UNIVERSITA' DEGLI STUDI DELL'INSUBRIA

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dottorato di Ricerca in Matematica del Calcolo:
Modelli, Strutture, Algoritmi ed Applicazioni

XXIII Ciclo

Categorical algebras for the compositional construction of probabilistic and distributed systems

PhD Candidate:

Luisa de Francesco Albasini

PhD Course Coordinator:

Prof. S. Serra Capizzano

Supervisors:

Prof. N. Sabadini
Prof. R.F.C. Walters

A.A. 2009-2010

Categorical algebras for the compositional construction of probabilistic and distributed systems

by Luisa de Francesco Albasini

Submitted to the Department of Physics and Mathematics, University of Insubria, Como (Italy), on March 16, 2011, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis supervisors: Prof. N. Sabadini, Prof. R.F.C. Walters.

Acknowledgements.

During the work on this thesis, I have been lucky to work with Prof. Nicoletta Sabadini and Prof. Robert F.C. Walters of the Department of Computer Science and Communication, University of Insubria, Varese, who introduced me to the enjoyment of scientific research and made this thesis possible. Their continuous support, their advice and their guidance have made an important difference in my work. I really enjoyed being part of this research group as well as interacting with them at both the professional and the personal level. I am very grateful to them.

I would also like to thank the Department of Physics and Mathematics, University of Insubria, Como, for the opportunity to do this research.

Finally, I would like to thank Davide Maglia for helpful and interesting conversations that clarified my thinking, my husband Giuseppe and everyone else who ever gave me help and support during this research.

Abstract.

Structured transition systems have been widely used in the formal specification of computing systems, including concurrent and probabilistic systems. In this thesis we extend the span of graphs algebra introduced in [40] to describe concurrent systems in a compositional way, in order to model probabilistic distributed systems.

The span algebra of [40] may be regarded as an extension of various automata models of computation based on distributed automata ([2, 70]). With the introduction of both parallel and sequential operations ([41]), this extension allows the compositional description of concurrent, distributed and mobile systems. An important aspect of span graph model is that there is also a geometric characterization associated with the algebra along the lines of Penrose's algebra of tensors [54].

The algebra of transition systems we describe is a well-supported compact closed category - a symmetric monoidal category for which every object has a separable algebra structure (satisfying Frobenius equation [12]) - whose arrows are automata in which the actions have probabilities. The operations of the category permit the compositional description of probabilistic processes and in particular of classical finite Markov chains. The systems we can model with our algebra are distributed, hierarchical and with an evolving geometry.

We also extend the span algebra of [40] in order to permit the inclusion of quantum components. Again, this algebra is a well-supported compact closed category. This extension permits a compositional description of quantum protocols, in which quantum components interact with classical finite state components. In our view, the inclusion of explicit components of finite state classical control adds conceptual clarity and precision to quantum protocols.

Keywords: Classical automaton, Weighted automaton, Monoidal category, Span, Probabilistic automaton, Markov chain, Compositionality, Quantum automaton, Compact closed, Frobenius equations, Teleportation.

Contents

Table of symbols

We list the symbols used in this thesis in the order they appear in the presentation. For each symbol we indicate a short description and the reference to the page where it is first defined.

$(\mathsf{Q}_{a,b})_{(a\in A,b\in B)}$	transition matrix of the weighted automaton $\mathbf{Q}$	16
γ_0, γ_1	sequential interface functions	16
$[\mathsf{Q}_{a,b}]_{q,q'}$	element q, q' of $(\mathsf{Q}_{a,b})$	16
$\mathcal{Q}$	total matrix of the weighted automaton $\mathbf{Q}$	16
$\mathbf{Q}^{X}_{Y;A,B}$	automaton with par. interfaces A, B, seq. interfaces X, Y	16
$\mathcal{W}$	set of isomorphism classes of weighted automata	16
ε	null signal	17
$\mathbf{N}(\mathbf{Q})$	normalization of the weighted automaton $\mathbf{Q}$	18
$\mathbf{Q}^k$	automaton of k step paths in $\mathbf{Q}$	18
$\mathbf{Reach}(\mathbf{Q}$,$q_0)$	reachable part of $\mathbf{Q}$ with initial state q_0	23
$+$	sum	23
$+_Y$ (or $\circ$)	sequential composition on Y	24
$\mathbf{Seq}(\rho)$	sequential constant	25
$\mathbf{1}_A$	identity on A	25
∇	codiagonal	25
∇^o	opposite of codiagonal	25
$\mathbf{twist}_{A,B}$	twist on A,B	26
$\mathbf{i}_A$	sequential wire corresponding to $\emptyset \subseteq A$	26
$\mathbf{i}^o_A$	opposite of $\mathbf{i}_A$	26
δ	bijection $X \times Y + X \times Z \to X \times (Y + Z)$	26
$\times$	product (parallel without communication)	26
$\otimes$	tensor product	26
$\times_B$ (or $\vert\vert$)	series composition on B (communicating parallel)	27
$\cdot$	series composition of Markov automata	29
$\mathbf{Par}(\rho)$	parallel constant	30
$\boldsymbol{\Delta}$	diagonal	30
$\boldsymbol{\Delta}^o$	opposite of diagonal	30

$\mathbf{p}_A$	parallel wire corresponding to $A \to \{*\}$	31
$\mathbf{p}_A^o$	opposite of $\mathbf{p}_A$	31
$+_{loc}$	local sum	31
$+_{Y,loc}$ (or $\bullet$)	local sequential composition on Y	31
Sfb	sequential feedback	31
Pfb	parallel feedback	32
$\mathbf{DP}_n$	n Dining Philosophers system	38
∂_0, ∂_1	left and right arrows of a span respectively	44
$Span(Graph)$	category of graphs and spans of graphs	44
$Cospan(Graph)$	dual category of $Span(Graph)$	44
G_Y^X	system with graph G, sequential interfaces X, Y	45
$G_{A,B}$	system with graph G, parallel interfaces A, B	45
$O^{\rightarrow}, O^{\leftarrow}, O^{\downarrow}, O^{\uparrow}$	adjacent objects to O	46
η_X	system corresponding to $0 \to X \xleftarrow{\nabla} X + X$	61
$\Pi(S)$	parallel expressions of elements in S	64
$\Sigma(S)$	sequential expressions of elements in S	64
E	set of automata with 2 states and 1 transition	64
$P(X \mid Y)$	conditional probability of X given Y	74
$P_q^{\mathbf{Q}}(b)$	probability in state q of a signal b	75
$\mathbf{O}_a$	observer of the signal a	75
$\mathbf{Q} \mid a$	automaton $\mathbf{Q}$ given a	75
$P_q^{\mathbf{Q}}(b \mid a)$	probability of b given a in q	75
$\varphi_{a,b(a \in A, b \in B)}$	family of linear transformations	89
$< \mid >$	hermitian inner product	89
$\{e_x\}_{(x \in X)}$ (or $\{x\}$)	standard basis of $\mathbb{C}^X$	89
$Bell_i$	i-th Bell state, $i = 1, \ldots, 4$	96
$\mathbf{TP}$	teleportation protocol	96

Introduction

The aim of this thesis is to present a model which permits a compositional description of probabilistic and distributed systems, also ones with a hierarchical and mobile structure. Our systems are in the first place doubly labelled graphs provided with an algebraic structure in order to allow the compositionality.

