedge \mathbf{D} \wedge \mathbf{C}$ 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 ($\mathbf{U}$) 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, K^*L is* the unique solution *if $\mathsf{c}(K) = 0$, of the equation $\mathrm{X} = K\,\mathrm{X} + L$.*

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 $\mathsf{c}(\mathsf{K}) = \mathsf{0}$. Then, $\mathsf{K}^*\mathsf{L}$ denotes the unique solution of $\mathrm{X} = |\mathsf{K}|\,\mathrm{X} + |\mathsf{L}|$.*

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

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

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

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

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

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:

$$\mathsf{E}+\mathsf{E} \equiv \mathsf{E} \ , \tag{I}$$

$$(\mathsf{E}^*)^* \equiv \mathsf{E}^* \ . \tag{J}$$

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.* Section 6).

3.2 The state-elimination method

The algorithm known as *state-elimination method*, originally due to Brzozowski and McCluskey [13], 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.* [61, 62]).

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(that can be decided beforehand or determined step by step). At the end, only remain states i and t, together 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 $\omega_1 = r < p < q$. It shows the result $\mathbf{B}_{\omega_1}(\mathcal{D}_3) = a^*b(ba^*b + ab^*a)^*ba^* + a^*$. The computation of $\mathbf{B}_\omega(\mathcal{A})$ may silently involve identities in $\mathbf{N}$. A common and natural way of performing the computation is to use identities **I** and **J** as well: it yields simpler results. It is then to be stressed that the use of **I** and **J** is not needed to establish these equivalence results.

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 other order $\omega_2 = r < q < p$ yields $\mathbf{B}_{\omega_2}(\mathcal{D}_3) = (a + b(ab^*a)^*b)^*$, and $\omega_3 = p < q < r$ yields

$\mathbf{B}_{\omega_3}(\mathcal{D}_3) = a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b + a(ba^*b)^*a)^*a(ba^*b)^*ba^*$.

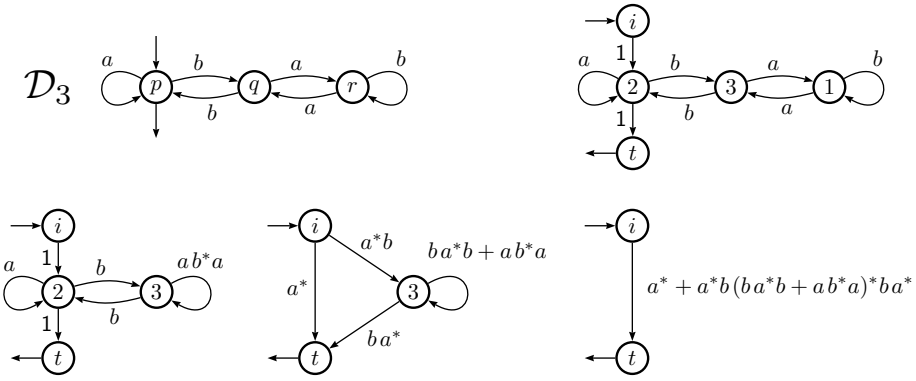


Figure 3. The state-elimination method exemplified on the automaton $\mathcal{D}_3$

Theorem 3.5 (Conway [18], Krob [35]). *Let ω and ω' be two orders on the set of states of an 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.*

Proof. We can go from any order ω to any other order ω' on Q by a sequence of transpositions. We therefore arrive at the situation illustrated in Figure 4 (left) and need to show that the expressions obtained when we first remove the state r and then r' are equivalent to those obtained from removing first r' and then r, modulo $\mathbf{S} \wedge \mathbf{P}$ (without mentioning the *natural identities*).

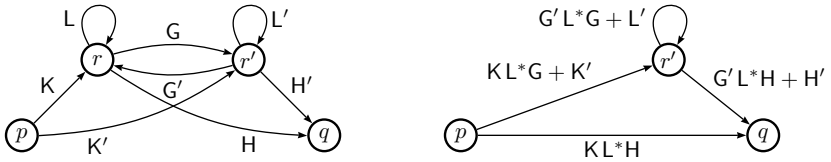


Figure 4. First step of two in the state-elimination method

The removal of state r gives the expressions in Figure 4 (right). The removal of state r'

gives the expression:

$$\mathsf{E} = \mathsf{K}\mathsf{L}^*\mathsf{H} + (\mathsf{K}\mathsf{L}^*\mathsf{G} + \mathsf{K}')\,[\mathsf{G}'\mathsf{L}^*\mathsf{G} + \mathsf{L}']^*\,(\mathsf{G}'\mathsf{L}^*\mathsf{H} + \mathsf{H}')\ ,$$

which using $\mathbf{S}$ (and the natural identities) becomes:

$$\begin{aligned}\mathsf{E} \equiv\ & \mathsf{K}\mathsf{L}^*\mathsf{H} + \mathsf{K}\mathsf{L}^*\mathsf{G}\,\big[\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{G}\big]^*\,\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{H} \\ & + \mathsf{K}'\,\big[\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{G}\big]^*\,\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{H} + \mathsf{K}\mathsf{L}^*\mathsf{G}\,\big[\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{G}\big]^*\,\mathsf{L}'^*\mathsf{H}' \\ & + \mathsf{K}'\,\big[\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{G}\big]^*\,\mathsf{L}'^*\mathsf{H}'\,.\end{aligned}$$

We write:

$$\mathsf{K}'\,\big[\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{G}\big]^*\,\mathsf{L}'^*\mathsf{H}' \equiv \mathsf{K}'\mathsf{L}'^*\mathsf{H}' + \mathsf{K}'\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\,\big[\mathsf{G}\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\big]^*\,\mathsf{G}\mathsf{L}'^*\mathsf{H}'$$

by using $\mathbf{P}$, and then, by 'switching the brackets' (using the identity $(\mathsf{X}\mathsf{Y})^*\mathsf{X} \equiv \mathsf{X}(\mathsf{Y}\mathsf{X})^*$ which is also a consequence of $\mathbf{P}$), we obtain:

$$\begin{aligned}\mathsf{E} \equiv\ & \mathsf{K}\mathsf{L}^*\mathsf{H} \\ & + \mathsf{K}\mathsf{L}^*\mathsf{G}\,\big[\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{G}\big]^*\,\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{H} + \mathsf{K}'\mathsf{L}'^*\mathsf{G}'\,\big[\mathsf{L}^*\mathsf{G}\mathsf{L}'^*\mathsf{G}'\big]^*\,\mathsf{L}^*\mathsf{H} \\ & + \mathsf{K}\mathsf{L}^*\mathsf{G}\,\big[\mathsf{L}'^*\mathsf{G}'\mathsf{L}^*\mathsf{G}\big]^*\,\mathsf{L}'^*\mathsf{H}' + \mathsf{K}'\mathsf{L}'^*\mathsf{G}'\,\big[\mathsf{L}^*\mathsf{G}\mathsf{L}'^*\mathsf{G}'\big]^*\,\mathsf{L}^*\mathsf{G}\mathsf{L}'^*\mathsf{H}' \\ & + \mathsf{K}'\mathsf{L}'^*\mathsf{H}'\end{aligned}$$

an expression that is perfectly symmetric in the letters with and without 'primes', which shows that we would have obtained the same result if we had started by removing r' then r. □

Aside from the formal proximity between expressions obtained from a given automaton, the question of *the length* of these expressions is of course of interest, both from a theoretical as well as practical point of view. The above example $\mathcal{D}$ is easily generalised so as to find an exponential gap between the length of expressions for two distinct orders. The search for short expressions is performed by heuristics, with more or less degree of sophistication (see Notes).

3.3 The system-solution 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, which allows then easier formal proofs.

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

$$|\mathcal{A}| = \sum_{p \in I} L_p = \sum_{p \in Q} \delta_{p,I}\, L_p \tag{3.2}$$

$$\forall p \in Q \qquad L_p = \sum_{q \in Q} |\mathsf{E}_{p,q}| L_q + |\delta_{p,T}| \tag{3.3}$$

where the L_p are the 'unknowns' and the entries $E_{p,q}$, which represent subsets of A, as expressions $\mathsf{E}_{p,q}$ are sums of letters labelling paths of length 1. The system (3.3) may be solved by successive *elimination* of the unknowns. The pivoting operations, which involve subtraction and division that are not available in the semiring $\mathfrak{P}(A^*)$, are replaced by the application of Arden's lemma, since $\mathsf{c}(\mathsf{E}_{p,q}) = \mathsf{0}$ for all p, q in Q.

After the elimination of a certain number of unknowns L_p — we write Q' for the set of indices of those which have not been eliminated — we obtain a system of the form:

$$|\mathcal{A}| = \sum_{p \in Q'} |\mathsf{G}_p| L_p + |\mathsf{H}| \tag{3.4}$$

$$\forall p \in Q' \qquad L_p = \sum_{q \in Q'} |\mathsf{F}_{p,q}| L_q + |\mathsf{K}_p| \tag{3.5}$$

If we choose (arbitrarily) one element q in Q', Corollary 3.3 applied to the corresponding equation from the system (3.5), yields:

$$L_q = |\mathsf{F}_{q,q}^*| \left(\sum_{p \in Q' \setminus q} |\mathsf{F}_{q,p}| L_p + |\mathsf{K}_q| \right) \tag{3.6}$$

which allows the elimination of L_q in (3.4)–(3.5) and gives:

$$|\mathcal{A}| = \sum_{r \in Q' \setminus q} \left(|\mathsf{G}_r + \mathsf{G}_q\, \mathsf{F}_{q,q}^*\, \mathsf{F}_{q,r}| \right) L_r + |\mathsf{H} + \mathsf{G}_q\, \mathsf{F}_{q,q}^*\, \mathsf{K}_q| \tag{3.7}$$

$$\forall r \in Q' \setminus p \quad L_r = \sum_{p \in Q' \setminus q} \left(|\mathsf{F}_{r,p} + \mathsf{F}_{r,q}\, \mathsf{F}_{q,q}^*\, \mathsf{F}_{r,p}| \right) L_p + |\mathsf{K}_r + \mathsf{F}_{r,q}\, \mathsf{F}_{q,q}^*\, \mathsf{K}_q| \ . \tag{3.8}$$

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 (3.7) becomes:

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

As for the state-elimination method, the identities $\mathbf{N}$ (and $\mathbf{I}$ and $\mathbf{J}$) 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-solution are indeed one and the same algorithm for computing the language accepted by a finite automaton, as stated by the following.

Proposition 3.6 ([54]). *For any order ω on the states of $\mathcal{A}$, it holds:*

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

Proof. We can build a generalised automaton $\mathcal{B}'$ corresponding to the system (3.4)–(3.5), with set of states is $Q' \cup \{i, t\}$, where i and t do not belong to Q', and such that, for all p and q in Q':

(i) the transition from p to q is labelled $F_{p,q}$;
(ii) the transition from p to t is labelled K_p;
(iii) the transition from i to p is labelled G_p; and
(iv) the transition from i to t is labelled H.

Note that this definition applied to the system (3.2)–(3.3) characterises the automaton constructed in the first phase of the state-elimination method applied to $\mathcal{A}$.

The elimination in the system (3.4)–(3.5) of the unknown L_q by substitutions and the application of Arden's lemma give the system (3.7)–(3.8) whose coefficients are exactly the transition labels of the generalised automaton obtained by removing the state q from $\mathcal{B}'$.

