Factor Theory Made Calculational

Roland Backhouse[*]

September 26, 2017

Abstract

In 1971, J.H. Conway introduced the notion of the factors of a language and the factor matrix of a regular language. Shortly afterwards, the author extended Conway's factor theory in several ways. The existence of a unique, minimal starth root of the factor matrix, dubbed the factor graph of the language, was established. It was also proved that the factor matrix of a factor of a language is a submatrix of the factor matrix of the language, and the factor graph of a factor of a language is pathwise homomomorphic to the factor graph of the language. The latter result was used to derive an algorithm for computing the factor matrix of a language from its factor graph with the property that the resulting regular expressions have star-height at most the cycle-rank of the factor graph, and could be strictly smaller.

Using the simple device of naming the factor operators, the current paper revisits Conway's and our own work on factor theory in a calculational style. We give explicit constructions of the factor matrix, the factor graph, submatrices of the factor matrix defined by subfactors and pathwise homomorphisms of factor graphs.

We also extend factor theory beyond this earlier work. We formulate the theory more abstractly so that we can rigorously justify the use of the syntactic monoid of a language in calculations involving factors. We also present Conway's theory of approximations of regular languages but extended to this more general abstract context. When specialised to regular languages, we prove the existence of a unique *minimal* approximating function (a *least* approximating function) and compare this with Conway's *maximal* constant+linear approximating function.

The closure algorithm we present does not always construct regular expressions of minimal star-height. However, we speculate in the conclusions on how the theory of approximations might be exploited to develop a novel, effective approach to solving the star-height problem.

[*]School of Computer Science, University of Nottingham, Nottingham NG8 1BB, England

Contents

List of Figures

1 Introduction

J.H. Conway [Con71] introduced the notion of the factors of a language and showed that the factor-of relation is reflexive and transitive. He also introduced the notion of the factor matrix of a regular language. This is a matrix of which every entry is a factor of the language and which is reflexive and transitive.

More than forty years ago, soon after the publication of Conway's work, I extended Conway's factor theory in several ways [Bac75]. I proved the existence of a unique, minimal starth root of the factor matrix which I called the *factor graph* of the language. I also proved that the factor matrix of a factor of a language is a submatrix of the factor matrix of the language, and the factor graph of a factor of a language is pathwise homomomorphic to the factor graph of the language. The latter result was used to derive an algorithm for computing the factor matrix of a language from its factor graph with the property that the resulting regular expressions have star-height at most the cycle-rank of the factor graph, and could be strictly smaller.

Conway's formulation of factor theory is wordy and, hence, often unclear; our own work at the time followed Conway's style and can be similarly criticised. Using the simple device of naming the factor operators, the current paper revisits Conway's and our own work on factor theory in a calculational style. We give explicit constructions of the factor matrix, the factor graph, submatrices of the factor matrix defined by subfactors and pathwise homomorphisms of factor graphs.

We also extend factor theory beyond this earlier work. We formulate the theory more abstractly so that we can rigorously justify the use of the syntactic monoid of a language in calculations involving factors. We also present Conway's theory of approximations of regular languages but extended to this more general abstract context and demonstrate its relevance in practical applications. In this way, we hope that factor theory will be better understood and more widely recognised than is presently the case.

1.1 A Simple Introductory Example

In order to give an overview of this paper, let us consider a simple example. Much of the terminology in this section is used in an imprecise, informal way, with the consequence that some of the explanation may not be completely clear on a first reading. We make the terminology precise in later sections, following which we invite the reader to review this section once more.

For our example, we consider the language denoted by the regular expression

$$(a{+}b)^* \, a \, (a{+}b)^*$$

over the alphabet $\{a,b\}$. (This example is an extension of one used in [Bac16].) Let us denote this language by E and the alphabet by T. Then, using equational properties of regular languages (with which we assume the reader is familiar), we have:

$$E \;=\; (a{+}b)^* \, a \, (a{+}b)^* \;=\; b^* \, a \, (a{+}b)^* \;=\; (a{+}b)^* \, a \, b^* \quad .$$

The driving concern in this paper is the development of methods for determining the "best" regular expression denoting a given language. In this case, it could be argued that the leftmost expression is the "best" because it conveys most clearly that the language is the set of words that have at least one a. (An alternative expression would be $\neg(b^*)$ but we exclude this because we consider only expressions formed from the symbols of the alphabet, concatenation, union and star.)

The reduced, deterministic finite-state automaton recognising E is shown in fig. 1(a). For brevity, we follow Conway and call it the *machine* of E. Fig. 1(b) shows the reflexive, transitive closure —the "star"— of fig. 1(a). In other words, the label annotating the edge from node i to node j in fig. 1(b) denotes the set of words recognised by transitions from node i to node j in fig. 1(a). The language E is a special case: it is the set of words recognised by transitions from the start state (indicated by an unlabelled arrow) to the final state (indicated by a double circle).

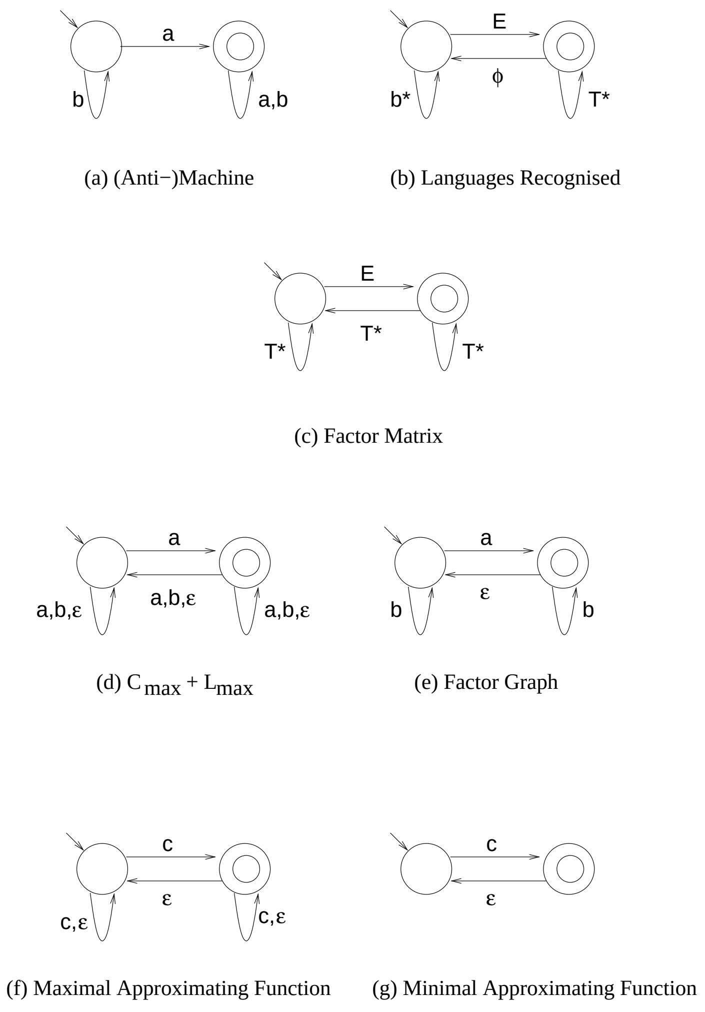


Figure 1: Example: $(a+b)^* \, a \, (a+b)^*$

For greater simplicity, the example language we have chosen is such that it is the reverse of itself. (The reverse of a language is the set consisting of the reverse of all words in the language.) This means that the "anti-machine" of E is identical to its machine, as indicated by the caption of fig. 1(a). The "anti-machine" of a language is the machine of the reverse of the language. Moreover, there are no inadmissible nodes in the machine (or anti-machine) of E; this means that there are no inadmissible entries in the factor matrix of E. (A node is inadmissible if it cannot be reached from the start state or the final state cannot be reached from it.)

Constructing regular expressions denoting a language inevitably involves constructing a transition graph to which a closure algorithm is applied. The machine of a language is a (deterministic) transition graph. Figs. 1(d) and (e) are both transition graphs which recognise our example language. Transition graphs have edges that are labelled by subsets of $\{\varepsilon\}\cup T$, where ε denotes the empty word and T is the alphabet.

We do not give the word "graph" a formal meaning. For us, a graph is a way of depicting a function that has domain a cartesian product $N{\times}N$ for some finite set (of so-called "nodes") N. Another way of depicting a graph is as a two-dimensional array, commonly called a "matrix" in the mathematical literature. The graph shown in fig. 1(c) depicts what Conway calls the *factor matrix* of E; it is depicted in the conventional way as a two-dimensional array below:

$$\begin{bmatrix} T^* & E \\ T^* & T^* \end{bmatrix}$$

Typically, we are only interested in a small number of entries in a matrix. The advantage of a graph as a means of communication is that we can identify so-called "start" and "final" nodes which determine the entries of interest. In each of the figures in fig. 1, there is one start node, indicated by an unlabelled incoming edge, and one final node, indicated by a double circle. Graphs can however quickly become very cluttered, in which case a matrix can be a better means of presentation. We switch between graphs and matrices, using whichever is most convenient at the time.

In Conway's terminology, figs. 1(d) and (e) depict "constant+linear matrices". (That is, the edge labels are either ε or elements of the alphabet.) They share several common properties. Both depict transition graphs that recognise the language E; both are also "starth roots" of the factor matrix of E. Fig. 1(d) is Conway's maximal constant+linear approximation of E, and fig. 1(e) is our *minimal* starth root of the factor matrix, the *factor graph* of E. Presented as matrix equations, we have:

$$\begin{bmatrix} \varepsilon{+}a{+}b & a \\ \varepsilon{+}a{+}b & \varepsilon{+}a{+}b \end{bmatrix}^* = \begin{bmatrix} b & a \\ \varepsilon & b \end{bmatrix}^* = \begin{bmatrix} T^* & E \\ T^* & T^* \end{bmatrix} \quad .$$

Our interest in the factor matrix is as a means of calculating a "best" regular expression denoting a particular language, where "best" entails some measure of simplicity. As we have already mentioned, such calculations have two components: a transition graph and the closure algorithm used to compute the star (reflexive-transitive closure) of the graph. It seems sensible, therefore, to begin with the simplest possible transition graph. Comparing figs. 1(d) and (e), it is clear that our factor graph is simpler than Conway's maximal constant+linear approximation. Indeed, as we show later, the factor graph of a regular language E is the *minimal* starth root of the factor matrix of E and, in this sense, is the "best" starting point for computing the factor matrix of E.

The next consideration is the closure algorithm to be used. Recall that our example language is denoted by the expressions $(a+b)^*a(a+b)^*$, $(a+b)^*ab^*$ and $b^*a(a+b)^*$. All of these are quite compact and are useful for communicating properties of the language. (For example, the equality between the first and second expressions can be used to explain the fact that any word that has at least one a must have a final occurrence of an a that is followed by a string of bs.) However, if a standard elimination algorithm is used, the resulting regular expressions are typically far from ideal. In this case, the machine and anti-machine (fig. 1(a)) do result in compact regular expressions —from the machine, we get the expression $b^*a(a+b)^*$ and, from the anti-machine, we get $(a+b)^*ab^*$— but applying a standard elimination algorithm to the factor graph (fig. 1(e)), we get the expression $b^*a(b+b^*a)^*$ or $(b+ab^*)^*ab^*$, depending on whether the leftmost or rightmost node is eliminated first. In both cases, the star-height of the resulting expression is two, which is undesirable. Of course, were we to apply an elimination algorithm to Conway's maximal constant+linear approximation (fig. 1(d)), we would get yet more complicated expressions, albeit of the same star-height.

The problem with standard elimination algorithms is that they fail to exploit all the algebraic properties of regular languages. Such algorithms have counterparts in algorithms for inverting matrices in linear algebra [BC75]. Consequently, they do not exploit the fact that "addition" of regular languages (the "$+$" operator in the above expressions) is idempotent —its meaning is, after all, set union— since addition is not idempotent in linear algebra (i.e. normal arithmetic). Elimination algorithms result in regular expressions that have star-height at least the so-called "cycle rank" [Egg63] of the graph to which they are applied. The cycle rank of fig. 1(a) is one; the cycle rank of both figs. 1(d) and (e) is two. Perhaps by inventing a new closure algorithm that properly exploits the algebraic properties of languages, we can do better than the cycle rank?

When applying the algorithm to the machine of a language, there is little prospect of improving on an elimination algorithm precisely because a machine is *deterministic*: the ambiguity that idempotency affords is not present in the machine. Only by considering non-deterministic recognisers of a language is there hope of an improved algorithm. The factor graph of a language is a non-deterministic recogniser of the language (indeed, in some cases a very practical recogniser: it is at the heart of the Knuth-Morris-Pratt pattern-matching algorithm [BL76, BL77]) and, as we show later, there is an algorithm to calculate its closure that constructs regular expressions that have star-height at most the cycle rank of the graph and sometimes smaller.

To get a first impression of the algorithm, examine the factor matrix of our example language, shown in fig. 1(c). The matrix has just two distinct entries, the language itself and T^*. These are the two "factors" of our example language. The factor matrix of the factor T^* is very simple: it is just the 1×1 matrix $[T^*]$. Its factor graph is equally simple: it is $[T]$ (equivalently, $[a+b]$). Moreover, this is a graph of cycle rank one, which is strictly less than the cycle rank of the factor graph of our example language. Most importantly, the factor matrix of the factor T^* is a *submatrix* of the factor matrix of our example language. In fact, it occurs twice as a submatrix. (Although T^* occurs three times, the off-diagonal entry is formally not a factor matrix according to our definition.)

The example illustrates several properties that hold in general of a regular language: factors of factors are themselves factors, the factor matrix of a factor is a submatrix of the factor matrix, and the factor graph of a factor has cycle rank at most the cycle rank of the factor graph. These properties are the basis of an algorithm to compute the factor matrix of a language that returns regular expressions that have star-height at most the cycle rank of the factor graph of the language and, in many cases, strictly less than the cycle rank.

Our algorithm involves some pre-processing to determine structural properties of the factors of a language. At first sight, it might seem that this entails constructing the factor matrix and then performing a number of calculations before calculating the factor matrix once more. Moreover, the intermediate calculations would appear to involve comparing regular expressions, which is very much a non-trivial task. The pre-processing is, however, relatively straightforward because it can be done by exploiting the syntactic monoid of the language. The syntactic monoid (aka semi-group) of a language is a monoid generated by a congruence relation on words [RS59]; the monoid is finite when the language is regular. Fundamental to our algorithm is that structural properties of the factor matrix can be determined by performing calculations in an abstract regular algebra (thus not the algebra of regular languages) whose carrier set is the set of subsets of the carrier set of the syntactic monoid. Crucially, when the given language is regular, the subsets are finite. This abstract regular algebra is an example of what we call a powerset algebra.

Our simple example has a simple syntactic monoid. It has just two elements which we name 1 and a. The product operatoris defined by $1\circ 1 = 1$, $1\circ\mathsf{a} = \mathsf{a}\circ 1 = \mathsf{a}$ (so 1 is the unit of the monoid, as its name suggests) and $\mathsf{a}\circ\mathsf{a} = \mathsf{a}$. The element 1 corresponds to the congruence class comprising the words in b^* and the element a corresponds to the congruence class comprising the words in the example language E (i.e. $(a+b)^*a(a+b)^*$). In the powerset algebra, it is easy to calculate that

$$\begin{bmatrix} \{1\} & \{a\} \\ \{1\} & \{1\} \end{bmatrix}^* = \begin{bmatrix} \{1,a\} & \{a\} \\ \{1,a\} & \{1,a\} \end{bmatrix} .$$

On the left of this equation is the factor graph of E, represented within the powerset algebra, and on the right is the factor matrix of E, represented in the same way. From this representation of each element of the factor matrix as a finite set, it is easy to deduce the structural properties required by our algorithm.

Much of the theory developed in this paper is an extension of Conway's theory of "approximations" of a regular language. So let us conclude this section with several examples of such "approximations".

Our first example is the sort of approximation envisaged by Conway. Suppose we want to "approximate" our example language by the language b^*a. A maximal "approximation" is the language denoted by $(b^*a)^+$. This is a function of b^*a: it is the transitive closure of b^*a. Conway defines (rather imprecisely as we shall see) the notion of an "approximating function" and the "best approximation" of a language by a finite set of languages. He then shows how the factor matrix is used to construct the "best approximation".

Figs. 1(f) and (g) depict "approximating functions" defined by the "approximation" b^*a. Informally, what is meant is this. First note that both graphs have the same reflexive, transitive closure. The language we are interested in is the language defined by the start and final nodes. From fig. 1(g), this is clearly[1] c^*c. The "best approximation" is obtained by instantiating c to b^*a; the "best approximation" is thus $(b^*a)^*b^*a$ (which can, of course, be simplified to $(a+b)^*a$). This language is "best" in the sense that it is the maximal set of words that is the result of applying an "approximating function" to b^*a and is a subset of $(a+b)^*\,a\,(a+b)^*$.

We dislike Conway's use of the word "best". We dislike it particularly because his "best approximating function" —illustrated by fig. 1(f)— is clearly not "best" since fig. 1(g) is simpler. Both graphs have the same reflexive, transitive closure, so both yield the same approximations. Our concern is that regular expressions denoting the approximations are inevitably more complex when using Conway's "best" approximation function. Fig. 1(f) is an example of what Conway calls the "factorial function"; our fig. 1(g) is *minimal* whereas Conway's fig. 1(f) is *maximal* among a class of "approximating functions", which we make precise later.

We use the theory of approximations in two other ways. First, Conway's maximal constant+linear approximation of E (fig. 1(e)) and our factor graph of E are instances of approximating functions: essentially we consider the symbols of the alphabet as the approximating events. The "best" (or, as we prefer to say, "maximal") "approximation"

[1]We assume that all closure algorithms exploit the fact that the empty word, ε, is the unit of concatenation. In this case, $(c\cdot\varepsilon)^*c$ is simplified to c^*c.

of E is then E itself. Second, we use the theory of approximations to formally justify the use of the syntactic monoid in our calculations.

1.2 Overview

This paper is a mixture of three components. First, it presents results due to Conway [Con71] on the factor matrix and approximations of regular languages in a calculational style. It does the same for extensions to Conway's theory first presented in the author's PhD thesis. Finally, it extends Conway's theory of approximations, arguing the case for minimal approximating functions as opposed to Conway's maximal approximating functions.

Unlike Conway, our exposition is not solely about regular languages: we begin with a general context and then specialise the context in stages to regular languages. In section 2.2 we give an abstract definition of a "regular algebra" and, in section 3, we introduce the "factor matrix" of an event in such an abstract algebra. For a gentler introduction see [Bac16]. Section 4 presents properties of the factor matrix of a factor, first developed in the author's PhD thesis [Bac75] but not published elsewhere. The main theorem is that the factor matrix of a factor of an event is a submatrix of the factor matrix of the event. This theorem is fundamental to later sections on improved closure algorithms.

Our axiomatic approach allows us to generalise Conway's and our own theorems. For example, whilst retaining Conway's terminology, our notion of a "matrix" is not the traditional finite-dimensional array of values. This allows us to generalise the theorem first proved in [Bac75] on factor matrices of factors to situations where an event does not have a finite number of factors.

Approximation theorems are presented in section 5. The results in this section are essentially due to Conway. The novelty of this paper is that the results are formulated in the general context of an abstract regular algebra rather than the specific context of the algebra of regular languages. This more general formulation is vital to justifying the use of the syntactic monoid in analysing the structure of the factor matrix. Our calculational presentation with explicit formulation of underlying Galois connections also clarifies Conway's work. As we point out, Conway's presentation has major omissions and some errors, making it difficult to understand.

Conway's approach to constructing maximal approximations to a regular event is to construct approximating "functions"; such "functions" are sets of words, i.e. languages. Each word has a length, and this simple fact is exploited to show that there is a "best" "constant+linear function" approximating the factor matrix of a language. We argue that Conway's use of the terminology "best" is inappropriate, particularly with regard to our goal of constructing a regular expression denoting a given regular language. Properties of a regular language that are necessary to our development are first introduced in section

6: a regular language has a finite number of factors and the factor matrix has a unique starth root, which we call the "factor graph" of the language. The notion of factor graph was first introduced by the author in [Bac75]. Construction of the factor graph is the basis of the well-known Knuth-Morris-Pratt pattern-matching algorithm and its generalisation to sets of patterns [KMP77, Wei73, AC75], as shown in [BL76, BL77]. See also [Bac16].

Section 7 is the first of three sections that are entirely original work of the author; it is about how to exploit the syntactic monoid of a regular language in order to construct its factor graph and to perform preprocessing of its factor matrix (without actually constructing the factor matrix) in order to apply the closure algorithm that is presented in section 9. Section 8 establishes the property that the factor graph of a factor of a regular language is pathwise homomorphic to the factor graph of the language; this theorem predicts the possibility of deriving a closure algorithm that constructs regular expressions denoting the factors of a regular language that have star-height at most, and sometimes less than, the rank of the language's factor graph. As we show by example in section 9, the algorithm often constructs regular expressions of minimal star-height.

Unfortunately, as we show in section 9.4, the algorithm does not solve the star-height problem: it sometimes fails to calculate regular expressions of minimal star-height. In the concluding section, we remark that our algorithm does determine regular expressions of minimal star-height for so-called pure-group languages (languages for which the syntactic monoid is a group).

Surprisingly little has been written on Conway's factor theory since the publication of his book in 1971. Apparently, it took more than 25 years before my notion of the "factor graph" was rediscovered: Lombardy and Sakarovitch [LS02] call it the "écorché". But many of the results in my thesis remain unknown and unexplored to this day. Section 10 points out the close parallels between my work of forty years ago and that of Lombardy and Sakarovitch.

2 Regular Algebra

In this section, we give a summary of the algebraic properties that we exploit later in the paper.

We begin in section 2.2 with an abstract definition of a regular algebra. The definition is equivalent to Conway's Standard Kleene algebra (S -algebra) [Con71, p.27], but emphasises the notion of factorisation. We don't give a formal proof of the equivalence, which follows from well-known properties of Galois connections. We assume that the reader is familiar with the theory of Galois connections. For those unfamiliar with the theory, a definition and reference to relevant literature is given in section 2.1.

Languages over a given (finite) alphabet form a regular algebra according to our definition. However, an abstract definition that encompasses other applications is vital, in particular to justify the use of matrices and the syntactic monoid. In sections 2.3 and 2.4 , we formulate the construction of matrix algebras and powerset algebras from given regular algebras. The combination of matrices and powersets is formulated in section 2.5.

Because we use a variety of regular algebras, often simultaneously, we have to make a difficult choice with regard to notation: do we overload operator symbols (like " * ", the so-called Kleene star operator) or do we invent a different notation for each individual algebra? The choice is compounded by the fact that certain choices of notation also clash with notation traditionally associated with other well-known operations. For example, we use the symbol " $\times$ " for Cartesian product (as in $S{\times}T$ where S and T are sets); so we avoid its use for the product operator in a regular algebra, in particular for matrix multiplication. Such notational choices are discussed in section 2.6. Finally section 2.7 recalls some properties of regular algebras that follow from the properties of Galois connections in combination with fixed-point calculus.

Various running examples are also introduced in this section.

2.1 Galois Connections

Galois connections feature heavily in this paper. A Galois connection comprises two partially ordered sets $(A,\preceq)$ and $(B,\sqsubseteq)$ and two functions f and g of types $A{\leftarrow}B$ and $B{\leftarrow}A$, respectively, with the property that, for all a in A and all b in B,

$$f.b\preceq a \;\equiv\; b\sqsubseteq g.a \quad .$$

There is extensive literature on Galois connections and we assume the reader is familiar with the concept. In particular, we assume that the reader is familiar with the theorem dubbed "the unity of opposites" by the author [Bac02].

Note that we use an infix dot to denote function application.

2.2 Definition, Factorisation and Suprema

Definition 1 (Regular Algebra) Suppose A is a set. Suppose that $(A,\cdot,1)$ is a monoid and $(A,\preceq)$ is a partially ordered set that is moreover a complete lattice; also, suppose that A *admits factorisation*: that is, there are operators $\backslash$ and $/$ such that, for all X, Y and Z in A,

$$(2)\qquad X{\cdot}Y\preceq Z \;\equiv\; Y\preceq X\backslash Z$$

and

(3) $\quad \mathsf{X{\cdot}Y} \preceq \mathsf{Z} \;\equiv\; \mathsf{X} \preceq \mathsf{Z/Y}$.

We call such a structure a *regular algebra*. Elements of the carrier set of a regular algebra are called *events*.
□

Another name for a regular algebra, according to Wikipedia, is a *unital quantale*. (In the literature on quantale theory, a factor is called a *residual*.) The operators / and \ are pronounced "over" and "under", respectively.

The simplest non-trivial example of a regular algebra has event set the booleans, as detailed below. We use this as the basis of one of several running examples. See examples 15, 66, 80, 85, 87 and 90.

Example 4 (Running Example: Booleans) Let Bool denote {false,true}. Then (Bool,∧,true) is a monoid and (Bool,⇒) is a complete lattice (with disjunction as supremum operator). Also, Bool admits factorisation with "only-if" as the "under" and "if" as the "over" operator since

$$\mathsf{X} \wedge \mathsf{Y} \Rightarrow \mathsf{Z} \;\equiv\; \mathsf{Y} \Rightarrow (\mathsf{X} \Rightarrow \mathsf{Z}) \quad \text{and} \quad \mathsf{X} \wedge \mathsf{Y} \Rightarrow \mathsf{Z} \;\equiv\; \mathsf{X} \Rightarrow (\mathsf{Z} \Leftarrow \mathsf{Y}) \quad .$$

(On the right side of these equations, read the non-parenthesised "⇒" term as an ordering; the parenthesised occurrence of "⇒" in the first equation is the under operator.)
□

So far as possible, we formulate theorems in the context of an arbitrary regular algebra (rather than just in the context of the algebra of languages as did Conway). In such cases, we implicitly assume that the carrier set of the algebra is A, and we use the notation of definition 1 to denote the ordering relation on A, the unit of the monoid, and the product and factor operators. That $(A,\preceq)$ is a complete lattice means that every function with range A has a supremum. The supremum of function f of type $A{\leftarrow}B$ will be denote by Σf. It has the defining property

(5) $\quad \langle \forall a : a{\in}A : \Sigma f \preceq a \;\equiv\; \langle \forall b : b{\in}B : f.b \preceq a \rangle \rangle$.

It is convenient to also use the quantifier notation $\langle \Sigma j :: f.j \rangle$, where the range of dummy j is (implicitly) the domain of f, particularly in the case that we don't want to name the function f.

When f has type $A{\leftarrow}\mathbf{2}$ (where $\mathbf{2}$ is a two-element set) we use the infix operator "$+$" to denote the supremum. That is, using subscripting to denote function application (as is conventionally done in such cases), we write f_0+f_1 for the supremum of f. Instantiating (5) with $f_0:=x$ and $f_1:=y$, we get

(6) $\langle\forall a : a\in A : x+y\preceq a \equiv x\preceq a \wedge y\preceq a\rangle$.

In the particular case that f has type $A\leftarrow\emptyset$ (that is, its domain is the empty set) we denote its supremum by 0. Instantiating (5) once again, the innermost universal quantification is vacuously true, so we get

(7) $\langle\forall a : a\in A : 0\preceq a\rangle$.

That is, the supremum of a function of type $A\leftarrow\emptyset$ is the least element of A. Moreover, 0 is the (left and right) zero of product (i.e. $0\cdot Y=0=X\cdot 0$ for all X and Y), as is easily verified. For example, we have: for all Y and Z,

$$
\begin{array}{ll}
 & 0\cdot Y=0 \\
= & \quad\{\quad \text{antisymmetry of } \preceq\text{; (7) with } a:=0\cdot Y \quad\} \\
 & 0\cdot Y\preceq 0 \\
= & \quad\{\quad (3) \quad\} \\
 & 0\preceq 0/Y \\
= & \quad\{\quad \text{(7) with } a:=0/Y \quad\} \\
 & \mathsf{true} \ .
\end{array}
$$

The use of the symbols "0", "1", "$+$" and "$\cdot$" suggests a strong connection with properties of the well-known arithmetic operators. The regular and arithmetic operators do share many properties but it is important to recognise that the algebras differ in important respects. Indeed, a brief summary of the current paper might be that it focuses on those properties of the regular operators that *distinguish* them from the arithmetic operators.

That $(A,\preceq)$ is a complete lattice also means that every function with range A has an infimum. This is a property that becomes important when we specialise the discussion to powerset algebras. (See section 2.4.) Choosing to denote binary suprema by the operator symbol "$+$" might suggest that we use "$\times$" (or some other symbol normally associated with multiplication) for binary infima. We don't do so in order to avoid confusion with the product operator in a regular algebra, and also the use of "$\times$" to denote cartesian product.

Our definition of a regular algebra does not include the so-called "Kleene star" operator as a primitive. This is not without precedent: Conway studies several different "Kleene" algebras [Con71, chapter 4, pp34–40] all of which are derivatives of what he calls a *standard Kleene algebra* or S-algebra [Con71, p27]. An S-algebra has just two primitive operators, product and supremum, and, whilst lacking our emphasis on admitting factorisation via explicit naming of the factor operators, is identical to our "regular algebra". (Conway postulates certain distributivity properties that are equivalent to admitting factorisation.)

In a regular algebra, the "Kleene star" operator can be defined in several ways. For us, the most convenient definition of X^* is the least fixed-point of the function mapping Y to $1+X+Y{\cdot}Y$. (Note that we use the same precedence conventions as in ordinary arithmetic.) By definition, X^* is thus *reflexive* (i.e. $1\preceq X^*$) and *transitive* (i.e. $X^*\cdot X^*\preceq X^*$). The function mapping X to X^* is also a closure operator: in general, a function f of type $B\leftarrow B$, where $(B,\leq)$ is a partially ordered set, is a *closure operator* if, for all X and Y in B,

$$X\leq f.Y \equiv f.X\leq f.Y \quad .$$

For the star operator, we have:

$$X\preceq Y^* \equiv X^*\preceq Y^* \quad .$$

These three properties justify the name *reflexive, transitive closure* of X. Similarly, the *transitive closure* of X, denoted as usual by X^+, is defined to be the least fixed-point of the function mapping Y to $X+Y{\cdot}Y$. As the name suggests X^+ is transitive and the function mapping X to X^+ is a closure operator. Other properties of these two operators are well-known and will be assumed without further ado throughout the paper. (For example, we exploit the fact that X^* is the least fixed point of the function mapping Y to $1+X{\cdot}Y$ and, also, the least fixed point of the function mapping Y to $1+Y{\cdot}X$. Note that many discussions of "Kleene" algebra postulate these properties as axioms. It is, however, a relatively straight-forward exercise to derive the properties from the definitions we have given as instances of general properties of fixed points and Galois connections.)

2.3 Event Matrices

Example 4 defines a primitive regular algebra. There are several ways to construct more complex regular algebras from simpler ones [Bac06]. Fundamental to our definition of a regular algebra is that (square) "matrices" over a regular algebra also form a regular algebra. (This property was also stressed by Conway in his axiomatisations of "Kleene" algebras.) Because we do not want to restrict our discussion to languages, we use a more general definition of "matrix" than the standard finite-dimensional array of values.

Definition 8 (Event Matrix) Suppose I and J are sets (possibly infinite) and suppose $\mathcal{R}$ is a regular algebra with event set A as in definition 1. An *event matrix of dimension* $I{\times}J$ and *type* $\mathcal{R}$ is a function with domain $I{\times}J$ and range A.

If event matrices $\mathbf{f}$ and $\mathbf{g}$ have the same type, and $\mathbf{f}$ has dimension $I{\times}J$ and $\mathbf{g}$ has dimension $J{\times}K$, for some I, J and K, their *product* is the function $\mathbf{f}{\otimes}\mathbf{g}$ of dimension $I{\times}K$ defined by

$$(\mathbf{f}{\otimes}\mathbf{g})(i,k) \;=\; \langle \Sigma j :: \mathbf{f}(i,j)\cdot\mathbf{g}(j,k)\rangle \quad .$$

Event matrices with the same dimension and same type are ordered pointwise: if $\mathbf{f}$ and $\mathbf{h}$ both have dimension $I{\times}J$ and have the same type,

$$\mathbf{f} \mathrel{\dot{\preceq}} \mathbf{h} \;\equiv\; \langle \forall i,j :: \mathbf{f}(i,j) \preceq \mathbf{h}(i,j)\rangle \quad .$$

A *square event matrix of dimension* I is an event matrix of dimension $I{\times}I$ for some (possibly infinite) set I.
□

Since all our matrices are event matrices, we drop the adjective "event" from now on.

Theorem 9 The square matrices with domain $I{\times}I$, for some I, and range a regular algebra themselves form a regular algebra. Product is matrix product (definition 8) and the unit is the identity matrix, which we denote by $\mathbf{I}$. (That is, $\mathbf{I}(i,i)=1$ and $\mathbf{I}(i,j)=0$ when $i{\neq}j$.) The ordering on matrices is the pointwise ordering defined above, and the supremum of a matrix-valued function is the pointwise supremum of matrix elements. See [Bac06, theorem 4.20] for specific formulae defining the "over" and "under" operators.

Proof The proof is straightforward. See [Bac06, theorem 4.20]. Conway [Con71, p.40] states that it is trivial. (Strictly, he only makes this claim for finite-dimensional matrices; however, the finiteness assumption is only relevant for non-standard Kleene algebras.)
□

An example of "matrix" algebras is afforded by binary relations. The booleans form a regular algebra with conjunction as the product operator and implication as the ordering. (See example 4.) "(Square) matrices" of booleans are (homogeneous) binary relations and the product operator is relational composition. Such "matrices" have finite dimension exactly when the set on which the relations are defined is finite. See [DBvdW97] for extensive applications of factor theory in relation algebra.

In many simple examples of regular algebras, such as the regular algebra of booleans, the star operator is so simple that it is rarely considered. However, as in this example, more complex regular algebras are often constructed from simpler ones and then the star operator does become significant. For the regular algebra of booleans, the star operator is such that $X^*=\mathsf{true}$ for all X but "star" of a (homogeneous, binary) relation is the reflexive, transitive closure of the relation.

2.4 Power Set Algebras

Another way of constructing a regular algebra begins with a monoid and then "extends" the monoid to a regular algebra with carrier set the set of subsets of the monoid (the "power set" of the monoid). The appropriate definition is as follows:

Definition 10 (Power-set Monoid) Suppose $(M, \cdot, 1)$ is a monoid. Let 2^M denote the set of all subsets of M The product operator is extended to 2^M by

$$X{\cdot}Y \;=\; \{x{\cdot}y \mid x{\in}X \wedge y{\in}Y\} \;\; .$$

It is easily verified that $(2^M, \cdot, \{1\})$ is a monoid, which we call a *powerset monoid*.
□

Theorem 11 Suppose $(M, \cdot, 1)$ is a monoid. Then 2^M is the carrier set of a regular algebra with product as given by definition 10 and the subset relation as the ordering relation.

Proof As already remarked, $(2^M, \cdot, \{1\})$ is a monoid; moreover $(2^M, \subseteq)$ is a complete lattice. In order to show that the algebra admits factorisation, we begin by introducing the notion of the *derivative* of an event[2] with respect to an element of the underlying monoid. Specifically, suppose $w{\in}M$. Then the *derivative* ∂_w, a function of type $2^M \leftarrow 2^M$, is defined by, for all events N,

$$x \in \partial_w N \;\equiv\; w{\cdot}x \in N \;\; .$$

That is, for all events N and P and all monoid elements w,

$$\{w\}{\cdot}P \subseteq N \;\equiv\; P \subseteq \partial_w N \;\; .$$

Then, for all events N, P, and Q,

$$N{\cdot}P \subseteq Q \;\;\equiv\;\; P \subseteq \langle \cap w : w{\in}N : \partial_w Q \rangle \;\; .$$

Denoting the event $\langle \cap w : w{\in}N : \partial_w Q \rangle$ by $N \backslash Q$ we thus have

$$N{\cdot}P \subseteq Q \;\equiv\; P \subseteq N \backslash Q \;\; .$$

That is, the function $(N\cdot)$ has an upper adjoint. Similarly, the function $(\cdot P)$ also has an upper adjoint, which we denote by the postfix operator $(/M)$. That is,

$$N{\cdot}P \subseteq Q \;\equiv\; N \subseteq Q/P \;\; .$$

[2]Recall that an event is an element of the carrier set of a regular algebra. So in this case an event is a subset of M.

(The specific formula for Q/P involves "anti"-derivatives. We discuss anti-derivatives in more detail later in the context of language theory.) Thus we have shown that the powerset algebra admits factorisation.
$\square$

We exploit theorem 11 to construct several regular algebras. We refer to the regular algebra that is constructed from a given monoid M as *the powerset algebra with underlying monoid* M.

Example 12 The archetypical example of a powerset regular algebra is the algebra of languages over a finite alphabet: Let T be a finite set. The set T is called the *alphabet* and elements of T are called *letters*. A *word of length* n is a sequence of symbols of length n. The *empty word* is the word of length 0; we denote it by the symbol ε. The set of all words is denoted by T^*. The set T^* is the carrier set of the *free* monoid over alphabet T. The product operation of the free monoid is *concatenation* of words, which is typically denoted by juxtaposition; concatenation is associative ($(uv)w = u(vw)$) and ε is both its left and right unit ($\varepsilon u = u = u\varepsilon$). In this context, "events" are called *languages*, a language being a subset of T^*.
$\square$

Reasoning about words in a language often involves exploiting properties of the length of a word. In particular, the length of the concatenation uv of words u and v is the sum of the lengths of u and v, and induction on the lengths of words is a valid proof technique. That the monoid is "free" means that we can exploit cancellation: if u, v and w are words, the statements $u = v$, $uw = vw$ and $wu = wv$ are all equivalent. These are simple and well-known properties. Our calculations are less explicit about their usage. See, for example, the calculations in Appendix A where we prove properties of matrices having entries that are sets of words whose length is zero or one.

Our use of the word "derivative" in the proof of theorem 11 is an acknowledgement to Brzozowski [Brz64] who introduced the notion in the context of languages (example 12). Brzozowski enumerated a number of algebraic properties of derivatives of languages that are very useful for practical calculation. Some remain valid in any powerset regular algebra. In particular, we have the following lemmas:

Lemma 13 For an arbitrary collection N of subsets of the carrier set of a monoid,

$$\partial_w \langle \cup i :: N.i \rangle \quad = \quad \langle \cup i :: \partial_w (N.i) \rangle \quad .$$

In particular, $\partial_w \emptyset = \emptyset$ and $\partial_w (P \cup Q) = \partial_w P \cup \partial_w Q$.

Proof We have, for all x,

$$
\begin{array}{ll}
 & x \in \partial_w \langle \cup i :: N.i \rangle \\
= & \{ \quad \text{definition of derivative} \quad \} \\
 & w \cdot x \in \langle \cup i :: N.i \rangle \\
= & \{ \quad \text{definition of set union} \quad \} \\
 & \langle \exists i :: w \cdot x \in N.i \rangle \\
= & \{ \quad \text{definition of derivative} \quad \} \\
 & \langle \exists i :: x \in \partial_w (N.i) \rangle \\
= & \{ \quad \text{definition of set union} \quad \} \\
 & x \in \langle \cup i :: \partial_w (N.i) \rangle \quad .
\end{array}
$$

The lemma follows by definition of set membership.
□

Lemma 14 For an arbitrary subset N of the carrier set of a monoid,

$$\partial_w(\neg N) \;=\; \neg(\partial_w N) \quad .$$

Proof We have, for all x,

$$
\begin{array}{ll}
 & x \in \neg(\partial_w N) \\
= & \{ \quad \text{definition of set complement and derivative} \quad \} \\
 & \neg(w \cdot x \in N) \\
= & \{ \quad \text{definition of set membership,} \\
 & \quad\quad \text{distributivity of negation over existential quantification} \quad \} \\
 & \langle \forall y : y \in N : \neg(w \cdot x = y) \rangle \\
= & \{ \quad \text{trading} \quad \} \\
 & \langle \forall y : w \cdot x = y : \neg(y \in N) \rangle \\
= & \{ \quad \text{one-point rule, definition of set complement} \quad \} \\
 & w \cdot x \in \neg N \\
= & \{ \quad \text{definition of derivative} \quad \} \\
 & x \in \partial_w(\neg N) \quad .
\end{array}
$$

The lemma follows by definition of set membership.
□

It is an immediate corollary of lemmas 13 and 14 that

$$\partial_w \langle \cap i :: N.i \rangle \;=\; \langle \cap i :: \partial_w(N.i) \rangle$$

(since set intersection is the conjugate of set union). Indeed, as Brzozowksi's points out [Brz64, appendix I], the operation of taking derivatives distributes through an arbitrary Boolean function of events — where by "Boolean function" is meant any composition of supremum, infimum and complements.

Using the equation $N\backslash Q = \langle \cap w : w{\in}N : \partial_w Q \rangle$, the above lemmas enable the practical calculation of factors, especially in the case that the carrier set of the monoid M is finite. Example 16 provides a good illustration, and forms our second running example. (See also examples 16, 67, 81, 88, 169.)

Example 15 (Running Example: Booleans) The simplest possible powerset algebra is constructed by taking the underlying monoid to be the "trivial" monoid with exactly one element. That is, let M equal $\{1\}$ and define the product $1{\cdot}1$ to be 1. Then 2^M has two elements, the empty set, $\emptyset$, and $\{1\}$. Let N be an element of 2^M. (So N is $\emptyset$ or $\{1\}$.) Then $\emptyset\backslash N = N/\emptyset = \{1\}$. Also, $\partial_1 N = N$. Hence, $\{1\}\backslash N = N/\{1\} = N$.

Via the mapping $\emptyset {\mapsto} \mathsf{false}$ and $\{1\} {\mapsto} \mathsf{true}$, the above powerset algebra is isomorphic to the regular algebra of Booleans introduced in example 4. That is, the regular algebra of Booleans is the simplest possible example of a powerset regular algebra: it is the powerset regular algebra with underlying monoid the "trivial" monoid.
$\square$

Example 16 (Running Example: Modulo Addition) Let m be a strictly positive natural number. As is of course very well known, the numbers $0\,..\,m{-}1$ form an Abelian group, commonly denoted by $\mathbb{Z}_m$, and thus a monoid, under addition modulo m.

The group (monoid) $\mathbb{Z}_m$ is generated by $\{n\}$ where n is any number that is coprime with m. For example, $\{2\}$ is a generator set for $\mathbb{Z}_3$ but not for $\mathbb{Z}_6$. Of course, $\{1\}$ is always a generator set[3]. See example 140 for further discussion of the generator set.

Let us denote addition modulo m by the symbol $\oplus$ and subtraction modulo m by the symbol $\ominus$ (both written as infix operators). Extend addition to sets by defining

$$I{\oplus}J \;=\; \langle \cup i,j : i{\in}I \wedge j{\in}J : \{i{\oplus}j\} \rangle$$

for all subsets I and J of $\{0\,..\,m{-}1\}$. This is the basis of the definition of the powerset regular algebra with underlying monoid $\mathbb{Z}_m$. The carrier set is $2^{\{0\,..\,m-1\}}$, the set of subsets of $\{0\,..\,m{-}1\}$. The operator $\oplus$, as just defined, is the product operator. And sets are ordered by set inclusion. Details of the factor operators are as follows.

[3]Here "1" denotes the number 1 and not the unit of the monoid. Overloading of notation can sometimes be confusing!

If I is a subset of $\{0..m-1\}$ and j is an element of $\{0..m-1\}$, the *derivative* of I with respect to j, denoted by $\partial_j I$ is defined by

$$\partial_j I \;=\; \langle \cup i : i \in I : \{i \ominus j\} \rangle$$

and if J is a subset of $\{0..m-1\}$, the *factor* $\frac{I}{J}$ is defined by

$$\frac{I}{J} \;=\; \langle \cap j : j \in J : \partial_j I \rangle \quad .$$

Note carefully: the right side is an intersection of derivatives, not a union. We have chosen the notation $\frac{I}{J}$ here because addition is symmetric and so the right factor $J\backslash I$ and the left factor I/J are equal. (Of course, subtraction is not symmetric.)

As a concrete example, suppose $m=6$. Consider the powerset regular algebra of numbers modulo 6 under addition. Then, taking advantage of lemma 14, we have $\{1,2,3,4,5\} = \neg\{0\}$, so

$$\partial_1\{1,2,3,4,5\} = \partial_1(\neg\{0\}) = \neg\partial_1\{0\} = \neg\{0\ominus 1\} = \neg\{5\} \quad .$$

Similarly, $\partial_2\{1,2,3,4,5\} = \neg\{0\ominus 2\} = \{0,1,2,3,5\}$, etc.
Also,

$$\frac{\{1,2,3,4,5\}}{\{1,2\}} \;=\; \neg\{0\ominus 1\} \cap \neg\{0\ominus 2\} \;=\; \neg(\{0\ominus 1\} \cup \{0\ominus 2\}) \;=\; \neg\{5,4\} \;=\; \{0,1,2,3\} \quad .$$

In general,

$$\frac{\{1,2,3,4,5\}}{J} \;=\; \neg\,\langle \cup j : j \in J : \{0\ominus j\} \rangle \quad .$$

Now, for arbitrary subset K of $\{0,1,2,3,4,5\}$, we have:

$$
\begin{array}{ll}
 & K \;=\; \neg\,\langle \cup j : j \in J : \{0\ominus j\} \rangle \\
= & \qquad \{ \qquad \text{set complement} \qquad \} \\
 & \neg K \;=\; \langle \cup j : j \in J : \{0\ominus j\} \rangle \\
= & \qquad \{ \qquad \text{definition of set membership} \qquad \} \\
 & \langle \cup i : i \in \neg K : \{i\} \rangle \;=\; \langle \cup j : j \in J : \{0\ominus j\} \rangle \\
\Leftarrow & \qquad \{ \qquad 0\ominus i = j \equiv i = 0\ominus j \qquad \} \\
 & J \;=\; \langle \cup i : i \in \neg K : \{0\ominus i\} \rangle \quad .
\end{array}
$$

Thus,

$$K \;=\; \frac{\neg\{0\}}{\langle \cup i : i \in \neg K : \{0\ominus i\} \rangle} \quad .$$

It follows that every subset of $\{0,1,2,3,4,5\}$ is a factor of $\{1,2,3,4,5\}$ and so $\{1,2,3,4,5\}$ has 2^6 factors.

In general, for arbitrary m, the same argument shows that the set $\{1\,..\,m{-}1\}$ in the powerset regular algebra of numbers under addition modulo m has 2^m factors.
□

We conclude this section with our third running example: an example of a regular language, which was Conway's sole interest.

Example 17 (Running Example: The Language $(aa)^*$) Let alphabet T comprise the single symbol a and consider the language $(aa)^*$ (the set of all words whose length is divisible by 2). This language has just two distinct derivatives: we have that, for all words w in $(aa)^*$, $\partial_w(aa)^* = (aa)^*$ (for example, $\partial_{aa}(aa)^* = (aa)^*$) and, for all words w in $a(aa)^*$, $\partial_w(aa)^* = a\,(aa)^*$ (for example, $\partial_a(aa)^* = a\,(aa)^*$). Thus, using the formula $N\backslash Q = \langle\cap w : w{\in}N : \partial_w Q\rangle$ for arbitrary languages N and Q, it is easy to verify that there are just four possibilities for $N\backslash(aa)^*$, as N ranges over arbitrary languages: these are $\emptyset$ if N contains both a word in $(aa)^*$ and a word in $a(aa)^*$, $(aa)^*$ if N is a subset of $(aa)^*$, $a(aa)^*$ if N is a subset of $a(aa)^*$, and a^* if N is the empty set.
□

2.5 Set-Valued Matrices

We can, of course, combine the construction of square matrices (section 2.3) with the construction of power-set algebras (section 2.4): if $(A,\cdot,1)$ is a monoid, we can consider square matrices (of dimension $I{\times}I$ for some I) over the powerset algebra with underlying monoid $(A,\cdot,1)$. (That is, functions with domain $I{\times}I$ and event set 2^A.) This is the carrier set of a regular algebra which, for brevity, we denote by $\mathcal{M}_I(A)$.

We use this construction extensively with two particular instances. The first is the standard example of a "Kleene Algebra" introduced in example 12: the languages (sets of "words") over alphabet T. The second is where the monoid is the so-called "syntactic monoid" of a language. (Those not already familiar with the syntactic monoid are recommended to look ahead to section 7.)

Powerset algebras are, of course, *complemented*: there is a "complement" or "negation" operator "$\neg$" such that, for all S, $S\cap\neg S = \emptyset$ and $S\cup\neg S = \neg\emptyset$. Powerset-matrices are also complemented whereby the negation operator is lifted pointwise. That is, $(\neg f)(i,j)$ is defined to be $\neg(f(i,j))$. In this case, we overload the operator and write $\neg f(i,j)$; this allows us to silently exploit the ambiguity in the notation.

We specialise the discussion to languages and their syntactic monoid from section 6 onwards. However, part of the current investigation has been to explore how far the theory of factor matrices can be developed for events in the more general context of (matrices over) an arbitrary powerset algebra. This is indeed the case for all the theory that precedes section 6.

2.6 Notational Considerations

In this paper we consider several different instances of a regular algebra, often at the same time, and this leads to notational issues about how to denote the many different product and supremum operators. Inevitably, we are forced to overload operator symbols and rely on the reader being able to identify which operator is meant. It is important that the reader takes care to do so in order to determine how the operator is defined.

We commonly use $\mathcal{R}$ and $\mathcal{S}$ for arbitrary regular algebras. When we do, we use A and B for their carrier sets. When only one such algebra is involved, we use the notation of definition 1 to denote the product and factorisation operators, and the ordering relation on the carrier set. We also use Σ, as in (5), for the general supremum operator, and $+$, as in (6) for the binary supremum. When two such algebras are involved in the discussion, we subscript the operator symbols to distinguish them — but we do not add subscripts to the factorisation operators.

When we specialise the discussion to powerset algebras, we use the conventional notation for set union and the subset relation, as in section 2.4. When we specialise the discussion to matrices, we use $\otimes$ for the product operator, as in definition 8. (It is important to have a special symbol for matrix multiplication because of its special definition.) We add dots above other operator symbols to indicate the pointwise extension of the operator in the underlying regular algebra of elements. So, for example, when we consider an algebra of matrices with elements from a powerset algebra, we use $\dot{\subseteq}$ for the ordering relation and $\dot{\cup}$ for the supremum operator.

We use the notation $f \bullet g$ for the composition of functions f and g, and we use $U \circ V$ for the composition of relations U and V. The notation $U^{\cup}$ is used for the converse of relation U. When we wish to view a function as a relation, we use relational notation: so, for example, we might write $f^{\cup} \circ U \circ g$ (where f and g are functions and U is a relation, all of appropriate type). This is the relation defined by

$$x\ (f^{\cup} \circ U \circ g)\ y \;\equiv\; f.x\ U\ g.y$$

for all x and y (of appropriate type).

2.7 Advanced Properties

In this section we anticipate the use of the syntactic monoid of a language in calculations on the factor matrix/graph of a language. The most significant applications begin in section 7.2 but we also exploit the concepts introduced here in earlier sections.

In general terms, the theorems in this section relate calculations in different regular algebras $\mathcal{R}$ and $\mathcal{S}$. The specific application we have in mind is where $\mathcal{R}$ is the regular algebra of languages and $\mathcal{S}$ is the powerset algebra constucted from syntactic monoid of a regular language: typically, a regular language is an infinite set whereas the syntactic monoid is finite (and, hence, so is its powerset). In such cases, the objective is to translate calculations in the algebra $\mathcal{R}$ into calculations in the algebra $\mathcal{S}$ and then "invert" the results of the calculations back into $\mathcal{R}$.

Definition 18 (Monoid Homomorphism) Suppose $\mathcal{R} = (A_{\mathcal{R}}\,,\cdot_{\mathcal{R}}\,,1_{\mathcal{R}})$ and $\mathcal{S} = (A_{\mathcal{S}}\,,\cdot_{\mathcal{S}}\,,1_{\mathcal{S}})$ are monoids. Suppose ζ is a function with domain $A_{\mathcal{R}}$ and range $A_{\mathcal{S}}$. Then, ζ is said to be a *monoid homomorphism* from $\mathcal{R}$ to $\mathcal{S}$ if ζ preserves units:

$$\zeta.1_{\mathcal{R}} = 1_{\mathcal{S}}$$

and preserves product: for all x and y in $A_{\mathcal{R}}$,

$$\zeta.(x \cdot_{\mathcal{R}} y) = \zeta.x \cdot_{\mathcal{S}} \zeta.y \quad .$$

The homomorphism is said to be *surjective* (or *onto*) if $\zeta \circ \zeta^{\cup} = \mathsf{Id}_{A_{\mathcal{S}}}$.
□

Definition 19 (Regular Homomorphism) Let $\mathcal{R}$ and be regular algebras. Suppose ζ is a function with domain $A_{\mathcal{R}}$ and range $A_{\mathcal{S}}$. Then, ζ is a *regular homomorphism* from $\mathcal{R}$ to $\mathcal{S}$ if ζ is a monoid homomorphism (from $(A_{\mathcal{R}}\,,\cdot_{\mathcal{R}}\,,1_{\mathcal{R}})$ to $(A_{\mathcal{S}}\,,\cdot_{\mathcal{S}}\,,1_{\mathcal{S}})$) and it is the lower adjoint in a Galois connection of the orderings $\preceq_{\mathcal{R}}$ and $\preceq_{\mathcal{S}}$.
□

Among several significant consequences of the unity-of-opposites theorem of Galois connections is the property that the image set of a lower adjoint is a complete lattice if the domain of the adjoint is complete. The following theorem is a straightforward application of this property.

Theorem 20 Suppose $\mathcal{R} = (A\,,\cdot\,,\Sigma\,,\leq\,,0\,,1)$ is a regular algebra, and $\mathcal{S} = (B\,,\preceq)$ is a partially ordered set. Suppose B is closed under a binary product operator "$\otimes$". Suppose m is a function with domain A and range B that is compositional, i.e. for all x and y in A

$$m.(x \cdot y) = m.x \otimes m.y \quad ,$$

and is the lower adjoint in a Galois connection between the orderings. Let $m^{\sharp}$ denote its upper adjoint and let $m.A$ be the image of A under m. Then $m.\mathcal{R} = (m.A\,,\ \otimes,$

$\oplus$, $\preceq$, $m.0$, $m.1$) is a regular algebra, where the supremum operator in $\mathcal{S}$ is given by, for all functions f with range B ,

$$\oplus f \;=\; m.\Sigma(m^\sharp \bullet f) \;\; .$$

Moreover, m is a regular homomorphism from $\mathcal{R}$ to $m.\mathcal{R}$.

□

Lemma 21 Suppose $\mathcal{R}$ is the powerset algebra with underlying monoid ($R, \times_R, 1_R$). Suppose $\mathcal{S}$ is a regular algebra with monoid structure ($S, \times_S, 1_S$) and suppose ζ is a monoid homomomorphism from ($R, \times_R, 1_R$) to ($S, \times_S, 1_S$). Define the *extension* $\zeta^\flat$ from 2^R to S by

$$\zeta^\flat.X \;=\; \langle \Sigma x : x{\in}X : \zeta.x \rangle$$

for all X in 2^R . Then $\zeta^\flat$ is a regular homomorphism from $\mathcal{R}$ to $\mathcal{S}$ with upper adjoint $\zeta^\sharp$ defined by

$$\zeta^\sharp.U \;=\; \langle \cup x \;:\; \zeta.x \preceq U \;:\; \{x\} \rangle$$

for all U in S .

Proof We have to show that the extension $\zeta^\flat$ is a monoid homomorphism and that it is a lower adjoint in a Galois connection between the two orderings. That $\zeta^\flat$ is a monoid homomorphism follows straightforwardly from the fact that product in the algebra $\mathcal{S}$ is universally distributive[4].

$$\begin{array}{cl}
 & \zeta^\flat.(X \times_R Y) \\
= & \{ \quad \text{definition of } \zeta^\flat \quad \} \\
 & \langle \Sigma z \;:\; z \in X \times_R Y \;:\; \zeta.z \rangle \\
= & \{ \quad \text{definition of } X \times_R Y \quad \} \\
 & \langle \Sigma\, x{,}y : x{\in}X \wedge y{\in}Y : \zeta.(x \times_R y) \rangle \\
= & \{ \quad \zeta \text{ is a monoid homomorphism} \quad \} \\
 & \langle \Sigma\, x{,}y : x{\in}X \wedge y{\in}Y : \zeta.x \;\times_S\; \zeta.y \rangle \\
= & \{ \quad \text{product in } \mathcal{S} \text{ is universally distributive} \quad \} \\
 & \langle \Sigma x : x{\in}X : \zeta.x \rangle \;\times_S\; \langle \Sigma y : y{\in}Y : \zeta.y \rangle \\
= & \{ \quad \text{definition} \quad \} \\
 & \zeta^\flat.X \;\times_S\; \zeta^\flat.Y \;\; .
\end{array}$$

[4]Universal distributivity is a consequence of admitting factorisation.

The construction of its upper adjoint $\zeta^\sharp$ proceeds as follows.

$$\begin{array}{ll}
& \zeta^\flat.X \preceq U \\
= & \{ \quad \text{definition of } \zeta^\flat \text{, definition of supremum} \quad \} \\
& \langle \forall x : x{\in}X : \zeta.x \preceq U \rangle \\
= & \{ \quad \text{set comprehension} \quad \} \\
& X \subseteq \{x \mid \zeta.x \preceq U\} \\
= & \{ \quad \text{by definition, } \zeta^\sharp.U = \{x \mid \zeta.x \preceq U\} \quad \} \\
& X \subseteq \zeta^\sharp.U \ .
\end{array}$$

□

Theorem 20 was stated and proved in [Bac06, theorem 6.2]. Lemma 21 was also stated there [Bac06, theorem 6.3] but the construction of the upper adjoint was omitted. We have included its construction here because we need the details later.

Recall that our objective is to translate calculations in an algebra $\mathcal{R}$ into calculations in an algebra $\mathcal{S}$ and then "invert" the results of the calculations back into $\mathcal{R}$. This is facilitated when there is a regular homomorphism from $\mathcal{R}$ into $\mathcal{S}$ but sometimes this is not enough. Suppose that $\zeta^\flat$ is such a regular homomorphism and $\zeta^\sharp$ is its upper adjoint. Then, in general $\zeta^\sharp$ and $\zeta^\flat$ are not inverse functions: it is the case that, for all Y, $Y \preceq \zeta^\sharp.(\zeta^\flat.Y)$ but the converse inclusion is not generally valid. The sets Y such that $Y = \zeta^\sharp.(\zeta^\flat.Y)$ are called *closed* elements of the Galois connection and enjoy special properties. For our purposes, lemma 22 is one that we exploit.

