second case there is more interest in the performance of the algorithms, in terms of complexity or of the size of the results produced.

After the definition of rational expressions, we shall touch on the problem of determining the equivalence of expressions from a formal, or syntactic, point of view (that is, without referring to the construction of the corresponding automata). We will then use the ideas thus defined to make explicit the 'degree of closeness' between the different expressions obtained for the language recognised by a single finite automaton by applying the MNY algorithm or state elimination method, using the different possible orders on the states of the automaton. In the fourth part, we shall present the operation of *derivation* of expressions, which is the symbolic counterpart of the quotient of a language.

The transformation of a rational expression into a finite automaton, which is at the heart of many a problem in computer science, will be treated in the next section.

4.1 Rational expressions and languages

We begin by defining rational expressions on a fixed alphabet, and the languages they denote, then rational expressions on a set of variables and the result of interpreting such an expression.

4.1.1 Rational expressions over an alphabet

Let A be a given (non-empty) alphabet and let $\{0, 1, +, \cdot, *\}$ be five function symbols. Naturally, the operations $+$ and $\cdot$ are binary, $*$ is unary, and 0 and 1 are nullary (they represent constants).

Definition 4.1 A *rational expression over* A is a formula obtained inductively from the letters of A and the functions $\{0, 1, +, \cdot, *\}$ in the following manner:

(i) 0, 1, and a, for a in A, are rational expressions;

(ii) if E and F are rational expressions, then $(\mathsf{E}+\mathsf{F})$, $(\mathsf{E}\cdot\mathsf{F})$, and (E^*) are rational expressions.

We write $\mathsf{RatE}\,A^*$ for the set of rational expressions over A. □

Stating it in another way, rational expressions are *well-formed formulas*, made from the letters of A and 0 and 1 taken as *atomic formulas*, the binary operators $+$ and $\cdot$, and the unary operator $*$. For example,

$$((a\cdot b)+(b\cdot a)) \ , \quad ((((a+b)^*)\cdot(a\cdot b))\cdot((a+b)^*)) \ , \quad \text{and} \quad ((a+(b\cdot((((a\cdot(b^*))\cdot a)^*)\cdot b)))^*) \tag{4.1}$$

are rational expressions.

The definition of rational expressions calls for two other typographical symbols as well as the atoms: we needed the open and close parenthesis to ensure that expressions can be written unambiguously, since we have chosen an infix notation for reasons of legibility for the operators $+$ and $\cdot$. Without parentheses we would be unable to distinguish the two expressions $((a+b)\cdot c)$ and $(a+(b\cdot c))$. (We could have used a postfix or prefix notation, in which case parentheses would not have been required.) *cf. Exer. 4.1*

As we see in the expressions (4.1), the parentheses rapidly become the most numerous symbols as the expressions grow in length, making reading difficult (rather compromising our aim of legibility!) and leading to frequent errors in writing. We therefore adopt an *operator precedence* convention: $*$ takes precedence over $\cdot$ which takes precedence over $+$. With this convention we write

$$a\cdot b+c^* \quad \text{for} \quad ((a\cdot b)+(c^*)) \ , \quad a\cdot(b+c)^* \quad \text{for} \quad (a\cdot((b+c)^*)) \ ,$$
$$\text{and} \quad (a\cdot b+c)^* \quad \text{for} \quad (((a\cdot b)+c)^*) \ .$$

The expressions (4.1) become

$$a\cdot b+b\cdot a \ , \quad ((a+b)^*\cdot(a\cdot b))\cdot(a+b)^* \quad \text{and} \quad (a+b\cdot(((a\cdot b^*)\cdot a)^*\cdot b))^* \ .$$

Most of the definitions and propositions about rational expressions will consider the process of making formulas from expressions. This process can be followed by defining the *depth*[37] of an expression E which is an integer written $\mathsf{d}(\mathsf{E})$ and calculated in the following manner:

$$\mathsf{d}(0)=\mathsf{d}(1)=\mathsf{d}(a)=0 \ , \quad \text{for all } a \text{ in } A \ , \qquad \mathsf{d}((\mathsf{E}^*))=1+\mathsf{d}(\mathsf{E}) \ ,$$
$$\mathsf{d}((\mathsf{E}+\mathsf{F}))=\mathsf{d}((\mathsf{E}\cdot\mathsf{F}))=1+\max(\mathsf{d}(\mathsf{E}),\mathsf{d}(\mathsf{F})) \ .$$

Definition 4.2 To each rational expression E in $\mathsf{RatE}\,A^*$ we assign a corresponding language of A^*, written[38] $L[\mathsf{E}]$, or $|\mathsf{E}|$, and defined inductively – that is, by induction on the depth of expressions:

(i) for atomic expressions:

$$L[0]=\emptyset \ , \quad L[1]=1_{A^*} \ , \quad \text{and} \quad L[a]=a \quad \text{for all } a \text{ in } A \ ; \tag{4.2}$$

(ii) for composite expressions:

$$L[(\mathsf{E}+\mathsf{F})]=\{L[\mathsf{E}]\}\cup\{L[\mathsf{F}]\} \ , \quad L[(\mathsf{E}\cdot\mathsf{F})]=\{L[\mathsf{E}]\}\{L[\mathsf{F}]\} \ , \quad \text{and} \quad L[(\mathsf{E}^*)]=\{L[\mathsf{E}]\}^* \ . \tag{4.3}$$

We say that E *denotes* the language $L[\mathsf{E}]$. □

For example,

$$L[(a+b)]=\{a,b\} \ , \tag{4.4}$$
$$L[((((a+b)^*)\cdot(a\cdot b))\cdot((a+b)^*))]=\{a,b\}^*ab\{a,b\}^* \ . \tag{4.5}$$

[37] I use the term 'depth' rather than 'height' (which is also possible and even more widely used) to avoid any confusion with 'star height', which will be defined later (*cf.* Section 6). Both terms use the image of a rational expression as a tree.

[38] I use square brackets in $L[\mathsf{E}]$ to be different from the parentheses used in expressions. This slightly cumbersome notation is used in the definitions to make them more legible. The more concise notation $|\mathsf{E}|$ will be used hereafter, when we are more accustomed to expressions.

Thus, $((((a+b)^*)\cdot(a\cdot b))\cdot((a+b)^*))$ denotes the language of words in A^* that contain at least one factor ab.

At first glance this last assertion, or equations such as (4.4) or (4.5), may seem utterly tautological. Why bother to make the subtle distinction between $(a+b)$ and $a\cup b$? To write pompously that $L[a]=a$? Are we not pointlessly adding unnecessary complexity? Actually, no. We must understand expressions as *syntactic* objects, and the languages which they denote as their *semantics*. For example, $(a+b)$ and $(b+a)$ are two *distinct* expressions which denote the *same* language $\{a,b\}$.

A procedure for automatic information processing – in short, a 'computer' – can only manipulate sequences of symbols and can therefore access only a syntactic description of the objects with which it deals. The role of programming is to ensure that objects which are syntactically different but semantically identical produce the same result.

If this argument has convinced the reader that I am not engaging in tetrapyloctomy for pure sadism, we can continue, starting by showing that the usage of the same adjective *rational* for languages and expressions is justified.

cf. p. 87 **Proposition 4.1** *A language of A^* is rational if and only if it is denoted by a rational expression over A.*

Proof. The family of languages denoted by a rational expression contains $\emptyset$, 1_{A^*}, and all the letters of A by (4.2); it is closed under union, product and star by (4.3): it therefore contains Rat A^*.

Conversely, by induction on the depth of expressions, every language denoted by an expression is obtained from letters and the empty word by a finite series of unions, products and stars; it is thus contained in any rationaly closed family (that contains letters and empty word), and hence rational. ■

cf. pp. 87–94 We can therefore give another version of Proposition 2.1 and of its proofs:

cf. Sec. 5, p. 145 **Corollary 4.2** *Every rational expression* E *can effectively be transformed into an automaton that recognises the language denoted by* E. ■

Definition 4.3 Two rational expressions over A are *equivalent* if they denote the same language; that is, E and F are equivalent if $L[\mathsf{E}]=L[\mathsf{F}]$ and in this case we write $\mathsf{E}\equiv\mathsf{F}$. □

Exam. 2.6, p. 96 **Example 4.1** In Section 2 we tried several methods for calculating the language recognised by the automaton $\mathcal{R}_1$ and obtained different expressions that denote the same language and are therefore equivalent:

$$(a^*\cdot b)^+\cdot a^+ + a^* \quad\equiv\quad 1_{A^*}+(a^*\cdot b)^*\cdot a^+ \quad\equiv\quad (b^*\cdot a)^*\ ,$$

where E^+ is an abbreviation for $\mathsf{E}\cdot\mathsf{E}^*\equiv\mathsf{E}^*\cdot\mathsf{E}$. □

Corollaries 3.6 and 4.2 immediately give us:

Theorem 4.1 *The equivalence of two rational expressions over A is decidable.* ■

Remark 4.1 This result, which 'falls out' without needing a particular proof, is nevertheless fundamental. It is also necessary to realise that *no other proof is known* of the decidability of the equivalence of expressions than by going via automata,[39] and hence by using Kleene's Theorem. □

4.1.2 Rational expressions over a set of variables

It is useful to give a more abstract definition of rational expressions in order to be able to distinguish atomic formulas from letters of the alphabet. Let $\mathbf{X}=\{\mathsf{x},\mathsf{y},\ldots\}$ be a set (*a priori* infinite, but countable) of symbols, called *variables*. A rational expression over $\mathbf{X}$ is, as in Definition 4.1, a well-formed formula constructed from 0, 1 and the elements of $\mathbf{X}$ taken as atomic formulas, and the operations $\{+,\cdot,*\}$. We write Var E for the set of variables that appear in the expression E. Thus for example

$$\mathsf{E}_1=(\mathsf{x}+\mathsf{y}\cdot(((\mathsf{x}\cdot\mathsf{t}^*)\cdot\mathsf{z})^*\cdot\mathsf{y})^*)^*$$

is a rational expression, and $\mathrm{Var}\,\mathsf{E}_1=\{\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{t}\}$.

Definition 4.4 An *interpretation* of a rational expression E over A^* is a map τ from Var E to $\mathfrak{P}(A^*)$, and hence a morphism from $(\mathrm{Var}\,\mathsf{E})^*$ to $\mathfrak{P}(A^*)$. The *result* of the interpretation τ of E, written $[\mathsf{E}]\tau$, is defined by induction on the depth of E:

(i) for atomic expressions:

$$[0]\tau=\emptyset\ ,\quad [1]\tau=1_{A^*}\ ,\quad\text{and}\quad [\mathsf{x}]\tau=\mathsf{x}\tau\quad\text{for all x in Var E}\ ; \tag{4.6}$$

(ii) for composite expressions:

$$[(\mathsf{F}+\mathsf{G})]\tau=\{[\mathsf{F}]\tau\}\cup\{[\mathsf{G}]\tau\}\ ,\quad [(\mathsf{F}\cdot\mathsf{G})]\tau=\{[\mathsf{F}]\tau\}\{[\mathsf{G}]\tau\}\ ,\quad\text{and}\quad [(\mathsf{F}^*)]\tau=\{[\mathsf{F}]\tau\}^*\ . \tag{4.7}$$

Example 4.2 If τ_1 is the interpretation of Var E_1 over A^* defined by

$$[\mathsf{x}]\tau_1=[\mathsf{z}]\tau_1=a\quad\text{and}\quad[\mathsf{y}]\tau_1=[\mathsf{t}]\tau_1=b\ ,$$

it follows that $[\mathsf{E}_1]\tau_1=(a\cup b(((ab^*)a)^*b))^*$. □

This more abstract definition allows us to broaden the use of rational expressions to semirings other than $\mathfrak{P}(A^*)$, and gives us some terminological abbreviations that we will use later. It also gives rise to a proposition more general than Proposition 4.1, with an identical proof.

[39] These sorts of assertion always cause objections to be raised. In particular, the computation of *derivatives* (and also of *derived terms*) that we shall see later (Sec. 4.4, Sec. 5.2, Sec. III.4.2) might give the impression that we could develop procedures for proving the equivalence of expressions entirely in the world of expressions. We shall see that the computation of derivatives corresponds in fact to the computation of an automaton.

Proposition 4.3 *Let C be a family of subsets of A^*. A subset R of A^* belongs to* $\mathrm{Rat}\, C$ *if and only if there exists a rational expression* E *and an interpretation of* E *whose value is in C and whose result is R.* ■

Definition 4.3 extends naturally: two expressions over the same set of variables are equivalent if they have the same result *for every interpretation.* Amongst all the possible interpretations of a rational expression E, one plays a special role: the *free interpretation* of E, the interpretation of E in $(\mathrm{Var}\, \mathsf{E})^*$ which associates $\{x\}$ with x and whose result is written $L[\mathsf{E}]$ (this notation is consistent with Definition 4.2). This free interpretation is the ancestor of all the other interpretations, in the sense that we have the following result immediately by induction on the depth of E.

Proposition 4.4 *Let* E *be a rational expression and τ any interpretation of* E *over A^*. Then* $[\mathsf{E}]\tau = (L[\mathsf{E}])\,\tau$. ■

cf. also Rem. II.1.1, p. 227

Hence we conclude:

Corollary 4.5 *Two rational expressions* E *and* F *are equivalent if and only if* $L[\mathsf{E}] = L[\mathsf{F}]$. ■

These results justify our giving ourselves the ability to confuse, *by abuse of terminology*, a language defined by rational operations with the corresponding rational expression: for example, $(a+b)^*ab(a+b)^*$ is both the language of words that contain at least one factor ab and a particular expression that denotes it.