Thus, since the starting points correspond and since each step maintains the correspondence, the expression obtained for $|\mathcal{A}|$ by the state-elimination method is the same as that obtained by the solution of the system (3.2)–(3.3). □

The state-elimination method reproduces, in the automaton $\mathcal{A}$, the computations corresponding to the solution 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,*

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

3.4 The McNaughton–Yamada algorithm

Given $\mathcal{A} = \langle\, Q, A, E, I, T \,\rangle$, the McNaughton–Yamada algorithm ([43]) — called here MN-Y algorithm for short — truly addresses the problem of computing the matrix E^*, whereas the two preceding methods rather compute the sum of some of the entries of E^*. Like the former methods, it relies on an ordering of Q but it is based on a different grouping of computations[7] within $\mathcal{A}$.

Description of the algorithm We write $M_{p,q}$ for $(E^*)_{p,q}$:

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

The set Q ordered by ω is identified with the set of integers from 1 to $n = \mathsf{Card}\,(Q)$. The key idea of the algorithm is to group the set of paths 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 paths 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 $|\mathsf{M}_{p,q}^{(k)}| = M_{p,q}^{(k)}$.

[7] In order to avoid confusion between the *computations* of expressions that denote the language accepted by $\mathcal{A}$ and whose variations are the subject of the chapter, and the *computations* within $\mathcal{A}$, which is the way we call the paths in the labelled directed graph $\mathcal{A}$, we use the latter terminology in this section.

A path that does not pass through any intermediate state of rank greater than 0 passes through no intermediate states, and therefore reduces to a single transition. Thus, $M_{p,q}^{(0)}$ is, for all p and q in Q, the set of labels of transitions which go from p to q; that is, $M_{p,q}^{(0)} = E_{p,q}$ and $\mathsf{M}_{p,q}^{(0)} = \mathsf{E}_{p,q}$. A path which goes from p to q without visiting intermediate states of rank greater than k is:

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

(b) or the concatenation:
- of a path from p to k without passing through states of rank greater than $k-1$;
- followed by an arbitrary number of paths which go from k to k without passing through intermediate states of rank greater than $k-1$;
- followed finally by a path from k to q without passing through intermediate states of rank greater than $k-1$.

This decomposition is sketched in Figure 5 and implies that for all p and q in Q, for all $k \leqslant n$, it holds:

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

The algorithm ends with the last equation:

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

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 it holds:

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

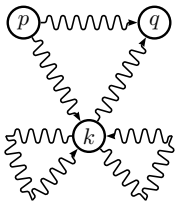


Figure 5. Step k of MN-Y algorithm

Example 3.1. The MN-Y algorithm applied to the automaton $\mathcal{R}_1$ of Figure 6 yields the following matrices (we group together, for each k, the four $\mathsf{M}_{p,q}^{(k)}$ into a matrix $\mathsf{M}^{(k)}$):

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

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

As in the first two methods, identities in $\mathbf{N}$ (as well as $\mathbf{I}$ and $\mathbf{J}$) are likely to be used at any step of the MN-Y algorithm. What is new is that identities $\mathbf{D}$ and $\mathbf{U}$ are particularly fit for the computations involved in the MN-Y algorithm. For instance, after using these identities, the above matrices become:

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

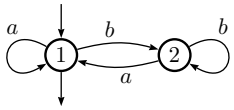


Figure 6. The automaton $\mathcal{R}_1$

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 the one hand, a $Q\times Q$-matrix obtained by successive transformations and on the other hand, an expression obtained by repeated modifications of an automaton, hence of a matrix, but one whose size decreases at each step. This leads us to a more detailed statement.

Proposition 3.8 ([54]). *Let $\mathcal{A} = \langle\, Q, A, E, I, T \,\rangle$ be an automaton and for every p and q in Q, let $\mathcal{A}_{p,q}$ be the automaton defined by $\mathcal{A}_{p,q} = \langle\, Q, A, E, \{p\}, \{q\} \,\rangle$. For every (total) order ω on Q and every p and q in Q, it holds:*

$$\mathbf{N} \wedge \mathbf{U} \;\vdash\; \mathbf{M}_\omega(\mathcal{A}_{p,q}) \;\equiv\; \mathbf{B}_\omega(\mathcal{A}_{p,q}) \;.$$

Proof. In the following, $\mathcal{A}$ and ω are fixed and remain implicit. The automaton $\mathcal{A}$ has n states, identified with the integers from 1 to n; the two algorithms perform n steps starting in a situation called 'step 0', the kth step of the state-elimination method consisting of the removal of state k, and that of algorithmMN-Y consisting of calculating the labels of paths that do not include nodes (strictly) greater than k. We write:

$$\mathsf{E}^{(k)}(r, s)$$

for the label of the transition from r to s in the automaton obtained from $\mathcal{A}$ (and ω) at the kth step of the state elimination method; necessarily, in this notation, $k+1 \leqslant r$ and $k+1 \leqslant s$ (abbreviated to $k+1 \leqslant r, s$). As above, we write:

$$\mathsf{M}^{(k)}_{r,s}$$

for the entry r, s of the $n\times n$ matrix computed by the kth step of MN-Y algorithm. At step 0, the automaton $\mathcal{A}$ has not been modified and we have:

$$\forall r, s\,,\ 1 \leqslant r, s \leqslant n \qquad \mathsf{M}^{(0)}_{r,s} = \mathsf{E}^{(0)}(r, s)\,, \tag{3.9}$$

which will be the base case of the inductions to come. The MN-Y algorithm is written:

$$\forall k\,,\ 0 < k \leqslant n\,,\ \forall r, s\,,\ 1 \leqslant r, s \leqslant n$$
$$\mathsf{M}^{(k)}_{r,s} = \mathsf{M}^{(k-1)}_{r,s} + \mathsf{M}^{(k-1)}_{r,k} \cdot \left(\mathsf{M}^{(k-1)}_{k,k}\right)^* \cdot \mathsf{M}^{(k-1)}_{k,s} \;. \tag{3.10}$$

The state-elimination algorithm is written:

$$\forall k\,,\ 0 < k \leqslant n\,,\ \forall r, s\,,\ k < r, s \leqslant n$$
$$\mathsf{E}^{(k)}(r,s) = \mathsf{E}^{(k-1)}(r,s) + \mathsf{E}^{(k-1)}(r,k) \cdot \Big(\mathsf{E}^{(k-1)}(k,k)\Big)^* \cdot \mathsf{E}^{(k-1)}(k,s) \quad (3.11)$$

Hence we conclude, for given r and s and by induction on k:

$$\forall r, s\,,\ 1 \leqslant r, s \leqslant n\,,\ \forall k\,,\ 0 \leqslant k < \mathsf{min}(r,s) \qquad \mathsf{M}^{(k)}_{r,s} = \mathsf{E}^{(k)}(r,s) \quad (3.12)$$

We see in fact (as there is even so something to see) that if $k < \mathsf{min}(r,s)$ then all integer triples (l,u,v) such that $\mathsf{M}^{(l)}_{u,v}$ occurs in the computation of $\mathsf{M}^{(k)}_{r,s}$ by the (recursive) use of (3.10), are such that $l < \mathsf{min}(u,v)$.

Suppose now that we have p and q, also fixed, such that $1 \leqslant p < q \leqslant n$ (the other cases are dealt with similarly). We call the initial and final states added to $\mathcal{A}$ in the first phase of the state-elimination method i and t respectively; i and t are not integers between 1 and n. The transition from i to p and that from q to t are labelled 1_{A^*}. Now let us consider step p of each algorithm. For every state s, $p < s$, $\mathsf{M}^{(p)}_{p,s}$ is given by (3.10):

$$\mathsf{M}^{(p)}_{p,s} = \mathsf{M}^{(p-1)}_{p,s} + \mathsf{M}^{(p-1)}_{p,p} \cdot \Big(\mathsf{M}^{(p-1)}_{p,p}\Big)^* \cdot \mathsf{M}^{(p-1)}_{p,s}$$

and $\mathsf{E}^{(p)}(i,s)$ by:

$$\mathsf{E}^{(p)}(i,s) = \Big(\mathsf{E}^{(p-1)}(p,p)\Big)^* \cdot \mathsf{E}^{(p-1)}(p,s)$$

and hence, by (3.12):

$$\forall s\,,\ p < s \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(p)}_{p,s} \equiv \mathsf{E}^{(p)}(i,s)\ . \quad (3.13)$$

Next we consider the steps following p (and row p of the matrices $\mathsf{M}^{(k)}$). For all k, $p < k$, and all s, $k < s \leqslant n$, $\mathsf{M}^{(k)}_{p,s}$ is always computed by (3.10) and $\mathsf{E}^{(k)}(i,s)$ by:

$$\mathsf{E}^{(k)}(i,s) = \mathsf{E}^{(k-1)}(i,s) + \mathsf{E}^{(k-1)}(i,k) \cdot \Big(\mathsf{E}^{(k-1)}(k,k)\Big)^* \cdot \mathsf{E}^{(k-1)}(k,s)\ . \quad (3.14)$$

From (3.13), and based on an observation analogous to the previous one, we conclude from the term-by-term correspondence of (3.10) and (3.14) that:

$$\forall k\,,\ p < k\,,\ \forall s\,,\ p < s \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(k)}_{p,s} \equiv \mathsf{E}^{(k)}(i,s)\ . \quad (3.15)$$

The analysis of step q gives a similar, and symmetric, result to that which we have just obtained from the analysis of step p: for all r, $q < r$, we have:

$$\mathsf{M}^{(q)}_{r,q} = \mathsf{M}^{(q-1)}_{r,q} + \mathsf{M}^{(q-1)}_{r,q} \cdot \Big(\mathsf{M}^{(q-1)}_{q,q}\Big)^* \cdot \mathsf{M}^{(q-1)}_{q,q}$$

and
$$\mathsf{E}^{(q)}(r,t) = \mathsf{E}^{(q-1)}(r,q) \cdot \Big(\mathsf{E}^{(q-1)}(q,q)\Big)^*$$

and hence

$$\forall r\,,\ q < r \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(q)}_{r,q} \equiv \mathsf{E}^{(q)}(r,t)\ . \quad (3.16)$$

The steps following q give rise to an equation symmetric to (3.15) (for column q of the

matrices $\mathsf{M}^{(k)}$):

$$\forall k\,,\ q<k\,,\ \forall r\,,\ q<r\leqslant n \qquad \mathbf{U}\vdash \qquad \mathsf{M}^{(k)}_{r,q}\equiv \mathsf{E}^{(k)}(r,t)\ . \tag{3.17}$$

Finally, from:

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

and $$\mathsf{E}^{(k)}(i,t)=\mathsf{E}^{(k-1)}(i,t)+\mathsf{E}^{(k-1)}(i,k)\cdot\left(\mathsf{E}^{(k-1)}(k,k)\right)^*\cdot\mathsf{E}^{(k-1)}(k,t)$$

Equations (3.12), (3.15) and (3.17) together allow us to conclude, by induction on k, that:

$$\forall k\,,\ q\leqslant k\leqslant n \qquad \mathbf{U}\vdash \qquad \mathsf{M}^{(k)}_{p,q}\equiv \mathsf{E}^{(k)}(i,t)\ . \tag{3.18}$$

When we reach $k=n$ in this equation we obtain the identity we want. □

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,*

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

3.5 The recursive method

The last method we want to quote appeared first in Conway's book *Regular Algebra and Finite Machines* [18] which gave a new start to the formal study of rational expressions (*cf.* Chapter 20). Originally, it yields a proof of Proposition 3.1(the entries of E^* belong to the rational closure of the entries of E). 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 The recursive method (for computing E', our goal) is based on computation on matrices via *bloc decomposition*.