Lemma 22 Suppose $\mathcal{R}$ and $\mathcal{S}$ are regular algebras and $\zeta^\flat$ is a regular homomorphism from $\mathcal{R}$ to $\mathcal{S}$. Suppose $\zeta^\sharp$ is its upper adjoint. Suppose Z in $\mathcal{R}$ has the property that

$$\zeta^\sharp.(\zeta^\flat.Z) = Z \ .$$

(In words, Z is a *closed* element of the Galois connection.) Then, for all X in $\mathcal{R}$,

$$X\backslash Z \;=\; \zeta^\sharp.(\zeta^\flat.X \setminus \zeta^\flat.Z) \;=\; \zeta^\sharp.(\zeta^\flat.(X\backslash Z)) \ , \tag{23}$$

$$Z/X \;=\; \zeta^\sharp.(\zeta^\flat.Z \,/\, \zeta^\flat.X) \;=\; \zeta^\sharp.(\zeta^\flat.(Z/X)) \quad \text{and} \tag{24}$$

$$X\backslash Z/Y \;=\; \zeta^\sharp.(\zeta^\flat.X \setminus \zeta^\flat.Z \,/\, \zeta^\flat.Y) \;=\; \zeta^\sharp.(\zeta^\flat.(X\backslash Z/Y)) \ . \tag{25}$$

NB: The symbols "\" and "/" are overloaded: on the left of the three equations it is the factorisation operator in $\mathcal{R}$ and on the right of each equation it is the factorisation operator in $\mathcal{S}$. We also overload symbols denoting the product and partial ordering relations in the proof below.

In words, if Z is a closed element of the Galois connection, then all right factors, all left factors, and all factors of Z are closed elements of the Galois connection.

Proof We present the proof of (25). The proofs of (23) and (24) are similar but slightly less complicated. (They involve one less variable.)

Suppose W, X, Y and Z are all elements of $\mathcal{R}$ and $\zeta^\sharp.(\zeta^\flat.Z) = Z$. Then

$$
\begin{array}{cl}
 & W \preceq \zeta^\sharp.(\zeta^\flat.X \setminus \zeta^\flat.Z \; / \; \zeta^\flat.Y) \\
= & \{ \quad \zeta^\sharp \text{ is upper adjoint of } \zeta^\flat \quad \} \\
 & \zeta^\flat.W \preceq \zeta^\flat.X \setminus \zeta^\flat.Z \; / \; \zeta^\flat.Y \\
= & \{ \quad \text{factors} \quad \} \\
 & \zeta^\flat.X \cdot \zeta^\flat.W \cdot \zeta^\flat.Y \preceq \zeta^\flat.Z \\
= & \{ \quad \zeta^\flat \text{ is a monoid homomorphism} \quad \} \\
 & \zeta^\flat.(X{\cdot}W{\cdot}Y) \preceq \zeta^\flat.Z \\
= & \{ \quad \zeta^\sharp \text{ is upper adjoint of } \zeta^\flat \quad \} \\
 & X{\cdot}W{\cdot}Y \preceq \zeta^\sharp.(\zeta^\flat.Z) \\
= & \{ \quad \text{assumption: } \zeta^\sharp.(\zeta^\flat.Z) = Z \quad \} \\
 & X{\cdot}W{\cdot}Y \preceq Z \\
= & \{ \quad \text{factors} \quad \} \\
 & W \preceq X\backslash Z/Y \ .
\end{array}
$$

It follows by indirect equality that $\zeta^\sharp.(\zeta^\flat.X \setminus \zeta^\flat.Z \; / \; \zeta^\flat.Y) = X\backslash Z/Y$. The second equality in (25) follows immediately from the first equality by applying the unity-of-opposites theorem (specifically $\zeta^\sharp \bullet \zeta^\flat \bullet \zeta^\sharp = \zeta^\sharp$).

□

The functions ζ, $\zeta^\flat$ and $\zeta^\sharp$ are extended pointwise to matrices simply by replacing function application by function composition. For example, if $\mathbf{G}$ is a matrix of dimension $I{\times}J$ and i and j are elements of I and J, $(\zeta^\sharp \bullet \mathbf{G})(i,j) = \zeta^\sharp.(\mathbf{G}(i,j))$. A basic property of the Galois connection $(\zeta^\flat, \zeta^\sharp)$ is that $(\zeta^\flat\bullet)$ is the lower adjoint and $(\zeta^\sharp\bullet)$ is the upper adjoint in a Galois connection of pointwise-ordered matrices. As a consequence, we have:

Theorem 26 Suppose $\mathcal{R}$ and $\mathcal{S}$ are regular algebras and $\zeta^\flat$ is a regular homomorphism from $\mathcal{R}$ to $\mathcal{S}$. Suppose $\zeta^\sharp$ is its upper adjoint. Suppose $\mathbf{G}$ and $\mathbf{H}$ are matrices of events in the algebra $\mathcal{R}$ (of the same dimension). Then, assuming that $\mathbf{G}$ and $\mathbf{H}$ have appropriate dimensions in each case,

$$\zeta^\flat \bullet (\mathbf{G}{+}\mathbf{H}) \;=\; (\zeta^\flat \bullet \mathbf{G}) + (\zeta^\flat \bullet \mathbf{H}) \quad ,$$

$$\zeta^\flat \bullet (\mathbf{G}{\otimes}\mathbf{H}) \;=\; (\zeta^\flat \bullet \mathbf{G}) \otimes (\zeta^\flat \bullet \mathbf{H}) \quad \text{, and}$$

$$\zeta^\flat \bullet \mathbf{G}^* \;=\; (\zeta^\flat \bullet \mathbf{G})^* \enspace .$$

Furthermore, if every event in $\mathbf{G}^*$ is closed,

$$\mathbf{G}^* \;=\; \zeta^\sharp \bullet (\zeta^\flat \bullet \mathbf{G})^* \enspace .$$

(For convenience, we make no distinction between the (matrix) product, supremum and star operators of $\mathcal{R}$ and $\mathcal{S}$ in the above statements. We also make no such distinction between the operators in the proof.)

Proof The first equation is an immediate consequence of the fact that lower adjoints preserve suprema, and supremum of matrices is defined pointwise.

The second equation is proved as follows:

$$\begin{array}{ll}
 & (\zeta^\flat \bullet (\mathbf{G} \otimes \mathbf{H}))(i,j) \\
= & \{ \quad \text{function composition} \quad \} \\
 & \zeta^\flat.((\mathbf{G} \otimes \mathbf{H})(i,j)) \\
= & \{ \quad \text{definition of matrix product (in } \mathcal{R}) \quad \} \\
 & \zeta^\flat.\langle \Sigma k :: \mathbf{G}(i,k) \cdot \mathbf{H}(k,j) \rangle \\
= & \{ \quad \zeta^\flat \text{ is a lower adjoint, so distributes over supremum} \quad \} \\
 & \langle \Sigma k :: \zeta^\flat.(\mathbf{G}(i,k) \cdot \mathbf{H}(k,j)) \rangle \\
= & \{ \quad \zeta^\flat \text{ is a monoid homomorphism} \quad \} \\
 & \langle \Sigma k :: \zeta^\flat.\mathbf{G}(i,k) \cdot \zeta^\flat.\mathbf{H}(k,j) \rangle \\
= & \{ \quad \text{definition of matrix product (in } \mathcal{S}) \quad \} \\
 & ((\zeta^\flat \bullet \mathbf{G}) \otimes (\zeta^\flat \bullet \mathbf{H}))(i,j) \enspace .
\end{array}$$

The third equation is proved using the well-known "fusion" theorem of fixed-point calculus. (See e.g. [Bac06, theorem 3.6].)

$$\begin{array}{ll}
 & \zeta^\flat \bullet \mathbf{G}^* \;=\; (\zeta^\flat \bullet \mathbf{G})^* \\
\Leftarrow & \{ \quad \zeta^\flat \text{ is a lower adjoint, } \mathbf{G}^* \text{ is the least fixed point of the function} \\
 & \qquad \text{mapping matrix } \mathbf{f} \text{ to } \mathbf{I} + \mathbf{G} + \mathbf{f} \otimes \mathbf{f} \;; \\
 & \qquad \text{fusion: [Bac06, theorem 3.6].} \quad \} \\
 & \langle \forall \mathbf{f} :: \zeta^\flat \bullet (\mathbf{I}_\mathcal{R} + \mathbf{G} + \mathbf{f} \otimes \mathbf{f}) \;=\; \mathbf{I}_\mathcal{S} + (\zeta^\flat \bullet \mathbf{G}) + ((\zeta^\flat \bullet \mathbf{f}) \otimes (\zeta^\flat \bullet \mathbf{f})) \rangle \\
\Leftarrow & \{ \quad \zeta^\flat \text{ is a lower adjoint and so preserves suprema} \quad \} \\
 & \zeta^\flat \bullet \mathbf{I}_\mathcal{R} = \mathbf{I}_\mathcal{S} \;\wedge\; \langle \forall \mathbf{f} :: \zeta^\flat \bullet (\mathbf{f} \otimes \mathbf{f}) \;=\; (\zeta^\flat \bullet \mathbf{f}) \otimes (\zeta^\flat \bullet \mathbf{f}) \rangle
\end{array}$$

$=\qquad\{\qquad \zeta^\flat$ is a monoid homomorphism, so $\zeta^\flat.1_{\mathcal{R}} = 1_{\mathcal{S}}$,
$\qquad\qquad\zeta^\flat$ is a lower adjoint, so $\zeta^\flat.0_{\mathcal{R}} = 0_{\mathcal{S}}$
$\qquad\qquad$hence $\zeta^\flat \bullet \mathbf{I}_{\mathcal{R}} = \mathbf{I}_{\mathcal{S}} \qquad \}$

$\langle \forall \mathbf{f} :: \zeta^\flat \bullet (\mathbf{f} \otimes \mathbf{f}) = (\zeta^\flat \bullet \mathbf{f}) \otimes (\zeta^\flat \bullet \mathbf{f}) \rangle$

$=\qquad\{\qquad$ second equation above with $\mathbf{G},\mathbf{H} := \mathbf{f},\mathbf{f} \qquad \}$

true .

The final equation is now straightforward:

$\mathbf{G}^*$

$=\qquad\{\qquad$ assumption: every element of $\mathbf{G}^*$ is closed $\qquad\}$

$\zeta^\sharp \bullet \zeta^\flat \bullet \mathbf{G}^*$

$=\qquad\{\qquad$ above $\qquad\}$

$\zeta^\sharp \bullet (\zeta^\flat \bullet \mathbf{G})^*$.

□

As remarked earlier, the statement of theorem 26 makes no notational distinction between the operators in the two algebras. However, the importance of the theorem is that there *is* a distinction. For example, the statement

$$\mathbf{G}^* = \zeta^\sharp \bullet (\zeta^\flat \bullet \mathbf{G})^*$$

expresses how to turn a calculation of the star of a matrix in the algebra $\mathcal{R}$ into the star of matrix in the algebra $\mathcal{S}$. A practical application of this is the computation of the factor matrix of a regular language by first cimputing the factor matrix in the powerset algebra of the syntactic monoid of the language. The former (typically) involves computations with infinite sets of words whereas the latter involves finite sets. See section 7.2.

2.8 Relations and Selectors

Suppose $\mathcal{R}$ is a regular algebra. As we remarked after theorem 9, homogeneous binary relations on a set form a regular algebra. For calculational purposes, it is useful to consider relations as special cases of events in a matrix algebra. Formally, given a regular algebra $\mathcal{R}$, we define the function Sel mapping relations to event matrices. In the definition, 1 denotes the unit of $\mathcal{R}$ and 0 its zero event (the least element of the complete lattice).

Definition 27 Suppose M and N are two sets and suppose R is a relation of type $M \leftarrow N$. Then we define the event matrix $Sel.R$ with dimension $M \times N$ by, for all $i \in M$ and $j \in N$,

$$(28) \qquad (i\ Sel.R\ j\ =\ 1\ \Leftarrow\ i\ R\ j)\ \wedge\ (i\ Sel.R\ j\ =\ 0\ \Leftarrow\ \neg(i\ R\ j))\ .$$

We call $Sel.R$ the *selector* corresponding to relation R.

We overload the converse operator on relations, denoted here by the symbol $\cup$ written as a postfix to its argument, to denote the *converse selector* $(Sel.R)\cup$; if R has type $M \leftarrow N$, this has dimension $N \times M$ and is defined by

$$(29) \qquad (Sel.R)\cup\ =\ Sel.(R\cup)$$

where ($R\cup$) denotes the converse of relation R. The conventional terminology for $(Sel.R)\cup$ is the *transpose* of matrix $Sel.R$.

□

We often apply the function Sel to functions, the function f of type $M \leftarrow N$ being regarded as a special kind of relation such that $i\ f\ j \equiv i = f.j$. In this case, the definition of Sel becomes

$$(i\ Sel.f\ j\ =\ 1\ \Leftarrow\ i = f.j)\ \wedge\ (i\ Sel.f\ j\ =\ 0\ \Leftarrow\ i \neq f.j)\ .$$

It is clear that Sel and Rel map identities to identities.

The use of Sel enables calculations to be so-called "point-free". That is, we can calculate with matrices without specific mention of the matrix elements. To this end, it is useful to formulate properties of relations in terms of Sel. The properties of particular interest are listed below. Suppose R is a relation of type $M \leftarrow N$. Let $\mathbf{I}_N$ denote the identity matrix of dimension $N \times N$ and $\mathbf{I}_M$ the identity matrix of dimension $M \times M$. Then

$$(30) \qquad R \text{ is functional } \equiv\ Sel.R \otimes (Sel.R)\cup\ \dot{\preceq}\ \mathbf{I}_M\ \ ,$$

$$(31) \qquad R \text{ is injective } \equiv\ (Sel.R)\cup \otimes Sel.R\ \dot{\preceq}\ \mathbf{I}_N\ \ ,$$

$$(32) \qquad R \text{ is surjective } \equiv\ \mathbf{I}_M\ \dot{\preceq}\ Sel.R \otimes (Sel.R)\cup\ \ ,$$

$$(33) \qquad R \text{ is total } \equiv\ \mathbf{I}_N\ \dot{\preceq}\ (Sel.R)\cup \otimes Sel.R\ \ .$$

As an example of how these properties are established, let us prove (32). In order to do so, we need the pointwise definition of surjective: for relation R of type $M \leftarrow N$,

$$(34) \qquad R \text{ is surjective } \equiv\ \langle \forall i : i \in M : \langle \exists k : k \in N : i\ R\ k \rangle \rangle\ \ .$$

Now we show that the right sides of (32) and (34) are equivalent:

$\mathbf{I}_M \;\dot{\preceq}\; Sel.R \otimes (Sel.R)^{\cup}$

$=\quad\{$ definition of pointwise ordering $\}$

$\langle\forall i,j : i{\in}M \wedge j{\in}M : i\ \mathbf{I}_M\ j \preceq i\ (Sel.R \otimes (Sel.R)^{\cup})\ j\rangle$

$=\quad\{$ $i\ \mathbf{I}_M\ i = 1$, otherwise $i\ \mathbf{I}_M\ j = 0$; $0 \preceq x$ for all x $\}$

$\langle\forall i : i{\in}M : 1 \preceq i\ (Sel.R \otimes (Sel.R)^{\cup})\ i\rangle$

$=\quad\{$ definition of matrix multiplication $\}$

$\langle\forall i : i{\in}M : 1 \preceq \langle\Sigma k : k{\in}N : (i\ Sel.R\ k) \cdot (k\ (Sel.R)^{\cup}\ i)\rangle\rangle$

$=\quad\{$ by definition of $Sel.R$ and $(Sel.R)^{\cup}$ and properties of multiplication,
$(i\ Sel.R\ k) \cdot (k\ (Sel.R)^{\cup}\ i) = 1 \;\Leftarrow\; i\ Sel.R\ k = 1$
and $(i\ Sel.R\ k) \cdot (k\ (Sel.R)^{\cup}\ i) = 0 \;\Leftarrow\; i\ Sel.R\ k = 0$,
addition is idempotent and $0 \neq 1$ $\}$

$\langle\forall i : i{\in}M : 0 \neq \langle\Sigma k : k{\in}N : i\ Sel.R\ k\rangle\rangle$

$=\quad\{$ by definition of $Sel.R$, $i\ Sel.R\ k = 1$ or $i\ Sel.R\ k = 0$,
$\langle\Sigma k : k{\in}N : 0\rangle = 0$ $\}$

$\langle\forall i : i{\in}M : \langle\exists k : k{\in}N : i\ Sel.R\ k = 1\rangle\rangle$

$=\quad\{$ definition of $Sel.R$ and (34) $\}$

R is surjective .

(Note that we have used infix notation for application of a matrix to a pair. For example, we write $i\ Sel.R\ k$ and not $(Sel.R)(i,k)$. This fits with the convention of using infix notation for relations, as in $i\ R\ k$. It is often helpful to use infix notation in this way and, where this is the case, this is what we do.)

Typically, we use the pointwise definition of surjectivity to establish that a relation is surjective and then use the point-free definition to derive consequences of the property. We do the same for injectivity. Its pointwise definition is:

(35) $\quad R$ is injective $\equiv \langle\forall i,j : i{\in}N \wedge j{\in}N : k\ R\ i \wedge k\ R\ j \Rightarrow i{=}j\rangle$.

We leave the reader to verify that (35) and (31) are equivalent. Functionality and totality are typically by definition so that we do not need the pointwise definitions.

A relation of type $M{\leftarrow}N$ that is functional and surjective is commonly referred to as a *function* from N *onto* M. It is easy to see from the point-free formulae that injectivity of relation R is equivalent to functionality of $R^{\cup}$; similarly totality of R is equivalent to surjectivity of $R^{\cup}$. A relation R of type $M{\leftarrow}N$ that is functional, injective, total and surjective is a *bijection* between M and N. A synonym for bijection is *(1–1) correspondence* (pronounced one-to-one correspondence).

Our use of the terminology "selector" alludes to one way that selectors are used. As an example, suppose $\mathbf{G}$ is a matrix of dimension $M{\times}N$ and suppose n is an element of N. Define the function $\mathbf{n}$ of type $N{\leftarrow}\mathbf{1}$, where $\mathbf{1}$ denotes a set having exactly one (anonymous) element, to be the constant function that always evaluates to n. Then, $\mathbf{G}\otimes Sel.\mathbf{n}$ is a matrix of dimension $M{\times}\mathbf{1}$; in the conventional terminology, it is the "column" of matrix $\mathbf{G}$ indexed by n. Similarly, if m is an element of M, $(Sel.\mathbf{m})^{\cup}\otimes\mathbf{G}$ is the "row" of matrix $\mathbf{G}$ indexed by m. Thus $(Sel.\mathbf{m})^{\cup}\otimes\mathbf{G}\otimes Sel.\mathbf{n}$ is a matrix of dimension $\mathbf{1}{\times}\mathbf{1}$ (i.e. a matrix with a single entry); ignoring the formal distinction between matrices of dimension $\mathbf{1}{\times}\mathbf{1}$ and matrix entries, it is the entry of $\mathbf{G}$ indexed by the pair (m,n). Rather than just selecting individual rows or columns of a matrix, we use Sel to select submatrices; the algebraic properties of Sel are exploited to combine concision with precision in our calculations. See [BC75, BvdEvG94] for further examples of such point-free calculations.

Another way that selectors are used is to construct so-called "pathwise homomorphisms" of graphs. Suppose that $\mathbf{G}$ is a matrix of dimension $M{\times}M$ for some M, and suppose f is a function of type $N{\leftarrow}M$. Then the matrix

$$Sel.f\otimes\mathbf{G}\otimes(Sel.f)^{\cup}$$

can be roughly described as a matrix formed by coalescing rows and columns of $\mathbf{G}$ that are mapped to the same value by the function f.

Fig. 2 illustrates both uses of selectors. At the top is a square matrix, depicted as a graph, that we will denote by $\mathbf{G}$. The index set of $\mathbf{G}$ is $\{0,1,2\}$. Now suppose f is the function of type $\{0,1,2\}{\leftarrow}\{X,Y\}$ defined by $f.X=1$ and $f.Y=2$. Then

$$(Sel.f)^{\cup}\otimes\mathbf{G}\otimes Sel.f$$

is the square matrix indexed by $\{X,Y\}$ defined by selecting the rows and columns of $\mathbf{G}$ indexed by $\{1,2\}$. Picturing the matrix as a graph, it is the graph on the bottom-left of fig. 2. Now suppose h is the function of type $\{X,Y\}{\leftarrow}\{0,1,2\}$ defined by $h.0=X$, $h.1=X$ and $h.2=Y$. Then the matrix

$$Sel.h\otimes\mathbf{G}\otimes(Sel.h)^{\cup}$$

is the matrix constructed from $\mathbf{G}$ by coalescing the nodes 0 and 1; the coalesced node is renamed X and the node 2 is renamed Y. The result is depicted at the bottom-right of fig. 2.

As an example of point-free calculations using selectors, and for later use, we have the following lemma. See [BC75, BvdEvG94] for similar examples of such point-free calculations.

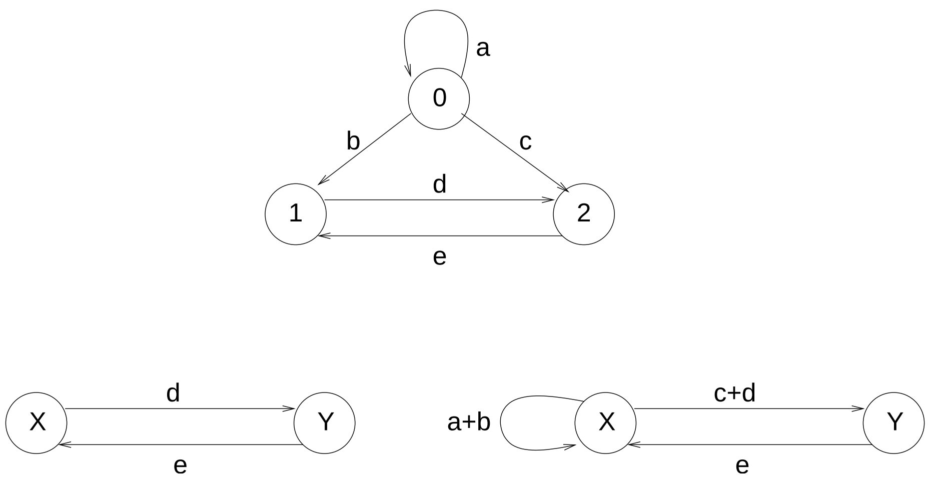


Figure 2: Use of Sel applied to functions

Lemma 36 Suppose $\mathbf{G}$ is an event matrix of dimension $M{\times}M$ and f is a total function of type $N{\leftarrow}M$. Then

$$Sel.f \otimes \mathbf{G}^* \otimes (Sel.f)^{\cup} \;\dot{\preceq}\; (Sel.f \otimes \mathbf{G} \otimes (Sel.f)^{\cup})^* \quad .$$

Proof

$$
\begin{array}{ll}
 & (Sel.f \otimes \mathbf{G} \otimes (Sel.f)^{\cup})^* \\
= & \{ \quad \text{definition of the star operator} \quad \} \\
 & \mathbf{I}_N \;+\; Sel.f \otimes \mathbf{G} \otimes (Sel.f)^{\cup} \otimes (Sel.f \otimes \mathbf{G} \otimes (Sel.f)^{\cup})^* \\
= & \{ \quad \text{leapfrog rule} \quad \} \\
 & \mathbf{I}_N \;+\; Sel.f \otimes \mathbf{G} \otimes ((Sel.f)^{\cup} \otimes Sel.f \otimes \mathbf{G})^* \otimes (Sel.f)^{\cup} \\
\dot{\succeq} & \{ \quad f \text{ is total; so, by (33), } (Sel.f)^{\cup} \otimes Sel.f \;\dot{\succeq}\; \mathbf{I}_M\,; \\
 & \quad\; \text{monotonicity of matrix product, star and supremum} \quad \} \\
 & \mathbf{I}_N \;+\; Sel.f \otimes \mathbf{G} \otimes (\mathbf{I}_M \otimes \mathbf{G})^* \otimes (Sel.f)^{\cup} \\
= & \{ \quad \mathbf{I}_M \text{ is the identity of matrix multiplication} \quad \} \\
 & \mathbf{I}_N \;+\; Sel.f \otimes \mathbf{G} \otimes \mathbf{G}^* \otimes (Sel.f)^{\cup} \\
\dot{\succeq} & \{ \quad f \text{ is functional; so, by (30), } \mathbf{I}_N \;\dot{\succeq}\; Sel.f \otimes (Sel.f)^{\cup} \quad \} \\
 & Sel.f \otimes (Sel.f)^{\cup} \;+\; Sel.f \otimes \mathbf{G} \otimes \mathbf{G}^* \otimes (Sel.f)^{\cup} \\
= & \{ \quad \text{distributivity, definition of the star operator} \quad \} \\
 & Sel.f \otimes \mathbf{G}^* \otimes (Sel.f)^{\cup} \quad .
\end{array}
$$

$\Box$

It is important to note that the definition of Sel depends critically on the type of the relation to which it is applied. For example, if A and B are two unequal sets such that $A \supseteq B$, we can define the injection $\iota_{A,B}$ of type $A \leftarrow B$ by $\iota_{A,B}.b = b$, for all b in B. The selector $Sel.\iota_{A,B}$ then has dimension $A \times B$; it is thus different from $Sel.\mathsf{id}_B$ where id_B is the identity function of type $B \leftarrow B$ defined by $\mathsf{id}_B.b = b$. This difference is, of course, crucial to our purpose of selecting submatrices of a given matrix.

Lemma 37 The function Sel is a monoid homomorphism. That is, Sel maps the identity relation (of a given type) to the identity event matrix (of the same type) and maps a composition of relations into a product of event matrices. Moreover, Sel is monotonic and commutes with converse.

Proof Clearly Sel maps the identity event matrix to the identity relation. Moreover, for all i and j of appropriate type, $Sel.R \otimes Sel.S$

$$
\begin{array}{ll}
 & i \; Sel.(R \circ S) \; j \\
= & \{ \quad \text{definition of } Sel\text{: (28)} \quad \} \\
 & \mathsf{if}\; i\;(R \circ S)\; j \rightarrow 1 \;\Box\; \neg(i\;(R \circ S)\; j) \rightarrow 0 \;\mathsf{fi} \\
= & \{ \quad \text{definition of composition of relations} \quad \} \\
 & \mathsf{if}\;\; \langle \exists k :: i\; R\; k \wedge k\; S\; j \rangle \rightarrow 1 \\
 & \Box\;\; \neg \langle \exists k :: i\; R\; k \wedge k\; S\; j \rangle \rightarrow 0 \\
 & \mathsf{fi} \\
= & \{ \quad \text{definition of } Sel\text{: (28)} \quad \} \\
 & \mathsf{if}\;\; \langle \exists k :: i\; Sel.R\; k = 1 \wedge k\; Sel.S\; j = 1 \rangle \rightarrow 1 \\
 & \Box\;\; \neg \langle \exists k :: i\; Sel.R\; k = 1 \wedge k\; Sel.S\; j = 1 \rangle \rightarrow 0 \\
 & \mathsf{fi} \\
= & \{ \quad i\; Sel.R\; k \neq 1 \equiv i\; Sel.R\; k = 0, \\
 & \quad\;\; k\; Sel.S\; j \neq 1 \equiv k\; Sel.S\; j = 0, \\
 & \quad\;\; \text{predicate calculus} \quad \} \\
 & \mathsf{if}\;\; \langle \exists k :: i\; Sel.R\; k = 1 \wedge k\; Sel.S\; j = 1 \rangle \rightarrow 1 \\
 & \Box\;\; \langle \forall k :: i\; Sel.R\; k = 0 \vee k\; Sel.S\; j = 0 \rangle \rightarrow 0 \\
 & \mathsf{fi} \\
= & \{ \quad 1 \text{ is unit of product, } 0 \text{ is zero of product and unit of addition,}
\end{array}
$$

$\qquad\qquad$ addition is idempotent,

$\qquad\qquad$ definition of product of transition graphs $\}$

$\quad i\ \ Sel.R \otimes Sel.S\ \ j\ \ .$

Next,

$$
\begin{array}{ll}
 & Sel.R \mathrel{\dot{\preceq}} Sel.S \\
= & \{ \quad \text{definition of pointwise ordering} \quad \} \\
 & \langle \forall i,j\ ::\ i\ Sel.R\ j \preceq i\ Sel.S\ j \rangle \\
= & \{ \quad \text{case analysis on values of } Sel \text{ (either } 0 \text{ or } 1\text{);} \\
 & \qquad 0 \preceq x \text{ for all } x \quad \} \\
 & \langle \forall i,j\ ::\ i\ Sel.R\ j = 1 \ \Rightarrow\ i\ Sel.S\ j = 1 \rangle \\
= & \{ \quad \text{definition of } Sel \quad \} \\
 & \langle \forall i,j\ ::\ i\ R\ j \ \Rightarrow\ i\ S\ j \rangle \\
= & \{ \quad \text{definition of ordering of relations} \quad \} \\
 & R \subseteq S\ \ .
\end{array}
$$

Finally,

$$
\begin{array}{ll}
 & i\ (Sel.R)^{\cup}\ j \\
= & \{ \quad \text{definition of transpose} \quad \} \\
 & j\ Sel.R\ i \\
= & \{ \quad \text{definition of } Sel.R\text{: (28)} \quad \} \\
 & \textsf{if}\ j\ R\ i \to 1\ \square\ \neg(j\ R\ i) \to 0\ \textsf{fi} \\
= & \{ \quad \text{definition of converse} \quad \} \\
 & \textsf{if}\ i\ R^{\cup}\ j \to 1\ \square\ \neg(i\ R^{\cup}\ j) \to 0\ \textsf{fi} \\
= & \{ \quad \text{definition of } Sel.R\text{: (28)} \quad \} \\
 & i\ Sel.(R^{\cup})\ j\ \ .
\end{array}
$$

□

3 The Factor Matrix

In this section, we introduce Conway's factor matrix and summarise a number of fundamental properties (due to Conway). Our presentation differs from Conway's in several ways. First, we have generalised Conway's theorems to arbitrary regular events (as opposed to regular languages) and give examples to illustrate the relevance of the generalisation. Second, our presentation is explicitly calculational. Third, Conway showed that there is a (1–1) correspondence between the so-called "left" and "right" "factors" of a given regular event; we observe the stronger property of an isomorphism between the posets of left and right factors. The definitions of factors, left and right factors of an event E are given in section 3.2.

In summary, for a given event E, Conway's factor matrix is a matrix indexed by the left factors (or equally the right factors) of E, this index set being finite in the case that E is a regular language (as opposed to A which may be infinite). Moreover, E itself and all left and right factors of E are elements of the matrix.

Before introducing the factor matrix formally in section 3.4, we present the isomorphism between the posets of left and right factors in section 3.3.

We begin in section 3.1 by listing a number of elementary properties of factorisation. All are easily verified. Typically the properties are instances of general properties of Galois connections. (Recall that we assume a good knowledge of the theory of Galois connections. See [Bac02] for the properties and terminology we assume.)

Throughout this section we omit the calculations that substantiate the claimed properties. This is because they have been given in detail elsewhere: see [Bac16].

3.1 Elementary Properties

Let X, Y and Z denote events in a regular algebra $\mathcal{R}$ as defined in definition 1. Because $(A,\preceq)$ is a complete lattice, all functions with range A have a supremum and an infimum. Denoting the supremum of f by Σf and the infimum of f by Πf, we have that product preserves suprema and the factor operators preserve infima. That is, for all functions f with range A,

$$(38)\qquad X\cdot(\Sigma f) \;=\; \langle \Sigma x :: X\cdot f.x\rangle$$

and

$$(39)\qquad X\backslash(\Pi f) \;=\; \langle \Pi x :: X\backslash f.x\rangle \quad\wedge\quad (\Pi f)/X \;=\; \langle \Pi x :: f.x/X\rangle \quad .$$

In particular, $(X\backslash)$ and $(/X)$ are both monotonic:

$$(40)\qquad X\backslash Y\preceq X\backslash Z \;\;\wedge\;\; Y/X\preceq Z/X \quad\Leftarrow\quad Y\preceq Z \quad .$$

If we combine the Galois connections in (2) and (3), we get: for all events X, Y and Z,

$$Z/Y\succeq X \quad\equiv\quad Y\preceq X\backslash Z \quad .$$

Note the switch in the ordering — rewritten in this way in order to more easily identify the Galois connection. It follows that, for all events Z and functions f with range A,

(41) $\quad Z/(\Sigma f) \;=\; \langle \Pi x :: Z\,/\,f.x \rangle \quad\wedge\quad (\Sigma f)\backslash Z \;=\; \langle \Pi x :: f.x \,\backslash\, Z \rangle \quad .$

In particular, $Z/$ and $\backslash Z$ are so-called “anti-monotonic” functions:

(42) $\quad X\backslash Z \preceq Y\backslash Z \;\;\wedge\;\; Z/X \preceq Z/Y \quad\Leftarrow\quad X \succeq Y \quad .$

(The prefix “anti” signifies the reversal of the ordering. More long-windedly but technically precise, $Z/$ and $\backslash Z$ are monotonic functions from $(A, \preceq)$ to $(A, \succeq)$.)

We frequently use *cancellation*:

(43) $\quad X \cdot X\backslash Y \preceq Y \;\;\wedge\;\; X/Y \cdot Y \preceq X \;\;\wedge\;\; Y \preceq X\backslash(X \cdot Y) \;\;\wedge\;\; X \preceq (X \cdot Y)/Y$

commutativity :

(44) $\quad (X \cdot Y)\backslash Z \;=\; Y\backslash(X\backslash Z) \;\;\wedge\;\; (X/Z)/Y \;=\; X/(Y \cdot Z) \quad ,$

and *associativity*:

(45) $\quad (X\backslash Y)/Z \;=\; X\backslash(Y/Z) \quad .$

Property (45) allows us to drop parentheses and write

$$X\backslash Y/Z$$

without ambiguity. We do this frequently, using (45) without explicit mention (in the same way that we write $X \cdot Y \cdot Z$ and exploit the associativity of concatenation without explicit mention).

Two less well-known cancellation properties are useful:

(46) $\quad X \preceq (Y/X)\backslash Y \quad\wedge\quad X \preceq Y/(X\backslash Y) \quad .$

By exploiting the fact that $(A, \cdot, 1)$ is a monoid (in particular 1 is a unit of product), the cancellation laws all become equalities in the case that the variables are identical:

(47) $\quad X\backslash(X \cdot X) \;=\; X \cdot X\backslash X \;=\; X \;=\; X/X \cdot X \;=\; (X \cdot X)/X \quad$, and

(48) $\quad (X/X)\backslash X = X = X/(X\backslash X) \quad .$

3.2 Left and Right Factors

In this section, we introduce the notion of a factor, a left factor and a right factor of an event. Theorem 50 is used extensively, although sometimes without explicit mention.

Definition 49 Let E denote a fixed event of a regular algebra. A *factor* of E is any event that can be expressed in the form $X\backslash E/Y$ for some X and Y. (That parentheses have been omitted here is permitted by virtue of (45).) An event is a *left factor* of E if it can be expressed in the form E/Y for some Y and a *right factor* of E if it can be expressed in the form $X\backslash E$ for some X.
□

Theorem 50 The relations factor-of, left-factor-of and right-factor-of are reflexive and transitive. (Reflexivity means that every event is a factor of, a left factor of, and a right factor of itself. Transitivity of the factor-of relation means that a factor of a factor of an event is a factor of the event. Similarly for left-factor-of and right-factor-of.)

Proof The proof is a simple, introductory exercise in the use of the laws given above. For example, the transitivity of the left-factor-of and right-factor-of relations follow from the commutativity property (44) and their reflexivity from (48). The reflexivity of the factor-of relation is a combination of the two equations in (48). Specifically,

(51) $\quad X = (X/X)\backslash X/(X\backslash X)$

for all events X. (Note how (45) is used here.) We leave the verification of factor-of relation to the reader.
□

Conway [Con71] states theorem 50 but his proof-style is quite different. The basic properties used are the same but the important difference is that our existence proofs are explicitly (rather than implicitly) constructive. For example, (51) establishes that an event is a factor of itself by explicitly exhibiting X/X and $X\backslash X$ as examples of events Y and Z such that $X = Y\backslash X/Z$. This difference in proof-style is, in my view, a significant contribution of the current paper.

3.3 Unity of Opposites

Throughout this and subsequent sections, the event E is a fixed, sometimes implicit, parameter of several definitions and theorems. From now on, we use lower-case identifiers (i, j, k etc.) to denote *left* factors of E. We continue to use upper-case letters at the end of the alphabet for arbitrary events.

Define the functions $\triangleleft$ and $\triangleright$ by

(52) $\quad X{\triangleleft} \;=\; E/X \;,$

(53) $\quad X{\triangleright} \;=\; X\backslash E \;.$

By definition, the range of $\triangleleft$ is the set of *left* factors of E and the range of $\triangleright$ is the set of *right* factors of E. It is an easy calculation to derive the Galois connection: for all X and Y,

(54) $\quad X \preceq Y{\triangleleft} \;\equiv\; Y \preceq X{\triangleright} \;.$

Note that because of the reversal of the ordering, both operators $\triangleleft$ and $\triangleright$ are anti-monotonic. That is,

(55) $\quad X{\triangleleft} \succeq Y{\triangleleft} \;\wedge\; X{\triangleright} \succeq Y{\triangleright} \quad\Leftarrow\quad X \preceq Y \;.$

Applying the unity-of-opposites theorem [Bac02] to the Galois connection (54), we deduce a (1–1) correspondence between the left and right factors of E:

(56) $\quad X{\triangleleft}{\triangleright}{\triangleleft} \;=\; X{\triangleleft} \;,$

(57) $\quad X{\triangleright}{\triangleleft}{\triangleright} \;=\; X{\triangleright} \;.$

and an isomorphism between the subset ordering on left factors and the superset ordering on right factors of E:

(58) $\quad X{\triangleleft} \preceq Y{\triangleleft} \;\equiv\; X{\triangleleft}{\triangleright} \succeq Y{\triangleleft}{\triangleright} \;,$

(59) $\quad X{\triangleright} \preceq Y{\triangleright} \;\equiv\; X{\triangleright}{\triangleleft} \succeq Y{\triangleright}{\triangleleft} \;.$

Additional properties are

(60) $\quad E{\triangleleft}{\triangleright} \;=\; E \;=\; E{\triangleright}{\triangleleft} \;,$

(61) $\quad X{\triangleleft} \backslash Y{\triangleleft} \;=\; X{\triangleleft}{\triangleright} \,/\, Y{\triangleleft}{\triangleright} \;,$

(62) $\quad X{\triangleright} / Y{\triangleright} \;=\; X{\triangleright}{\triangleleft} \,\backslash\, Y{\triangleright}{\triangleleft} \;,$

(63) $\quad X \backslash Y{\triangleright} \;=\; (Y{\cdot}X){\triangleright} \;,$

(64) $\quad X{\triangleleft} / Y \;=\; (Y{\cdot}X){\triangleleft} \;.$

The rightmost equality in (60) is established as follows. The proof of the leftmost equality is similar.

$$
\begin{array}{ll}
 & E{\rhd}{\lhd} = E \\
= & \quad \{ \quad \text{antisymmetry} \quad \} \\
 & E{\rhd}{\lhd} \subseteq E \ \wedge \ E{\rhd}{\lhd} \supseteq E \\
= & \quad \{ \quad X \subseteq Y{\lhd} \equiv Y \subseteq X{\rhd} \ \text{ with } \ X,Y := E\,,E{\rhd}\,, \\
 & \quad\quad\ \ \text{reflexivity of } \geq \quad \} \\
 & E{\rhd}{\lhd} \subseteq E \\
= & \quad \{ \quad \text{definition} \quad \} \\
 & E/(E\backslash E) \subseteq E \\
\Leftarrow & \quad \{ \quad E/ \ \text{ is an anti-monotonic function,} \quad E/1 = E \quad \} \\
 & E\backslash E \supseteq 1 \\
= & \quad \{ \quad \text{factors, unit} \quad \} \\
 & \text{true} \ .
\end{array}
$$

To establish (61) and (62), and for later use, it is convenient to observe that (45), appropriately instantiated, gives the identity

(65) $X\backslash(Z{\lhd}) \ = \ (X{\rhd})/Z$.

The calculation of (61) is now easy:

$$
\begin{array}{ll}
 & X{\lhd} \backslash Y{\lhd} \\
= & \quad \{ \quad (56) \quad \} \\
 & X{\lhd} \ \backslash \ Y{\lhd}{\rhd}{\lhd} \\
= & \quad \{ \quad (65) \quad \} \\
 & X{\lhd}{\rhd} \ / \ Y{\lhd}{\rhd} \ .
\end{array}
$$

Property (62) is calculated in the same way. We leave the reader the task of verifying (63) and (64).

Most properties in this section formulate as equations theorems that Conway expressed in words. So, for example, property (60) establishes the theorem that any event is both a left factor and a right factor of itself. Properties (56) and (57) establish a (1–1) correspondence between the left and right factors of E. However, Conway does not appear to have been aware of the Galois connection (54); the (1–1) correspondence is a cornerstone of his account but he does not observe the poset isomorphism expressed by (58) and (59).

Properties (61) and (62) will be discussed in the next section when we introduce the factor matrix.

Example 66 (Running Example: Booleans) Recall (example 4) that $\mathsf{Bool} = \{\mathsf{false},\mathsf{true}\}$ is a regular algebra and the over and under operators are, respectively, "if" and "only-if". Property (60) is thus

$$((E{\Leftarrow}E){\Rightarrow}E) \;=\; E \;=\; (E{\Leftarrow}(E{\Rightarrow}E)) \;\; .$$

The reader can easily check that this is a valid identity for both cases of $E = \mathsf{false}$ and $E = \mathsf{true}$. The reader is invited to also check the validity of all the properties (61) thru (64) for all booleans E, X and Y. (Recall that product is conjunction so that, for example, (64) is the property

$$((E{\Leftarrow}X) \Leftarrow Y) \;\;=\;\; (E \;\Leftarrow\; Y{\wedge}X)$$

for all booleans E, X and Y.)
□

Example 67 (Running Example: Modulo Addition) Let m be a strictly positive natural number and consider the powerset regular algebra with underlying monoid $\mathbb{Z}_m$, the Abelian group of numbers modulo m under addition. (See example 16.) As we have seen, the set $\neg\{0\}$ has 2^m factors. Recall that, because of the symmetry of addition, we use the notation $\frac{I}{J}$ for factorisation. In order to verify properties (60) thru (64) for an arbitrary subset E of $\{0\,..\,m{-}1\}$, it is necessary to consider three distinct cases: $E = \emptyset$, $E = \{0\,..\,m{-}1\}$ and $\emptyset \subset E \subset \{0\,..\,m{-}1\}$. We leave the first two cases to the reader. In the third case, $\frac{E}{E} = \{0\}$ and $\frac{E}{\{0\}} = E$. This establishes the validity of (60) in this case. Because of the symmetry of addition, $X{\lhd} = X{\rhd}$ for all X. Moreover, because the monoid is in fact a group, $X{\rhd}{\lhd} = X$. So properties (61) and (62) predict that $\frac{X\lhd}{Y\lhd} = \frac{Y}{X}$. We leave the reader to check these identities from the definitions. (Expanding the definition of the $\lhd$ operator, the properties look like familiar properties of division and subtraction in arithmetic; because of the very special nature of the example, this is what one would expect.)
□

3.4 Definition and Properties of the Factor Matrix

In this section, we define the factor matrix of an event and state a number of properties. The notion of the factor matrix and the properties listed in this section are due to Conway [Con71, chapter 6]. Our presentation differs considerably in that we continue to give explicit constructions witnessing existential properties. See [Bac16] for the relevant calculations.

Definition 68 (Factor Matrix) Let $\mathcal{L}.E$ denote the set of left factors of E. The *factor matrix* of E is defined to be the binary operator $\backslash$ restricted to $\mathcal{L}.E \times \mathcal{L}.E$. Thus, the factor matrix has dimension $\mathcal{L}.E \times \mathcal{L}.E$ and entries in the matrix take the form $i\backslash j$ where i and j are left factors of E. It is denoted by $|\overline{E}|$.

Formally, $|\overline{E}| = (Sel.(\iota_{A\,,\,\mathcal{L}.E}))^{\cup} \otimes \text{under} \otimes Sel.(\iota_{A\,,\,\mathcal{L}.E})$, where under is the matrix defined by $X \text{ under } Y = X\backslash Y$, for all events X and Y, and $\iota_{A\,,\,\mathcal{L}.E}$ is the function that injects left factors of E into the carrier set , A, of the algebra $\mathcal{R}$. (Pre-multiplying by $(Sel.(\iota_{A\,,\,\mathcal{L}.E}))^{\cup}$ and post-multiplying by $Sel.(\iota_{A\,,\,\mathcal{L}.E})$ selects the left factors of E.)
$\Box$

An equivalent definition of the factor matrix of E is the binary operator $/$ restricted to $\mathcal{T}.E \times \mathcal{T}.E$, where $\mathcal{T}.E$ denotes the set of right factors of E. Properties (61) and (62) encode the equivalence.

Suppose that F is a factor, i is a left factor, and R is a right factor of E. We now construct left factors i_0, i_1, i_2, i_3, i_4, i_5 such that

(69) $\quad F \;=\; i_0\backslash i_1$,

(70) $\quad i \;=\; i_2\backslash i_3$,

(71) $\quad R \;=\; i_4\backslash i_5$.

Moreover, i_2 is independent of i and i_5 is independent of R and

(72) $\quad E = i_2\backslash i_5$.

Suppose F is a factor of E. In particular, suppose that $F{=}U\backslash E/V$. We construct X and Y such that $F = X\lhd\backslash Y\lhd$ as follows.

$$
\begin{array}{ll}
 & X\lhd \backslash Y\lhd = U\backslash E/V \\
= & \qquad \{ \quad X\backslash Y\lhd = X\rhd / Y \text{ with } X,Y := X\lhd, Y,\ U\backslash E = U\rhd \quad \} \\
 & X\lhd\rhd / Y \;=\; U\rhd / V \\
\Leftarrow & \qquad \{ \quad \textit{Postulate}\ Y{=}V \quad \} \\
 & X\lhd\rhd = U\rhd \\
\Leftarrow & \qquad \{ \quad U\rhd\lhd\rhd \;=\; U\rhd \quad \} \\
 & X{=}U\rhd \;\; .
\end{array}
$$

Thus $U\backslash E/V = U\rhd\lhd\backslash V\lhd$.

Now consider the left factor $V\lhd$. This is written in the form $X\lhd\backslash Y\lhd$ as follows.

$$V\lhd$$

$=$ { definition }

E/V

$=$ { (60) }

$E{\lhd}{\rhd}\,/\,V$

$=$ { $X\backslash Y{\lhd} = X{\rhd}/Y$ with $X,Y := E{\lhd}, V$ }

$E{\lhd}\backslash V{\lhd}$.

Thus $V{\lhd} = E{\lhd}\backslash V{\lhd}$. Finally, consider the right factor $U{\rhd}$. We write this in the form $X{\lhd}\backslash Y{\lhd}$ as follows:

$U{\rhd}$

$=$ { definition }

$U\backslash E$

$=$ { (60) }

$U\backslash E{\rhd}{\lhd}$

$=$ { (65) with $X,Z := U, E{\rhd}$ }

$U{\rhd}\,/\,E{\rhd}$

$=$ { (57) }

$U{\rhd}{\lhd}{\rhd}\,/\,E{\rhd}$

$=$ { (65) with $X,Z := U{\rhd}{\lhd}\,,\,E{\rhd}$ }

$U{\rhd}{\lhd}\backslash E{\rhd}{\lhd}$.

Thus, $U{\rhd} = (U{\rhd}){\lhd}\backslash(E{\rhd}){\lhd}$.

The terms i_2 and i_5 in (70) and (71) are thus $E{\lhd}$ and E. The verification of (72) is then:

$E{\lhd}\backslash E$

$=$ { definition }

$E{\lhd}{\rhd}$

$=$ { (60) }

E .

By these explicit constructions, we have established Conway's theorem 4 [Con71, p.48]. Let us do some renaming in order to make it easier to compare our formulation of the theorem with Conway's. As always, E denotes a fixed "event" (thus not necessarily a regular language). Consider the "matrix" defined by the binary operator $\backslash$ indexed by left factors of E. So, each "entry" in the matrix has the form $i\backslash j$ for some left factors i and j of E. Conway uses the notation E_{ij} for such an entry. Then each entry is a factor of E since j is a left factor of E equivales $j = j{\rhd}{\lhd} = E/j{\rhd}$ and, hence, $i\backslash j = i\backslash E/j{\rhd}$. Moreover, we have shown that each factor of E is an entry in the matrix, specifically:

(73) $\quad U\backslash E/V \;=\; U{\rhd}{\lhd}\backslash V{\lhd}\;\;.$

In addition, let l denote the left factor $E{\lhd}$ and r denote E (which is a left factor of E since $E = E{\rhd}{\lhd}$). Then

(74) $\quad E = r = l{\rhd} = l\backslash r\;\;.$

In words, E is the left factor r, the right factor corresponding to l, and the (l,r)th entry in the matrix. Also, for all left factors i of E

(75) $\quad i = l\backslash i$

and

(76) $\quad i{\rhd} = i\backslash r\;\;.$

In words, the left factor i is the (l,i)th entry in the matrix and its corresponding right factor $i{\rhd}$ is the (i,r)th entry in the matrix.

We conclude by showing that —in Conway's words— any subfactorisation of E is dominated by a factorisation of E. In particular, we show that:

(77) $\quad A{\cdot}B \subseteq E \;\;\equiv\;\; A \subseteq B{\lhd} \wedge B \subseteq B{\lhd}{\rhd}\;\;.$

More generally, we show that, for all X and Y and all U and V,

(78) $\quad X{\cdot}Y \subseteq U{\lhd}\backslash V{\lhd} \;\;\equiv\;\; \langle \exists W :: X \subseteq U{\lhd}\backslash W{\lhd} \wedge Y \subseteq W{\lhd}\backslash V{\lhd}\rangle\;\;.$

The proof of (77) is:

$$
\begin{array}{ll}
 & A{\cdot}B \subseteq E \\
= & \{\quad \text{factors, } B{\lhd} = E/B\,;\ \text{cancellation} \quad\} \\
 & A \subseteq B{\lhd} \wedge B{\lhd}{\cdot}B \subseteq E \\
= & \{\quad \text{factors, } B{\lhd}{\rhd} = B{\lhd}\backslash E \quad\} \\
 & A \subseteq B{\lhd} \wedge B \subseteq B{\lhd}{\rhd}\;\;.
\end{array}
$$

Whence, we prove (78). First, we determine a specific instance for the existentially quantified variable W:

$$
\begin{array}{ll}
 & X{\cdot}Y \subseteq U{\triangleleft}\backslash V{\triangleleft} \\
= & \{ \quad \text{factors, definition of } V{\triangleleft} \quad \} \\
 & U{\triangleleft}\cdot X\cdot Y\cdot V \subseteq E \\
= & \{ \quad \text{(77) with } A,B := U{\triangleleft}\cdot X\,,\, Y{\cdot}V \quad \} \\
 & U{\triangleleft}\cdot X \subseteq (Y{\cdot}V){\triangleleft} \;\wedge\; Y{\cdot}V \subseteq (Y{\cdot}V){\triangleleft\triangleright} \\
= & \{ \quad \text{factors} \quad \} \\
 & X \subseteq U{\triangleleft}\backslash(Y{\cdot}V){\triangleleft} \;\wedge\; Y \subseteq (Y{\cdot}V){\triangleleft\triangleright}/V \\
= & \{ \quad (65) \quad \} \\
 & X \subseteq U{\triangleleft}\backslash(Y{\cdot}V){\triangleleft} \;\wedge\; Y \subseteq (Y{\cdot}V){\triangleleft}\backslash V{\triangleleft} \;\; .
\end{array}
$$

Then, we have:

$$
\begin{array}{ll}
 & X{\cdot}Y \subseteq U{\triangleleft}\backslash V{\triangleleft} \\
\Rightarrow & \{ \quad \text{above, } W{:=}Y{\cdot}V \quad \} \\
 & \langle \exists W \;::\; X \subseteq U{\triangleleft}\backslash W{\triangleleft} \wedge Y \subseteq W{\triangleleft}\backslash V{\triangleleft} \rangle \\
\Rightarrow & \{ \quad \text{monotonicity of composition} \quad \} \\
 & \langle \exists W \;::\; X{\cdot}Y \subseteq (U{\triangleleft}\backslash W{\triangleleft})\cdot(W{\triangleleft}\backslash V{\triangleleft}) \rangle \\
\Rightarrow & \{ \quad \text{cancellation} \quad \} \\
 & X{\cdot}Y \subseteq U{\triangleleft}\backslash V{\triangleleft} \;\; .
\end{array}
$$

Let $|\overline{E}|$ denote the factor matrix of E. From (78) it follows that

$$|\overline{E}|\otimes|\overline{E}| \;\dot{\preceq}\; |\overline{E}|$$

and, since obviously $1 \preceq i\backslash i$ for all i,

$$\mathbf{I} \;\dot{\preceq}\; |\overline{E}|$$

(where $\mathbf{I}$ denotes the identity matrix). Hence

(79) $\quad |\overline{E}| = |\overline{E}|^{*}$.

This completes our formulation of Conway's theorems [Con71]. A "matrix" has been exhibited containing all factors and only the factors of E, indexed by left factors of E, that is reflexive and transitive and hence equal to its own star. The import of (75) and (76) is that the $E\lhd$ "row" of the matrix (the set of entries all having $E\lhd$ as first index) contains all (and only) the left factors of E, and the E "column" of the matrix (the set of entries all having E as second index) all (and only) the right factors of E. In addition, from (72) we see that E is the matrix entry at the intersection of this row and column.

Example 80 (Running Example: Booleans) For a simple example of the factor matrix, let us return to the regular algebra of Booleans, introduced in example 4 and continued in example 66. The event true has just itself as left factor; the event false has two left factors: false and true. The "factor matrix" of true thus has dimension $\{\mathsf{true}\}\times\{\mathsf{true}\}$ and just one "entry": $\mathsf{true}\Rightarrow\mathsf{true}$, which evaluates to true. The "factor matrix" of false has dimension $\mathsf{Bool}\times\mathsf{Bool}$. Displayed in the conventional way, it is the matrix

$$\begin{bmatrix} \mathsf{false}\Rightarrow\mathsf{false} & \mathsf{false}\Rightarrow\mathsf{true} \\ \mathsf{true}\Rightarrow\mathsf{false} & \mathsf{true}\Rightarrow\mathsf{true} \end{bmatrix}$$

which, of course, evaluates to

$$\begin{bmatrix} \mathsf{true} & \mathsf{true} \\ \mathsf{false} & \mathsf{true} \end{bmatrix}$$

The left factors l and r in Conway's theorem are, in this case, true and false, respectively. We invite the reader to verify all the other properties listed above for this simple example.
□

Example 81 (Running Example: Modulo Addition) Let m be a strictly positive natural number and consider the powerset regular algebra with underlying monoid $\mathbb{Z}_m$, the Abelian group of numbers modulo m under addition. (See examples 16 and 67.) As we have seen, the set $\neg\{0\}$ has 2^m factors. Because of the symmetry of addition, the sets of factors, left factors and right factors are all identical. (In other words, every factor is both a left factor and a right factor.) The left factor l in Conway's theorem is $\{0\}$ and $\neg\{0\}$ is the left factor r. Trivially (since 0 is the unit of the group), for all factors i, $i=\frac{i}{\{0\}}$.
□

Example 82 (Running Example: The Language $(aa)^*$) Let us return to the language $(aa)^*$ over the alphabet $\{a\}$. We saw in example 17 that this language has four right factors, namely $\emptyset$, $(aa)^*$, $a(aa)^*$ and a^*. The corresponding left factors are

$(aa)^*/\emptyset$, $(aa)^*/(aa)^*$, $(aa)^*/a(aa)^*$ and $(aa)^*/a^*$. These simplify to a^*, $(aa)^*$, $a(aa)^*$ and $\emptyset$, respectively. (In general, anti-derivatives would be used to calculate the left factors but in this case the left-right symmetry of all words in T^* makes the calculations much easier.)

The factor matrix of $(aa)^*$ is shown in three ways below. First, as the "under" function indexed by its left factors:

$$\begin{bmatrix} \emptyset\backslash\emptyset & \emptyset\backslash(aa)^* & \emptyset\backslash a(aa)^* & \emptyset\backslash a^* \\ (aa)^*\backslash\emptyset & (aa)^*\backslash(aa)^* & (aa)^*\backslash a(aa)^* & (aa)^*\backslash a^* \\ a(aa)^*\backslash\emptyset & a(aa)^*\backslash(aa)^* & a(aa)^*\backslash a(aa)^* & a(aa)^*\backslash a^* \\ a^*\backslash\emptyset & a^*\backslash(aa)^* & a^*\backslash a(aa)^* & a^*\backslash a^* \end{bmatrix}$$

Second, as the "over" function indexed by right factors:

$$\begin{bmatrix} a^*/a^* & a^*/(aa)^* & a^*/a(aa)^* & a^*/\emptyset \\ (aa)^*/a^* & (aa)^*/(aa)^* & (aa)^*/a(aa)^* & (aa)^*/\emptyset \\ a(aa)^*/a^* & a(aa)^*/(aa)^* & a(aa)^*/a(aa)^* & a(aa)^*/\emptyset \\ \emptyset/a^* & \emptyset/(aa)^* & \emptyset/a(aa)^* & \emptyset/\emptyset \end{bmatrix}$$

Finally, after elimination of the factor operators:

$$\begin{bmatrix} a^* & a^* & a^* & a^* \\ \emptyset & (aa)^* & a(aa)^* & a^* \\ \emptyset & a(aa)^* & (aa)^* & a^* \\ \emptyset & \emptyset & \emptyset & a^* \end{bmatrix}$$

The left factors l and r in equation (72) are both $(aa)^*$ so that the entry $l\backslash r$ is $(aa)^*\backslash(aa)^*$.
□

We have criticised Conway for an over-reliance on words. Example 82 is included partly to provide evidence for this criticism. Conway assumes that left and right factors are indexed by natural numbers in a way that reflects the one-to-one correspondence. He uses L_i, R_i and E_{ij} to denote the ith left factor, corresponding ith right factor and (i,j)th element in the factor matrix (where i and j are natural numbers). Then he states one of the main theorems on the factor matrix as follows [Con71, p.48]:

> Each E_{ij} is a factor and each factor is one of the E_{ij}. There exist unique indices l, r such that $E = L_r = R_l = E_{lr}$ and $L_i = E_{li}$ and $R_i = E_{ir}$ for each i. Hence the factors naturally form a square matrix among the entries of which is E.

The issue here is with the meaning of the words "one" and "unique". Note that in example 82, the central 2×2 matrix contains *two* occurrences of $(aa)^*$ and *two* occurrences of $a\,(aa)^*$, the central two rows are identical but for an interchange of these two languages, and the same is true of the central two columns. These two rows and columns contain all the left factors and right factors of the language exactly once, and no other factors. Now Conway's "indices" are numbers, not left or right factors. That means that there appear to be *two* choices for the "unique" indices l, r! Even worse, the factor matrix of $(a^m)^*$, for arbitrary, strictly positive, natural number m, offers a choice of m "unique" (numerical) indices l and r.