An important aspect of our model is that there is an intuitive graphical representation of probabilistic systems closely related to their algebraic characterization. One of the first attempts to have a strict relation between graphical and algebraic/categorical descriptions was the work [54] of R. Penrose about the tensor calculus. This subject was later much developed in relation to the geometry of manifolds, and quantum field theory ([36, 65]).

The algebra we propose to describe probabilistic systems is an extension of the categorical algebra of spans of graphs introduced in [40] in order to permit operations analogous to the parallel, series and feedback of classical circuits.

The model of weighted and Markov automata described in this thesis was introduced in [21], adding probabilities on the actions into the algebra of [40] and then it was extended in [22] to sequential operations in order to describe also hierarchical and mobile systems. Various parts of this model were presented at the international conferences *CT2009* [21], *CT2010* [23] and the Italian conference *ICTCS'09* [22] .

We have mentioned that the algebra of [40] is the monoidal category of spans of graphs. Spans were introduced in [5] and are a generalization of relations. For example, in the category of sets a span R from A to B may be thought of as a $A \times B$ indexed family of sets $R_{a,b}$ $(a \in A,\ b \in B)$, or alternatively as a matrix of sets, whereas relations are matrices of boolean values. Span composition of R from A to B and S from B to C is given by pullback, or by the formula $\Sigma_{b \in B} R_{a,b} \times S_{b,c}$, $(a \in A,\ c \in C)$ clearly reminiscent of the formula for composition of relations. The monoidal categories of spans, and of relations, have been intensively studied beginning in [12] where the Frobenius equation was discovered. Recent papers are [10], and [69]. The paper [40] in considering spans of graphs, in particular between one vertex graphs, introduced a new aspect. Such spans may be considered to be doubly indexed families of graphs; but a graph may be thought of as a *square matrix of sets*.

In this thesis we describe a natural extension of the algebra of spans of graphs, namely doubly indexed families $\mathsf{Q}_{a,b}$ $(a \in A, b \in B)$ of *non negative real square matrices*, the main operation being the tensor product $\otimes$ of matrices. In chapter 5 we present a similar extension - *Quantum automata* - to doubly indexed families $\varphi_{a,b}$ $(a \in A, b \in B)$ of *operators* on finite dimensional vector spaces in order to permit a compositional description of quantum protocols in terms of communicating quantum and classical automata. This further extension of span graph algebra was first presented to *Quantum Physics and Logic workshop*, Oxford [19].

Markov automata are in the first place graphs - the vertices represent states and the arcs transitions. Each transition has a non negative real number associated, which we may think of as the probability that the transition occurs. We require that the sum of probabilities out of a given state is 1.

In [19] weighted automata (where the weighting of a transition is a non-negative real number without others assumptions) played a subsidiary role. However for sequential composition weighted automata are more fundamental, since in identifying states of two different automata it is the relative weight given to decisions which is important rather than the probabilities. Technically this appears in the fact that normalization is not compositional with respect to sequential operations, whereas for parallel operations it is. For a summary of recent work on the various kinds of weighted automata see [24], and in particular [52].

There are many well-known examples of probability problems in which the apparent clarity of the system configuration doesn't correspond to an analogous clarity of the solution. Sometimes such problems generate long discussions between mathematicians and the possible answers are very far from each other. In particular the meaning of conditional probability, which is of huge importance in the classical model of probability theory, seems sometimes to be not clear in practical examples and gives us counter-intuitive answers to the problem. In this thesis we show how many of the paradoxes of conditional probability arise from the non-compositionality of normalization with respect to sequential operations.

Another aim of this thesis is to illustrate how hierarchical and mobile systems may be modelled in this algebra, using the combined sequential and parallel operations. Given a set of automata S, let us denote the set of automata given as expressions in terms of parallel operations in the automata S as $\Pi(S)$ and given as expressions in terms of sequential operations as

$\Sigma(S)$. Let E be the set of elementary automata with two states and only one transition. Then any automaton has a representation in $\Sigma(E)$. The Dining Philosopher problem of [19] is described as an element of $\Pi\Sigma(E)$, that is of communicating sequential systems ([18]). An element of $\Sigma\Pi\Sigma(E)$ is one in which the net of automata may evolve; $\Pi\Sigma\Pi\Sigma(E)$ expressions describe nets of evolving nets of automata, and so on. In chapter 3 we describe some examples.

The power of the sequential and parallel operations is that they may be alternated, as the alternating quantifiers in logic, or the alternation in alternating Turing machines. There is a close relation between this aspect of the algebra and the statecharts of [31].

0.1 Parallel and sequential interfaces

The automata described in this thesis are equipped with *parallel* and *sequential interfaces* that are used to compose automata. Parallel interfaces are distinguished into *left* interfaces and *right* interfaces. A parallel interface is represented mathematically by a labelling of the transitions of the automaton on an alphabet. The idea is that the transitions that occur have effects, which we may think of as *signals*, on the two parallel interfaces of the automaton, which signals are represented by letters in the alphabets. It is important not to think of the letters as inputs or outputs, but rather signals induced by transitions of the automaton on the interfaces. Parallel interfaces allow us to describe parallel operations (parallel composition denoted by $\times$, and parallel composition with communication denoted by $||$). An important aspect is the use of conditional probability since, for example, composing introduces restrictions on possible transitions and hence changes probabilities.

Very often the state space of an automaton decomposes naturally into a disjoint sum of *cases*. Sequential interfaces represent cases relevant to activating (creating) or disactivating (destroying) automata and channels between automata, what we call *in-conditions* and *out-conditions* respectively. A condition is a state in which the configuration of the automaton may change in a particular way. Formally, an in-condition of an automaton is represented by a function in the set of states (usually not the inclusion of a subset). Similarly, an out-condition consists of a function into the state space of the automaton.

It is important to note that only in purely sequential programming it is reasonable to think of the conditions just as initial and terminal states. In fact when we have several active processes and there is, for example, a state in which only one process may die while the others are in general activity, the global state of the system in which a change of configuration occurs is a terminal state in only one component.

Sequential interfaces allow us to define sequential operations (the sum denoted $+$, and the sequential composite denoted $+_Y$, where Y is the common sequential interface). This kind of compositions permits the description of hierarchical and mobile systems.