Exercises

• 4.1 (a) Give a definition of rational expressions using a postfix notation for the operators $+$, $\cdot$ and $*$. Give the expressions that correspond to the expressions (4.1) with this convention.

(b) Do the same using prefix notation. (c) Draw conclusions.

• 4.2 Give a rational expression for the language of words that do not contain a factor ba.

4.2 Rational identities

Corollary 4.5 and Theorem 4.1 ensure that we can decide whether two rational expressions over a set of variables $\mathbf{X}$ are equivalent. We know in any case that this decision process implies the transcription of expressions into automata and assumes that the latter are determinised if necessary. Thus we are led to ask if we can determine whether two expressions are equivalent directly, working with the expressions themselves, without resorting to automata.

cf. Rem. 4.1, p. 127

We have already said that no other proof is known of the equivalence of expressions than via automata, and hence that the answer to the question above is negative. It is nonetheless interesting to try to solve it, as much to understand the properties that we can prove in this direction as to grasp the reasons that prevent us from arriving at a complete solution and which, *a contrario*, shed new light on our appeal to automata.[40]

If we try to analyse the reasons why two expressions can be equivalent – that is, can denote the same language – there are some cases that seem obvious. For example, for all expressions E and F, we have

$$\mathsf{E}+\mathsf{F} \quad \equiv \quad \mathsf{F}+\mathsf{E} \tag{4.8}$$

since $|\mathsf{E}| \cup |\mathsf{F}| = |\mathsf{F}| \cup |\mathsf{E}|$: that is, since union is commutative. If we return to the two equivalences of Example 4.1:

$$(a^* \cdot b)^+ \cdot a^+ + a^* \quad \equiv \quad (a^* \cdot b)^* \cdot a^+ + 1_{A^*} \quad \equiv \quad (b^* \cdot a)^* \ ,$$

the first does not seem much more complicated than the previous example: it suffices to write $a^* \equiv 1_{A^*} + a^+$, to take out the factor a^+ and use the same identity, in the opposite direction (and for $u = a^* \cdot b$), to go from the left-hand expression to that in the middle. The second example is not so simple. And what of the identity

$$(a+b(ab^*a)^*b)^* \equiv a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b+a(ba^*b)^*a)^*a(ba^*b)^*ba^* ,$$

obtained by two distinct computations[41] of the language recognised by $\mathcal{P}_2$, the divider by 3?

Our objective in this subsection is to formalise this intuition with a mathematical rigour that allows us to state some properties and prove them. An identity between rational expressions, or *rational identity*, is an equation such as (4.8) which expresses the equivalence of two expressions over a set of variables (the expressions themselves being variables written E, F, *etc.*). In general, an equivalence corresponds to a property of the interpretation of the expressions. We can then reuse identities by substituting them into other identities to obtain new equivalences.

4.2.1 Classical identities

Trivial and natural identities. The first series of identities expresses the fact that expressions will be *interpreted in a semiring* – that is, that $+$ and $\cdot$, 0 and 1 will satisfy the corresponding axioms: $+$ and $\cdot$ are associative, $+$ is commutative, 0 is an identity element for $+$ and a zero for $\cdot$, 1 is an identity element for $\cdot$, and $\cdot$ is left and right distributive over $+$:

$$\mathsf{E}+0 \equiv 0+\mathsf{E} \equiv \mathsf{E}\ , \qquad \mathsf{E}\cdot 0 \equiv 0\cdot \mathsf{E} \equiv 0\ , \qquad \mathsf{E}\cdot 1 \equiv 1\cdot \mathsf{E} \equiv \mathsf{E}\ , \tag{T}$$

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

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

$$\mathsf{E}+\mathsf{F} \equiv \mathsf{F}+\mathsf{E}\ . \tag{C}$$

[40] This ambitious programme will not be fully realised in the course of this work.

[41] As the conscientious reader who has done Exercise 2.15 will remember.

The identities in the first line (**T**) may be described as trivial. Other than being obvious, they have the particular property that if, in any expression E, one of the six terms on the left or in the middle is replaced by the corresponding right-hand term, iteratively, until all occurrences of the six terms have been eliminated, a *unique* equivalent expression is obtained (that is, independent of the order in which the replacements were made) which is effectively computable,[42] which we call a *reduced expression*. From now on, all computations on expressions will be performed modulo reduction by trivial identities without further mention.

The three other lines of identities **A**, **D** and **C** are no less obvious, since they reflect the properties of operations in a semiring. With the foregoing they are called *natural identities*. They are also often used without explicit mention (but not always). In particular, we allow ourselves from now on to write $\mathsf{E}+\mathsf{F}+\mathsf{G}$ and $\mathsf{E}\cdot\mathsf{F}\cdot\mathsf{G}$. (Still aiming to lighten the notation, we will often replace the $\cdot$ operator by simple juxtaposition and write EFG for $\mathsf{E}\cdot\mathsf{F}\cdot\mathsf{G}$.) Equally, we can and shall be led to distinguish the associativity of the sum $\mathbf{A_s}$ from that of the product $\mathbf{A_p}$. cf. Rem. 4.5, p. 140

The identities that really matter are those that express, or at least reflect, the properties of the operator $*$ and those of the semiring $\mathfrak{P}(A^*)$, or a combination of the two, that is, of the operator $*$ in $\mathfrak{P}(A^*)$.

cf. p. 85 **Aperiodic and cyclic identities.** The relation (2.5), which we have been able to see since the definition of the star operation, can now be written as a pair of identities:

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

By the definition of star, we also have

$$0^* \equiv 1\ . \qquad (\mathbf{Y})$$

The following identities, called *aperiodic identities*,[43] are collected in a statement which requires proof.

Proposition 4.6 *For all rational expressions* E *and* F

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

Proof. Corollary 4.5 implies that an identity between two expressions is true if and only if the free interpretations of these expressions are equal: that is, if the languages that they denote when the symbols E, F, *etc.* are replaced by distinct letters from an alphabet A are equal.

[42] By a simple algorithm, linear in the size of the expression E.

[43] This term was proposed by D. Krob, and is justified in the bibliographic notes p. 215.

To prove $\mathbf{S_l}$, we therefore need to show that $(a+b)^* = a^*(ba^*)^*$. Arden's Lemma tells us that $(a+b)^*$ is the unique solution of Lem. 2.9, p. 100

$$X = (a+b)\,X + 1_{A^*}\ .$$

We can verify using $\mathbf{U_l}$ (and the natural identities), that $a^*(ba^*)^*$ is a solution of the same equation:

$$(a+b)\big(a^*(ba^*)^*\big)+1_{A^*} = a\,a^*(ba^*)^* + \underbrace{b\,a^*(ba^*)^*+1_{A^*}}_{\text{by }(\mathbf{U_l})}$$
$$= a\,a^*(ba^*)^* + (ba^*)^* = \underbrace{(a\,a^*+1_{A^*})}_{\text{by }(\mathbf{U_l})}(ba^*)^* = a^*(ba^*)^*\ .$$

A symmetric calculation shows that $(a^*b)^*a^*$ is a solution of $X = X\,(a+b)+1_{A^*}$, by which $\mathbf{S_r}$. Analogously, the language $1_{A^*}+a\,(ba)^*b$ is a solution of $X = a\,b\,X + 1_{A^*}$:

$$a\,b\,(1_{A^*}+a\,(ba)^*b)+1_{A^*} = a\,b + a\,b\,a\,(ba)^*b + 1_{A^*}$$
$$= a\,\underbrace{(1_{A^*}+b\,a\,(ba)^*)}_{\text{by }(\mathbf{U_l})}\,b + 1_{A^*} = a\,(ba)^*b+1_{A^*}\ .$$

This proves **P**. ■

We will use the aperiodic identities in the next subsection to compare the different methods of computing the language accepted by a finite automaton. The following proposition introduces an infinity of identities, indexed by $\mathbb{N}$, and called *cyclic identities*.

Proposition 4.7 *For every rational expression* E*, and for every* n *in* $\mathbb{N}$*, we have*[44]

$$\mathsf{E}^* \equiv \mathsf{E}^{<n}\cdot(\mathsf{E}^n)^*\ . \qquad (\mathbf{Z}_n)$$

Proof. We reuse the method of the proof of Proposition 4.6. For each n, $a^{<n}(a^n)^*$ is a solution of $X = a\,X + 1_{A^*}$, since

$$a\,\big(a^{<n}(a^n)^*\big)+1_{A^*} = [a+\cdots+a^{n-1}+\underbrace{a^n](a^n)^*+1_{A^*}}_{\text{by }(\mathbf{U_l})}$$
$$= [a+\cdots+a^{n-1}]\,(a^n)^* + (a^n)^*$$
$$= [1_{A^*}+a+\cdots+a^{n-1}](a^n)^*\ . \qquad ■$$

Idempotent identities. Union, which is addition in $\mathfrak{P}(A^*)$, is idempotent, which implies that star is also an idempotent operation in $\mathfrak{P}(A^*)$. These observations give us two more identities:

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

[44] Admitting that $\mathsf{E}^{<n}$ is a suitable abbreviation for the expression $1+\mathsf{E}+\mathsf{E}^2+\cdots+\mathsf{E}^{n-1}$.

We shall call the collection of identities from **T** to **J** *classical identities*.[45]

4.2.2 A formal computation

From one or more identities we can prove new ones. We note for example that $\mathbf{U_l}$ and $\mathbf{U_r}$ are obtained by replacing F by 1 (resp. E by 1) in **P**.[46] Another example is the second equivalence in Example 4.1:

$$(a^*b)^*a^+ + 1_{A^*} \equiv \underbrace{(a^*b)^*a^*}_{\text{by }(\mathbf{S_r})} a + 1_{A^*} \equiv (a+b)^*a + 1_{A^*}$$
$$\equiv \underbrace{(b+a)^*}_{\text{by }(\mathbf{S_r})} a + 1_{A^*} \equiv \underbrace{(b^*a)^*b^*a + 1_{A^*}}_{\text{by }(\mathbf{U_l})} \equiv (b^*a)^* \ .$$

Notation To be able to express the degree of similarity of two expressions, we will use a notation familiar to logicians and write

$$\mathbf{X} \vdash \qquad \mathsf{E} \equiv \mathsf{F}$$

to mean that the identity $\mathsf{E} \equiv \mathsf{F}$ can be deduced from the identity, or set of identities, **X**. The preceding identity could therefore be written

$$\mathbf{S_r} \wedge \mathbf{U_l} \vdash \qquad (a^*b)^*a^+ + 1_{A^*} \equiv (b^*a)^*$$

(the symbol $\wedge$ denoting the conjunction of identities). We will write $\mathbf{U} = \mathbf{U_l} \wedge \mathbf{U_r}$ and $\mathbf{S} = \mathbf{S_l} \wedge \mathbf{S_r}$.

Remark 4.2 To be rigorous, we would have to specify that the above identities are valid for expressions interpreted in $\mathfrak{P}(A^*)$.

cf. Prop. III.2.5, p. 396 and Prop. III.5.3, p. 473

The proof of Propositions 4.6 and 4.7 show that the aperiodic and cyclic identities are satisfied when the operator $*$ and the semiring in which the expressions are interpreted satisfy the identities **U** and Arden's Lemma. □

We could continue our consideration of axioms and identities by reversing the question, and seeking to describe sets of axioms which allow us to find the identity of all equivalent expressions (complete systems) and, among them, the minimal sets (independent systems).

This is a rich area of study, replete with often difficult results, which we shall not explore. The most important point is that every complete axiomatic system is necessarily infinite, and that if we allow, as well as substitution, a stronger inference rule deduced from Arden's Lemma, then we can build a complete finite system (see bibliographic notes).

---

[45] These are the same as what J. H. Conway called *classical axioms* in his book *Regular Algebra and Finite Machines* [66].

[46] But this would only really have been interesting if **P** had been proved without using $\mathbf{U_l}$.

Exercises

- 4.3 Prove the following identities using the classical identities (but without Lemma 2.9).
  (a) $(a+b)^* = a^* + a^*b(a+b)^*$; (b) $(a+b)^* = a^*(b^+a^+)^*b^*$;
  (c) $(a+b)^+ = a^+ + a^*(ba^*)^+$.
- 4.4 Do the same for:
  (a) $(a+b+c)^* = [a+b+c(c+b)^*a]^*[1+c(c+b)^*]$;
  (b) $(x+y)^* = [x+y(y^+x)^*x]^*[1+y(y^+x)^*y^*]$. [Hint: use (a) to show (b).]

4.5 $\mathbf{P} \vdash \qquad (\mathsf{EF})^*\mathsf{E} \equiv \mathsf{E}(\mathsf{FE})^*$.

4.6 Show that, for all integers n and m, $\quad \mathbf{Z}_n \wedge \mathbf{Z}_m \vdash \qquad \mathbf{Z}_{n\,m}$.

4.3 Expressions for the behaviour of a finite automaton

In Section 2 we described three methods, distinct in their formulation at least, for computing the language recognised by a finite automaton $\mathcal{A}$. We can now be more precise and say that we described three distinct methods for computing *an expression that denotes* $L(\mathcal{A})$, the computation and its result, depending, in each case, on a total order assigned to the states of $\mathcal{A}$. We also said that these three methods are three presentations of a *single proof*; we now have concepts adequate for justifying this assertion. We will now make explicit the degree of relationship between the various expressions that can be obtained by these methods and by the different ways of putting them into practice. In doing this we describe syntactic procedures that allow us to turn one such expression into another.