The earlier use of the word "one" is an ambiguous use of English: it could mean that each factor occurs exactly once in the matrix, but it could also mean that each factor occurs at least once in the matrix. Conway compounds the confusion by stating on page 49:

> The theorem does prevent E from occurring twice in its factor matrix.

There is a missing "not" in this sentence: it should read "The theorem does *not* prevent E from occurring twice in its factor matrix."!

In my view, Conway's statement of the theorem is extremely confusing and easily open to misinterpretation. The "uniqueness" claimed by Conway depends on the particular indexing chosen for the left and right factors: our presentation makes it clear that the "unique" entry is the entry $(aa)^*\setminus(aa)^*$, and not the entry $a\,(aa)^*\backslash a\,(aa)^*$ even though the two entries are equal. By eliminating a spurious, irrelevant and entirely arbitrary numerical indexing function, the exposition is made clearer and more precise.

Before leaving this section, let us briefly mention that there is a connection between example 81 and example 82. The connection is that the syntactic monoid of the language $(aa)^*$ is $\mathbb{Z}_2$. Note, however, that the syntactic monoid of $(a^m)^*$ is not $\mathbb{Z}_m$ when m is greater than 2.

4 The Factor Matrix of a Factor

This section is the first where we present results in the author's PhD thesis [Bac75] that have not previously been published elsewhere.

Suppose F is a factor of E. Equivalently, suppose F is an entry in the factor matrix of E. We show that the factor matrix of F is a "submatrix" of the factor matrix of E. Recall, however, that we have defined a "matrix" to be a binary function. In particular, the factor matrix of E is a function from pairs of left factors of E to factors of E, and the factor matrix of F is a function from pairs of left factors of F to factors of F. Thus, the domain of the factor matrix of F is quite different from the domain of the factor matrix of E if $F{\neq}E$. We must, therefore, say precisely what the meaning of "submatrix" is.

We prefer to say that the factor matrix of F is "represented by a submatrix" of the factor matrix of E in the following sense.

Definition 83 Suppose $\mathbf{G}'$ is a function of type $A{\leftarrow}B{\times}B$ and $\mathbf{G}$ is a function of type $A{\leftarrow}C{\times}C$. We say that $\mathbf{G}'$ is *represented by a submatrix of* $\mathbf{G}$ if there is a set C' such that $C{\supseteq}C'$ and a bijection h of type $C'{\leftarrow}B$ such that $\mathbf{G}' = \mathbf{G} \bullet h{\times}h$.

(Here and elsewhere $f{\times}g$ denotes the function from pairs to pairs defined by, for all x and y of appropriate type, $(f{\times}g).(x,y) = (f.x\,,\,g.y)$.)
□

We apply definition 83 exclusively in the special case that f and g are square event matrices —indeed, factor matrices— but prefer a more general formulation of the definition.

Informally, the subset C' of C identifies a "submatrix" of $\mathbf{G}$ (specifically, the entries indexed by elements of C') and the bijection h translates indices of the matrix $\mathbf{G}'$ into indices of $\mathbf{G}$. For calculational purposes (see, for example, the proof of lemma), it is useful to express definition 83 in terms of the selectors introduced in definition 27.

Lemma 84 Suppose $\mathbf{G}'$ is a function of type $A{\leftarrow}B{\times}B$ and $\mathbf{G}$ is a function of type $A{\leftarrow}C{\times}C$, where A is the event set of a regular algebra $\mathcal{R}$. Then $\mathbf{G}'$ is *represented by a submatrix of* $\mathbf{G}$ if there is an injective function k of type $C{\leftarrow}B$ such that

$$\mathbf{G}' \;=\; (Sel.k)^{\cup}{\otimes}\mathbf{G}{\otimes}Sel.k \;\;.$$

Proof Straightforward expansion of the definitions. Given injective function k of type $C{\leftarrow}B$, we let C' be the image set of k; conversely, given bijection h of type $C'{\leftarrow}B$, we let k equal $\iota_{C,C'}{\circ}h$, where $\iota_{C,C'}$ is the function of type $C{\leftarrow}C'$ that injects elements of C' into C (i.e. $\iota_{C,C'}.x{=}x$ for all x). Then, for all y and z of type B,

$$
\begin{array}{ll}
 & y\ ((Sel.k)^{\cup}{\otimes}\mathbf{G}{\otimes}Sel.k)\ z \\
= & \{\quad \text{definition of } k\text{, distributivity properties of } Sel \quad\} \\
 & y\ ((Sel.h)^{\cup}{\otimes}(Sel.\iota_{C,C'})^{\cup}{\otimes}\mathbf{G}{\otimes}Sel.\iota_{C,C'}{\otimes}Sel.h)\ z \\
= & \{\quad \text{definition of matrix multiplication, } Sel \text{ and converse} \quad\} \\
 & \langle\Sigma\, u,v,w,x :: (y\ (Sel.h)^{\cup}\ u)\cdot(u\ (Sel.\iota_{C,C'})^{\cup}\ v)\cdot(v\ \mathbf{G}\ w)\cdot(w\ Sel.\iota_{C,C'}\ x)\cdot(x\ Sel.h\ z)\rangle \\
= & \{\quad \text{definition of } Sel\text{, converse and } \iota_{C,C'} \\
 & \qquad 1 \text{ is the unit and } 0 \text{ is the zero of multiplication} \quad\}
\end{array}
$$

$$
\begin{array}{ll}
 & \langle \Sigma u,v,w,x \;:\; u{=}v \wedge w{=}x \;:\; (y\ (Sel.h)^{\cup}\ u)\cdot(v\ \mathbf{G}\ w)\cdot(x\ Sel.h\ z)\rangle \\
= & \quad\{\quad \text{one-point rule} \quad\} \\
 & \langle \Sigma u,x :: (y\ (Sel.h)^{\cup}\ u)\cdot(u\ \mathbf{G}\ x)\cdot(x\ Sel.h\ z)\rangle \\
= & \quad\{\quad \text{definition of } Sel \text{ and converse,} \\
 & \qquad\quad 1 \text{ is the unit and } 0 \text{ is the zero of multiplication} \quad\} \\
 & \langle \Sigma u,x \;:\; u{=}h.y \wedge x{=}h.z \;:\; u\ \mathbf{G}\ x\rangle \\
= & \quad\{\quad \text{one-point rule} \quad\} \\
 & h.y\ \mathbf{G}\ h.z \\
= & \quad\{\quad \text{assumption: } \mathbf{G}' = \mathbf{G} \bullet h{\times}h \quad\} \\
 & y\ \mathbf{G}'\ z\ \ .
\end{array}
$$

The lemma follows by extensionality (equality of functions).
□

Definition 83 sets the scene for this section. We assume that F is a factor of E. Then, supposing that $\mathcal{L}.E$ denotes the set of left factors of E and $\mathcal{L}.F$ the set of left factors of F, we calculate a subset M of $\mathcal{L}.E$ and a bijection β of type $M \leftarrow \mathcal{L}.F$ such that $|\overline{F}| = (Sel.\beta)^{\cup} \otimes |\overline{E}| \otimes Sel.\beta$. The calculation is complicated by the fact that factors may occur repeatedly in a factor matrix. This gives us an additional task. Given F we must first locate an occurrence of F in the factor matrix of E with particular properties. This task is accomplished in subsection 4.1; subsection 4.2 then calculates the subset M and the bijection β, leading to the representation theorem, theorem 98.

Example 85 (Running Example: Booleans) As always, we return to our running example. In example 80 it was observed that the factor matrix of true has exactly one entry and the factor matrix of false has four entries, of which three are true. Given one of the true entries, we need to identify a particular submatrix (with just one entry) that is the factor matrix of true.
□

4.1 Identifying A Suitable Matrix Entry

Throughout this subsection, we suppose that i and j are left factors of E such that $F{=}i\backslash j$. (In words, $i\backslash j$ is one of the possibly multiple occurrences of F in the factor matrix of E.) Let $s{=}j/(i\backslash j)$ and $t{=}((s\triangleright / j\triangleright)\backslash s\triangleright)\triangleleft$. We prove that $F{=}s\backslash t$. Importantly, we prove that s and t have a number of vital properties.

Lemma 86 Suppose i and j are left factors of E. Let $s=j/(i\backslash j)$ and $t=((s\rhd/j\rhd)\backslash s\rhd)\lhd$. Then s and t are left factors of E and

$$i\backslash j=s\backslash t \;\wedge\; s=t/(s\backslash t) \;\wedge\; t\rhd=(s\rhd/t\rhd)\backslash s\rhd \;.$$

Proof It is clear that t is a left factor of E since it is in the range of $\lhd$. It is also the case that s is a left factor of E because it is a left factor of the left factor j and left factors of left factors are left factors (theorem 50).

We prove the following facts in order:

(a) $i\backslash j=s\backslash j$

(b) $t\rhd=(s\rhd/j\rhd)\backslash s\rhd$

(c) $s\backslash j=s\backslash t$

(d) $t\rhd=(s\rhd/t\rhd)\backslash s\rhd$

(e) $s=t/(s\backslash t)$

The first conjunct of the lemma clearly follows from (a) and (c) by transitivity of equality. The second conjunct is (e) and the third conjunct is (d).
(a) By mutual inclusion: first,

$$\begin{array}{ll}
 & i\backslash j\preceq s\backslash j \\
= & \quad\{\quad \text{factors} \quad\} \\
 & s\cdot i\backslash j\preceq j \\
= & \quad\{\quad \text{factors} \quad\} \\
 & s\preceq j/(i\backslash j) \\
= & \quad\{\quad \text{definition of } s\text{, reflexivity of } \preceq \quad\} \\
 & \text{true} \;.
\end{array}$$

Second,

$$\begin{array}{ll}
 & s\backslash j\preceq i\backslash j \\
\Leftarrow & \quad\{\quad \text{anti-monotonicity of } \backslash \quad\} \\
 & s\succeq i \\
= & \quad\{\quad \text{definition of } s\text{, lemma 46 with } X,Y:=i,j \quad\} \\
 & \text{true} \;.
\end{array}$$

(b)

$t\triangleright$

$=$ $\{$ definition of t $\}$

$((s\triangleright / j\triangleright) \backslash s\triangleright)\triangleleft\triangleright$

$=$ $\{$ (63) $\}$

$(s \cdot s\triangleright / j\triangleright)\triangleright\triangleleft\triangleright$

$=$ $\{$ unity of opposites $\}$

$(s \cdot s\triangleright / j\triangleright)\triangleright$

$=$ $\{$ (63) $\}$

$(s\triangleright / j\triangleright) \backslash s\triangleright$.

(c) The proof is entirely symmetric to the proof of (a) with left factors being replaced by right factors. For completeness, we give it anyway. First we translate the proof obligation from left factors to right factors.

$s\backslash j = s\backslash t$

$=$ $\{$ j , s and t are all left factors; (61) $\}$

$s\triangleright / j\triangleright = s\triangleright / t\triangleright$.

Now we proceed by mutual inclusion. First,

$s\triangleright / j\triangleright \preceq s\triangleright / t\triangleright$

$=$ $\{$ factors: (3) $\}$

$s\triangleright / j\triangleright \cdot t\triangleright \preceq s\triangleright$

$=$ $\{$ factors: (3) $\}$

$t\triangleright \preceq (s\triangleright / j\triangleright) \backslash s\triangleright$

$=$ $\{$ (b) $\}$

true .

Second,

$s\triangleright / t\triangleright \preceq s\triangleright / j\triangleright$

$\Leftarrow$ $\{$ anti-monotonicity of $/$ $\}$

$t\triangleright \succeq j\triangleright$

$=$ { (b) and lemma 46 with $X,Y := s\triangleright , j\triangleright$ }

true .

(d)

$t\triangleright$

$=$ { (b) }

$(s\triangleright / j\triangleright) \backslash s\triangleright$

$=$ { s and j are left factors of E, (61) }

$(s\backslash j) \backslash s\triangleright$

$=$ { (c) }

$(s\backslash t) \backslash s\triangleright$

$=$ { s and t are left factors of E, (61) }

$(s\triangleright / t\triangleright) \backslash s\triangleright$.

(e) By mutual inclusion. First,

$s \preceq t/(s\backslash t)$

$=$ { lemma 46 with $X,Y := s,t$ }

true .

Second,

$t/(s\backslash t) \preceq s$

$=$ { definition of s and (a) and (c) }

$t/(s\backslash t) \preceq j/(s\backslash t)$

$\Leftarrow$ { monotonicity of $/(s\backslash t)$ }

$t \preceq j$

$=$ { (58), t and j are left factors of E }

$t\triangleright \succeq j\triangleright$

$=$ { (b) and lemma 46 with $X,Y := s\triangleright , j\triangleright$ }

true .

□

Example 87 (Running Example: Booleans) We return once more to our running example (examples 4, 66, 80).

First, suppose $E = \mathsf{false}$, $F = \mathsf{true}$, $i = \mathsf{false}$ and $j = \mathsf{true}$. That is we begin with the entry $\mathsf{false}{\Rightarrow}\mathsf{true}$ in the factor matrix of false and we wish to determine a submatrix that is the factor matrix of this entry. Then lemma 86 identifies s as $\mathsf{true}{\Leftarrow}(\mathsf{false}{\Rightarrow}\mathsf{true})$, which evaluates to true, and t as $((s{\rhd} \Leftarrow j{\rhd}) \Rightarrow s{\rhd}){\lhd}$. For this instance of E, $\mathsf{false}{\rhd} = \mathsf{false}{\lhd} = \mathsf{true}$ and $\mathsf{true}{\rhd} = \mathsf{true}{\lhd} = \mathsf{false}$. Thus t has the value false and the submatrix of the factor matrix of false that is the factor matrix of true is the single entry $\mathsf{false}{\Rightarrow}\mathsf{false}$.

Now, suppose $E = \mathsf{false}$, $F = \mathsf{true}$, $i = \mathsf{true}$ and $j = \mathsf{true}$. That is, E and F are unchanged but we begin with the entry $\mathsf{true}{\Rightarrow}\mathsf{true}$ in the factor matrix of false. Then lemma 86 identifies both s and t as true. (Details left to the reader.) Thus the submatrix of the factor matrix of false that is identified as the factor matrix of true is the single entry $\mathsf{true}{\Rightarrow}\mathsf{true}$.

In general, factors may appear repeatedly in a factor matrix and the submatrix that is identified as the factor matrix of a factor will depend on the entry in the matrix with which the calculation begins.
□

Example 88 (Running Example: Modulo Addition) Let m be a strictly positive natural number and consider the powerset regular algebra with underlying monoid $\mathbb{Z}_m$, the Abelian group of numbers modulo m under addition. (See examples 16, 67 and 81.) The factor matrix of $\neg\{0\}$ has $2^m{\times}2^m$ entries (since $\neg\{0\}$ has 2^m left factors). Factors and factor matrices of factors appear repeatedly in the factor matrix of $\neg\{0\}$.

Clearly, as m increases the size of the factor matrix quickly becomes very large. But we can illustrate the general structure by taking the case when m is 3.

As explained in example 67, all subsets of $\{0,1,2\}$ are left factors, right factors and factors of $\neg\{0\}$ (in the powerset regular algebra with underlying monoid $\mathbb{Z}_3$). The factor matrix of $\neg\{0\}$ before simplification is thus displayed below in the conventional fashion as a two-dimensional array of values. (Note the pattern in the entries in each row and each column.)

$$\begin{bmatrix}
\frac{\neg\emptyset}{\neg\emptyset} & \frac{\neg\{0\}}{\neg\emptyset} & \frac{\neg\{1\}}{\neg\emptyset} & \frac{\neg\{2\}}{\neg\emptyset} & \frac{\{0\}}{\neg\emptyset} & \frac{\{1\}}{\neg\emptyset} & \frac{\{2\}}{\neg\emptyset} & \frac{\emptyset}{\neg\emptyset} \\
\frac{\neg\emptyset}{\neg\{0\}} & \frac{\neg\{0\}}{\neg\{0\}} & \frac{\neg\{1\}}{\neg\{0\}} & \frac{\neg\{2\}}{\neg\{0\}} & \frac{\{0\}}{\neg\{0\}} & \frac{\{1\}}{\neg\{0\}} & \frac{\{2\}}{\neg\{0\}} & \frac{\emptyset}{\neg\{0\}} \\
\frac{\neg\emptyset}{\neg\{1\}} & \frac{\neg\{0\}}{\neg\{1\}} & \frac{\neg\{1\}}{\neg\{1\}} & \frac{\neg\{2\}}{\neg\{1\}} & \frac{\{0\}}{\neg\{1\}} & \frac{\{1\}}{\neg\{1\}} & \frac{\{2\}}{\neg\{1\}} & \frac{\emptyset}{\neg\{1\}} \\
\frac{\neg\emptyset}{\neg\{2\}} & \frac{\neg\{0\}}{\neg\{2\}} & \frac{\neg\{1\}}{\neg\{2\}} & \frac{\neg\{2\}}{\neg\{2\}} & \frac{\{0\}}{\neg\{2\}} & \frac{\{1\}}{\neg\{2\}} & \frac{\{2\}}{\neg\{2\}} & \frac{\emptyset}{\neg\{2\}} \\
\frac{\neg\emptyset}{\{0\}} & \frac{\neg\{0\}}{\{0\}} & \frac{\neg\{1\}}{\{0\}} & \frac{\neg\{2\}}{\{0\}} & \frac{\{0\}}{\{0\}} & \frac{\{1\}}{\{0\}} & \frac{\{2\}}{\{0\}} & \frac{\emptyset}{\{0\}} \\
\frac{\neg\emptyset}{\{1\}} & \frac{\neg\{0\}}{\{1\}} & \frac{\neg\{1\}}{\{1\}} & \frac{\neg\{2\}}{\{1\}} & \frac{\{0\}}{\{1\}} & \frac{\{1\}}{\{1\}} & \frac{\{2\}}{\{1\}} & \frac{\emptyset}{\{1\}} \\
\frac{\neg\emptyset}{\{2\}} & \frac{\neg\{0\}}{\{2\}} & \frac{\neg\{1\}}{\{2\}} & \frac{\neg\{2\}}{\{2\}} & \frac{\{0\}}{\{2\}} & \frac{\{1\}}{\{2\}} & \frac{\{2\}}{\{2\}} & \frac{\emptyset}{\{2\}} \\
\frac{\neg\emptyset}{\emptyset} & \frac{\neg\{0\}}{\emptyset} & \frac{\neg\{1\}}{\emptyset} & \frac{\neg\{2\}}{\emptyset} & \frac{\{0\}}{\emptyset} & \frac{\{1\}}{\emptyset} & \frac{\{2\}}{\emptyset} & \frac{\emptyset}{\emptyset}
\end{bmatrix}$$

After elimination of all operators, we get the following matrix:

$$\begin{bmatrix}
\{0,1,2\} & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
\{0,1,2\} & \{0\} & \{1\} & \{2\} & \emptyset & \emptyset & \emptyset & \emptyset \\
\{0,1,2\} & \{2\} & \{0\} & \{1\} & \emptyset & \emptyset & \emptyset & \emptyset \\
\{0,1,2\} & \{1\} & \{2\} & \{0\} & \emptyset & \emptyset & \emptyset & \emptyset \\
\{0,1,2\} & \{1,2\} & \{0,2\} & \{0,1\} & \{0\} & \{1\} & \{2\} & \emptyset \\
\{0,1,2\} & \{0,1\} & \{1,2\} & \{0,2\} & \{2\} & \{0\} & \{1\} & \emptyset \\
\{0,1,2\} & \{0,2\} & \{0,1\} & \{1,2\} & \{1\} & \{2\} & \{0\} & \emptyset \\
\{0,1,2\} & \{0,1,2\} & \{0,1,2\} & \{0,1,2\} & \{0,1,2\} & \{0,1,2\} & \{0,1,2\} & \{0,1,2\}
\end{bmatrix}$$

The empty set, $\emptyset$, and its complement, $\neg\emptyset$, (i.e. $\{0..m-1\}$ in the general case and $\{0,1,2\}$ in this specific case) are entirely analogous to false and true in example 87: the factor matrix of $\emptyset$ has 2×2 entries, the entry $\neg\emptyset$ occurs three times and the fourth entry is $\emptyset$ itself. The factor matrix of $\neg\emptyset$ has exactly one entry (which is $\neg\emptyset$ itself).

In the factor matrix of other subsets of $\neg\emptyset$, there is always a row and a column indexed by $\emptyset$ and a row and a column indexed by $\neg\emptyset$ (because both $\emptyset$ and $\neg\emptyset$ are (left and right) factors of any proper subset of $\neg\emptyset$). In the row indexed by $\emptyset$ all entries are $\neg\emptyset$ (i.e. $\frac{i}{\emptyset}=\neg\emptyset$ for all left factors i, and in the column indexed by $\emptyset$ all entries are $\emptyset$ except for the row indexed by $\emptyset$ (i.e. $\frac{\emptyset}{i}=\emptyset$ for all non-empty left factors i); in the column indexed by $\neg\emptyset$ all entries are $\neg\emptyset$ (i.e. $\frac{\neg\emptyset}{i}=\neg\emptyset$ for all left factors i) and in the row indexed by $\neg\emptyset$ all entries are $\emptyset$ except for the column indexed by $\neg\emptyset$ (i.e. $\frac{i}{\neg\emptyset}=\emptyset$

if left factor i is a proper subset of $\neg\emptyset$). This is apparent in the above matrices: see the border formed by first and last rows and first and last columns.

In general, in any regular algebra, the maximal element of the lattice is a left (and right) factor of all events, and the least element of the lattice is a left (and right) factor of all events except for the maximal element of the lattice. When displaying a factor matrix as a two-dimensional array,. the two corresponding rows and columns are not of interest and so we usually omit them. (We see later that, for languages, these correspond to so-called "inadmissible" nodes in the factor graph of the language. That is, they correspond to nodes that can be ignored when using the factor graph as a recogniser of the language.)

Ignoring the border of the above array and looking at only the central $6{\times}6$ array, we see that there are four subarrays, each of size $3{\times}3$ and in each of which the entries are sets of the same size. Two subarrays are identical: the subarrays with entries that are singleton sets. Together with the border formed by entries indexed by $\emptyset$ and $\neg\emptyset$, these represent the factor matrix of $\{0\}$. That is, the factor matrix of $\{0\}$ displayed as a two-dimensional array is as shown below.

$$\begin{bmatrix} \{0,1,2\} & \emptyset & \emptyset & \emptyset & \emptyset \\ \{0,1,2\} & \{0\} & \{1\} & \{2\} & \emptyset \\ \{0,1,2\} & \{1\} & \{0\} & \{2\} & \emptyset \\ \{0,1,2\} & \{2\} & \{1\} & \{0\} & \emptyset \\ \{0,1,2\} & \{0,1,2\} & \{0,1,2\} & \{0,1,2\} & \{0,1,2\} \end{bmatrix}$$

Formally, there are two distinct representations of the factor matrix of $\{0\}$ as a submatrix of the factor matrix of $\neg\{0\}$ (in this particular case).
□

This subsection was begun by an implicit existential quantification over the dummies i and j. At this point, we close the scope of the quantification. Subsequent sections re-use dummies i and j for other purposes.

4.2 Constructing the Representation

Now that we have identified a particular entry $s\backslash t$ in the factor matrix (see lemma 86), we identify the submatrix of the factor matrix of E that represents the factor matrix of $s\backslash t$. The subset M of the left factors of E defined below is what we need.

Definition 89 Define the subset M of the left factors of E by

$$k{\in}M \;\equiv\; k = t/(k\backslash t) \;\wedge\; k\backslash t = (s\backslash k)\backslash(s\backslash t) \;\; .$$

□

Example 90 (Running Example: Booleans) Suppose, as in the first part of example 87, that $E = \mathsf{false}$, $F = \mathsf{true}$, $i = \mathsf{true}$ and $j = \mathsf{true}$. We recall that lemma 86 identifies both s and t as true. Then definition 89 defines the subset M of the left factors of E by

$$k{\in}M \;\equiv\; (k = (\mathsf{true}{\Leftarrow}(k{\Rightarrow}\mathsf{true}))) \wedge ((k{\Rightarrow}\mathsf{true}) = ((\mathsf{true}{\Rightarrow}k){\Rightarrow}(\mathsf{true}{\Rightarrow}\mathsf{true}))) \;\; .$$

The second equation is a tautology, and the first equation simplifies to $k = \mathsf{true}$. That is, $M = \{\mathsf{true}\}$ as expected.

The reader may wish to check that when s and t are both false, M is calculated to be $\{\mathsf{false}\}$. (See example 87.)
□

The next step is to show that the submatrix defined by the set M represents the factor matrix of F. This is expressed formally in theorem 98. Informally, our goal is to show that the submatrix of events $k \backslash m$, where k and m are both elements of M, is the factor matrix of $s \backslash t$. First we establish that $s \backslash t$ is itself an entry in this submatrix.

Lemma 91

$$s{\in}M \wedge t{\in}M \;\; .$$

Proof We have:

$$
\begin{array}{ll}
 & s = t/(s\backslash t) \\
= & \{ \quad \text{lemma 86(e)} \quad \} \\
 & \mathsf{true} \;\; .
\end{array}
$$

Also,

$$
\begin{array}{ll}
 & (s\backslash s)\backslash(s\backslash t) \\
= & \{ \quad s = t/(s\backslash t) \quad \} \\
 & (s\backslash t/(s\backslash t))\backslash(s\backslash t) \\
= & \{ \quad \text{lemma 48 with } E{:=}s\backslash t \quad \} \\
 & s\backslash t \;\; .
\end{array}
$$

(Note the implicit use of the associativity of the $\backslash$ and $/$ operators.) This establishes that $s{\in}M$. Now, for $t{\in}M$ we have:

$$
\begin{array}{ll}
 & t = t/(t\backslash t) \\
= & \{ \quad \text{lemma 48 with } X{:=}t \quad \} \\
 & \mathsf{true} \;\; ,
\end{array}
$$

and

$$
\begin{array}{ll}
 & (s\backslash t)\backslash(s\backslash t) \\
= & \{ \quad (61),\ s \text{ and } t \text{ are left factors of } E \quad \} \\
 & (s\triangleright / t\triangleright) \backslash s\triangleright / t\triangleright \\
= & \{ \quad \text{by lemma 86(d): } t\triangleright \ = \ (s\triangleright / t\triangleright)\backslash s\triangleright \quad \} \\
 & t\triangleright / t\triangleright \\
= & \{ \quad (61),\ s \text{ and } t \text{ are left factors of } E \quad \} \\
 & t\backslash t \ .
\end{array}
$$

□

Now we observe that the entries $s\backslash k$ in the submatrix are left factors of $s\backslash t$ and the entries $k\backslash t$ are the corresponding right factors.

Lemma 92

$$s\backslash k = (s\backslash t)/(k\backslash t) \ \wedge \ k\backslash t = (s\backslash k)\backslash(s\backslash t) \quad \Leftarrow \quad k\in M \ .$$

In words, $s\backslash k$ and $k\backslash t$ are corresponding left and right factors of $s\backslash t$ if $k\in M$.

Proof Immediate from the definition of M and $s\backslash(t/(k\backslash t)) = (s\backslash t)/(k\backslash t)$.

□

Next we show that there is no duplication in the entries $s\backslash k$, where k ranges over elements of M.

Lemma 93

$$k = k' \quad \Leftarrow \quad k\in M \wedge k'\in M \wedge s\backslash k = s\backslash k' \ .$$

(In words, the entries in the row indexed by s in the submatrix of the factor matrix of E defined by M are unique.)

Proof Assume $k\in M \wedge k'\in M$. Then

$$
\begin{array}{ll}
 & k = k' \\
\Leftarrow & \{ \quad \text{definition of } M,\ k\in M \wedge k'\in M \quad \} \\
 & t/(k\backslash t) = t/(k'\backslash t) \\
\Leftarrow & \{ \quad \text{Leibniz} \quad \} \\
 & k\backslash t = k'\backslash t
\end{array}
$$

$$
\begin{array}{ll}
= & \{ \quad \text{definition of } M,\ k{\in}M \wedge k'{\in}M \quad \} \\
& (s\backslash k)\backslash(s\backslash t) = (s\backslash k')\backslash(s\backslash t) \\
\Leftarrow & \{ \quad \text{Leibniz} \quad \} \\
& s\backslash k = s\backslash k' \ .
\end{array}
$$

□

The next step is to establish the converse of lemma 92. That is, we show that every left factor of $s\backslash t$ is equal to $s\backslash k$ for some k in M. We exploit the unity of opposites to construct k.

Lemma 94 Suppose L is a left factor of $s\backslash t$ with corresponding right factor R. That is, suppose

$$R = L\backslash(s\backslash t) \quad \wedge \quad L = (s\backslash t)/R \ .$$

Let $k = t/R$. Then

$$L = s\backslash k \quad \wedge \quad R = k\backslash t \quad \wedge \quad k{\in}M \ .$$

In particular,

$$R \ = \ (t/R)\backslash t \ .$$

Proof We establish the three conjuncts in order. The first conjunct is trivial:

$$
\begin{array}{ll}
& s\backslash k \\
= & \{ \quad \text{definition of } k \quad \} \\
& s\backslash(t/R) \\
= & \{ \quad \text{associativity} \quad \} \\
& (s\backslash t)/R \\
= & \{ \quad \text{definition of } L \quad \} \\
& L \ .
\end{array}
$$

The second conjunct is proved by mutual inclusion. First,

$$
\begin{array}{ll}
& k\backslash t \succeq R \\
= & \{ \quad k = t/R,\ \text{lemma 46 with } X,Y := R,t \quad \} \\
& \text{true} \ .
\end{array}
$$

To prove the converse inclusion, we exploit the first conjunct (i.e. $L = s\backslash k$).

$k\backslash t \preceq R$

$=\qquad \{\quad R = L\backslash(s\backslash t) = (s\backslash k)\backslash(s\backslash t)$, factors $\}$

$s \cdot s\backslash k \cdot k\backslash t \preceq t$

$=\qquad \{\quad$ cancellation $\}$

true .

The third conjunct now follows:

$k{\in}M$

$=\qquad \{\quad$ definition of M $\}$

$k = t/(k\backslash t) \;\wedge\; k\backslash t = (s\backslash k)\backslash(s\backslash t)$

$=\qquad \{\quad k = t/R ,\quad R = k\backslash t ,\quad L = s\backslash k \quad\}$

$t/R = t/R \;\wedge\; R = L\backslash(s\backslash t)$

$=\qquad \{\quad$ assumption $\}$

true .

The final property is an obvious expansion of the equations for R and k.
□

We have now calculated a bijection between the left factors of F and the elements of M:

Theorem 95 Suppose F is a factor of E. Suppose, in particular, that $F = i\backslash j$ where i and j are left factors of E. Let $s = j/(i\backslash j)$ and $t = ((s\rhd / j\rhd) \backslash s\rhd)\lhd$. Let M be the subset of left factors of E defined in definition 89. Then there is a bijection β from the set of left factors of F to M such that, for all left factors L of F,

$$L = s\backslash\beta.L \quad .$$

Specifically,

(96) $\quad \beta.L = t/(L\backslash F) \quad .$

Proof This is a combination of lemmas 93 and 94. Lemma 94 states that if L is a left factor of F then there is a k in M such that $L = s\backslash k$ and lemma 93 asserts that such a k is unique.
□

The final step is to show that an event is a factor of $s\backslash t$ equivales it equals $k\backslash m$ for some k and some m. Here we exploit Conway's theorem that every factor of an event

Z has the form $X\backslash Y$ where X and Y are left factors of Z. Since we have established that the left factors of $s\backslash t$ are events of the form $s\backslash k$ for some k in M, this amounts to the following lemma.

Lemma 97 Suppose $k{\in}M$ and $m{\in}M$. Then

$$k\backslash m \;=\; (s\backslash k)\backslash(s\backslash m) \quad .$$

Proof

$$\begin{array}{lll}
 & (s\backslash k)\backslash(s\backslash m) & \\
= & \{ & k{\in}M\,.\ \text{So}\ k = t/(k\backslash t)\quad ; \\
 & & m{\in}M\,.\ \text{So}\ m = t/(m\backslash t)\quad \} \\
 & (s\backslash t/(k\backslash t))\backslash(s\backslash t/(m\backslash t)) & \\
= & \{ & \text{by lemma 92,}\ s\backslash t/(k\backslash t)\ \text{and}\ k\backslash t\ \text{are} \\
 & & \text{corresponding left and right factors of}\ s\backslash t; \\
 & & \text{similarly for}\ s\backslash t/(m\backslash t)\ \text{and}\ m\backslash t \\
 & & X{\lhd}\backslash Y{\lhd} = (X{\lhd}){\rhd}/(Y{\lhd}){\rhd}\ \text{with}\ E{:=}s\backslash t\quad \} \\
 & k\backslash t/(m\backslash t) & \\
= & \{ & m{\in}M\,.\ \text{So}\ m = t/(m\backslash t)\quad \} \\
 & k\backslash m\quad . &
\end{array}$$

(Note that the rule $X\backslash(Y/Z) = (X\backslash Y)/Z$ has been used implicitly at each step.)
□

We summarise the foregoing lemmas in the following theorem.

Theorem 98 The factor matrix of a factor F of E is represented by a submatrix of the factor matrix of E.

Specifically, the factor matrix of F is constructed from the factor matrix of E as follows. First, identify a pair of left factors i and j of E such that $F = i\backslash j$. Then let $s = j/(i\backslash j)$ and $t = ((s{\rhd}/j{\rhd})\backslash s{\rhd}){\lhd}$. Note that, by lemma 86, $F = s\backslash t$. Define the subset M of the left factors of E as in definition 89. Then the matrix $k\backslash m$ where k and m are elements of M represents the factor matrix of F.

Moreover, letting the function β from left factors of F to left factors of E be defined by

$$\beta.i' = t/(i'\backslash F)$$

for all left factors i' of F. Then β is an injective function with domain $\mathcal{L}.F$ and image set M, and

(99) $\quad |\overline{F}| \; = \; (Sel.\beta)^{\cup} \otimes |\overline{E}| \otimes Sel.\beta \quad .$

Proof Let $\mathcal{L}.F$ denote the set of left factors of F. Then, by definition, the factor matrix of F, denoted $|\overline{F}|$ is a function with domain $\mathcal{L}.F \times \mathcal{L}.F$. The value of $|\overline{F}|$ at the pair (i',j'), where i' and j' are left factors of F, is $i'\backslash j'$. But

$$
\begin{array}{ll}
& i'\backslash j' \\
= & \{ \quad \text{theorem 95} \quad \} \\
& (s\backslash\beta.i')\backslash(s\backslash\beta.j') \\
= & \{ \quad \text{by theorem 95, } \beta.i'\in M \text{ and } \beta.j'\in M\text{; lemma 97} \quad \} \\
& \beta.i'\backslash\beta.j' \quad .
\end{array}
$$

Equation (99) follows from the definition of Sel and matrix multiplication. From the type of β, we conclude that the factor matrix of F is represented by

$$\langle i,j : i\in M \wedge j\in M : i\backslash j\rangle$$

which, by definition 89, is a submatrix of $|\overline{E}|$.
□

4.3 A Surjective Mapping

In preparation for section 8, we establish rather more than theorem 98: we exhibit a *surjective* function from the left factors of E to the left factors of F. This function is used in section 8 to prove that the factor graph of a factor F of E has cycle rank at most the cycle rank of the factor graph of E.

We continue to assume that s and t are the left factors of E defined in lemma 86 such that $F = s\backslash t$.

Let i be a left factor of E Define the functions γ of type $\mathcal{L}.F \leftarrow \mathcal{L}.E$ and α of type $\mathcal{L}.E \leftarrow \mathcal{L}.E$ by

(100) $\quad \gamma.i \; = \; F/(i\backslash t) \quad .$

and

(101) $\quad \alpha.i \; = \; t/(\gamma.i\backslash F) \quad .$

Note that γ has type $\mathcal{L}.\mathsf{F}\leftarrow\mathcal{L}.\mathsf{E}$ —in particular, $\gamma.i$ is a left factor of F— and α has type $\mathcal{L}.\mathsf{E}\leftarrow\mathcal{L}.\mathsf{E}$ —in particular, $\alpha.i$ is a left factor of E— . Note also that

(102) $\alpha = \beta\circ\gamma$

where β is the function of type $\mathcal{L}.\mathsf{E}\leftarrow\mathcal{L}.\mathsf{F}$ and image set M identified in corollary 95. Because β is injective, it follows that

(103) $\gamma = \beta^{\cup}\circ\alpha$.

The relations γ and $\beta^{\cup}$ have the same type. However, they are clearly not equal since γ is total whereas $\beta^{\cup}$ is not. Nevertheless:

Lemma 104 The function γ inverts the function β. That is,

$$\gamma\circ\beta = \mathsf{id}_{\mathcal{L}.\mathsf{F}} .$$

Hence,

$$\gamma\circ\alpha = \gamma \;\wedge\; \alpha\circ\alpha = \alpha \;\wedge\; \alpha\circ\beta = \beta .$$

Proof Suppose L is a left factor of F. Then

$$
\begin{array}{ll}
 & (\gamma\circ\beta).\mathsf{L} \\
= & \{ \quad \text{definition of } \gamma\text{: (100) and } \beta\text{: (96)} \quad \} \\
 & \mathsf{F}/((\mathsf{t}/(\mathsf{L}\backslash\mathsf{F}))\backslash\mathsf{t}) \\
= & \{ \quad \text{lemma 94, with } \mathsf{R}:=\mathsf{L}\backslash\mathsf{F} \quad \} \\
 & \mathsf{F}/(\mathsf{L}\backslash\mathsf{F}) \\
= & \{ \quad \mathsf{L} \text{ is a left factor of } \mathsf{F}, \\
 & \quad\;\; \text{unity of opposites: (56) with } \mathsf{E}:=\mathsf{F} \quad \} \\
 & \mathsf{L} \ .
\end{array}
$$

Hence,

$$
\begin{array}{ll}
 & \gamma\circ\alpha \\
= & \{ \quad \text{definition of } \alpha\text{: (101)} \quad \} \\
 & \gamma\circ\beta\circ\gamma \\
= & \{ \quad \text{above, } \mathsf{id}_{\mathcal{L}.\mathsf{F}} \text{ is unit of composition} \quad \} \\
 & \gamma \ .
\end{array}
$$

The proof of the final two equations is similar.

□

Informally, our goal is to show that the matrix indexed by the events $\alpha.i$ (where i ranges over left factors of E) and the matrix indexed by the events $\gamma.i$ are both “equal” to the factor matrix of F. The following lemma is significant because it asserts that the range of α is a subset of M which, we recall from corollary 95, represents the left factors of F.

Lemma 105 For all i such that i is a left factor of E,

$$\gamma.i \;=\; s\backslash\alpha.i \quad\wedge\quad \gamma.i\backslash F \;=\; \alpha.i\backslash t \quad\wedge\quad \alpha.i\in M \quad .$$

Proof Immediate from the definitions of α and γ, and lemma 94 with L instantiated to $\gamma.i$ and R instantiated to $\gamma.i\backslash F$. (The variable k in the lemma is $\alpha.i$.)

□

Summarising lemma 104 and our previous observation that γ maps left factors of E to left factors of F, we have:

Theorem 106 The function γ is a surjective function from the set of left factors of E onto the set of left factors of F. That is,

$$\langle\forall i\;:\; i \text{ is a left factor of } E \;:\; \gamma.i \text{ is a left factor of } F\rangle \quad .$$

Moreover,

$$\langle\forall L\;:\; L \text{ is a left factor of } F \;:\; \langle\exists i : i \text{ is a left factor of } E : L=\gamma.i\rangle\rangle \quad .$$

Specifically,

$$\langle\forall L \;\;:\;\; L \text{ is a left factor of } F \;\;:\;\; L=\gamma.(\beta.L)\rangle$$

where

$$\beta.L \;=\; t/(L\backslash F) \quad .$$

□

Some additional lemmas give further insight into the functions γ and α. We observe some specific values of α and γ (primarily to gain reassurance about the correctness of their definitions).

Lemma 107

$$\gamma.s=s\backslash s \;\wedge\; \alpha.s=s \;\wedge\; \gamma.t=s\backslash t \;\wedge\; \alpha.t=t \quad .$$

Proof First,

$\gamma.s$

$=$ { definition of γ: (100) }

$s\backslash t/(s\backslash t)$

$=$ { by lemma 86(e), $s = t/(s\backslash t)$ }

$s\backslash s$.

Second,

$\alpha.s$

$=$ { definition of α: (101) }

$t/((s\backslash s)\backslash(s\backslash t))$

$=$ { $s\cdot s\backslash s = s$ }

$t/(s\backslash t)$

$=$ { by lemma 86(e), $s = t/(s\backslash t)$ }

s .

Third,

$\gamma.t$

$=$ { definition of γ: (100) }

$s\backslash t/(t\backslash t)$

$=$ { lemma 48 }

$s\backslash t$.

Finally,

$\alpha.t$

$=$ { definition of α: (101) }

$t/((s\backslash t)\backslash(s\backslash t))$

$=$ { by lemma 91, $t\in M$;
so, by definition of M (definition 89) , $(s\backslash t)\backslash(s\backslash t) = t\backslash t$ }

$t/(t\backslash t)$

$=$ { lemma 48 }

t .

□

The next step is to observe that the matrices indexed by events in the ranges of α and γ are identical. It is convenient to observe an inclusion at the same time.

Lemma 108 For all left factors i and j of E ,

$$i\backslash j \;\preceq\; \gamma.i\backslash\gamma.j \;=\; \alpha.i\backslash\alpha.j \;\; .$$

Proof For the inclusion, we have:

$$
\begin{array}{ll}
 & i\backslash j \;\preceq\; \gamma.i\backslash\gamma.j \\
= & \{ \quad \text{definition of } \gamma\text{: (100)} \quad \} \\
 & i\backslash j \;\preceq\; (F/(i\backslash t))\backslash F/(j\backslash t) \\
= & \{ \quad \text{factors} \quad \} \\
 & F/(i\backslash t)\cdot i\backslash j\cdot j\backslash t \;\preceq\; F \\
= & \{ \quad \text{cancellation} \quad \} \\
 & \mathsf{true} \;\; .
\end{array}
$$

Now for the equality:

$$
\begin{array}{ll}
 & \alpha.i\backslash\alpha.j \\
= & \{ \quad \text{definition of } \alpha\text{: (101)} \quad \} \\
 & (t/(\gamma.i\backslash F))\backslash t/(\gamma.j\backslash F) \\
= & \{ \quad \gamma.i\backslash F \text{ is a right factor of } F\text{, lemma 94} \quad \} \\
 & \gamma.i\,\backslash\,F\,/\,(\gamma.j\backslash F) \\
= & \{ \quad \gamma.j \text{ is a left factor of } F\text{,} \\
 & \quad\;\; \text{unity of opposites: (56) with } E{:=}F \quad \} \\
 & \gamma.i\backslash\gamma.j \;\; .
\end{array}
$$

□

Lemma 108 is a statement about the factor matrix of E expressed in terms of individual elements. It is necessary to state it this way as a first step because its proof unavoidably exploits specific properties of the elements. However, for subsequent calculations it is unnecessary to reason in this way, and a point-free formulation is preferable. Specifically, we have:

Lemma 109

(110) $|\overline{\mathsf{E}}| \mathrel{\dot{\preceq}} (Sel.\gamma)^{\cup} \otimes |\overline{\mathsf{F}}| \otimes Sel.\gamma = (Sel.\alpha)^{\cup} \otimes |\overline{\mathsf{E}}| \otimes Sel.\alpha$

Hence,

(111) $Sel.\gamma \otimes |\overline{\mathsf{E}}| \otimes (Sel.\gamma)^{\cup} \mathrel{\dot{\preceq}} |\overline{\mathsf{F}}|$

and

(112) $Sel.\alpha \otimes |\overline{\mathsf{E}}| \otimes (Sel.\alpha)^{\cup} \mathrel{\dot{\preceq}} |\overline{\mathsf{E}}|$

Proof The proof of (110) is straightforward: expand the definitions of its components (the pointwise ordering, $|\overline{\mathsf{E}}|$, $|\overline{\mathsf{F}}|$ etc.), taking care to note from theorem 106 that γ is a surjective function from $\mathcal{L}.\mathsf{E}$ onto $\mathcal{L}.\mathsf{F}$, and then apply lemma 108. The proof of (111) is as follows.

$Sel.\gamma \otimes |\overline{\mathsf{E}}| \otimes (Sel.\gamma)^{\cup}$

$\dot{\preceq}$ { (110) and monotonicity of matrix product }

$Sel.\gamma \otimes (Sel.\gamma)^{\cup} \otimes |\overline{\mathsf{F}}| \otimes Sel.\gamma \otimes (Sel.\gamma)^{\cup}$

$=$ { by theorem 106, γ is a surjective function onto $\mathcal{L}.\mathsf{F}$,
so, by (32) and (30), $Sel.\gamma \otimes (Sel.\gamma)^{\cup} = \mathbf{I}_{\mathcal{L}.\mathsf{F}}$ }

$\mathbf{I}_{\mathcal{L}.\mathsf{F}} \otimes |\overline{\mathsf{F}}| \otimes \mathbf{I}_{\mathcal{L}.\mathsf{F}}$

$=$ { $\mathbf{I}_{\mathcal{L}.\mathsf{F}}$ is the identity of multiplication
on matrices indexed by $\mathcal{L}.\mathsf{F}$ }

$|\overline{\mathsf{F}}|$.

Finally, we prove (112).

$Sel.\alpha \otimes |\overline{\mathsf{E}}| \otimes (Sel.\alpha)^{\cup}$

$\dot{\preceq}$ { (110) and monotonicity of matrix product }

$Sel.\alpha \otimes (Sel.\alpha)^{\cup} \otimes |\overline{\mathsf{E}}| \otimes Sel.\alpha \otimes (Sel.\alpha)^{\cup}$

$\dot{\preceq}$ { α is a function of type $\mathcal{L}.\mathsf{E} \leftarrow \mathcal{L}.\mathsf{E}$,
so $Sel.\alpha \otimes (Sel.\alpha)^{\cup} \mathrel{\dot{\preceq}} \mathbf{I}_{\mathcal{L}.\mathsf{E}}$ }

$\mathbf{I}_{\mathcal{L}.\mathsf{E}} \otimes |\overline{\mathsf{E}}| \otimes \mathbf{I}_{\mathcal{L}.\mathsf{E}}$

$=$ { $\mathbf{I}_{\mathcal{L}.\mathsf{E}}$ is the identity of multiplication
on matrices indexed by $\mathcal{L}.\mathsf{E}$ }

$|\overline{\mathsf{E}}|$.

Take care to note the difference in the second step of these calculations. In the first of the two, the first equality step could have been replaced by an inequality, using only the fact that γ is a function (and thus not that it is surjective). We have established the stronger equality in order to better illustrate this form of point-free calculation.
□

Rather than give an explicit formula for $\alpha.\mathrm{i}$, Conway's proof technique is to observe that $s\backslash i \cdot i\backslash t$ is a "subfactorisation" of $s\backslash t$ and then make the claim that there is some "factorisation" $L{\cdot}R$ that "dominates" this subfactorisation. The following lemma shows that our definition of α gives such a factorisation.

Lemma 113 For all i such that i is a left factor of E,

$$s\backslash i \preceq s\backslash\alpha.i \quad\wedge\quad i\backslash t \preceq \alpha.i\backslash t$$

Proof For the first conjunct, we have:

$$\begin{array}{ll}
 & s\backslash i \preceq s\backslash\alpha.i \\
= & \{\quad \text{lemma 105, } F = s\backslash t \quad\} \\
 & s\backslash i \preceq s\backslash t/(i\backslash t) \\
\Leftarrow & \{\quad \text{monotonicity} \quad\} \\
 & i \preceq t/(i\backslash t) \\
= & \{\quad \text{lemma 46 with } X,Y := i,t \quad\} \\
 & \mathsf{true}\ .
\end{array}$$

For the second conjunct, we have:

$$\begin{array}{ll}
 & i\backslash t \preceq \alpha.i\backslash t \\
= & \{\quad \text{lemma 105} \quad\} \\
 & i\backslash t \preceq (F/(i\backslash t))\backslash F \\
= & \{\quad \text{lemma 46 with } X,Y := F, i\backslash t \quad\} \\
 & \mathsf{true}\ .
\end{array}$$

□

Theorem 98 shows how the factor matrix of F is represented by a submatrix of the factor matrix of E using the bijection β between a subset of $\mathcal{L}.E$ and the set $\mathcal{L}.F$. An informal summary of the following theorem is that γ maps $|\overline{E}|$ to $|\overline{F}|$ by coalescing left factors of E.

Theorem 114

$$|\overline{F}| \;=\; Sel.\gamma \otimes |\overline{E}| \otimes (Sel.\gamma)^{\cup} \;\; .$$

Proof The proof is by mutual inclusion. By (111), it suffices to prove

$$|\overline{F}| \;\dot{\preceq}\; Sel.\gamma \otimes |\overline{E}| \otimes (Sel.\gamma)^{\cup} \;\; .$$

We have:

$$\begin{array}{ll}
 & |\overline{F}| \\
= & \{ \quad (99) \quad \} \\
 & (Sel.\beta)^{\cup} \otimes |\overline{E}| \otimes Sel.\beta \\
= & \{ \quad \text{lemma 104, so } Sel.(\gamma \circ \beta) = \mathbf{I}_{\mathcal{L}.F} \quad \} \\
 & Sel.(\gamma \circ \beta) \otimes (Sel.\beta)^{\cup} \otimes |\overline{E}| \otimes Sel.\beta \otimes (Sel.(\gamma \circ \beta))^{\cup} \\
= & \{ \quad \text{distributivity of } Sel \text{ over function composition,} \\
 & \quad\;\; \text{and transpose over } Sel \text{ and matrix multiplication} \quad \} \\
 & Sel.\gamma \otimes Sel.\beta \otimes (Sel.\beta)^{\cup} \otimes |\overline{E}| \otimes Sel.\beta \otimes (Sel.\beta)^{\cup} \otimes (Sel.\gamma)^{\cup} \\
\dot{\preceq} & \{ \quad \beta \text{ is an injection with image set } M, \\
 & \quad\;\; \text{so } Sel.\beta \otimes (Sel.\beta)^{\cup} \dot{\preceq} \mathbf{I}_M \;; \\
 & \quad\;\; \text{monotonicity of matrix multiplication} \quad \} \\
 & Sel.\gamma \otimes |\overline{E}| \otimes (Sel.\gamma)^{\cup} \;\; .
\end{array}$$

The equality follows from the anti-symmetry of the pointwise ordering relation.
□

4.4 Diagonal Factors

Under construction*

When constructing the factor graphs of factors of an event E, it suffices to consider the factors on the diagonal of the factor matrix of E. This section justifies this claim and gives a condition under which a "diagonal" factor is inseparable from E. First, the formal definition of "diagonal factor":

Definition 115 A factor F of an event E is called a *diagonal factor* of E if F equals $i \backslash i$ for some left factor i of E.
□

Lemma 116 Suppose i is a left factor of E. Let s and t be left factors of E defined by $s = i/(i\backslash i)$ and $t = ((s\rhd / i\rhd) \backslash s\rhd)\lhd$. (That is, we instantiate lemma 86 with $i,j := i,i$.) Then $s = i$ and $t = i$.

Proof The equation $s = i/(i\backslash i)$ is an instance of the general property (48). In order to establish that $t = i$, we need the fact that i is a left factor of E:

$$
\begin{array}{ll}
 & ((s\rhd / i\rhd) \backslash s\rhd)\lhd \\
= & \{ \quad s = i \quad \} \\
 & ((i\rhd / i\rhd) \backslash i\rhd)\lhd \\
= & \{ \quad (48) \quad \} \\
 & (i\rhd)\lhd \\
= & \{ \quad i \text{ is a left factor of (so equals } X\lhd \text{ for some } X), \\
 & \quad\quad \text{unity of opposites} \quad \} \\
 & i \ .
\end{array}
$$

□

In words, the factor matrix of a diagonal factor of E is a diagonal submatrix of the factor matrix of E. (It is a submatrix such that, under a suitable reordering of rows and columns, it sits on the diagonal when the factor matrix is displayed in the conventional way as a two-dimensional array.)

5 Approximation Theorems

An important property of the factor matrix is that it facilitates "approximation" of one event by a set of other events.

Conway formulated a theorem [Con71, chapter 6, theorem 8] on the use of what he called the "factorial function" to determine the "best approximation" to the factor matrix of an event E by a given set of events. Below, we generalise his theorem to an arbitrary regular algebra (rather than just the algebra of languages)

When specialised to the algebra of languages, we argue that his use of the word "best" is unfortunate: for the purpose of minimising star-height, his notion of "best" is certainly not the best, and for other applications the claim is also questionable. We therefore formulate Conway's theorem as determining a "*maximal* constant+linear approximating function"; this is also the terminology we use in the general case.

For regular languages, we also show in section 6 how to determine a "*minimal* constant+linear approximating function". Both Conway's maximal and our minimal approximating functions are parameterised by a function on an alphabet T. When the event E is a regular language over alphabet T, we obtain the "factor graph" of E by specialising the function to the identity function on T: the factor graph of E is the unique minimal starth root of the factor matrix of E.

5.1 Conway's Notion of Approximation

First, let us explain Conway's notion of approximation.

Let T be an alphabet and let $\mathcal{R}$ be the algebra of languages over the alphabet T. Let $\mathcal{S}$ be a regular algebra with carrier S. Suppose E is an event in $\mathcal{S}$.

Suppose we are given a set of events in $\mathcal{S}$ and we want to "approximate" E by the set. We begin by encoding the set of events by a function ζ from T into S, where T is an alphabet whose size is the size of the set: the set of events is, in fact, the image set of ζ. Next, ζ is extended to a function from T^* into S by defining

(117) $\quad \zeta.\varepsilon \;=\; 1$

and, for all a in T and all u in T^*,

(118) $\quad \zeta.(au) \;=\; \zeta.a \cdot \zeta.u \quad .$

The function $\zeta^\flat$ from $\mathcal{R}$ to $\mathcal{S}$ is defined by

(119) $\quad \zeta^\flat.X \;=\; \langle \Sigma x : x{\in}X : \zeta.x \rangle$

for all languages X over the alphabet T. Then, by a *function approximating* the event E is meant a set of words X such that $\zeta^\flat.X \preceq E$; the *approximation* of E is $\zeta^\flat.X$.

(The strange terminology is Conway's: the approximation is obtained by applying the function $\zeta^\flat$ to the approximating "function" (a set of words), rather than the other way around. But Conway does not explicitly mention the extension of ζ to words or the subsequent extension of ζ to the function $\zeta^\flat$ even though it is a vital element of his analysis. More importantly, Conway makes no mention of the algebraic properties of the function $\zeta^\flat$ that enable the evaluation of $\zeta^\flat.X$ to be carried out.)

Conway calls a set of words X a "best approximation" to E if it is a maximal solution of the equation

$$X{::}\ \zeta^\flat.X \preceq E \quad .$$

For reasons discussed further below, we refrain from using the word "best", preferring instead the terminology "*maximal* approximating function".

(Conway also briefly discusses the dual problem of, given a set of words X over some alphabet T, determine maximal solutions of the equation

$$\zeta{::}\ \zeta^\flat.X \preceq E \quad .$$

See [Con71, chapter 6, theorem 9]. We don't discuss the dual problem any further here.)

The existence of a maximal approximation is guaranteed by observing that $\zeta^\flat$ is the lower adjoint in a Galois connection. Specifially, define the function $\zeta^\sharp$ from $\mathcal{S}$ to $\mathcal{R}$ by

(120) $\zeta^\sharp.U \;=\; \{x \mid \zeta.x \preceq U\}$.

Then we have the following lemma and theorem:

Lemma 121 The function ζ is a monoid homomorphism. That is, $\zeta.\varepsilon = 1$ and $\zeta.(uv) = \zeta.u \cdot \zeta.v$, for all u and v in T^*.

Proof The first equation is by definition. The second equation is an easy induction on the length of u.
□

Theorem 122 The function $\zeta^\flat$ is a regular homomorphism from the algebra of languages over T to $\mathcal{S}$. Moreover, $\zeta^\sharp$ is the upper adjoint of $\zeta^\flat$.

Proof The theorem is an instance of lemma 21. Instantiate R in the lemma to the set of words over alphabet T. Then the algebra $\mathcal{R}$ is the algebra of languages over T. The lemma is applicable on account of lemma 121.
□

Note: for all languages X over alphabet T and all events E,

$$X \subseteq \zeta^\sharp.E \;\;\equiv\;\; \langle \forall x : x{\in}X : \zeta.x \preceq E \rangle \quad .$$

Conway describes this equation in words (roughly) as follows. Suppose we are given a set of events $\zeta.a$ indexed by a in some set T and we want to find the most general expression for E in terms of these events. Defining the *"best" approximation to* E *in terms of* ζ as the set of all words x in T^* such that $\zeta.x \preceq E$, the above equation states that the "best" approximating function is the function that maps ζ to $\zeta^\sharp.E$.

We have used inverted commas here because Conway's notion of "best" approximating function is not the "best" for our purposes. More appropriate terminology is "maximal" approximating function, as illustrated by the following simple example.

Example 123 Suppose E is an event in regular algebra $\mathcal{S}$ such that $E = E^*$. Suppose we want to approximate E by E. Then we take T to be a singleton set, say $\{a\}$, and define ζ by $\zeta.a = E$. The extension of ζ to T^* is given by $\zeta.\varepsilon \;=\; 1$ and, for all u in T^*, $\zeta.(au) \;=\; E$. (Note that $E = E^*$ implies that $E = E{\cdot}E$.) Instantiating the definition, of $\zeta^\sharp$, we find that $\zeta^\sharp.E = T^*$. Thus Conway's "best" approximating function maps ζ to T^*. An alternative approximating function is the function that maps ζ to T. The

former is maximal, the latter is minimal. When we interpret the languages by applying the function ζ, both give the same approximation and so could be described as equally good. But the latter function can be argued to be more useful because it is easier to interpret.
□

We conclude this subsection with a precise definition of our terminology.

Definition 124 Given an event E in a regular algebra $\mathcal{S}$, an alphabet T and a function ζ of type $\mathcal{S} \leftarrow T^*$, the language $\zeta^\sharp.E$ is called the *maximal approximating function for* E, and $\zeta^\flat.(\zeta^\sharp.E)$ is called the *maximal approximation of* E *by* ζ.

Similarly, the matrix $\zeta^\sharp \bullet |\overline{E}|$ is called the *maximal approximating function for the factor matrix of* E, and $\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{E}|$ is called the *maximal approximation of the factor matrix of* E *by* ζ.
□

5.2 Maximal Constant+Linear Matrix

The matrix $\zeta^\sharp \bullet |\overline{E}|$ is the maximal approximating function for the factor matrix of E by the function ζ. In this section, we formulate and prove Conway's theorem that $\zeta^\sharp \bullet |\overline{E}|$ is the reflexive, transitive closure of the maximal constant+linear approximating function for the factor matrix of E by the function ζ. See theorem 132. First, we need to define what the terminology means and to establish some lemmas.