0.2 Comparison with other models

0.2.1 Weighted and Markov automata

There is a huge literature on probabilistic and weighted automata, transducers, and process calculi (see for example, [56, 61, 33, 9, 63, 49, 24]). However our model of Markov and weighted automata is distinguished from the others in the following ways:

(i) In many other probabilistic automata models ([56],[3]) the sum of probabilities of actions out of a given state *with a given label* is 1. This means that the probabilities are conditional on the existence of the label (or the "pushing of a button"). Our model of Markov automata is instead *generative* in the sense of [29] in that the sum of probabilities of actions out of a given state *for all labels* is 1. We explain our intuition in the next point. The intuition of [29] is perhaps slightly different since they also speak of "pushing buttons".

(ii) The actual origins of our model are the work of Schützenberger and Eilenberg [25] on weighted transducers - our operations are variations on those introduced there - modified by further category theoretic experience, beginning, for example, with [12, 40]. We view the automata and the operations on them rather differently from [25]. Instead of modelling devices which translate input to output, the idea is that we model devices with number of parallel interfaces, and when a transition occurs in the device this induces a transition on each of the parallel interfaces (the interfaces are part of the device). In order to have binary operations of composition the interfaces are divided into left and right interfaces. The notions of initial and final states are also generalized in our notion of weighted automaton to become instead functions into the state space. These sequential interfaces are not to be thought of as initial and final states, but hooks into the state space at which a behaviour may enter or leave the device. The application of our weighted automata is to concurrent hierarchical and distributed systems rather than language recognition or processing. In [20] we have shown how data types and also state on the parallel interfaces (shared variables) may be added to our model.

(iii) For many compositional models the communication is based on process algebras like CCS [51] and CSP [34], with interleaving semantics and underlying broadcast topology. Instead, our algebra models truly concurrent systems with explicit network topologies. We have shown in [17] that communication such as that of CCS and of CSP may be modelled in our algebra, but that they correspond to very particular network topologies. One of the key aspects in our algebra are the connectors: parallel and sequential ”wires”, which give the hierarchical network topology to expressions in the algebra.

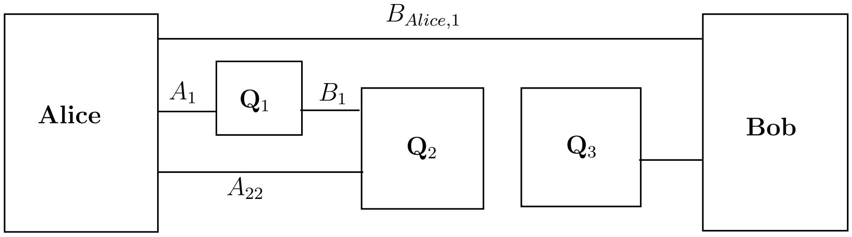
(iv) Our automata with respect to the parallel operations form the arrows of a compact closed symmetric monoidal category, with other well-known categorical properties. (This would have been also the case for the sequential operations if we had not chosen to make the technical simplification of not considering state on the parallel interfaces.) The operations are in fact based on the operations available in categories of spans and cospans [20]. Similar algebras occur in developments in many other areas of computer science, mathematics and physics. With respect to parallel operations our algebra is *a fortiori* traced monoidal [37], and hence has close relations with the work of Stefanescu [64] on network algebras, and the theory of fix point operators studied by many authors including Bloom and Esik [8] - connections are described in [32]. The diagrams we introduce are related to those of topological field theory described, for example, in [45], and also those of the diagrammatic approach to quantum mechanics described for example in [15, 16].

(v) Other compositional probabilistic automata models admit non-determinism as well as probabilistic transitions, in order to model concurrent behaviour. The most natural way to model asynchrony in our algebra is instead through null (ε) labels on transitions; variation in timing would be represented by different weights of null transitions in different states. However we show that it is also possible to model asynchrony through non-determinism in our algebra. In particular we show how (finite examples of) the probabilistic automata of Segala and Lynch in [61, 49] fit naturally in our context and we give a simplified description of their behaviour in terms of the operations of our algebra (see section 3.4).

(vi) In our model, the communicating parallel composition involves conditional probability (see chapter 4). The reason is that communication restricts the possibilities of the participants of the composition; in an extreme case communication may produce deadlock in which all processes must idle. This restriction of movement in the composite means that the probabilities of acting must be adjusted to be probabilities *given the synchronization*. This crucial aspect of our model is not shared by many other models. Indeed, Segala in section 3.4 of [61] argues against such a parallel composition.

0.2.2 Quantum automata

A mixed algebra of quantum and classical phenomena has already been introduced by Coecke and Pavlovic in [15], with further work in [14], following the categorical twist on quantum logic introduced in [1]. The idea of those works is to describe data flow in quantum protocols, involving also classical measurements, as expressions in a symmetric monoidal category with extra structure. Such a formulation yields geometric pictures (following [54],[44]) of the flow in protocols, as well as pictorial equations which may be used to prove correctness. Another mixed algebra of quantum and classical phenomena was introduced by Selinger in [62], in the context of a functional programming language, which incorporates flow of data and flow of classical control. Neither of these approaches is an algebra of entities with states and actions, and their communication, and neither can explain the following picture introduced in chapter 5 to describe the teleportation protocol:



The importance of the *distributive law* of tensor product over direct sum in making classical choices is evident, as first observed for data by Selinger [62]. The situation is entirely analogous to classical Turing machines where an infinite state tape interacts with, and is controlled by, a finite state automaton (see the (non-compositional) description of Turing machines in [68]; and also [60],[67] for relations with the Blum-Shub-Smale theory of computable functions).

Our automata are not to be confused with the quantum automata of [46] or [53], and hence we use the name $\mathbb{C}$-*automaton* for the general notion and *quantum or classical* $\mathbb{C}$-*automaton* for those which represent respectively quantum or classical components.

0.3 Organization of the thesis

Chapter 1 is devoted to describing probabilistic processes and in particular Markov chains, and for this reason we call these automata *Markov automata.* We introduce the basic notions of the model giving first the definition of weighted automata with parallel and sequential interfaces, where the weighting of a transition is a non-negative real number, and then the less general definition of Markov automata. In particular, we define a Markov automaton with a given set A of *signals on the left interface*, and set B of *signals on the right interface* to consist of a Markov matrix $\mathcal{Q}$ (whose rows and columns are the states) which is the sum

$$\mathcal{Q} = \mathsf{Q}_{a_1,b_1} + \mathsf{Q}_{a_2,b_1} + \cdots + \mathsf{Q}_{a_m,b_n}$$

of non-negative matrices Q_{a_i,b_j} whose elements are the probabilities of transitions between states for which the signals a_i and b_j occur on the interfaces. In addition, each alphabet is required to contain a special symbol ε (the null signal) and the matrix $\mathsf{Q}_{\varepsilon_A,\varepsilon_B}$ is required to have row sums strictly positive. Then, we show how to compose such automata, calculating the probabilities in composed systems and we introduce the graphical representation associated to weighted and Markov automata.

As an illustration of the algebra we show how to specify a system of n Dining Philosophers (a system with 12^n states) and to calculate the probability of reaching deadlock in k steps, and we show that this probability tends to 1 as k tends to ∞, using the methods of Perron-Frobenius theory[1].