Let M and M' be two $Q\times Q$-matrices (over any semiring indeed) and let Q be the disjoint union of R and S. Let us write their bloc decomposition according to $Q=R\cup S$ as:

$$M=\begin{pmatrix}F & G\\ H & K\end{pmatrix}\qquad\qquad M'=\begin{pmatrix}F' & G'\\ H' & K'\end{pmatrix}$$

where F and F' are $R\times R$-matrices, K and K' are $S\times S$-matrices, G and G' $R\times S$-matrices, and H and H' $S\times R$-matrices. The bloc decomposition is consistent with the matirx operations in the sense that we have:

$$M+M'=\begin{pmatrix}F+F' & G+G'\\ H+H' & K+K'\end{pmatrix}\qquad\text{and}$$

$$M\cdot M'=\begin{pmatrix}F\cdot F'+G\cdot H' & F\cdot G'+G\cdot K'\\ H\cdot F'+K\cdot H' & H\cdot G'+K\cdot K'\end{pmatrix}$$

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)[8], 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 other equations:

$$U = 1 + |\mathsf{F}|U + |\mathsf{G}|W, \qquad Z = 1 + |\mathsf{H}|V + |\mathsf{K}|Z, \tag{3.19}$$

$$V = |\mathsf{F}|V + |\mathsf{G}|Z, \quad \text{and} \quad W = |\mathsf{H}|U + |\mathsf{K}|W. \tag{3.20}$$

Corollary 3.3 applies to (3.20) and then, after substitution, to (3.19) gives:

$$V = |\mathsf{F}|^*|\mathsf{G}|Z \quad \text{and} \quad W = |\mathsf{K}|^*|\mathsf{H}|U.$$

$$U = (|\mathsf{F}| + |\mathsf{G}||\mathsf{K}|^*|\mathsf{H}|)^* \quad \text{and} \quad Z = (|\mathsf{K}| + |\mathsf{H}||\mathsf{F}|^*|\mathsf{G}|)^*.$$

This procedure leads to the computation of E^* by induction on its dimension. 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 $\mathsf{Rat}\mathsf{E}\, A^*$.

Example 3.2. The recursive method applied to the automaton $\mathcal{R}_1$ of Example 3.1 (*cf.* Figure 6) directly gives (there is no choice for the recursive division):

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

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 an easy inference of teh equivalence 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:*

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

More generally, and as a conclusion of the description of these four methods, one would conjecture that *the rational expressions computed from a same finite automaton*

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

are all equivalent modulo the natural identities and the aperiodic ones $\mathbf{S}$ *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

The purpose of this last subsection is to present a refinement of Kleene's Theorem — or, rather, of the Fundamental Theorem of Finite Automata — which relates even more closely than above an automaton and the rational expressions that are computed from it.

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 finding the degree of nestedness 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 'entangled'. The intuitive idea of entanglement of circuits will be captured by the notion of *loop complexity*. We show how 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 ([21]).

Star height of an expression Let E be an expression over A^*. The *star height* of E, denoted by $\mathsf{h}[\mathsf{E}]$, is inductively defined by

$$\begin{aligned}
&\text{if} \quad \mathsf{E} = 0\,,\ \mathsf{E} = 1 \ \text{ or } \ \mathsf{E} = a \in A\,, && \mathsf{h}[\mathsf{E}] = 0\ ,\\
&\text{if} \quad \mathsf{E} = \mathsf{E}' + \mathsf{E}'' \ \text{ or } \ \mathsf{E} = \mathsf{E}' \cdot \mathsf{E}''\,, && \mathsf{h}[\mathsf{E}] = \mathsf{max}(\mathsf{h}[\mathsf{E}'], \mathsf{h}[\mathsf{E}''])\ ,\\
&\text{if} \quad \mathsf{E} = \mathsf{F}^*\,, && \mathsf{h}[\mathsf{E}] = 1 + \mathsf{h}[\mathsf{F}]\ .
\end{aligned}$$

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

These examples draw attention to the fact that two equivalent expressions may have different star heights and that star height is unrelated to the length. They also naturally give rise to the so-called *star-height problem*. As it does not directly pertain to the matter developed in this chapter, we postpone the few indications we give on this problem to the Notes section ((see also Chapter 5).

Loop complexity of an automaton Let $\mathcal{A} = \langle\, Q, A, E, I, T \,\rangle$ be an automaton; we call *balls*[9] the strongly connected components of $\mathcal{A}$ that contain at least one transition. In other words, a strongly connected component that contains at least two states is a ball, and a strongly connected component reduced to a single state s is a ball if and only if s is the source (and the destination) of at least one loop. Balls are pairwise disjoint but do not form a covering (hence a partition) of Q since a state may belong to no ball (*cf.* Figure 7).

[9] Translation of the French: *pelote*.

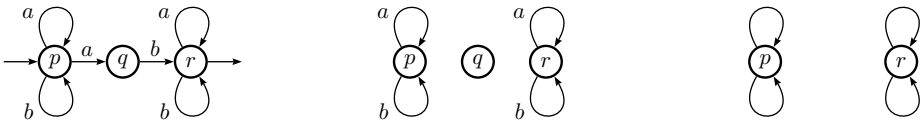
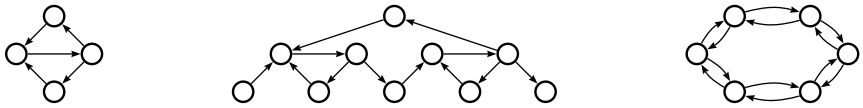


Figure 7. An automaton, its strongly connected components and its two balls

Definition 3.1. The *loop complexity* of an automaton $\mathcal{A}$ is the integer $\mathsf{lc}(\mathcal{A})$ defined inductively by the following equations:

$\mathsf{lc}(\mathcal{A}) = 0$ if $\mathcal{A}$ contains no balls (in particular if $\mathcal{A}$ is empty);
$\mathsf{lc}(\mathcal{A}) = \mathsf{max}\,\{\mathsf{lc}(\mathcal{P}) \mid \mathcal{P}$ a ball in $\mathcal{A}\}$ if $\mathcal{A}$ is not strongly connected;
$\mathsf{lc}(\mathcal{A}) = 1 + \mathsf{min}\,\{\mathsf{lc}(\mathcal{A} \setminus \{s\}) \mid s$ state of $\mathcal{A}\}$ if $\mathcal{A}$ is strongly connected.

Figure 8 shows automata with loop complexity 1, 2 and 3 respectively.



(a) $\mathsf{lc}(\mathcal{A}) = 1$ (b) $\mathsf{lc}(\mathcal{A}) = 2$ (c) $\mathsf{lc}(\mathcal{A}) = 3$

Figure 8. Automata with differents loop complexities

Eggan's Theorem We are now ready to state the announced refinement of the Fundamental theorem of finite automata.

Theorem 3.11 (Eggan [21]). *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*.

Let $\mathcal{A} = \langle Q, A, E, I, T \rangle$ be an automaton. If ω is an order on Q, we write $\overline{\omega}$ for the *greatest element* of Q according to ω. If $\mathcal{S}$ is a sub-automaton of $\mathcal{A}$, we also write ω for the trace of the order ω over the set R of states of $\mathcal{S}$ and, in such a context, $\overline{\omega}$ for the greatest element of 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
$$\mathfrak{I}_\omega(\mathcal{A}) = 0 \ ; \tag{3.21}$$
- if $\mathcal{A}$ is not itself a ball, then
$$\mathfrak{I}_\omega(\mathcal{A}) = \mathsf{max}\,(\{\mathfrak{I}_\omega(\mathcal{P}) \mid \mathcal{P} \text{ ball in } \mathcal{A}\}) \ ; \tag{3.22}$$
- if $\mathcal{A}$ is a ball, then
$$\mathfrak{I}_\omega(\mathcal{A}) = 1 + \mathfrak{I}_\omega(\mathcal{A} \setminus \overline{\omega}) \ . \tag{3.23}$$

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

Property 3.12. $\mathsf{lc}(\mathcal{A}) = \min\{\mathfrak{I}_\omega(\mathcal{A}) \mid \omega \text{ is an order on } Q\}$. □

Theorem 3.11 is then a consequence of the following.

Proposition 3.13 ([38]). *For any order ω on the states of $\mathcal{A}$,* $\mathfrak{I}_\omega(\mathcal{A}) = \mathsf{h}[\mathbf{B}_\omega(\mathcal{A})]$.

At this point, let us note that in the inductive definition of $\mathfrak{I}_\omega(\mathcal{A})$ it is the *greatest* element $\overline{\omega}$ of Q that is considered whereas in the construction of the expression $\mathbf{B}_\omega(\mathcal{A})$ it is the *smallest* element of Q suppressed first.

In the course of the computation of $\mathbf{B}_\omega(\mathcal{A})$ by the state elimination method we consider automata whose transitions are labelled not by letters only but by rational expressions in general. In order to define the index of such generalised automata, we first define the *index*, written $\mathsf{i}(e)$, of a transition e as the star height of the label of e:

$$\mathsf{i}(e) = \mathsf{h}[|e|] \ .$$

The index of a generalised automaton is then defined inductively, by formulas that take into account the index of every transition:

- if $\mathcal{A}$ is empty, then
$$\mathfrak{I}_\omega(\mathcal{A}) = 0 \ ; \tag{3.24}$$
- if $\mathcal{A}$ is not itself a ball, then
$$\begin{aligned}\mathfrak{I}_\omega(\mathcal{A}) = \mathsf{max}\,(\{\mathsf{i}(e) \mid e \text{ does not belong to a ball in } \mathcal{A}\} \\ \cup \{\mathfrak{I}_\omega(\mathcal{P}) \mid \mathcal{P} \text{ ball in } \mathcal{A}\}) \ ;\end{aligned} \tag{3.25}$$
- if $\mathcal{A}$ is a ball, then
$$\mathfrak{I}_\omega(\mathcal{A}) = 1 + \mathsf{max}\,(\{\mathsf{i}(e) \mid e \text{ is adjacent to } \overline{\omega}\}, \mathfrak{I}_\omega(\mathcal{A} \setminus \overline{\omega})) \ . \tag{3.26}$$

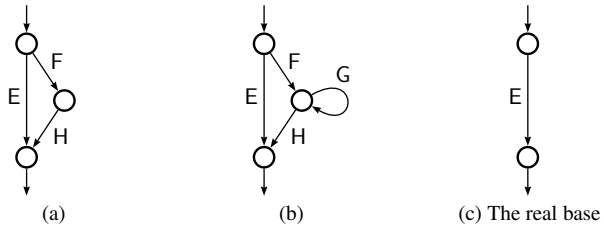
It is obvious that equations (3.24)–(3.26) reduce to (3.22)–(3.21) in the case of a 'classic' automaton whose transitions are all labelled with letters, that is, have index 0. Figure 9 shows two generalised automata and their index.