For the rest of this subsection let $\mathcal{A} = \langle Q, A, E, I, T \rangle$ be a finite automaton and ω a total order on Q which fixes the operation of the algorithms that calculate $L(\mathcal{A})$.

4.3.1 The state elimination and equation solution methods

Let us begin by verifying the exact identity of the state elimination method and the solution of a system of linear equations taken from the automaton (solution by Gaussian elimination). Recall the notation of Section 2 for writing this system: for p and q in Q, the *set of words* that are the label of a computation which goes from p to a final state of $\mathcal{A}$ is written *cf. p. 99*

$$L_p = \{f \mid \exists t \in T \quad p \xrightarrow[\mathcal{A}]{f} t\} \ ,$$

and we write $E_{p,q}$ for the *set of labels of transitions* that go from p to q and the symbol $\delta_{p,R}$ for a subset R of Q; this equals 1_{A^*} if p is in R and $\emptyset$ if not. The system of equations associated with $\mathcal{A}$ is written

$$L(\mathcal{A}) = \sum_{p \in I} L_p \quad = \sum_{p \in Q} \delta_{p,I}\, L_p \tag{4.9}$$

$$\forall p \in Q \qquad L_p = \sum_{q \in Q} E_{p,q}\, L_q + \delta_{p,T} \ . \tag{4.10}$$

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

$$L(\mathcal{A}) = \sum_{p\in Q'} G_p L_p + H \tag{4.11}$$

$$\forall p \in Q' \qquad L_p = \sum_{q\in Q'} F_{p,q} L_q + K_p \ . \tag{4.12}$$

We can make a generalised automaton $\mathcal{B}'$ corresponding to such a system, whose 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':

- the transition from p to q is labelled $F_{p,q}$;
- the transition from p to t is labelled K_p;
- the transition from i to p is labelled G_p;
- and the transition from i to t is labelled H.

cf. p. 97 Note that this definition applied to the system (4.9)–(4.10) characterises the automaton constructed in the first phase of the state elimination method applied to $\mathcal{A}$.

The elimination in the system (4.11)–(4.12) of the unknown L_p by substitutions and the application of Arden's Lemma give the system

$$L(\mathcal{A}) = \sum_{r\in Q'\setminus p} \left[G_r + G_p F_{p,p}^* F_{p,r}\right] L_r + \left[H + G_p F_{p,p}^* K_p\right] \tag{4.13}$$

$$\forall r \in Q' \setminus p \qquad L_r = \sum_{q\in Q'\setminus p} \left[F_{r,q} + F_{r,p} F_{p,p}^* F_{r,q}\right] L_q + \left[K_r + F_{r,p} F_{p,p}^* K_p\right] \ , \tag{4.14}$$

whose coefficients are exactly the transition labels of the generalised automaton obtained by removing the state p from $\mathcal{B}'$.

Thus, since the starting points correspond and since each step maintains the correspondence, the expression obtained for $L(\mathcal{A})$ by the state elimination method is the same as that obtained by the solution of the system (4.9)–(4.10). More precisely, we can say that the state elimination method, or BMC algorithm, reproduces in the automaton $\mathcal{A}$ the computations corresponding to the solution of the system.

4.3.2 The BMC and MNY algorithms, identical orders

The order ω fixes the operation of the state elimination method whose result is a rational expression over A^*, written[47] $\mathsf{E}_{\mathrm{BMC}}(\mathcal{A}, \omega)$. For greater precision, we write the result of this algorithm $\mathsf{E}_{\mathrm{BMC}}(\mathcal{A}, \omega, (p, q))$ when we take p as the initial state and q as the final state.

On the other hand, we will write $\mathsf{M}_{\mathrm{MNY}}(\mathcal{A}, \omega)$ for the matrix of rational expressions obtained when we apply the McNaughton–Yamada algorithm to the automaton $\mathcal{A}$ whose states are ordered by ω. It then follows that:

[47] A reminder that this algorithm is due to J. Brzozowski and E. McCluskey.

Proposition 4.8 *Let $\mathcal{A} = \langle Q, A, E, I, T \rangle$ an automaton over A^*. For every (total) order ω on Q and all p and q in Q, we have*

$$\mathbf{U} \vdash \qquad [\mathsf{M}_{MNY}(\mathcal{A}, \omega)]_{p,q} \quad \equiv \quad \mathsf{E}_{BMC}(\mathcal{A}, \omega, (p, q)) \ .$$

Proof. This result is not so surprising: to prove it, we will show a correspondence between the operations performed by the two algorithms. The difficulty, if it can be called that, is that we have to compare two objects whose form and mode of construction are rather different: on one hand a $Q\times Q$ matrix obtained by successive transformations, from which we choose one entry; and on the other an expression obtained by repeated modification of an automaton, hence of a matrix, but one whose size decreases at each step.

In the following, $\mathcal{A}$ and ω are fixed and remain implicit. The automaton $\mathcal{A}$ has n states, identified by 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 the MNY algorithm 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 $k+1 \leqslant r, s$). We write

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

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

$$\forall r, s\,,\ 1 \leqslant r, s \leqslant n \qquad \mathsf{M}^{(0)}_{r,s} = \mathsf{E}^{(0)}(r, s) \ , \tag{4.15}$$

which will be the base case of the inductions to come. Algorithm MNY is written:

$$\forall k\,,\ 0 < k \leqslant n\,,\ \forall r, s\,,\ 1 \leqslant r, s \leqslant n$$
$$\mathsf{M}^{(k)}_{r,s} = \mathsf{M}^{(k-1)}_{r,s} + \mathsf{M}^{(k-1)}_{r,k} \cdot [\mathsf{M}^{(k-1)}_{k,k}]^* \cdot \mathsf{M}^{(k-1)}_{k,s} \ . \tag{4.16}$$

The state elimination algorithm is written

$$\forall k\,,\ 0 < k \leqslant n\,,\ \forall r, s\,,\ k < r, s \leqslant n$$
$$\mathsf{E}^{(k)}(r, s) = \mathsf{E}^{(k-1)}(r, s) + \mathsf{E}^{(k-1)}(r, k) \cdot [\mathsf{E}^{(k-1)}(k, k)]^* \cdot \mathsf{E}^{(k-1)}(k, s) \ . \tag{4.17}$$

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

$$\forall r, s\,,\ 1 \leqslant r, s \leqslant n\,,\ \forall k\,,\ 0 \leqslant k < \min(r, s) \qquad \mathsf{M}^{(k)}_{r,s} = \mathsf{E}^{(k)}(r, s) \ . \tag{4.18}$$

We see in fact (as there is, even so, something to see) that if $k < \min(r, s)$ 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 (4.16), are such that $l < \min(u, v)$.

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 (4.16):

cf. the proof of Prop. 2.8, p. 97

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

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

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

and hence, by (4.18)

$$\forall s\,,\ p < s \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(p)}_{p,s} \equiv \mathsf{E}^{(p)}(i,s) \ . \tag{4.19}$$

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 still computed by (4.16) and $\mathsf{E}^{(k)}(i,s)$ by

$$\mathsf{E}^{(k)}(i,s) = \mathsf{E}^{(k-1)}(i,s) + \mathsf{E}^{(k-1)}(i,k) \cdot [\mathsf{E}^{(k-1)}(k,k)]^* \cdot \mathsf{E}^{(k-1)}(k,s) \ . \tag{4.20}$$

From (4.19), and based on an observation analogous to the previous one, we conclude from the term-by-term correspondence of (4.16) and (4.20) that

$$\forall k\,,\ p < k\,,\ \forall s\,,\ p < s \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(k)}_{p,s} \equiv \mathsf{E}^{(k)}(i,s) \ . \tag{4.21}$$

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

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

and

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

and hence

$$\forall r\,,\ q < r \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(q)}_{r,q} \equiv \mathsf{E}^{(q)}(r,t) \ . \tag{4.22}$$

The steps following q give rise to an equation symmetric to (4.21) (for column q of the matrices $\mathsf{M}^{(k)}$):

$$\forall k\,,\ q < k\,,\ \forall r\,,\ q < r \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(k)}_{r,q} \equiv \mathsf{E}^{(k)}(r,t) \ . \tag{4.23}$$

Finally, from

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

and

$$\mathsf{E}^{(k)}(i,t) = \mathsf{E}^{(k-1)}(i,t) + \mathsf{E}^{(k-1)}(i,k) \cdot [\mathsf{E}^{(k-1)}(k,k)]^* \cdot \mathsf{E}^{(k-1)}(k,t) \ ,$$

Equations (4.18), (4.21) and (4.23) together allow us to conclude, by induction on k, that

$$\forall k\,,\ q \leqslant k \leqslant n \qquad \mathbf{U} \vdash \qquad \mathsf{M}^{(k)}_{p,q} \equiv \mathsf{E}^{(k)}(i,t) \ . \tag{4.24}$$

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

4.3.3 The BMC and MNY algorithms, distinct orders

Having compared the state elimination method and algorithm MNY, we can compare the results of these algorithms for different execution conditions.

Theorem 4.2 *Let $\mathcal{A} = \langle Q, A, E, I, T\rangle$ be an automaton over A^*. The expressions denoting $L(\mathcal{A})$ computed by the McNaughton–Yamada algorithm, like those computed by the state elimination method or the solution of a system of equations, are all equivalent modulo* $\mathbf{S}$ *and* $\mathbf{P}$, *that is, for all orders ω and ω' on Q and all p and q in Q*

$$\mathbf{S} \wedge \mathbf{P} \quad \vdash \quad [\mathsf{M}_{MNY}(\mathcal{A},\omega)]_{p,q} \quad \equiv \quad [\mathsf{M}_{MNY}(\mathcal{A},\omega')]_{p,q} \ ,$$
$$\mathbf{S} \wedge \mathbf{P} \quad \vdash \quad \mathsf{E}_{BMC}(\mathcal{A},\omega,(p,q)) \quad \equiv \quad \mathsf{E}_{BMC}(\mathcal{A},\omega',(p,q)) \ .$$

Proof. The previous proposition allows us to show the property for expressions computed by the state elimination method, which is easier to deal with (remembering that $\mathbf{P} \vdash \mathbf{U}$). Furthermore, we can go from an order ω to any other order ω', a permutation of Q, by a series of transpositions.

We therefore arrive at the situation illustrated in Figure 4.1(a) and need to show that the expressions obtained by the state elimination method when we first remove the state r and then r' are equivalent to those obtained from removing first r' and then r, modulo $\mathbf{S} \wedge \mathbf{P}$.

The removal of state r gives the expressions in Figure 4.1(b). The removal of state r' gives the expression

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

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

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

We write

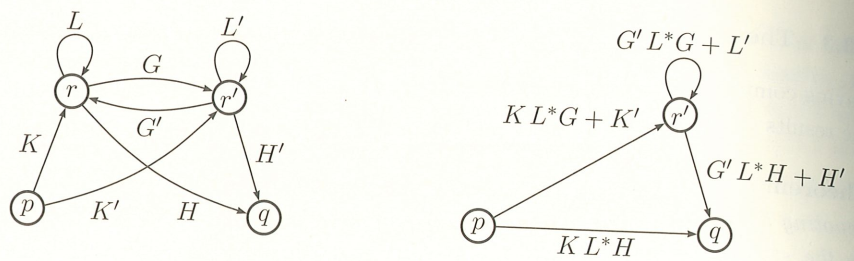
$$K'\,\left[L'^*G'\,L^*G\right]^* L'^*H' \equiv K'\,L'^*H' + K'\,L'^*G'\,L^*\left[G\,L'^*G'\,L^*\right]^* G\,L'^*H'$$

by using $\mathbf{P}$ 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

cf. Exer. 4.5

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

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. ■



(a) (b)

Figure 4.1: A step of the state elimination method

Remark 4.3 We conclude in particular from Theorem 4.2 that, since the identities **P** and **S** were obtained without using the idempotence of union in $\mathfrak{P}(A^*)$, the identities deduced from the computation of the language recognised by a fixed automaton by the state elimination method (or by algorithm MNY), with two different orders on the states, are valid even if the expressions are interpreted in a semiring that is not necessarily idempotent.

cf. also Prop. II.1.11, p. 230

This property is also the consequence of the more general result, which we shall see in Chapter III, that a rational expression obtained from an automaton describes not only the set of words accepted but also the *weight* with which each word is accepted in the automaton. □

4.4 Derivation of expressions

A (rational) expression denotes a (rational) language; these two notions describe the same object at two levels: let us say that the language is the reality and that the expression is the symbolic, or syntactic, representation of this reality. In this subsection we will bring to the level of expressions a technique which has been very successful with languages: taking the quotient.

cf. Sec. 5.2, p. 149

This construction, which we will call *derivation* to distinguish the two approaches, is of a different nature, but ensures a perfect parallel between the two levels: the quotient of a language L by a word f is a language, $f^{-1}L$, while the derivative with respect to f of an expression E that denotes L is an expression that denotes $f^{-1}L$; a rational language has only a finite number of quotients, while a rational expression, modulo certain conventions, has only a certain number of derivatives; the quotients of L are the states of the minimal automaton that recognises L, while the derivatives of E are the states of a deterministic automaton that recognises the language denoted by E. The interest of this construction is as much practical as theoretical: we shall see that, with suitable alterations, it forms the basis of powerful algorithms for computing an automaton from an expression.