Chapter 2 presents a detailed description of the notion of system with parallel and sequential interfaces as introduced in [20]. The model we present is an abstract guide in describing concrete situations.

From an algebraic point of view, these systems are cospan of spans of graphs. In fact, a n-dimensional square matrix may be thought of as a graph with n vertices. An $A \times B$ family of square matrices of the same size may be thought of as a graphs with double-labelled edges. Hence a

[1] For numerical calculations of the probability of reaching deadlock in the Dining Philosopher system see appendix A.

weighted automaton consists of five graphs and four graphs morphism as in the following figure:

$$\begin{tikzcd}
& X \arrow[d, "\gamma_0"] & \\
A & G \arrow[l, "\delta_0"'] \arrow[r, "\delta_1"] & B \\
& Y \arrow[u, "\gamma_1"'] &
\end{tikzcd}$$

where X, Y are graphs without edges and A, B graphs with only one vertex. G is the graph of states and transitions of the system, X, Y are the graphs of the sequential interfaces and A, B graphs of parallel interfaces. But also the sequential interfaces are systems and so need parallel interfaces. On the other hand, parallel interfaces are systems and need sequential interfaces. Hence a *system with sequential and parallel interfaces* is a *cospan of spans of graphs* and consists of a commutative diagram

$$\begin{tikzcd}
G_0 \arrow[d, "\gamma_1"] & X \arrow[l, "\delta_0"'] \arrow[r, "\delta_1"] \arrow[d, "\gamma_0"] & G_1 \arrow[d, "\gamma0"] \\
A & G \arrow[l, "\delta_0"'] \arrow[r, "\delta_1"] & B \\
G_2 \arrow[u, "\gamma_1"] & Y \arrow[l, "\delta_0"'] \arrow[r, "\delta_1"] \arrow[u, "\gamma_1"'] & G_3 \arrow[u, "\gamma_1"]
\end{tikzcd}$$

of graphs and graphs morphism. A system may be regarded also a span of cospans of graphs.

In defining sequential and parallel operations on systems we will note a strong analogy with the weighted and Markov model. An important aspect of the system model is that the two kinds of operations with the necessary constants give to the category of systems two structures of a well-supported compact closed category.

In **chapter 3** we explain with examples how to describe hierarchical and mobile probabilistic systems in our weighted and Markov automata model. We show that each automaton is an expression of sequential operations Σ and that each hierarchical system is obtained alternating sequential operations Σ with parallel operations Π.

We describe then the system Sofia's birthday party as an expression in $\Pi\Sigma\Sigma(E)$ but illustrates also the form of systems of type $\Pi\Sigma\Pi\Sigma(E)$. Then we introduce a Fork bomb which is an expression of $\Sigma(\Pi\Sigma)^n$. In the last section of the chapter we describe in our model the probabilistic automata of Segala and Lynch.

In **chapter 4** we introduce the notion of conditional probability for weighted and Markov automata. We prove results analogous to *Bayes' theorem* and *Total probability theorem* into the algebra of weighted and Markov automata. Describing famous examples of conditional probability problems we show the benefits of this approach.

In **chapter 5** we illustrate an extension of the algebra of spans of graphs to doubly indexed families $\varphi_{a,b}$ $(a \in A, b \in B)$ of *operators* on finite dimensional vector spaces in order to describe quantum protocols. We introduce the notion of quantum $\mathbb{C}$-automaton and classical $\mathbb{C}$-automaton. Then we describe the teleportation protocol of [6] in this algebra and we prove its correctness.

Chapter 1

Weighted and Markov automata

In this chapter we introduce the notion of Markov automaton, together with parallel and sequential operations which permit the compositional description of Markov processes ([21],[22]). For technical reasons we give first the definition of weighted automaton which is more general than the definition of Markov automaton. It will be more clear in chapter 4 why it is important to distinguish between Markov automaton and the more general notion of weighted automaton.

We add also sequential operations to the algebra (and the necessary structure to support them) following analogous developments in [42, 20]. As we will explain in more detail in chapter 3, the extra operations permit the description of hierarchical and mobile systems.

1.1 Weighted automata with parallel and sequential interfaces

In this section we define weighted and Markov automata *with sequential and parallel interfaces*, which however we shall call just weighted and Markov automata. The reader should be aware that the definitions of [20] differs in lacking sequential interfaces. We also do not require here for weighted automata the special symbols ε and the condition that the rows of the total matrix are strictly positive: we reserve those conditions for what we now call *positive weighted automata*.

Notice that in order to conserve symbols in the following definitions we shall use the same symbol for the automaton, its state space and its family of matrices of transitions, distinguishing the separate parts only by the font.

Definition 1.1.1. *Consider two finite alphabets A and B, and two finite sets X and Y. A* weighted automaton $\mathbf{Q}$ with left parallel interface A, right parallel interface B, top sequential interface X, *and* bottom sequential interface Y, *consists of a finite set Q of states, and an $A \times B$ indexed family $\mathsf{Q} = (\mathsf{Q}_{a,b})_{(a \in A, b \in B)}$ of $Q \times Q$ matrices with non-negative real coefficients, and two functions, $\gamma_{0,\mathbf{Q}} : X \to Q$, and $\gamma_{1,\mathbf{Q}} : Y \to Q$. We denote the elements of the matrix $\mathsf{Q}_{a,b}$ by $[\mathsf{Q}_{a,b}]_{q,q'}$ $(q, q' \in Q)$.*

We call the matrix

$$\mathcal{Q} = \sum_{a \in A, b \in B} \mathsf{Q}_{a,b}$$

the total matrix *of the automaton.*

In other words, a weighted automaton $\mathbf{Q}$ with left parallel interface A, right parallel interface B, top sequential interface X, and bottom sequential interface Y, consists of a finite graph with set of vertices Q, with arcs labelled on $A \times B$ and with two functions $\gamma_0 : X \to Q$, and $\gamma_1 : Y \to Q$. Each arc of the graph has a non negative real number attached.

We will use a brief notation for the automata $\mathbf{Q}$ indicating its interfaces, namely $\mathbf{Q}^X_{Y;A,B}$. We shall often use the same symbols γ_0, γ_1 for the sequential interface functions of any automata, and we will sometimes refer to these functions as the sequential interfaces. Notice that the terms 'left', 'right', 'top' and 'bottom' for the interfaces have no particular semantic significance - they are chosen to be semantically neutral in order not to suggest input, output, initial or final.

We will often consider weighted automata up to isomorphisms.