(a) $\mathfrak{I}_{\omega_1}(\mathcal{B}_1) = 2; \mathfrak{I}_{\omega_2}(\mathcal{B}_1) = 3$ (b) $\mathfrak{I}_{\omega_1}(\mathcal{B}_2)\,\mathfrak{I}_{\omega_2}(\mathcal{B}_2) = 3$

Figure 9. Computation of the index of two automata for the two possible orders on the states: $\omega_1 = p < q$, $\omega_2 = q < p$.

Proof of Proposition 3.13. We proceed by induction on the number of states of $\mathcal{A}$. The state-elimination method consists in the first place of transforming $\mathcal{A}$ into an automaton $\mathcal{B}$ by adding two states to $\mathcal{A}$, an initial state and a final state, and transitions which are all labelled by the empty word, the index of $\mathcal{B}$ being equal to that of $\mathcal{A}$. By convention, the added states are greater than all the other states of $\mathcal{A}$ in the order ω and are never removed by the state-elimination method. On the other hand, the transition labels, including those of the new transitions, may be modified in the course of the state-elimination method.

The base case of the induction is therefore a generalised automaton with 3 states of the form of Figure 10 (a) or (b).



(a) (b) (c) The real base

Figure 10. Base case of the induction

In case (a), $\mathcal{B}$ contains no balls and we have

$$\mathfrak{I}_\omega(\mathcal{B}) = \mathsf{max}\big(\mathsf{h}[\mathsf{E}], \mathsf{h}[\mathsf{F}], \mathsf{h}[\mathsf{H}]\big) = \mathsf{h}[\mathsf{E} + \mathsf{F} \cdot \mathsf{H}] = \mathsf{h}[\mathbf{B}_\omega(\mathcal{B})] \ . \tag{3.27}$$

In case (b), the unique state of $\mathcal{B}$ that is neither initial nor final is a ball whose index is $1 + \mathsf{h}[\mathsf{G}]$, and we have

$$\mathfrak{I}_\omega(\mathcal{B}) = \mathsf{max}\big(\mathsf{h}[\mathsf{E}], \mathsf{h}[\mathsf{F}], \mathsf{h}[\mathsf{H}], (1 + \mathsf{h}[\mathsf{G}])\big) = \mathsf{h}[\mathsf{E} + \mathsf{F} \cdot \mathsf{G}^* \cdot \mathsf{H}] = \mathsf{h}[\mathbf{B}_\omega(\mathcal{B})] \ . \tag{3.28}$$

Note that this reasoning is the essential part of the induction step and that for rigour, if not for clarity, we could have taken the automaton of Figure 10(c) as our base case, for which the statement is even more easily verified (and which corresponds to an automaton $\mathcal{A}$ with no state).

Now let $\mathcal{B}$ be an automaton of the prescribed form with $n + 2$ states, q the smallest state in the order ω and $\mathcal{B}'$ the automaton which results from the first step of the state-elimination method applied to $\mathcal{B}$ (which consists of the elimination of q). Because the same states (other than q) are adjacent in $\mathcal{B}$ as in $\mathcal{B}'$, and since q is the *smallest element* in the order ω, the algorithm for computing the index runs the same way in $\mathcal{B}$ and $\mathcal{B}'$, that is, the succession of balls constructed in each automaton is identical, excluding the examination of q in $\mathcal{B}$. It remains to show that the values calculated are also identical.

Let $\mathcal{P}$ be the smallest ball in $\mathcal{B}$ that strictly contains q – if no such ball exists, take $\mathcal{P} = \mathcal{B}$; and let $\mathcal{P}'$ the 'image' of $\mathcal{P}$ in $\mathcal{B}'$. There are two possible cases: either (a) q is not the source (and the destination) of a loop, or (b) it is, that is, q is a ball in $\mathcal{B}$ all by itself, and the label of this loop is an expression G.

The transitions of $\mathcal{P}'$ are either identical to those of $\mathcal{P}$, or, in case (a), labelled by products $\mathsf{F} \cdot \mathsf{H}$, where F and H are labels of transitions of $\mathcal{P}$, or, in case (b), labelled by

products $\mathsf{F}\cdot\mathsf{G}^*\cdot\mathsf{H}$. It therefore follows that, in case (a)

$$\begin{aligned}\mathfrak{I}_\omega(\mathcal{P}') &= \mathsf{max}\big(\mathsf{max}\,\{\mathsf{i}(e)\mid e \text{ does not belong to a ball in } \mathcal{P}'\}\,,\\ &\qquad\qquad \mathsf{max}\,\{\mathfrak{I}_\omega(\mathcal{Q})\mid \mathcal{Q} \text{ ball in } \mathcal{P}'\}\big)\\ &= \mathsf{max}\big(\mathsf{max}\,\{\mathsf{i}(e)\mid e \text{ does not belong to a ball in } \mathcal{P}\}\,,\\ &\qquad\qquad \mathsf{max}\,\{\mathfrak{I}_\omega(\mathcal{Q})\mid \mathcal{Q} \text{ ball in } \mathcal{P}\}\big)\\ &= \mathfrak{I}_\omega(\mathcal{P})\quad . \end{aligned}\tag{3.29}$$

In case (b), since $\mathfrak{I}_\omega(\{q\}) = 1+\mathsf{h}[\mathsf{G}]$, we have:

$$\begin{aligned}\mathfrak{I}_\omega(\mathcal{P}') &= \mathsf{max}\big(\,\mathsf{max}\,\{\mathsf{i}(e)\mid e \text{ does not belong to a ball in } \mathcal{P}'\}\,,\\ &\qquad\qquad \mathsf{max}\,\{\mathfrak{I}_\omega(\mathcal{Q})\mid \mathcal{Q} \text{ ball in } \mathcal{P}'\}\,\big)\\ &= \mathsf{max}\big(\,\mathsf{max}\,\{\mathsf{i}(e)\mid e \text{ does not belong to a ball in } \mathcal{P}\}\,,\ (1+\mathsf{h}[\mathsf{G}]),\\ &\qquad\qquad \mathsf{max}\,\{\mathfrak{I}_\omega(\mathcal{Q})\mid \mathcal{Q} \text{ ball in } \mathcal{P}'\}\,\big)\\ &= \mathsf{max}\big(\,\mathsf{max}\,\{\mathsf{i}(e)\mid e \text{ does not belong to a ball in } \mathcal{P}\}\,,\ \mathfrak{I}_\omega(\{q\})\,,\\ &\qquad\qquad \mathsf{max}\,\{\mathfrak{I}_\omega(\mathcal{Q})\mid \mathcal{Q} \text{ ball in } \mathcal{P}, \text{different from } \{q\}\}\,\big)\\ &= \mathfrak{I}_\omega(\mathcal{P})\quad . \end{aligned}\tag{3.29$'$}$$

If $\mathcal{P}=\mathcal{B}$ (and $\mathcal{P}'=\mathcal{B}'$), the equalities (3.29) and (3.29)$'$ become

$$\mathfrak{I}_\omega(\mathcal{B}') = \mathfrak{I}_\omega(\mathcal{B})\tag{3.30}$$

which proves the induction and hence the proposition. If not, and without starting an induction on the number of overlapping balls that contain $\{q\}$, we can get from (3.29) to (3.30) by noting that the transitions of $\mathcal{B}'$ are either identical to those of $\mathcal{B}$, or correspond to transitions that are adjacent to q.

In case (a), the labels of these transitions (those corresponding to transitions adjacent to q) are products of transitions of $\mathcal{B}$: their index is obtained by taking a maximum, and (3.30) is the result of the identity $\mathsf{max}(a,b,c)=\mathsf{max}(a,\mathsf{max}(b,c))$.

In case (b), the labels of the same transitions are, as before, of the form $\mathsf{F}\cdot\mathsf{G}^*\cdot\mathsf{H}$, with index $\mathsf{max}(\mathsf{h}[\mathsf{F}],\mathsf{h}[\mathsf{H}],1+\mathsf{h}[\mathsf{G}])$. The corresponding transition in $\mathcal{B}$ has label F (or H); it is inspected in the algorithm for computing the index when the indices of the transition with label H (or F) and that of the *ball* $\{q\}$, with index $1+\mathsf{h}[\mathsf{G}]$, have already been taken into account. The result, which is (3.30), follows for the same reason as above. □

Theorem 3.11 admits a kind of converse stated in the following proposition whose proof is postponed to the next section where we build automata from expressions.

Proposition 3.14. *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 Δ-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 might be mapped to several automata, each of them being computed in different ways. We distinguish the objects themselves, that is, the computed automata, which we try to characterise as intrinsically as possible, from the algorithms that allow to compute them. We 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 (Section 2.2) as graphs labelled by *letters* of an alphabet. It is known that the family of languages accepted by finite automata is not enlarged 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) $\forall (p,a,q) \in E \quad \big(\varphi(p), a, \varphi(q)\big) \in E'$.

The automaton $\mathcal{A}'$ is *a quotient* of $\mathcal{A}$ if, moreover,

(iv) $\varphi(Q) = Q'$, that is, φ is *surjective*,
(v) $\varphi(I) = I'$,
(vi) $\varphi^{-1}(T') = T$,
(vii) $\forall (r,a,s) \in E'\,,\ \forall p \in \varphi^{-1}(r)\,,\ \exists q \in \varphi^{-1}(s) \quad (p,a,q) \in E$.

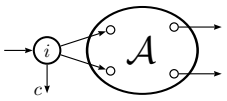
If φ is a morphism, we write $\varphi\colon \mathcal{A} \to \mathcal{A}'$, and the inclusion $|\mathcal{A}| \subseteq |\mathcal{A}'|$ holds. If $\mathcal{A}'$ is a quotient of $\mathcal{A}$, then $|\mathcal{A}| = |\mathcal{A}'|$ holds.

Definition 4.1 generalises the classical notion of quotient of complete deterministic automata to arbitrary automata. Every automaton $\mathcal{A}$ admits a *minimal quotient*, which is a quotient of every quotient of $\mathcal{A}$. In contrast with the case of deterministic automata, 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 plays a central role in our presentation, has first been defined by Glushkov (in [29]). For the same purpose, McNaughton and Yamada computed the determinisation of $\mathcal{S}_{\mathsf{E}}$ in their paper [43] that we already quoted. 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 of that class.

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 11 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 $|\mathcal{A}|$.