4.4.1 Derivatives of an expression

We begin by defining the concept corresponding to the *constant term of a language.*

Definition 4.5 The *constant term* of a rational expression E over A, written $\mathsf{c}(\mathsf{E})$, is the function 1 or 0, computed by induction on the depth of E by the following formulas:

$$\mathsf{c}(1) = 1\ , \qquad \mathsf{c}(0) = \mathsf{c}(a) = 0 \qquad \forall a \in A\ ,$$
$$\mathsf{c}(\mathsf{E}+\mathsf{F}) = \mathsf{c}(\mathsf{E}) + \mathsf{c}(\mathsf{F})\ , \qquad \mathsf{c}(\mathsf{E}\cdot\mathsf{F}) = \mathsf{c}(\mathsf{E})\cdot\mathsf{c}(\mathsf{F}) \qquad \text{and} \qquad \mathsf{c}(\mathsf{E}^*) = 1\ . \tag{4.25}$$

We can easily verify that this definition makes sense:

Property 4.1 *The* constant term *of an expression* E *is* 1 *or* 0 *according to whether* 1_{A^*} *is in the language denoted by* E *or not.*[48] ■

Remark 4.4 It might seem simpler to take the statement of Property 4.1 as a definition and then verify the formulas (4.25) which would then become properties. It is not. The formulas (4.25) enable us to *compute effectively* $\mathsf{c}(\mathsf{E})$ – and hence $\mathsf{c}(|\mathsf{E}|)$ – from E: it is $|\mathsf{E}|$ which is obtained from E, not the other way around. □

Definition 4.6 Let E be a rational expression over A and a a letter of A. The *derivative* of E with respect to a, written $\frac{\partial}{\partial a}\mathsf{E}$, is a rational expression over A, defined recursively by the following formulas:

$$\frac{\partial}{\partial a}0 = \frac{\partial}{\partial a}1 = \frac{\partial}{\partial a}b = 0 \qquad \forall b \in A\ ,\ b \neq a$$
$$\frac{\partial}{\partial a}a = 1 \tag{4.26}$$
$$\frac{\partial}{\partial a}(\mathsf{E}+\mathsf{F}) = \frac{\partial}{\partial a}\mathsf{E} + \frac{\partial}{\partial a}\mathsf{F} \tag{4.27}$$
$$\frac{\partial}{\partial a}(\mathsf{E}\cdot\mathsf{F}) = \left[\frac{\partial}{\partial a}\mathsf{E}\right]\cdot\mathsf{F} + \mathsf{c}(\mathsf{E})\cdot\frac{\partial}{\partial a}\mathsf{F} \tag{4.28}$$
$$\frac{\partial}{\partial a}\mathsf{E}^* = \left[\frac{\partial}{\partial a}\mathsf{E}\right]\cdot\mathsf{E}^*\ . \tag{4.29}$$

The (vague) resemblance of (4.27) and (4.28) to the formulas for the derivation of a sum or product of functions could be considered as justifying the terminology.

Example 4.3 The derivatives, with respect to a and b, of the expression

$$\mathsf{E}_1 = (a+b)^*\cdot\Big(a\cdot\big(b\cdot(a+b)^*\big)\Big)$$

[48] In fact, it would be more accurate to write $\mathsf{c}(|\mathsf{E}|) = \mathsf{c}(\mathsf{E})$ but we will stick to an elementary version of the constant term of a language (*cf.* Note 25, p. 89).

are obtained by the following calculation:

$$\begin{aligned}\frac{\partial}{\partial a}\mathsf{E}_1 &= \left[\frac{\partial}{\partial a}(a+b)^*\right]\cdot\Big(a\cdot\big(b\cdot(a+b)^*\big)\Big) + \mathsf{c}\big((a+b)^*\big)\cdot\left[\frac{\partial}{\partial a}\Big(a\cdot\big(b\cdot(a+b)^*\big)\Big)\right]\\ &= \left[\frac{\partial}{\partial a}(a+b)\right]\cdot(a+b)^*\cdot\Big(a\cdot\big(b\cdot(a+b)^*\big)\Big)\\ &\qquad + \left[\frac{\partial}{\partial a}a\right]\cdot\big(b\cdot(a+b)^*\big) + \mathsf{c}(a)\cdot\left[\frac{\partial}{\partial a}\big(b\cdot(a+b)^*\big)\right]\\ &= \mathsf{E}_1 + \big(b\cdot(a+b)^*\big)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial b}\mathsf{E}_1 &= \left[\frac{\partial}{\partial b}(a+b)^*\right]\cdot\Big(a\cdot\big(b\cdot(a+b)^*\big)\Big) + \mathsf{c}\big((a+b)^*\big)\cdot\left[\frac{\partial}{\partial b}\Big(a\cdot\big(b\cdot(a+b)^*\big)\Big)\right]\\ &= \left[\frac{\partial}{\partial b}(a+b)\right]\cdot(a+b)^*\cdot\Big(a\cdot\big(b\cdot(a+b)^*\big)\Big)\\ &\qquad + \left[\frac{\partial}{\partial b}a\right]\cdot\big(b\cdot(a+b)^*\big) + \mathsf{c}(a)\cdot\left[\frac{\partial}{\partial b}\big(b\cdot(a+b)^*\big)\right] = \mathsf{E}_1\ .\end{aligned}$$ □

These computations call for two remarks.

Remarks 4.5 (i) The calculations were conducted modulo the trivial identities $\mathbf{T}$: at each step of the calculation, the expression obtained was replaced by the *equivalent reduced expression*.

cf. p. 130 (ii) The expression E_1 is 'completely bracketed', contrary to the usage which we adopted in Subsection 4.2. The reason is that the derivation of expressions *is not invariant* modulo the associativity of the product operation. □ cf. Exer. 4.10

The derivative of an expression with respect to a letter a is consistent with the (left) quotient of languages by the same letter a since we can easily verify the analogues of the formulas (4.26)–(4.29):

Properties 4.2 $\forall a \in A$, $\forall L, K \subseteq A^*$ (i) $a^{-1}(L \cup K) = a^{-1}L \cup a^{-1}K$, (ii) $a^{-1}(LK) = (a^{-1}L)\,K \cup \mathsf{c}(L)\,a^{-1}K$, (iii) $a^{-1}(L^*) = (a^{-1}L)\,L^*$. ■

From which we conclude, by induction on the depth of expressions:

Property 4.3 $\forall \mathsf{E} \in \mathsf{RatE}\,A$, $\forall a \in A$ $\quad \left|\frac{\partial}{\partial a}\mathsf{E}\right| = a^{-1}|\mathsf{E}|$. ■

Definition 4.7 Let E be a rational expression over A and g a non-empty word from A^*: that is, $g = fa$ with a in A. The *derivative* of E with respect to g, written $\frac{\partial}{\partial g}\mathsf{E}$, is the rational expression over A, defined recursively by the formulas (4.26)–(4.29) and by

$$\forall f \in A^*,\ \forall a \in A \qquad \frac{\partial}{\partial fa}\mathsf{E} = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial f}\mathsf{E}\right). \tag{4.30}$$

Example 4.3 (continued) The derivatives of E_1 with respect to words of length 2 are

$$\frac{\partial}{\partial a^2}\mathsf{E}_1 = \frac{\partial}{\partial a}[\mathsf{E}_1 + b(a+b)^*] = \frac{\partial}{\partial a}\mathsf{E}_1 + \frac{\partial}{\partial a}(b(a+b)^*) = \frac{\partial}{\partial a}\mathsf{E}_1\ ,$$

$$\frac{\partial}{\partial ab}\mathsf{E}_1 = \frac{\partial}{\partial b}[\mathsf{E}_1 + b(a+b)^*] = \frac{\partial}{\partial b}\mathsf{E}_1 + \frac{\partial}{\partial b}(b(a+b)^*) = \mathsf{E}_1 + (a+b)^*\ ,$$

$$\frac{\partial}{\partial ba}\mathsf{E}_1 = \frac{\partial}{\partial a}\mathsf{E}_1 \quad\text{and}\quad \frac{\partial}{\partial b^2}\mathsf{E}_1 = \frac{\partial}{\partial b}\mathsf{E}_1 = \mathsf{E}_1\ .$$ □

From Property 4.3 we then deduce, by induction on the length of f, the close relationship that exists between the derivative of an expression by f and the (left) quotient of a language by f:

Proposition 4.9 $\forall \mathsf{E} \in \mathsf{RatE}\,A$, $\forall f \in A^*$ $\quad \left|\frac{\partial}{\partial f}\mathsf{E}\right| = f^{-1}|\mathsf{E}|$. ■

The set of quotients of a rational language is finite; the same is not true of the set of derivatives of a rational expression, *but it is almost true*. This is what we shall now see.

4.4.2 A theorem of J. Brzozowski

We start by observing that the formulas (4.26)–(4.30) can produce an infinite number of expressions derived from a single expression E.

Example 4.3 (continued) If we calculate the derivatives of E_1 with respect to words of the form $(ab)^*$ and $(ab)^*a$, we obtain the following expressions:

$$\frac{\partial}{\partial aba}\mathsf{E}_1 = \frac{\partial}{\partial a}[\mathsf{E}_1 + (a+b)^*] = \frac{\partial}{\partial a}\mathsf{E}_1 + \frac{\partial}{\partial a}((a+b)^*) = \mathsf{E}_1 + b(a+b)^* + (a+b)^*\ ,$$

$$\frac{\partial}{\partial (ab)^2}\mathsf{E}_1 = \frac{\partial}{\partial b}\mathsf{E}_1 + \frac{\partial}{\partial b}(b(a+b)^*) + \frac{\partial}{\partial b}((a+b)^*) = \mathsf{E}_1 + (a+b)^* + (a+b)^*\ ,$$

and more generally

$$\frac{\partial}{\partial (ab)^{n-1}a}\mathsf{E}_1 = \mathsf{E}_1 + b(a+b)^* + \underbrace{(a+b)^* + \cdots + (a+b)^*}_{n-1\text{ times}},$$

$$\frac{\partial}{\partial (ab)^{n}}\mathsf{E}_1 = \mathsf{E}_1 + \underbrace{(a+b)^* + \cdots + (a+b)^*}_{n\text{ times}}\ ;$$

these expressions are all distinct. But we also see that all of them – the $\frac{\partial}{\partial (ab)^n a}\mathsf{E}_1$ on one hand and the $\frac{\partial}{\partial (ab)^n}\mathsf{E}_1$ on the other – are equivalent modulo $\mathbf{I}$. This is the property that we will now prove in general. □

Theorem 4.3 *The set of derivatives of a rational expression is finite modulo the identities* $\mathbf{A_s}$, $\mathbf{C}$ *and* $\mathbf{I}$, *and it is effectively computable.*[49]

First, let us prove an elementary property.

Lemma 4.10 *Let* $\mathcal{F}$ *be a finite set of expressions. The set of expressions formed by sums of elements of* $\mathcal{F}$ *is finite modulo* $\mathbf{A_sCI}$ *and its cardinal is bounded by* $2^{\|\mathcal{F}\|}$.

Proof. In any sum of elements of $\mathcal{F}$, we use the commutativity and associativity of the sum to regroup identical expressions, which are the same modulo **I**. There is therefore a bijection between the sums of elements of $\mathcal{F}$, where each element appears at most once, and the subsets of $\mathcal{F}$. ■

Proof of Theorem 4.3. Write $\mathbf{nd}(\mathsf{E})$ for the number of distinct derivatives of E modulo $\mathbf{A_s}$, $\mathbf{C}$, and $\mathbf{I}$. We shall show by induction on the depth of expressions that $\mathbf{nd}(\mathsf{E})$ is finite.

(a) Atoms. If E is an atomic expression, $\mathbf{nd}(\mathsf{E})$ is either 1 or 2.

(b) Sum. Formulas (4.30) and (4.27) imply

$$\forall f \in A^* \qquad \frac{\partial}{\partial f}(\mathsf{E}+\mathsf{F}) = \frac{\partial}{\partial f}\mathsf{E} + \frac{\partial}{\partial f}\mathsf{F} \ .$$

There are $\mathbf{nd}(\mathsf{E})$ possible derivatives for E and $\mathbf{nd}(\mathsf{F})$ possible derivatives for F, so

$$\mathbf{nd}(\mathsf{E}+\mathsf{F}) \leqslant \mathbf{nd}(\mathsf{E})\,\mathbf{nd}(\mathsf{F}) \ .$$

The inequality comes from the fact that some of these combinations may be identical.

(c) Product. Let $f = a_1 a_2 \cdots a_n$ be a word of length n. By expanding (4.30), we obtain

$$\frac{\partial}{\partial a_1 a_2 \cdots a_n}(\mathsf{E}\cdot\mathsf{F}) = \left[\frac{\partial}{\partial a_1 a_2 \cdots a_n}(\mathsf{E})\right]\cdot\mathsf{F} + \mathsf{c}\left(\frac{\partial}{\partial a_1 a_2 \cdots a_{n-1}}(\mathsf{E})\right)\cdot\frac{\partial}{\partial a_n}\mathsf{F} + \mathsf{c}\left(\frac{\partial}{\partial a_1 a_2 \cdots a_{n-2}}(\mathsf{E})\right)\cdot\frac{\partial}{\partial a_{n-1}a_n}\mathsf{F} + \cdots + \mathsf{c}(\mathsf{E})\cdot\frac{\partial}{\partial a_1 a_2 \cdots a_n}\mathsf{F} \ .$$

Thus, $\frac{\partial}{\partial f}(\mathsf{E}\cdot\mathsf{F})$ is the sum of $\left(\frac{\partial}{\partial f}\mathsf{E}\right)\cdot\mathsf{F}$ and at most $|f|$ derivatives of F. By Lemma 4.10 this sum, with its arbitrarily large number of terms, must, by the inductive hypothesis that $\mathbf{nd}(\mathsf{F})$ is finite, be bounded, and we have

$$\mathbf{nd}(\mathsf{E}\cdot\mathsf{F}) \leqslant \mathbf{nd}(\mathsf{E})\,2^{\mathbf{nd}(\mathsf{F})} \ .$$

[49] We do not mention the trivial identities **T** explicitly since in any case every computation on derivatives is performed modulo **T**; *cf.* Rem. 4.5 (i).