Definition 125 Let I be a set and let $\mathcal{R}$ be a regular algebra with carrier set A and unit 1. Define the function Mat from A to $\mathcal{M}_I(A)$ by

$$i\ \mathsf{Mat}.U\ j \;=\; U$$

for all U in A and all elements i, j of I. In words, Mat constructs a matrix from an element of A all of whose entries are identical to that element.
□

The following lemma underpins some of our calculations.

Lemma 126 For all U in A,

$$(\mathsf{Mat}.U)^+ \;=\; \mathsf{Mat}.(U^+) \quad .$$

Hence,

$$(\mathsf{Mat}.U)^* \;=\; \mathsf{Mat}.1 \;\dot{\cup}\; \mathsf{Mat}.(U^+) \quad .$$

Proof Mat is the lower adjoint in a Galois connection: its upper adjoint is the intersection operator of shape $I{\times}I$. The lemma is then an easy consequence of the fusion theorem using the definition of U^+ as the least fixed point of the function mapping X to $U \dot{\cup} X{\otimes}X$.

□

Conway calls a matrix of languages that is at most $\mathsf{Mat}.\{\varepsilon\}$ a *constant* matrix and a matrix of languages that is at most $\mathsf{Mat}.T$ a *linear* matrix. He calls a matrix of languages that is at most $\mathsf{Mat}.\{\varepsilon\} \dot{\cup} \mathsf{Mat}.T$ a *constant+linear* matrix. Although we sometimes use Conway's terminology (in order to make the link with his work) we prefer to use the terminology *transition graph* instead of "constant+linear matrix". A "transition graph" that is square and has finite dimension can be depicted in the conventional way as a set of nodes connected by edges that are labelled by a subset of $\{\varepsilon\}{\cup}T$.

When reasoning about constant and/or linear matrices, we frequently give as hint "length considerations". Constant matrices are matrices each of whose entries is either the empty set or the singleton set containing the empty word, which has length 0. The set of constant matrices is closed under matrix product and under reflexive and/or transitive closure. The linear matrices are matrices each of whose entries is a (possibly empty) set of words each of length 1. The set of linear matrices is thus closed under product with a constant matrix but the product of two or more linear matrices is not linear. It is these properties that we assume when referring to "length considerations".

The following lemma is fundamental to Conway's account but he does not explicitly state or prove it. The proof, which involves well-known techniques, is given in appendix A.

Lemma 127 Let $\mathbf{A}$ be an arbitrary matrix of languages over the alphabet T. Suppose $\mathbf{A} = \mathbf{A}^*$. Let $\mathbf{C} = \mathbf{A} \dot{\cap} \mathsf{Mat}.\{\varepsilon\}$ and $\mathbf{L} = \mathbf{A} \dot{\cap} \mathsf{Mat}.T$. Then

$$\mathbf{C} = \mathbf{C}^* \quad ,$$

$$\mathbf{L} = \mathbf{L}{\otimes}\mathbf{C} = \mathbf{C}{\otimes}\mathbf{L} \quad ,$$

$$(\mathbf{C} \dot{\cup} \mathbf{L})^* = \mathbf{C} \dot{\cup} \mathbf{L}^+ = \mathbf{C} \otimes \mathbf{L}^* = \mathbf{L}^* \otimes \mathbf{C} \quad \text{, and}$$

$$\mathbf{L} \otimes (\mathbf{C} \dot{\cup} \mathbf{L})^* = \mathbf{L}^+ \quad .$$

□

Let us now turn to the properties of the matrix $\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{E}|$, which we recall is the maximal approximation of $|\overline{E}|$ by the function ζ. In order to be able to apply lemma 127, we begin by observing that both it and $\zeta^\sharp \bullet |\overline{E}|$ are their own reflexive, transitive closures:

Lemma 128

$$\zeta^\sharp \bullet |\overline{\mathsf{E}}| = (\zeta^\sharp \bullet |\overline{\mathsf{E}}|)^* \quad \text{, and}$$
$$\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{\mathsf{E}}| = (\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{\mathsf{E}}|)^* \quad .$$

Proof Suppose I is the set of left factors of E. We use $\mathbf{I}_{\mathcal{S}}$ to denote the identity matrix indexed by I in the algebra $\mathcal{S}$.

$$\begin{array}{ll}
 & \zeta^\sharp \bullet |\overline{\mathsf{E}}| \;=\; (\zeta^\sharp \bullet |\overline{\mathsf{E}}|)^* \\
= & \quad \{ \quad \text{definition of } {}^*\text{, monotonicity of } (\,\zeta^\sharp \bullet\,) \quad \} \\
 & \zeta^\sharp \bullet |\overline{\mathsf{E}}| \;\dot{\supseteq}\; (\zeta^\sharp \bullet |\overline{\mathsf{E}}|)^* \\
= & \quad \{ \quad \zeta^\sharp \text{ has lower adjoint } \zeta^\flat \quad \} \\
 & |\overline{\mathsf{E}}| \;\dot{\succeq}\; \zeta^\flat \bullet (\zeta^\sharp \bullet |\overline{\mathsf{E}}|)^* \\
= & \quad \{ \quad \zeta^\flat \text{ is a regular homomorphism (theorem 122),} \\
 & \quad\quad \text{theorem 26} \quad \} \\
 & |\overline{\mathsf{E}}| \;\dot{\succeq}\; (\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{\mathsf{E}}|)^* \\
\Leftarrow & \quad \{ \quad \mathbf{I}_{\mathcal{S}} \;\dot{\succeq}\; \zeta^\flat \bullet \zeta^\sharp \text{, monotonicity and transitivity} \quad \} \\
 & |\overline{\mathsf{E}}| \;\dot{\succeq}\; |\overline{\mathsf{E}}|^* \\
= & \quad \{ \quad |\overline{\mathsf{E}}| = |\overline{\mathsf{E}}|^* \quad \} \\
 & \mathsf{true} \quad .
\end{array}$$

The second equation is an application of theorem 26 with $\mathbf{G} := \zeta^\sharp \bullet |\overline{\mathsf{E}}|$ (as in the middle step above).
□

Now we introduce the maximal constant and linear approximations to $|\overline{\mathsf{E}}|$ by the function ζ.

Definition 129 (Maximal Constant and Linear Approximations) We define the constant matrix $\mathbf{C}_{max}(\mathsf{E},\zeta)$ by

$$\mathbf{C}_{max}(\mathsf{E},\zeta) \;=\; (\zeta^\sharp \bullet |\overline{\mathsf{E}}|) \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\}$$

and the linear matrix $\mathbf{L}_{max}(\mathsf{E},\zeta)$ by

$$\mathbf{L}_{max}(\mathsf{E},\zeta) \;=\; (\zeta^\sharp \bullet |\overline{\mathsf{E}}|) \,\dot{\cap}\, \mathsf{Mat}.\mathsf{T} \quad .$$

The matrices $\mathbf{C}_{max}(\mathsf{E},\zeta)$ and $\mathbf{L}_{max}(\mathsf{E},\zeta)$ are, respectively, the *maximal constant approximating function* and the *maximal linear approximating function* for the factor matrix of E.

The *maximal constant approximation* of the factor matrix of E by ζ is $\zeta^\flat \bullet \mathbf{C}_{max}(\mathsf{E},\zeta)$ and the *maximal linear approximation* of the factor matrix of E by ζ is $\zeta^\flat \bullet \mathbf{L}_{max}(\mathsf{E},\zeta)$.
□

Lemma 130

$$\mathbf{C}_{max}(\mathsf{E},\zeta) \;=\; (\mathbf{C}_{max}(\mathsf{E},\zeta))^* \quad ,$$

$$\mathbf{L}_{max}(\mathsf{E},\zeta) \;=\; \mathbf{L}_{max}(\mathsf{E},\zeta) \otimes \mathbf{C}_{max}(\mathsf{E},\zeta) \;=\; \mathbf{C}_{max}(\mathsf{E},\zeta) \otimes \mathbf{L}_{max}(\mathsf{E},\zeta) \quad ,$$

$$(\mathbf{C}_{max}(\mathsf{E},\zeta) \,\dot{\cup}\, \mathbf{L}_{max}(\mathsf{E},\zeta))^* \;=\; \mathbf{C}_{max}(\mathsf{E},\zeta) \otimes (\mathbf{L}_{max}(\mathsf{E},\zeta))^* \quad ,$$

$$(\mathbf{C}_{max}(\mathsf{E},\zeta) \,\dot{\cup}\, \mathbf{L}_{max}(\mathsf{E},\zeta))^* \;=\; (\mathbf{L}_{max}(\mathsf{E},\zeta))^* \otimes \mathbf{C}_{max}(\mathsf{E},\zeta) \quad ,$$

$$(\mathbf{C}_{max}(\mathsf{E},\zeta) \,\dot{\cup}\, \mathbf{L}_{max}(\mathsf{E},\zeta))^* \;=\; \mathbf{C}_{max}(\mathsf{E},\zeta) \,\dot{\cup}\, (\mathbf{L}_{max}(\mathsf{E},\zeta))^+ \quad \text{, and}$$

$$\mathbf{L}_{max}(\mathsf{E},\zeta) \otimes (\mathbf{C}_{max}(\mathsf{E},\zeta) \,\dot{\cup}\, \mathbf{L}_{max}(\mathsf{E},\zeta))^* \;=\; (\mathbf{L}_{max}(\mathsf{E},\zeta))^+ \quad .$$

Proof Immediate application of lemmas 128 and 127. (Instantiate lemma 127 with $\mathbf{A},\mathbf{C},\mathbf{L} := (\zeta^\sharp \bullet |\overline{\mathsf{E}}|), \mathbf{C}_{max}(\mathsf{E},\zeta), \mathbf{L}_{max}(\mathsf{E},\zeta)$.)
□

We note that the matrix $\mathbf{C}_{max}(\mathsf{E},\zeta)$ is (almost) independent of the function ζ:

Lemma 131 For all left factors i and j of E,

$$i \;\mathbf{C}_{max}(\mathsf{E},\zeta)\; j \;=\; \text{if } i \preceq j \rightarrow \{\varepsilon\} \;\square\; \neg(i \preceq j) \rightarrow \emptyset \text{ fi} \quad .$$

Hence, for all approximating functions ζ and ξ with the same domain,

$$\mathbf{C}_{max}(\mathsf{E},\zeta) \;=\; \mathbf{C}_{max}(\mathsf{E},\xi) \quad .$$

Proof

$\quad\quad \varepsilon \in i \; (\zeta^\sharp \bullet |\overline{\mathsf{E}}|) \; j$

$=\quad\quad \{ \quad i \, |\overline{\mathsf{E}}| \, j = i \backslash j \quad \}$

$\quad\quad \varepsilon \in \zeta^\sharp.(i \backslash j)$

$=\quad\quad \{ \quad$ definition of $\zeta^\sharp$: (120) $\quad \}$

$\quad\quad \zeta.\varepsilon \preceq i \backslash j$

$=\quad\quad \{ \quad$ definition of extension of ζ to T^* $\quad \}$

$\quad\quad 1 \preceq i \backslash j$

$=\quad\quad \{ \quad$ factors and unit $\quad \}$

$\quad\quad i \preceq j \quad .$

The lemma follows by definition of $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)$ and $\mathsf{Mat}.\{\varepsilon\}$: all entries in $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)$ are either $\{\varepsilon\}$ or $\emptyset$.

$\square$

The qualification in lemma 131 that ζ and ξ must have the same domain arises, of course, because the empty word has a type that depends on the alphabet T (the notation "ε" for the empty word is overloaded). Thus, the function $\mathbf{C}_{\mathsf{max}}$ retains some dependency on its second argument. Ignoring this dependency, $\mathbf{C}_{\mathsf{max}}$ depends only on E. We are tempted to abbreviate the notation to reflect this fact, but resist the temptation. See section 5.3 for further discussion.

Theorem 132 (Generalised Approximation Theorem) Suppose ζ is a function from alphabet T into $\mathcal{S}$ (as in section 5.1). Then for any event E in $\mathcal{S}$,

$$\zeta^\sharp\bullet|\overline{\mathsf{E}}| \;=\; \mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \dot{\cup}\ (\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^+ \;=\; (\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \dot{\cup}\ \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^* \;\;.$$

Proof We use induction on the length of words. The induction hypothesis is that, for all words u of length at most n, and for all left factors i and j of E,

$$u\;\in\; i\,(\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\,j \;\;\equiv\;\; u\;\in\; i\,(\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \dot{\cup}\ ((\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^+))\,j \;\;.$$

Applying lemma 130, this is the same as

$$u\;\in\; i\,(\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\,j \;\;\equiv\;\; u\;\in\; i\,(\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\dot{\cup}\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^*\,j \;\;.$$

It is convenient to switch between the two formulations of the inductive hypothesis.

For the basis of the induction we have, for all left factors i and j of E,

$\qquad\varepsilon\;\in\; i\,(\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\,j$

$=\qquad\{\qquad$ definition of $\mathsf{Mat}.\{\varepsilon\}$ and $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\quad\}$

$\qquad\varepsilon\;\in\; i\,\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\,j$

$=\qquad\{\qquad$ words in $(\zeta^\sharp\circ|\overline{\mathsf{E}}|)\,\dot{\cap}\,\mathsf{Mat}.\mathsf{T}$ have length at least 1,

$\qquad\qquad\quad$ definitions of $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)$ and $\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)\quad\}$

$\qquad\varepsilon\;\in\; i\,(\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \dot{\cup}\ (\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^+)\,j\;\;.$

Now assume the induction hypothesis and suppose v is a word in T^* of length $n{+}1$. Then $v = au$ for some symbol a in T and word u in T^* of length n. So

$\qquad v\;\in\; i\,(\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\,j$

$=\qquad\{\qquad v = au$, set membership $\quad\}$

$\qquad\{au\}\;\subseteq\; i\,(\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\,j$

$=$ { definition of extension of $\zeta^\sharp$ to matrices,
$i\,|\overline{E}|\,j \;=\; i\backslash j$ }

$\{au\} \;\subseteq\; \zeta^\sharp.(i\backslash j)$

$=$ { $\zeta^\sharp$ is an upper adjoint with lower adjoint $\zeta^\flat$ }

$\zeta^\flat.\{au\} \preceq i\backslash j$

$=$ { theorem 122 }

$\zeta^\flat.\{a\} \;\times_S\; \zeta^\flat.\{u\} \;\preceq\; i\backslash j$

$=$ { (78) }

$\langle \exists k \;::\; \zeta^\flat.\{a\} \preceq i\backslash k \;\wedge\; \zeta^\flat.\{u\} \preceq k\backslash j \rangle$

$=$ { Galois connection }

$\langle \exists k \;::\; \{a\} \subseteq \zeta^\sharp.(i\backslash k) \;\wedge\; \{u\} \subseteq \zeta^\sharp.(k\backslash j) \rangle$

$=$ { set membership, defn. of extension of $\zeta^\sharp$ to matrices,
$k\,|\overline{E}|\,j \;=\; k\backslash j$ }

$\langle \exists k \;::\; a \in i\;(\zeta^\sharp \bullet |\overline{E}|)\;k \wedge u \in k\;(\zeta^\sharp \bullet |\overline{E}|)\;j \rangle$

$=$ { a has length 1, so
$a \in i\;(\zeta^\sharp \bullet |\overline{E}|)\;k \equiv a \in i\;((\zeta^\sharp \bullet |\overline{E}|) \dot{\cap} \mathsf{Mat.T})\;k$
definition of $\mathbf{L}_{max}(E,\zeta)$ and inductive hypothesis }

$\langle \exists k \;::\; a \in i\;\mathbf{L}_{max}(E,\zeta)\;k \;\wedge\; u \;\in\; k\;(\mathbf{C}_{max}(E,\zeta) \dot{\cup} \mathbf{L}_{max}(E,\zeta))^*\;j \rangle$

$=$ { definition of matrix product,
lemma 127 with $\mathbf{A},\mathbf{C},\mathbf{L} := \zeta^\sharp \bullet |\overline{E}|$, $\mathbf{C}_{max}(E,\zeta)$, $\mathbf{L}_{max}(E,\zeta)$
(which is applicable by lemma 128) }

$au \;\in\; i\;(\mathbf{L}_{max}(E,\zeta))^+\;j$

$=$ { au has length at least 1, elements of $\mathbf{C}_{max}(E,\zeta)$ have length 0 }

$au \;\in\; i\;(\mathbf{C}_{max}(E,\zeta) \;\dot{\cup}\; (\mathbf{L}_{max}(E,\zeta))^+)\;j$.

The theorem now follows from the definition of equality of matrices of languages.
□

5.3 The Factorial Function

We conclude this section with an explanation of Conway's factorial function (again generalised to an arbitrary regular algebra). We begin with the formal statement of the theorem.

Theorem 133 (Factorial Function) Suppose $\mathcal{S}$ is a regular algebra and T is an alphabet. Suppose ζ is a function from alphabet T into $\mathcal{S}$ (as in section 5.1). Then for any event E in $\mathcal{S}$,

$$\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{\mathsf{E}}| \;=\; \mathbf{C}_{\mathsf{max}}.\mathsf{E} \otimes (\zeta^\flat \bullet \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^* = \mathbf{C}_{\mathsf{max}}.\mathsf{E} \otimes (\zeta^\flat \bullet (\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^*)$$

where the matrix $\mathbf{C}_{\mathsf{max}}.\mathsf{E}$ is defined by

$$\mathsf{i}\ \mathbf{C}_{\mathsf{max}}.\mathsf{E}\ \mathsf{j} \;=\; \text{if } \mathsf{i} \preceq \mathsf{j} \to 1 \;\square\; \neg(\mathsf{i} \preceq \mathsf{j}) \to 0 \text{ fi} \quad .$$

Moreover,

$$\mathsf{i}\ (\zeta^\flat \bullet \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))\ \mathsf{j} \;=\; \langle \Sigma \mathsf{a} : \mathsf{a} \in \mathsf{T} \wedge \zeta.\mathsf{a} \preceq \mathsf{i} \backslash \mathsf{j} : \zeta.\mathsf{a} \rangle \quad .$$

The function that maps ζ to $\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)$ is called the *factorial function* of E.

Proof We apply theorem 26 with $\mathbf{G}$ instantiated to $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta) \,\dot{\cup}\, \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)$. First, we note that all elements of this matrix are closed since

$$\begin{array}{ll}
 & \zeta^\sharp \bullet \zeta^\flat \bullet (\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta) \,\dot{\cup}\, \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^* \\
= & \quad \{ \quad \text{theorem 132} \quad \} \\
 & \zeta^\sharp \bullet \zeta^\flat \bullet \zeta^\sharp \bullet |\overline{\mathsf{E}}| \\
= & \quad \{ \quad \text{unity of opposites} \quad \} \\
 & \zeta^\sharp \bullet |\overline{\mathsf{E}}| \\
= & \quad \{ \quad \text{theorem 132} \quad \} \\
 & (\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta) \,\dot{\cup}\, \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))^* \quad .
\end{array}$$

Next, we note that $\zeta^\flat \bullet \mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)$ is independent of ζ. For all left factors i and j,

$$\begin{array}{ll}
 & \mathsf{i}\ (\zeta^\flat \bullet \mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta))\ \mathsf{j} \\
= & \quad \{ \quad \text{definition of function composition} \quad \} \\
 & \zeta^\flat . (\mathsf{i}\ \mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \mathsf{j}) \\
= & \quad \{ \quad \text{lemma 131} \quad \} \\
 & \zeta^\flat . (\text{if } \mathsf{i} \preceq \mathsf{j} \to \{\varepsilon\} \;\square\; \neg(\mathsf{i} \preceq \mathsf{j}) \to \emptyset \text{ fi})
\end{array}$$

$=$ { definition of $\zeta^\flat$ }

if $i \preceq j \rightarrow 1 \,\square\, \neg(i \preceq j) \rightarrow 0$ fi

$=$ { definition of $\mathbf{C}_{max}.E$ }

$i\ (\mathbf{C}_{max}.E)\ j$.

So

$\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{E}|$

$=$ { theorem 132 }

$\zeta^\flat \bullet (\mathbf{C}_{max}(E,\zeta) \mathbin{\dot{\cup}} \mathbf{L}_{max}(E,\zeta))^*$

$=$ { star decomposition }

$\zeta^\flat \bullet (\mathbf{C}_{max}(E,\zeta))^* \otimes (\mathbf{L}_{max}(E,\zeta) \otimes (\mathbf{C}_{max}(E,\zeta))^*)^*$

$=$ { by lemma 130, $(\mathbf{C}_{max}(E,\zeta))^* = \mathbf{C}_{max}(E,\zeta)$
and $\mathbf{L}_{max}(E,\zeta) \otimes \mathbf{C}_{max}(E,\zeta) = \mathbf{L}_{max}(E,\zeta)$ }

$\zeta^\flat \bullet \mathbf{C}_{max}(E,\zeta) \otimes (\mathbf{L}_{max}(E,\zeta))^*$

$=$ { theorem 26 (applied twice) }

$(\zeta^\flat \bullet \mathbf{C}_{max}(E,\zeta)) \otimes (\zeta^\flat \bullet \mathbf{L}_{max}(E,\zeta))^*$

$=$ { by above, $\zeta^\flat \bullet \mathbf{C}_{max}(E,\zeta) = \mathbf{C}_{max}.E$ }

$\mathbf{C}_{max}.E \otimes (\zeta^\flat \bullet \mathbf{L}_{max}(E,\zeta))^*$

$=$ { theorem 26 }

$\mathbf{C}_{max}.E \otimes (\zeta^\flat \bullet (\mathbf{L}_{max}(E,\zeta))^*)$.

The final equation is a straightforward expansion of the definitions of $\zeta^\flat$ and $\mathbf{L}_{max}(E,\zeta)$.

□

Let us interpret this theorem in words.

Recall that ζ is a function that represents a finite set of approximating events: the set corresponding to ζ is the image set of ζ, i.e. $\{a{:}\, a{\in}T{:}\, \zeta.a\}$. The entry in the matrix $\zeta^\sharp \bullet |\overline{E}|$ indexed by left factors i and j of E is the maximal set of words w such that $\zeta^\flat.w \preceq i\backslash j$. The entry in the matrix $\zeta^\flat \bullet \zeta^\sharp \bullet |\overline{E}|$ that is indexed by left factors i and j of E is the maximal approximation to the factor $i\backslash j$ by the function ζ. In particular, the entry indexed by the left factors l and r (where (l,r) is the defining occurrence of E in its factor matrix: see section 3.4) is the maximal approximation to E by the function ζ.

The matrix $\mathbf{C}_{max}.E$ is independent of ζ: it is a matrix of 1s and 0s where the entry is 1 iff the corresponding factor of E is at least 1. This justifies omitting the parameter ζ. (It also gives a second reason for calling it the maximal "constant" approximation to the factor matrix of E. However, Conway's uses the word "constant" because the matrix entries are the "constants" 1 and 0 and not for this reason.)

Each entry in the matrix $\zeta^\flat \bullet \mathbf{L}_{max}(E,\zeta)$ is a sum of linear approximations to the factors of E : the term $\zeta.a$, where $a \in T$, is a summand of the (i,j)th entry if the approximating event represented by a, i.e. $\zeta.a$, is at most the factor $i \backslash j$.

The theorem gives a recipe for determining the maximal approximation to E by the events represented by ζ in three steps: First determine the matrix $\mathbf{C}_{max}.E$. Next determine $\zeta^\flat \bullet \mathbf{L}_{max}(E,\zeta)$. That is, for each factor F of E , determine the maximal linear approximation to F by ζ; enter the result in the matrix $\zeta^\flat \bullet \mathbf{L}_{max}(E,\zeta)$ in positions corresponding to occurrences of F in the factor matrix of E. Finally, compute the (l,r)th entry of $\mathbf{C}_{max}.E \otimes (\zeta^\flat \bullet \mathbf{L}_{max}(E,\zeta))^*$. This is the maximal approximation to E by the set of events represented by the function ζ. Since $(\zeta^\flat \bullet \mathbf{L}_{max}(E,\zeta))^*$ and $\zeta^\flat \bullet (\mathbf{L}_{max}(E,\zeta))^*$ are equal the order in which the two operators (composition and star) are applied is irrelevant to the result. Note, however, that if the latter formula is used, it will be necessary to exploit theorem 26 in order to evalute the approximations: the entries of $(\mathbf{L}_{max}(E,\zeta))^*$ will typically be evaluated as regular expressions and the theorem is needed to convert the expressions to approximations.

Our exposition is substantially longer than Conway's because he omits any mention whatsoever of the algebraic properties that are fundamental to the proof of theorem 133 and its exploitation when calculating approximations. The properties are a consequence of the fact that $\zeta^\flat$ is the lower adjoint in a Galois connection between the subset ordering on languages and the partial ordering in the algebra of the event E, but he does not even define $\zeta^\flat$, let alone show that it distributes through supremum, matrix multiplication and the star operator.

6 Least Approximating Functions and the Factor Graph

Rather than exploit Conway's maximal constant+linear approximating functions, we introduce in this section the *least* constant+linear approximating function. Whereas, the Generalised Approximation Theorem is valid for an event E in an arbitrary regular algebra, least approximating functions do not necessarily exist and, so, we are forced to restrict the class of events that we consider to the regular languages. Even then it is not obvious that such functions exist. It depends on the fact that the maximal constant approximation function encodes the subset ordering on left factors and this subset ordering has a unique minimal "starth root", i.e. a unique minimal reflexive, transitive reduction.

The next section defines the notion of a “starth root” and gives examples of where minimal starth roots do not exist. Section 6.2 then introduces the notion of “definiteness”, which is the key to establishing the existence of minimal approximating function.

Note that we have been careful to use the term “constant approximating *function*”. It is important to note that our minimal function and Conway’s maximal function yield the same approximation when the functions are applied to their arguments.

6.1 Starth Root

Definition 134 (Starth Root) Suppose U is an event of a regular algebra. A *starth root* of U is any event V that satisfies $V^* = U^*$; it is *minimal* if no smaller event has this property. It is *least* if it is at most all starth roots. Formally, V is *a minimal starth root* of U if

$$V^* = U^* \;\wedge\; \langle \forall W : W \preceq V \wedge W^* = U^* : W = V \rangle$$

and V is *the least starth root* of U if

$$V^* = U^* \;\wedge\; \langle \forall W : W^* = U^* : V \preceq W \rangle \quad .$$

□

Note the use of the indefinite article (“a”) for minimal starth roots, and the definite article (“the”) for least starth roots. It is easily shown that a least starth root is unique, which justifies the use of “the”. Moreover, it is clear that the least starth root of U is a minimal starth root.

By definition, every event is a starth root of itself. So starth roots exist for every event of a regular algebra. In a *free* regular algebra, it is the case that every event has a unique, minimal starth root [Brz67] but in general this is not always the case. Even when minimal starth roots exist, uniqueness is not guaranteed. For example, a minimal starth root of a relation is called a *transitive reduction* of the relation but there may be several different transitive reductions of a given relation. A specific example is the relation $\{(1,2),(2,3),(3,1)\}$ on $\{1,2,3\}$. It is a minimal starth root of itself but so also is $\{(2,1),(3,2),(1,3)\}$. (Confusingly, the literature sometimes refers to *the* transitive reduction of a relation/graph even though in a case like this one an arbitrary choice must be made.) See example 139 for further explanation. Unique minimal starth roots (i.e. unique transitive reductions) are guaranteed to exist for well-founded relations on a finite set (equivalently, relations that can be represented by a finite, acyclic graph).

The fact that, nevertheless, the factor matrix of a regular language has a unique minimal starth root was first proved in [Bac75] and later, using an improved argument, in [BL77]. For reasons that will be explained shortly, the name *factor graph* was given to this matrix.

(Note: Conway [Con71, p55] includes minimality in his definition of a starth root but restricts the discussion to regular languages. We prefer to separate out the minimality requirement.)

Theorem 137 below is the basic starting point for the construction of starth roots. Effectively, the theorem is the basis for Brzozowski's [Brz67] proof of the existence of minimal starth roots of regular languages but we state it more abstractly here because minimal starth roots do not necessarily exist in matrix algebras. Example 140, immediately following theorem 137, illustrates the theorem using the example of the transitive reduction of a relation.

First, we need a couple of (well-known) lemmas. Recall that we use "$\cup$" and "$\cap$" to denote the supremum and infimum operators in a powerset algebra, and the symbols "$\dot{\cup}$" and "$\dot{\cap}$" to denote their pointwise extension to matrices. In this section, we use the symbol "$\neg$" to denote the complement of a set. It too can be extended pointwise to matrices, and the extended operators enjoy all the properties of the set operators. So, although we don't use the dotted notation "$\dot{\cup}$", "$\dot{\cap}$", etc., all the lemmas in this section are valid for matrices based on a powerset regular algebra, in particular matrices of languages.

Lemma 135 Let X be an event in a powerset regular algebra with unit 1. Then

$$X^* = (X \cap \neg 1)^* \ .$$

Proof

$$\begin{array}{ll}
 & X^* = (X\cap\neg 1)^* \\
\Leftarrow & \{ \quad X \supseteq X\cap\neg 1 \text{, monotonicity of star} \quad \} \\
 & X^* \subseteq (X\cap\neg 1)^* \\
= & \{ \quad {}^* \text{ is a closure operator} \quad \} \\
 & X \subseteq (X\cap\neg 1)^* \\
\Leftarrow & \{ \quad 1\cup Y \subseteq Y^* \text{ with } Y := X\cap\neg 1 \quad \} \\
 & X \subseteq 1\cup(X\cap\neg 1) \\
= & \{ \quad \text{absorption rule} \quad \} \\
 & X \subseteq 1\cup X
\end{array}$$

$=\qquad \{\quad \text{set calculus} \quad\}$

$\qquad \text{true} \quad .$

□

Lemma 136 Let X and Y be events in a powerset regular algebra with unit 1. Then

$$X^* \subseteq Y^* \;\equiv\; (X \cap \neg 1)^+ \subseteq (Y \cap \neg 1)^+ \quad ,$$

$$X^* = Y^* \;\equiv\; (X \cap \neg 1)^+ = (Y \cap \neg 1)^+ \quad .$$

Proof First,

$\qquad (X \cap \neg 1)^+ \subseteq (Y \cap \neg 1)^+$

$=\qquad \{\quad {}^+ \text{ is a closure operator} \quad\}$

$\qquad X \cap \neg 1 \subseteq (Y \cap \neg 1)^+$

$=\qquad \{\quad \text{set calculus} \quad\}$

$\qquad X \subseteq (Y \cap \neg 1)^+ \cup 1$

$=\qquad \{\quad \text{for all } Z,\ Z^+ \cup 1 = Z^* \text{ with } Z := Y \cap \neg 1 ,$

$\qquad\qquad \text{lemma 135 with } X := Y \quad\}$

$\qquad X \subseteq Y^*$

$=\qquad \{\quad {}^* \text{ is a closure operator} \quad\}$

$\qquad X^* \subseteq Y^* \quad .$

The second property follows immediately from the anti-symmetry of set union.
□

Theorem 137 (Least Starth Root) Let A be an event in a powerset regular algebra with unit 1. Suppose $B = A \cap \neg 1$ and suppose $A^* = (B \cap \neg(B \cdot B^+))^*$. Suppose X is an event such that $A^* = X^*$. Then

$$B \cap \neg(B \cdot B^+) \subseteq X \quad .$$

That is, if $A \cap \neg 1 \cap \neg((A \cap \neg 1) \cdot (A \cap \neg 1)^+)$ is a starth root of A, it is the least starth root of A.

Proof For brevity, let $C = B \cap \neg(B \cdot B^+)$ and $Y = X \cap \neg 1$. By applying lemma 135 and including the two assumptions, we have

$$A^* = B^* = C^* = X^* = Y^* \quad .$$

Next we note that

$$
\begin{array}{ll}
 & C \\
= & \{ \quad \text{definition of } C \text{ and } B \quad \} \\
 & A\cap\neg 1\cap\neg(B\cdot B^+) \\
= & \{ \quad \text{idempotency and symmetry of intersection} \quad \} \\
 & (A\cap\neg 1\cap\neg(B\cdot B^+))\cap\neg 1 \\
= & \{ \quad \text{definition of } C \text{ and } B \quad \} \\
 & C\cap\neg 1 \;\; .
\end{array}
$$

It follows that we can apply lemma 136 with $X,Y{:=}A,C$ and $X,Y{:=}C,X$ to deduce that

$$B^+ = C^+ = Y^+ \;\; .$$

We can now proceed with the calculation.

$$
\begin{array}{ll}
 & B\cap\neg(B\cdot B^+)\subseteq X \\
= & \{ \quad B\cap\neg(B\cdot B^+) = C = C\cap C^+ = C\cap Y^+ \quad \} \\
 & B\cap\neg(B\cdot B^+)\cap Y^+ \subseteq X \\
= & \{ \quad \text{set calculus} \quad \} \\
 & B\cap Y^+ \subseteq X\cup B\cdot B^+ \\
\Leftarrow & \{ \quad B\cap Y^+\subseteq Y^+ \quad \} \\
 & Y^+ \subseteq X\cup B\cdot B^+ \\
\Leftarrow & \{ \quad Y^+ = Y\cup Y\cdot Y^+ \quad \} \\
 & Y\subseteq X \;\wedge\; Y\cdot Y^+ \subseteq B\cdot B^+ \\
= & \{ \quad Y = X\cap\neg 1 \quad \} \\
 & Y\cdot Y^+ \subseteq B\cdot B^+ \\
= & \{ \quad X\cdot X^+ = X^+\cdot X^+ \text{ for all } X \\
 & \quad\quad \text{(well-known property, simple proof left to reader)} \quad \} \\
 & Y^+\cdot Y^+ \subseteq B^+\cdot B^+ \\
= & \{ \quad B^+ = Y^+ : \text{ see above} \quad \} \\
 & \text{true} \;\; .
\end{array}
$$

□

Theorem 137 postulates a candidate for a least starth root. In some cases, the candidate is indeed a least starth root, as illustrated by example 138 below, but this is not always the case, as illustrated by examples 139 and 140. Fortunately, the candidate is indeed a starth root in the case relevant to the current discussion: when A is Conway's maximal approximation function. See the subsections below.

Note that both examples 138 and 139 rely on the fact that Bool is the carrier set of a powerset algebra, and the homogenous binary relations on a set are the carrier set of a matrix algebra over Bool. See the discussion following theorem 9. In the case of example 138, the "matrices" have infinite dimension.

Example 138 Consider the at-most relation on integers. This is normally denoted by the symbol "$\leq$" but it is more convenient here to use the symbol atmost. The at-most relation is, of course, reflexive and transitive. That is, $\mathsf{atmost} = \mathsf{atmost}^*$. Instantiating the variable A in theorem 137 with atmost, the relation B is the less-than relation. This normally denoted by the symbol "$<$" but let us write less instead. The reader may easily verify that the relation $\mathsf{less} \cap \neg(\mathsf{less} \circ \mathsf{less}^+)$ is the predecessor relation, pred, given by, for all integers i and j,

$$i \ \mathsf{pred} \ j \equiv i+1 = j \quad .$$

The theorem states that, if the predecessor relation is a starth root of the at-most relation, then it is the least starth root of that relation. And, indeed, $\mathsf{pred}^* = \mathsf{atmost}$. So, we conclude that

$$\langle \forall R : R^* = \mathsf{atmost} : \mathsf{pred} \subseteq R \rangle \quad .$$

□

Example 139 Suppose we consider the universal relation on the set $\{1,2,3\}$. Fig. 3(a) depicts the relation as a graph. Figs. 3(b) and (c) depict starth roots of the relation; they are both minimal but are distinct.

Denoting the universal relation by $\top\!\!\top$ on $\{1,2,3\}$ and the identity relation on $\{1,2,3\}$ by I, the relation $\top\!\!\top \cap \neg I \cap \neg((\top\!\!\top \cap \neg I) \circ (\top\!\!\top \cap \neg I)^+)$ is the empty relation and the reflexive-transitive closure of the empty relation is the identity relation. Thus, it is not a starth root of the universal relation.

(The reader may also wish to explore the cases of $\{1\}$ and $\{1,2\}$. In both cases, the universal relation on the set does have a least starth root. What this is is predicted by the theorem in the first case but not in the second.)

On the other hand, it can be shown that if R is a homogeneous binary relation on a finite set and the graph of the relation is acyclic, then $R = R \cap \neg I$ and $R \cap \neg(R \circ R^+)$ is the unique, minimal starth root of R. Although we do not go into the details in this paper, an acyclic relation on a finite set is an example of what we call a "definite" event

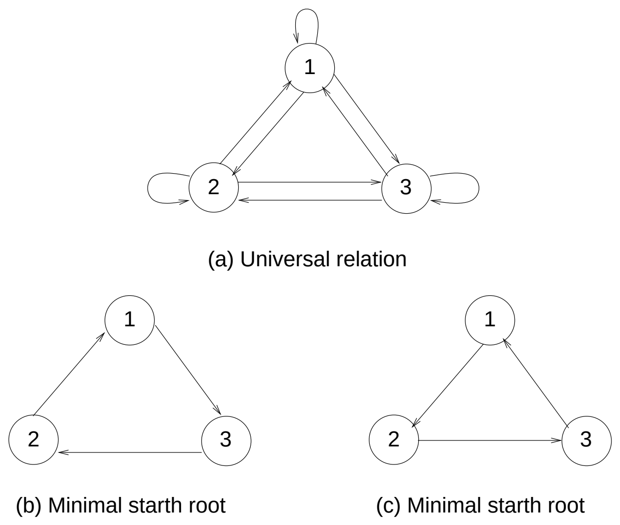


Figure 3: Distinct minimal starth roots of the universal relation

in a regular algebra. (See definition 141.)
□

Example 140 (Running Example: Modulo Addition) The factor matrix of {0} in the powerset regular algebra with underlying monoid $\mathbb{Z}_3$ was shown in example 88. It does *not* have a unique minimal starth root. This is because both 1 and 2 are generators of the group. Taking 1 as the generator, one starth root is shown in the conventional way as a two-dimensional array below:

$$\begin{bmatrix} \{1\} & \emptyset & \emptyset & \emptyset & \emptyset \\ \{0\} & \emptyset & \{1\} & \emptyset & \emptyset \\ \{0\} & \emptyset & \emptyset & \{1\} & \emptyset \\ \{0\} & \{1\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \{0\} & \{0\} & \{0\} & \emptyset \end{bmatrix}$$

Taking 2 as the generator, we get a different starth root:

$$\begin{bmatrix} \{2\} & \emptyset & \emptyset & \emptyset & \emptyset \\ \{0\} & \emptyset & \emptyset & \{2\} & \emptyset \\ \{0\} & \{2\} & \emptyset & \emptyset & \emptyset \\ \{0\} & \emptyset & \{2\} & \emptyset & \emptyset \\ \emptyset & \{0\} & \{0\} & \{0\} & \emptyset \end{bmatrix}$$

We could, of course, have made the example yet simpler since the factor matrix of $\neg\emptyset$ (i.e. {0,1,2}) also does not have a unique starth root: the factor matrix has exactly one entry which is $\neg\emptyset$ itself, and this has two minimal starth roots, namely the matrix with single entry $\{1\}$ and the matrix with single entry $\{2\}$. Exploiting this fact gives two more minimal starth roots of the factor matrix of $\{0\}$: interchange the top-left entries in the matrices above.

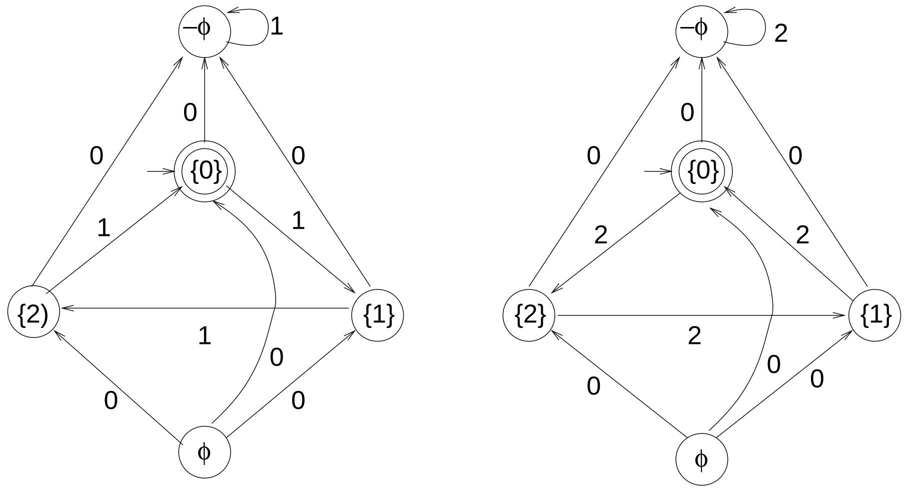


Figure 4: Another example of distinct minimal starth roots

Fig. 4 displays the two starth roots above as labelled graphs. (It is much easier for human beings to check that the reflexive-transitive closure of each of the graphs displayed in fig. 4 is the factor matrix than it is to check that the reflexive-transitive closure of the two-dimensional array is the factor matrix.)
□

6.2 Definiteness

Formally, our calculations rely on the fact that certain (matrix) equations have unique solutions. We could formulate the relevant properties in terms of Salomaa's "empty word property" [Sal66], but we prefer the algebraic formulation of "definiteness" introduced in [BC75]. In order to facilitate later discussion, we distinguish between "left-" and "right-"definite.

Definition 141 (Left- and Right-Definite) Let A be an event in a regular algebra. Then A is said to be *left-definite* if, for all events X,

$$X \preceq A \cdot X \;\equiv\; X \preceq 0$$

and A is said to be *right-definite* if, for all events X,

$$X \preceq X \cdot A \;\equiv\; X \preceq 0 \quad .$$

Finally, A is said to be *definite* if it is both left- and right-definite.
□

Note that any of the occurrences of "$\preceq$" in definition 141 can be replaced by equality. (Of course, $X\preceq 0$ and $X=0$ are equivalent. Replacing, eg, $X\preceq A{\cdot}X$ by $X=A{\cdot}X$ is an easy calculation: the hint is multiply both sides by A^*.)

The importance of the concept of definiteness is what we have called the *unique extension property* (UEP) of regular algebra.

Theorem 142 (UEP of regular algebra) Suppose A is a left-definite event in a regular algebra. Then, for all events X and B,

$$X = (A{\cdot}X)+B \quad\equiv\quad X = A^*\cdot B \enspace .$$

Dually, if A is a right-definite event in a regular algebra. Then, for all events X and B,

$$X = (X{\cdot}A)+B \quad\equiv\quad X = B\cdot A^* \enspace .$$

□

Theorem 142 was postulated as an axiom of regular algebra in [BC75]. Here, a proof is needed because the star operator is not a primitive but defined in terms of least fixed points. A proof can be found in [DBvdW97, section 7]. Note that [DBvdW97] uses the terminology "well-founded" rather than "right definite" in order to fit with the standard terminology of the principle application considered in the paper.

At this point, I would like to be able to claim that a suitably chosen submatrix of $\zeta^{\sharp}\bullet|\overline{E}|$ is both left- and right-definite, irrespective of the regular algebra of events. But I have failed to find a proof. (I have not yet looked for counter-examples.) Such a theorem can, however, be formulated if E is a *regular* language. See theorem 147. As usual, we need some preliminary lemmas.

Lemma 143 If A is left-definite and $B\preceq A$, then B is left-definite. Dually, if A is right-definite and $B\preceq A$, then B is right-definite.

Proof Suppose A is left-definite and $B\preceq A$. Then, for all X,

$$\begin{array}{ll}
 & X\preceq B{\cdot}X \\
\Rightarrow & \qquad\{\qquad B\preceq A\text{, monotonicity}\qquad\} \\
 & X\preceq A{\cdot}X \\
= & \qquad\{\qquad A\text{ is left-definite}\qquad\} \\
 & X\preceq 0 \\
\Rightarrow & \qquad\{\qquad 0\preceq B{\cdot}X\qquad\} \\
 & X\preceq B{\cdot}X \enspace .
\end{array}$$

□

This is the point at which we must specialise the discussion to regular languages. Corollary 145 states that, for matrices of languages, definiteness is dependent only on the “constant” part of the matrix (i.e. that part of the matrix that is a subset of $\mathsf{Mat}.\{\varepsilon\}$).

Lemma 144 Suppose $\mathbf{A}$ is a square matrix of languages of dimension $I{\times}I$, and $\mathbf{X}$ is a vector of languages such that $\mathbf{X}\dot{\subseteq}\mathbf{A}\otimes\mathbf{X}$ and $\mathbf{X}\neq\mathbf{0}$. Then there is a vector $\mathbf{Y}$ such that

$$\mathbf{Y}\neq\mathbf{0} \;\wedge\; \mathbf{Y}\;\dot{\subseteq}\;\mathsf{Mat}.\{\varepsilon\} \;\wedge\; \mathbf{Y}\;\dot{\subseteq}\;(\mathbf{A}\dot{\cap}\mathsf{Mat}.\{\varepsilon\})\otimes\mathbf{Y} \;\;.$$

Proof Consider $\langle\cup i :: \mathbf{X}.i\rangle$. This set is non-empty since $\mathbf{X}\neq\mathbf{0}$. Suppose u is a shortest word in the set. Define vector $\mathbf{Y}$ so that, for all y and i,

$$y\in\mathbf{Y}.i \;\equiv\; y=\varepsilon\wedge u\in\mathbf{X}.i \;\;.$$

Clearly $\mathbf{Y}\neq\mathbf{0}\wedge\mathbf{Y}\;\dot{\subseteq}\;\mathsf{Mat}.\{\varepsilon\}$. Moreover,

$$\begin{array}{ll}
 & \mathbf{Y}\;\dot{\subseteq}\;(\mathbf{A}\dot{\cap}\mathsf{Mat}.\{\varepsilon\})\otimes\mathbf{Y} \\
= & \quad\{\quad \text{definition of } \mathbf{Y}\text{, definition of } \dot{\subseteq} \text{ and one-point rule} \quad\} \\
 & \langle\forall i : u\in\mathbf{X}.i : \varepsilon\in((\mathbf{A}\dot{\cap}\mathsf{Mat}.\{\varepsilon\})\otimes\mathbf{Y}).i\rangle
\end{array}$$

But,

$$\begin{array}{ll}
 & \varepsilon\in((\mathbf{A}\dot{\cap}\mathsf{Mat}.\{\varepsilon\})\otimes\mathbf{Y}).i \\
= & \quad\{\quad \langle\forall x,y :: \varepsilon=xy \;\equiv\; \varepsilon=x\wedge\varepsilon=y\rangle \\
 & \quad\quad\; \text{definition of matrix product and } \dot{\cap} \quad\} \\
 & \langle\exists j : \varepsilon\in i\mathbf{A}j : \varepsilon\in\mathbf{Y}.j\rangle \\
= & \quad\{\quad \varepsilon\in\mathbf{Y}.j \;\equiv\; u\in\mathbf{X}.j \quad\} \\
 & \langle\exists j : \varepsilon\in i\mathbf{A}j : u\in\mathbf{X}.j\rangle \\
\Leftarrow & \quad\{\quad u \text{ is a shortest word in } \langle\cup i :: \mathbf{X}.i\rangle \quad\} \\
 & \langle\exists j :: u\in(i\mathbf{A}j)\cdot\mathbf{X}.j\rangle \\
\Leftarrow & \quad\{\quad \mathbf{X}\dot{\subseteq}\mathbf{A}\otimes\mathbf{X} \quad\} \\
 & u\in\mathbf{X}.i \;\;.
\end{array}$$

Combining the two calculations, we conclude that $\mathbf{Y}\;\dot{\subseteq}\;(\mathbf{A}\dot{\cap}\mathsf{Mat}.\{\varepsilon\})\otimes\mathbf{Y}$.

□

Corollary 145 If $\mathbf{A}$ is a square matrix of languages then

$$\mathbf{A} \text{ is left-definite } \equiv \mathbf{A} \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\} \text{ is left-definite}$$

and

$$\mathbf{A} \text{ is right-definite } \equiv \mathbf{A} \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\} \text{ is right-definite.}$$

Proof We prove the first equivalence. The proof of the second is symmetric.

By applying lemma 143, we have that if $\mathbf{A}$ is left-definite then $\mathbf{A} \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\}$ is left-definite. But the contrapositive of this statement is an immediate corollary of lemma 144: if $\mathbf{A}$ is not left-definite then $\mathbf{A} \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\}$ is not left -definite.
□

6.3 Definiteness of the Maximal Approximation Function

In this section, we establish the definiteness of maximal approximation functions. See corollary 148.

A possibly confusing complication is that the alphabets of the given regular language E and the "set" of approximations ζ are typically different. This means that E and any approximating "function" are elements of different algebras, which formally have different units. Conventional accounts would silently overload notation so that the difference is hidden. We also overload notation in the same way but observe the overloading at the appropriate point in the discussion.

Lemma 146 Suppose E is a regular language. Let $\mathbf{I}$ denote the identity matrix of dimension $\mathcal{L}.\mathsf{E} \times \mathcal{L}.\mathsf{E}$ (where, as usual, $\mathcal{L}.\mathsf{E}$ is the set of left factors of E). Then the matrix $\mathbf{C}_{max}.\mathsf{E} \,\dot{\cap}\, \neg\mathbf{I}$ is definite.

Proof We prove that $\mathbf{C}_{max}.\mathsf{E} \,\dot{\cap}\, \neg\mathbf{I}$ is left-definite. The proof that it is also right-definite is symmetric.

Suppose $\mathbf{C}_{max}.\mathsf{E} \,\dot{\cap}\, \neg\mathbf{I}$ is not left-definite. Applying lemma 144 (and noting that $\mathbf{C}_{max}.\mathsf{E} = \mathbf{C}_{max}.\mathsf{E} \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\}$), there is a vector $\mathbf{Y}$ such that

$$\mathbf{Y} \neq \mathbf{0} \quad \wedge \quad \mathbf{Y} \,\dot{\subseteq}\, \mathsf{Mat}.\{\varepsilon\} \quad \wedge \quad \mathbf{Y} \,\dot{\subseteq}\, (\mathbf{C}_{max}.\mathsf{E} \,\dot{\cap}\, \neg\mathbf{I}) \otimes \mathbf{Y} \quad .$$

From the first two conjuncts, we infer that there is at least one left factor i_0, say, of E such that $\varepsilon \in \mathbf{Y}.i_0$. Now we exploit the third conjunct. We have, for all left factors i,

$$\begin{array}{ll} & \varepsilon \in \mathbf{Y}.i \\ \Rightarrow & \{ \qquad \mathbf{Y} \,\dot{\subseteq}\, (\mathbf{C}_{max}.\mathsf{E} \,\dot{\cap}\, \neg\mathbf{I}) \otimes \mathbf{Y} \,, \end{array}$$

$$
\begin{array}{ll}
 & \quad \text{definition of } \mathsf{Mat}.\{\varepsilon\} \dot{\cap} \neg\mathbf{I},\ i\,|\overline{E}|\,j = i\backslash j \quad \} \\
 & \langle \exists j : i \neq j : \varepsilon \in i\backslash j \wedge \varepsilon \in Y.j \rangle \\
= & \quad \{ \quad \varepsilon \in i\backslash j \equiv i \subseteq j \quad \} \\
 & \langle \exists j : i \neq j : i \subseteq j \wedge \varepsilon \in Y.j \rangle \quad .
\end{array}
$$

It follows that we can construct a sequence of left factors i_k, beginning with i_0, such that $i_k \subseteq i_{k+1}$ and $i_k \neq i_{k+1}$. If E is regular, it has only a finite number of left factors and so the sequence must eventually repeat itself. That is, there is a subsequence $i_m, \ldots, i_{m+p}$ of left factors such that $0 < p$, $i_{m+k} \neq i_{m+k+1}$ whenever $0 \leq k < p$, and $i_m \subseteq i_{m+1} \subseteq \ldots \subseteq i_{m+p} \subseteq i_m$. It follows by anti-symmetry and transitivity of the subset relation that all events of the sequence are equal, which contradicts consecutive events being different.
□

Theorem 147 Suppose E is a regular language. Suppose ζ of type $S \leftarrow T$ is a function encoding a set of events in regular algebra $\mathcal{S}$. (Note that the alphabet T is typically different from the alphabet of E.) Let $\mathbf{I}$ denote the identity matrix of dimension $\mathcal{L}.E \times \mathcal{L}.E$ in the matrix algebra with underlying algebra $\mathcal{S}$. Then the matrix $\mathbf{C}_{max}(E,\zeta) \dot{\cap} \neg\mathbf{I}$ is definite.

Proof We first prove that if $\mathbf{C}_{max}(E,\zeta) \dot{\cap} \neg\mathbf{I}$ is not left-definite then $\mathbf{C}_{max}.E \dot{\cap} \neg\mathbf{I}$ is not left-definite. (NB: the notation "$\mathbf{I}$" is overloaded here. See below.)

Suppose $\mathbf{X}$ is a vector such that

$$\mathbf{X} \ \dot{\subseteq}\ (\mathbf{C}_{max}(E,\zeta) \dot{\cap} \neg\mathbf{I}) \otimes \mathbf{X} \quad \wedge \quad \mathbf{X} \neq \mathbf{0} \quad .$$

Note that the dimension of $\zeta^\sharp \bullet |\overline{E}|$ means that $\mathbf{X}$ is necessarily indexed by left factors of E. However, the alphabet of E is typically different from the alphabet of events in $\mathbf{X}$. Applying lemma 144 (noting that $\mathbf{C}_{max}(E,\zeta) \dot{\subseteq} \mathsf{Mat}.\{\varepsilon\}$), there is a vector $\mathbf{Y}$ such that

$$\mathbf{Y} \neq \mathbf{0} \ \wedge\ \mathbf{Y} \dot{\subseteq} \mathsf{Mat}.\{\varepsilon\} \ \wedge\ \mathbf{Y} \ \dot{\subseteq}\ (\mathbf{C}_{max}(E,\zeta) \dot{\cap} \neg\mathbf{I}) \otimes \mathbf{Y} \quad .$$

We now want to exhibit a vector $\mathbf{Z}$ such that

$$\mathbf{Z} \neq \mathbf{0} \ \wedge\ \mathbf{Z} \dot{\subseteq} \mathsf{Mat}.\{\varepsilon\} \ \wedge\ \mathbf{Z} \ \dot{\subseteq}\ (\mathbf{C}_{max}.E \dot{\cap} \neg\mathbf{I}) \otimes \mathbf{Z} \quad .$$

Formally, we are precluded from using $\mathbf{Y}$ directly since the entries in $\mathbf{C}_{max}(E,\zeta)$ and the entries in $\mathbf{C}_{max}.E$ are languages over different alphabets[5]. However, the empty word

[5]A simple trick like taking the union of the alphabets doesn't work because the factor matrix of a language depends on the alphabet.

is polymorphic —the occurrences of " ε " in the specification of $\mathbf{Z}$ has a different type to the occurrence of " ε " in the specification of $\mathbf{Y}$— , and so too is the empty set —the two occurrences of " $\mathbf{0}$ " above have different type— . That is, we can take $\mathbf{Z}$ to be identical to $\mathbf{Y}$ except for an appropriate change of type of the empty word (and the empty set). Then, clearly, $\mathbf{Z}\neq\mathbf{0}\wedge\mathbf{Z}\ \dot{\subseteq}\ \mathsf{Mat}.\{\varepsilon\}$. Moreover, applying lemma 131, we have, for all left factors i and j of E,

$$\varepsilon\in\mathsf{i}\ (\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\ \mathsf{j}\ \ \equiv\ \ \varepsilon\in\mathsf{i}\backslash\mathsf{j}\ \ .$$

(Here the two occurrences of " ε " are of different type.) It follows that $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)$ and $\mathbf{C}_{\mathsf{max}}.\mathsf{E}$ are identical matrices except for an appropriate change of type of the empty word. The same is thus true of $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\,\dot{\cap}\,\neg\mathbf{I}$ and $\mathbf{C}_{\mathsf{max}}.\mathsf{E}\,\dot{\cap}\,\neg\mathbf{I}$. That is,

$$\mathbf{Z}\ \ \dot{\subseteq}\ \ (\mathbf{C}_{\mathsf{max}}.\mathsf{E}\,\dot{\cap}\,\neg\mathbf{I})\otimes\mathbf{Z}\ \ .$$

We have thus shown that $\mathbf{C}_{\mathsf{max}}.\mathsf{E}$ is not left-definite. As before, a symmetric argument shows that if ($\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\,\dot{\cap}\,\neg\mathbf{I}$) is not right-definite, $\mathbf{C}_{\mathsf{max}}.\mathsf{E}\,\dot{\cap}\,\neg\mathbf{I}$ is not right-definite.

Taking the contrapositive of both statements together with theorem 146, we obtain the theorem.
□

Corollary 148 If E is a regular language, then the matrices $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\,\dot{\cap}\,\neg\mathbf{I}$, $\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)$ and $(\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \dot{\cup}\ \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))\cap\neg\mathbf{I}$ are all definite.

Proof Theorem 147 states that $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\,\dot{\cap}\,\neg\mathbf{I}$ is definite, By corollary 145, $\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)$ is definite because $\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)\,\dot{\cap}\,\mathsf{Mat}.\{\varepsilon\}$ is $\mathbf{0}$ (the zero of product). Similarly, $(\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \dot{\cup}\ \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))$ is definite because its intersection with $\mathsf{Mat}.\{\varepsilon\}$ equals $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\,\dot{\cap}\,\neg\mathbf{I}$.
□

6.4 Reducing Maximal to Least

Recall that $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)$ denotes $(\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\,\dot{\cap}\,\mathsf{Mat}.\{\varepsilon\}$ and $\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)$ denotes $(\zeta^\sharp\bullet|\overline{\mathsf{E}}|)\,\dot{\cap}\,\mathsf{Mat}.\mathsf{T}$. We define $\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta)$ and $\mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta)$ as follows.

Definition 149 (Least Constant and Linear Approximating Functions) Suppose E is a regular language. Let

$$\mathbf{B}\ =\ (\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\ \dot{\cup}\ \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta))\cap\neg\mathbf{I}\ \ ,$$

$$\mathbf{D}=\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)\,\dot{\cap}\,\neg\mathbf{I}\ \ ,$$

$$\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta)=\mathbf{D}\,\dot{\cap}\,\neg(\mathbf{D}\otimes\mathbf{D}^{+})\quad\text{and}$$

$$\mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta) \;=\; \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta) \mathbin{\dot\cap} \neg(\mathbf{D}{\otimes}\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta)) \mathbin{\dot\cap} \neg(\mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta){\otimes}\mathbf{D}) \enspace .$$

The function that maps ζ to $\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta)$ is called the *minimal constant approximating function* for E by ζ, and the function that maps ζ to $\mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta)$ is called the *minimal linear approximating function* for E by ζ.
□

Our goal in this section is to prove that $\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta) \mathbin{\dot\cup} \mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta)$ is the least starth root of $\zeta^\sharp \bullet |\overline{\mathsf{E}}|$. We exploit theorem 137. Specifically, $\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta) \mathbin{\dot\cup} \mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta)$ is the least starth root of $\zeta^\sharp \bullet |\overline{\mathsf{E}}|$ if it is a starth root of $\zeta^\sharp \bullet |\overline{\mathsf{E}}|$ and

(150) $\quad \mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta) \mathbin{\dot\cup} \mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta) \;=\; \mathbf{B} \mathbin{\dot\cap} \neg(\mathbf{B} \otimes \mathbf{B}^+) \enspace .$

(See definition 149 for the definition of $\mathbf{B}$.) First, lemma 151 proves (150). Then theorem 154 shows that it is a starth root.