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

Figure 11. A standard automaton

Every automaton is equivalent to a standard one. More important for our purpose, their special form allows to define *operations* on standard automata that are parallel to the *rational operations*. Let $\mathcal{A}$ (as in Figure 11) and $\mathcal{B}$ (with obvious notation) be two standard automata; the following standard automata are defined:

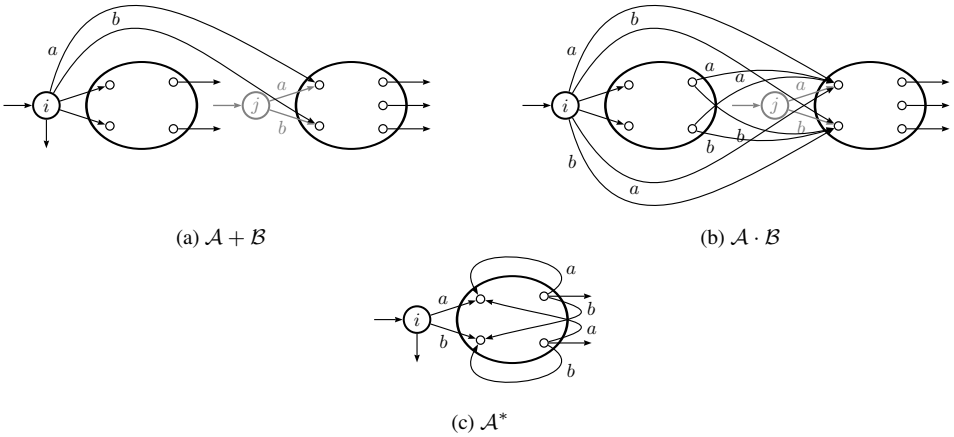
$$\mathcal{A} + \mathcal{B} = \left\langle \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & J & K \\ 0 & F & 0 \\ 0 & 0 & G \end{pmatrix}, \begin{pmatrix} c+d \\ U \\ V \end{pmatrix} \right\rangle, \tag{4.1}$$

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

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

where $H = U \cdot J + F$. The use of the constants c and d allows a uniform treatment of the cases whether the initial states of $\mathcal{A}$ and $\mathcal{B}$ are final or not.

These constructions are shown at Figure 12.



(a) $\mathcal{A} + \mathcal{B}$

(b) $\mathcal{A} \cdot \mathcal{B}$

(c) $\mathcal{A}^*$

Figure 12. Operations on standard automata

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 induction and the computations show that the map $\mathsf{E} \mapsto \mathcal{S}_\mathsf{E}$ is a Δ-map:

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

The inductive construction of $\mathcal{S}_\mathsf{E}$ also implies:

Property 4.2. If E is a rational expression, the dimension of $\mathcal{S}_\mathsf{E}$ is $\ell(\mathsf{E}) + 1$.

Example 4.1. Figure 13 shows $\mathcal{S}_{\mathsf{E}_1}$, where $\mathsf{E}_1 = (a^*b + bb^*a^*)^*$.

The example of $\mathsf{E} = \left(\left((a^* + b^*)^* + c^*\right)^* + d^*\right)^* \ldots$ 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 leads to a construction that is interesting *per se*.

4.2.2 The star-normal form of an expression The *star-normal form* of an expression has been defined by Brüggemann-Klein (in [11]) in order to design a quadratic algorithm

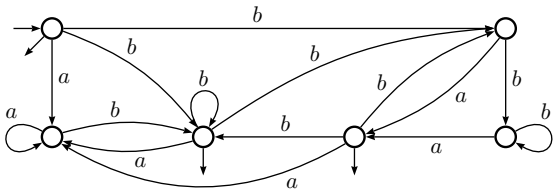


Figure 13. The automaton $\mathcal{S}_{\mathsf{E}_1}$.

for the computation of the standard automaton of an expression. The interest of this notion certainly goes beyond that complexity improvement.

Definition 4.2 ([11]). 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, $\mathsf{c}(\mathsf{F}) = 0$.[10]

Two operators on expressions, written $\bullet$ and $\square$, are defined by a mutual recursion on the depth of the expression that defines and allows to compute *the star-normal form* of the expression.

$$0^{\square} = 0\,, \quad 1^{\square} = 0\,, \quad \forall a \in A \quad a^{\square} = a\ , \tag{4.4}$$

$$(\mathsf{F} + \mathsf{G})^{\square} = \mathsf{F}^{\square} + \mathsf{G}^{\square}\ , \tag{4.5}$$

$$(\mathsf{F} \cdot \mathsf{G})^{\square} = \begin{cases} \mathsf{F}^{\square} + \mathsf{G}^{\square} & \text{if} \quad \mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 1\ , \\ \mathsf{F}^{\bullet} \cdot \mathsf{G}^{\bullet} & \text{otherwise}\ , \end{cases} \tag{4.6}$$

$$(\mathsf{F}^*)^{\square} = \mathsf{F}^{\square}\ . \tag{4.7}$$

$$0^{\bullet} = 0\,, \quad 1^{\bullet} = 1\,, \quad \forall a \in A \quad a^{\bullet} = a\ , \tag{4.8}$$

$$(\mathsf{F} + \mathsf{G})^{\bullet} = \mathsf{F}^{\bullet} + \mathsf{G}^{\bullet}\ , \tag{4.9}$$

$$(\mathsf{F} \cdot \mathsf{G})^{\bullet} = \mathsf{F}^{\bullet} \cdot \mathsf{G}^{\bullet}\ , \tag{4.10}$$

$$(\mathsf{F}^*)^{\bullet} = (\mathsf{F}^{\square})^*\ . \tag{4.11}$$

Example 4.2. Let $\mathsf{E}_2 = (a^*b^*)^*$. Then
$\mathsf{E}_2{}^{\bullet} = \left((a^*b^*)^{\square}\right)^* = \left((a^*)^{\square} + (b^*)^{\square}\right)^* = \left((a)^{\square} + (b)^{\square}\right)^* = (a+b)^*$.

Theorem 4.3 ([11]). *For any expression* E, $\mathsf{E}^{\bullet}$ *is in star-normal form and* $\mathcal{S}_{\mathsf{E}^{\bullet}} = \mathcal{S}_{\mathsf{E}}$.

Theorem 4.3 implies in particular that $\mathsf{E}^{\bullet}$ is equivalent to E. It relies on three computations on Boolean standard automata which are the direct consequence of the formulas (4.1)–(4.3). and of the idempotency identity **I**.

Property 4.4. Let $\mathcal{A}$, $\mathcal{A}'$, $\mathcal{B}$ and $\mathcal{B}'$ be four (Boolean) standard automata.
(1) If $\mathcal{A}^* = \mathcal{A}'^*$ and $\mathcal{B}^* = \mathcal{B}'^*$ then $(\mathcal{A} + \mathcal{B})^* = (\mathcal{A}' + \mathcal{B}')^*$.
(2) If $\mathsf{c}(|\mathcal{A}|) = \mathsf{c}(|\mathcal{B}|) = 1$, then $(\mathcal{A} + \mathcal{B})^* = (\mathcal{A} \cdot \mathcal{B})^*$.
(3) $\mathcal{A}^* = (\mathcal{A}^*)^*$.

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

Proof. (i) Let

$$\mathcal{A} = \left\langle (1 \quad 0), \begin{pmatrix} 0 & J \\ 0 & F \end{pmatrix}, \begin{pmatrix} c \\ U \end{pmatrix} \right\rangle \quad \text{and} \quad \mathcal{A}' = \left\langle (1 \quad 0), \begin{pmatrix} 0 & J' \\ 0 & F' \end{pmatrix}, \begin{pmatrix} c' \\ U' \end{pmatrix} \right\rangle .$$

The hypothesis implies

$$\mathcal{A}^* = \left\langle (1 \quad 0), \begin{pmatrix} 0 & J \\ 0 & U \cdot J + F \end{pmatrix}, \begin{pmatrix} 1 \\ U \end{pmatrix} \right\rangle = \mathcal{A}'^* = \left\langle (1 \quad 0), \begin{pmatrix} 0 & J' \\ 0 & U' \cdot J' + F' \end{pmatrix}, \begin{pmatrix} 1 \\ U' \end{pmatrix} \right\rangle ,$$

hence $J = J'$, $U = U'$ and $U \cdot J + F = U' \cdot J' + F'$. Accordingly, if we have

$$\mathcal{B} = \left\langle (1 \quad 0), \begin{pmatrix} 0 & K \\ 0 & G \end{pmatrix}, \begin{pmatrix} d \\ V \end{pmatrix} \right\rangle \quad \text{and} \quad \mathcal{B}' = \left\langle (1 \quad 0), \begin{pmatrix} 0 & K' \\ 0 & G' \end{pmatrix}, \begin{pmatrix} d' \\ V \end{pmatrix} \right\rangle ,$$

then $K = K'$, $V = V'$ and $V \cdot K + G = V' \cdot K' + G'$ hold. From (4.1) and (4.3) follow

$$\begin{aligned} (\mathcal{A} + \mathcal{B})^* &= \left\langle (1 \quad 0 \quad 0), \begin{pmatrix} 0 & J & K \\ 0 & U \cdot J + F & U \cdot K \\ 0 & V \cdot K & V \cdot K + G \end{pmatrix}, \begin{pmatrix} 1 \\ U \\ V \end{pmatrix} \right\rangle \\ &= \left\langle (1 \quad 0 \quad 0), \begin{pmatrix} 0 & J' & K' \\ 0 & U' \cdot J' + F' & U' \cdot K' \\ 0 & V' \cdot K' & V' \cdot K' + G' \end{pmatrix}, \begin{pmatrix} 1 \\ U' \\ V' \end{pmatrix} \right\rangle = (\mathcal{A}' + \mathcal{B}')^* . \end{aligned}$$

(ii) With the same notation as before we have on one hand-side

$$\mathcal{A} + \mathcal{B} = \left\langle (1 \quad 0 \quad 0), \begin{pmatrix} 0 & J & K \\ 0 & F & 0 \\ 0 & 0 & G \end{pmatrix}, \begin{pmatrix} 1 \\ U \\ V \end{pmatrix} \right\rangle \qquad \text{and then}$$

$$(\mathcal{A} + \mathcal{B})^* = \left\langle (1 \quad 0 \quad 0), \begin{pmatrix} 0 & J & K \\ 0 & U \cdot J + F & U \cdot K \\ 0 & V \cdot K & V \cdot K + G \end{pmatrix}, \begin{pmatrix} 1 \\ U \\ V \end{pmatrix} \right\rangle ,$$

and on the other

$$\mathcal{A} \cdot \mathcal{B} = \left\langle (1 \quad 0 \quad 0), \begin{pmatrix} 0 & J & K \\ 0 & F & U \cdot K \\ 0 & 0 & G \end{pmatrix}, \begin{pmatrix} 1 \\ U \\ V \end{pmatrix} \right\rangle \qquad \text{and then}$$

$$(\mathcal{A} \cdot \mathcal{B})^* = \left\langle (1 \quad 0 \quad 0), \begin{pmatrix} 0 & J & K \\ 0 & U \cdot J + F & U \cdot K + U \cdot K \\ 0 & V \cdot K & V \cdot K + G \end{pmatrix}, \begin{pmatrix} 1 \\ U \\ V \end{pmatrix} \right\rangle ,$$

and the equality follows from the idempotency identity **I**.

(iii) Again with the same notation, we have:

$$\mathcal{A}^* = \left\langle (1 \quad 0), \begin{pmatrix} 0 & J \\ 0 & U \cdot J + F \end{pmatrix}, \begin{pmatrix} 1 \\ U \end{pmatrix} \right\rangle \quad \text{and} \quad (\mathcal{A}^*)^* = \left\langle (1 \quad 0), \begin{pmatrix} 0 & J \\ 0 & U \cdot J + U \cdot J + F \end{pmatrix}, \begin{pmatrix} 1 \\ U \end{pmatrix} \right\rangle$$

and the equality follows from the idempotency identity **I**. □

Proof of Theorem 4.3. We establish by a simultaneous induction that the following *two*

statements hold:

$$\mathsf{E}^\bullet \quad \text{is in SNF} \qquad \text{and} \qquad \mathcal{S}_{\mathsf{E}\bullet} = \mathcal{S}_{\mathsf{E}} \tag{4.12}$$
$$\mathsf{E}^\square \quad \text{is in SNF} \qquad \text{and} \qquad (\mathcal{S}_{\mathsf{E}^\square})^* = \mathcal{S}_{\mathsf{E}^*} \ . \tag{4.13}$$

Both (4.12) and (4.13) clearly hold for the base clauses.