(d) Star. Equations (4.30) and (4.29) give, where a, b and c are letters in A:

$$\frac{\partial}{\partial ab}(\mathsf{E}^*) = \left[\frac{\partial}{\partial ab}\mathsf{E}\right]\cdot\mathsf{E}^* + \mathsf{c}\left(\frac{\partial}{\partial a}\mathsf{E}\right)\cdot\frac{\partial}{\partial b}\mathsf{E}\cdot\mathsf{E}^* \ ,$$

$$\frac{\partial}{\partial abc}(\mathsf{E}^*) = \left[\frac{\partial}{\partial abc}\mathsf{E}\right]\cdot\mathsf{E}^* + \mathsf{c}\left(\frac{\partial}{\partial ab}\mathsf{E}\right)\cdot\frac{\partial}{\partial b}\mathsf{E}\cdot\mathsf{E}^* + \mathsf{c}\left(\frac{\partial}{\partial a}\mathsf{E}\right)\cdot\frac{\partial}{\partial bc}\mathsf{E}\cdot\mathsf{E}^* + \mathsf{c}\left(\frac{\partial}{\partial a}\mathsf{E}\right)\cdot\mathsf{c}\left(\frac{\partial}{\partial b}\mathsf{E}\right)\cdot\frac{\partial}{\partial c}\mathsf{E}\cdot\mathsf{E}^* \ .$$

We conclude that in general $\frac{\partial}{\partial f}(\mathsf{E}^*)$ is a sum with (at most $2^{|f|-1}$) terms of the form $\frac{\partial}{\partial g}\mathsf{E}\cdot\mathsf{E}^*$ (where g is a suffix of f) and hence:

$$\mathbf{nd}(\mathsf{E}^*) \leqslant 2^{\mathbf{nd}(\mathsf{E})} \ .$$

The assertion that D_E, the set of derivatives of E modulo $\mathbf{A_sCI}$, is *effectively computable*, does not follow directly from $\mathbf{nd}(\mathsf{E})$ being finite, but is easily deducible. We note first that computations modulo $\mathbf{A_sCI}$ are effective: by, for example, ordering the expressions lexicographically we can arrange that each expression is equivalent to a *unique expression* of minimal length.

If we then write $D_\mathsf{E}^{(k)}$ for the set (modulo $\mathbf{A_sCI}$) of derivatives of E with respect to all the words in A^* of length less than or equal to k, we obtain an increasing sequence of sets of expressions. If for some l, $D_\mathsf{E}^{(l)} = D_\mathsf{E}^{(l+1)}$ then, by a chain of reasoning which is now familiar to us *cf. Prop. 3.12, p. 115*

$$D_\mathsf{E}^{(l+2)} = \left\{\frac{\partial}{\partial a}\mathsf{F}\;\middle|\;\mathsf{F}\in D_\mathsf{E}^{(l+1)}\quad a\in A\right\} = \left\{\frac{\partial}{\partial a}\mathsf{F}\;\middle|\;\mathsf{F}\in D_\mathsf{E}^{(l)}\quad a\in A\right\} = D_\mathsf{E}^{(l+1)} = D_\mathsf{E}^{(l)} \ ,$$

and we see that $D_\mathsf{E}^{(l+p)} = D_\mathsf{E}^{(l)}$ for all p and hence $D_\mathsf{E} = D_\mathsf{E}^{(l)}$. ■

4.4.3 Derivative automaton

Theorem 4.3 gives us as a corollary an effective procedure for computing from an expression E a finite deterministic automaton which accepts the language denoted by E.

Definition 4.8 Let E be a rational expression over A and D_E the set of its derivatives modulo $\mathbf{A_sCI}$. The automaton $\mathcal{B}_\mathsf{E} = \langle D_\mathsf{E}, A, \delta_\mathsf{E}, \{\mathsf{E}\}, T_\mathsf{E}\rangle$ defined by

$$\forall \mathsf{F}\in D_\mathsf{E},\ \forall a\in A \qquad (\mathsf{F},a)\delta_\mathsf{E} = \frac{\partial}{\partial a}\mathsf{F} \qquad \text{and} \qquad T_\mathsf{E} = \{\mathsf{F}\in D_\mathsf{E} \mid \mathsf{c}(\mathsf{F}) = 1\}$$

is a *finite* deterministic automaton effectively computable from E. It is called the *derivative automaton* of E. □

From Proposition 4.9 and Proposition 3.7 there follows:

Proposition 4.11 *The automaton* $\mathcal{B}_\mathsf{E}$ *recognises* $|\mathsf{E}|$. ■

Example 4.3 (continued) Consider $\mathsf{E}_1 = (a+b)^*ab(a+b)^*$ once more and note

$$\mathsf{E}_2 = \frac{\partial}{\partial a}\,\mathsf{E}_1 = \mathsf{E}_1 + b(a+b)^* \;, \qquad \mathsf{E}_3 = \frac{\partial}{\partial ab}\,\mathsf{E}_1 = \mathsf{E}_1 + (a+b)^* \;,$$

$$\mathsf{E}_4 = \frac{\partial}{\partial aba}\,\mathsf{E}_1 = \mathsf{E}_1 + b(a+b)^* + (a+b)^* \;.$$

From the above calculations, we deduce that $D_{\mathsf{E}_1} = \{\mathsf{E}_1, \mathsf{E}_2, \mathsf{E}_3, \mathsf{E}_4\}$, and the automaton $\mathcal{B}_{\mathsf{E}_1}$ shown in Figure 4.2. □

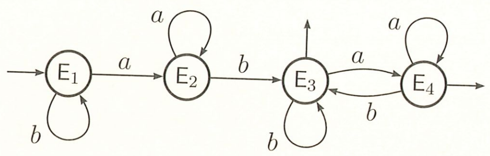


Figure 4.2: The automaton $\mathcal{B}_{\mathsf{E}_1}$

We see from this example that the derivative automaton of an expression E is not, in general, equal (isomorphic) to the automaton of quotients, the minimal automaton of |E|. We will return in the next section to the problem of the computation of an automaton from a rational expression.

Exercises

4.7 Verify Property 4.1.

4.8 Verify Property 4.2.

• 4.9 Calculate the derivative automaton of the following expressions (we write EFG for E(FG)):

(a) $(a^*b + bb^*a)^*$; (b) $a^*b(ab^*a + ba^*b)^*ba^* + a^*$; (c) $(a + b(ab^*a)^*b)^*$;
(d) $a^*aa^*aa^*$.

• 4.10 **Derivation and associativity of product.**

(a) Calculate the derivative automaton of $\mathsf{E}'_1 = \Big(\big((a+b)^* \cdot a\big) \cdot b\Big) \cdot (a+b)^*$.

(b) Do the same for $\mathsf{E}'_2 = \Big(\big((a^* \cdot a) \cdot a^*\big) \cdot a\Big) \cdot a^*$.

(c) And again for $\Big((a \cdot b) \cdot \big(c \cdot (a \cdot b)^*\big)\Big)$ and $\Big(a \cdot \Big(b \cdot \big(c \cdot (a \cdot b)^*\big)\Big)\Big)$.

(d) Comment and draw conclusion.

DEEPER & BROADER

It is rather arbitrary to classify topics in a domain as principal and complementary, and anyone whose speciality or favourite tool has been relegated to a seemingly inferior category will take it as a mark of bad taste or want of discernment. There is no question of hierarchy or value judgement in our mode of presentation, but rather of a change of tone. In the preceding sections, I attempted to be comprehensive and didactic; in those which follow I will allow myself more personal choices, a more elliptic style, and references to other works and articles.

As promised at the start of the chapter, this part comprises four sections. The first two can be seen as two completely different variations on Kleene's Theorem. Section 5 presents the construction of automata whose interest lies in their translation into algorithms which are used ubiquitously. Section 6 tackles one of the hardest problems in the study of rational languages, whose solution still holds many mysteries; the fascination that it excites is of a rather theoretical order. Section 7 presents different models of 'machines' equivalent to finite automata, and Section 8 describes, in the form of problems, several properties of rational languages.

5 From expressions to automata

The computation of an automaton from a rational expression – that is, the algorithmic version of one direction of Kleene's Theorem – is a problem which has been tackled many times. Several works deal only with this aspect of automata. This is because it was the central question in one of the first applications of finite automata: the lexical analysis of programming languages and searching for sequences or patterns, which are abilities used by all word processors and many other utility programs.

In the first subsection we shall revisit the construction of a standard automaton from an expression. The variety of methods that all lead, directly or indirectly, to the same result gives this object a fundamental character. The second subsection describes another construction based on a variant of the notion of derivation of an expression, which produces a different result from the standard automaton. The last subsection is devoted to the problem of finding a word in a text, which comes down to considering some rather special rational expressions.

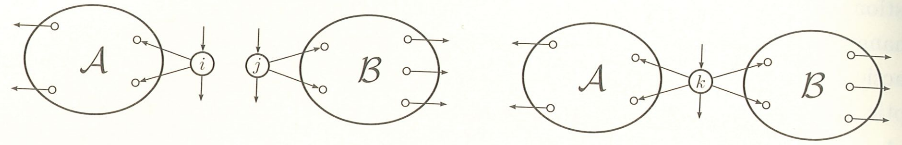
5.1 The standard automaton of an expression

We discussed in Section 2.2 how the proof of Proposition 2.2 not only demonstrates that Rec A^* is rationally closed but that it is itself an *algorithm* that produces an *cf.* p. 92 and p. 94

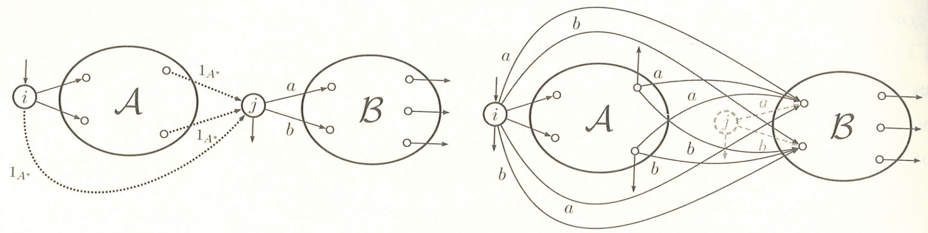
automaton from a rational expression.[50] The automaton thus constructed clearly depends on the method used: we imagined that it would be a normalised or standard automaton.

cf. Exam. 2.3, p. 90

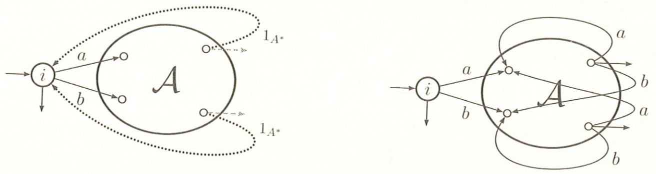
The use of normalised automata leads to an almost absurd explosion in the number of states, which does not invalidate the idea of a normalised automaton itself but does make it unsuitable for making an expression into an automaton. We shall see that standard automata, on the other hand, are reasonably compact, and susceptible to a variety of interpretations that testify to the canonical nature of the link they create between rational expressions and automata.



(a) for union



(b) for product



(c) for star

Figure 5.1: Construction of standard automata

5.1.1 Direct construction

cf. Fig. 2.7, p. 93 and Fig. 2.8, p. 94

The two constructions of standard automata for the product and star of languages recognised by standard automata are shown again in Figure 5.1(b) and (c); the construction for the union consists of simply merging the initial states (Figure 5.1(a)). Combined with the trivial observation that a single letter and the empty word are

[50] But at the time we had not yet given a precise definition of rational expressions.

both languages recognised by standard automata, they enable us to associate, with every rational expression E, a corresponding standard automaton $\mathcal{S}_{\mathsf{E}}$, unique and well defined.