Note that $\mathbf{B}$ is definite. (Compare the definition of $\mathbf{B}$ with corollary 148.) This is crucial to our calculations.

Lemma 151 Let $\mathbf{B}$, $\mathbf{D}$, $\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta)$ and $\mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta)$ be as in definition 149. Then

$$\mathbf{B} \;=\; \mathbf{D} \mathbin{\dot\cup} \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta) \enspace ,$$

$$\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta) = \mathbf{D} \mathbin{\dot\cap} \neg(\mathbf{B} \otimes \mathbf{B}^+) \enspace ,$$

$$\mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta) = \mathbf{L}_{\mathsf{max}}(\mathsf{E},\zeta) \mathbin{\dot\cap} \neg(\mathbf{B} \otimes \mathbf{B}^+) \quad \text{, and}$$

$$\mathbf{C}_{\mathsf{min}}(\mathsf{E},\zeta) \mathbin{\dot\cup} \mathbf{L}_{\mathsf{min}}(\mathsf{E},\zeta) \;=\; \mathbf{B} \mathbin{\dot\cap} \neg(\mathbf{B} \otimes \mathbf{B}^+) \enspace .$$

Proof For brevity, we omit the parameters of $\mathbf{C}_{\mathsf{max}}$, $\mathbf{C}_{\mathsf{min}}$, $\mathbf{L}_{\mathsf{min}}$ and $\mathbf{L}_{\mathsf{max}}$. For example, throughout the following calculation, $\mathbf{C}_{\mathsf{max}}$ denotes $\mathbf{C}_{\mathsf{max}}(\mathsf{E},\zeta)$. We also let $\mathbf{B}$ and $\mathbf{D}$ be the matrices defined in definition 149.

The calculations below exploit “length considerations”: entries in $\mathbf{C}_{\mathsf{max}}$ and $\mathbf{C}_{\mathsf{min}}$ are words of length 0 and entries in $\mathbf{L}_{\mathsf{max}}$ and $\mathbf{L}_{\mathsf{min}}$ are words of length 1. Formally, if $\mathbf{L}$ is a linear matrix (in particular, $\mathbf{L}_{\mathsf{max}}$ or $\mathbf{L}_{\mathsf{min}}$),

$$\mathbf{L} = \mathbf{L} \mathbin{\dot\cap} \mathsf{Mat.T}$$

and if $\mathbf{C}$ is a constant matrix (in particular, $\mathbf{C}_{\mathsf{max}}$ or $\mathbf{C}_{\mathsf{min}}$),

$$\mathbf{C} = \mathbf{C} \mathbin{\dot\cap} \mathsf{Mat.}\{\varepsilon\} \enspace .$$

We exploit these properties by combining them with properties of $\mathsf{Mat.T}$ and $\mathsf{Mat.}\{\varepsilon\}$. Examples are:

$$\mathbf{0} \;=\; \mathsf{Mat.T} \mathbin{\dot\cap} \mathsf{Mat.}\{\varepsilon\} \;=\; \mathsf{Mat.T} \mathbin{\dot\cap} \mathsf{Mat.T} \otimes \mathsf{Mat.T}$$

(where $\mathbf{0}$ denotes the zero matrix) and, for linear matrix $\mathbf{L}$ and arbitrary matrix $\mathbf{X}$.

(152) $\quad \mathbf{L} \dot{\cap} \neg \mathbf{X} = \mathbf{L} \dot{\cap} \neg (\mathbf{X} \dot{\cap} \mathsf{Mat.T})$

and, for constant matrix $\mathbf{C}$

(153) $\quad \mathbf{C} \dot{\cap} \neg \mathbf{X} = \mathbf{C} \dot{\cap} \neg (\mathbf{X} \dot{\cap} \mathsf{Mat.}\{\varepsilon\})$.

Equations (152) and (153) are instances of the general property (extended pointwise to matrices) that, for all x, y and z such that $y \subseteq x$,

$$y \cap \neg z = y \cap \neg (z \cap x) \ .$$

This is proved by the following simple calculation.

$$
\begin{array}{ll}
 & y \cap \neg (z \cap x) \\
= & \{ \quad \text{distributivity of negation over intersection} \\
 & \quad\;\; \text{and assumption: } y \subseteq x \quad \} \\
 & y \cap x \cap (\neg z \cup \neg x) \\
= & \{ \quad \text{distributivity of intersection over union} \quad \} \\
 & y \ \cap \ ((x \cap \neg z) \cup (x \cap \neg x)) \\
= & \{ \quad \text{for all } x ,\ x \cap \neg x = \emptyset \quad \} \\
 & y \cap x \cap \neg z \\
= & \{ \quad \text{assumption: } y \dot{\subseteq} x \quad \} \\
 & y \cap \neg z \ .
\end{array}
$$

Now we can proceed to establish the lemma. For the first equation, we have:

$$
\begin{array}{ll}
 & \mathbf{B} \\
= & \{ \quad \text{definition} \quad \} \\
 & (\mathbf{C}_{max} \dot{\cup} \mathbf{L}_{max}) \dot{\cap} \neg \mathbf{I} \\
= & \{ \quad \text{distributivity} \quad \} \\
 & (\mathbf{C}_{max} \dot{\cap} \neg \mathbf{I}) \dot{\cup} (\mathbf{L}_{max} \dot{\cap} \neg \mathbf{I}) \\
= & \{ \quad \text{definition of } \mathbf{D} \text{: definition 149} \quad \} \\
 & \mathbf{D} \dot{\cup} (\mathbf{L}_{max} \dot{\cap} \neg \mathbf{I}) \\
= & \{ \quad (152) \text{ with } \mathbf{X} := \mathbf{I} ,\ \mathbf{I} \dot{\cap} \mathsf{Mat.T} = \mathbf{0} \quad \} \\
 & \mathbf{D} \dot{\cup} \mathbf{L}_{max} \ .
\end{array}
$$

We now give a relatively detailed proof of the second equation so that the reader can see how length considerations are used. First, we have:

$$\mathbf{C}_{min} \;=\; \mathbf{D} \dot\cap \neg(\mathbf{B} \otimes \mathbf{B}^{+})$$

$=$ $\{$ definition of $\mathbf{C}_{min}$ and $\mathbf{B} = \mathbf{D} \,\dot\cup\, \mathbf{L}_{max}$ (see above) $\}$

$$\mathbf{D} \,\dot\cap\, \neg(\mathbf{D} \otimes \mathbf{D}^{+}) \;=\; \mathbf{D} \dot\cap \neg(\mathbf{B} \otimes \mathbf{B}^{+})$$

$=$ $\{$ $\mathbf{D}$ is a constant matrix, (153) $\}$

$$\mathbf{D} \,\dot\cap\, \neg(\mathbf{D} \otimes \mathbf{D}^{+}) \;=\; \mathbf{D} \dot\cap \neg(\mathbf{B} \otimes \mathbf{B}^{+} \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\})$$

$\Leftarrow$ $\{$ Leibniz $\}$

$$\mathbf{D} \otimes \mathbf{D}^{+} \;=\; \mathbf{B} \otimes \mathbf{B}^{+} \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\} \;\;.$$

We now prove the above equation.

$$\mathbf{B} \otimes \mathbf{B}^{+} \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\}$$

$=$ $\{$ $\mathbf{B} = \mathbf{D} \,\dot\cup\, \mathbf{L}_{max}$ (see above) $\}$

$$(\mathbf{D} \,\dot\cup\, \mathbf{L}_{max}) \otimes (\mathbf{D} \,\dot\cup\, \mathbf{L}_{max})^{+} \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\}$$

$=$ $\{$ distributivity of matrix product over union,
length considerations: specifically $\mathbf{L}_{max} \dot\cap \mathsf{Mat}.\{\varepsilon\} = \mathbf{0}$ $\}$

$$\mathbf{D} \otimes (\mathbf{D} \,\dot\cup\, \mathbf{L}_{max})^{+} \;\dot\cap\; \mathsf{Mat}.\{\varepsilon\}$$

$=$ $\{$ definition of transitive closure $\}$

$$(\mathbf{D} \otimes (\mathbf{D} \,\dot\cup\, \mathbf{L}_{max}) \otimes (\mathbf{D} \,\dot\cup\, \mathbf{L}_{max})^{*}) \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\}$$

$=$ $\{$ distributivity of matrix product over union,
length considerations: specifically $\mathbf{L}_{max} \dot\cap \mathsf{Mat}.\{\varepsilon\} = \mathbf{0}$ $\}$

$$(\mathbf{D} \otimes \mathbf{D} \otimes (\mathbf{D} \,\dot\cup\, \mathbf{L}_{max})^{*}) \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\}$$

$=$ $\{$ star decomposition $\}$

$$(\mathbf{D} \otimes \mathbf{D} \otimes \mathbf{D}^{*} \otimes (\mathbf{L}_{max} \otimes \mathbf{D}^{*})^{*}) \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\}$$

$=$ $\{$ for all $\mathbf{X}$, $\mathbf{X}^{*} \;=\; \mathbf{I} \,\dot\cup\, \mathbf{X} \otimes \mathbf{X}^{*}$ with $\mathbf{X} := \mathbf{L}_{max} \otimes \mathbf{D}^{*}$,
distributivity of matrix product over union
and length considerations $\}$

$$(\mathbf{D} \otimes \mathbf{D} \otimes \mathbf{D}^{*}) \,\dot\cap\, \mathsf{Mat}.\{\varepsilon\}$$

$=$ $\{$ definition of transitive closure, $\mathbf{D}$ is a constant matrix $\}$

$$\mathbf{D} \otimes \mathbf{D}^{+} \;\;.$$

This completes the proof of the second equation in the lemma. The proof of the penultimate equation is similar. We omit the details:

$\mathbf{L}_{max} \dot{\cap} \neg((\mathbf{D} \dot{\cup} \mathbf{L}_{max}) \otimes (\mathbf{D} \dot{\cup} \mathbf{L}_{max})^{+})$

$=$ { $\mathbf{L}_{max}$ is a linear matrix, $\mathbf{D}$ is a constant matrix; length considerations }

$\mathbf{L}_{max} \dot{\cap} \neg(\mathbf{D}^{+} \otimes \mathbf{L}_{max} \ \dot{\cup} \ \mathbf{L}_{max} \otimes \mathbf{D}^{+})$

$=$ { by lemma 135 and lemma 130, $\mathbf{D}^{*} = (\mathbf{C}_{max})^{*} = \mathbf{C}_{max}$ }

$\mathbf{L}_{max} \dot{\cap} \neg(\mathbf{D} \otimes \mathbf{C}_{max} \otimes \mathbf{L}_{max} \ \dot{\cup} \ \mathbf{L}_{max} \otimes \mathbf{C}_{max} \otimes \mathbf{D})$

$=$ { lemma 130 }

$\mathbf{L}_{max} \dot{\cap} \neg(\mathbf{D} \otimes \mathbf{L}_{max} \ \dot{\cup} \ \mathbf{L}_{max} \otimes \mathbf{D})$

$=$ { definition of $\mathbf{L}_{min}$: definition 149 }

$\mathbf{L}_{min}$.

The final equation is a straightforward combination of the first three equations together with distributivity of union over intersection.
□

Lemma 154 Let $\mathbf{C}_{min}(E,\zeta)$ and $\mathbf{L}_{min}(E,\zeta)$ be as in definition 149. Then

$$\zeta^{\sharp} \bullet |\overline{E}| \;=\; (\mathbf{C}_{min}(E,\zeta) \dot{\cup} \mathbf{L}_{min}(E,\zeta))^{*} \quad ,$$

$$\mathbf{C}_{max}(E,\zeta) = (\mathbf{C}_{min}(E,\zeta))^{*} \quad \text{and}$$

$$\mathbf{L}_{max}(E,\zeta) \;=\; \mathbf{C}_{max}(E,\zeta) \otimes \mathbf{L}_{min}(E,\zeta) \otimes \mathbf{C}_{max}(E,\zeta) \quad .$$

Proof As in the proof of lemma 151, we write $\mathbf{C}_{min}$, $\mathbf{C}_{max}$, $\mathbf{L}_{min}$ and $\mathbf{L}_{max}$ (thus omitting the parameters).

We first note that $\mathbf{C}_{min} \dot{\cup} \mathbf{L}_{min} \dot{\subseteq} \mathbf{B}$. (See lemma 151.) Since $\mathbf{B}$ is both left- and right-definite (corollary 148), it follows from lemma 143 that $\mathbf{C}_{min} \dot{\cup} \mathbf{L}_{min}$ is both left- and right-definite. Now

$\zeta^{\sharp} \bullet |\overline{E}| \;=\; (\mathbf{C}_{min} \dot{\cup} \mathbf{L}_{min})^{*}$

$=$ { Conway's approximation theorem: theorem 132 }

$(\mathbf{C}_{max} \dot{\cup} \mathbf{L}_{max})^{*} = (\mathbf{C}_{min} \dot{\cup} \mathbf{L}_{min})^{*}$

$=$ { lemma 135 with $X := \mathbf{C}_{max} \dot{\cup} \mathbf{L}_{max}$, $(\mathbf{C}_{max} \dot{\cup} \mathbf{L}_{max}) \dot{\cap} \neg\mathbf{I} \;=\; \mathbf{B}$ }

$$\begin{array}{ll}
 & \mathbf{B}^* \;=\; (\mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min})^* \\
= & \quad \{ \quad \mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min} \text{ is right-definite,} \\
 & \quad\quad \text{UEP of regular algebra: theorem 142} \quad \} \\
 & \mathbf{B}^* \;=\; \mathbf{I} \;\dot{\cup}\; \mathbf{B}^* \otimes (\mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min}) \\
\Leftarrow & \quad \{ \quad \mathbf{B}^* = \mathbf{I} \dot{\cup} \mathbf{B}^+ \text{, Leibniz} \quad \} \\
 & \mathbf{B}^+ \;=\; \mathbf{B}^* \otimes (\mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min}) \\
= & \quad \{ \quad \mathbf{B} \text{ is left-definite: UEP of regular algebra: theorem 142} \quad \} \\
 & \mathbf{B}^+ \;=\; (\mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min}) \,\dot{\cup}\, \mathbf{B} \otimes \mathbf{B}^+ \\
= & \quad \{ \quad \text{lemma 151} \quad \} \\
 & \mathbf{B}^+ \;=\; (\mathbf{B} \,\dot{\cap}\, \neg(\mathbf{B} \otimes \mathbf{B}^+)) \,\dot{\cup}\, \mathbf{B} \otimes \mathbf{B}^+ \\
= & \quad \{ \quad \text{absorption rule of set calculus} \quad \} \\
 & \mathbf{B}^+ \;=\; \mathbf{B} \,\dot{\cup}\, \mathbf{B} \otimes \mathbf{B}^+ \\
= & \quad \{ \quad \text{fixed-point definition of transitive closure} \quad \} \\
 & \mathsf{true} \quad .
\end{array}$$

The remaining two equations follow from the first equation by applying theorem 132:

$$\begin{array}{ll}
 & \mathbf{C}_{max} \\
= & \quad \{ \quad \text{definition} \quad \} \\
 & (\zeta^\sharp \bullet |\overline{\mathsf{E}}|) \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\} \\
= & \quad \{ \quad \zeta^\sharp \bullet |\overline{\mathsf{E}}| \;=\; (\mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min})^* \quad \} \\
 & (\mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min})^* \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\} \\
= & \quad \{ \quad \text{algebra of regular expressions} \quad \} \\
 & ((\mathbf{C}_{min})^* \;\dot{\cup}\; \mathbf{L}_{min} \otimes (\mathbf{C}_{min} \,\dot{\cup}\, \mathbf{L}_{min})^*) \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\} \\
= & \quad \{ \quad \text{length considerations (details omitted)} \quad \} \\
 & (\mathbf{C}_{min})^* \,\dot{\cap}\, \mathsf{Mat}.\{\varepsilon\} \\
= & \quad \{ \quad \mathbf{C}_{min} \,\dot{\subseteq}\, \mathsf{Mat}.\{\varepsilon\} \quad \} \\
 & (\mathbf{C}_{min})^* \quad .
\end{array}$$

Finally,

$$\mathbf{L}_{max}$$

$=\quad\{\quad$ definition $\quad\}$

$(\zeta^\sharp \bullet |\overline{E}|)\ \dot{\cap}\ \mathsf{Mat}.\{T\}$

$=\quad\{\quad$ as above $\quad\}$

$(\mathbf{C}_{min}\ \dot{\cup}\ \mathbf{L}_{min})^*\ \dot{\cap}\ \mathsf{Mat}.\{T\}$

$=\quad\{\quad$ star decomposition $\quad\}$

$(\mathbf{C}_{min})^* \otimes (\mathbf{L}_{min} \otimes (\mathbf{C}_{min})^*)^*\ \dot{\cap}\ \mathsf{Mat}.\{T\}$

$=\quad\{\quad$ length considerations (details omitted) $\quad\}$

$(\mathbf{C}_{min})^* \otimes \mathbf{L}_{min} \otimes (\mathbf{C}_{min})^*$

$=\quad\{\quad \mathbf{C}_{max} = (\mathbf{C}_{min})^*$ (just proved) $\quad\}$

$\mathbf{C}_{max} \otimes \mathbf{L}_{min} \otimes \mathbf{C}_{max}$.

□

Note the use of both right-definiteness and left-definiteness in this proof. These two notions coincide for finite matrices. For infinite matrices, they are different. (For example, the less-than relation on natural numbers is well-founded —"left-definite"— but the greater-than relation is not —less-than is not "right-definite". This is why we have been forced to limit the theorems in this section to regular languages.

Theorem 151 establishes equation (150) as required. So we conclude:

Theorem 155 (Least Factorial Function) Suppose $\mathbf{C}_{min}(E,\zeta)$ and $\mathbf{L}_{min}(E,\zeta)$ are as defined in definition 149. Then $\mathbf{C}_{min}(E,\zeta)\ \dot{\cup}\ \mathbf{L}_{min}(E,\zeta)$ is the least starth root of $\zeta^\sharp \bullet |\overline{E}|$. (That is, it is a minimal starth root and is unique.)

□

Example 156 (Conway's Example) To enable direct comparison with Conway's "best" approximation, this example revisits the example he used [Con71, p49–p53]. Figs. 5(a) and (b) show the machine and anti-machine of the language E denoted by the regular expression $(a{+}b)^*b(ba)^* + (ba)^*$. (The machine and anti-machine are reproduced from [Con71, p49]. The regular expression is not the same as that given by Conway: we have used the anti-machine to construct a simpler expression.)

Start and final nodes have been indicated in the usual way for each graph.

Figs. 5(c) and (d) show, respectively, the factor graph of this language and the maximal constant+linear approximation to the factor matrix of the language, but omitting an inadmissible node. Both are explained in detail in section 6.5. For the moment, however, note that the factor graph (fig. 5(c)) is the reflexive-transitive reduction of fig. 5(d), and both are starth roots of the factor matrix of the language.

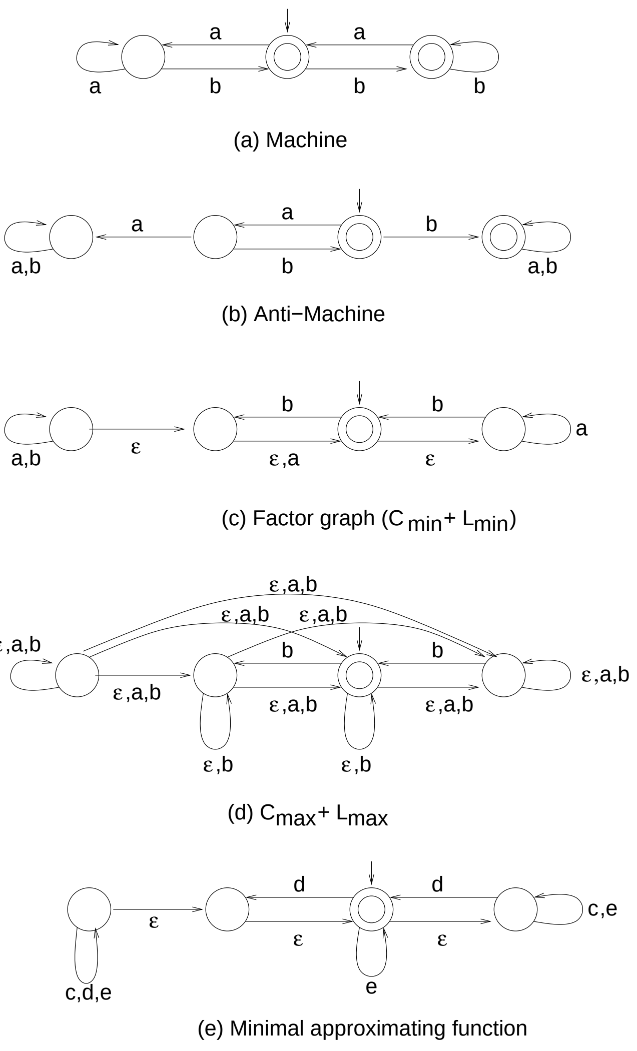


Figure 5: Conway's Example of Approximation

Fig. 5(e) shows the graph corresponding to $\mathbf{C}_{min}(E,\zeta)\,\dot{\cup}\,\mathbf{L}_{min}(E,\zeta)$ where the alphabet of the approximating function ζ is $\{c,d,e\}$ and the function ζ is defined to be $\zeta.c=\{aa\}$, $\zeta.d=\{b\}$ and $\zeta.e=\{ba\}$. Using this graph, it is easy to construct the regular expression $(e + (c{+}e)^*d)^*$ as the "best" approximating function to the event E by ζ. The maximal approximation to E by the languages $\{aa\}$, $\{b\}$ and $\{ba\}$ is thus $(ba + (aa{+}ba)^*b)^*$. Conway's "best" approximating function [Con71, fig. 6.3] is much more complicated than fig. 5(e): the latter is the reflexive-transitive reduction of the graph constructed by Conway. Because the approximating events in this example are finite languages (indeed, singleton sets), it is easy to construct fig. 5(e) from fig. 5(c). (Note that the leftmost node is inadmissible.) Specifically, the empty-word edges are the same in both figures,

and there is an edge in fig. 5(e) labelled x (where x is one of c, d and e) from node i to j in fig. 5(e) if and only if there is a path from node i to j in the linear subgraph of fig. 5(c) that spells $\zeta.x$ and there is no other such path of smaller edge length. (The "linear subgraph" is obtained by omitting the empty-word edges.)
□

6.5 The Factor Graph

We now introduce the "factor graph" of a regular language, a concept that was first introduced by the author in [Bac75] and subsequently, in [BL77], shown to form the basis of the Knuth-Morris-Pratt pattern-matching algorithm [KMP77] and Aho and Corasick's generalisation [AC75] of the KMP algorithm to a set of patterns. (For an example of the relationship between the KMP algorithm and the factor graph, see section 9.1.)

Example 156 gave a foretaste of a factor graph; the factor graph of a regular language E is the minimal "starth root" of the factor matrix of E. The proof of existence of the factor graph, for arbitrary regular language E, is an elementary instance of theorem 154: see theorem 158. An effective way of constructing the factor graph is more complicated. The full details are given in section 6.7. Analysing the structure of the factor matrix of a given language in order to identify the factor graphs of factors of the language is yet more complicated since it involves constructing the "syntactic monoid" of the language. See section 7.

The following lemma is essentially due to Conway; in both [Bac75] and [Bac16], it is attributed to Conway since it is a straightforward consequence of his approximation theorem. It is, however, never explicitly stated by Conway.

Lemma 157 Suppose E is a language over the alphabet T. Then

$$|\overline{E}| \;=\; \mathbf{C}_{max}.E \;\dot{\cup}\; (\mathbf{L}_{max}.E)^{+} \;=\; (\mathbf{C}_{max}.E \;\dot{\cup}\; \mathbf{L}_{max}.E)^{*}$$

where $\mathbf{C}_{max}.E$ denotes $|\overline{E}|\dot{\cap}\mathsf{Mat}.\{\varepsilon\}$ and $\mathbf{L}_{max}.E$ denotes $|\overline{E}|\dot{\cap}\mathsf{Mat}.T$.

(Note that our use of $\mathbf{C}_{max}.E$ here is consistent with its definition in theorem 133. In that theorem, E is assumed to be an event in a regular algebra $\mathcal{S}$; we have now specialised the algebra to the algebra of languages over the alphabet T.)

Proof Instantiate the Generalised Approximation Theorem (theorem 132) with ζ defined to be the "lifted identity" function on T. That is, suppose $\zeta.a = \{a\}$ for all a in T. Then it is easily verified that both $\zeta^{\flat}$ and $\zeta^{\sharp}$ (as defined by (119) and (120)) are equal to the identity function on languages. The theorem follows immediately.
□

Recalling definition 124, the matrix ($|\overline{\mathsf{E}}|\,\dot{\cap}\,\mathsf{Mat}.\{\varepsilon\}$) is the ***maximal constant*** approximating "function" for $|\overline{\mathsf{E}}|$ and the matrix $|\overline{\mathsf{E}}|\,\dot{\cap}\,\mathsf{Mat}.\mathsf{T}$ is the ***maximal linear*** approximating "function" for $|\overline{\mathsf{E}}|$. Perhaps confusingly —because the function ζ is the (lifted) identity function— these are also the maximal constant and linear *approximations* to $|\overline{\mathsf{E}}|$. The factor *graph* of a regular language E is the pointwise union of the ***minimal constant and linear approximations***:

Theorem 158 (The Factor Graph) Suppose E is a *regular* language over the alphabet T. Let

$$\mathbf{C}_{max}.\mathsf{E} \;=\; |\overline{\mathsf{E}}|\,\dot{\cap}\,\mathsf{Mat}.\{\varepsilon\}\quad ,$$

$$\mathbf{D} \;=\; |\overline{\mathsf{E}}|\,\dot{\cap}\,\mathsf{Mat}.\{\varepsilon\}\,\dot{\cap}\,\neg\mathbf{I}\quad \text{and}$$

$$\mathbf{C}_{min}.\mathsf{E} = \mathbf{D}\,\dot{\cap}\,\neg(\mathbf{D}\otimes\mathbf{D}^{+})\quad .$$

Also, let

$$\mathbf{L}_{max}.\mathsf{E} \;=\; |\overline{\mathsf{E}}|\,\dot{\cap}\,\mathsf{Mat}.\mathsf{T}\quad \text{and}$$

$$\mathbf{L}_{min}.\mathsf{E} \;=\; \mathbf{L}_{max}.\mathsf{E}\,\dot{\cap}\,\neg(\mathbf{D}\otimes\mathbf{L}_{max}.\mathsf{E})\,\dot{\cap}\,\neg(\mathbf{L}_{max}.\mathsf{E}\otimes\mathbf{D})\quad .$$

Then

$$\mathbf{C}_{max}.\mathsf{E} \;=\; (\mathbf{C}_{min}.\mathsf{E})^{*}\quad ,$$

$$\mathbf{L}_{max}.\mathsf{E} \;=\; \mathbf{C}_{max}.\mathsf{E}\otimes\mathbf{L}_{min}.\mathsf{E}\otimes\mathbf{C}_{max}.\mathsf{E}\quad \text{, and}$$

$$|\overline{\mathsf{E}}| \;=\; \mathbf{C}_{min}.\mathsf{E}\;\dot{\cup}\;(\mathbf{L}_{min}.\mathsf{E})^{+} \;=\; (\mathbf{C}_{min}.\mathsf{E}\;\dot{\cup}\;\mathbf{L}_{min}.\mathsf{E})^{*}\quad .$$

Moreover, $\mathbf{C}_{min}.\mathsf{E}\;\dot{\cup}\;\mathbf{L}_{min}.\mathsf{E}$ is the *least* starth root of the factor matrix of E.

Proof Combine lemma 157, theorem 154 and 155 with ζ defined as in lemma 157 to be the "lifted identity" function on T. The theorem follows immediately.
□

The matrix $\mathbf{C}_{min}.\mathsf{E}\;\dot{\cup}\;\mathbf{L}_{min}.\mathsf{E}$ is a constant+linear matrix, i.e. a transition graph, and so it is appropriate to call it the *factor graph* of the regular language E.

6.6 Equivalence Relations on Languages

In order to calculate the factor graph of a regular language, it suffices to calculate the machine and anti-machine of the language. This is explained in section 6.7. In this section, we recall well-known properties that underlie the construction of finite-state machines and are crucial to constructing factor graphs, as we show in section 6.7. It is convenient to also summarise properties of the so-called "syntactic monoid" of a language.

These properties are exploited in section 7 as an effective means to determine the factor graphs and factor matrices of factors of a regular language.

Suppose E is a language over the alphabet T. Then E defines three equivalence relations on T^* — E_l, E_r and E_c — given by, for all x and y in T^*:

$$\begin{aligned} xE_ly &\equiv \langle \forall z : z\in T^* : zx\in E \equiv zy\in E\rangle \\ xE_ry &\equiv \langle \forall z : z\in T^* : xz\in E \equiv yz\in E\rangle \\ xE_cy &\equiv \langle \forall u,v : u\in T^* \wedge v\in T^* : uxv\in E \equiv uyv\in E\rangle \end{aligned}$$

These are the so-called *left-invariant* equivalence relation, *right-invariant* equivalence relation and *congruence* relation introduced by Rabin and Scott [RS59]. Being equivalence relations, each partitions T^* into equivalence classes. We call an equivalence class modulo E_l an r-*class* of E, an equivalence class modulo E_r an l-*class* of E, and an equivalence class modulo E_c a c-*class* of E. We use $E_r(x)$ to denote the l-class that includes word x. Similarly for $E_l(x)$ and $E_c(x)$. (To avoid confusion in explanatory prose, we use a different font for the relations E_l, E_r and E_c and the functions E_l, E_r and E_c. In the context of formal statements, which is intended should be clear.)

Note the switch: an equivalence class modulo $E_{\underline{l}}$ is an $\underline{r}$-class. The reason for the switch is the following theorem [Bac75]:

Theorem 159 Each left factor of E is a union of l-classes of E, each right factor of E is a union of r-classes of E, and each factor of E is a union of c-classes of E. Specifically, if F is a left factor of E,

$$F = \langle \cup y : y\in F : E_r(y)\rangle \quad .$$

If F is a right factor of E,

$$F = \langle \cup y : y\in F : E_l(y)\rangle \quad .$$

Finally, if F is a factor of E,

$$F = \langle \cup y : y\in F : E_c(y)\rangle \quad .$$

Proof We show that each left factor of E is a union of l-classes of E as follows. First, for all Z, $Z\subseteq T^*$,

$$\begin{aligned} & x\in E/Z \\ = &\qquad \{ \quad \text{definition of } / \quad \} \\ & \{x\}\cdot Z\subseteq E \\ = &\qquad \{ \quad \text{definition of concatenation} \quad \} \\ & \langle \forall z : z\in Z : xz\in E\rangle \quad . \end{aligned}$$

Hence,

$$
\begin{array}{ll}
 & E/Z \\
= & \{ \quad \text{definition} \quad \} \\
 & \langle \cup x : x \in E/Z : \{x\} \rangle \\
= & \{ \quad x E_r x \text{, idempotency of set union} \quad \} \\
 & \langle \cup x \ :\ x \in E/Z \ :\ \langle \cup y : x E_r y : \{x\} \rangle \rangle \\
= & \{ \quad \text{above and nesting} \quad \} \\
 & \langle \cup x,y \ :\ \langle \forall z : z{\in}Z : xz{\in}E \rangle \ \wedge x E_r y \ :\ \{x\} \rangle \\
= & \{ \quad \text{definition of } E_r \text{ and} \\
 & \quad\ \ \text{substitution of equals for equals} \quad \} \\
 & \langle \cup x,y \ :\ \langle \forall z : z{\in}Z : yz{\in}E \rangle \ \wedge x E_r y \ :\ \{x\} \rangle \\
= & \{ \quad \text{above} \quad \} \\
 & \langle \cup x,y \ :\ y \in E/Z \wedge x E_r y \ :\ \{x\} \rangle \\
= & \{ \quad \text{nesting and definition of } E_r(y) \quad \} \\
 & \langle \cup y : y \in E/Z : E_r(y) \rangle \quad .
\end{array}
$$

The remaining two properties are proved similarly.
□

(The above calculational proof of theorem 159 was included in [Bac16] . It is included again here for completeness.)

Earlier we showed that a right factor of E is an intersection of derivatives of E (specifically, $L\backslash E = \langle \cap w : w{\in}L : \partial_w E \rangle$). Similarly, a left factor of E is the reverse of an intersection of anti-derivatives of E. (The reverse of E is the set of words obtained by reversing words in E and an anti-derivative of E is a derivative of the reverse of E.) This is the characterisation given by Conway. The above characterisation, introduced in [Bac75], is much more useful when calculating the factors of a language because such calculations use only *representatives* of the three types of equivalence class rather than the full class. We see how this works below. The formal theory justifying the use of representatives is given in section 7.2.

The basic theorem of regular languages, attributed to J.R.Myhill by Rabin and Scott [RS59], is that the following statements are all equivalent:

- E is regular (i.e. can be denoted by a regular expression).
- The relation E_l has finite index.

- The relation E_r has finite index.
- The relation E_c has finite index.

It follows that E is regular if and only if it has a finite number of factors. Thus the factor "matrix" exists for all languages E but it is only for regular languages that the use of the word "matrix" complies with standard conventions: entries are indexed by pairs of left factors but, precisely when E is a regular language, the left factors can themselves by indexed by numbers from 1 to n, for some (finite) number n.

As is well-known, the relation E_r defines the ("reduced, deterministic finite-state") *machine* of E and the relation E_l defines the *anti-machine* of E. The equivalence classes of E_r are called the *states* of the machine. That is, each state of the machine is an l-class of E. (Recall the switch: equivalence classes of E_r are l-classes and equivalence classes of E_l are r-classes.) Its *transition function* δ maps a state and a symbol of the alphabet to a state; it is defined by $\delta(E_r(x),a) = E_r(xa)$, for all words x and all symbols a. (It is easy to verify that this is a valid definition, i.e. the choice of representative element x of the l-class $E_r(x)$ is irrelevant — formally, $E_r(xa) = E_r(ya) \Leftarrow xE_ry$.) The *start* state is $E_r(\varepsilon)$ and $E_r(x)$ is a *final* state if $x{\in}E$. Similarly, the equivalence classes of E_l are the states of the anti-machine of E; its transition function $\overleftarrow{\delta}$ is defined by $\overleftarrow{\delta}(E_l(x),a) = E_l(ax)$. As suggested by the notation "$\overleftarrow{\delta}$", the anti-machine of E is the machine of the reverse of E. We use these facts without further explanation below.

As already mentioned, the relation E_c is a congruence relation on words. That is, for all words u, v, w and x, $uv\ E_c\ wx$ if $u\ E_c\ w$ and $v\ E_c\ x$. This has the important corollary that the set of equivalence classes defined by E_c is the carrier set of a monoid, called the *syntactic monoid* of the language E. Specifically, let $\mathsf{SM}.E$ denote the set of equivalence classes of the relation E_c; let E_c denote the function that maps word u to the equivalence class containing u, and let 1 denote $E_c(\varepsilon)$. Then

Theorem 160 $(\mathsf{SM}.E\ ,\circ,\ 1)$ is a monoid where product is defined by $E_c(u){\circ}E_c(v) = E_c(uv)$. It is finitely generated by $\{a : a{\in}T : E_c(a)\}$ and the function E_c is a monoid homomorphism from T^* onto $\mathsf{SM}.E$.

□

Theorem 160 is well-known; we omit the proof, which is straightforward.

We use the symbol "$\circ$" to denote the product in $\mathsf{SM}.E$ because elements of the syntactic monoid can be identified with relations on l-classes (or equivalently, relations on r-classes) and, indeed, this is how the syntactic monoid is constructed. The details are somewhat hidden in the first part of the proof of Myhill's theorem [RS59, Theorem 1, p.117], so let us make them explicit. The fundamental properties are summarised in the following definition and lemma.

Definition 161 Given language E over alphabet T, we define, for each word u in T^*, the relation $Ctx.u$ on l-classes of E by,

$$\langle\forall x,y \; : \; x\in T^* \wedge y\in T^* \; : \; E_r(x) \; Ctx.u \; E_r(y) \;\; \equiv \;\; E_r(xu) = E_r(y)\rangle \quad .$$

□

Of course, it is necessary to establish that definition 161 is sound: that is, the property $E_r(xu) = E_r(y)$ is independent of the representative element x chosen from the class $E_r(x)$. That this is so is an immediate consequence of the right invariance of E_r, specifically, for all words u,

$$E_r(xu) = E_r(x'u) \;\; \Leftarrow \;\; E_r(x) = E_r(x')$$

which we prove as follows. Suppose $E_r(x) = E_r(x')$. Then, for all words y,

$$\begin{array}{ll}
 & y\in E_r(x'u) \\
= & \{ \quad \text{definition of } E_r \quad \} \\
 & \langle\forall z : z\in T^* : x'uz\in E \;\equiv\; yz\in E\rangle \\
= & \{ \quad \text{assumption: } E_r(x) = E_r(x'), \\
 & \quad\;\; \text{thus } x'uz\in E \equiv xuz\in E, \\
 & \quad\;\; \text{Leibniz} \quad \} \\
 & \langle\forall z : z\in T^* : xuz\in E \;\equiv\; yz\in E\rangle \\
= & \{ \quad \text{definition of } E_r \quad \} \\
 & y\in E_r(xu) \quad .
\end{array}$$

It follows that $E_r(xu) = E_r(x'u)$ by set comprehension.

Lemma 162

$$\langle\forall u,v \; : \; u\in T^* \wedge v\in T^* \; : \; Ctx.u = Ctx.v \;\; \equiv \;\; u \; E_c \; v\rangle \quad .$$

Proof

$$\begin{array}{ll}
 & Ctx.u = Ctx.v \\
= & \{ \quad \text{definition 161} \quad \} \\
 & \langle\forall x,y \;::\; E_r(xu) = E_r(y) \;\equiv\; E_r(xv) = E_r(y)\rangle \\
= & \{ \quad \text{definition of } E_r \quad \}
\end{array}$$

$$\langle\forall x,y :: \langle\forall z :: xuz{\in}E \equiv yz{\in}E\rangle \;\equiv\; \langle\forall z :: xvz{\in}E \equiv yz{\in}E\rangle\rangle$$

$\Rightarrow \quad \{ \quad y := xv \quad \}$

$$\langle\forall x :: \langle\forall z :: xuz{\in}E \equiv xvz{\in}E\rangle \;\equiv\; \langle\forall z :: xvz{\in}E \equiv xvz{\in}E\rangle\rangle$$

$= \quad \{$ reflexivity of equivales, unit of equivales $\}$

$$\langle\forall x :: \langle\forall z :: xuz{\in}E \equiv xvz{\in}E\rangle\rangle$$

$= \quad \{$ nesting and definition of E_c $\}$

$$u\ E_c\ v$$

$= \quad \{$ definition of E_c and E_r $\}$

$$\langle\forall x :: E_r(xu) = E_r(xv)\rangle$$

$\Rightarrow \quad \{$ Leibniz $\}$

$$\langle\forall x,y :: E_r(xu) = E_r(y) \;\equiv\; E_r(xv) = E_r(y)\rangle$$

$= \quad \{$ definition 161 $\}$

$$\mathsf{Ctx}.u = \mathsf{Ctx}.v \quad .$$

The lemma follows by mutual implication.
□

Lemma 163

$$\langle\forall u,v : u \in T^* \wedge v \in T^* : \mathsf{Ctx}.u \circ \mathsf{Ctx}.v \;=\; \mathsf{Ctx}.uv\rangle \quad .$$

(The symbol "∘" here denotes composition of relations.)

Proof Suppose $x \in T^*$ and $z \in T^*$. Then

$$E_r(x)\ \ \mathsf{Ctx}.u \circ \mathsf{Ctx}.v\ \ E_r(z)$$

$= \quad \{$ definiition of composition of relations $\}$

$$\langle\exists y : y \in T^* : E_r(x)\ \mathsf{Ctx}.u\ E_r(y) \;\wedge\; E_r(y)\ \mathsf{Ctx}.u\ E_r(z)\rangle$$

$= \quad \{$ definition of Ctx: definition 161 $\}$

$$\langle\exists y : y \in T^* : E_r(xu) = E_r(y) \;\wedge\; E_r(yv) = E_r(z)\rangle$$

$= \quad \{ \quad E_r(xu) = E_r(y) \wedge E_r(yv) = E_r(z)$

$\qquad = \quad \{$ definition of E_r $\}$

$$\langle\forall w :: xuw{\in}E \equiv yw{\in}E\rangle$$

$$\wedge\ \langle\forall w' :: yvw'{\in}E \equiv zw'{\in}E\rangle$$

$\Rightarrow \quad \{ \quad w := vw' \text{, Leibniz} \quad \}$

$\langle \forall w' :: xuvw' \in E \equiv zw' \in E \rangle$

$= \quad \{ \quad \text{definition of } E_r \quad \}$

$E_r(xuv) = E_r(z) \quad \}$

$\langle \exists y : y \in T^* : E_r(xu) = E_r(y) \ \wedge \ E_r(xuv) = E_r(z) \rangle$

$= \quad \{ \quad y := xu$ and distributivity of conjunction over disjunction $\quad \}$

$E_r(xuv) = E_r(z)$

$= \quad \{ \quad$ definition of Ctx: definition 161 $\quad \}$

$E_r(x) \ \text{Ctx}.uv \ E_r(z) \quad .$

□

Lemmas 162 and 163 embody the standard algorithm for constructing the syntactic monoid. Recall that the equivalence classes of E_r are the states of the machine of E. Beginning with the relation $\text{Ctx}.\varepsilon$, which is the identity relation on states of the machine, we construct the relations $\text{Ctx}.u$ for words u in lexicographic order, checking at each step that $\text{Ctx}.u$ is not equal to $\text{Ctx}.v$ for a preceeding word v. Now, by lemma 163, for $u \in T^*$ and $a \in T$,

$$\text{Ctx}.ua = \text{Ctx}.u \circ \text{Ctx}.a$$

(where "$\circ$" denotes the composition of relations) and

$$E_r(x) \ \text{Ctx}.a \ E_r(y) \ \equiv \ \delta(E_r(x), a) = E_r(y) \quad .$$

Thus, knowing $\text{Ctx}.u$ we can calculate $\text{Ctx}.ua$ from the transition relation δ defined by the machine of E. Appendix C gives some examples to illustrate how this is done.

One way of phrasing lemma 162 is that there is an injective function from the equivalence classes of E_c to relations on the states of the machine of E. The product operator of SM.E, introduced in theorem 160, is such that, for words u and v,

$$E_c(u) \circ E_c(v) \ = \ E_c(uv) \quad .$$

and

$$\text{Ctx}.u \circ \text{Ctx}.v \ = \ \text{Ctx}.uv \quad .$$

Formally, the two different occurrences of the symbol "$\circ$" have different meanings (because their types are different) but essentially they are the same. On the other hand, it is NOT the case (in general) that $E_c(u) \cdot E_c(v) \ = \ E_c(uv)$, where the symbol "$\cdot$" denotes concatenation of languages. (It is the case that $E_c(u) \cdot E_c(v) \subseteq E_c(uv)$ but the inclusion may be proper.) It is for this reason that we have chosen to use the symbol "$\circ$" to denote the product operator in the syntactic monoid.

Note that, although it is convenient to introduce the syntactic monoid here, it is *not* relevant to the construction of factor graphs. Its construction is relevant to the identification of factor matrices/graphs of factors of a language. See section 7.

6.7 Constructing the Factor Graph

Suppose E is a regular language. The construction of the factor graph involves calculating all the left factors of E whilst simultaneously calculating $\mathbf{C}_{min}$ and $\mathbf{L}_{min}$. The definition of $\mathbf{C}_{min}$ and $\mathbf{L}_{min}$ in terms of $\mathbf{C}_{max}$ and $\mathbf{L}_{max}$ is exploited in this process but calculating the full details of the latter matrices is avoided as far as possible. In this section, we show how this is done.

We illustrate the construction using the language $((a{+}b)^*\, c\, a^*\, (a{+}b))^*$ with alphabet $\{a,b,c\}$.

Fig. 6 shows the machine and anti-machine for our example language.

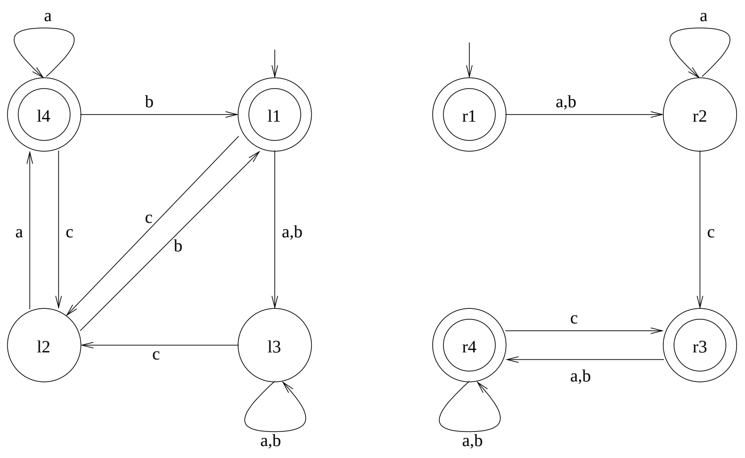


Figure 6: Machine and Anti-Machine of $((a{+}b)^*\, c\, a^*\, (a{+}b))^*$

The machine and anti-machine as shown are "all-admissible". This means that in each a node has been omitted from which there is no path to the final state. Such nodes are said to be *inadmissible*.

We recall that each left factor of E is a union of l-classes of E and each right factor is a union of r-classes of E. Moreover, the l-classes of E are in one-to-one correspondence with the nodes of the machine of E (the reduced, deterministic finite automaton that recognises E) and the r-classes of E are in one-to-one correspondence with the nodes of the anti-machine of E (the reduced, deterministic finite automaton that recognises the reverse of E). Specifically, for a given state of the machine, the corresponding l-class is the set of all words recognised by that state (i.e. the set of all words spelt out by a path from the start state to the state); for a given state of the anti-machine, the corresponding r-class is the reverse of the set of all words recognised by that state.

The table below names each l-class and each r-class for our example language. The names are those used in fig. 6 with the addition of the names $l5$ and $r5$ of the inadmissible states of machine and anti-machine, respectively. Next to each name of an l-class or r-class is an element of the class. We call these *representatives* of the class. For example, ca is a representative of class $l4$ because the word ca spells out a path from the start state $l1$ to the state $l4$. Similarly, ca is a representative of $r3$ because its reverse ac spells out a path from the start state $r1$ to the state $r3$.

l-class	representative	r-class	representative
$l1$	ε	$r1$	ε
$l2$	c	$r2$	a
$l3$	a	$r3$	ca
$l4$	ca	$r4$	aca
$l5$	cc	$r5$	c

After constructing the machine and anti-machine and choosing representatives of the l- and r-classes in this way, the next step is to calculate $l\triangleright$ and $r\triangleleft$ for each of the classes. Each entry $l\triangleright$ is a right factor and thus a union of r-classes, and each entry $r\triangleleft$ is a left factor and thus a union of l-classes. The table below shows the result of the calculation for our example language.

l-class l	$l\triangleright$	r-class r	$r\triangleleft$
$l1$	$r1 \cup r3 \cup r4$	$r1$	$l1 \cup l4$
$l2$	$r2 \cup r4$	$r2$	$l2 \cup l4$
$l3$	$r3 \cup r4$	$r3$	$l1 \cup l3 \cup l4$
$l4$	$r1 \cup r2 \cup r3 \cup r4$	$r4$	$l1 \cup l2 \cup l3 \cup l4$
$l5$	$\emptyset$	$r5$	$\emptyset$

Calculating the entries is made easy by the use of representatives. Specifically, suppose u is the representative of l and v is the representative of r. Then $r \subseteq l\triangleright$ exactly when $uv \in E$ (which is easily checked using the machine of E). In our example, $l3\triangleright = r3 \cup r4$ because aca and $aaca$ are elements of E but no other concatenation of a

(the chosen representative of $\mathtt{l3}$) and a representative of an $\mathtt{r}$-class is in $\mathtt{E}$. The same process is used to calculate $\mathtt{r}\triangleleft$ for each $\mathtt{r}$-class $\mathtt{r}$.

The next step is to determine all the left factors and all the right factors. Simultaneously, we calculate the matrix $\mathbf{C}_{\mathit{min}}$ exploiting the correspondence between $\mathbf{C}_{\mathit{min}}$ and the transitive reduction of the subset ordering on left factors. This is the most laborious process since, for each $\mathtt{r}$-class $\mathtt{r}$, the set $\mathtt{r}\triangleleft$ is a left factor but there will typically be more left factors. It is necessary to consider *all* subsets of the set of $\mathtt{l}$-classes to determine whether or not it is a left factor. Suppose $\mathtt{L}$ is a union of $\mathtt{l}$-classes. Then $\mathtt{L}$ is a left factor of $\mathtt{E}$ equivales $\mathtt{L} = \mathtt{L}\triangleright\triangleleft$. Fortunately this property is straightforward to check using the information that has already been computed. We have

$$\mathtt{L}\triangleright \;=\; \langle \cup \mathtt{l} : \mathtt{l} \subseteq \mathtt{L} : \mathtt{l}\rangle \triangleright \;=\; \langle \cap \mathtt{l} : \mathtt{l} \subseteq \mathtt{L} : \mathtt{l}\triangleright\rangle$$

and, for any $\mathtt{R}$ that is a union of $\mathtt{r}$-classes,

$$\mathtt{R}\triangleleft \;=\; \langle \cup \mathtt{r} : \mathtt{r} \subseteq \mathtt{R} : \mathtt{r}\rangle \triangleleft \;=\; \langle \cap \mathtt{r} : \mathtt{r} \subseteq \mathtt{R} : \mathtt{r}\triangleleft\rangle \quad .$$

(In these equations, the dummies $\mathtt{l}$ and $\mathtt{r}$ range over $\mathtt{l}$- and $\mathtt{r}$-classes, respectively.) For example,we can compute from the table above that

$$(\mathtt{l2} \cup \mathtt{l4})\triangleright = \mathtt{l2}\triangleright \cap \mathtt{l4}\triangleright = (\mathtt{r2} \cup \mathtt{r4}) \cap (\mathtt{r1} \cup \mathtt{r2} \cup \mathtt{r3} \cup \mathtt{r4}) = \mathtt{r2} \cup \mathtt{r4}$$

and

$$(\mathtt{r2} \cup \mathtt{r4})\triangleleft = \mathtt{r2}\triangleleft \cap \mathtt{r4}\triangleleft = (\mathtt{l2} \cup \mathtt{l4}) \cap (\mathtt{l1} \cup \mathtt{l2} \cup \mathtt{l3} \cup \mathtt{l4}) = \mathtt{l2} \cup \mathtt{l4} \quad .$$

In this way, we have checked that $\mathtt{l2} \cup \mathtt{l4}$ is indeed a left factor. However, we have that

$$(\mathtt{l1} \cup \mathtt{l3})\triangleright = \mathtt{l1}\triangleright \cap \mathtt{l3}\triangleright = (\mathtt{r1} \cup \mathtt{r2} \cup \mathtt{r3}) \cap (\mathtt{r3} \cup \mathtt{r4}) = \mathtt{r3}$$

and

$$\mathtt{r3}\triangleleft = \mathtt{l1} \cup \mathtt{l3} \cup \mathtt{l4} \quad .$$

So $(\mathtt{l1} \cup \mathtt{l3})\triangleright\triangleleft \neq \mathtt{l1} \cup \mathtt{l3}$ and, hence, $\mathtt{l1} \cup \mathtt{l3}$ is not a left factor.

Computing the poset of left factors and simultaneously $\mathbf{C}_{\mathit{min}}$ involves a search of the set of subsets of the $\mathtt{l}$-classes in decreasing order of size. In our example, the sizes of the subsets range from 5 to 0. So 2^5 subsets must be examined. The subsets that are determined to be left factors are accumulated in a set that we call Γ below.

Beginning with $\mathtt{T}^*$ (the union of all $\mathtt{l}$-classes) —which is always a left factor— as the only element of Γ, each subset $\mathtt{L}$ is checked for the property $\mathtt{L} = \mathtt{L}\triangleright\triangleleft$; if it does, then the $(\mathtt{L}, \mathtt{L}')$th entry in $\mathbf{C}_{\mathit{min}}$ is set to ε for all $\mathtt{L}'$ in Γ that satisfy $\mathtt{L} \subseteq \mathtt{L}'$ and there is no L'' different from L' in Γ such that $L \subseteq L'' \subseteq L'$. On completion, all other entries in $\mathbf{C}_{min}$ are set to $\emptyset$.

The table below shows for our example language each left factor as a union of l-classes and each right factor as a union of r-classes. Fig. 7 shows the reflexive-transitive reduction of the subset relation on left factors. The corresponding entries in the matrix $\mathbf{C}_{min}$ are ε where there is an edge and $\emptyset$ where there is no edge.

Left factor		Right factor	
L0	T^*	R0	$\emptyset$
L1	$l1 \cup l2 \cup l3 \cup l4$	R1	$r4$
L2	$l1 \cup l3 \cup l4$	R2	$r3 \cup r4$
L3	$l2 \cup l4$	R3	$r2 \cup r4$
L4	$l1 \cup l4$	R4	$r1 \cup r3 \cup r4$
L5	$l4$	R5	$r1 \cup r2 \cup r3 \cup r4$
L6	$\emptyset$	R6	T^*

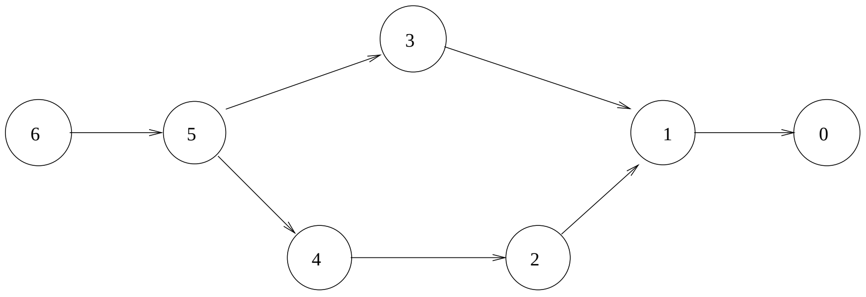


Figure 7: Poset of left factors, Inverted poset of right factors

It is interesting to observe how the construction of $\mathbf{C}_{min}$ provides a non-trivial example of the unity of opposites (section 3.3). Not only is there the one-to-one correspondence between left and right factors observed by Conway but there is also an isomorphism between the poset of left factors and the poset of right factors. Fig. 7 also shows the reflexive-transitive reduction of the superset ordering on right factors. Moreover, infima and suprema of left and right factors correspond in the way predicted by the unity-of-opposites theorem. For example, the supremum of L2 and L3 is L5; correspondingly, the infimum of R2 and R3 is R5.

The next step is to construct the matrices $\mathbf{L}_{max}$ and $\mathbf{L}_{min}$. Suppose L and L' are left factors; let the factor corresponding to L' be R'. Then the symbol a is an entry in the (L, L')th position in $\mathbf{L}_{max}$ if $L \cdot a \cdot R' \subseteq E$. The representatives of the l- and r-classes can be used to check this property. We have to check for each representative u of an l-class in L and each representative v of an r-class in R' whether or not $uav \in E$. The symbol a is an entry in the (L, L')th position in $\mathbf{L}_{min}$ if L and R' are maximal in $L \cdot a \cdot R' \subseteq E$. (That is, if left factor L'' is such that $L'' \cdot a \cdot R' \subseteq E$ then $L'' \subseteq L'$, and if right factor R'' is such that $L \cdot a \cdot R'' \subseteq E$ then $R'' \subseteq R'$.)

The process of constructing the factor graph is completed by identifying the start and final states: the left factors l and r in (74). The left factor r is the easiest to identify: formally,

$$r \;=\; \langle \cup x : x \in E : E_r(x) \rangle \quad .$$

Since a word x is an element of E if x maps the start state of the machine of E to a final state, this means that r is the left factor that is represented by the set of l-classes corresponding to the final states of the machine. In our example, $E = l1 \cup l4 = L4$, so this is the value of r. The left factor l is defined to be E/E. That is, exploiting the equation for r,

$$l \;=\; \langle \cup x : E_r(x) \cdot E \subseteq E : E_r(x) \rangle \quad .$$

The machine for E can be used to determine l-classes that comprise the left factor l. For each representative x of an l-class, determine for each representative y of an l-class in r whether or not xy is an element of E. If this is indeed the case for all such y, the l-class represented by x is a component of the left factor l. In our example, it is easily verified that the l-classes $l1$ and $l4$ are the only ones that satisfy this criterion. Thus, $E/E = l1 \cup l4 = L4$. (In general, $E = E^*$ equivales $l = r$. This property is applicable to our example, obviating the need to calculate r separately.)

In summary, the factor graph of our example language is shown in fig. 8.

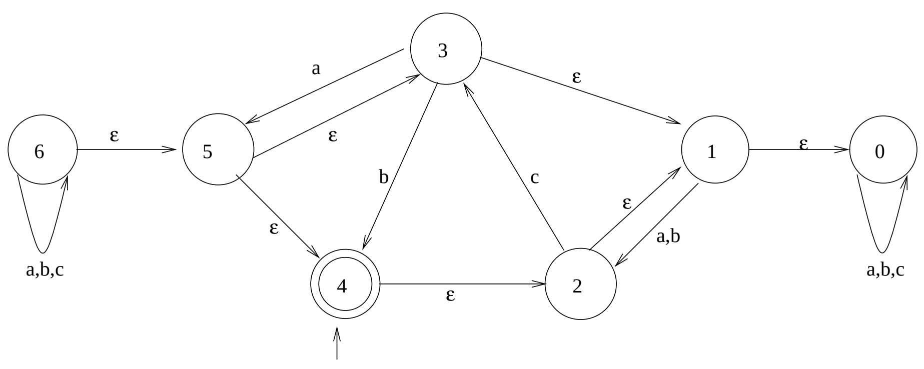


Figure 8: Factor Graph

Before leaving this example, let us note that the factor graph has two *inadmissible* nodes: the nodes labelled 0 and 6 in fig. 8 The node labelled 0 is inadmissible because there are no paths from it to the final node of the graph, and the node labelled 6 is inadmissible because there are no paths to it from the start node of the graph. "Inadmissible" means that they make no contribution when using the factor graph as a (nondeterministic) recogniser of the language.

That $\emptyset$ and T^* are both left and right factors (and, indeed, also factors) is a general phenomenon. They can be disregarded in all calculations involving some event E. (For example, when calculating approximations to E they make no contribution.) As a result, we usually omit the corresponding entries in the factor matrix and the corresponding nodes in the factor graph of an event E.