Definition 1.1.2. *Given two weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Z_{W;C,D}$, an* isomorphism of weighted automata $\phi : \mathbf{Q}^X_{Y;A,B} \to \mathbf{R}^Z_{W;C,D}$ *consists of five bijective arrows*

$$\begin{aligned} &\phi_s : Q \to R \\ &\phi_l : A \to C, \quad \phi_r : B \to D \\ &\phi_0 : X \to Z, \quad \phi_1 : Y \to W \end{aligned}$$

such that:

(i) $[\mathsf{Q}_{a,b}]_{q,q'} = [\mathsf{R}_{\phi_l(a),\phi_r(b)}]_{\phi_s(q),\phi_s(q')}$ *for all* $a \in A, b \in B, q, q' \in Q$,

(ii) $\phi_s \circ \gamma_{i,\mathbf{Q}} = \gamma_{i,\mathbf{R}} \circ \phi_i, \quad i = 0, 1.$

We denote by $\mathcal{W}$ the set of isomorphism classes of weighted automata.

Definition 1.1.3. *A weighted automaton* $\mathbf{Q}$ *is* positive *if the parallel interfaces A and B contain special elements (null signals), the symbols ε_A and ε_B, and satisfies the property that the row sums of the matrix $\mathsf{Q}_{\varepsilon_A,\varepsilon_B}$ are strictly positive.*

We require for a positive weighted automaton that the total matrix has strictly positive row sums.

Definition 1.1.4. *A* Markov automaton $\mathbf{Q}$ with left interface A*,* right interface B*,* top sequential interface X*, and* bottom sequential interface Y*, written briefly $\mathbf{Q}^X_{Y;A,B}$, is a positive weighted automaton $\mathbf{Q}^X_{Y;A,B}$ satisfying the extra condition that the row sums of the total matrix $\mathcal{Q}$ are all* 1*. That is, for all q*

$$\sum_{q'} \sum_{a\in A, b\in B} [\mathsf{Q}_{a,b}]_{q,q'} = 1.$$

We call $[\mathsf{Q}_{a,b}]_{q,q'}$ the *probability of the transition from q to q' with left signal a and right signal b.*

The idea is that in a given state various transitions to other states are possible and occur with various probabilities, the sum of these probabilities being 1. The transitions that occur have effects, which we may think of as *signals*, on the two interfaces of the automaton, which signals are represented by letters in the alphabets. We repeat that it is fundamental *not* to regard the letters in A and B as inputs or outputs, but rather signals induced by transitions of the automaton on the interfaces. For examples see section 1.1.2.

The positivity assumption for a Markov automaton means that in each state there is a strictly positive probability of a silent move (idle transition which induces the null signal ε on the two parallel interfaces).

When both A and B are one element sets and $X = Y = \emptyset$ a Markov automaton is just a Markov matrix.

Definition 1.1.5. *Consider a weighted automaton $\mathbf{Q}^X_{Y;A,B}$. A* behaviour *of length k of $\mathbf{Q}^X_{Y;A,B}$ consists of two words of length k, one $u = a_1a_2\cdots a_k$ in A^* and the other $v = b_1b_2\cdots b_k$ in B^* and a sequence of non-negative row vectors*

$$x_0,\ x_1 = x_0\mathsf{Q}_{a_1,b_1},\ x_2 = x_1\mathsf{Q}_{a_2,b_2},\ \cdots,\ x_k = x_{k-1}\mathsf{Q}_{a_k,b_k}.$$

Notice that, even for Markov automata, x_i is *not* generally a distribution of states; for example often x_i is the row vector 0.

From a positive weighted automaton we can always obtain a Markov automaton with a *normalization* of the weights.

Definition 1.1.6. *The* normalization *of a positive weighted automaton* $\mathbf{Q}$*, denoted* $\mathbf{N}(\mathbf{Q})$*, is the Markov automaton with the same interfaces and states, but with*

$$\left[\mathsf{N}(\mathsf{Q})_{a,b}\right]_{q,q'} = \frac{[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'\in Q}[\mathcal{Q}]_{q,q'}} = \frac{[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'\in Q}\sum_{a\in A,b\in B}[\mathsf{Q}_{a,b}]_{q,q'}}.$$

To see that $\mathbf{N}(\mathbf{Q})$ is Markov, notice that the qth row sum of $\mathsf{N}(\mathsf{Q})$ is

$$\begin{aligned}\sum_{q'}\sum_{a,b}\left[\mathsf{N}(\mathsf{Q})_{a,b}\right]_{q,q'} &= \sum_{q'}\sum_{a,b}\frac{[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'}\sum_{a,b}[\mathsf{Q}_{a,b}]_{q,q'}}\\ &= \frac{\sum_{q'}\sum_{a,b}[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'}\sum_{a,b}[\mathsf{Q}_{a,b}]_{q,q'}} = 1.\end{aligned}$$

Lemma 1.1.7. *(i) If* $\mathbf{Q}$ *is a Markov automaton then* $\mathbf{N}(\mathbf{Q}) = \mathbf{Q}$*.*

(ii) If c_q *are positive real numbers and* $\mathbf{Q}$ *and* $\mathbf{R}$ *are weighted automata (with the same interfaces* A *and* B*, and the same state spaces* $Q = R$*) such that*

$$[\mathsf{Q}_{a,b}]_{q,q'} = c_q[\mathsf{R}_{a,b}]_{q,q'}$$

then $\mathbf{N}(\mathbf{Q}) = \mathbf{N}(\mathbf{R})$*.*

Proof. (ii) follows since

$$\sum_{q'}\sum_{a,b}[\mathsf{Q}_{a,b}]_{q,q'} = \sum_{q'}\sum_{a,b}c_q[\mathsf{R}_{a,b}]_{q,q'} = c_q\sum_{q'}\sum_{a,b}[\mathsf{R}_{a,b}]_{q,q'}$$

and hence

$$\frac{[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'}\sum_{a,b}[\mathsf{Q}_{a,b}]_{q,q'}} = \frac{c_q[\mathsf{R}_{a,b}]_{q,q'}}{c_q\sum_{q'}\sum_{a,b}[\mathsf{R}_{a,b}]_{q,q'}} = \frac{[\mathsf{R}_{a,b}]_{q,q'}}{\sum_{q'}\sum_{a,b}[\mathsf{R}_{a,b}]_{q,q'}}.$$

□

An important operation on weighted automata is the *power* construction.

1.1.1 The power construction

Definition 1.1.8. *If* $\mathbf{Q}$ *is a weighted automaton and* k *is a natural number, then the* automaton of k step paths *in* $\mathbf{Q}$*, which we denote as* $\mathbf{Q}^k$ *is defined as follows: the states of* $\mathbf{Q}^k$ *are those of* $\mathbf{Q}$*; the sequential interfaces are the same; the left and right interfaces are* A^k *and* B^k *respectively. If* $u = (a_1, a_2, \cdots, a_k) \in A^k$ *and* $v = (b_1, b_2, \cdots, b_k) \in B^k$ *then*

$$(\mathsf{Q}^k)_{u,v} = \mathsf{Q}_{a_1,b_1}\mathsf{Q}_{a_2,b_2}\cdots\mathsf{Q}_{a_k,b_k}.$$

The definition for positive weighted automata requires in addition that $\varepsilon_{A^k} = (\varepsilon_A, \cdots, \varepsilon_A), \varepsilon_{B^k} = (\varepsilon_B, \cdots, \varepsilon_B)$*.*

If $\mathbf{Q}$ is a weighted automaton and $u = (a_1, a_2, \cdots, a_k) \in A^k$, $v = (b_1, b_2, \cdots, b_k) \in B^k$, then $[(\mathsf{Q}^k)_{u,v}]_{q,q'}$ is the sum over all paths from q to q' with left signal sequence u and right signal sequence v of the weights of paths, where the weight of a path is the product of the weights of the steps.