- $\mathsf{E} = \mathsf{F} + \mathsf{G}$
  - $\mathsf{E}^\bullet = \mathsf{F}^\bullet + \mathsf{G}^\bullet$ is in star-normal form by induction. Moreover,
$$\begin{aligned} \mathcal{S}_{\mathsf{E}\bullet} = \mathcal{S}_{\mathsf{F}\bullet} + \mathcal{S}_{\mathsf{G}\bullet} &= \mathcal{S}_{\mathsf{F}} + \mathcal{S}_{\mathsf{G}} && \text{by induction} \\ &= \mathcal{S}_{\mathsf{E}} && \text{since } \mathsf{E} = \mathsf{F} + \mathsf{G} \ . \end{aligned}$$
  - $\mathsf{E}^\square = \mathsf{F}^\square + \mathsf{G}^\square$ is in star-normal form by induction. Moreover,
$$\begin{aligned} (\mathcal{S}_{\mathsf{E}^\square})^* = (\mathcal{S}_{\mathsf{F}^\square} + \mathcal{S}_{\mathsf{G}^\square})^* &= (\mathcal{S}_{\mathsf{F}} + \mathcal{S}_{\mathsf{G}})^* && \text{by induction and by Property 4.4 (i)} \\ &= \mathcal{S}_{\mathsf{E}^*} && \text{since } \mathsf{E} = \mathsf{F} + \mathsf{G} \ . \end{aligned}$$
- $\mathsf{E} = \mathsf{F} \cdot \mathsf{G}$
  - $\mathsf{E}^\bullet = \mathsf{F}^\bullet \cdot \mathsf{G}^\bullet$ is in star-normal form by induction. Moreover,
$$\begin{aligned} \mathcal{S}_{\mathsf{E}\bullet} = \mathcal{S}_{\mathsf{F}\bullet} \cdot \mathcal{S}_{\mathsf{G}\bullet} &= \mathcal{S}_{\mathsf{F}} \cdot \mathcal{S}_{\mathsf{G}} && \text{by induction} \\ &= \mathcal{S}_{\mathsf{E}} && \text{since } \mathsf{E} = \mathsf{F} \cdot \mathsf{G} \ . \end{aligned}$$
  - $\mathsf{E}^\square = \mathsf{F}^\square + \mathsf{G}^\square$ or $\mathsf{E}^\square = \mathsf{F}^\bullet \cdot \mathsf{G}^\bullet$ is in star-normal form by induction. Moreover:
    (i) if $\mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 1$, then
$$\begin{aligned} (\mathcal{S}_{\mathsf{E}^\square})^* = (\mathcal{S}_{\mathsf{F}^\square} + \mathcal{S}_{\mathsf{G}^\square})^* &= (\mathcal{S}_{\mathsf{F}} + \mathcal{S}_{\mathsf{G}})^* && \text{by induction and by Property 4.4 (i)} \\ &= (\mathcal{S}_{\mathsf{F}} \cdot \mathcal{S}_{\mathsf{G}})^* && \text{by Property 4.4 (ii)} \\ &= \mathcal{S}_{\mathsf{E}^*} && \text{since } \mathsf{E} = \mathsf{F} \cdot \mathsf{G} \ . \end{aligned}$$
    (ii) if $\mathsf{c}(\mathsf{F})\mathsf{c}(\mathsf{G}) = 0$, then
$$\begin{aligned} (\mathcal{S}_{\mathsf{E}^\square})^* = (\mathcal{S}_{\mathsf{F}\bullet} \cdot \mathcal{S}_{\mathsf{G}\bullet})^* &= (\mathcal{S}_{\mathsf{F}} \cdot \mathcal{S}_{\mathsf{G}})^* && \text{by induction} \\ &= \mathcal{S}_{\mathsf{E}^*} && \text{since } \mathsf{E} = \mathsf{F} \cdot \mathsf{G} \ . \end{aligned}$$
- $\mathsf{E} = \mathsf{F}^*$
  - $\mathsf{E}^\bullet = (\mathsf{F}^\square)^*$ is in star-normal form by induction and since $\mathsf{c}(\mathsf{F}^\square) = 0$. Moreover,
$$\begin{aligned} \mathcal{S}_{\mathsf{E}\bullet} = (\mathcal{S}_{\mathsf{F}^\square})^* = \mathcal{S}_{\mathsf{F}^*} &&& \text{by induction and (4.13)} \\ = \mathcal{S}_{\mathsf{E}} &&& \text{since } \mathsf{E} = \mathsf{F}^* \ . \end{aligned}$$
  - $\mathsf{E}^\square = (\mathsf{F}^*)^\square = \mathsf{F}^\square$ is in star-normal form by induction. Moreover
$$\begin{aligned} (\mathcal{S}_{\mathsf{E}^\square})^* = (\mathcal{S}_{\mathsf{F}^\square})^* = \mathcal{S}_{\mathsf{F}^*} &&& \text{by induction} \\ = \mathcal{S}_{\mathsf{E}^*} &&& \text{since } \mathsf{E} = \mathsf{F} \text{ and by Property 4.4 (iii)} \ . \end{aligned}$$

□

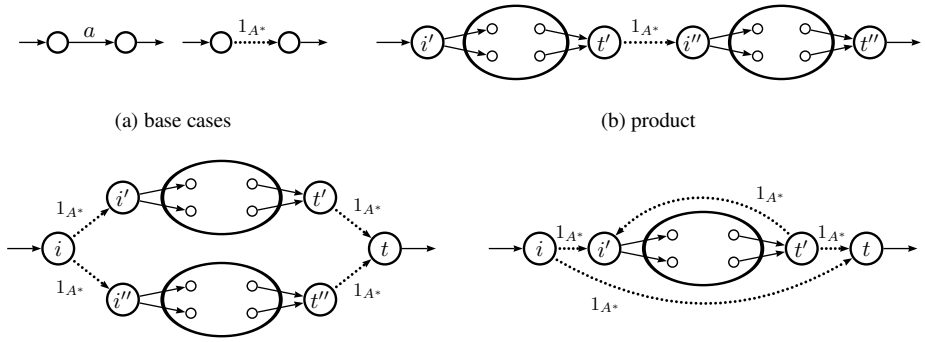
As the computation of $\mathsf{E}^\bullet$ is linear in $\ell(\mathsf{E})$, the goal is achieved by the following:

Theorem 4.5 ([11]). *Let* E *be a rational expression in star-normal form. Then, the inductive computation of* $\mathcal{S}_{\mathsf{E}}$ *by (4.1)–(4.3) has a quadratic complexity in* $\ell(\mathsf{E})$.

Proof. to be done □

4.2.3 The Thompson automaton A survey on Δ-maps cannot miss out the method due to Thompson [58]. It was designed to be directly implementable as a program, primarily for searching with rational expressions in text. It is based on the use of spontaneous transitions. Figure 14 shows the basic steps of the construction, which, by induction, associates with an expression E a unique (and well-defined) automaton $\mathcal{T}_{\mathsf{E}}$. It is remarkable that this construction corresponds indeed to another way of defining the standard automaton.

Proposition 4.6. *The* backward closure *of* $\mathcal{T}_{\mathsf{E}}$ *is equal to* $\mathcal{S}_{\mathsf{E}}$.



(a) base cases (b) product

(c) union (d) star

Figure 14. Thompson's construction

Figure 15 shows the construction applied to the expression $\mathsf{E}_2 = (a^*b + bb^*a)^*$.

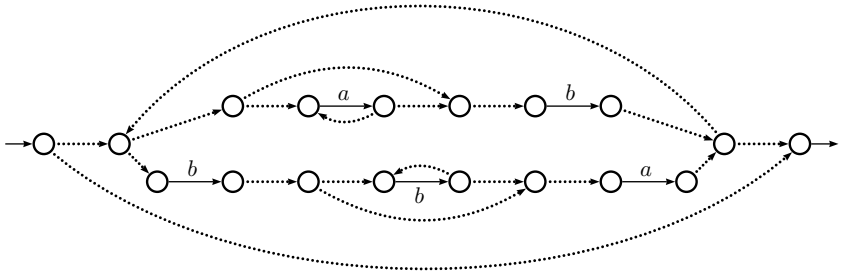


Figure 15. The automaton $\mathcal{T}_{\mathsf{E}_2}$

4.2.4 Loop complexity of standard automata We end this section with the proof of Proposition 3.14: *With every rational expression* E*, we can associate an automaton equivalent to* E *and whose loop complexity is equal to the star height of* E.

The sketches at Figure 12 (a) and (b) make clear that if $\mathcal{A}$ and $\mathcal{A}'$ are standard automata, it holds

$$\mathsf{lc}((\mathcal{A}+\mathcal{A}')) = \mathsf{lc}((\mathcal{A}\cdot\mathcal{A}')) = \mathsf{max}\{\mathsf{lc}(\mathcal{A}), \mathsf{lc}(\mathcal{A}')\} \ .$$

It follows then from Definition 3.1 that the loop complexity of the sum and product of standard automata is equal to the star height of the expressions for the sum and product, provided equality hold for the operands.

The same relation does not holds for the star of a standard automaton, as seen on the example shown at Figure 16: the loop complexity of the automaton is not necessarily incremented by the star operation. In the opposite way, the star operation (on standard automata) may well increase the loop complexity by more than 1, as shown by $\mathcal{S}_{\mathsf{E}_1}$. Hence, it is not true that $\mathsf{lc}(\mathcal{S}_{\mathsf{E}}) = \mathsf{h}[\mathsf{E}]$ holds.

Figure 16. The standard automaton $\mathcal{A}_3$ of $a\,(a^2)^*$ and $\mathcal{A}_3^*$

In order to circumvent this difficulty, we replace the star operation on standard automata by a more elaborate one. A standard automaton is *normalised* if it has only one final state and if this final state is not the origin of any transition. An obvious construction transforms any standard automaton $\mathcal{A}$ into an equivalent normalised one, which we write $\mathcal{A}_{\mathsf{nor}}$, and we have:

Property 4.7. $\mathsf{lc}(\mathcal{A}_{\mathsf{nor}}) = \mathsf{lc}(\mathcal{A})$.

We further write $\mathcal{A}^0$ for the standard automaton $\mathcal{A}$ in which the initial state is not final. The automaton $\left(\mathcal{A}_{\mathsf{nor}}{}^0 \cdot \mathcal{A}^0\right)$ has a cut-vertex t. Finally, let $\mathcal{B} = \left(\mathcal{A}_{\mathsf{nor}}{}^0 \cdot \mathcal{A}^0\right)^*$ and $\mathcal{B}'$ the automaton in which t has been made final. Clearly, $|\mathcal{B}| = \mathcal{A}^*$ and Proposition 3.14 is established with the proof of the following lemma.

Lemma 4.8. $\mathsf{lc}(\mathcal{B}') = \mathsf{lc}(\mathcal{A}) + 1$.