7 Exploiting the Syntactic Monoid

In section 6.3, we were forced to specialise the discussion to regular languages (rather than events in an arbitrary regular algebra). Only by doing so were we able to establish the existence of a unique starth root of the factor matrix. Calculations with languages are, however, difficult. For example, identifying a submatrix of a factor matrix using the theorems in section 4 is "decidable" (in the technical sense of the word) but decidedly non-trivial if standard techniques are used. The calculations become straight-forward by translating them into calculations on the "syntactic monoid" of the given language. This section makes that process precise.

The main subsection is section 7.1 where we relate calculations on factors of a language to calculations on the syntactic monoid of the language. Section 7.2 gives several examples.

7.1 Factors of Sets of c-Classes

We can, of course, view the equivalence classes of E_c as "approximations" of the event E in the sense of section 5. Specifically, since the syntactic monoid is generated by the equivalence classes $E_c(a)$ where a ranges over elements of the alphabet T, the function E_c restricted to domain T is a suitable approximating function. The "approximation" of E that we obtain by instantiating the theorems in section 5 should then be equal to E (as a corollary of the fact that E is a union of c-classes of E). That is, contrary to the normal meaning of the English word "approximation" —which suggests something that is less good— , the "approximation" is exact. This is what we do in this section.

Referring back to section 5.1, we make the following instantiations: The alphabet that indexes approximations is T, the alphabet of E. The algebra $\mathcal{R}$ is the algebra of languages over alphabet T. The algebra $\mathcal{S}$ is the powerset algebra with carrier set $2^{SM.E}$ and underlying monoid $(SM.E, \circ, 1)$. (See theorem 160.) The event that we "approximate" is not E but the set of c-classes that form E, as defined shortly.

The function ζ_c is defined by, for all a in T,

$$(164) \quad \zeta_c.a = \{E_c(a)\} \ .$$

(We have added the subscript c to emphasise that this is a particular instance of the function ζ in section 5.1.)

Note that the extension of ζ_c to words, as defined by equations (117) and (118), satisfies, for all w in T^*,

$$(165) \quad \zeta_c.w = \{E_c(w)\} \ .$$

The easy proof is by induction on the length of words:

$$\begin{array}{ll}
 & \zeta_c.\varepsilon \\
= & \{ \quad \text{definition of the extension of } \zeta_c \text{ to words} \quad \} \\
 & 1_{\mathcal{S}} \\
= & \{ \quad \text{definition of powerset algebra } \mathcal{S} \quad \} \\
 & \{1\} \\
= & \{ \quad E_c \text{ is a congruence relation, } \varepsilon \text{ is the unit of } T^* \quad \} \\
 & \{E_c(\varepsilon)\}
\end{array}$$

and

$$\begin{array}{ll}
 & \zeta_c.(au) \\
= & \{ \quad \text{definition of the extension of } \zeta_c \text{ to words} \quad \} \\
 & \zeta_c.a \circ \zeta_c.u \\
= & \{ \quad \text{definition of } \zeta_c \text{ and induction hypothesis} \quad \} \\
 & \{E_c(a)\} \circ \{E_c(u)\} \\
= & \{ \quad \text{distributivity of product over set union in} \\
 & \quad\quad \text{the powerset algebra of the syntactic monoid} \quad \} \\
 & \{E_c(a) \circ E_c(u)\} \\
= & \{ \quad E_c \text{ is a congruence relation} \quad \} \\
 & \{E_c(au)\} \ .
\end{array}$$

Instantiating ζ as above in the definitions (119) and (120), we get, for all languages U,

(166) $\zeta_c^\flat.U \;=\; \langle\cup u : u{\in}U : \{E_c(u)\}\rangle$

and, for all sets of c-classes C,

(167) $\zeta_c^\sharp.C \;=\; \{u \mid E_c(u) \in C\}$.

Thus $\zeta_c^\flat.U$ is a set of c-classes whilst $\zeta_c^\sharp.C$ is the union of all elements in the c-classes in the set C.

For later reference, let us record the fact that $\zeta_c^\flat$ distributes through concatenation of languages:

Lemma 168 For all languages X and Y,

$$\zeta_c^\flat.(X{\cdot}Y) \;=\; \zeta_c^\flat.X \circ \zeta_c^\flat.Y \;\; .$$

Proof We have, for all words u and v,

$$\begin{array}{ll}
 & \zeta_c.uv \\
= & \quad \{ \quad (165) \quad \} \\
 & \{E_c(uv)\} \\
= & \quad \{ \quad E_c \text{ is a congruence relation:} \quad \} \\
 & \{E_c(u){\circ}E_c(v)\} \\
= & \quad \{ \quad \text{definition of product in the powerset algebra} \quad \} \\
 & \{E_c(u)\}{\circ}\{E_c(v)\} \\
= & \quad \{ \quad (165) \quad \} \\
 & \zeta_c.u \circ \zeta_c.v
\end{array}$$

The lemma follows from theorem 122 with ζ instantiated to ζ_c

□

Example 169 (Running Example: Modulo Addition) Fig. 9 is the reduced finite-state automaton of a regular language. In words, the language E recognised is the set of strings of as and bs such that the difference between the number of as and the number of bs is not divisible by 6. Note that we use English rather than regular expressions to describe the language E because this is a classic example of how complex regular expressions can become. Were we to provide a regular expression denoting the language, it would be unlikely to be enlightening to the reader!

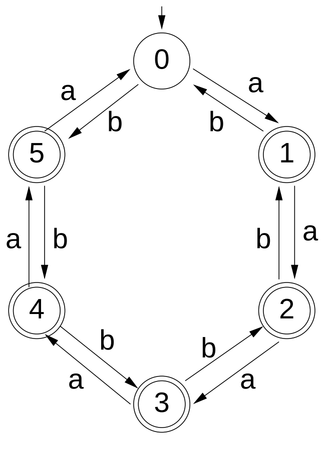


Figure 9: Automaton recognising indivisibility

The syntactic monoid of this language is a group: the group $\mathbb{Z}_6$ of addition of numbers in the set {0,1,2,3,4,5} modulo 6 first introduced in example 16.

Given a subset N of {0,1,2,3,4,5}, an automaton that recognises the set of words such that the difference between the number of as and bs modulo 6 in each word is an element of N is obtained simply by changing the set of final states to be those labelled by an element of N. (The resulting automaton may not be minimal but that is of no consequence.) For instance, if the given set N is the empty set, the set of final states would also be chosen to be the empty set. The syntactic monoid is either $\mathbb{Z}_6$, $\mathbb{Z}_3$, $\mathbb{Z}_2$ or $\mathbb{Z}_1$. For instance, if N is {0,1,4,5}, the syntactic monoid is $\mathbb{Z}_6$ and if N is {0,2,4} the syntactic monoid is $\mathbb{Z}_2$.

The function $\zeta_c^\flat$ maps an arbitrary language into a subset of {0,1,2,3,4,5}: the number i is an element of $\zeta_c^\flat.X$ if there is a word in X such that the difference between the number of as and bs in the word is equal to i modulo 6. Conversely, the function $\zeta_c^\sharp$ maps a subset N of {0,1,2,3,4,5} into a set of words: the set of all words such that the difference between the number of as and the number of bs modulo 6 in the word is an element of N.

In example 16, we explained how to compute factors in this group. Using theorem 172 below, we can easily calculate factors of the language E as well as factors of factors of E. For instance, if X is the set of words such that the difference between the number of as and the number of bs modulo 6 in the word is 1 or 2, the factor $X\backslash E$ is $\zeta_c^\sharp.\frac{\{1,2,3,4,5\}}{\{1,2\}}$, which equals $\zeta_c^\sharp.\{0,1,2,3\}$. (Refer back to example 16 for the calculation of $\frac{\{1,2,3,4,5\}}{\{1,2\}}$.) That is, $X\backslash E$ is the set of words such that the difference between the number of as and the number of bs modulo 6 in the word is 0, 1, 2 or 3.

This example can, of course, be generalised by replacing "6" by an arbitrary number

m. As mentioned earlier, each left factor of $\neg\{0\}$ corresponds to a subset of $\{0\,..\,m{-}1\}$. There are of course $\binom{m}{n}$ subsets of $\{0\,..\,m{-}1\}$ of size n. Subsets are divided into groups. For example, when $m=6$, the subsets $\{0,2,4\}$ and $\{1,3,5\}$ form one group, and $\{0,1,4\}$, $\{1,2,5\}$, $\{2,3,0\}$, $\{3,4,1\}$, $\{4,5,2\}$ and $\{5,0,3\}$ form another group. (We leave the reader the exercise of formulating precisely how many elements there are in each group given one element of the group.) The number of times that the factor matrix of $\{0\}$ occurs as a submatrix of $\neg\{0\}$ is the number of groups of size m, viz.

$$\langle \Sigma k : 0 \le n < m : \lfloor \binom{m}{n}/_{m} \rfloor \rangle \quad .$$

For example, when $m=6$, the the factor matrix of $\{0\}$ occurs as a submatrix of $\neg\{0\}$ a total of

$$0+1+2+3+2+1+0$$

times (i.e. 9 times). Similar calculations enable the entire structure of the factor matrix of a language defined by a subset N of $\{0\,..\,m{-}1\}$ to be predicted. However, only for the very simplest case ($m=1$, $m=2$ or $m=3$), is it feasible to display the matrix in the conventional way; in other cases, it is just too big for human consumption!
□

The fact that ζ_c returns singleton sets means that $\zeta_c^\flat$ is the left inverse of $\zeta_c^\sharp$:

Lemma 170 For all sets of c-classes C,

$$\zeta_c^\flat.(\zeta_c^\sharp.C) \;=\; C$$

Proof

$$
\begin{array}{cl}
 & \zeta_c^\flat.(\zeta_c^\sharp.C) \\
= & \{ \quad \text{definitions (167) and (166)} \quad \} \\
 & \langle \cup u \;:\; u \in \{u \mid E_c(u) \in C\} \;:\; \{E_c(u)\} \rangle \\
= & \{ \quad \text{set comprehension} \quad \} \\
 & \langle \cup u : E_c(u) \in C : \{E_c(u)\} \rangle \\
= & \{ \quad C \text{ is a set of } c\text{-classes, definition of } c\text{-class} \quad \} \\
 & C \quad .
\end{array}
$$

□

Conversely, when restricted to factors of E, $\zeta_c^\sharp$ is the inverse of $\zeta_c^\flat$:

Lemma 171 The factors of E are closed elements of the Galois connection with adjoints $\zeta_c^\flat$ and $\zeta_c^\sharp$. That is, for all factors F of E,

$$\zeta_c^\sharp.(\zeta_c^\flat.F) \;=\; F$$

Proof Let F be a factor of E. Then

$$\begin{array}{ll}
 & \zeta_c^\sharp.(\zeta_c^\flat.F) \\
= & \{ \quad \text{definitions: (166) and (167)} \quad \} \\
 & \langle \cup v : E_c(v) \in \langle \cup u : u\in F : \{E_c(u)\}\rangle : \{v\}\rangle \\
= & \{ \quad \text{associativity and symmetry of set union} \quad \} \\
 & \langle \cup u : u\in F : \langle \cup v : E_c(u) = E_c(v) : \{v\}\rangle\rangle \\
= & \{ \quad E_c \text{ is an equivalence relation,} \\
 & \quad\;\; \text{so } E_c(u) = E_c(v) \equiv v\in E_c(u) \quad \} \\
 & \langle \cup u : u\in F : \langle \cup v : v\in E_c(u) : \{v\}\rangle\rangle \\
= & \{ \quad \text{set comprehension} \quad \} \\
 & \langle \cup u : u\in F : E_c(u)\rangle \\
= & \{ \quad F \text{ is a factor of } E, \\
 & \quad\;\; \text{so is a union of } c\text{-classes of } E \text{ (theorem 159)} \quad \} \\
 & F \;.
\end{array}$$

□

The combination of lemmas 170 and 171 is that $\zeta_c^\flat$ and $\zeta_c^\sharp$ are inverse functions when restricted to factors of E and sets of c-classes of E.

As a corollary, we get:

Theorem 172 Let T be an alphabet of symbols and let $\mathcal{R}$ denote the powerset regular algebra with underlying monoid T^*. Let E be a language over the alphabet T with syntactic monoid $SM.E$ and let $\mathcal{S}.E$ denote the powerset regular algebra with underlying monoid $SM.E$. Let ζ_c be the function that maps symbol a in T to $\{E_c(a)\}$ as in (164). Then

$$(173) \quad |\overline{E}| \;=\; \zeta_c^\sharp \bullet ((\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \,\dot{\cup}\, (\zeta_c^\flat \bullet \mathbf{L}_{min}.E))^* \quad .$$

Proof

$$|\overline{E}|$$

$= \quad \{ \quad$ theorem 158 $\quad \}$

$(\mathbf{C}_{min}.E \ \dot{\cup} \ \mathbf{L}_{min}.E)^*$

$= \quad \{ \quad$ lemma 171 and theorem 26 $\quad \}$

$\zeta_c^\sharp \bullet (\zeta_c^\flat \bullet (\mathbf{C}_{min}.E \ \dot{\cup} \ \mathbf{L}_{min}.E))^*$

$= \quad \{ \quad$ distributivity $\quad \}$

$\zeta_c^\sharp \bullet ((\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \ \dot{\cup} \ (\zeta_c^\flat \bullet \mathbf{L}_{min}.E))^* \quad .$

□

Theorem 172 expresses formally how the syntactic monoid is used to calculate the factor matrix of a language E given the minimal constant approximation $\mathbf{C}_{min}.E$ and the minimal linear approximation $\mathbf{L}_{min}.E$.

To understand the significance of the theorem it is vital to observe that the subterm

$$(\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \ \dot{\cup} \ (\zeta_c^\flat \bullet \mathbf{L}_{min}.E)$$

is a matrix in the powerset algebra with underlying monoid $SM.E$. Thus the calculation of the reflexive, transitive closure of the matrix

$$(\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \ \dot{\cup} \ (\zeta_c^\flat \bullet \mathbf{L}_{min}.E)$$

involves the use of an algorithm that computes products and unions of *finite* sets of c-classes. In particular, at the level of elements, the star operator maps a finite set to a finite set — unlike for languages where the operator typically maps finite sets to infinite sets. The leftmost term $\zeta_c^\sharp$ then maps the sets of c-classes into languages. The first of several examples is example 199 below. See, in particular, fig. 12.

Closure algorithms that can be used are well-known and are documented in [BC75]. But it is not necessary to use a closure algorithm at all since the elements of the matrix

$$((\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \ \dot{\cup} \ (\zeta_c^\flat \bullet \mathbf{L}_{min}.E))^*$$

are the factors of $\zeta_c^\flat.E$, and these can be calculated directly in the algebra $\mathcal{S}$. Formally, this is expressed by theorem 193 below.

Lemma 174 For all factors F of E and all sets of c-classes C of E,

$$C \subseteq \zeta_c^\flat.F \ \equiv \ \zeta_c^\sharp.C \subseteq F \quad .$$

(Note the reversal of $^\flat$ and $^\sharp$. This says that $\zeta_c^\flat$ is the *upper* adjoint and $\zeta_c^\sharp$ is the *lower* adjoint in a Galois connection of the two posets.)

Proof We have:

$$
\begin{array}{ll}
 & C \subseteq \zeta_c^\flat.F \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & \zeta_c^\sharp.C \subseteq \zeta_c^\sharp.(\zeta_c^\flat.F) \\
= & \{ \quad \text{lemma 171} \quad \} \\
 & \zeta_c^\sharp.C \subseteq F \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & \zeta_c^\flat.(\zeta_c^\sharp.C) \subseteq \zeta_c^\flat.F \\
= & \{ \quad \text{lemma 170} \quad \} \\
 & C \subseteq \zeta_c^\flat.F \quad .
\end{array}
$$

The lemma follows by mutual implication.

□

Lemma 175 For all factors F and G of E,

$$F \subseteq G \;\equiv\; \zeta_c^\flat.F \subseteq \zeta_c^\flat.G \quad .$$

Proof Straight-forward combination of lemmas 171 and 174.

□

Lemmas 170, 171, 174 and 175 express in precise, calculational rules the fact that the lattice of factors of E is represented by a sublattice of the set of sets of c-classes of E. In particular, if E is a regular language and it is required to determine whether or not $F \subseteq G$, for given factors F and G, it suffices to compare the *finite* sets $\zeta_c^\flat.F$ and $\zeta_c^\flat.G$. Since we also know that $\zeta_c^\flat$ is a regular homomorphism (instantiate theorem 122 with $\zeta := \zeta_c$), the computation of factors of factors is reduced to computations with finite sets using lemma 22. Specifically, by instantiating lemma 22 with $\zeta := \zeta_c$ we get:

Lemma 176 For all factors F of E and all languages X and Y,

(177) $\quad X \backslash F / Y \;=\; \zeta_c^\sharp.(\zeta_c^\flat.X \backslash \zeta_c^\flat.F / \zeta_c^\flat.Y)$

where $\zeta_c^\flat$ and $\zeta_c^\sharp$ are as defined in (166) and (167).

□

On the right side of (177), $\zeta_c^\flat.X$, $\zeta_c^\flat.F$ and $\zeta_c^\flat.Y$ are *finite* sets of c-classes (assuming E is a regular language) and the under and over operators are evaluated in the algebra $\mathcal{S}.E$ (i.e. the powerset algebra with underlying monoid the syntactic monoid of E). Their computation thus involves a straight-forward comparison of finite sets.

Below we establish additional properties that reverse the roles of $\zeta_c^\flat$ and $\zeta_c^\sharp$. In this way, we establish the precise relationship between calculations with factors and calculations with sets of c-classes.

We begin with a negative result: *unlike* the function $\zeta_c^\flat$ (lemma 168), the function $\zeta_c^\sharp$ does *not* distribute through product. We establish this fact by means of an example.

Example 178 (Running Example: The Language $(aa)^*$) Consider the language $(aa)^*$ over the alphabet $\{a\}$. Its factor matrix (including inadmissible entries) was given in example 82. In fig. 10(b), we show its syntactic monoid. This has two elements, which we have named "1" and "a".

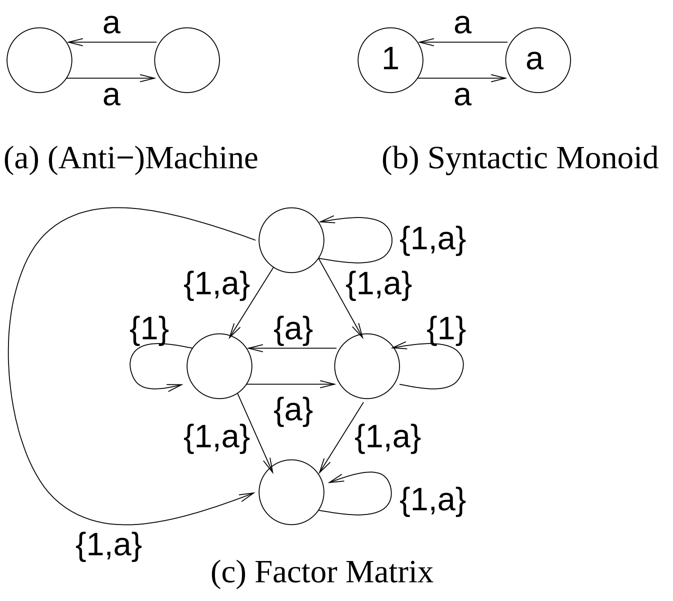


Figure 10: Aspects of the Language $(aa)^*$

Fig. 10(c) shows the matrix $\overline{|\zeta_c^\flat.(aa)^*|}$, i.e. the factor matrix[6] where each entry is expressed as a set of c-classes. Now, in the syntactic monoid, $a \circ a = 1$. So, in the regular algebra $\mathcal{S}$, $\{a\} \circ \{a\} = \{1\}$. Also, $\zeta_c^\sharp.\{1\} = (aa)^*$ and $\zeta_c^\sharp.\{a\} = (aa)^* a$. If $\zeta_c^\sharp$ were to distribute over product, we would have

$$\zeta_c^\sharp.\{a\} \cdot \zeta_c^\sharp.\{a\} = \zeta_c^\sharp.(\{a\} \circ \{a\}) = \zeta_c^\sharp.\{1\} = (aa)^* \quad .$$

However, this is not the case. We have:

$$\zeta_c^\sharp.\{a\} \cdot \zeta_c^\sharp.\{a\} = (aa)^* a (aa)^* a$$

[6]For consistency with example 82 we deviate from our usual practice and include edges that are inadmissible with respect to the language $(aa)^*$: these are the edges to or from the top and bottom nodes.

and

$$\zeta_c^\sharp.\{1\} = (aa)^* \quad .$$

The two right sides are clearly not equal. So we conclude that, in general, $\zeta_c^\sharp$ does not distribute over product.
□

In spite of the above negative result, we can establish an inclusion.

Lemma 179 For all events C and D in the algebra $\mathcal{S}$ (i.e. sets of c-classes),

$$\zeta_c^\sharp.C \cdot \zeta_c^\sharp.D \subseteq \zeta_c^\sharp.(C \circ D) \quad .$$

Proof We have, for all events C and D in the algebra $\mathcal{S}$ and all words u in T^*,

$$\begin{array}{ll} & \zeta_c^\sharp.C \cdot \zeta_c^\sharp.D \subseteq \zeta_c^\sharp.(C \circ D) \\ = & \{ \quad \text{Galois connection of } \zeta_c^\flat \text{ and } \zeta_c^\sharp \quad \} \\ & \zeta_c^\flat.(\zeta_c^\sharp.C \cdot \zeta_c^\sharp.D) \subseteq C \circ D \\ = & \{ \quad \text{distributity: lemma 168} \quad \} \\ & \zeta_c^\flat.(\zeta_c^\sharp.C) \circ \zeta_c^\flat.(\zeta_c^\sharp.D) \subseteq C \circ D \\ = & \{ \quad \text{lemma 170} \quad \} \\ & \text{true} \quad . \end{array}$$

□

As a consequence of lemma 179, $\zeta_c^\sharp$ does distribute through factorisation.

Lemma 180 For all events B, C and D in the algebra $\mathcal{S}$ (i.e. sets of c-classes),

(181) $\zeta_c^\sharp.(B\backslash C) = \zeta_c^\sharp.B \backslash \zeta_c^\sharp.C$,

(182) $\zeta_c^\sharp.(C/D) = \zeta_c^\sharp.C / \zeta_c^\sharp.D$, and

(183) $\zeta_c^\sharp.(B\backslash C/D) = \zeta_c^\sharp.B \backslash \zeta_c^\sharp.C / \zeta_c^\sharp.D$.

Proof We show the proof of (181). The other two properties are proved similarly. First,

$$\begin{array}{ll} & \zeta_c^\sharp.(B\backslash C) \supseteq \zeta_c^\sharp.B \backslash \zeta_c^\sharp.C \\ = & \{ \quad \zeta_c^\sharp \text{ is upper adjoint, } \zeta_c^\flat \text{ is lower adjoint, factors} \quad \} \\ & C \supseteq B \circ \zeta_c^\flat.(\zeta_c^\sharp.B \backslash \zeta_c^\sharp.C) \end{array}$$

$=$ { lemma 170 }

$C \supseteq \zeta_c^\flat.(\zeta_c^\sharp.B) \circ \zeta_c^\flat.(\zeta_c^\sharp.B \setminus \zeta_c^\sharp.C)$

$=$ { $\zeta_c^\flat$ is a monoid homomorphism (theorem 122) }

$C \supseteq \zeta_c^\flat.(\zeta_c^\sharp.B \cdot \zeta_c^\sharp.B \setminus \zeta_c^\sharp.C)$

$\Leftarrow$ { factors, monotonicity of $\zeta_c^\flat$ }

$C \supseteq \zeta_c^\flat.(\zeta_c^\sharp.C)$

$=$ { $\zeta_c^\flat$ is lower adjoint with upper adjoint $\zeta_c^\sharp$ }

true .

Second,

$\zeta_c^\sharp.(B\backslash C) \subseteq \zeta_c^\sharp.B \setminus \zeta_c^\sharp.C$

$=$ { factors }

$\zeta_c^\sharp.B \cdot \zeta_c^\sharp.(B\backslash C) \subseteq \zeta_c^\sharp.C$

$\Leftarrow$ { lemma 179 with $C,D := B\,,B\backslash C$ }

$\zeta_c^\sharp.(B \circ B\backslash C) \subseteq \zeta_c^\sharp.C$

$\Leftarrow$ { monotonicity of $\zeta_c^\sharp$, factors }

true .

The lemma follows by the anti-symmetry of set equality.
□

As we have just seen, the function $\zeta_c^\sharp$ does not distribute through product but does distribute through factorisation (lemma 180). Likewise, the function $\zeta_c^\flat$ distributes through product (lemma 168) but —in general— it does not distribute through factorisation. However, if we examine closely the properties of $\zeta_c^\sharp$ that allowed us to prove lemmas 179 and 180 (for example, it is the upper adjoint in a Galois connection with lower adjoint $\zeta_c^\flat$) we see that the function $\zeta_c^\flat$ enjoys the same properties when its domain is restricted to the factors of E: see lemmas 174 and 171. Thus we have:

Lemma 184 For all factors F, G and H of E,

(185) $\zeta_c^\flat.(F\backslash G) = \zeta_c^\flat.F \setminus \zeta_c^\flat.G$,

(186) $\zeta_c^\flat.(G/H) = \zeta_c^\flat.G \,/\, \zeta_c^\flat.H$, and

(187) $\zeta_c^\flat.(F\backslash G/H) = \zeta_c^\flat.F \setminus \zeta_c^\flat.G \,/\, \zeta_c^\flat.H$.

Proof Repeat the proof of lemma 180 with $\zeta_c^\sharp$ replaced by $\zeta_c^\flat$ (and vice-versa), changing hints as indicated above.
□

Lemma 188 For all left factors i and j of E,

$$\zeta_c^\flat.i \;=\; \zeta_c^\flat.E \,/\, \zeta_c^\flat.i\rhd$$

and

$$\zeta_c^\flat.(i\backslash j) \;=\; \zeta_c^\flat.i \,\backslash\, \zeta_c^\flat.E \,/\, \zeta_c^\flat.j\rhd \;\; .$$

Thus $\zeta_c^\flat.i$ is a left factor of $\zeta_c^\flat.E$, and $\zeta_c^\flat.(i\backslash j)$ is a factor of $\zeta_c^\flat.E$.

Proof For all sets of c-classes C and left factors i of E,

$$\begin{array}{ll}
 & C \subseteq \zeta_c^\flat.i \\
= & \{ \quad i = i\rhd\lhd = E/i\rhd \quad \} \\
 & C \subseteq \zeta_c^\flat.(E/i\rhd) \\
= & \{ \quad \text{lemma 174, factors} \quad \} \\
 & \zeta_c^\sharp.C \cdot i\rhd \subseteq E \\
= & \{ \quad i\rhd \text{ is a factor, lemma 171 (with } F{:=}i\rhd) \quad \} \\
 & \zeta_c^\sharp.C \cdot \zeta_c^\sharp.(\zeta_c^\flat.i\rhd) \subseteq E \\
= & \{ \quad \text{lemma (179)} \quad \} \\
 & \zeta_c^\sharp.(C \cdot \zeta_c^\flat.i\rhd) \subseteq E \\
= & \{ \quad \text{lemma 174, factors} \quad \} \\
 & C \subseteq \zeta_c^\flat.E \,/\, \zeta_c^\flat.i\rhd \;\; .
\end{array}$$

Thus, the first equality follows by indirect equality. The second equality follows immediately from (185) and the above equality (with $i{:=}j$).
□

Lemma 188 establishes that $\zeta_c^\flat$ maps left factors of E to left factors of $\zeta_c^\flat.E$. When restricted to left factors, function $\zeta_c^\flat$ has inverse $\zeta_c^\sharp$ (lemmas 171 and 170). In order to show that it is an isomorphism it remains to show that $\zeta_c^\flat$ (restricted to left factors of E) is onto the set of left factors of $\zeta_c^\flat.E$ and, vice-versa, $\zeta_c^\sharp$ (restricted to left factors of $\zeta_c^\flat.E$) is onto the set of left factors of E. This is shown in lemma 192.

Lemma 189 For all c-classes B and D,

(190) $\zeta_c^\flat.E/D \;=\; \zeta_c^\flat.(E/\zeta_c^\sharp.D)$, and

(191) $B\backslash\zeta_c^\flat.E/D \;=\; \zeta_c^\flat.(\zeta_c^\sharp.B\backslash E/\zeta_c^\sharp.D)$.

Proof First,

$$\begin{array}{ll}
 & \zeta_c^\flat.E/D \\
= & \{\quad \text{lemma 170} \quad\} \\
 & \zeta_c^\flat.(\zeta_c^\sharp.(\zeta_c^\flat.E/D)) \\
= & \{\quad (181) \quad\} \\
 & \zeta_c^\flat.((\zeta_c^\sharp.(\zeta_c^\flat.E))/(\zeta_c^\sharp.D)) \\
= & \{\quad E \text{ is a factor of itself, lemma 171} \quad\} \\
 & \zeta_c^\flat.(E/(\zeta_c^\sharp.D)) \ .
\end{array}$$

The second part is proved similarly, using (183) instead of (181).
□

Lemma 192 A set of c-classes C is a left factor of $\zeta_c^\flat.E$ equivales $C=\zeta_c^\flat.i$ for some left factor i of E. A set of c-classes C is a right factor of $\zeta_c^\flat.E$ equivales $C=\zeta_c^\flat.i$ for some right factor i of E. The set of c-classes C is a factor of $\zeta_c^\flat.E$ equivales $C=\zeta_c^\flat.i\backslash\zeta_c^\flat.j$ for some left factors i and j of E.

Proof We prove the first part by mutual implication. (Dummy D ranges over sets of c-classes of E and dummy i ranges over left factors of E.)

$$\begin{array}{ll}
 & C \text{ is a left factor of } \zeta_c^\flat.E \\
= & \{\quad \text{definition} \quad\} \\
 & \langle \exists D :: C=\zeta_c^\flat.E/D \rangle \\
= & \{\quad (190) \quad\} \\
 & \langle \exists D :: C=\zeta_c^\flat.(E/(\zeta_c^\sharp.D)) \rangle \\
\Rightarrow & \{\quad (E/(\zeta_c^\sharp.D)) \text{ is a left factor of } E,\ i:=E/(\zeta_c^\sharp.D) \quad\} \\
 & \langle \exists i :: C=\zeta_c^\flat.i \rangle \\
= & \{\quad \text{lemma 188} \quad\} \\
 & \langle \exists i :: C=\zeta_c^\flat.E/\zeta_c^\flat.i\rhd \rangle \\
\Rightarrow & \{\quad \text{definition of left factor} \quad\} \\
 & C \text{ is a left factor of } \zeta_c^\flat.E \ .
\end{array}$$

The second part is proved similarly. For the third part, we calculate as follows. (Dummies B and D range over sets of c-classes of E and dummies i and j range over left factors of E.)

$$
\begin{array}{ll}
 & C \text{ is a factor of } \zeta_c^\flat.E \\
= & \quad \{ \quad \text{definition} \quad \} \\
 & \left\langle \exists B,D :: C = B \backslash \zeta_c^\flat.E / D \right\rangle \\
= & \quad \{ \quad (191) \quad \} \\
 & \left\langle \exists B,D :: C = \zeta_c^\flat.(\zeta_c^\sharp.B \backslash E / \zeta_c^\sharp.D) \right\rangle \\
\Rightarrow & \quad \{ \quad (\, \zeta_c^\sharp.B \backslash E / \zeta_c^\sharp.D\,) \text{ is a factor of } E,\ (69) \quad \} \\
 & \left\langle \exists i,j :: C = \zeta_c^\flat.(i\backslash j) \right\rangle \\
= & \quad \{ \quad \text{lemma 184} \quad \} \\
 & \left\langle \exists i,j :: C = \zeta_c^\flat.i \backslash \zeta_c^\flat.j \right\rangle \\
\Rightarrow & \quad \{ \quad \zeta_c^\flat.i \text{ and } \zeta_c^\flat.j \text{ are left factors of } \zeta_c^\flat.E \\
 & \quad\quad \text{(first part of this lemma);} \\
 & \quad\quad \text{factors of factors are factors} \quad \} \\
 & C \text{ is a factor of } \zeta_c^\flat.E \quad .
\end{array}
$$

□

The conclusion is that the factor matrix of E is represented by the factor matrix of $\zeta_c^\flat.E$ as expressed formally by:

Theorem 193

(194) $|\overline{E}| \;=\; \zeta_c^\sharp \bullet |\overline{\zeta_c^\flat.E}| \bullet \zeta_c^\flat \times \zeta_c^\flat$.

Conversely,

(195) $|\overline{\zeta_c^\flat.E}| \;=\; \zeta_c^\flat \bullet |\overline{E}| \bullet \zeta_c^\sharp \times \zeta_c^\sharp$.

Finally,

(196) $\zeta_c^\flat \bullet |\overline{E}| \;=\; |\overline{\zeta_c^\flat.E}| \bullet \zeta_c^\flat \times \zeta_c^\flat$

and

(197) $\zeta_c^\sharp \bullet |\overline{\zeta_c^\flat.E}| \;=\; |\overline{E}| \bullet \zeta_c^\sharp \times \zeta_c^\sharp$.

Proof Recall that $|\overline{E}|$ is, by definition, the under operator ($\backslash$) in the algebra of languages restricted to the left factors of E. Similarly, $|\overline{\zeta_c^\flat.E}|$ is the under operator in the algebra $\mathcal{S}.E$ (the powerset algebra with underlying monoid the syntactic monoid of E) restricted to the left factors of $\zeta_c^\flat.E$. Equation (194) is thus the statement that for all left factors i and j of E, $\zeta_c^\flat.i$ and $\zeta_c^\flat.j$ are left factors of $\zeta_c^\flat.E$ and $\zeta_c^\sharp.(\zeta_c^\flat.i \backslash \zeta_c^\flat.j) = i\backslash j$. (The leftmost under operator is in the algebra $\mathcal{S}.E$ and the rightmost under operator is in the algebra of languages.) It is thus a combination of lemmas 188, 180 and 171. Similarly, equation (195) is a combination of lemmas 180, 188 and 170. Equations (196) and (197) state, respectively, that the factor matrix of $\zeta_c^\flat.E$ is obtained by applying the function $\zeta_c^\flat$ to the entries of the factor matrix of E, and the factor matrix of E is obtained by applying the function $\zeta_c^\sharp$ to the entries of the factor matrix of $\zeta_c^\flat.E$.
□

Corollary 198

$$|\overline{\zeta_c^\flat.E}| \;=\; ((\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \mathbin{\dot{\cup}} (\zeta_c^\flat \bullet \mathbf{L}_{min}.E))^* \bullet \zeta_c^\sharp{\times}\zeta_c^\sharp \quad .$$

Proof The theorem is a simple combination of (195) and (173):

$$\begin{aligned}
& |\overline{\zeta_c^\flat.E}| \\
= & \qquad \{ \quad (195) \quad \} \\
& \zeta_c^\flat \bullet |\overline{E}| \bullet \zeta_c^\sharp{\times}\zeta_c^\sharp \\
= & \qquad \{ \quad (173) \quad \} \\
& \zeta_c^\flat \bullet \zeta_c^\sharp \bullet ((\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \mathbin{\dot{\cup}} (\zeta_c^\flat \bullet \mathbf{L}_{min}.E))^* \bullet \zeta_c^\sharp{\times}\zeta_c^\sharp \\
= & \qquad \{ \quad (170) \quad \} \\
& ((\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \mathbin{\dot{\cup}} (\zeta_c^\flat \bullet \mathbf{L}_{min}.E))^* \bullet \zeta_c^\sharp{\times}\zeta_c^\sharp \quad .
\end{aligned}$$

□

Theorems 172 and 198 form the basis of how we use the syntactic monoid to explore the structure of the factor matrix $|\overline{E}|$, for a given event E, before any attempt to calculating regular expressions denoting each of its entries. . The matrices $\mathbf{C}_{min}.E$ and $\mathbf{L}_{min}.E$ are first computed, and then (by applying the function $\zeta_c^\flat$ to each entry) converted to matrices of c-classes. The closure of this matrix is then calculated in the algebra $\mathcal{S}$. That is, we compute

$$((\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \mathbin{\dot{\cup}} (\zeta_c^\flat \bullet \mathbf{L}_{min}.E))^*$$

in the algebra —finite— algebra $\mathcal{S}$. (The difference between this and the right side of the equation in corollary 198 is just the indexing of matrix elements: in the former case, elements are indexed by left factors of $\zeta_c^\flat.E$ and in the latter case by left factors of E.) Factors of factors of $\zeta_c^\flat.E$ can then be computed and their factor matrices identified. The (1–1) correspondence between factors of E and factors of $\zeta_c^\flat.E$ (lemmas (170) and (171)) means that the results can then be used to compute the factor graphs of the factors of E.

7.2 Use of the Syntactic Monoid

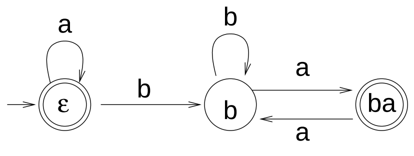
This section illustrates the theorems and lemmas in section 7.1 by means of examples.

Example 199 Consider the language E recognised by the machine shown in fig. 11(a). Its anti-machine and syntactic monoid are shown in figs. 11(b) and (c), respectively. (Inadmissible nodes are omitted in all three of these figures. See below.) The nodes of each graph are labelled by representative elements: those of the machine are labelled by a representative element of the l-class to which the node corresponds; those of the anti-machine are labelled by a representative element of the r-class to which the node corresponds ; and those of the syntactic monoid are labelled by a representative element of the syntactic monoid to which the node corresponds. (The representative of an r-class is the *reverse* of a word from the start state to the corresponding node in the anti-machine. In this case, the simplicity of the anti-machine means that it is a poor illustration: in fig. 11, the nodes are labelled ε, a and b. These are the shortest words from the start state to the corresponding node but all are the reverse of themselves. A better illustration would be to take, for example, ab as the representative of the r-class containing b. This is the reverse of the word ba from the start state to the node labelled b in fig. 11. But this node is inadmissible. For the other two states, all representative elements are their own reverse.)

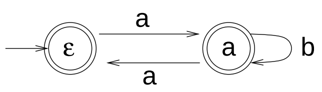
The machine and anti-machine each have one *inadmissible* node: a node from which there are no paths to the final state. In the machine, the representative element of the inadmissible node is bab and in the anti-machine it is b. Correspondingly, there is one inadmissible element of the syntactic monoid: a c-class that is an element of inadmissible factors only. Our practice is to always omit inadmissible elements from the figures since it is always safe to do so.

In the following table, we compute the value of $l\rhd$ and $r\lhd$ for each l-class l and each r-class r.

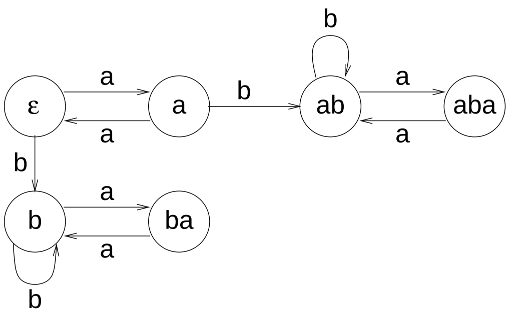
l-class l	$l\rhd$	r-class r	$r\lhd$
$E_l(\varepsilon)$	$E_r(\varepsilon)\cup E_r(a)$	$E_r(\varepsilon)$	$E_l(\varepsilon)\cup E_l(ba)$
$E_l(b)$	$E_r(a)$	$E_r(a)$	$E_l(\varepsilon)\cup E_l(b)$
$E_l(ba)$	$E_r(\varepsilon)$		



(a) Machine



(b) Anti−machine



(c) Semigroup (Syntactic Monoid)

Figure 11: Machine, Anti-Machine and Syntactic Monoid

From this table, we deduce that E has five left factors, which we divide into two sets: the empty set $\emptyset$ and $\{a,b\}^*$ form one set, the *inadmissible* left factors, and $E_l(\varepsilon)$, $E_l(\varepsilon)\cup E_l(b)$, and $E_l(\varepsilon)\cup E_l(ba)$ form the second set, the *admissible* left factors. (The language $\{a,b\}^*$ is, of course, the union of all l-classes.)

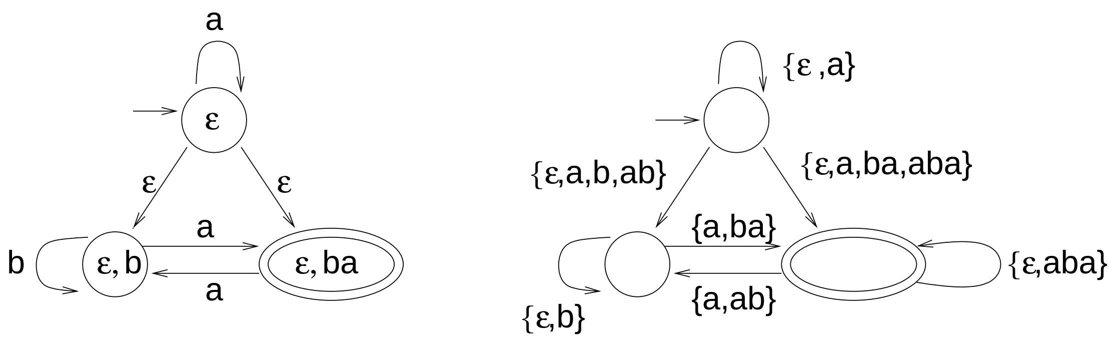
The corresponding right factors of the inadmissible factors are, respectively, $\{a,b\}^*$ and $\emptyset$. The corresponding right factors of the admissible left factors are, respectively, $E_r(\varepsilon)\cup E_r(a)$, $E_r(a)$ and $E_r(\varepsilon)$.

Conway's best constant+linear approximation to the factor matrix can now be easily deduced and then reduced to the factor graph. This is shown in fig. 12(a); given start and final nodes (indicated in the usual way), this is a non-deterministic recogniser of the language E. (Nodes are labelled by representatives of the l-classes.)

Displayed as a two-dimensional array, and omitting inadmissible rows and columns, the matrix

$$(\zeta_c^\flat \bullet \mathbf{C}_{min}.E) \,\dot\cup\, (\zeta_c^\flat \bullet \mathbf{L}_{min}.E)$$

in theorem 172 is thus:



(a) Factor Graph (b) Factor Matrix

Figure 12: Factor Graph and its Closure in the Algebra $\mathcal{S}$

$$\begin{bmatrix} \{a\} & \{\varepsilon\} & \{\varepsilon\} \\ \emptyset & \{b\} & \{a\} \\ \emptyset & \{a\} & \emptyset \end{bmatrix}$$

Note that we choose a shortest word in the corresponding equivalence class to name elements of the syntactic monoid.

The closure of this matrix —the matrix $|\overline{\zeta^{\flat}_c.E}|$— is calculated by chasing paths in the factor graph. The result is shown in fig. 12(b). Hopefully the reader can see how easily this is done: each entry is a set of c-classes, and the representative element u of a c-class is an element of the (i,j)th entry if there is a path spelling u from i to j in the factor graph. More conventionally, displayed as a two-dimensional array, it is the following:

$$\begin{bmatrix} \{\varepsilon,a\} & \{\varepsilon,a,b,ab\} & \{\varepsilon,a,ba,aba\} \\ \emptyset & \{\varepsilon,b\} & \{a,ba\} \\ \emptyset & \{a,ab\} & \{\varepsilon,aba\} \end{bmatrix}$$

Note that E is represented by $\{\varepsilon,a,ba,aba\}$. That is,

$$E \;=\; E_c(\varepsilon)\cup E_c(a)\cup E_c(ba)\cup E_c(aba)\ .$$

The left factor l in Conway's theorem is $E_l(\varepsilon)$ (which equals $E_c(\varepsilon)\cup E_c(a)$) and the left factor r is $E_l(\varepsilon)\cup E_l(ba)$; E is the entry indexed by the pair (l,r) (which, in this case, equals r).

We can use a standard elimination algorithm to compute entries in the factor matrix from the factor graph as regular expressions. Eliminating the nodes of fig. 12 from bottom to top and left to right, the regular expressions obtained for the left factors and their corresponding right factors are shown in the table below (in the reverse order of their calculation).

left factor	right factor
a^*	$a^* \cdot ((b+a{\cdot}a)^* \cdot a + \varepsilon + a \cdot (b+a{\cdot}a)^* \cdot a)$
$a^* \cdot (b+a{\cdot}a)^*$	$(b+a{\cdot}a)^* \cdot a$
$a^* \cdot ((b+a{\cdot}a)^* \cdot a + \varepsilon + a \cdot (b+a{\cdot}a)^* \cdot a)$	$\varepsilon + a \cdot (b+a{\cdot}a)^* \cdot a$

The language E recognised by fig. 11(a) is the last entry in the list of left factors and the first entry in the list of right factors. (The expressions have been simplified using the fact that ε is the unit of product, that $\emptyset^* = \varepsilon$ and that $\emptyset$ is the zero of product. We assume that these properties are always exploited. No other simplifications have been made.)

As is often the case with the use of elimination algorithms, the regular expressions in the above table are quite complex. By exploiting our insights into factors of factors together with the syntactic monoid, simpler expressions can be obtained — in a non-ad hoc way. Let us explain how this is done.

The fact that the factor graph is not strongly connected immediately suggests how to decompose it into simpler factor graphs of factors. Specifically, the factor graph (fig. 12(a)) is a combination of two factor graphs: the factor graphs of a^* and $(b+a{\cdot}a)^*$. See fig. 13. (Note that, as is our usual practice, inadmissible factors $\emptyset$ and T^* have been omitted from both graphs. If the alphabet T is $\{a\}$, the factor graph of a^* has no inadmissible nodes; however, in this case, the alphabet is $\{a,b\}$.)

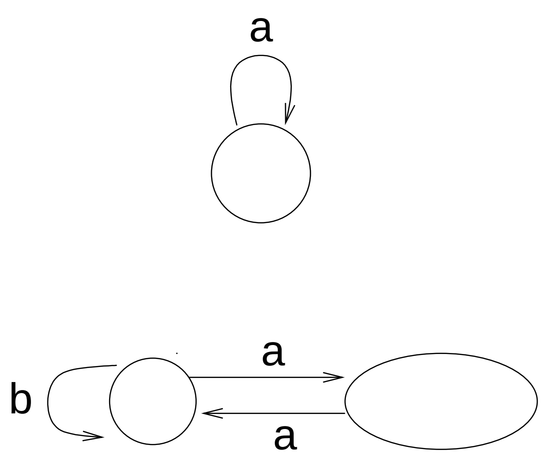


Figure 13: Factor Graphs of Factors

Our algorithm calculates the closure of each of these factor graphs and then seeks to fill in remaining entries in the factor matrix using the syntactic monoid. This is the motivation for calculating the factor matrix in the algebra $\mathcal{S}$ shown in fig. 12(b).

Now, as indicated in fig. 12(b), the entry of interest is $\{\varepsilon,a,ba,aba\}$. But

$$\{\varepsilon,a,ba,aba\} \;=\; \{\varepsilon,aba\}\cup\{a,ba\}$$

and both $\{\varepsilon,aba\}$ and $\{a,ba\}$ are entries in the factor matrix of $(b+a{\cdot}a)^*$. Regular expressions denoting these entries are calculated to be $\varepsilon + a\cdot(b+a{\cdot}a)^*\cdot a$ and $(b+a{\cdot}a)^*\cdot a$, respectively. Thus a regular expression denoting our language E is

(200) $\quad \varepsilon + a\cdot(b+a{\cdot}a)^*\cdot a + (b+a{\cdot}a)^*\cdot a \quad .$

This is simpler than the expression obtained by applying a standard elimination algorithm: the expression

(201) $\quad a^*\cdot((b+a{\cdot}a)^*\cdot a + \varepsilon + a\cdot(b+a{\cdot}a)^*\cdot a)$

in the table above. We leave it to the reader to check that both expressions denote the same language.

Remark This example is example 2 in [LS02]. One reason for including the example in detail here is that the factorisations given in [LS02, example 2] obscure a basic property of the factor matrix, namely that every event is both a left and right factor of itself. This means that the language recognised by the factor graph should always appear both in the list of left factors and in the list of right factors. In [LS02, example 2], the language recognised does appear twice but this is far from clear: the list of left factors includes the expression

$$(\varepsilon + b^*\cdot a)\cdot(a\cdot b^*\cdot a)^*$$

and the list of right factors includes the expression

$$\varepsilon + a^*\cdot(a{\cdot}a+b)^*\cdot a \quad .$$

In fact, these expressions are equal —in the sense that they denote the same language— but this is certainly not obvious[7]. They are also equal to the expressions on the right sides of (200) and (201). Which expression is the "best" is not important. What is important, however, is that the two occurrences of the language recognised are denoted by identical expressions.

[7]In an earlier draft (January 2017), I claimed that there were "obvious errors" in [LS02, example 2], but without giving further explanation. The "obvious error" I was referring to is that the expressions are not identical. I should, however, have taken more care to check whether or not the expressions denote the same language (which they do); instead, I jumped to the conclusion that one or both was incorrect, for which I apologise.

Another reason for including it here is to observe that the factor graph has a unique start state and a unique final state; in [LS02, example 2], the graph displayed is $\mathbf{L}_{\mathsf{max}}.\mathsf{E}$ but all three nodes are identified as start states and two nodes are identified as final states. This more fundamental difference is discussed further in section 10.

End of Remark

□

Example 202 Lombardy and Sakarovitch discuss a closely related example [LS08, example 5.7]. Since the calculations are very similar to those in example 199, we summarise the calculations briefly here, leaving the details to the reader.

Consider the language E denoted by the regular expression

$$a^* \cdot (a \cdot a + b)^* \cdot a^* \quad .$$

The machine, syntactic monoid and factor graph are shown in fig. 14. (The language is its own reverse so the anti-machine and machine are identical. Take care, however, when choosing representatives of the r-classes. The syntactic monoid has an additional element to those shown in the figure, with representative bab. Since it is not an element of the c-classes of any admissible factor, it has been omitted.)

As in example 199, the languages denoted by a^* and $(a \cdot a + b)^*$ are factors; their factor graphs are shown in fig. 13. The factor matrix, as represented by sets of c-classes (that is, the matrix $|\overline{\zeta_c^\flat.\mathsf{E}|}$) is the following :

$$\begin{bmatrix} \{\varepsilon,a\} & \{\varepsilon,a,b,ab\} & \{\varepsilon,a,ba,aba\} & \{\varepsilon,a,b,ab,ba,aba\} \\ \emptyset & \{\varepsilon,b\} & \{a,ba\} & \{\varepsilon,a,b\} \\ \emptyset & \{a,ab\} & \{\varepsilon,aba\} & \{\varepsilon,a,aba\} \\ \emptyset & \emptyset & \emptyset & \{\varepsilon,a\} \end{bmatrix}$$

(The nodes of the factor graph have been taken in order from bottom to top and from left to right.) As indicated in the conventional way by the start and final nodes in the factor graph, the language E is represented by $\{\varepsilon,a,b,ab,ba,aba\}$. Now,

$$\{\varepsilon,a,b,ab,ba,aba\} \;=\; \{\varepsilon,b\} \cup \{a,ab\} \cup \{a,ba\} \cup \{\varepsilon,aba\} \quad .$$

The right side is the set union of the middle four terms in the factor matrix: the four admissible factors of $(a \cdot a + b)^*$. Exploiting this fact, it suffices to calculate regular expressions denoting the entries in the factor matrix of $(a \cdot a + b)^*$ and add them together to get a regular expression for the language E. This is a slight improvement on the expression calculated using a standard elimination algorithm.

(Intriguingly, we have

$$\{\varepsilon,a,b,ab,ba,aba\} \;=\; \{\varepsilon,a\} \circ \{\varepsilon,b\} \circ \{\varepsilon,a\}$$

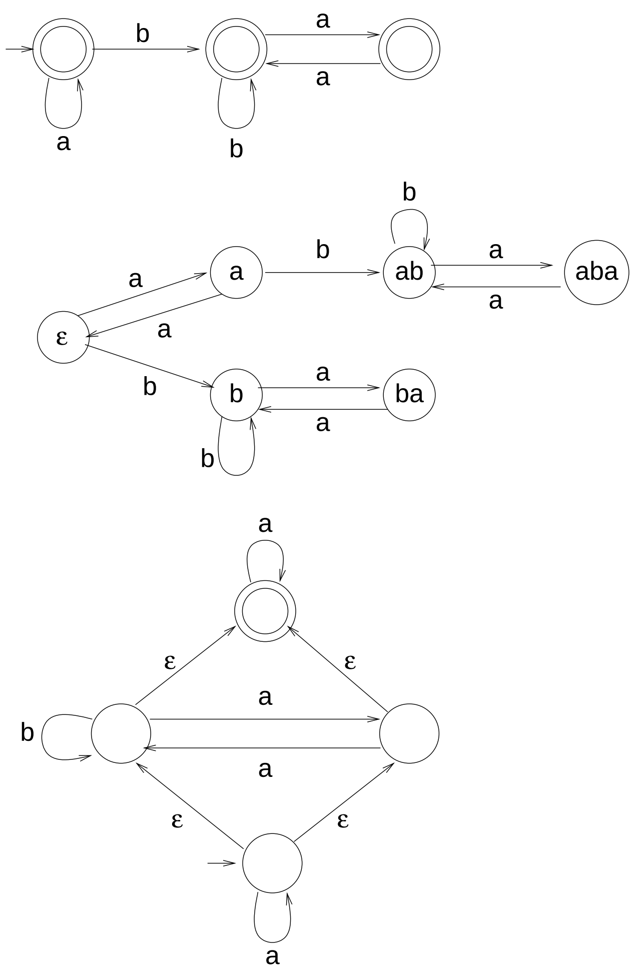


Figure 14: Machine, Syntactic Monoid and Factor Graph

and

$$\zeta_c^\sharp.\{\varepsilon,a,b,ab,ba,aba\} \;=\; \zeta_c^\sharp.\{\varepsilon,a\}\cdot\zeta_c^\sharp.\{\varepsilon,b\}\cdot\zeta_c^\sharp.\{\varepsilon,a\} \;\;.$$

So, in this case, the function $\zeta_c^\sharp$ does distribute through product. The right side gives the regular expression $a^*\cdot(a{\cdot}a+b)^*\cdot a^*$ for the language E. However, as shown in example 178, the function $\zeta_c^\sharp$ does not distribute through product in general. Theorems that predict when the distributivity law does hold might prove very useful in determining more compact regular expressions.)

□

8 The Rank of a Subfactor Graph

We showed in section 4 that the factor matrix of a factor is represented by a submatrix of the factor matrix. We now turn our attention to the factor graph of a factor of E.

It is not the case that the factor graph of a factor of E is a subgraph of the factor graph of E We prove, however, that the so-called *cycle rank* of the factor graph of a factor of E is at most the cycle rank of the factor graph of E. This then enables us in section 9 to derive a closure algorithm that calculates the factor matrix of E from its factor graph in such a way that the star-height of all the resulting regular expressions does not exceed the cycle rank of the factor graph and, in some cases, may be strictly smaller. The basis of our proof is the construction of a so-called *pathwise homomorphism* of the factor graph of E.

8.1 Cycle Rank and Pathwise Homomorphism

The notions of cycle rank and pathwise homomorphism were introduced by Eggan [Egg63] and McNaughton [McN67], respectively. In this subsection we recall the definitions and McNaughton's theorem.

A *subgraph* of a graph $\mathbf{G}$ is a graph determined by a subset of the nodes of $\mathbf{G}$; the edges of the subgraph are those edges of $\mathbf{G}$ that are both to and from a node in the given subset. A subgraph is *proper* of $\mathbf{G}$ if it is not equal to $\mathbf{G}$. A graph $\mathbf{G}$ is *strongly connected* if there is a path of edge-length at least 1 from x to y for every pair of nodes x and y in $\mathbf{G}$. A *section* of a graph is a strongly connected subgraph of $\mathbf{G}$ that is not a proper subgraph of any strongly connected subgraph of $\mathbf{G}$.

Definition 203 (Rank of a Graph) Suppose $\mathbf{G}$ is a graph. The *(cycle) rank* of $\mathbf{G}$ is a natural number defined as follows.

(i) If $\mathbf{G}$ is not strongly connected, then

  (a) If $\mathbf{G}$ has no strongly connected subgraph then the rank of $\mathbf{G}$ is 0.

  (b) Otherwise, the rank of $\mathbf{G}$ is the maximum rank of all the sections of $\mathbf{G}$.

(ii) If $\mathbf{G}$ is strongly connected, then the rank of $\mathbf{G}$ is $n+1$ where n satisfies

  (a) $\mathbf{G}$ does not have rank m for any m that is at most n.

  (b) $\mathbf{G}$ has a node x whose deletion from $\mathbf{G}$ results in a graph of rank n.

□

Theorem 204 (Eggan's Theorem) Consider the use of the escalator method[8] to calculate $\mathbf{G}^*$ for a given transition graph $\mathbf{G}$. Then

(a) for a suitable ordering of the nodes of $\mathbf{G}$, the resulting regular expressions for all entries $i\,\mathbf{G}^*\,j$ have star-height at most the rank of $\mathbf{G}$.

(b) If the graph $\mathbf{G}$ is all-admissible for nodes i and j (that is, for every node k in $\mathbf{G}$, there is a path from i to k and a path from k to j) then, for a suitable ordering of the nodes, the resulting entry $i\,\mathbf{G}^*\,j$ has star-height equal to the rank of $\mathbf{G}$.

Proof (Outline) The definition of rank determines an order of elimination of the nodes of $\mathbf{G}$. This is the ordering referred to in the theorem. The proof —which is straightforward— proceeds by induction on the rank.

□

Note that the definition of rank makes no reference to the edge labels; it is purely about the connectivity of a graph. That is, it is a function on the relation on nodes defined by the edges of the graph. Specifically, a transition graph $\mathbf{G}$ defines a binary relation $Rel.\mathbf{G}$ on nodes by $i\ Rel.\mathbf{G}\ j \equiv i\ \mathbf{G}\ j \neq \emptyset$ for all nodes i and j and it is this relation on which the rank is defined. Eggan's theorem does, however, assume that the graph is a transition graph: that is, edges are labelled by subsets of $T \cup \{\varepsilon\}$, where T is an alphabet of symbols. In fact, Eggan assumes that edge labels are subsets of T. The theorem is unaffected by allowing the empty word to be included[9]; accordingly, we relax the assumption (as does McNaughton [McN67]).

We emphasise the use of the escalator method because it is an instance of what we call an *elimination method*. Elimination methods for calculating $\mathbf{G}^*$ all have direct counterparts of methods used in linear algebra (for example, so-called Gauss-Seidel elimination) to invert a (real) matrix [BC75]. Their validity depends on algebraic properties common to both real numbers and languages. That means they do not exploit properties of languages like the idempotency of set union that are not enjoyed by real numbers. In appendix B, we formulate the essential characteristics of an elimination method and show that, using any elimination method to compute $\mathbf{G}^*$, the rank of the graph reflects the best that one can do in respect of the star-height of the resulting expressions. However, as we show, there is an algorithm to compute the closure of the factor graph that yields regular expressions that have star-height at most the rank of the factor graph and may have star-height less than its rank.