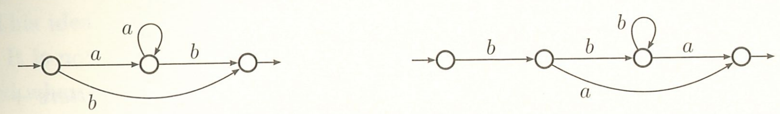
Definition 5.1 The *literal length* of an expression E, written $\ell(\mathsf{E})$, is the number of occurrences of letters of A in E. □

We observe that $\ell((\mathsf{E}^*)) = \ell(\mathsf{E})$ and that $\ell((\mathsf{E}+\mathsf{F})) = \ell((\mathsf{E}\cdot\mathsf{F})) = \ell(\mathsf{E}) + \ell(\mathsf{F})$. These formulas, combined with an obvious induction, allow us to show:

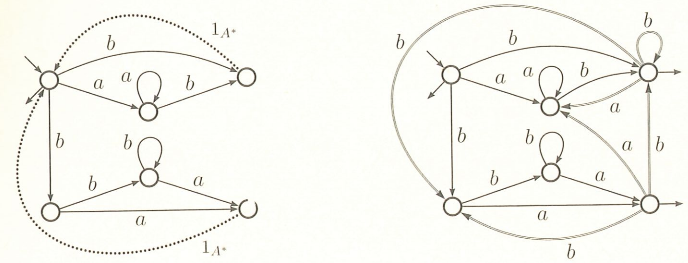
Proposition 5.1 *The standard automaton $\mathcal{S}_{\mathsf{E}}$ associated with the rational expression E has $\ell(\mathsf{E})+1$ states.* ■

cf. p. 94 and Exer. 2.13

Example 2.5 anticipated this proposition. Figure 5.2 shows the construction of the standard automaton of the expression $\mathsf{E}_2 = (a^*b + bb^*a)^*$.



(a) standard automata for a^*b and bb^*a



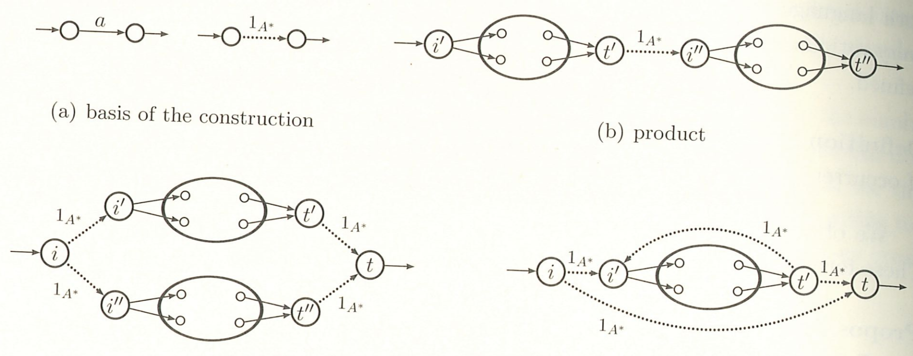
(b) Last step: star of a^*b+bb^*a. The transitions generated by the removal of spontaneous transitions are drawn with double lines.

Figure 5.2: Construction of $\mathcal{S}_{\mathsf{E}_2}$

5.1.2 Thompson's construction

Another way to construct an automaton from a rational expression is based on the use of spontaneous transitions and produces, in fact, an automaton with spontaneous transitions. The method is due to K. Thompson [243]. Conceived to be directly implementable as a program, it produces a representation that is most efficient for scanning text. Figure 5.3 shows the construction, which allows every rational expression E to be associated with a unique, well-defined automaton with spontaneous transitions $\mathcal{T}_{\mathsf{E}}$. Figure 5.4 shows the construction applied to the expression $\mathsf{E}_2 = (a^*b + bb^*a)^*$.

Notice that this construction does not encode the associativity of union, so $(\mathsf{E}+\mathsf{F})+\mathsf{G}$ and $\mathsf{E}+(\mathsf{F}+\mathsf{G})$ produce two different automata. We also observe the following



(a) basis of the construction (b) product (c) union (d) star

Figure 5.3: Thompson's construction

properties:

Property 5.1 *If the rational expression* E *comprises n symbols (letters + operators), the automaton* $\mathcal{T}_{\mathsf{E}}$ *has at most* $2n$ *states.* ■

Property 5.2 *Every state of* $\mathcal{T}_{\mathsf{E}}$ *is the source (resp. destination) of one or two transitions, except for the final state (resp. the initial state).* ■

It is remarkable that this construction corresponds to another way of defining the standard automaton, a result whose proof we shall leave to an exercise.

Proposition 5.2 *Let* E *be a rational expression. The automaton obtained from* $\mathcal{T}_{\mathsf{E}}$ *by* backward closure *is equal to* $\mathcal{S}_{\mathsf{E}}$. ■

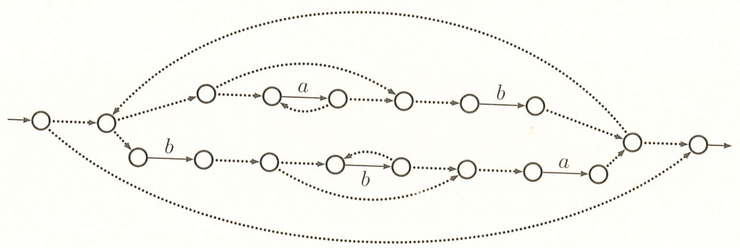


Figure 5.4: The automaton $\mathcal{T}_{\mathsf{E}_2}$

Exercises

• 5.1 Compute the standard automaton of the expression $(a + b(ab^*a)^*b)^*$.

5.2 Verify Proposition 5.1.

5.3 Verify Proposition 5.2.

5.2 The derived term automaton

We have seen that the derivation of rational expressions is also a way of associating an automaton with an expression. This automaton has the disadvantage of being deterministic, which is not a defect in itself (on the contrary) but which raises the possibility of an exponential explosion. The process of computation itself assumes that expressions can be compared for equality modulo the identities $\mathbf{A_sCI}$, which is algorithmically expensive.

cf. Def. 4.8, p. 143
cf. Th. 4.3, p. 142

A modification to the definition of derivation will overcome these disadvantages: first, the expressions computed will no longer have to be considered modulo any identities, and second, the automata produced will be no larger than the standard automata of the expressions.

The idea underlying this transformation is that the derivative of a sum, instead of producing an expression which is the sum of the derivatives of the terms of the sum, produces a *set of expressions* that is the set of derivatives of the terms of the sum. This idea may seem baroque and artificial, excused only by the quality of the result. It is nothing of the sort: it has a solid theoretical foundation and will be fully justified when we have defined the derivation of *weighted expressions* in Chapter III.

cf. Sec. III.4.2, p. 443

5.2.1 Derived terms

Definition 5.2 Let E be a rational expression over A and a a letter in A. The $\mathbb{B}$-*derivation* of E with respect to a, written $\frac{\partial'}{\partial' a}\mathsf{E}$, is *a set of rational expressions over* A, defined recursively by the following formulas:

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

$$\frac{\partial'}{\partial' a}(\mathsf{E}+\mathsf{F}) = \left\{\frac{\partial'}{\partial' a}\mathsf{E}\right\} \cup \left\{\frac{\partial'}{\partial' a}\mathsf{F}\right\} \tag{5.1}$$

$$\frac{\partial'}{\partial' a}(\mathsf{E}\cdot\mathsf{F}) = \left\{\left[\frac{\partial'}{\partial' a}\mathsf{E}\right]\cdot\mathsf{F}\right\} \cup \left\{\mathsf{c}(\mathsf{E})\ \frac{\partial'}{\partial' a}\mathsf{F}\right\} \tag{5.2}$$

$$\frac{\partial'}{\partial' a}(\mathsf{E}^*) = \left[\frac{\partial'}{\partial' a}\mathsf{E}\right]\cdot\mathsf{E}^* \tag{5.3}$$

As for derivation, all the calculations are performed modulo the trivial identities $\mathbf{T}$. Furthermore, the induction implied by (5.2) and (5.3) must be realised by distributing $\mathbb{B}$-derivation over union:

cf. Rem. 4.5, p. 140

$$\frac{\partial'}{\partial' a}\left(\bigcup_{i\in I}\mathsf{E}_i\right) = \bigcup_{i\in I}\frac{\partial'}{\partial' a}\mathsf{E}_i \ ,$$

and product over union:

$$\left(\bigcup_{i\in I}\mathsf{E}_i\right)\cdot\mathsf{F} = \bigcup_{i\in I}(\mathsf{E}_i\cdot\mathsf{F}) \ . \tag{D}$$

Definition 5.3 Let E be a rational expression over A and g a non-empty word from A^*: that is, $g = f a$ with a in A. The $\mathbb{B}$-*derivation* of E with respect to g, written $\frac{\partial'}{\partial' g}\mathsf{E}$, is a set of rational expressions over A, defined recursively by the formulas (5.1)–(5.3), and by

$$\forall f \in A^*,\ \forall a \in A \qquad \frac{\partial'}{\partial' fa}\mathsf{E} = \frac{\partial'}{\partial' a}\left(\frac{\partial'}{\partial' f}\mathsf{E}\right) . \tag{5.4}$$

We will call any rational expression which belongs to a set $\frac{\partial'}{\partial' g}\mathsf{E}$ for some g in A^* a *derived term* of E. □

One thing that emerges from Definitions 5.2 and 5.3 is that the computation of derived terms of an expression E corresponds to taking the $\mathbb{B}$-derivation of some sub-expressions of derivatives of E.

Example 5.1 The $\mathbb{B}$-derivation with respect to a and b of the expression

$$\mathsf{E}_1 = (a+b)^* \cdot \left(a \cdot \left(b \cdot (a+b)^*\right)\right)$$

yields $\frac{\partial'}{\partial' a}\mathsf{E}_1 = \{\mathsf{E}_1, b\cdot(a+b)^*\}$ and $\frac{\partial'}{\partial' b}\mathsf{E}_1 = \mathsf{E}_1$. The derived terms of E_1 are therefore $\{\mathsf{E}_1, b\cdot(a+b)^*, (a+b)^*\}$. □

Having acknowledged that the language denoted by a set of rational expressions is the union of the languages denoted by the expressions, we immediately conclude from Properties 4.2 and 4.3 and from Proposition 4.9 the corresponding result for $\mathbb{B}$-derivation:

cf. p. 140

Proposition 5.3 $\forall \mathsf{E} \in \mathsf{RatE}\,A^*,\ \forall f \in A^*$ $\quad \left|\frac{\partial'}{\partial' f}\mathsf{E}\right| = f^{-1}|\mathsf{E}|$. ■

5.2.2 A theorem of V. Antimirov

The interest of $\mathbb{B}$-derivation (with respect to the derivation seen in Section 4.4) is that it enables us to consider derived terms directly, rather than via the identities $\mathbf{A_s CI}$.

Theorem 5.1 *A rational expression* E *has at most* $\ell(\mathsf{E})$ *derived terms.*

We shall prove a more precise proposition, but with a different formulation: instead of considering $\mathbb{B}$-derivation with respect to all the words in A^*, we shall look just at $\mathbb{B}$-derivation with respect to the letters of A; in return we shall require global closure.

Theorem 5.2 *Let* E *be in* $\mathsf{RatE}\,A^*$. *The derived terms of* E *form a set* P *of expressions* $\{\mathsf{K}_p\}_{p\in P}$, *such that both* $\|P\| \leqslant \ell(\mathsf{E})$, *and for each letter* a *in* A, *there exists a subset* $Q^{(a)}$ *of* P *and subsets* $\{H_p^{(a)}\}_{p\in P}$ *of* P, *such that*

(i) $\frac{\partial'}{\partial' a}\mathsf{E} = \bigcup_{p\in Q^{(a)}} \mathsf{K}_p$ *and* (ii) $\forall p \in P \quad \frac{\partial'}{\partial' a}\mathsf{K}_p = \bigcup_{j\in H_p^{(a)}} \mathsf{K}_j$.

Proof. A set P which satisfies (i) and (ii), and where $\|P\| \leqslant \ell(\mathsf{E})$, is constructed by induction on the depth of the expression E. The fact that it constitutes the set of derived terms is shown by induction on the length of words at each step of the preceding induction. The proposition is clearly true for $\mathsf{E} = 1$ and $\mathsf{E} = a$, for each a in A (the expression 0 is not counted as a derived term).

(a) If it is true for E and F, it is true for $(\mathsf{E}+\mathsf{F})$. In fact, we have

$$\frac{\partial'}{\partial' a}(\mathsf{E}+\mathsf{F}) = \left\{\frac{\partial'}{\partial' a}\mathsf{E}, \frac{\partial'}{\partial' a}\mathsf{F}\right\} = \bigcup_{p\in Q^{(a)}} \mathsf{K}_p \cup \bigcup_{s\in R^{(a)}} \mathsf{L}_s$$

using the obvious notation. The set $\bigcup_{p\in P}\mathsf{K}_p \cup \bigcup_{s\in S}\mathsf{L}_s$ satisfies condition (ii) and is clearly the set of derived terms of $(\mathsf{E}+\mathsf{F})$.

(b) If it is true for E and F, it is true for $(\mathsf{E}\cdot\mathsf{F})$. It follows that

$$\frac{\partial'}{\partial' a}(\mathsf{E}\cdot\mathsf{F}) = \left(\left[\frac{\partial'}{\partial' a}\mathsf{E}\right]\cdot\mathsf{F}\right) \cup \mathsf{c}(\mathsf{E})\,\frac{\partial'}{\partial' a}\mathsf{F} = \bigcup_{p\in Q^{(a)}} (\mathsf{K}_p\cdot\mathsf{F}) \cup \bigcup_{s\in R^{(a)}} \mathsf{c}(\mathsf{E})\,\mathsf{L}_s ,$$

and, for each p in P, $\quad \frac{\partial'}{\partial' a}(\mathsf{K}_p\cdot\mathsf{F}) = \bigcup_{j\in H_p^{(a)}} (\mathsf{K}_j\cdot\mathsf{F}) \cup \bigcup_{s\in R^{(a)}} \mathsf{c}(\mathsf{K}_p)\,\mathsf{L}_s$,

from which we see that the set $\bigcup_{p\in P}(\mathsf{K}_p\cdot\mathsf{F}) \cup \bigcup_{s\in S}\mathsf{L}_s$ satisfies condition (ii) and, by induction on the length of words, that it is the set of derived terms of $(\mathsf{E}\cdot\mathsf{F})$.