Proof. The automaton $\mathcal{B}'$ without its initial state i is a ball; by definition, we have

$$\mathsf{lc}(\mathcal{B}') = 1 + \mathsf{min}\left\{\mathsf{lc}(\mathcal{B}' \setminus \{i, s\}) \,|\, s \in \mathcal{B}' \setminus i\right\} \ . \tag{4.14}$$

The final state t of $\mathcal{A}_{\mathsf{nor}}$ is a *cut vertex* (of the underlying graph) of $\mathcal{A}_{\mathsf{nor}}{}^0 \cdot \mathcal{A}^0$ If we set $s = t$ in (4.14), the balls of $\mathcal{B}'' = \mathcal{B}' \setminus \{i, s\}$ are those of $\mathcal{A}_{\mathsf{nor}}{}^0$ and $\mathcal{A}^0$ and hence $\mathsf{lc}(\mathcal{B}') \leqslant \mathsf{lc}(\mathcal{A}) + 1$. If we take an arbitrary s in $\mathcal{A}_{\mathsf{nor}}{}^0$ or $\mathcal{A}^0$, $\mathcal{B}''$ contains either $\mathcal{A}^1$ or $\mathcal{A}_{\mathsf{nor}}{}^0$ and $\mathsf{lc}(\mathcal{B}') \geqslant \mathsf{lc}(\mathcal{A}) + 1$. □

4.3 The derived-term automaton of an expression

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

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

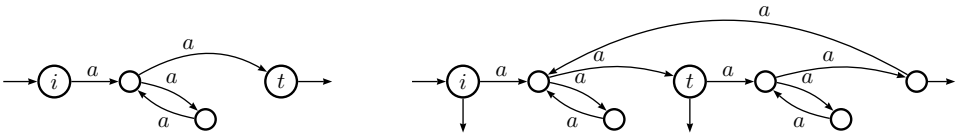


Figure 17. The standard automata $\mathcal{A}_{3\mathsf{nor}}$ and $\mathcal{B}'_3$ buit from $\left(\mathcal{A}_{3\mathsf{nor}}{}^{0} \cdot \mathcal{A}_3{}^{0}\right)^*$

The quotient is a *(right) action* of A^* on $\mathfrak{P}(A^*)$:

$$\forall L \in \mathfrak{P}(A^*)\,,\ \forall u, v \in A^* \qquad (u\,v)^{-1}L = v^{-1}\left(u^{-1}L\right)\ . \tag{4.15}$$

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 [12], 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^*.

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:

$$\frac{\partial}{\partial a}\mathsf{0} = \frac{\partial}{\partial a}\mathsf{1} = \emptyset, \qquad \forall b \in A \quad \frac{\partial}{\partial a}b = \begin{cases} \{1\} & \text{if} \quad b = a\,, \\ \emptyset & \text{otherwise,} \end{cases}\ , \tag{4.16}$$

$$\frac{\partial}{\partial a}(\mathsf{F}+\mathsf{G}) = \frac{\partial}{\partial a}\mathsf{F} \cup \frac{\partial}{\partial a}\mathsf{G}\ , \tag{4.17}$$

$$\frac{\partial}{\partial a}(\mathsf{F}\cdot\mathsf{G}) = \left(\frac{\partial}{\partial a}\mathsf{F}\right)\cdot\mathsf{G} \cup \mathsf{c}(\mathsf{F})\,\frac{\partial}{\partial a}\mathsf{G}\ , \tag{4.18}$$

$$\frac{\partial}{\partial a}(\mathsf{F}^*) = \left(\frac{\partial}{\partial a}\mathsf{F}\right)\cdot\mathsf{F}^*\ . \tag{4.19}$$

Equation (4.18) should be understood with the convention that the product $x\,X$ of a set X by a Boolean value x is X if $x = 1$ and $\emptyset$ if $x = 0$. The induction involved in Equations (4.17)–(4.19) 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 natural identies $\mathbf{N}$ — nor the idempotent identites $\mathbf{I}$ and $\mathbf{J}$.

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 (4.17)–(4.19) for letters in A and by induction on the length of v by:

$$\forall u \in A^+\,,\ \forall a \in A \qquad \frac{\partial}{\partial ua}\mathsf{E} = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{E}\right)\ . \tag{4.20}$$

The derivation of expressions is indeed parallel to the quotient of languages as we have the following property.

Property 4.9.

$$\forall L, K \subseteq A^* \,,\ \forall a \in A \qquad \begin{aligned} a^{-1}(L \cup K) &= a^{-1}L \cup a^{-1}K \ , \\ a^{-1}(L\,K) &= (a^{-1}L) \cup \mathsf{c}(L)\,a^{-1}K \ , \\ a^{-1}(L^*) &= (a^{-1}L)\,L^* \ . \end{aligned}$$

It follows, by induction on the depth of the expression:

$$\forall \mathsf{E} \in \mathsf{RatE}\,A^* \,,\ \forall a \in A \qquad \Big|\frac{\partial}{\partial a}\,\mathsf{E}\Big| = a^{-1}|\mathsf{E}| \tag{4.21}$$

which in turn implies, by induction on the length of words and the parallel between (4.20) and (4.15):

$$\forall \mathsf{E} \in \mathsf{RatE}\,A^* \,,\ \forall u \in A^+ \qquad \Big|\frac{\partial}{\partial u}\,\mathsf{E}\Big| = u^{-1}|\mathsf{E}| \ . \tag{4.22}$$

In particular, we have:

Property 4.10. If E is not a constant, there exists u in A^+ such that $\mathsf{c}(\frac{\partial}{\partial u}\,\mathsf{E}) = 1$.

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

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

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:

$$\mathrm{TD}\,(\mathsf{E}) = \textstyle\bigcup_{w \in A^+} \frac{\partial}{\partial w}\,\mathsf{E} \ . \tag{4.23}$$

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 4.11. *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 4.11 by induction on $\mathsf{d}(\mathsf{E})$ and *without reference to*

the derivation operation (*cf.* the *prebases* in [44] and Definition 6.2 below). It follows in particular that $\mathsf{Card}\,(\mathrm{TD}\,(\mathsf{E})) \leqslant \ell\,(\mathsf{E})$ and thus:

Corollary 4.12. $\mathsf{Card}\,(\mathrm{D}\,(\mathsf{E})) \leqslant \ell\,(\mathsf{E}) + 1\,.$

The computation of $\mathrm{D}\,(\mathsf{E})$ is a Δ-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 4.13 ([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 18.

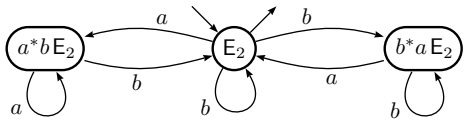


Figure 18. The automaton $\mathcal{A}_{\mathsf{E}_1}$.

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 4.12:

Theorem 4.14 ([17]). *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 4.15 ([3]). *Let* E, F *and* G *be three rational expressions. Then:*

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

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), free commutative monoids (for counting purpose), partially commutative, or trace, monoids (for modelling 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 $\operatorname{Rat} M$, is the smallest subset of $\mathfrak{P}(M)$ which contains the finite sets (including the empty set) 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 $M = G^*$. 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[11] seems to be in Elgot and Mezei's paper on rational relations.

[11] Hidden in a footnote!

Theorem 5.2 ([23]). *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 (Γ-maps). *For every finite automaton $\mathcal{A}$ over M, there exist rational expressions over M which denote $|\mathcal{A}|$.*

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

Proposition 5.4 (Δ-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 Section 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.

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 $\alpha\colon M \to N$ 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* $\mathrm{Rec}\, M$.

Recognisable and rational subsets We can then reproduce almost *verbatim* the converse part of the proof of Theorem 2.5. Let P be in $\mathrm{Rec}\, M$, 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 [42]). *If M is finitely generated, then* $\mathrm{Rec}\, M \subseteq \mathrm{Rat}\, M$.

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 $a^* \times c^*$ (where the product is formed component wise). 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, $\mathrm{Rat}\, M$ 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 [48]). *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 [24]). *The equivalence of finite automata, and hence of rational expressions, over $\{a,b\}^* \times \{c,d\}^*$ is undecidable.*

In contrast, the cases where $\mathrm{Rat}\, M$ is an effective Boolean algebra — such as when M is a (finitely generated) *free commutative monoid* [28] or *free group* [26] — 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 that former computations such as the *derivation* remain valid.[12]

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.* Chapter 20), $\mathbb{N}$ 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 restrict 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

[12] 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 how we define 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 4 and 20) and other works ([57, 9, 37, 54, 20, 41]).

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, $\langle s\,t, m\rangle = \sum_{uv=m} \langle s, u\rangle\langle t, v\rangle$. As M is graded, the product is well-defined, and the three operations make $\mathbb{K}\langle\langle M\rangle\rangle$ a semiring (*cf.* Chapter 4).

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 following definition 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 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, $t^* = \sum_{n\in\mathbb{N}} t^n$ and t is said to be *starable*. 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 $1-t$. Generally in semirings, the star of an element may be viewed as a substitute of taking the inverse in a poor structure that has no inverse. Hence is the name *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* $\mathrm{X} = s\,\mathrm{X} + t$.

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 $\mathbb{K}\langle M\rangle$.

Weighted rational 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 $\mathbb{K}\,\mathsf{Rat}\mathsf{E}\,M$. 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 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 $\mathbb{K}\,\mathsf{Rat}\mathsf{E}\,M$. If $(\mathsf{c}(\mathsf{U}))^*$ is defined, 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 Section 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.*

6.1.4 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.* Section 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. A proof which is very similar to the one given at Section 2.3, and which is *independent from* $\mathbb{K}$, yields the following.

Theorem 6.5 (Kleene–Schützenberger). *If A is finite, then* $\mathbb{K}\mathrm{Rat}\, A^* = \mathbb{K}\mathrm{Rec}\, A^*$.

6.2 From automata to expressions: the Γ-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 6.6. *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 Section 3: the state-elimination and system-solution methods, the McNaughton–Yamada and recursive algorithms, establish the weighted version of Proposition 2.3:

Proposition 6.7. *Let M be a graded monoid. For every finite $\mathbb{K}$-automaton $\mathcal{A}$ over M, there exist rational expressions over M which denote $|\mathcal{A}|$*

If the algorithms are the same, one has to establish nevertheless their correctness in this new and more complex framework. We develop the case of the system-solution 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 Section 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-solution 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', where the entries $E_{p,q}$, which are linear combinations of elements of M, are considered as expressions and denoted as such and where $|T_p\, 1|$ is the series reduced to the monomial $T_p\, 1_M$. 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 following an order ω on Q, the computation yields an expression that we denote by $\mathbf{E}_\omega(\mathcal{A})$, as in Section 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 6.8. *For every order ω on Q, $\mathbf{B}_\omega(\mathcal{A}) = \mathbf{E}_\omega(\mathcal{A})$ holds.*

Proposition 6.9. *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 extends to the weighted case (and it is now clear why it was important that identities $\mathbf{I}$ and $\mathbf{J}$ do not play a role in that result).