Lemma 1.1.9. *If $\mathbf{Q}$ is a weighted automaton then the total matrix of $\mathbf{Q}^k$ is the matrix power $\mathcal{Q}^k$. Hence if $\mathbf{Q}$ is Markov then so is $\mathbf{Q}^k$.*

Proof. The q, q' entry of the total matrix of $\mathbf{Q}^k$ is

$$\begin{aligned}
&\sum_{u \in A^k, v \in B^k} \left[(\mathsf{Q}^k)_{u,v}\right]_{q,q'} \\
&= \sum_{u \in A^k, v \in B^k} \left[\mathsf{Q}_{a_1,b_1} \mathsf{Q}_{a_2,b_2} \cdots \mathsf{Q}_{a_k,b_k}\right]_{q,q'} \\
&= \sum_{u \in A^k, v \in B^k} \sum_{q_1, \cdots, q_{k-1}} \left[\mathsf{Q}_{a_1,b_1}\right]_{q,q_1} \left[\mathsf{Q}_{a_2,b_2}\right]_{q_1,q_2} \cdots \left[\mathsf{Q}_{a_k,b_k}\right]_{q_{k-1},q'} \\
&= \sum_{q_1, \cdots, q_{k-1}} \sum_{a_1, \cdots, a_k} \sum_{b_1, \cdots, b_k} \left[\mathsf{Q}_{a_1,b_1}\right]_{q,q_1} \left[\mathsf{Q}_{a_2,b_2}\right]_{q_1,q_2} \cdots \left[\mathsf{Q}_{a_k,b_k}\right]_{q_{k-1},q'} \\
&= \sum_{q_1, \cdots, q_{k-1}} \sum_{a_1, b_1} \left[\mathsf{Q}_{a_1,b_1}\right]_{q,q_1} \sum_{a_2, b_2} \left[\mathsf{Q}_{a_2,b_2}\right]_{q_1,q_2} \cdots \sum_{a_k, b_k} \left[\mathsf{Q}_{a_k,b_k}\right]_{q_{k-1},q'} \\
&= \left[\mathcal{Q}\mathcal{Q} \cdots \mathcal{Q}\right]_{q,q'}.
\end{aligned}$$

□

Definition 1.1.10. *If $\mathbf{Q}$ is a Markov automaton then we call $\mathbf{Q}^k$ the* automaton of k step paths *in* $\mathbf{Q}$. *We define the probability in $\mathbf{Q}$ of passing from state q to q' in exactly k steps with left signal u and right signal v to be $[(\mathsf{Q}^k)_{u,v}]_{q,q'}$.*

It is important to understand the precise meaning of this definition. The probability of passing from state q to q' in precisely k steps, so defined, is the weighted proportion of all paths of length k beginning at q and ending at q' amongst all paths of precisely length k beginning at q.

1.1.2 Graphical representation of weighted automata

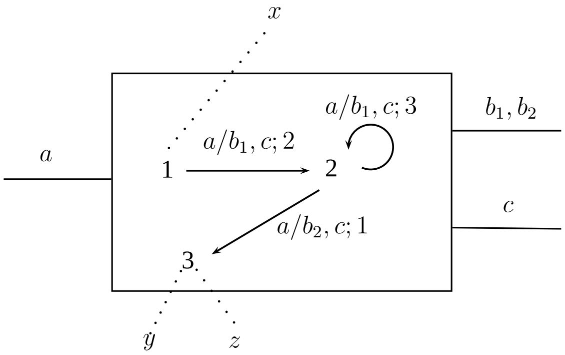
Although the definitions above are mathematically straightforward, in practice a graphical notation is more intuitive. We may compress the description of an automaton with parallel interfaces A and B, which requires $A \times B$ matrices, into a single labelled graph, like the ones introduced in [40]. Further, expressions of automata in this algebra may be drawn as "circuit diagrams" also as in [40]. We indicate both of these matters by describing some examples.

An example

Consider the automaton with parallel interfaces $\{a\}$, $\{b_1, b_2\} \times \{c\}$ and sequential interfaces $\{x\}$, $\{y, z\}$; with states $\{1, 2, 3\}$, sequential interface functions $x \mapsto 1$ and $y, z \mapsto 3$, and transition matrices

$$\mathsf{Q}_{a,(b_1,c)} = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ \mathsf{Q}_{a,(b_2,c)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

This information may be put in the diagram:



The following two examples have both sequential interfaces $\emptyset$ and hence we will omit the sequential information.

A philosopher

Consider the alphabet $A = \{t, r, \varepsilon\}$. A philosopher is an automaton **Phil** with left interface A and right interface A, state space $\{1, 2, 3, 4\}$, both sequential interfaces $\emptyset \subseteq \{1, 2, 3, 4\}$, and transition matrices

$$\mathsf{Phil}_{\varepsilon,\varepsilon} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix},$$

$$\mathsf{Phil}_{t,\varepsilon} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ \mathsf{Phil}_{\varepsilon,t} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathsf{Phil}_{r,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ \mathsf{Phil}_{\varepsilon,r} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \end{bmatrix}.$$

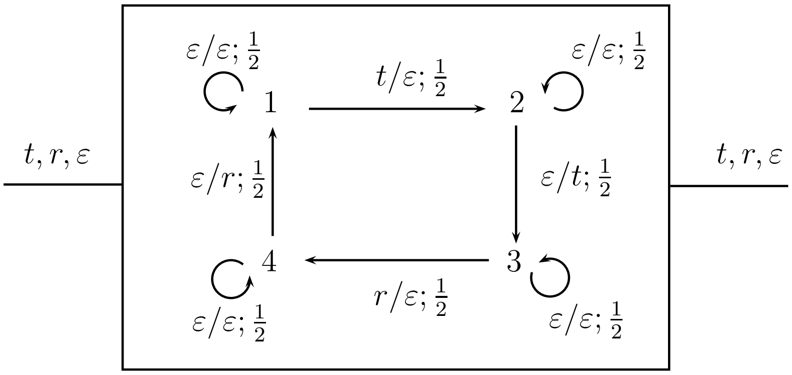
The other four transition matrices are zero matrices.

Notice that the total matrix of **Phil** is

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

which is clearly stochastic, so **Phil** is a Markov automaton.