[8] Eggan's paper [Egg63] includes the description of an algorithm to compute $\mathbf{G}^*$; this algorithm has become known as the "escalator method".

[9] Strictly, this statement is incorrect if the algorithm exploits the property that $\varepsilon^* = \varepsilon$. The opportunity to exploit this property is rare and certainly does not occur when the transition graph is definite, which is the case for factor graphs.

In this section we prove that the rank of the factor graph of a factor of E is at most the rank of the factor graph of E. We appeal to a basic theorem due to McNaughton [McN69], namely that the rank of graph G' is at most the rank of graph G if there is a so-called "pathwise homomorphism" from G to G'.

For our purposes a slightly simpler definition of pathwise homomorphism suffices:

Definition 205 (Pathwise Homomorphism: McNaughton) Suppose G and G' are graphs. A *pathwise homomorphism* of G *onto* G' is a function f from the nodes of G to the nodes of G' such that the following two conditions hold:

(a) If there is an edge in G from u to v then there is an edge in G' from $f.u$ to $f.v$.

(b) If there is a path from node x to node y in G', there is a path from u to v in G for some nodes u and v such that $f.u = x$ and $f.v = y$.

□

Definition 205 is simpler than McNaughton's in that McNaughton requires the domain of f to be the nodes *and the edges* of G'. When applied to an edge of G', McNaughton allows f to be either an edge of G or a node of G, and weakens requirement (b) accordingly; definition 205 disallows the second possibility. .

Theorem 206 (McNaughton's Pathwise Homomorphism Theorem) Suppose G and G' are graphs. Then the rank of G is at least the rank of G' if there is a pathwise homomorphism of G onto G'.

Proof It is easy to check that a pathwise homomorphism according to our definition is a pathwise homomorphism according to McNaughton's definition. So the theorem follows from [McN67, theorem 3].
□

We also need the following simple theorem [McN69, theorem 2.4].

Theorem 207 The rank of graph G is at most the rank of transition graph G' if G and G' have the same set of nodes and the set of edges of G is a subset of the set of edges of G'.

Proof Straightforward from the definition of rank.
□

From a calculational viewpoint, it is desirable to exploit the algebra of regular languages to the full: specifically, the fact that matrices of regular events form a regular algebra. To this end, we reformulate a calculational form of definition 205. Specifically, the definition of pathwise homomorphism that we use is as follows.

Definition 208 (Pathwise Homomorphism: Calculational) Suppose $\mathbf{G}$ and $\mathbf{G}'$ are graphs with node sets N and N', respectively. A *pathwise homomorphism* of $\mathbf{G}$ *onto* $\mathbf{G}'$ is a function f from N to N' such that the following conditions hold:

(a) $\mathrm{I}_{\mathsf{N}'} = \mathsf{f} \circ \mathsf{f}^{\cup} \wedge \mathrm{I}_{\mathsf{N}} \subseteq \mathsf{f}^{\cup} \circ \mathsf{f}$,

(b) $\mathbf{G} \;\dot{\subseteq}\; (\mathit{Sel}.\mathsf{f})^{\cup} \otimes \mathbf{G}' \otimes \mathit{Sel}.\mathsf{f}$, and

(c) $(\mathbf{G}')^{*} \;\dot{\subseteq}\; \mathit{Sel}.\mathsf{f} \otimes \mathbf{G}^{*} \otimes (\mathit{Sel}.\mathsf{f})^{\cup}$.

If a pathwise homomorphism exists from $\mathbf{G}$ onto $\mathbf{G}'$, we say that $\mathbf{G}'$ is *pathwise homomorphic* to $\mathbf{G}$.
□

Our definition of pathwise homomorphism appears to be stricter than McNaughton's since the inequality $\mathrm{I}_{\mathsf{N}'} \subseteq \mathsf{f} \circ \mathsf{f}^{\cup}$ implied by property 208(a) states formally that f is *surjective*. However, as remarked by McNaughton, because there is a path from every node to itself in a graph —the empty path— condition 208(c) implies that a pathwise homomorphism f is necessarily surjective. The remaining inequalities in 208(a) are calculational formulations of the property that f is functional and total. Condition 208(a) is the same as McNaughton's condition 205(b) and condition 208(c) is the same as 205(b).

8.2 The Pathwise Homomorphism

Suppose F is a factor of E. We show that the factor graph of F has rank at most the rank of the factor graph of E. Specifically, we construct a graph $\mathbf{G}'$ that is pathwise homomorphic to the factor graph of E and then show that the factor graph of F is a subgraph of $\mathbf{G}'$.

Throughout this section, we use $\mathcal{FG}.\mathsf{F}$ to denote the factor graph of F. Similarly for $\mathcal{FG}.\mathsf{E}$. We assume that $\mathsf{F} = \mathsf{s}\backslash\mathsf{t}$ where

(209) $\mathsf{s} = \mathsf{t}/(\mathsf{s}\backslash\mathsf{t}) \;\wedge\; \mathsf{t}{\rhd} = (\mathsf{s}{\rhd}/\mathsf{t}{\rhd})\backslash\mathsf{s}{\rhd}$.

(See lemma 86.) Recall that the nodes of the factor graph of a language are the left factors of that language.

We define the function γ from left factors of E to left factors of F by $\gamma.\mathsf{i} = \mathsf{F}/(\mathsf{i}\backslash\mathsf{t})$. (See (100).) Recall theorem 106 which states that γ is a total, surjective function mapping the left factors of E onto the left factors of F. The graph $\mathbf{G}'$ is defined by:

(210) $\mathbf{G}' = \mathit{Sel}.\gamma \otimes \mathcal{FG}.\mathsf{E} \otimes (\mathit{Sel}.\gamma)^{\cup}$.

Informally, the nodes of $\mathbf{G}'$ are the left factors of F; there is an edge labelled x from i' to j' in $\mathbf{G}'$ if, for some left factors i and j of E, there is an edge labelled x from i to j in the factor graph of E and

$$i' = \gamma.i \wedge j' = \gamma.j \quad .$$

Formally, for all left factors i' and j' of F,

(211) $\quad i'\, \mathbf{G}'\, j' \;=\; \langle \cup i,j \;:\; i' = \gamma.i \wedge j' = \gamma.j \;:\; i\, \mathcal{FG}.\mathsf{E}\, j \rangle$

where the dummies i and j range over left factors of E.

The graph $\mathbf{G}'$ acts as an intermediary between the factor graph of E and the factor graph of F. We begin by relating $\mathbf{G}'$ to the maximal constant+linear approximation, $\mathbf{C}_{max}.\mathsf{F} \mathbin{\dot\cup} \mathbf{L}_{max}.\mathsf{F}$, to the factor matrix of the event F. Specifically:

Lemma 212

$$\mathbf{G}' \;\dot\subseteq\; \mathbf{C}_{max}.\mathsf{F} \mathbin{\dot\cup} \mathbf{L}_{max}.\mathsf{F} \quad .$$

Proof

$$\begin{array}{ll}
 & \mathbf{G}' \;\dot\subseteq\; \mathbf{C}_{max}.\mathsf{F} \mathbin{\dot\cup} \mathbf{L}_{max}.\mathsf{F} \\
= & \qquad \{ \qquad \text{definition: (210)} \quad \} \\
 & Sel.\gamma \otimes \mathcal{FG}.\mathsf{E} \otimes (Sel.\gamma)^{\cup} \;\dot\subseteq\; \mathbf{C}_{max}.\mathsf{F} \mathbin{\dot\cup} \mathbf{L}_{max}.\mathsf{F} \\
\Leftarrow & \qquad \{ \qquad \gamma \text{ is a surjective function onto the set of left factors of } \mathsf{F}; \\
 & \qquad\qquad \text{so, by (32) and (30), } Sel.\gamma \otimes (Sel.\gamma)^{\cup} \;=\; \mathbf{I}_{\mathcal{L}.\mathsf{F}} \\
 & \qquad\qquad \text{monotonicity of matrix product} \quad \} \\
 & \mathcal{FG}.\mathsf{E} \;\dot\subseteq\; (Sel.\gamma)^{\cup} \otimes (\mathbf{C}_{max}.\mathsf{F} \mathbin{\dot\cup} \mathbf{L}_{max}.\mathsf{F}) \otimes Sel.\gamma \quad .
\end{array}$$

Continuing with the right side of the above inequality, we have:

$$\begin{array}{ll}
 & (Sel.\gamma)^{\cup} \otimes (\mathbf{C}_{max}.\mathsf{F} \mathbin{\dot\cup} \mathbf{L}_{max}.\mathsf{F}) \otimes Sel.\gamma \\
= & \qquad \{ \qquad \text{definition of } \mathbf{C}_{max}.\mathsf{F} \mathbin{\dot\cup} \mathbf{L}_{max}.\mathsf{F} \text{ (see theorem 158)} \\
 & \qquad\qquad \text{lemma 114} \quad \} \\
 & (Sel.\gamma)^{\cup} \otimes (Sel.\gamma \otimes |\overline{\mathsf{E}}| \otimes (Sel.\gamma)^{\cup} \mathbin{\dot\cap} \mathsf{Mat}.(\mathsf{T} \cup \{\varepsilon\})) \otimes Sel.\gamma \\
= & \qquad \{ \qquad Sel.\gamma \text{ is a constant matrix, length considerations} \quad \} \\
 & ((Sel.\gamma)^{\cup} \otimes Sel.\gamma \otimes |\overline{\mathsf{E}}| \otimes (Sel.\gamma)^{\cup} \otimes Sel.\gamma) \mathbin{\dot\cap} \mathsf{Mat}.(\mathsf{T} \cup \{\varepsilon\}) \\
\dot\supseteq & \qquad \{ \qquad \gamma \text{ is a total function with domain the set of left factors of } \mathsf{E};
\end{array}$$

so, by (33), $(Sel.\gamma)^{\cup} \otimes Sel.\gamma \ \dot{\supseteq} \ \mathbf{I}_{\mathcal{L}.\mathsf{E}}$

monotonicity of matrix product and intersection }

$|\overline{\mathsf{E}}| \,\dot{\cap}\, \mathsf{Mat}.(\mathsf{T}\cup\{\varepsilon\})$

$\dot{\supseteq}$ { by definition (see theorem 158), $\mathcal{FG}.\mathsf{E} \ \dot{\subseteq}\ |\overline{\mathsf{E}}| \,\dot{\cap}\, \mathsf{Mat}.(\mathsf{T}\cup\{\varepsilon\})$ }

$\mathcal{FG}.\mathsf{E}$.

Combining the two calculations, the proof is complete.

□

Lemma 212 exploits the surjectivity of γ to bound $\mathbf{G}'$ from above. Now we exploit its totality to bound it from below. We have:

Lemma 213

$$(\mathcal{FG}.\mathsf{F})^* \ = \ Sel.\gamma \otimes (\mathcal{FG}.\mathsf{E})^* \otimes (Sel.\gamma)^{\cup} \ \dot{\subseteq} \ (\mathbf{G}')^* \quad .$$

Proof

$(\mathbf{G}')^*$

$=$ { definition: (210) }

$(Sel.\gamma \otimes \mathcal{FG}.\mathsf{E} \otimes (Sel.\gamma)^{\cup})^*$

$\dot{\supseteq}$ { γ is a total function of type $\mathcal{L}.\mathsf{F} \leftarrow \mathcal{L}.\mathsf{E}$, lemma 36 }

$Sel.\gamma \otimes (\mathcal{FG}.\mathsf{E})^* \otimes (Sel.\gamma)^{\cup}$

$=$ { $\mathcal{FG}.\mathsf{E}$ is the factor graph of E,

so $(\mathcal{FG}.\mathsf{E})^*$ is the factor matrix of E }

$Sel.\gamma \otimes |\overline{\mathsf{E}}| \otimes (Sel.\gamma)^{\cup}$

$=$ { theorem 114 }

$|\overline{\mathsf{F}}|$

$=$ { $\mathcal{FG}.\mathsf{F}$ is a starth root of $|\overline{\mathsf{F}}|$ }

$(\mathcal{FG}.\mathsf{F})^*$.

□

Corollary 214

$$(\mathcal{FG}.\mathsf{F})^* = (\mathbf{G}')^* = |\overline{\mathsf{F}}| \quad .$$

It follows that $\mathcal{FG}.\mathsf{F} \,\dot{\subseteq}\, \mathbf{G}'$ and the rank of $\mathbf{G}'$ is at least the rank of $\mathcal{FG}.\mathsf{F}$.

Proof From lemmas 213 and 212 (and monotonicity of the star operator), we have

$$(\mathcal{FG}.\mathsf{F})^* \;\dot{\subseteq}\; (\mathbf{G}')^* \;\dot{\subseteq}\; (\mathbf{C}_{max}.\mathsf{F} \;\dot{\cup}\; \mathbf{L}_{max}.\mathsf{F})^* \quad .$$

But $\mathcal{FG}.\mathsf{F}$ and $\mathbf{C}_{max}.\mathsf{F} \;\dot{\cup}\; \mathbf{L}_{max}.\mathsf{F}$ are both starth roots of $|\overline{\mathsf{F}}|$. (See theorem 158 .) So all of $(\mathcal{FG}.\mathsf{F})^*$, $(\mathbf{G}')^*$ and $(\mathbf{C}_{max}.\mathsf{F} \;\dot{\cup}\; \mathbf{L}_{max}.\mathsf{F})^*$ are equal to $|\overline{\mathsf{F}}|$. It follows that $\mathcal{FG}.\mathsf{F} \dot{\subseteq} \mathbf{G}'$ because $\mathcal{FG}.\mathsf{F}$ is the *least* starth root of $|\overline{\mathsf{F}}|$ (theorem 158). That the rank of $\mathcal{FG}.\mathsf{F}$ is at most the rank of $\mathbf{G}'$ is a simple application of theorem 207.
□

Lemma 215 $\mathbf{G}'$ is pathwise homomorphic to $\mathcal{FG}.\mathsf{E}$. Hence, the rank of $\mathcal{FG}.\mathsf{E}$ is at least the rank of $\mathbf{G}'$.

Proof Referring to the definition of pathwise homomorphism (definition 208), we instantiate $\mathbf{G}$ to $\mathcal{FG}.\mathsf{E}$ and the function f to γ; $\mathbf{G}'$ is as in (210). The function γ is, indeed, a total, surjective function from the left factors of E (the node set of $\mathcal{FG}.\mathsf{E}$) to the left factors of F (the node set of $\mathbf{G}'$), as proved in theorem 106. This is part (a) of the definition. Part (b) is established as follows:

$$\begin{array}{ll}
 & (Sel.\gamma)^{\cup} \otimes \mathbf{G}' \otimes Sel.\gamma \\
= & \qquad \{ \qquad (210) \quad \} \\
 & (Sel.\gamma)^{\cup} \otimes Sel.\gamma \otimes \mathcal{FG}.\mathsf{E} \otimes (Sel.\gamma)^{\cup} \otimes Sel.\gamma \\
\dot{\supseteq} & \qquad \{ \qquad \gamma \text{ is total, i.e. } (Sel.\gamma)^{\cup} \otimes Sel.\gamma \;\dot{\supseteq}\; \mathbf{I}_{\mathcal{L}.\mathsf{E}} \quad \} \\
 & \mathcal{FG}.\mathsf{E} \quad .
\end{array}$$

Part (c) is established as follows.

$$\begin{array}{ll}
 & (\mathbf{G}')^* \\
= & \qquad \{ \qquad \text{corollary 214} \quad \} \\
 & (\mathcal{FG}.\mathsf{F})^* \\
= & \qquad \{ \qquad \text{lemma 213} \quad \} \\
 & Sel.\gamma \otimes (\mathcal{FG}.\mathsf{E})^* \otimes (Sel.\gamma)^{\cup} \quad .
\end{array}$$

(That is, we have established an equality rather than just an inclusion.) This completes the verification of all three parts of the definition of pathwise homomorphism.

Applying McNaughton's theorem (theorem 206) we conclude that the rank of $\mathcal{FG}.\mathsf{E}$ is at least the rank of $\mathbf{G}'$.
□

Theorem 216 The rank of the factor graph of an event is at least the rank of the factor graph of any factor of the event.

Proof Suppose E is an event and F is a factor of E. Let $\mathbf{G}'$ be as defined in (210), let $\mathcal{FG}.F$ be the factor graph of F and let $\mathcal{FG}.E$ be the factor graph of E. By lemma 215, the rank of $\mathcal{FG}.E$ is at least the rank of $\mathbf{G}'$ and, by corollary 214, the rank of $\mathbf{G}'$ is at least the rank of $\mathcal{FG}.F$. The theorem follows by transitivity of the at-least relation.

□

9 Closure Algorithm

In this section we present a closure algorithm for determining the factor matrix of a regular event that constructs a regular expression for the event with star-height at most the rank of the factor graph of the event. We provide examples that demonstrate that the star-height may be strictly less than the rank of the factor graph. We also provide an example that shows that the algorithm does not always yield a regular expression of minimal star-height.

Aside The starting point for our closure algorithm is the factor graph of the given language. This is a better starting point than Conway's best constant+linear approximation to the factor matrix because, in general, the cycle rank of the factor graph may be strictly smaller than the cycle rank of the best constant+linear approximation. This is demonstrated by fig. 15 which depicts both the best constant+linear approximation to the factor matrix of a^*a (assuming alphabet $\{a\}$) and the factor graph of a^*a.

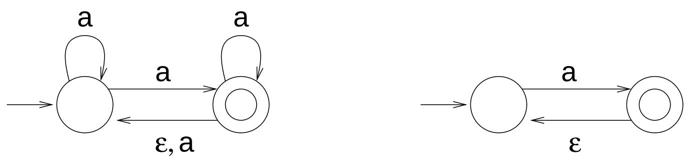


Figure 15: Best constant+linear approximation (left) and factor graph (right) of a^*a.

The best constant+linear approximation has cycle rank 2 whereas the factor graph has cycle rank 1. **End of Aside**

Suppose $\mathbf{G}$ is a graph with at least two nodes. Suppose we split the nodes of $\mathbf{G}$ into two distinct subsets M and N, say. This splits $\mathbf{G}$ into four subgraphs of dimension $M{\times}M$, $M{\times}N$, $N{\times}M$ and $N{\times}N$. Let these be denote by a, b, c and d, respectively. Suppose we split $\mathbf{G}^*$ in the same way into four subgraphs of dimension $M{\times}M$, $M{\times}N$,

$N{\times}M$ and $N{\times}N$ and denote them by A, B, C and D. That is, suppose

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Then

$$\begin{aligned} B &= A\cdot b\cdot d^* \\ C &= d^*\cdot c\cdot A \\ D &= d^* + d^*\cdot c\cdot A\cdot b\cdot d^* \end{aligned}$$

An elimination method would use the formula

$$A = (a + b\cdot d^*\cdot c)^*$$

to complete the set of equations. It is the use of this formula that gives rise to nested star terms in the resulting regular expressions[10]. If we can use an alternative means of calculating A then we may be able to do better.

This is indeed the case when $\mathbf{G}$ is the factor graph of an event E and A is the factor matrix of a subfactor F of E. In that case, let $\mathbf{G}'$ be the factor graph of F. Then $A = (\mathbf{G}')^*$. And, of course, this process can be applied recursively both to the calculation of $(\mathbf{G}')^*$ and to the calculation of d^*.

9.1 An Example

Let us show how this is done. Fig. 16 is the factor graph of T^*abaab where $T = \{a,b\}$, omitting inadmissible nodes. It is the recogniser that underlies the Knuth-Morris Pratt algorithm [KMP77] when used to search for occurrences of the pattern $abaab$ in a text[11]. The "backbone" of the recogniser is formed by the edges labelled in order from left to right by the successive symbols of the pattern. The edges labelled by the empty word act as "failure" transitions: when a symbol in the text fails to match a symbol in the pattern, the empty-word transitions are followed until a matching transition becomes possible [BL77, Bac16]. Even though some edges are labelled by the empty word, the recogniser is still "deterministic" in the sense that at no stage is there a choice of transition and nor is there ambiguity in when a word has been recognised.

[10]We assume that when d is a $1{\times}1$ matrix with entry $\emptyset$, the fact that $\emptyset^*$ is the unit of product is exploited so that $b\cdot d^*\cdot c$ is simplified to $b{\cdot}c$.

[11]For more details of Aho and Corasick's algorithm, see [BL76, BL77]. Essentially, given a finite set of words W over alphabet T, their algorithm computes the factor graph of $T^*{\cdot}W$. But there is a slight complication: there is a left-to-right bias in the Aho-Corasick algorithm. So, to make the correspondence precise, it is necessary to append a distinguishing terminal symbol to each word in W.

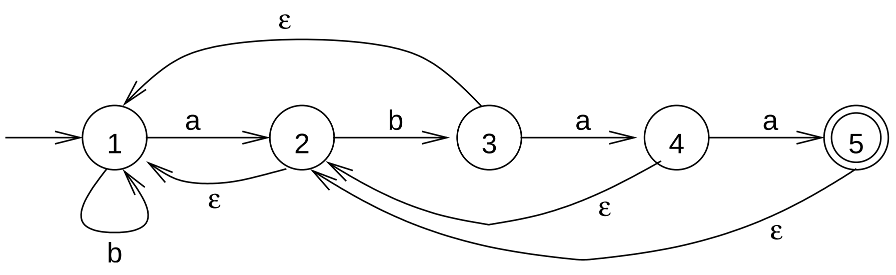


Figure 16: Factor Graph of T^*abaa (omitting inadmissible node).

The cycle rank of fig. 16 is 2. That is, if we use an elimination technique to compute its closure, the regular expression we obtain for T^*abaa would have star-height at least 2. It would be a much more complex expression, making it difficult to recognise that it denotes the same language! The nodes in fig. 16 have been numbered in the order that we intend to eliminate them. We now show how we can recover the expression T^*abaab from the graph by exploiting theorem 98.

Obviously, T^* is a factor of T^*abaa. Also obvious in fig. 16 is that T^* is the set of words that spell (sequences of) transitions from node 1 to itself. That is, the factor matrix of T^* is the submatrix of the factor matrix of T^*abaa identified by the single node 1. In other words, the factor matrix of T^* is the $(1,1)$th entry in the factor matrix of T^*abaa. Its factor graph is shown in fig. 17.

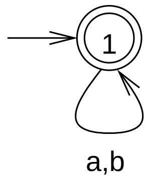


Figure 17: Factor Graph of T^*.

The remaining left factors of T^*abaa are T^*a, T^*ab, T^*aba, and T^*abaa; their factor graphs are the subgraphs of fig. 16 defined by the subsets {1,2}, {1,2,3} {1,2,3,4}, {1,2,3,4,5}, respectively, of the nodes of the graph.

Below we show the submatrices of the factor matrix of T^*abaa obtained by computing the factor matrices of T^*, T^*a, T^*ab, T^*aba, and T^*abaa in turn, as detailed above. From the second stage onwards, the matrix A is the factor matrix that has just been computed. One simplification has been made in the entries: the fact that the empty word (or, strictly, the set containing the empty word) is the unit of multiplication has been exploited. For example, the entry T^*a in the second matrix is a simplification of εT^*a.

$$\begin{bmatrix} T^* \end{bmatrix}$$

$$\begin{bmatrix} T^* & T^*a \\ T^* & \varepsilon+T^*a \end{bmatrix}$$

$$\begin{bmatrix} T^* & T^*a & T^*ab \\ T^* & \varepsilon+T^*a & (\varepsilon+T^*a)b \\ T^* & T^*a & \varepsilon+T^*ab \end{bmatrix}$$

$$\begin{bmatrix} T^* & T^*a & T^*ab & T^*aba \\ T^* & \varepsilon+T^*a & (\varepsilon+T^*a)b & (\varepsilon+T^*a)ba \\ T^* & T^*a & \varepsilon+T^*ab & (\varepsilon+T^*ab)a \\ T^* & T^*a & (\varepsilon+T^*a)b & \varepsilon+(\varepsilon+T^*a)ba \end{bmatrix}$$

$$\begin{bmatrix} T^* & T^*a & T^*ab & T^*aba & T^*abaa \\ T^* & \varepsilon+T^*a & (\varepsilon+T^*a)b & (\varepsilon+T^*a)ba & (\varepsilon+T^*a)baa \\ T^* & T^*a & \varepsilon+T^*ab & (\varepsilon+T^*ab)a & (\varepsilon+T^*ab)aa \\ T^* & T^*a & (\varepsilon+T^*a)b & \varepsilon+(\varepsilon+T^*a)ba & (\varepsilon+(\varepsilon+T^*a)ba)a \\ T^* & \varepsilon+T^*a & (\varepsilon+T^*a)b & \varepsilon+(\varepsilon+T^*a)ba & \varepsilon+((\varepsilon+T^*a)baa) \end{bmatrix}$$

Of course, it is only the first row that we are interested in. All the other entries are included so that the reader can see that none has star-height greater than 1. If the word $abaa$ is lengthened, the star-height of the resulting regular expressions will not increase and all the entries in the first row will be what one would wish them to be. Indeed, for any word w, all factors of T^*w have star-height 1 and our algorithm will compute appropriate regular expressions. However, the cycle-rank of the factor graph of T^*w increases with the length of the word w and can have unlimited value.

9.2 The Algorithm

In this subsection, we formulate an algorithm for calculating the factor matrix $|\overline{E}|$ for a given regular language E as the reflexive, transitive closure of the factor graph of E. The algorithm is guaranteed to yield a regular expression denoting E that has star-height at most the cycle rank of the factor graph of E and, as our examples illustrate, may have smaller star-height.

Using the fact that the left factors of E are in one-to-one correspondence with the nodes of its factor graph, we deliberately confuse the two. So, if i is a left factor of E, we sometimes refer to “node i” of the factor graph of E.

Assume that $\mathbf{G}$ is the factor graph of event E. (The inadmissible nodes can, of course, be ignored if so desired.) Recall that the set of "nodes" of $\mathbf{G}$ is the set $\mathcal{L}.E$ of left factors of E. Recall from section 7.1 that $\mathcal{S}.E$ denotes the powerset algebra with carrier set $2^{SM.E}$ and underlying monoid ($SM.E$, $\circ$,1), where $SM.E$ is the syntactic monoid of E.

Step 0 Construct the syntactic monoid $SM.E$ of E. Construct the factor graph $\mathbf{G}$ of E. Construct $\zeta_c^\flat.\mathbf{G}$ (the representation of each entry of $\mathbf{G}$ as a set of c-classes of E). Calculate $(\zeta_c^\flat.\mathbf{G})^*$ in the algebra $\mathcal{M}_{\mathcal{L}.E}(\mathcal{S})$. Use $(\zeta_c^\flat.\mathbf{G})^*$ to construct the "Hasse diagram" of the factors of E: the reflexive-transitive reduction of the set of entries in $(\zeta_c^\flat.\mathbf{G})^*$ ordered by the subset relation.

Step 1 For each pair of nodes i and j of $\mathbf{G}$ (i.e pair of left factors of E), use lemma 86 and definition 89 to construct the set of nodes $N.(i,j)$ that forms a submatrix of $|\overline{E}|$ that is the factor matrix of $i\backslash j$.

The event $i\backslash j$ is the (i,j)th entry in the factor matrix of E. Since factors typically occur repeatedly in the factor matrix, several different submatrices of the factor matrix may be identified in this way for the same event. This is intentional.

For the purposes of executing this step, $(\zeta_c^\flat.\mathbf{G})^*$ (calculated in step 0) should be used.

Step 2 The purpose of this step is to choose a subset of the set of pairs (i,j) that represents all the distinct submatrices of the the factor matrix of E that are factor matrices of the factors of E. The step also constructs a partial ordering of the chosen subset.

Consider the set of sets of nodes $\langle\cup i,j :: \{N.(i,j)\}\rangle$. This is partially ordered by the subset relation. The greatest element of the set is the set of all nodes of $\mathbf{G}$ (since $\mathbf{G}$ is the factor graph of E, which equals $l\backslash r$: see (72)).

Define the preorder $\preceq$ on pairs of nodes (i,j) by

$$(i,j) \preceq (k,l) \;\;\equiv\;\; N.(i,j) \subseteq N.(k,l) \;\; .$$

Note that

$$(i,j) \preceq (k,l) \wedge (k,l) \preceq (i,j) \;\;\equiv\;\; N.(i,j) = N.(k,l) \;\; .$$

That is, two pairs of nodes are equivalent under the preorder if they have the "same" factor matrix. In this case, we say that the factors $i\backslash j$ and $k\backslash l$ are *inseparable*.

For each set of nodes M in the set $\langle\cup i,j :: \{N.(i,j)\}\rangle$, choose one pair (i,j) such that $M = N.(i,j) \wedge i{\in}M \wedge j{\in}M$. (The second and third conjuncts are required because of the complication of repeated entries in the factor matrix. It is nevertheless always possible to choose at least one such pair. Otherwise, the choice is arbitrary.)

Let P denote the function that is defined in this way. That is, P is chosen so that $M = N.(P.M)$. Let $\mathsf{Im.P}$ denote the image set of P. That is,

$$\mathsf{Im.P} \;=\; \langle \cup\, i{,}j :: \{P.(N.(i,j))\}\rangle \quad .$$

By this construction, the set $\mathsf{Im.P}$ is partially ordered (as opposed to pre-ordered) by the $\preceq$ relation.

Step 3 Construct the factor graphs of $i\backslash j$ for each pair (i,j) in $\mathsf{Im.P}$. The factor graphs should be constructed in topological order according to the relation $\preceq$. That is, factor graphs are constructed for each of the sets $N.(i,j)$ in order of increasing size.

Step 4 Construct the factor matrix of $i\backslash j$ for each pair (i,j) in $\mathsf{Im.P}$ in the following way.

We use the variable $Done$ to record the set of pairs (i,j) for which a regular expression denoting $i\backslash j$ has already been computed. Initially $Done := \emptyset$. We also use the variable $\mathbf{A}$ to record the factor matrix that has been computed thus far. Initially $\mathbf{A}$ is the matrix of dimension $\emptyset\times\emptyset$.

Now the following step is executed in (topological) order for each pair (i,j) in $\mathsf{Im.P}$.

Suppose $p=(i,j)$. Let $M' := N.p$ and let $\mathbf{G}'$ be the factor graph of $i\backslash j$. Let $M := N.p \cap Done$. Note that M' is the set of nodes of $\mathbf{G}'$ and M is a proper subset of M'. Thus the graph $\mathbf{G}'$ is split into four subgraphs of dimension $M\times M$, $M\times(M'-M)$, $(M'-M)\times M$ and $(M'-M)\times(M'-M)$. Let these be denote by $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$, respectively. Suppose we split $(\mathbf{G}')^*$ in the same way into four subgraphs of dimension $M\times M$, $M\times(M'-M)$, $(M'-M)\times M$ and $(M'-M)\times(M'-M)$. The submatrix of $(\mathbf{G}')^*$ of dimension $M\times M$ has already been computed; it is the matrix $\mathbf{A}$. Suppose we denote the remaining three submatrices by $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$. That is, suppose

$$\begin{bmatrix} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{a} \end{bmatrix}^* \;=\; \begin{bmatrix} \mathbf{D} & \mathbf{C} \\ \mathbf{B} & \mathbf{A} \end{bmatrix}$$

Using $(\zeta_c^b.\mathbf{G})^*$ and the Hasse diagram of the factors of E, some of the entries in $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ can be expressed as the set union of entries in the matrix $\mathbf{A}$. If so, enter these expressions in the appropriate place. For the remaining entries in $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$, use the formulae

$$\begin{aligned}\mathbf{B} &= \mathbf{A}\otimes\mathbf{b}\otimes\mathbf{d}^*\\ \mathbf{C} &= \mathbf{d}^*\otimes\mathbf{c}\otimes\mathbf{A}\\ \mathbf{D} &= \mathbf{d}^*\,\dot{\cup}\,\mathbf{d}^*\otimes\mathbf{c}\otimes\mathbf{A}\otimes\mathbf{b}\otimes\mathbf{d}^*\end{aligned}$$

combined with a standard elimination method to calculate remaining elements of $\mathbf{d}^*$. Although not relevant to the star-height of the resulting regular expressions, it is convenient and practical to exploit the fact that ε is the unit of product at the same time. The final step in this iterative procedure is to execute the assignment $Done := M'$ and to assign to $\mathbf{A}$ the value of the factor matrix that has just been computed.

The final value of $\mathbf{A}$ is the required factor matrix. The loop invariant is that $\mathbf{A}$ is the factor matrix of $i\backslash j$.

The factor graph of an event E might not be strongly connected, in which case the acyclic structure of the sections of the graph should be exploited. We conjecture that each section of the factor graph of E is the factor graph of a factor of E but we have, as yet, no proof.

9.3 Detailed Example

Consider the event denoted by the regular expression $a\,(a{+}b)^*\,b\,(a{+}b)^*\,a$. We also use the numeral 3 to denote this event. (So 3 does not denote a number in just the same way that the symbol $+$ in a regular expression does not denote addition of numbers.)

Step 0 is to construct the factor graph and semigroup of the event 3 and then to construct the sets of nodes that form the factor graphs of factors of 3. The syntactic monoid of the event is depicted in fig. 18.

Each node of fig. 18 is labelled by an element of the c-class that the node represents. The product of two c-classes is determined by chasing paths. For example, the product of the c-class of ab and the c-class of ba is the c-class of aba because the path that begins at the node labelled ab and spells ba ends at the node labelled aba.

In anticipation of step 3, which requires the calculation of the factor graphs as well as the node sets, Fig. 19 shows the factor graphs of all the factors of the event 3. In the figures, the nodes are labelled by the numerals 1, 2, 3 and 4. As in section 9.2, we confuse the terms “node” and “left factor” so that these four numerals also denote the left factors of the event 3.

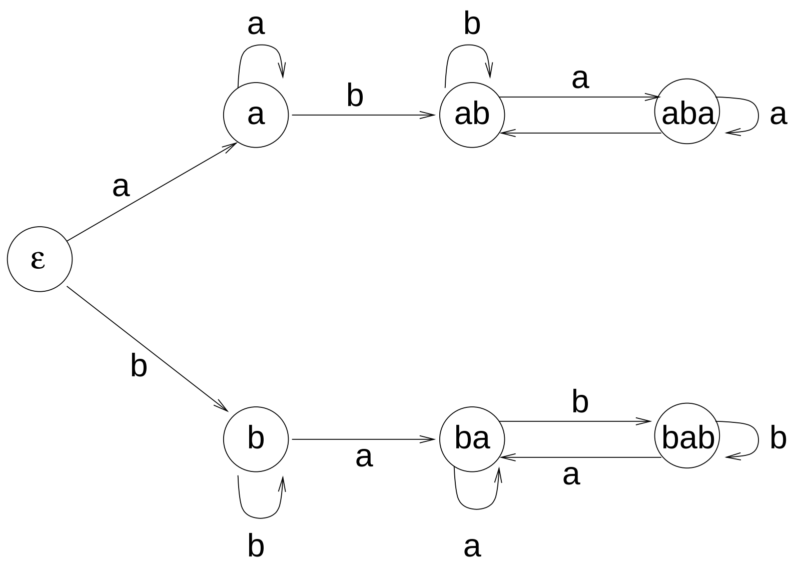


Figure 18: Syntactic Monoid of $a\ (a+b)^*\ b\ (a+b)^*\ a$.

The topmost graph is the factor graph of the event 3 itself. The conventional method of indicating that it is a recogniser has also been indicated: the event 3 is the set of words that spell a path from the node 1 (the "start" node indicated by an unlabelled arrow) to the node 3 (the "finish" node indicated by a double circle). The set of words that spell a path from node i to node j is $i\backslash j$, which is the (i,j)th element of the factor matrix of the event 3. This fits with Conway's theorem on the factor matrix: for the event 3, the left factor l is the event 1 and the left factor r is the event 3. (See (72) and (75).) Moreover, $1=1\backslash 1$, $2=1\backslash 2$, $3=1\backslash 3$ and $4=1\backslash 4$.

Step 1 is achieved by constructing the table in fig. 20: it gives the relation between each graph and the factors of which the graph is the factor graph. For example, the fourth graph from the top has nodes 2 and 4 and is the factor graph of the event $2\backslash 4$, as shown in the fourth row of the table. Note that all the factors in the right column of the last two rows are equal. As forewarned, they nevertheless define two distinct sets of nodes. See below for how the syntactic monoid is used to calculate the collection of factor graphs.

The factor matrix of the event 3, where each entry is represented by the corresponding event in the powerset algebra of the syntactic monoid, is displayed in fig. 21. (This is what we called $(\zeta_c^{\flat}.\mathbf{G})^*$ in the algorithm.) Each element of the syntactic monoid is denoted by a word of shortest length in the corresponding c-class ; the symbol S denotes the set of all elements of the syntactic monoid.

It is easy to calculate the set of c-classes that comprise each factor using representative elements of the c-, l- and r-classes; once constructed such a representation enables

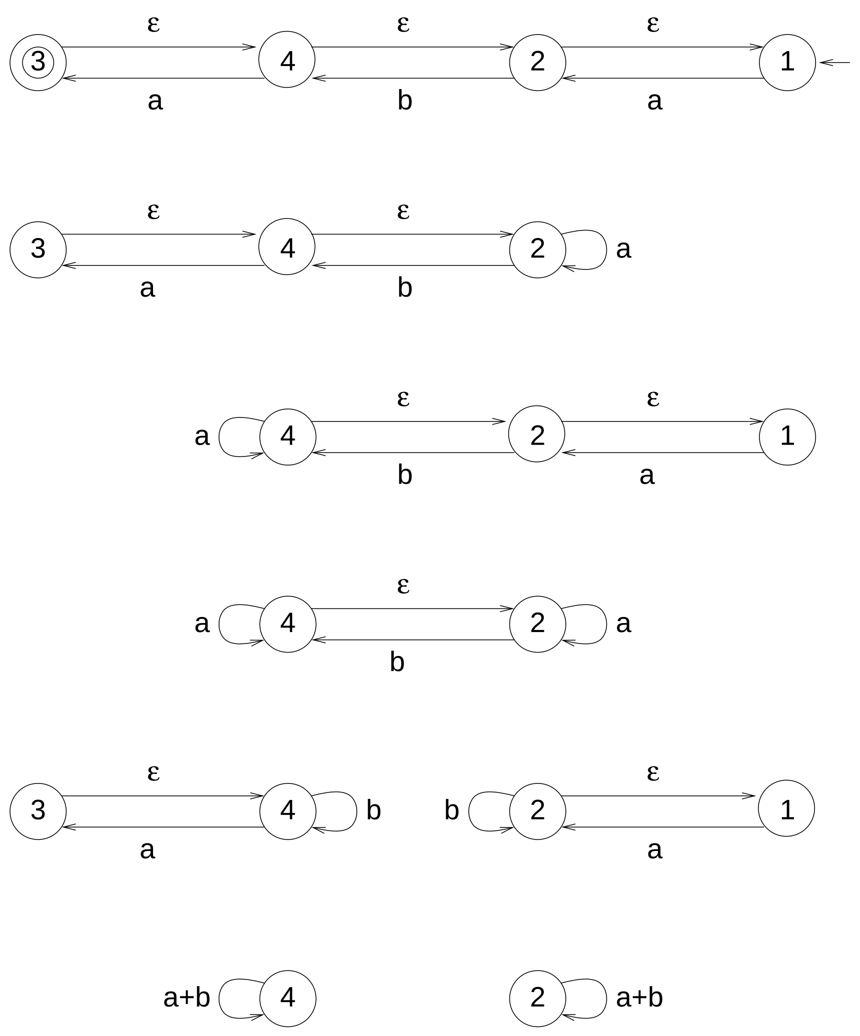


Figure 19: Factor graphs of the factors of $a\ (a{+}b)^*\ b\ (a{+}b)^*\ a$.

straight-forward calculation of factor graphs of factors. In this case, the Hasse diagram of the factors does not help to simplify regular expressions so we do not display it. (The fact that S occurs repeatedly is the only fact that is exploited.)

Fig. 22 depicts the subset ordering on subsets of the nodes that define factor graphs. Step 2 requires us to choose one pair (i,j) for each of these factor graphs: the transition graph formed from the factor graph by designating node i as start node and node j as finish node is a recogniser of $i\backslash j$. Labels of the form $i\backslash j$ have been added to each node.

Step 3 (the construction of the factor graphs) has already been completed so the final step is to calculate regular expressions denoting the entries in the factor matrix. Applying the algorithm as prescribed, we get the following sequence of matrices.

First, we compute the factor matrices of $4\backslash 4$ and $2\backslash 2$ (which happen to be equal).

Nodes, i.e. set of left factors of 3	Factor graph of these factors of 3
{1,2,3,4}	{1\3}
{2,3,4}	{2\3}
{1,2,4}	{1\4}
{2,4}	{2\4}
{3,4}	{4\3 , 3\3}
{1,2}	{1\2 , 1\1}
{4}	{4\4 , 3\4}
{2}	{2\2 , 3\2 , 4\2 , 2\1 , 3\1 , 4\1}

Figure 20: Nodes and Factor Graphs

$$\begin{bmatrix} \{\varepsilon,a,ab,aba\} & \{a,ab,aba\} & \{aba\} & \{ab,aba\} \\ S & S & \{ba,aba\} & \{b,ab,ba,aba,bab\} \\ S & S & \{\varepsilon,a,ba,aba\} & S \\ S & S & \{a,ba,aba\} & S \end{bmatrix}$$

Figure 21: Factor Matrix Expressed As Sets Of c-Classes

The computed expression is also entered wherever $4\backslash 4$ (or equally $2\backslash 2$) occurs in factor matrix. (It is the entry "S" in the matrix of sets of c-classes above.) We obtain the following, where a question mark indicates an entry that has not yet been computed.

$$\begin{bmatrix} ? & ? & ? & ? \\ (a+b)^* & (a+b)^* & ? & ? \\ (a+b)^* & (a+b)^* & ? & (a+b)^* \\ (a+b)^* & (a+b)^* & ? & (a+b)^* \end{bmatrix}$$

Second, we compute missing entries in the factor matrices of $2\backslash 4$, $4\backslash 3$ (and $3\backslash 3$) and $1\backslash 2$ (and $1\backslash 1$):

$$\begin{bmatrix} \varepsilon + a(a+b)^* & a(a+b)^* & ? & ? \\ (a+b)^* & (a+b)^* & ? & (a+b)^*\, b\, (a+b)^* \\ (a+b)^* & (a+b)^* & \varepsilon + (a+b)^* a & (a+b)^* \\ (a+b)^* & (a+b)^* & (a+b)^*\, a & (a+b)^* \end{bmatrix}$$

Third, we do the same for the factor matrices of $2\backslash 3$ and $1\backslash 4$:

$$\begin{bmatrix} \varepsilon + a(a+b)^* & a(a+b)^* & ? & a(a+b)^*\, b(a+b)^* \\ (a+b)^* & (a+b)^* & (a+b)^* b(a+b)^* a & (a+b)^* b(a+b)^* \\ (a+b)^* & (a+b)^* & \varepsilon + (a+b)^* a & (a+b)^* \\ (a+b)^* & (a+b)^* & (a+b)^*\, a & (a+b)^* \end{bmatrix}$$

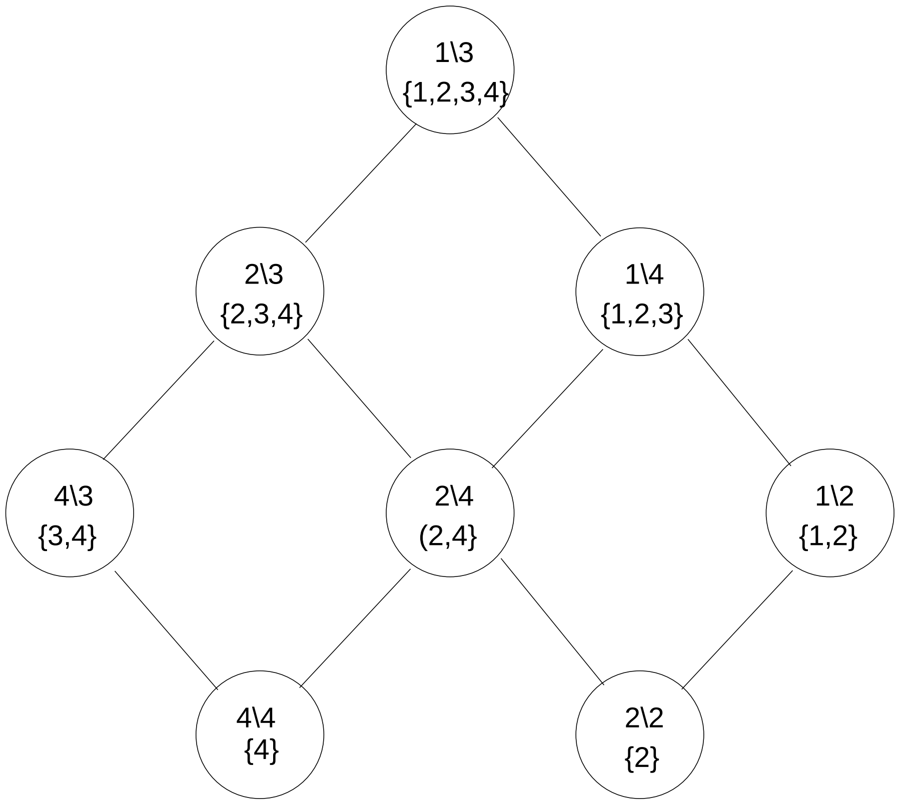


Figure 22: Ordering of (Sub)Factor Graphs

Finally, we complete the one remaining entry in the factor matrix of $1\backslash 3$:

$$\begin{bmatrix} \varepsilon + a(a+b)^* & a(a+b)^* & a(a+b)^*b(a+b)^*a & a(a+b)^*b(a+b)^* \\ (a+b)^* & (a+b)^* & (a+b)^*b(a+b)^*a & (a+b)^*b(a+b)^* \\ (a+b)^* & (a+b)^* & \varepsilon + (a+b)^*a & (a+b)^* \\ (a+b)^* & (a+b)^* & (a+b)^*a & (a+b)^* \end{bmatrix}$$

In this way we have calculated the regular expression $a(a+b)^*b(a+b)^*a$ denoting the event $1\backslash 3$.

9.4 Counterexample

Our algorithm does not always yield a regular expression of minimal star-height. A necessary condition for it to do so is that, for all regular languages E all of whose admissible[12] factors are inseparable from E, the star-height of E is equal to the rank

[12]Recall that T^* and $\emptyset$ are typically factors of E. A factor is admissible if it equals $U\backslash E/V$ for some events U and V both of which are different from $\emptyset$. The empty set, $\emptyset$, is thus inadmissible except for when it equals E; on the other hand, T^* may or may not be admissible.

of the factor graph of E. This section presents an example for which this is not the case. The example was discovered by testing our algorithm on all languages that others had investigated as part of their own research on the problem; the example below was introduced by McNaughton [McN69].

The language denoted by the regular expression $(b+aa+ac+aaa+aac)^*$ has star-height one. Its machine and anti-machine are shown in fig. 23.

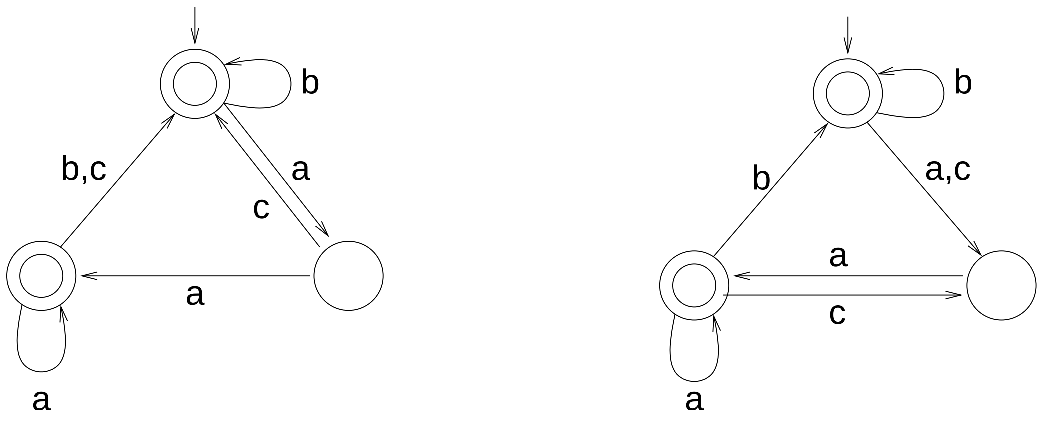


Figure 23: Machine and Anti-Machine

Its factor graph (omitting inadmissible nodes), shown in fig. 24 has cycle-rank 2 and all admissible factors are inseparable. (The nodes have been labelled in order to ease comparison with its factor matrix.)

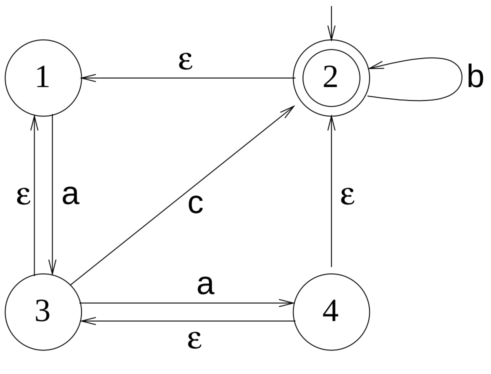


Figure 24: Factor Graph

This can be verified by comparing entries in the factor matrix. For this purpose, we use the syntactic monoid of the language shown in fig. 25.

The factor matrix is shown in fig. 26 (omitting inadmissible factors, as usual). The shortest word in each c-class is chosen as the representative element of the c-class.

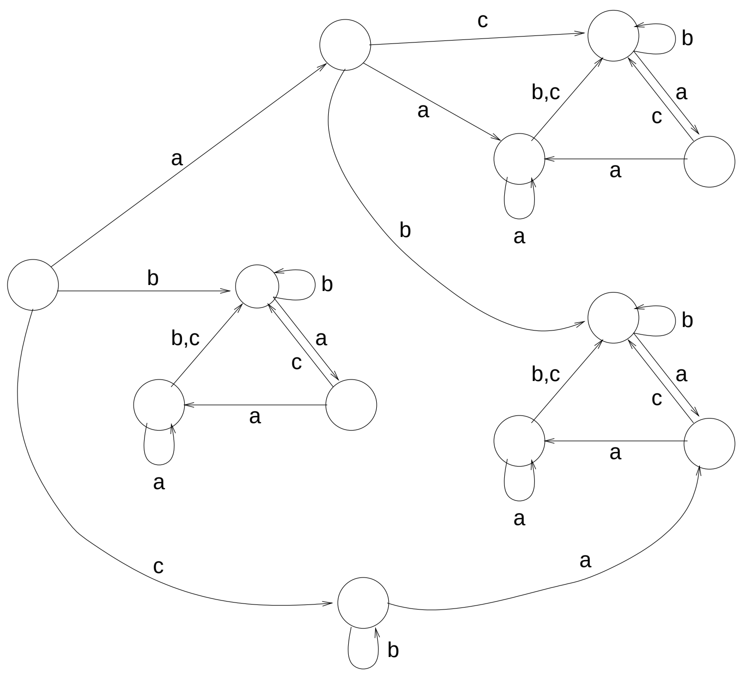


Figure 25: Syntactic Monoid

From this matrix, it is possible to determine that all admissible factors are inseparable.

$$\begin{bmatrix}
\{\varepsilon,a,aa,ac,aca\} & \{aa,ac\} & \{a,aa,aca\} & \{aa\} \\
\begin{array}{l}\{\varepsilon,a,aa,ac,aca,\\ b,ba,baa\}\end{array} & \begin{array}{l}\{\varepsilon,aa,ac,\\ b,baa\}\end{array} & \begin{array}{l}\{a,aa,aca,\\ ba,baa\}\end{array} & \begin{array}{l}\{aa,\\ baa\}\end{array} \\
\begin{array}{l}\{\varepsilon,a,aa,ab,ac,aca,\\ c,ca,caa\}\end{array} & \begin{array}{l}\{a,aa,ab,ac,\\ c,caa\}\end{array} & \begin{array}{l}\{\varepsilon,a,aa,aca,\\ ca,caa\}\end{array} & \begin{array}{l}\{a,aa,\\ caa\}\end{array} \\
\begin{array}{l}\{\varepsilon,a,aa,ab,ac,aca,\\ b,ba,baa,\\ c,ca,caa\}\end{array} & \begin{array}{l}\{\varepsilon,a,aa,ab,ac,\\ b,baa,\\ c,caa\}\end{array} & \begin{array}{l}\{\varepsilon,a,aa,aca,\\ ba,baa,\\ ca,caa\}\end{array} & \begin{array}{l}\{\varepsilon,a,aa,\\ baa,\\ caa\}\end{array}
\end{bmatrix}$$

Figure 26: The Factor Matrix

Since all (admissible) factors are inseparable, our closure algorithm reduces to a standard elimination technique; the star-height of the expression computed by the algorithm is thus at least the cycle-rank of the graph (whatever the order of elimination of nodes), which is greater than the star-height of the language. The conclusion is that our algorithm is not guaranteed to compute a regular expression of minimal star-height.

10 The "Universal Automaton"

Conway's "best" constant+linear approximation to a regular language E, denoted above by $\mathbf{C}_{max}.E \,\dot{\cup}\, \mathbf{L}_{max}.E$, has been studied by Lombardy and Sakarovitch [LS08]. It is fundamental to what they call the "universal automaton". There are many overlaps between their work and that in my thesis. In particular, they give the name "écorché" to what I called the "factor graph". This section discusses in some detail the overlap and the differences between Conway's contribution, my own contribution and the contribution of Lombardy and Sakarovitch.

Before doing so, it is necessary to briefly clarify the contribution made by Conway. In addition, it is necessary to make a number of detailed comments on the terminology used by Lombardy and Sakarovitch. (The differences in terminology complicate the comparison we wish to make.)

Although I have always attributed the introduction of $\mathbf{C}_{max}.E \,\dot{\cup}\, \mathbf{L}_{max}.E$ to Conway, he did not introduce it explicitly. Conway's concern was with the general notion of the "best constant+linear" approximation to the event E by a set of events —the image set of a function ζ— which we denote above by $\mathbf{C}_{max}(E,\zeta) \dot{\cup} \mathbf{L}_{max}(E,\zeta)$. The specific case that ζ is the (lifted) identity function —that is, the set of approximations is the alphabet of E— is a very obvious instance. No doubt, Conway did not consider it worth mentioning for the simple reason that the "best approximation" to E is then —also very obviously— E itself. It is because it is a very obvious instance that I chose to attribute it to Conway in my thesis without providing further explanation.

10.1 LS-style "Automaton"

The "universal automaton" [LS08] is essentially $\mathbf{L}_{max}.E$ together with adaptations of the start and final states of the automaton that incorporate $\mathbf{C}_{max}.E$. As Lombardy and Sakarovitch explain, the "universal automaton" is the topic of other publications that make no mention of Conway's factor theory and, indeed, its "unversality" (which we discuss shortly) is certainly not mentioned by Conway. Nevertheless, to anyone familiar with both Conway's theory and the publications cited by Lombardy and Sakarovitch, the connection is obvious. The precise details are, however, tricky to explain because Lombardy and Sakarovitch have a very curious definition of an "automaton" (in common with the papers that they cite).

In my thesis and in this paper, I have used the terminology "transition graph", in accordance with the terminology used by McNaughton [McN67, McN69] and many others; a "transition graph" is exactly what Conway calls a "constant+linear matrix". Together with a set of start states and a set of final states, a transition graph defines a recogniser of a language. The entries in the matrix (the edges of the graph) are its “transitions”. Importantly, a transition graph allows empty-word transitions: the empty-word transitions form the “constant” part of the graph. This is of great practical importance: in the Knuth-Morris-Pratt pattern-matching algorithm [KMP77] (and Aho and Corasick’s generalisation [AC75]), the empty-word transitions are the failure transitions. Lombardy and Sakarovitch [LS08] use the terminology “automaton” but, contrary to the convention that I am used to, give it a very curious definition (which, as we shall see, they do not adhere to). According to them, an automaton comprises a *linear* matrix, a set of start states and a set of final states. (Formally, [LS08, p.2] states that a transition of an automaton is an element of $Q{\times}A{\times}Q$ where Q is the set of states and A is the alphabet.) Below, I call this an *LS-style automaton.*

Given a transition graph $\mathbf{C} \mathbin{\dot\cup} \mathbf{L}$, where $\mathbf{C}$ is its constant part and $\mathbf{L}$ is its linear part, and appropriately defined selectors $\mathbf{S}$ and $\mathbf{F}$ for the start and final states (constant matrices of dimension $N{\times}\mathbb{1}$ where N is the set of nodes and $\mathbb{1}$ is a set with exactly one element), the language recognised is

$$\mathbf{S}^{\cup} \otimes (\mathbf{C} \mathbin{\dot\cup} \mathbf{L})^{*} \otimes \mathbf{F} \quad .$$

(Strictly, the above is a matrix of dimension $\mathbb{1}{\times}\mathbb{1}$ and the language recognised is its unique entry.) Using the star-decomposition rules, this is equal to

$$(\mathbf{S}^{\cup} \otimes \mathbf{C}^{*}) \otimes (\mathbf{L} \otimes \mathbf{C}^{*})^{*} \otimes \mathbf{F}$$

and to

$$\mathbf{S}^{\cup} \otimes (\mathbf{C}^{*} \otimes \mathbf{L})^{*} \otimes (\mathbf{C}^{*} \otimes \mathbf{F}) \quad .$$

In this way, the combination of a transition graph, a set of start states and a set of final states can be transformed into an LS-style automaton: either the set of start states is augmented so that its selector becomes $\mathbf{S}^{\cup} \otimes \mathbf{C}^{*}$ and the transition graph $\mathbf{C} \mathbin{\dot\cup} \mathbf{L}$ is replaced by $\mathbf{L} \otimes \mathbf{C}^{*}$ (which is a linear matrix), or the set of final states is augmented so that its selector becomes $\mathbf{C}^{*} \otimes \mathbf{F}$ and the transition graph $\mathbf{C} \mathbin{\dot\cup} \mathbf{L}$ is replaced by $\mathbf{C}^{*} \otimes \mathbf{L}$ (which is a linear matrix).