Theorem 6.10. *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 Δ-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 automaton for the Boolean case: a unique initial 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 11 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 11, 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

$$k\,\mathcal{A} = \left\langle \left(1 \;\boxed{0}\right), \left(\begin{array}{c|c} 0 & k\,J \\ \hline 0 & F \end{array}\right), \left(\begin{array}{|c|}\hline k\,c \\ \hline U \\ \hline \end{array}\right) \right\rangle,\quad \mathcal{A}\,k = \left\langle \left(1 \;\boxed{0}\right), \left(\begin{array}{c|c} 0 & J \\ \hline 0 & F \end{array}\right), \left(\begin{array}{|c|}\hline c\,k \\ \hline U\,k \\ \hline \end{array}\right) \right\rangle,$$

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

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

where $H = U\cdot c^*\,J + F$. As in Section 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 Δ-map:

Proposition 6.11 ([14, 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 19 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 $\mathsf{E}_4 = (1-a)\,a^*$.

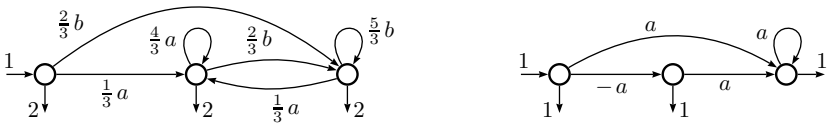


Figure 19. The $\mathbb{Q}$-automaton $\mathcal{S}_{\mathsf{E}_3}$ and the $\mathbb{Z}$-automaton $\mathcal{S}_{\mathsf{E}_4}$

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 rational if and only if it is contained in a finitely generated stable submodule of* $\mathbb{K}\langle\langle A^*\rangle\rangle$, *cf.* [9, 56].

Derivation The *derivation* of weighted rational 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 *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 (4.16) 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 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 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 Section 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 4.11 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 6.12 insures consistency between Definitions 6.1 and 6.2; the usefulness of the latter follows from Theorem 6.13.

Theorem 6.12 ([51, 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*

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

The 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 6.12 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 Δ-map since $|\mathcal{A}_\mathsf{E}| = |\mathsf{E}|$ holds.

Morphisms and quotients of (Boolean) automata are generalised to *Out-morphisms* and *quotients* of $\mathbb{K}$-automata (*cf.* [56, 5]). Theorem 4.14 is then extended to the weighted case.

Theorem 6.13 ([39]). *Let* E *be in* $\mathbb{K}\,\mathsf{Rat}\mathsf{E}\,A^*$. *Then* $\mathcal{A}_{\mathsf{E}}$ *is a quotient of* $\mathcal{S}_{\mathsf{E}}$.

Remark 6.14. This statement is a justification for Definition 6.2. The monomials that appear in the 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 occur in a derivation, as it can be observed for instance on the $\mathbb{Z}$-expression $\mathsf{E}_4 = (1-a)\,a^*$ (*cf.* Figure 20). With a definition of derived terms based on derivation only, Theorem 6.13 would not hold anymore.

Figure 20. The $\mathbb{Z}$-automaton $\mathcal{S}_{\mathsf{E}_4}$ and its $\mathbb{Z}$-quotient $\mathcal{A}_{\mathsf{E}_4}$

7 Notes

Most of the material presented in this chapter has appeared in previous work of the author [54, 55, 56].

Section 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 [46]. Berstel [7] attributes to Eilenberg the idea of distinguishing the family of recognisable from that of rational sets.

Besides the already quoted Elgot and Mezei's paper [23], other authors have certainly noticed the equality of expressiveness of automata and expressions beyond free monoids. It is part of Walljasper's thesis [59]; it can be found in Eilenberg's treatise [22]. The splitting of Kleene's theorem has been proposed in [53].

Section 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.* Chapter 20), each finite simple *group* gives rise to an identity that is independent from the others [18]. 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. The search for short expressions is performed by heuristics; as reported in [30], the naive one, modified or not as in [19], appears to be good (*cf.* Chapter 12 for more information on the subject).

McNaughton–Yamada algorithm is the implementation in the semiring of languages of the comtemporary Floyd–Roy–Warshall algorithms (in the Boolean or tropical semirings) [27, 49, 60].

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 [31] and then by D. Kirsten [33].

Section 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 [25] 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 [43] they are used to compute directly the *determinisation* of $\mathcal{S}_{\mathsf{E}}$, in [32] 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* [12] are obtained by replacing '$\cup$' by a '$+$' in (4.17) and (4.18). 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 Δ-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.* [43, 32]). Berry–Sethi [6] showed that the (Brzozowski) derivatives of $\overline{\mathsf{E}}$ coincide with the states of $\mathcal{S}_{\mathsf{E}}$, whereas Berstel–Pin [8] 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 [44] and Antimirov's derived terms was noted by Champarnaud–Ziadi [16], who called *equation automaton* the derived-term automaton.

Allauzen–Mohri have generalised Proposition 4.6 and Theorem 4.14 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 [15], an algorithm is given which is a kind of converse of a Δ-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 Γ-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].

Section 5. Changing the monoid Proposition 5.5 leads naturally to consider monoids M in which $\mathrm{Rat}\, M = \mathrm{Rec}\, M$ holds, and which one could call *Kleene monoids*. In [52] was defined the family of *rational monoids* which contains all previously known examples of Kleene monoids; still the inclusion is strict [45]. *Commutative* Kleene monoids, as well as finitely generated submonoids of $\mathrm{Rat}\, a^*$ are rational monoids [50, 1].

Section 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 [14, 51, 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 6.12 [51].

Krob [36] and Berstel–Reutenauer [10] have considered 'weighted rational expression' slightly different from those expression dealt with in this chapter. With their *differentiation* and *derivation*, they have tackled different problems than the construction of Δ-maps.

Acknowledgements The author is grateful to Z. Ésik and to J. Brzozowski who read a first draft of this chapter and made numerous and helpful remarks. P. Gastin, A. Demaille, and H. Grüber sent corrections on the first version. The careful reading of the final version by A. Szilard has been very encouraging and most helpful, and is heartily 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. 46

[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. 45

[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. 34, 45

[4] V. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Computer Sci.*, 155:291–319, 1996. 32, 34, 45

[5] M.-P. Béal, S. Lombardy, and J. Sakarovitch. Conjugacy and equivalence of weighted automata and functional transducers. In D. Grigoriev, editor, *CSR 2006*, number 3967 in Lect. Notes in Comput. Sci., pages 58–69, 2006. 44

[6] G. Berry and R. Sethi. From regular expressions to deterministic automata. *Theoret. Computer Sci.*, 48:117–126, 1986. 45

[7] J. Berstel. *Transductions and Context-Free Languages*. Teubner, 1979. 44

[8] J. Berstel and J.-E. Pin. Local languages and the Berry-Sethi algorithm. *Theoret. Computer Sci.*, 155:439–446, 1996. 45

[9] J. Berstel and C. Reutenauer. *Les séries rationnelles et leurs langages*. Masson, 1984. Translation: *Rational Series and Their Languages.* Springer, 1988. 37, 42

[10] 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. 46

[11] A. Brügemann-Klein. Regular expressions into finite automata. *Theoret. Computer Sci.*, 120:197–213, 1993. 26, 27, 30

[12] J. A. Brzozowski. Derivatives of regular expressions. *J. Assoc. Comput. Mach.*, 11:481–494, 1964. 32, 45

[13] 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

[14] 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. 42, 46

[15] P. Caron and D. Ziadi. Characterization of Glushkov automata. *Theoret. Computer Sci.*, 233:75–90, 2000. 45

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

[17] J.-M. Champarnaud and D. Ziadi. Canonical derivatives, partial derivatives and finite automaton constructions. *Theoret. Computer Sci.*, 289:137–163, 2002. 34

[18] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971. 10, 17, 44

[19] 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. 45

[20] M. Droste, W. Kuich, and H. Vogler. (Ed.), *Handbook of Weighted Automata*, Springer, 2009. 37

[21] L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10:385–397, 1963. 19, 20

[22] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974. 44

[23] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. and Develop.*, 9:47–68, 1965. 36, 44

[24] P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *J. Computer System Sci.*, 2:88–101, 1968. 37

[25] 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. 45

[26] M. Fliess. Deux applications de la représentation matricielle d'une série non commutative. *J. Algebra*, 19:344–353, 1971. 37

[27] R. W. Floyd. Algorithm 97. *Comm. Assoc. Comput. Mach.*, 5:345, 1962. 45

[28] S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas and languages. *Pacif. J. Math.*, 16:285–296, 1966. 37

[29] V. M. Glushkov. The abstract theory of automata. *Russian Math. Surveys*, 16:1–53, 1961. 25

[30] 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. 45

[31] K. Hashiguchi. Algorithms for determining relative star height and star height. *Inform. and Comput.*, 78:124–169, 1988. 45

[32] L. Ilie and S. Yu. Follow automata. *Inform. and Comput.*, 186(1):140–162, 2003. 45

[33] D. Kirsten. Distance desert automata and the star height problem. *RAIRO Theor. Informatics and Appl.*, 39(3):455–509, 2005. 45

[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, 44

[36] D. Krob. Differentiation of K-rational expressions. *Int. J. of Algebra and Computation*, 2:57–87, 1992. 46

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

[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. 21

[39] S. Lombardy and J. Sakarovitch. Derivation of rational expressions with multiplicity. *Theoret. Computer Sci.*, 332:141–177, 2005. 42, 43, 44, 46

[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. 45

[41] S. Lombardy and J. Sakarovitch. The validity of weighted automata. *Int. J. of Algebra and Computation*, 23(4):863–914, 2013. 37

[42] J. McKnight. Kleene's quotient theorems. *Pacific J. Math.*, 14:43–52, 1964. 36

[43] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Trans. Electronic Computers*, 9:39–47, 1960. 13, 25, 45

[44] B. G. Mirkin. An algorithm for constructing a base in a language of regular expressions. *Engineering Cybernetics*, 5:51–57, 1966. 34, 45

[45] M. Pelletier and J. Sakarovitch. Easy multiplications II. Extensions of rational semigroups. *Inform. and Comput.*, 88:18–59, 1990. 46

[46] D. Perrin. Les débuts de la théorie des automates. *Technique et Science Informatique*, 14:409–443, 1995. 44

[47] J.-É. Pin. AutoMathA Handbook, to appear. Vol. 1 and 2. 1, 2

[48] 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. 37

[49] B. Roy. Transitivité et connexité. *C. R. Acad. Sci. Paris Sér. A*, 249:216–218, 1959. 45

[50] C. P. Rupert. On commutative Kleene monoids. *Semigroup Forum*, 43:163–177, 1991. 46

[51] J. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoret. Computer Sci.*, 308:1–53, 2003. 43, 46

[52] J. Sakarovitch. Easy multiplications I. The realm of Kleene's theorem. *Inform. and Comput.*, 74:173–197, 1987. 46

[53] 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. 44

[54] J. Sakarovitch. *Eléments de théorie des automates*. Vuibert, 2003. Corrected English translation: *Elements of Automata Theory*, Cambridge University Press, 2009. 4, 12, 15, 37, 44

[55] 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. 44

[56] J. Sakarovitch. Rational and recognisable power series, 2009. in M. Droste et al., editors, *Handbook of Weighted Automata*, Springer, pages 105–174. 42, 44

[57] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer, 1977. 37

[58] K. Thompson. Regular expression search algorithm. *Comm. Assoc. Comput. Mach.*, 11:419–422, 1968. 30

[59] S. J. Walljasper. *Non-Deterministic Automata and Effective Languages*. PhD thesis, Univ. Iowa, 1970. 44

[60] S. Warshall. A theorem on Boolean matrices. *J. Assoc. Comput. Mach.*, 9:11–12, 1962. 45

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

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

Index