The intention behind these matrices is as follows: in all states the philosopher does a transition labelled ε/ε (*idle transition*) with probability $\frac{1}{2}$; in state 1 he does a transition to state 2 with probability $\frac{1}{2}$ labelled t/ε (*take the left fork*); in state 2 he does a transition to state 3 with probability $\frac{1}{2}$ labelled ε/t (*take the right fork*); in state 3 he does a transition to state 4 with probability $\frac{1}{2}$ labelled r/ε (*release the left fork*); and in state 4 he does a transition to state 1 with probability $\frac{1}{2}$ labelled ε/r (*release the right fork*). All this information may be put in the following diagram.



A fork

Consider again the alphabet $A = \{t, r, \varepsilon\}$. A fork is an automaton **Fork** with left interface A and right interface A, state space $\{1, 2, 3\}$, both sequential interfaces $\emptyset \subseteq \{1,2,3\}$, and transition matrices

$$\mathsf{Fork}_{\varepsilon,\varepsilon} = \begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix},$$

$$\mathsf{Fork}_{t,\varepsilon} = \begin{bmatrix} 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ \mathsf{Fork}_{\varepsilon,t} = \begin{bmatrix} 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathsf{Fork}_{r,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ \mathsf{Fork}_{\varepsilon,r} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}.$$

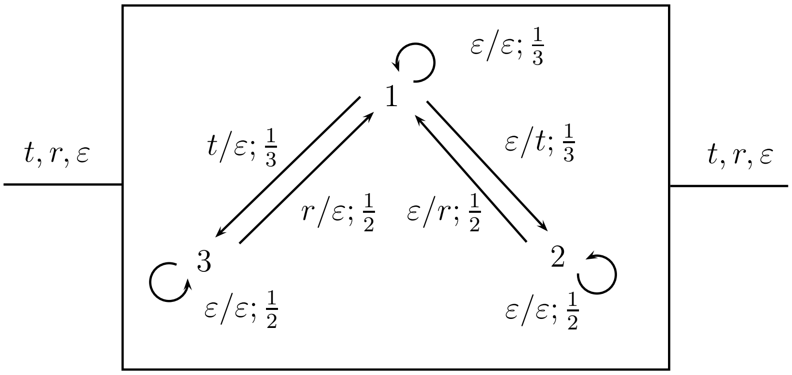
The other four transition matrices are zero.

Fork is a Markov automaton since its total matrix is

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$

The intention behind these matrices is as follows: in all states the fork does a transition labelled ε/ε (*idle transition*) with positive probability (either $\frac{1}{3}$ or $\frac{1}{2}$); in state 1 it does a transition to state 2 with probability $\frac{1}{3}$ labelled t/ε (*taken to the left*); in state 1 it does a transition to state 3 with probability $\frac{1}{3}$ labelled ε/t (*taken to the right*); in state 2 it does a transition to state 1 with probability $\frac{1}{2}$ labelled r/ε (*released to the left*); in state 3 it does a transition to state 1 with probability $\frac{1}{2}$ labelled ε/r (*released to the right*).

All this information may be put in the following diagram:


$\varepsilon/\varepsilon; \frac{1}{3}$
1
t, r, ε
$t/\varepsilon; \frac{1}{3}$
$\varepsilon/t; \frac{1}{3}$
t, r, ε
$r/\varepsilon; \frac{1}{2}$
$\varepsilon/r; \frac{1}{2}$
3
2
$\varepsilon/\varepsilon; \frac{1}{2}$
$\varepsilon/\varepsilon; \frac{1}{2}$

1.1.3 Reachability

For many applications we are interested only in states reachable from a given initial state by a path of positive probability. Given a Markov automaton **Q** and an initial state q_0 there is a subautomaton $\mathbf{Reach}(\mathbf{Q}, q_0)$ whose states are the reachable states, and whose transitions are those of $\mathbf{Q}$ restricted to the reachable states.

1.2 The algebra of weighted automata: operations

Now we define operations on the set $\mathcal{W}$ of weighted automata up to isomorphisms[1] analogous (in a precise sense) to those defined in [40, 42].

1.2.1 Sequential operations

Sum

Definition 1.2.1. *Given weighted automata* $\mathbf{Q}^{X}_{Y;A,B}$ *and* $\mathbf{R}^{Z}_{W;C,D}$ *the* sum $\mathbf{Q}+\mathbf{R}$ *is the weighted automaton which has set of states the disjoint union* $Q+R$*, left interface* $A+C$*, right interface* $B+D$*, top interface* $X+Z$*, bottom interface* $Y+W$*, (all disjoint sums),* $\gamma_0 = \gamma_{0,\mathbf{Q}} + \gamma_{0,\mathbf{R}}$*,* $\gamma_1 = \gamma_{1,\mathbf{Q}} + \gamma_{1,\mathbf{R}}$*. The transition matrices are*

$$[(\mathsf{Q}+\mathsf{R})_{a,b}]_{q,q'} = [\mathsf{Q}_{a,b}]_{q,q'},$$
$$[(\mathsf{Q}+\mathsf{R})_{c,d}]_{r,r'} = [\mathsf{R}_{c,d}]_{r,r'},$$

all other values being 0*.*

The idea of the sum of two Markov automata is that in each state the system is described by one and only of the automata. There is no communication between them.

Proposition 1.2.2. $(\mathcal{W},+)$ *is a commutative monoid.*

Proof. We need to check that given weighted automata $\mathbf{Q}^{X}_{Y;A,B}$ and $\mathbf{R}^{Z}_{W;C,D}$ there exists an isomorphism $\mathbf{Q}+\mathbf{R} \to \mathbf{R}+\mathbf{Q}$. Let

$$Q+R = \{(q,0) | q \in Q\} \cup \{(r,1) | r \in R\}.$$

We define the bijective arrow $\phi_s : Q+R \to R+Q$ such that for each $(u,i) \in Q+R$, $\phi_s(u,i) = (u, \neg i)$, where $\neg$ is the *not* function. Analogously, we define the bijective arrows $\phi_l : A+C \to C+A$, $\phi_r : B+D \to D+B$, $\phi_0 :$

[1] See p.16.

$X + Z \to Z + X, \quad \phi_1 : Y + W \to W + Y$. Given $a \in A, b \in B, q, q' \in Q$ we have[2]:

$$\begin{aligned}
[(\mathsf{Q}+\mathsf{R})_{(a,0),(b,0)}]_{(q,0),(q',0)} &= [\mathsf{Q}_{a,b}]_{q,q'} \\
&= [(\mathsf{R}+\mathsf{Q})_{(a,1),(b,1)}]_{(q,1),(q',1)} \\
&= [(\mathsf{R}+\mathsf{Q})_{\phi_l(a,0),\phi_r(b,0)}]_{\phi_s(q,0),\phi_s(q',0)}
\end{aligned}$$

Similarly, given $c \in C, d \in D, r, r' \in R$,

$$[(\mathsf{Q}+\mathsf{R})_{(c,1),(d,1)}]_{(r,1),(r',1)} = [(\mathsf{R}+\mathsf{Q})_{\phi_l(c,1),\phi_r(d,1)}]_{\phi_s(r,1),\phi_s(r',1)}$$

It is easy to see that the diagram

$$\begin{tikzcd}
X+Z \arrow[r, "\phi_0"] \arrow[d, "\gamma_{0,\mathbf{Q+R}}"'] & Z+X \arrow[d, "\gamma_{0,\mathbf{R+Q}}"] \\
Q+R \arrow[r, "\phi_s"] & R+Q \\
Y+W \arrow[r, "\phi_1"] \arrow[u, "\gamma_{1,\mathbf{Q+R}}"] & W+Y \arrow[u, "\gamma_{1,\mathbf{R+Q}}"']
\end{tikzcd}$$

is commutative. It follows that ϕ is an isomorphism and $(\mathcal{W}, +)$ is commutative.