Now recall that the reflexive-transitive closure of the transition graph $\mathbf{C}_{max}.E \mathbin{\dot\cup} \mathbf{L}_{max}.E$ is the factor matrix of E and E is the (l,r)th entry in the factor matrix. Recall also that

$$\mathbf{L}_{max}.E \;=\; (\mathbf{C}_{max}.E)^{*} \otimes \mathbf{L}_{max}.E \;=\; \mathbf{L}_{max}.E \otimes (\mathbf{C}_{max}.E)^{*} \quad .$$

So, we have that

$$E \;=\; \mathbf{l}^{\cup} \otimes (\mathbf{C}_{max}.E \mathbin{\dot\cup} \mathbf{L}_{max}.E)^{*} \otimes \mathbf{r}$$

where $\mathbf{l}$ is the selector matrix corresponding to the unique start state l and $\mathbf{r}$ is the selector matrix corresponding to the unique final state r and, also,

$$\mathsf{E} \;=\; \mathbf{S}^{\cup} \otimes (\mathbf{L}_{max}.\mathsf{E})^{*} \otimes \mathbf{F}$$

where

$$\mathbf{S} \;=\; ((\mathbf{C}_{max}.\mathsf{E})^{\cup})^{*} \otimes \mathbf{l}$$

and

$$\mathbf{F} \;=\; (\mathbf{C}_{max}.\mathsf{E})^{*} \otimes \mathbf{r} \quad .$$

In this way, the combination of the transition graph $\mathbf{C}_{max}.\mathsf{E} \;\dot{\cup}\; \mathbf{L}_{max}.\mathsf{E}$, start state l and final state r correspond to an LS-style automaton comprising the linear matrix $\mathbf{L}_{max}.\mathsf{E}$, set of start states given by $((\mathbf{C}_{max}.\mathsf{E})^{\cup})^{*} \otimes \mathbf{l}$ and set of final states given by $(\mathbf{C}_{max}.\mathsf{E})^{*} \otimes \mathbf{r}$. The latter is what Lombardy and Sakarovitch call the "universal automaton".

For an example, the reader is invited to compare the "universal automaton" in [LS02, fig. 2], which has three start states, two final states, and eight edge labels of which none is the empty-word, with the factor graph of fig. 12, which has one start state, one final state and six edge labels of which two are the empty-word.

(Note that Lombardy and Sakarovitch choose to augment both the set of start states and the set of final states, rather than one or the other. This is valid in this case but not in general. One reason for doing so is in order not to introduce a bias between left and right. There may be other reasons too.)

This then is the basis for our assertion that Lombardy and Sakarovitch's "universal automaton" of a regular language E is essentially Conway's "best linear approximation" $\mathbf{L}_{max}.\mathsf{E}$ together with adaptations of the start and final states of the automaton that incorporate Conway's "best constant approximation" $\mathbf{C}_{max}.\mathsf{E}$.

The "unversality" of the "universal automaton" is the property that there is a "morphism" from any LS-style automaton that recognises any subset of E to $\mathbf{L}_{max}.\mathsf{E}$. (A pathwise homomorphism is a "morphism" but not necessarily the other way around. Lombardy and Sakarovitch [LS02, Definition 3] call a pathwise homomorphism —as defined here rather than McNaughton's definition— a "conformal" morphism.)

Conway makes no mention of the "universality" problem. In my view, the practicality of the problem is seriously undermined by the curious definition of an LS-style automaton. As mentioned several times above, the factor graph of $\mathsf{T}^{*} \cdot \{w\}$, for a given "pattern" w, defines a recogniser that is the basis of the Knuth-Morris-Pattern pattern-matching algorithm, but that recogniser is *not* an LS-style automaton. So the definition excludes the possibly best-known example of the practical contribution of automata theory. I suspect that it can be shown that $\mathbf{C}_{max}.\mathsf{E} \;\dot{\cup}\; \mathbf{L}_{max}.\mathsf{E}$ is "universal" relative to a definition of "automaton" that allows arbitrary transition graphs but leave that exercise to the reader.

Lombardy and Sakarovitch [LS08] do give some credit to Conway but (in my view) do not do so to the extent that is properly merited. In section 2.1 of their paper, for example, they introduce "subfactorisation", "factorisation", "left factor", "right factor", etc., exactly in the way that Conway does but with no reference to Conway's book. Several facts about factors due to Conway are stated but using unconventional terminology (for example, "left quotient" rather than Brzozowski's [Brz64] "derivative"). Later, in section 3.1, they introduce the factor matrix but do not credit it to Conway; also, their proposition 3.2 is an instance of a theorem stated by Conway [Con71, theorem 7, p.31] (since $\mathbf{C}_{max}.E \ \dot{\cup}\ \mathbf{L}_{max}.E$ is an instance of $\mathbf{C}_{max}(E,\zeta)\dot{\cup}\mathbf{L}_{max}(E,\zeta)$). However, apart from a brief reference to Conway in the introduction (which does attribute the "universal automaton" to Conway), there appears to be no further mention of Conway until section 6 when the approximation problem is introduced. But there is no mention of the fact that the "universal automaton" is obtained by specialising Conway's theory of approximations.

It has to be said, of course, that Conway gives no references whatsoever! The preface of [Con71] serves as an acknowledgement and claims that his work dates from 1966. But insufficient information is given for anyone wishing to trace its source.

10.2 Factor Graph = "Écorché"

Lombardy and Sakarovitch [LS08] claim that the "écorché" is due to Lombardy and cite his 2001 PhD thesis (which I have not read). It is obvious that it is identical to the factor graph that I introduced in my 1975 PhD thesis and presented in 1977 at a conference on Automata, Languages and Programming [Bac75, BL76, BL77]. Some remarks are, however, in order.

First, Lombardy and Sakarovitch [LS08, Definition 3.11] describe the "écorché" as an (LS-style) "automaton". Yet they allow empty-word transitions: the set of transitions is given as a set $D^L \cup H^L$, where L denotes the language (which Conway and I denote by E), D^L corresponds to our $\mathbf{C}_{min}.E$ and H^L corresponds to our $\mathbf{L}_{min}.E$. (They call empty-word transitions "spontaneous transitions".) Thus an "écorché" is not an "automaton" according to their own definition but is, indeed, a transition graph (as it must be) according to our definition. Second, just as for the factor graph, [LS08, Definition 3.11] defines an "écorché" as having a unique start and a unique final state. Yet the example that follows depicts the "écorché" as having multiple start states: the rightmost figure of [LS08, Figure 7] has two start states. To add to the confusion, the leftmost figure has a unique final state, and not two. Yet subsequent examples of the "écorché" depict both multiple start and multiple final states. See, for example, [LS08, Figure 11(c)] and [LS08, Figure 15].

Lombardy and Sakarovitch's only use of the factor graph appears to be as a more informative way of depicting the "universal automaton", in the same way that a Hasse diagram is used to depict a partial ordering on a finite set. They do not study its properties in the way that I did in my thesis; nor do they appear to recognise its relevance to the star-height problem. For pure-group events, which they also study in [LS02], the cycle ranks of $\mathbf{C}_{min}.E \dot{\cup} \mathbf{L}_{min}.E$ and $\mathbf{C}_{max}.E \dot{\cup} \mathbf{L}_{max}.E$ are equal. So, when the sole goal is to minimise star-height it makes no difference which is used. Relative to other measures of the complexity of the regular expressions obtained, the use of the former is far superior, but they choose to use the latter. I suspect that, in combination with their curious definition of an "automaton", this is why their figures do not comply with their definition.

10.3 "Complex" and "complicated to compute"

Lombardy and Sakarovitch introduce their paper by describing the universal automaton as follows.

It is large, it is complex, it is complicated to compute.

I would dispute this, in particular the claim that it is complicated to compute. As pointed out above, the very practical pattern matching algorithms developed by Knuth, Morris and Pratt [KMP77] and Aho and Corasick [AC75] boil down to computing the factor graph of a simple regular language. These algorithms are ingenious but not complicated.

In the general case, the use of representative elements of l- and r- classes of the given language E in the process of calculating $\mathbf{C}_{max}.E$, $\mathbf{L}_{max}.E$ and the factor graph of E makes the calculations very straightforward. Although Lombardy and Sakarovitch begin with Conway's characterisation of factors as *intersections* of derivatives [LS08, Proposition 2.1], they *do* appear to have recognised that factors are *unions* of c-classes [LS08, Proposition 3.5(iii) and Example 3.7] and that left factors are unions of l-classes [LS08, section 4.1, paragraph preceding theorem 4.1]. They do not, however, provide any examples of how these facts are exploited: [LS08, section 4], entitled "Construction of the universal automaton", is devoid of examples.

The reality is that the hardest part of constructing the factor graph is the construction of the machine and anti-machine. These are, indeed, non-trivial tasks (for example, the construction of the machine and anti-machine involve identifying "similar" derivatives of regular expressions [Brz64, Definition 5.2]) but they are tasks that are frequently asked of students of automata theory and, once completed, the remaining tasks are straight-forward manipulations of finite sets. The beauty of the KMP algorithm (and

Aho and Corasick's generalisation) is that it avoids the difficult task of constructing the appropriate machine. It may therefore be the case that constructing the factor graph of other quite different classes of languages is similarly easy and of practical importance. Identifying the factor graphs and factor matrices of the factors of a language exploits the syntactic monoid of the language and thus adds an extra layer of complexity to the calculations (both in theory and in practice, because the number of c-classes may be an exponential function of the number of l- or r-classes, and this bound is often attained). Even so, although the calculations may be long and tedious, they are straightforward.

10.4 Pure-group Events

My work in 1972 and 1973 was strongly motivated by McNaughton's algorithm [McN67] for determining the star-height of a pure-group event (a language whose syntactic monoid is a group). Very roughly, his algorithm involves searching a space of transition graphs, so-called μ-graphs, defined by sets of c-classes of the given language. The "universal automaton" is a particular example of a μ-graph. My conjecture was that it would be sufficient to consider just one graph rather than engage in a massive combinatorial search of all μ-graphs. The step from $\mathbf{C}_{max}.E \;\dot{\cup}\; \mathbf{L}_{max}.E$ to the factor graph was an obvious first step in exploring this conjecture, and the second step was to develop a novel closure algorithm that fully exploits the algebraic properties of languages. This strategy was supported by examining the example with which McNaughton ended his paper: the example of modulo addition (in particular addition modulo 6) that I have used as a running example here. My conjecture was that the factor graph would suffice in all cases, and not just for pure-group events. But that conjecture proved to be false (see section 9.4)! My disappointment was such that I did not return to the pure-group events.

Lombardy and Sakarovitch [LS02, LS08] do show that it suffices to consider just the "universal automaton", but their algorithm still involves a combinatorial search. On the basis of my, as yet limited, understanding of their paper, I strongly believe that the algorithm presented above is guaranteed to construct regular expressions of minimal star-height for all pure-group events but I have not attempted to find a proof. The algorithm is, however, likely to be of no practical value since regular expressions denoting pure-group events are typically very complicated and, therefore, have little value.

11 Conclusion

This paper has been about revisiting the results first formulated by the author more than forty years ago [Bac75], exploiting an improved understanding of the underlying theory. In the process, I have taken the opportunity to expand on and extend the theory.

However, the primary contribution is to present the theory in a calculational style, taking full advantage of the experience I have gained in developing regular algebra in application areas quite different from regular languages.

I began this exercise after being asked to contribute to a festschrift dedicated to José Nuno Oliveira [Bac16]. José is a devotee of algebra and, in particular, of Galois connections. So, to me, an obvious source of material was my thesis: although I had not been aware of the notion of a Galois connection when conducting the research for my PhD thesis (in 1972 and 1973), I began to recognise its relevance soon after the publication of Cousot and Cousot's [CC77, CC79] work on static analysis. The relevance became fully clear to me in the late 1980s when I began research on a relational theory of datatypes. (The algebra of relations is a special case of a regular algebra.) This led, for example, to the development of a calculational theory of well-foundedness [DBvdW97]. But by then I was too occupied with other developments to spare the time to revisit my thesis in detail.

As already mentioned, much of the content of this paper is a re-presentation of the results published[13] forty years ago in my thesis. The proof-style is substantially different but concepts and most theorems are unchanged.

Adopting a different proof-style has the advantage of offering a semi-independent check on the veracity of the claimed results. I am pleased to say that I have not found any errors in the thesis. (I had hoped that I might find an error in the example presented in section 9.4 demonstrating that I had been unable to solve the star-height problem. However, that was not to be!)

A major novel contribution of this paper is the exposition of Conway's theory of approximations in section 5. I felt it necessary to include because Conway's work appears to be little understood but might offer new insights on the star-height problem I also wanted to use it as another example of Galois connections — one that does not appear to have been recognised even now, so long after the publication of Conway's book.

Some of the examples have changed — I have added additional examples but also omitted some examples — but, more importantly, modern technology[14] has enabled me

[13]The meaning of the word "published" has changed in the last forty years. Forty years ago, a PhD thesis was regarded as a refereed publication: theses were readily available via copyright libraries such as the Science Museum Library in London (which I used often); publications in a journal could take several years to appear and it was vital to keep abreast of other theses in order to verify the originality of one's own work. Nowadays, given the vastly greater number of conferences and journals (and the greater speed of publication), a thesis is unlikely to be regarded as a "publication".

[14]Forty years ago, a hand-written manuscript was turned into a type-script using a golfball type-writer. Mathematical symbols and ordinary text were entered using different "golf-balls". This was extremely time-consuming and error-prone. In contrast, this document was prepared almost entirely on-screen using Math/pad [BVW97], a system that enables WYSIWYG-like editing of mathematical documents which are then exported into LaTeX.

to expand in much more detail on the examples I present. For instance, the running example of modulo addition first introduced in example 16 is a substantial amplification of the presentation in my thesis.

The closure algorithm presented in section 9 has been improved: effectively, the algorithm proposed in my thesis involved searching for an optimal total ordering of the sets in the set $\langle\cup i,j :: \{N.(i,j)\}\rangle$; instead, the algorithm presented here processes these sets in topological order, making more use of the syntactic monoid to optimise the regular expressions that are obtained. The primary conclusion of section 9 is unaffected: Eggan's notion of cycle-rank is an inappropriate measure on the factor graph of a language since, using the algorithm, it is possible to construct regular expressions denoting the given language that may have star-height strictly less than the cycle-rank of the graph. The change does not, however, affect the conclusion that the algorithm may not construct a regular expression of minimal star height.

As mentioned in the introduction, surprisingly little has been written on Conway's factor theory since the publication of his book in 1971, and it appears to have taken more than 25 years before my notion of the "factor graph" was rediscovered. I find this surprising because, for me, the property that the factor matrix is its own star immediately suggested investigating whether or not it has a minimal starth root. Indeed, the practical lesson I had learnt from investigating the relation between closure algorithms in regular algebra and algorithms for inverting matrices in linear algebra [BC75] was that, given a matrix $\mathbf{A}$, one should try to avoid calculating $\mathbf{A}^*$ at all costs: it is much more informative to calculate its starth root! (For example, a Hasse diagram of a partial ordering is much more useful than a graph depicting the partial ordering.) This is how I concluded my presentation at the ICALP meeting in 1977 and it remains the most important conclusion to this day.

References

[AC75] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[Bac75] R.C. Backhouse. *Closure algorithms and the star-height problem of regular languages*. PhD thesis, University of London, 1975. Scanned-in copy of the chapters on factor theory available from `www.cs.nott.ac.uk/~psarb2/MPC/FactorGraphs.pdf`.

[Bac02] Roland Backhouse. Galois connections and fixed point calculus. In Roland Backhouse, Roy Crole, and Jeremy Gibbons, editors, *Algebraic and Coal-*

gebraic Methods in the Mathematics of Program Construction, volume 2297 of *LNCS Tutorial*, chapter 4, pages 89–148. Springer, 2002. International Summer School and Workshop, Oxford, UK, April 20000, Revised Lectures.

[Bac06] Roland Backhouse. Regular algebra applied to language problems. *Journal of Logic and Algebraic Programming*, 66:71–111, 2006.

[Bac16] Roland Backhouse. Factor theory and the unity of opposites. *J. Logical and Algebraic Methods in Programming*, 85(5):824–846, 2016.

[BC75] R.C. Backhouse and B.A. Carré. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and its Applications*, 15:161–186, 1975.

[BL76] Roland C. Backhouse and Rüdiger K. Lutz. Factor theory, failure functions and bi-trees. Technical Report 4, Department of Computer Science, Heriot-Watt University, October 1976.

[BL77] R.C. Backhouse and R.K. Lutz. Factor graphs, failure functions and bi-trees. In A. Salomaa and M. Steinby, editors, *Fourth Colloquium on Automata, Languages and Programming*, pages 61–75. Springer-Verlag, LNCS 52, July 1977.

[Brz64] J.A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, October 1964.

[Brz67] J.A. Brzozowski. Roots of star events. *Journal of the ACM*, 14(3):466–477, July 1967.

[BvdEvG94] Roland C. Backhouse, J.P.H.W. van den Eijnde, and A.J.M. van Gasteren. Calculating path algorithms. *Science of Computer Programming*, 22(1–2):3–19, 1994.

[BVW97] R.C. Backhouse, R. Verhoeven, and O. Weber. Math/pad: A system for on-line preparation of mathematical documents. *Software – Concepts and Tools*, 18:80–89, 1997.

[CC77] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unifed lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, January 1977.

[CC79] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, January 1979.

[Con71] J.H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, 1971.

[DBvdW97] H. Doornbos, R.C. Backhouse, and J. van der Woude. A calculational approach to mathematical induction. *Theoretical Computer Science*, 179(1–2):103–135, 1997.

[Egg63] L.C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10(4):385–397, 1963.

[KMP77] D.E. Knuth, J.H. Morris, and V.R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6:325–350, June 1977.

[LS02] Sylvain Lombardy and Jacques Sakarovitch. Star height of reversible languages and universal automata. In *Latin American Symposium on Theoretical Informatics*, pages 76–90. Springer, 2002.

[LS08] Sylvain Lombardy and Jacques Sakarovitch. The universal automaton. *Logic and automata*, 2:457–504, 2008.

[McN67] R. McNaughton. The loop complexity of pure-group events. *Info. and Control*, 11:167–176, 1967.

[McN69] R. McNaughton. The loop complexity of regular events. *Information Sciences*, 1:305–328, 1969.

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J.Research and Development*, 1959.

[Sal66] Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. Assoc. Comp. Mach.*, 13(1):158–169, January 1966.

[Wei73] P. Weiner. Linear pattern matching algorithms. In *Conf. Record IEEE 14th Annual Symposium on Switching and Automata*, pages 1–11, 1973.

All cited publications by the author are freely available from the author's website at the University of Nottingham, or from ResearchGate. (Some are preprints. Older publications are scanned-in and published with permission of the publisher.) An exception is the author's PhD thesis: only that part of the thesis on factor theory is available. Note that URLs are likely to change over time.

Appendix A. Properties of Constant and Linear Matrices

Here we prove lemma 127. Recall that $\mathbf{A} = \mathbf{A}^*$, $\mathbf{C} = \mathbf{A} \dot{\cap} \mathsf{Mat}.\{\varepsilon\}$ and $\mathbf{L} = \mathbf{A} \dot{\cap} \mathsf{Mat}.\mathsf{T}$. First,

$\mathbf{C} = \mathbf{C}^*$

$=$ { for all $\mathbf{X}$, $\mathbf{X} \dot{\subseteq} \mathbf{X}^*$ }

$\mathbf{C} \dot{\supseteq} \mathbf{C}^*$

$\Leftarrow$ { definition of * }

$\mathbf{C} \dot{\supseteq} \mathbf{I} \dot{\cup} \mathbf{C} \otimes \mathbf{C}$

$=$ { $\mathbf{C} \dot{\supseteq} \mathbf{I}$

  $=$ { definition of $\mathbf{C}$ }

  $\mathbf{A} \dot{\supseteq} \mathbf{I} \wedge \mathsf{Mat}.\{\varepsilon\} \dot{\supseteq} \mathbf{I}$

  $=$ { $\mathbf{A} = \mathbf{A}^* \dot{\supseteq} \mathbf{I}$; definition of Mat }

  true . }

$\mathbf{C} \dot{\supseteq} \mathbf{C} \otimes \mathbf{C}$

$=$ { definition of $\mathbf{C}$, definition of $\dot{\cap}$ }

$\mathbf{A} \dot{\supseteq} \mathbf{C} \otimes \mathbf{C} \ \wedge\ \mathsf{Mat}.\{\varepsilon\} \dot{\supseteq} \mathbf{C} \otimes \mathbf{C}$

$=$ { $\mathsf{Mat}.\{\varepsilon\} \dot{\supseteq} \mathbf{C} \otimes \mathbf{C}$

  $\Leftarrow$ { $\mathsf{Mat}.\{\varepsilon\} \dot{\supseteq} \mathbf{C}$, monotonicity and transitivity }

  $\mathsf{Mat}.\{\varepsilon\} \dot{\supseteq} \mathsf{Mat}.\{\varepsilon\} \otimes \mathsf{Mat}.\{\varepsilon\}$

  $=$ { definition of $\mathsf{Mat}.\{\varepsilon\}$ }

  true . }

$\mathbf{A} \dot{\supseteq} \mathbf{C} \otimes \mathbf{C}$

$\Leftarrow$ { $\mathbf{A} = \mathbf{A}^*$. So $\mathbf{A} \dot{\supseteq} \mathbf{A} \otimes \mathbf{A}$, monotonicity of product }

$\mathbf{A} \dot{\supseteq} \mathbf{C}$

$=$ { definition of $\mathbf{L}$ and $\mathbf{C}$ }

true .

The proof that $\mathbf{L} = \mathbf{L} \otimes \mathbf{C}$ is very similar to the latter part of the above proof:

$\mathbf{L} = \mathbf{L} \otimes \mathbf{C}$

$=$ { $\mathbf{A} = \mathbf{A}^* \dot{\supseteq} \mathbf{I}$, so $\mathbf{I} \dot{\subseteq} \mathbf{C}$, monotonicity of product }

$\mathbf{L} \dot{\supseteq} \mathbf{L} \otimes \mathbf{C}$

$=$ { $\mathbf{L} = \mathbf{A} \dot{\cap} \mathsf{Mat.T}$, $\mathsf{Mat.T} \dot{\supseteq} \mathsf{Mat.T} \otimes \mathbf{C}$ }

$\mathbf{A} \dot{\supseteq} \mathbf{L} \otimes \mathbf{C} \ \wedge\ \mathsf{Mat.T} \dot{\supseteq} \mathbf{L} \otimes \mathbf{C}$

$=$ { $\mathsf{Mat.T} \dot{\supseteq} \mathbf{L} \otimes \mathbf{C}$

$\Leftarrow$ { $\mathsf{Mat.\{T\}} \dot{\supseteq} \mathbf{L}$, $\mathsf{Mat.}\{\varepsilon\} \dot{\supseteq} \mathbf{C}$,
monotonicity and transitivity }

$\mathsf{Mat.\{T\}} \dot{\supseteq} \mathsf{Mat.\{T\}} \otimes \mathsf{Mat.}\{\varepsilon\}$

$=$ { definitions of $\mathsf{Mat.\{T\}}$ and $\mathsf{Mat.}\{\varepsilon\}$ }

true .

$\mathsf{Mat.T} \dot{\supseteq} \mathsf{Mat.T} \otimes \mathbf{C} \dot{\supseteq} \mathbf{L} \otimes \mathbf{C}$ }

$\mathbf{A} \dot{\supseteq} \mathbf{L} \otimes \mathbf{C}$

$\Leftarrow$ { $\mathbf{A} = \mathbf{A}^*$. So $\mathbf{A} \dot{\supseteq} \mathbf{A} \otimes \mathbf{A}$, monotonicity of product }

$\mathbf{A} \dot{\supseteq} \mathbf{L} \ \wedge\ \mathbf{A} \dot{\supseteq} \mathbf{C}$

$=$ { definition of $\mathbf{L}$ and $\mathbf{C}$ }

true .

A symmetric calculation shows that $\mathbf{L} = \mathbf{C} \otimes \mathbf{L}$. Now,

$(\mathbf{C} \dot{\cup} \mathbf{L})^*$

$=$ { star decomposition }

$\mathbf{C}^* \otimes (\mathbf{L} \otimes \mathbf{C}^*)^*$

$=$ { $\mathbf{C} = \mathbf{C}^*$, $\mathbf{L} = \mathbf{L} \otimes \mathbf{C}$ }

$\mathbf{C} \otimes \mathbf{L}^*$

$=$ { $\mathbf{L}^* = \mathbf{I} \dot{\cup} \mathbf{L} \otimes \mathbf{L}^*$, distributivity, }

$\mathbf{C} \ \dot{\cup}\ \mathbf{C} \otimes \mathbf{L} \otimes \mathbf{L}^*$

$=$ { $\mathbf{L} = \mathbf{C} \otimes \mathbf{L}$ }

$\mathbf{C} \ \dot{\cup}\ \mathbf{L} \otimes \mathbf{L}^*$.

This establishes that $(\mathbf{C} \dot{\cup} \mathbf{L})^* = \mathbf{C} \otimes \mathbf{L}^* = \mathbf{C} \ \dot{\cup}\ \mathbf{L}^+$. That $(\mathbf{C} \dot{\cup} \mathbf{L})^* = \mathbf{L}^* \otimes \mathbf{C}$ follows by using the symmetric form of star decomposition in the first step. Finally,

$\mathbf{L} \otimes (\mathbf{C} \dot{\cup} \mathbf{L})^*$

$=\quad\{\quad (\mathbf{C}\dot{\cup}\mathbf{L})^* = \mathbf{C}\ \dot{\cup}\ \mathbf{L}^+$, distributivity $\quad\}$

$\quad\mathbf{L}\otimes\mathbf{C}\ \dot{\cup}\ \mathbf{L}\otimes\mathbf{L}^+$

$=\quad\{\quad \mathbf{L} = \mathbf{L}\otimes\mathbf{C}\,,\ \mathbf{L}^+ = \mathbf{L}\ \dot{\cup}\ \mathbf{L}\otimes\mathbf{L}^+ \quad\}$

$\quad\mathbf{L}^+$.

Now we prove lemma 151.

Appendix B

Here we demonstrate that the algorithms for finding $\mathbf{G}^*$ that are loosely called "elimination methods" invariably result in regular expressions having star-height at least the rank of $\mathbf{G}$. We begin by abstracting the salient features of an elimination method and then show that any elimination method defines an "order of elimination of edges". We then investigate in detail the effect of eliminating edges of the graph, and finally show how we can build an "analysis" of the edges. (The assumption in this section is that graphs are finite and a "matrix" is a square finite-dimensional array.)

Before doing so, we need some definitions and a fundamental theorem from McNaughton [McN69]. A *subgraph* of a graph $\mathbf{G}$ is defined by a subset of the nodes of $\mathbf{G}$: the set of edges of the subgraph is the set of all edges that are to and from a node in the given subset. A *section* of a graph is a maximal connected component of the graph. An *analysis* of a graph $\mathbf{G}$ is a partial ordering of pairs $(\mathsf{N}',\mathbf{G}')$, where $\mathbf{G}'$ is a strongly connected component of $\mathbf{G}$ and N' is a node of $\mathbf{G}'$, having the following properties:

1. For each section H of $\mathbf{G}$ there is a node N of H such that (N,H) is maximal in the partial ordering.

2. For no subgraph H are there two nodes N and N' such that (N,H) and (N',H) occur in the partial ordering.

3. If (N,H) occurs and K is a section of that subgraph that has all the nodes of H except N, then, for some N', (N',K) is an immediate inferior of (N,H) in the ordering.

4. All of the immediate inferiors of (N,H) are of the kind mentioned in 3.

An analysis of a graph is always a forest of as many trees as the graph has sections. The *height* of the analysis is the length of the maximal length chain in the partial ordering.

Theorem 217 (McNaughton [McN69]) The rank of a graph is the minimum height of all analyses of the graph.

□

Our aim is to show how an "elimination" method defines an analysis of the graph such that the height of the analysis is at most the maximum star-height of regular expressions computed by the method. First we state more precisely what we mean by an elimination method.

Definition 218 An *elementary matrix* is a matrix whose non-null elements all lie in the same row, or the same column. An *elementary graph* is a graph whose edges are all from or all to a single node.
□

Clearly, elementary matrices represent elementary graphs.

We note that it is "elementary" to find the closure of a row or column matrix using the identity $(H \otimes K)^* = \mathbf{I} \dot{\cup} H \otimes (K \otimes H)^* \times K$. (Hint: an $n \times n$ row or column matrix can always be decomposed into $H \otimes K$ where H and K have dimensions $n \times 1$ and $1 \times n$, respectively. Then $K \otimes H$ is a 1×1 matrix — i.e. a single entry.) Hence the terminology.

Definition 219 An *elementary elimination step* is a step in an algorithm that involves solely the computation of the closure of an elementary matrix.
□

We can abstract three essential features of the methods discussed in [BC75].

1. The star-decomposition rules

   (220) $(C \dot{\cup} D)^* = (C^* \otimes D)^* \otimes C^*$

   and

   (221) $(C \dot{\cup} D)^* = C^* \otimes (D \otimes C^*)^*$

   are applied exclusively to derive an expression for A^* as a product

   (222) $J_1^* \otimes J_2^* \otimes \ldots \otimes J_m^*$

   of elementary matrices.

If at some stage in the derivation of the product form the matrix B is split into matrices C and D such that $B = C \dot{\cup} D$, and one of the two star-decomposition rules is used, then C and D are chosen so that

2. C is null wherever D is non-null and, vice-versa, D is null wherever C is non-null, and

3. if (220) is used, $C^* \otimes D$ is null wherever C is non-null and vice-versa, and if (221) is used $D \otimes C^*$ is null wherever C is non-null and vice-versa.

Some terminology is useful here. We refer to calculating the closure C^* of C as *eliminating* C. Evaluating the product $C^* \otimes D$ (or $D \otimes C^*$, as the case may be) is called *forward substitution* of C in D, and finding the product $(C^* \otimes D)^* \otimes C^*$ (or $C^* \otimes (D \otimes C^*)^*$) is called *back substitution* of D in C. An *elimination method* consists of expressing the computation of A^* as a sequence of forward and back substitutions and elementary elimination steps, in which subsequences of these steps may be interpreted collectively as eliminating C for some matrix C.

The definition of an elimination method has a number of implications which we now consider. Firstly, an elimination method always defines an ordering on the edges of $\mathbf{G}$ that we call the *order of elimination* of the edges. Specifically, if at some stage (220) (or (221)) is used we say that the edges of C are eliminated before the edges of D. By virtue of 2 and 3 this ordering is well-defined and total, except that the edges of each elementary matrix J are incommensurate — these are eliminated simultaneously.

Secondly, conditions 2 and 3 imply that we can always evaluate the non-null elements of J_1, J_2, ..., J_m by successive transformations of $\mathbf{G}$. Specifically, we can set $M^{(0)}$ to $\mathbf{G}$, then perform in-situ modifications of the elements of M to transform it to a matrix $M^{(f)}$ which contains the non-null elements of the matrices J_1^*, J_2^*, ..., J_m^* (other than the empty-word entries on the diagonal) in their appropriate positions. If, for instance, at some stage in the derivation of (222) the formula (220) is used, the appropriate action would be to evaluate $C^* \otimes D$ (possibly using additional storage) and store the non-null elements of this matrix in the appropriate positions of M. The remaining elements of M are left unchanged. Once $M^{(f)}$ has been calculated, $\mathbf{G}^*$ can be calculated using (222). (This may not be the most efficient way of evaluating $\mathbf{G}^*$ by the particular elimination method but our concern here is solely with the star-height of the resulting regular expressions.)

From the order of elimination of the edges of $\mathbf{G}$ we can construct an analysis of $\mathbf{G}$. We begin with the edges eliminated first and proceed in order to the edges eliminated last. Suppose at some stage we have considered the edges of the subgraph $\mathbf{G}_1$ and have constructed an analysis of $\mathbf{G}_1$. Suppose the constituent trees of this analysis have roots given by the set R_1. Suppose that J is the set of edges to be eliminated in the next elementary elimination step, and all edges in J are in row/column i; suppose $\mathbf{G}_2 = \mathbf{G}_1 \cup J$. If the edge (i, i) is not in J then do not alter the analysis . Otherwise, consider the largest subset R of R_1 such that each node r in R is strongly connected to node i in the graph $\mathbf{G}_2$. If R is non-empty, add a new root labelled i to the forest and connect it by branches to each element of R. If R is empty and the (i, i)th element of M is not the empty set before elimination, add a new root labelled i to the forest (but do not connect it to any other roots); if the (i,i)th element of M is the empty set before elimination do not alter the analysis. Finally, in all cases reset $\mathbf{G}_1$ to $\mathbf{G}_2$.

Now we need to related the height of the analysis to the star-height of expressions in $\mathbf{G}^*$.

To investigate what actually happens when we perform an elimination step, let us suppose that at some stage

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

where M_{11}, M_{22} and M_{33} are square matrices. We make no assumptions about the size of the various submatrices.

Suppose that the next step is to eliminate some subset of these 9 submatrices. A number of remarks are in order.

Remark 1. If the submatrix M_{ij} is an element of the subset to be eliminated, the result of all previous eliminations will have been forward-substituted into M_{ij}.

(To see this consider the context-free grammar

$$E ::= EfEb \quad ; \quad E ::= e$$

where f represents forward substitutions, b represents back substitutions, and e represents elementary eliminations. An elimination algorithm for finding $\mathbf{G}^*$ may be regarded as constructing a left-to-right, bottum-up parse of a sentence of this grammar. If an E has just been recognised an f must follow, possibly preceded by bs, before a new E may be recognised. The remark may now be proved by induction on the length of the derivation of the current state of the parse.)

Remark 2. No submatrix M_{ij} for $i \neq j$ may be eliminated before either M_{ii} or M_{jj} is eliminated without violating condition 3.

(Suppose otherwise. By remark 1, the result of elimination M_{ij} must be forward-substituted into M_{ii} and M_{jj} before they are eliminated. But this will result in a modification of M_{ij} itself, thus violating condition 3 — either it is pre-multiplied b M_{ii} or post-multiplied by M_{jj} depending on which of (220) or (221) is used at the time.)

Remark 3. If M_{ii} and M_{ij} have been eliminated, M_{ji} may not be eliminated after M_{jj}.

(If M_{ii} and M_{ij} have been eliminated then, by remark 1, the results must be forward-substituted into M_{ji} before this is eliminated. This involves pre- or post-multiplying M_{ji} by $M_{ii}^* \cdot M_{ij}$. However, post-multiplication changes M_{ii}, violating condition 3. Pre-multiplication changes M_{jj} and thus also violates condition 3 unless M_{jj} has not already been eliminated. Hence the remark.

Remark 4. M_{ik} and M_{ji}, where $k \neq i$ and $j \neq i$, may not both be eliminated using a single application of (220) or (221) without violating condition 3 since this would in general affect M_{jk}.

Remark 5. The star-height of elements of M is increased if and only if an elementary elimination step is executed in which the edge (k,k) is eliminated for some k.

This is obvious because substitutions only involve multiplication of submatrices of M.

A consequence of the above remarks is that, without loss of generality, we may assume that some subset of the matrices M_{11}, M_{12}, M_{13}, M_{21}, M_{31} is to be eliminated (i.e. some subset of the first row or the first column of M).

Let us suppose, the next step is to eliminate M_{11} and M_{12}. (The case of eliminating M_{11} and M_{21} can be considered similarly.) That is, suppose the step is to execute the assignment

$$M := \begin{bmatrix} M_{11}^* & M_{11}^* \otimes M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

In view of remark 5, we would like to see what effect this has on M_{22} and M_{33}. Suppose, therefore, that at some later stage we wish to eliminate the edges of M_{22}. By remark 3, we must already have eliminated M_{21} or must do it simultaneously with the elimination of M_{22}. Hence, by remark 1, M_{11}^* and $M_{11}^* \otimes M_{12}$ must already have been forward-substituted into M_{21} and M_{22}. In order not to violate condition 3, this can only be done by post-multiplying by M_{11}^* and $M_{11}^* \otimes M_{12}$. Thus after the forward substitution

$$M \supseteq \begin{bmatrix} M_{11}^* & M_{11}^* \otimes M_{12} & M_{13} \\ M_{21} \otimes M_{11}^* & M_{22} \dot{\cup} M_{21} \otimes M_{11}^* \otimes M_{12} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

(Containment rather than equality is used here because the elements of M may also have been changed between the elimination of M_{11} and M_{12} and their forward substitution into M_{21} and M_{22}.) Note that the same result will be obtained if we consider M_{21} and M_{22} eliminated separately or simultaneously.

Thus we see that the star-height of expressions obtained by eliminating submatrices of M_{22} will now depend on the star-height of elements in M_{11} provided $M_{21} \otimes M_{11}^* \otimes M_{12}$ is non-null. In contrast, if we eliminate submatrices of M_{33} before eliminating M_{31} or M_{13}, the star-height of the resulting regular expressions will not depend on the star-height of elements of M_{11} since forward substitution of M_{21} and M_{11} does not affect M_{33}.

We restate this in the form of a lemma on the elements of the matrix M.

Lemma 223 At some stage in the elimination processs, let $M = [m_{ij}]$ and let the set of edges that have currently been eliminated be $\mathbf{G}_1$. Suppose edge (r, r) is the last diagonal edge to have been eliminated in an elementary elimination step, and the next step in the elimination process involves the elimination of the edge (i, i). Suppose after this step the set of eliminated edges is $\mathbf{G}_2$. Then the entry m_{ii} has the form

$$u + v\ m_{rr}\ w$$

(after possibly exploiting the symmetry of addition of regular expressions) for some expression u and some non-empty expressions v and w, if node i is strongly connected to node r in $\mathbf{G}_2$.

Proof The lemma follows from the previous discussion by considering r as a node of M_{11} and i as a node of M_{22}. The conditions on r imply that the (i, i)th entry of $M_{21} \otimes M_{11}^* \otimes M_{12}$ is a sum of regular expressions, one of which is $v\ m_{rr}\ w$.
□

Lemma 224 The star-height of m_{ii} after elimination of (i, i) is 0 if i is not a node in the analysis and otherwise isa at least the height of the tree with root i.

Proof This follows easily from lemma 223 by induction on the height of the tree. If the height is 0 or 1, the lemma is obvious. Otherwise, recalling the construction of the analysis, each node r in the subset R satisfies the properties in lemma 223. Hence after elimination of (i, i), the (i, i)th entry of M is an expression of the form $(u + v\ m_{rr}\ w)^*$. (It has this form for each r in R after, of course, rearranging terms in the summation. For example, if R has two elements r and s, the entry would have the form $(u + v\ m_{rr}\ w + x\ m_{ss}\ y)^*$ for some u, v, w, x and y.) By induction, for each r in R, m_{rr} has star-height at least the height of the tree with root r. Hence m_{ii} has star-height at least 1 greater than this. Hence the lemma.
□

We could strengthen the lemma to an equality, as did Eggan [Egg63] for the escalator method, but this is not relevant here.

Combining theorem 217 and lemma 224, we have our theorem:

Theorem 225 If an elimination method is used to find $\mathbf{G}^*$ for a graph $\mathbf{G}$, then $\mathbf{G}^*$ will contain regular expressions having star-height at least the rank of $\mathbf{G}$.

□

Appendix C

This section gives practical advice on how to calculate the syntactic monoid of a language given a machine for the language. The advice is particularly relevant for hand calculations when the size of the syntactic monoid is not too large.

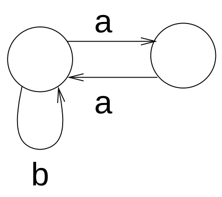
Fig. 27(a) is the machine for the language $(aa+b)^*$. The start and final states are not marked because the information is irrelevant to the construction of the syntactic monoid.) There is also one inadmissible state which has also been omitted from the diagram.

The construction of the syntactic monoid involves constructing graphs that depict $Ctx.u$ for words u in increasing lexicographic order. Formally, $Ctx.u$ is a total relation on the states of the machine. However, it is safe to ignore the inadmissible state of the machine; $Ctx.u$ is then a partial relation on the admissible states of the machine.

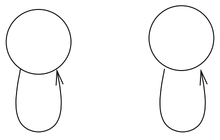
The relation $Ctx.\varepsilon$ is the identity relation, shown in fig. 27(b). The other relations that form the admissible elements of the syntactic monoid are shown in figs. 27(c) thru (g). Comparison of these figures with transitions in the machine should enable the reader to see how they are constructed. For example, the graph of $Ctx.a$ is the subgraph of the machine defined by the a-transitions, and similarly for $Ctx.b$. The graph of $Ctx.ab$ has only one edge because there is only one path in the machine that spells the word ab. Missing are inadmissible elements of the syntactic monoid: words u such that $Ctx.u$ restricted to admissible states of the machine is the empty relation. Examples are bab and $abab$.

The graphs in fig. 27 form a complete set because no new graphs are generated by words greater in the lexicographic order. Fig. 28 shows the complete syntactic monoid, except for the one inadmissible element. The nodes are labelled by a representative element of the c-class. The graph depicts post-multiplication: in order to compute the value of $u{\circ}v$, start from the node labelled u and follow the path spelled by v. (If there is no such path, the value of $u{\circ}v$ is the inadmissible element of the syntactic monoid.)

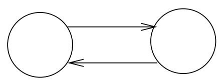
Of course, the same process can be applied starting with the anti-machine. In that case, the graph obtained depicts pre-multiplication in the syntactic monoid.



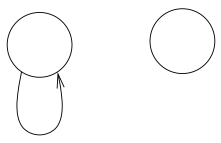
(a) Machine



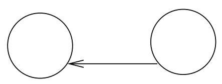
(b) Ctx. ε



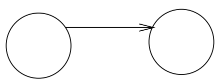
(c) Ctx.a



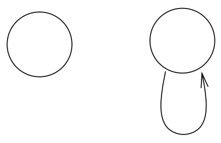
(d) Ctx.b



(e) Ctx.ab



(f) Ctx.ba



(g) Ctx.aba

Figure 27: Machine and Individual Elements of the Syntactic Monoid

Appendix D

Example 226 This example is based on [LS08, example 5.7]. Consider the language recognised by the nondeterministic finite-state machine shown in fig. 29. Lombardy and Sakarovitch have given the name $\mathcal{Z}_3$ to this automaton.

Note that fig. 29 omits one inadmissible node: there is no `b`-edge from the node labelled `0`. Throughout the following discussion, inadmissible values (nodes, classes, etc.) are omitted everywhere.

Our interest in this example is in calculating a regular expression of minimal star-height denoting the language recognised by this machine. The transition graph in fig. 29 has cycle rank two, so our objective is to construct a regular expression of star-height

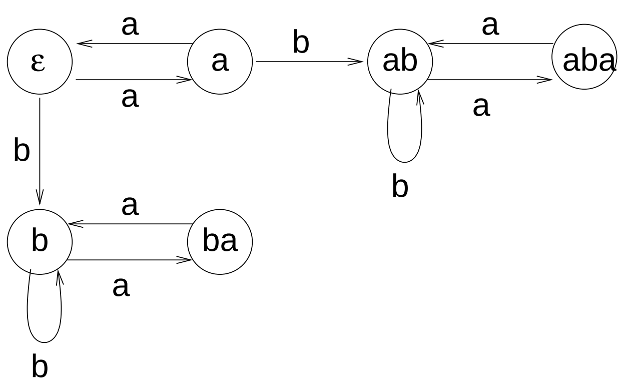


Figure 28: The Syntactic Monoid

one.

Lombardy and Sakarovitch [LS02, LS08] use this example as one of a series that demonstrates how big factor graphs can grow. Its size precludes us from giving all the details of the application of our closure algorithm. Our presentation may therefore seem somewhat *ad hoc*, but this is not the case.

Fig. 30 is the (deterministic) machine. Its anti-machine is identical. Note that there are a-transitions from every state but no b-transition from the state labelled $baba$. The table below names admissible l- and r-classes and selects a representative of each. Omitted from the graph is the inadmissible l-class of which a representative element is $babab$.

l-class	representative	r-class	representative
l1	ε	r1	ε
l2	b	r2	b
l3	ba	r3	ab
l4	baa	r4	aab
l5	bab	r5	bab
l6	baab	r6	baab
l7	baba	r7	abab

The following table shows corresponding left and right factors. Each is represented by a subset of l- or r-classes. The symbol "¬" denotes complement with respect to the admissible classes. For example, ¬{baba,bab} is {ε,b,ba,baa,baab} and, in the left-factor column, ¬∅ denotes the set of all admissible l-classes, and, in the right-factor column, ¬∅ denotes the set of all admissible r-classes. (The primary reason for using this notational trick is to emphasise symmetries rather than for greater brevity.)

There are 18 entries in the table. Together with the two inadmissible left/right

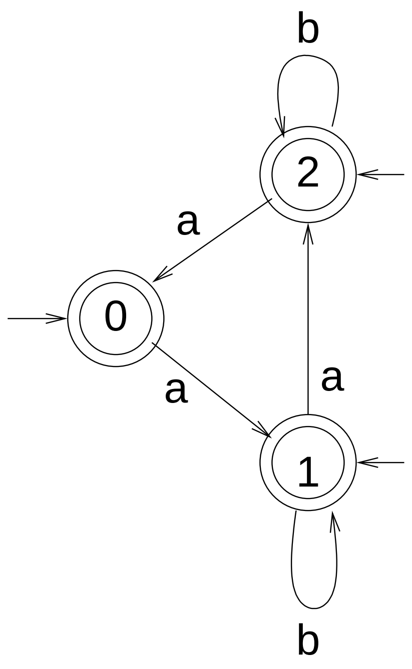


Figure 29: Nondeterministic Finite-State Machine

factors there are thus 20 left/right factors. The factor graph thus has a total of 20 nodes but, as usual, we choose not to depict the inadmissible nodes.

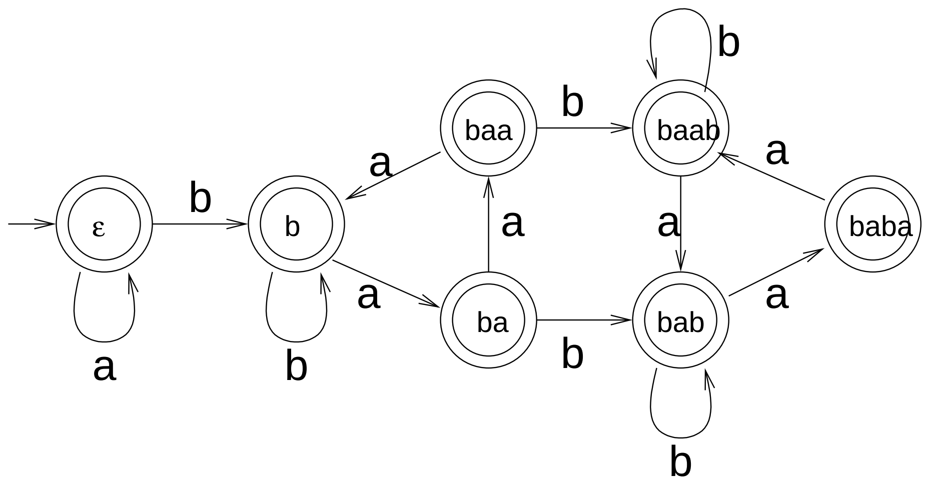


Figure 30: Machine

Left factor	Right Factor
$\{\varepsilon\}$	$\neg\emptyset$
$\{\varepsilon,b\}$	$\neg\{abab\}$
$\{\varepsilon,ba\}$	$\neg\{bab\}$
$\{\varepsilon,baa\}$	$\neg\{baab\}$
$\{\varepsilon,b,ba\}$	$\neg\{abab,bab\}$
$\{\varepsilon,b,ba\}$	$\neg\{abab,baab\}$
$\{\varepsilon,b,ba\}$	$\neg\{bab,baab\}$
$\{\varepsilon,b,ba,bab\}$	$\{\varepsilon,b,aab,baab\}$
$\{\varepsilon,b,baa,baab\}$	$\{\varepsilon,b,ab,bab\}$
$\{\varepsilon,ba,baa,baba\}$	$\{\varepsilon,ab,aab,abab\}$
$\neg\{bab,baba\}$	$\{\varepsilon,b,ab\}$
$\neg\{baab,baba\}$	$\{\varepsilon,b,aab\}$
$\neg\{bab,baab\}$	$\{\varepsilon,ab,aab\}$
$\neg\{baba\}$	$\{\varepsilon,b\}$
$\neg\{bab\}$	$\{\varepsilon,ba\}$
$\neg\{baab\}$	$\{\varepsilon,baa\}$
$\neg\emptyset$	$\{\varepsilon\}$

The syntactic monoid is shown in fig. 30. The nodes are labelled by a representative element of the corresponding c-class.

Each l-class is a union of c-classes. The table below shows the relationship. For example, the first entry summarises the property

$$E_l(\varepsilon) \quad = \quad E_c(\varepsilon) \cup E_c(a) \cup E_c(aa) \quad .$$

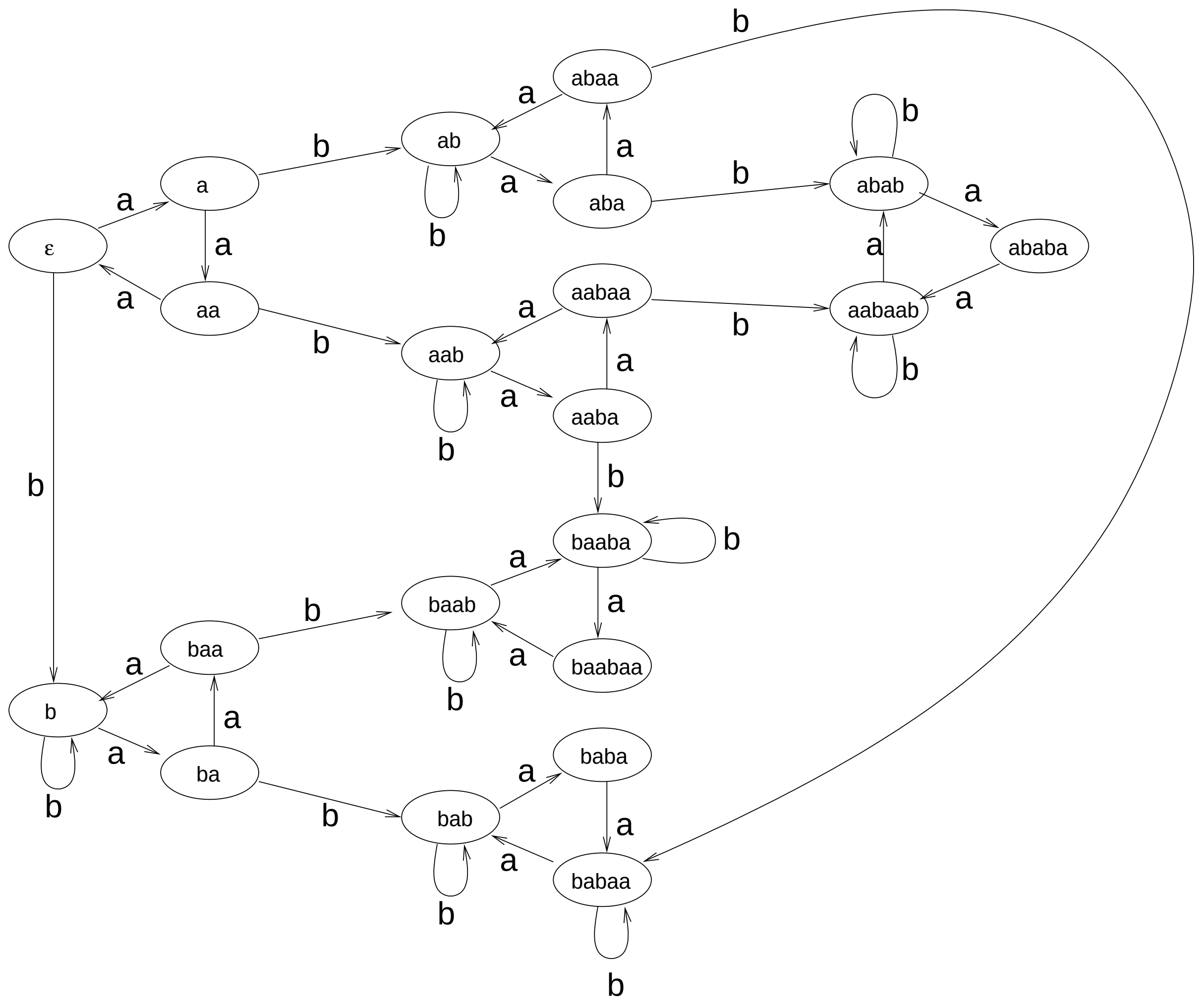


Figure 31: Syntactic Monoid

This fact is not exploited in our algorithm for constructing the factor graph but is useful for checking calculations — particularly, as in this case, the calculations are rather long and tedious.

l-class	c-classes
ε	ε,a,aa
b	b,ab,aab
ba	ba,aba,aaba
baa	baa,abaa,aabaa
bab	bab,abab,aabab
baab	baab,abaab,aabaab
baba	baba,ababa,aababa

The factor graph —including inadmissible nodes— of the event recognised by the machine in fig. 30 is shown in [LS08, fig. 11]. (Lombardy and Sakarovitch label every admissible node[15] as both a start and a final node. One inadmissible node is labelled as a final node only, and the other inadmissible node is labelled as a start node only. As explained in section 10.2, a unique start and a unique final state identify the factor graph as a (non-deterministic) recogniser of the event “$\mathcal{Z}_3$”: the start state is the state they name “012” and the final state is the state they name “0,1,2”.)

The factor graph (excluding inadmissible nodes) has 8 sections. Five of these have three nodes and three have one node. Fig. 32 depicts these sections: the graph with one node occurs three times, the rightmost of the graphs with three nodes occurs once, and the remaining two occur twice.

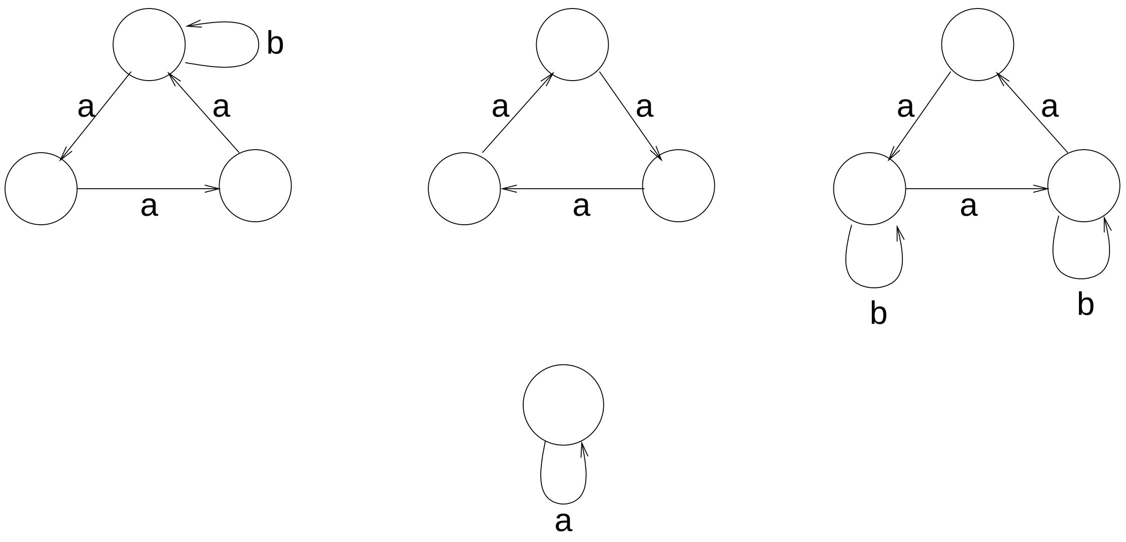


Figure 32: Sections of the Factor Graph

Every section of the factor graph is a factor graph of a factor. Taking the topmost node as both start and final node in each of the graphs in fig. 32, and expressing the factor in terms of c-classes, the factors are the events represented by the sets of c-classes $\{\varepsilon,b\}$, $\{\varepsilon\}$, $\{\varepsilon,abaa,aaba,ababa\}$ and $\{\varepsilon,a,aa\}$.

The reflexive-transitive closure (“star”) of each of these graphs is easily computed in the powerset algebra with underlying monoid the syntactic monoid shown in fig. 31. These are shown in figs. 33, 34, 35 and 36. (The graphs in fig. 32 have been taken in order from left to right and top to bottom.) One entry in fig. 35 has been highlighted in

[15]Actually, *almost every* admissible node To be consistent, the nodes named “02” and “12” in their diagram should be both start and final nodes. As a recogniser, this mistake has no effect: their labelling is a gross overkill. See section 10.1 for the justification of this remark.

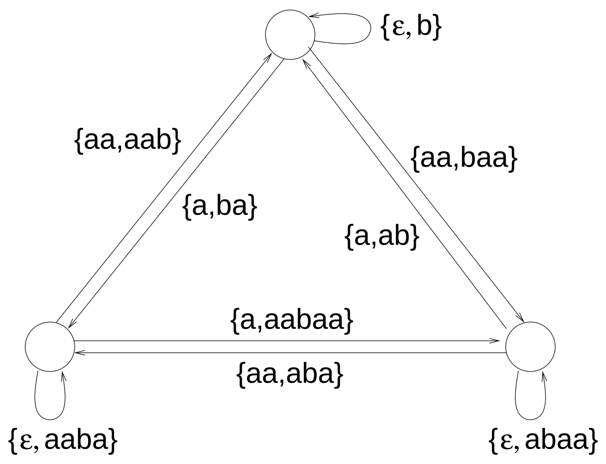


Figure 33: Star of Top-Left Section

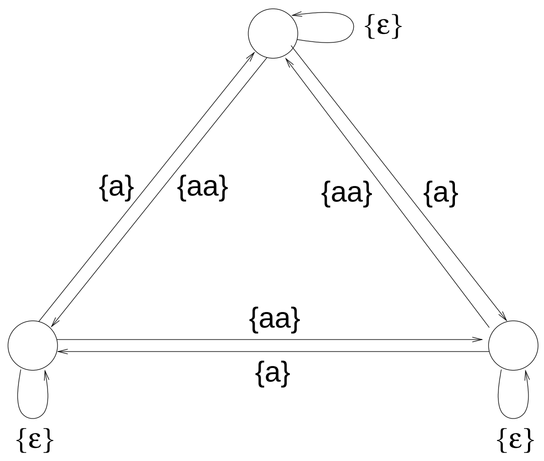


Figure 34: Star of Top-Middle Section

the usual way by identifying particular start and final nodes.

To be completed

□

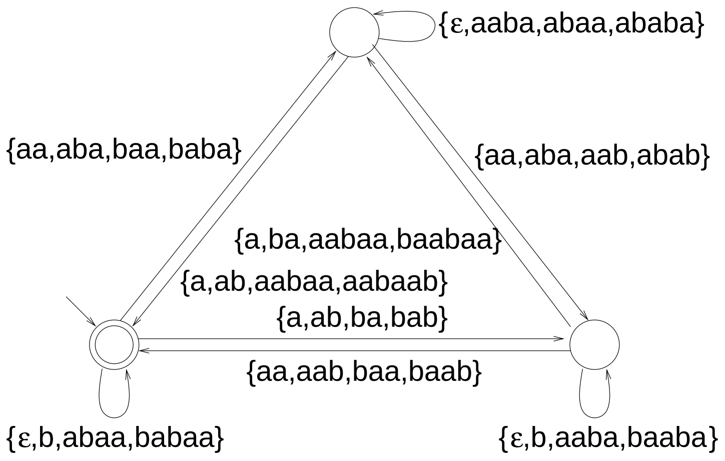


Figure 35: Star of Top-Right Section

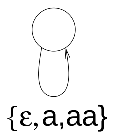


Figure 36: Star of Bottom-Middle Section