(c) If it is true for E, it is true for (E^*). We have

$$\frac{\partial'}{\partial' a}(\mathsf{E}^*) = \left(\left[\frac{\partial'}{\partial' a}\mathsf{E}\right]\cdot(\mathsf{E}^*)\right) = \bigcup_{p\in Q^{(a)}} (\mathsf{K}_p\cdot\mathsf{E}^*) ,$$

and, for each p in P, $\quad \frac{\partial'}{\partial' a}(\mathsf{K}_p\cdot\mathsf{E}^*) = \bigcup_{j\in H_p^{(a)}} (\mathsf{K}_j\cdot\mathsf{E}^*) \cup \bigcup_{j\in Q^{(a)}} \mathsf{c}(\mathsf{K}_p)\,(\mathsf{K}_j\cdot\mathsf{E}^*)$,

from which we see that the set $\bigcup_{p\in P}(\mathsf{K}_p\cdot\mathsf{E}^*)$ satisfies condition (ii) and, by induction on the length of words, that it is the set of derived terms of (E^*).

At each of these three steps, we verify that the number of derived terms computed is really less than the literal length of the expression. ■

The statement of Theorem 5.2 can itself be seen as the definition of an automaton.

Definition 5.4 Let E be in $\mathsf{RatE}\,A^*$. Let P_E be the set of its derived terms and P'_E the union of P_E and E (that is, $P'_\mathsf{E} = P_\mathsf{E}$ if E is a derived term of itself). Set $\mathsf{E} = \mathsf{K}_i$ and $Q^{(a)} = H_i^{(a)}$ (and we have that $i \in P'$). The automaton

$$\mathcal{A}_\mathsf{E} = \langle P'_\mathsf{E}, A, \tau_\mathsf{E}, \{\mathsf{E}\}, U_\mathsf{E}\rangle$$

defined by $\quad \forall p \in P'_\mathsf{E},\ \forall a \in A \quad (\mathsf{K}_p, a)\tau_\mathsf{E} = \{\mathsf{K}_q \mid q \in H_p^{(a)}\}$ and

$$U_\mathsf{E} = \{\mathsf{K}_p \in P'_\mathsf{E} \mid \mathsf{c}(\mathsf{K}_p) = 1_{A^*}\}$$

is called the *derived term automaton* of E. □

From Proposition 5.3 it follows that:

cf. also Propty 3.7, p. 112

Proposition 5.4 *Let* E *be in* $\mathsf{Rat}\mathsf{E}\,A^*$. *The derived term automaton of* E *recognises the language denoted by* E: $|\mathcal{A}_\mathsf{E}| = |\mathsf{E}|$. ■

Example 5.1 (continued) Consider again $\mathsf{E}_1 = (a+b)^* \cdot \big(a \cdot (b \cdot (a+b)^*)\big)$. The derived terms of E_1 are E_1, $b(a+b)^*$, and $(a+b)^*$, and the automaton of derived terms of E_1 is shown in Figure 5.5; we recognise it as the automaton $\mathcal{A}_1$. □

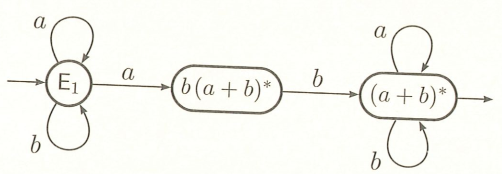


Figure 5.5: The automaton $\mathcal{A}_{\mathsf{E}_1}$

With no further assumption about the expression, the derived term automaton (its determinisation in fact) and the derivative automaton have no specific relationship as can be seen via simple examples (see below). We leave to Chapter II the description of the link of the derived term automaton with the standard automaton. *cf. Exer. II.3.8, p. 264*

Exercises

- 5.4 Compute the derived term automaton of the expression $\mathsf{E}_2 = (a^*b + bb^*a)^*$.
- 5.5 Do the same for the expression *cf. Exer. 2.15, p. 101 and Fig. 6.1 (c), p. 158*
  $\mathsf{E}_3 = a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b + a(ba^*b)^*a)^*a(ba^*b)^*ba^*$.
- 5.6 Verify from the expressions $\mathsf{F}_1 = a(a+b)^* + b(b+a)^*$ and $\mathsf{F}_2 = (aa+aaa)^*$ that the determinisation of the derived term automaton may be larger, or smaller, than the derivative automaton.

5.3 String matching

The problem of finding a word in a text appears in different forms in various domains. This type of algorithm is used for online search engines, for the search and replace function of word processors, in spelling checkers which must look words up in dictionaries, as well as in lexical analysers, which, as the first pass of a compiler, break the stream of characters forming a program into a sequence of lexemes (keywords, identifiers *etc.*). Syntactic analysis and translation into another language are performed by other parts of the compiler.

So many situations, so many different algorithms, but the fundamental problem remains of finding occurrences of words belonging to a finite set X in a text written in an alphabet A; that is, to recognise the language A^*X, a rational language, and hence to construct the corresponding automaton. Here we will only study the case where X is reduced to a single word. The case where X is an arbitrary finite set gives rise to standard generalisations.[51]

[51] *Cf.* Notes and references section. §5.3.1 is adapted from [183], and §5.3.2 from [23, Ch. 10].

5.3.1 Automaton for finding a word

Let $f = a_1 a_2 \cdots a_n$ be a word in A^* (with the a_i in A) and let $\mathcal{A}_f$ be the automaton below which recognises the language A^*f; we write P_f for the set of *prefixes* of f:

$$P_f = \{1_{A^*}, a_1, a_1 a_2, \ldots, a_1 a_2 \cdots a_{n-1}, a_1 a_2 \cdots a_{n-1} a_n\}$$

The set of states of $\mathcal{A}_f$ can be identified with P_f (via the identification between the integer i and the prefix $a_1 a_2 \cdots a_i$). We write $\mathcal{D}_f$ for the determinisation of $\mathcal{A}_f$. *A priori*, the set of states of $\mathcal{D}_f$ is a subset of the *set of subsets* of P_f. We will show that the set of states of $\mathcal{D}_f$ can also be identified with P_f.

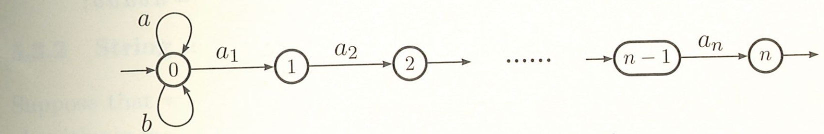


Figure 5.6: The automaton $\mathcal{A}_f$

Let w be an arbitrary word in A^*. A computation $0 \xrightarrow{w} k$ in $\mathcal{A}_f$ necessarily decomposes into

$$0 \xrightarrow{u} 0 \xrightarrow{a_1} 1 \xrightarrow{a_2} 2 \cdots \xrightarrow{a_k} k \tag{5.5}$$

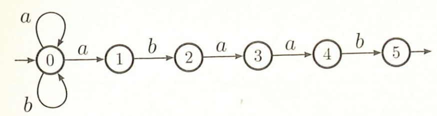
where $w = u a_1 a_2 \cdots a_k$. Conversely, if $a_1 a_2 \cdots a_k$ is a suffix of w, (5.5) is a computation in $\mathcal{A}_f$ with label w.

Let v be the *longest suffix* of w which is an element of P_f; that is, which is a prefix of f: the suffixes of w that are prefixes of f are the suffixes of v which are prefixes of f. Conversely, each element v of P_f takes $\mathcal{D}_f$ to the state which is the set of suffixes of v that belong to P_f and of which v is the greatest element: there is a bijection between the states of $\mathcal{D}_f$ and P_f, and $\mathcal{D}_f$ has $|f|+1$ states (like $\mathcal{A}_f$).

Furthermore, as $\mathcal{A}_f$ is co-deterministic and co-accessible, $\mathcal{D}_f$ is the *minimal automaton* of A^*f. *cf. Prop. 3.13, p. 116*

Example 5.2 The automaton $\mathcal{A}_{abaab}$ and the construction of $\mathcal{D}_{abaab}$ by the subset construction are shown in Figure 5.7. □

The automaton $\mathcal{A}_{abaab}$:



The table for the determinisation of $\mathcal{A}_{abaab}$:

	{0}	{0,1}	{0,2}	{0,1,3}	{0,1,4}	{0,2,5}
a	{0,1}	{0,1}	{0,1,3}	{0,1,4}	{0,1}	{0,1,3}
b	{0}	{0,2}	{0}	{0,2}	{0,2,5}	{0}

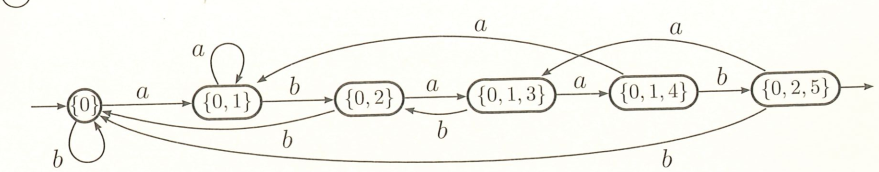


Figure 5.7: The automaton $\mathcal{D}_{abaab}$

The automaton $\mathcal{D}_f$ 'finds' the word f in the sense that if a word t in A^* (the 'text') is 'read' (from left to right) by $\mathcal{D}_f$, each letter of t which takes $\mathcal{D}_f$ to its unique final state is the last letter of an occurrence of a factor f in t, even if the factor f overlaps other preceding occurrences. It is not quite enough to have defined $\mathcal{D}_f$; we must also show how to compute it from f without just reusing the subset construction.

Let us start with a definition: the *border* of a non-empty word u is the longest *proper* prefix of u which is also a suffix of u. Thus, ab is the border of $abaab$, and $abab$ is the border of $ababab$.[52] Then $\beta \colon A^* \to A^*$ is the function which takes each word to its border (β is undefined for 1_{A^*}). From this definition, and using the preceding notations, it follows that for all p in P_f, $(p)\beta$ is the longest proper suffix of p which is in P_f. The following table gives the values of β for $f = abaab$:

p	a	ab	aba	$abaa$	$abaab$
$(p)\beta$	1_{A^*}	1_{A^*}	a	a	ab

Property 5.3 *The transition function of $\mathcal{D}_f$ is defined by the recursive equation*

$$\forall a \in A,\ \forall p \in P_f,\ p \neq 1_{A^*} \qquad p \cdot a = \begin{cases} pa & \text{if } pa \in P_f \\ (p)\beta \cdot a & \text{otherwise} \end{cases}$$

and the initial conditions

$$\forall a \in A \qquad 1_{A^*} \cdot a = \begin{cases} a & \text{if } a \in P_f \\ 1_{A^*} & \text{otherwise .} \end{cases}$$

Proof. According to the foregoing, (p, a, q), where p is in P_f and a in A, is a transition of $\mathcal{D}_f$ if and only if q is the longest suffix of pa which is in P_f; that is, which is a prefix of f.

Let $p = 1_{A^*}$, then if $a \in P_f$, $q = a$, and if $a \notin P_f$, $q = 1_{A^*}$.

Suppose now that $|p| > 0$. If $pa \in P_f$ then $q = pa$. If $pa \notin P_f$, let q be the longest suffix of pa which is a prefix of f. Suppose that q is strictly longer than $(p)\beta a$; then $q = ta$ and t is a suffix of p, prefix of f and strictly longer than $(p)\beta$: this contradicts the definition of $(p)\beta$. Thus q is a *suffix* of $(p)\beta a$; it is the longest suffix of $(p)\beta a$ which is a prefix of f: $q = (p)\beta \cdot a$. ■

Property 5.4 *The function β can itself be computed by an analogous recursive equation*

$$\forall a \in A,\ \forall p \in P_f,\ p \neq 1_{A^*} \qquad (pa)\beta = \begin{cases} (p)\beta a & \text{if } (p)\beta a \in P_f \\ ((p)\beta a)\beta & \text{otherwise} \end{cases}$$

and the initial conditions

$$\forall a \in A \qquad (a)\beta = 1_{A^*} \ .$$

[52] In some works, *a* border of u is a factor of u which is both a prefix and a suffix; *the* border of u is then the longest such (in [68]) or *maximal border* of u (in [23]).

Proof. By definition of β, $(a)\beta = 1_{A^*}$ for all a in A. We therefore must compute, for p and pa in P_f, $s = (pa)\beta$, the longest proper suffix of pa which is a prefix of f. If $(p)\beta a$ is a suffix of s, $s = ta$, then t is a proper suffix of p and a prefix of f, and t cannot be longer than $(p)\beta$, hence $t = (p)\beta$ and $s = (p)\beta a$. If not, s is a proper suffix of $(p)\beta a$, in fact the longest proper suffix of $(p)\beta a$ that is a prefix of f: $s = ((p)\beta a)\beta$ by definition. ■

We conclude from the above that we can compute all the $(p)\beta$, for p in P_f, in $O(|f|)$, then all the $p \cdot a$, for p in P_f and a in A, in $O(\|A\|\,|f|)$. Finding f in t then corresponds to reading t with the deterministic automaton $\mathcal{D}_f$ and therefore has a complexity of $O(|t|)$.

5.3.2 String matching with a sliding window

Suppose that we wish to solve the problem of finding a word in a text in a purely algorithmic manner, without needing to keep the automaton model in mind.[53] The idea that no doubt springs to mind is to imagine the text t written on a tape in the classic manner, an array whose ith square is occupied by the ith letter t_i of t; the word f is written in a window (another table), which is slid along the tape (*cf.* Figure 5.8).

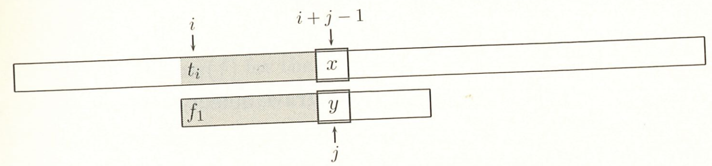


Figure 5.8: Sliding a window along a text

The first square of the window is in front of the ith square of t and we compare f_1 and t_i; if they are equal, we compare f_2 and t_{i+1}, then if they too are equal, f_3 and t_{i+2}, and so on. There are two possible outcomes. We may reach the end of the window: all the comparisons have succeeded and the word f indeed occurs in t starting at position i. Alternatively the jth comparison failed: f_j and t_{i+j-1} are different and f is not a factor of t starting at position i. In both cases, we slide the window along and start again. The question is how far to slide the window.

If we slide one position each time we have the naive algorithm; it is slow and makes $O(|f|\,|t|)$ comparisons, but it will definitely find all occurrences of f in t.

We can be cleverer and take into account what we learn about t from successful comparisons. If we have found the prefix p of f in t starting at position i, we can slide the window in such a way that we do not need to retest any of the letters of t that we have already tested. In the new position, the window should overlap the prefix p by a factor q which is the longest prefix of p that is also a suffix of p: q is the *border* of p

[53] *There are those minds whose agile train of thought is by lacunae made fragile and short* (with apologies to Nicolas Boileau).

(*cf.* Figure 5.9). This algorithm is known as the Morris–Pratt algorithm; we can show that the number of comparisons is $O(|t|)$ and that the computation of the distance to slide at each step (performed once as an initialisation step) is $O(|f|)$.

We can improve this algorithm by noticing that the first letter z of f to test after sliding the window must be different from the y that caused the slide (see Figure 5.9). We thus obtain a sliding function that is always at least as good as the previous one, which improves the efficiency of the resulting algorithm, known as the Knuth–Morris–Pratt or KMP algorithm (henceforth simply KMP). It is notable that the computation of the sliding function in this case is exactly equal to that of the transition function of the automaton $\mathcal{D}_f$: modelling it as an automaton turns this most subtle algorithm into a canonical construction, which is an important merit of modelling.[54]

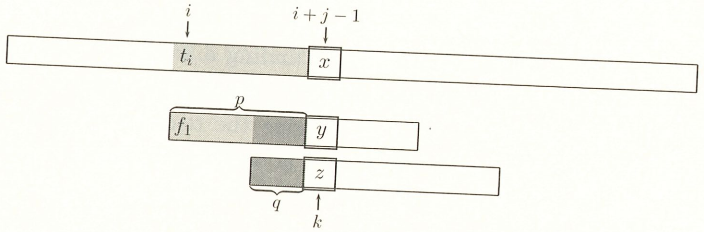


Figure 5.9: Sliding the window over the text intelligently

5.3.3 Implementation with a default successor

It remains to note that the automaton $\mathcal{D}_f$ has a grave defect compared with KMP. Its size, and therefore the number of operations required to describe it, is proportional to the number of letters in the alphabet A, which becomes prohibitive when we go from the alphabet used in our examples to the ASCII alphabet of 256 letters that is actually used by word processors and compilers.[55] (The initial phase of KMP associates with each position of the word f the distance to slide the window on a failed comparison; its computation and the amount of storage it requires is $O(|f|)$, and is independent of the size of the alphabet.)

However, the amount of essential, of 'useful', information encoded in the automaton $\mathcal{D}_f$ is indeed linear in $|f|$, independent of $\|A\|$. Imagine that we wished to recognise the word $abaab$ from Example 5.2 in a text written in ASCII; the new automaton $\mathcal{D}_{abaab}$ would be identical to the previous one except that in each state, and for each letter other than a and b, it would have a transition to state $\{0\}$. But instead we can describe a *complete* automaton $\mathcal{A}$ over A, using the idea of a 'default successor': a state s is distinguished and, for each state p, certain transitions, called *active transitions*, are given explicitly; the letters of A which do not appear as the label of one of these transitions are defined to be the label of a transition from p to s. The size of $\mathcal{A}$ is by definition equal to the number of active transitions. Figure 5.10 shows the automaton $\mathcal{D}_{abaab}$ with default successors.

[54] *Cf.* also Note II.75, p. 342.

[55] *Note added in translation:* Not to mention Unicode, which has anywhere up to 2^{32} letters.

The construction of an automaton for finding f with a default successor is due to Imre Simon. It can be shown that its size is linear in $|f|$. Furthermore, for such an automaton to read a word t requires that for each letter read the transition can be found which is labelled with that letter and leaves from the state reached at that point in the reading; that implies a certain number of comparisons. It can also be shown that the list of active transitions which leave from each state can be organised in such a way that the total number of comparisons required to read a word t by Simon's automaton is no greater than the number of comparisons made by KMP applied to the same word t.

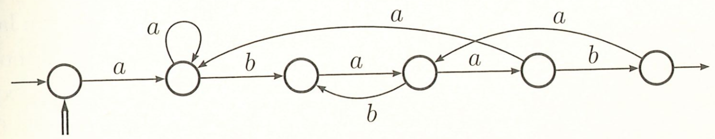


Figure 5.10: The automaton $\mathcal{D}_{abaab}$ (again); the default successor state is shown by a double incoming arrow.

Exercise

5.7 **Simon's automaton.** We want to show that the number of active transitions of the automaton that finds f is less than $2|f|$.

(a) Recall that the *period* of a word f is an integer p such that for all i, $1 \leqslant i \leqslant |f| - p$, we have $f_{i+p} = f_i$. Write $\mathsf{per}(f)$ for the *shortest period* of f and $f\beta$ for the *border* of f. Show that, for all f, $\mathsf{per}(f) + |f\beta| = |f|$. *cf. Exer. 0.3.6, p. 23*

(b) The states of $\mathcal{D}_f$ are the prefixes of f. A transition (p, a, q) in $\mathcal{D}_f$ is active if $q \neq 1_{A^*}$; an active transition is called 'forward' if $q = pa$ and 'backward' otherwise. Show that if (p, a, q) and (r, b, s) are two backward transitions, then $\mathsf{per}\, pa \neq \mathsf{per}\, qb$.

(c) Draw conclusions.

6 · Star height

The object of this section is two-fold. First, to present a sort of refinement of Kleene's Theorem. Not only do automata and expressions correspond to each other and characterise the same class of languages, but for both of them a measure of complexity can be defined: *loop complexity* for automata and *star height* for expressions, which we shall also show to correspond to each other (Eggan's Theorem).

Next we shall tackle the problem of applying this measure to languages themselves. The determination of the star height of a language turns out to be one of the most difficult problems in automata theory. We shall confine ourselves here to its presentation and the proof, due to Dejean and Schützenberger, that this notion really corresponds to a measure of complexity; that is, that there are rational languages with an arbitrarily large star height. We will end by touching on another, even more mysterious notion, the *generalised star height*, which may well flatten the family of rational languages.

cf. Sec. II.8, p. 342

In the next chapter we will see how to calculate the star height of a particular family of languages. The solution of the general case, due to K. Hashiguchi, is left to another volume,[56] but in the next chapter we shall at least see why it is difficult to justify its being entirely satisfactory.[57]

6.1 Two heights and a degree

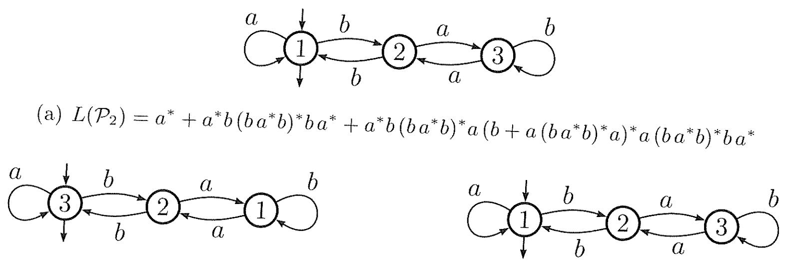
We have seen that the same language can be represented by distinct rational expressions and that, except for finite languages, we can unroll an infinity of these expressions.

We have also seen that if we choose an automaton $\mathcal{A}$ that recognises a language L – its minimal deterministic automaton, for example – the different methods of calculating, from $\mathcal{A}$, a rational expression that denotes L lead to distinct expressions. Consider for example $\mathcal{P}_2$, the 'divider by 3'. Figure 6.1 shows the result of applying the McNaughton–Yamada algorithm to the states of $\mathcal{P}_2$ using three different orders.

cf. Exer. 2.15, solution p. 191

The question naturally arises therefore whether, from all the expressions that denote L, we can designate one to be canonically associated with L, or failing that, if we can find common characteristics in all the expressions that can be linked to L.

cf. Exer. 6.1



(a) $L(\mathcal{P}_2) = a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b+a(ba^*b)^*a)^*a(ba^*b)^*ba^*$

(b) $L(\mathcal{P}_2) = (a + b(ab^*a)^*b)^*$

(c) $L(\mathcal{P}_2) = a^*+a^*b(ab^*a+ba^*b)^*ba^*$

Figure 6.1: The McNaughton–Yamada algorithm on $\mathcal{P}_2$

6.1.1 Star height of an expression

We defined rational expressions (over an alphabet A) as well-formed formulas made from atomic formulas, which are 0, 1 and the elements of A, and from the binary operators $+$ and $\cdot$ and the unary operator $*$.

The operator $*$ is the one that 'gives access to the infinite', hence the idea of measuring the complexity of expressions by counting the number of nested uses of this operator, a number which is usually called the *star height*, which we will write $\mathsf{h}[\mathsf{E}]$ (for $\mathsf{E} \in \mathsf{RatE}\, A^*$) and which is defined by induction:

[56] Of all such promises, this is without doubt the most presumptuous.

[57] *Note added in translation:* D. Kirsten has recently given (in [132] and subsequent works) a new proof of Hashiguchi's result which seems far more intelligible. My promise now seems within reach of fulfilment.

$$\text{if } \mathsf{E} = 0,\ \mathsf{E} = 1 \text{ or } \mathsf{E} = a \in A, \qquad \mathsf{h}[\mathsf{E}] = 0 \ , \tag{6.1}$$

$$\text{if } \mathsf{E} = \mathsf{E}' + \mathsf{E}'' \text{ or } \mathsf{E} = \mathsf{E}' \cdot \mathsf{E}'', \qquad \mathsf{h}[\mathsf{E}] = \max(\mathsf{h}[\mathsf{E}'], \mathsf{h}[\mathsf{E}'']) \ , \tag{6.2}$$

$$\text{if } \mathsf{E} = \mathsf{F}^*, \qquad \mathsf{h}[\mathsf{E}] = 1 + \mathsf{h}[\mathsf{F}] \ . \tag{6.3}$$

Examples 6.1 (i) $\mathsf{h}[(a+b)^*] = 1$; $\mathsf{h}[a^*(ba^*)^*] = 2$.

(ii) The heights of the three expressions in Figure 6.1 are

$\mathsf{h}[a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b+a(ba^*b)^*a)^*a(ba^*b)^*ba^*] = 3$,

$\mathsf{h}[(a+b(ab^*a)^*b)^*] = 3$; $\mathsf{h}[a^* + b(ab^*a+ba^*b)^*ba^*] = 2$. □

By induction on h, the distributivity of sum over product implies:

Property 6.1 *Every rational expression* E *over* A *of height less than or equal to* h *is equivalent (modulo* $\mathbf{D}$*) to a finite sum of expressions:*

$$\mathsf{E} = \sum_{i \in I} \mathsf{E}_i \ ,$$

where each $\mathsf{E}_i = \mathsf{F}$ *is a product of the form*

$$\mathsf{F} = u_0 (\mathsf{F}_1)^* u_1 (\mathsf{F}_2)^* u_2 \cdots u_{k-1} (\mathsf{F}_k)^* u_k \ ,$$

where each u_j, $0 \leqslant j \leqslant k$, *is a word in* A^* *and each* F_j, $1 \leqslant j \leqslant k$, *is a rational expression over* A *of height less than or equal to* $h-1$, *and of the same form.* ■

The expressions in Examples 6.1 have this form, which one could call normalised.

6.1.2 Star height of a language

The preceding examples draw attention to the fact that two equivalent expressions (that is, that denote the same language) can have different star heights.

Definition 6.1 The star height of a rational language L over A^*, written $\mathsf{h}[L]$, is the minimum of the star heights of the rational expressions that denote L:

$$\mathsf{h}[L] = \min\{\mathsf{h}[\mathsf{E}] \mid \mathsf{E} \in \mathsf{RatE}\, A^* \quad |\mathsf{E}| = L\} \ . \qquad \square$$

This definition brings with it two questions:

- Are there rational languages with arbitrarily large star heights?
- Can the star height of a rational language given by, for example, a finite automaton be calculated effectively?

The answer to the first question, which is affirmative, will be given a little later in a purely combinatorial fashion, and we shall return to it in the next chapter. The answer to the second, also affirmative, constitutes without doubt, as we have said,