Similarly, given weighted automata $\mathbf{Q}^X_{Y;A,B}$, $\mathbf{R}^Z_{W;C,D}$ and $\mathbf{S}^U_{V;E,F}$, we can define an obvious isomorphism $(\mathbf{Q}+\mathbf{R})+\mathbf{S} \to \mathbf{Q}+(\mathbf{R}+\mathbf{S})$.

It is straightforward to check that the empty automaton $\varnothing^{\varnothing}_{\varnothing;\varnothing,\varnothing}$ is the neutral element for the sum of weighted automata.
□

Sequential composition

In many concrete examples there are states in which a system may change its configuration. Sequential composition of two weighted automata communicating on a common sequential interface expresses this possibility.

Definition 1.2.3. *Given weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Y_{Z;C,D}$, the* sequential composite of weighted automata $\mathbf{Q} +_Y \mathbf{R}$ *(or briefly $\mathbf{Q} \circ \mathbf{R}$) has set of states the equivalence classes of $Q + R$ under the equivalence relation generated by the relation $\gamma_{1,\mathbf{Q}}(y) \sim \gamma_{0,\mathbf{R}}(y)$, $(y \in Y)$. The left interface is the disjoint sum $A+C$, right interface $B+D$, the top interface is X, the bottom interface is Z. The interface functions are*

$$\gamma_0 = X \overset{\gamma_0}{\to} Q \to Q+R \to (Q+R)/\sim\ ,\ \gamma_1 = Z \overset{\gamma_1}{\to} R \to Q+R \to (Q+R)/\sim\ .$$

[2] Notice that in definition 1.2.1 for simplicity we denote $[(\mathsf{Q}+\mathsf{R})_{(a,0),(b,0)}]_{(q,0),(q',0)}$ by $[(\mathsf{Q}+\mathsf{R})_{a,b}]_{q,q'}$ and $[(\mathsf{Q}+\mathsf{R})_{(c,1),(d,1)}]_{(r,1),(r',1)}$ by $[(\mathsf{Q}+\mathsf{R})_{c,d}]_{r,r'}$.

Denoting the equivalence class of a state s by $[s]$ the transition matrices are:

$$[(\mathsf{Q} +_{\mathsf{Y}} \mathsf{R})_{a,b}]_{[q],[q']} = \sum_{s\in[q],s'\in[q']} [\mathsf{Q}_{a,b}]_{s,s'},$$
$$[(\mathsf{Q} +_{\mathsf{Y}} \mathsf{R})_{c,d}]_{[r],[r']} = \sum_{s\in[r],s'\in[r']} [\mathsf{R}_{c,d}]_{s,s'},$$

all other values being 0.

The idea is that a behaviour of $\mathbf{Q} +_Y \mathbf{R}$ is initially a behaviour of $\mathbf{Q}^X_{Y;A,B}$ and then, when a state in the image of $\gamma_{1,Q}$ is reached, the behaviour may change to a behaviour of $\mathbf{R}^Y_{Z;C,D}$ (but not necessarily because it is also possible that $\mathbf{Q} +_Y \mathbf{R}$ does not change its configuration). We think of γ_0, γ_1 as conditions under which a change of geometry might occur. The correct interpretation is that initially only a process Q exists and then, when a state in its out-conditions is reached, Q may die and the process R be created.

Proposition 1.2.4. *Given weighted automata $\mathbf{Q}^X_{Y;A,B}$, $\mathbf{R}^Y_{Z;C,D}$ and $\mathbf{S}^Z_{W;E,F}$, there exists an isomorphism $(\mathbf{Q} +_Y \mathbf{R}) +_Z \mathbf{S} \to \mathbf{Q} +_Y (\mathbf{R} +_Z \mathbf{S})$.*

Sequential constants

Given two finite sets X and Y, there is a straightforward way of converting a relation $\rho \subset X \times Y$ into a weighted automaton $\mathbf{Seq}(\rho)$ that is a constant of our algebra. We say that $\mathbf{Seq}(\rho)$ is a *sequential constant* because its parallel interfaces are empty sets.

Definition 1.2.5. *Given a relation $\rho \subset X \times Y$ we define a weighted automaton $\mathbf{Seq}(\rho)$ as follows: it has state space the equivalence classes of $X + Y$ under the equivalence relation $\sim$ generated by ρ. It has parallel interfaces $\emptyset$, so there are no transition matrices. The sequential interfaces are $\gamma_0 : X \to (X+Y)/\sim$, $\gamma_1 : Y \to (X+Y)/\sim$, both functions taking an element to its equivalence class.*

Sequential connectors Some special cases have particular importance and are called *sequential connectors* or *wires*:

(i) the automaton corresponding to the identity function 1_A, considered as a relation on $A \times A$ is also called $\mathbf{1}_A$;

(ii) the automaton corresponding to the codiagonal function $\nabla_A : A+A \to A$ (considered as a relation) is called ∇_A, or briefly ∇ when there is no confusion; the automaton corresponding to the opposite relation ∇_A is called ∇^o_A (or briefly ∇^o);

(iii) the automaton corresponding to the function $twist_{A,B} : A \times B \to B \times A$ is called $\mathbf{twist}_{A,B}$;

(iv) the automaton corresponding to the function $\emptyset \subseteq A$ is called $\mathbf{i}_A$; the automaton corresponding to the opposite relation of the function $\emptyset \subseteq A$ is called $\mathbf{i}^o_A$.

Notice that we have overworked the symbol ∇ and it will be used again in another sense; however the context should make clear which use we have in mind.

The role of sequential wires is to *equate states.*

The distributive law The bijection $\delta : X \times Y + X \times Z \to X \times (Y + Z)$ and its inverse $\delta^{-1} : X \times (Y + Z) \to X \times Y + X \times Z$ considered as relations yield weighted automata which we will refer to with the same names $\delta, \boldsymbol{\delta^{-1}}$.

1.2.2 Parallel operations

As done for sequential composition, we define now two parallel operations characterized by the absence or presence of communication channels connecting the components.

Parallel composition

Definition 1.2.6. *Given weighted automata* $\mathbf{Q}^X_{Y;A,B}$ *and* $\mathbf{R}^Z_{W;C,D}$ *the* product *(or* parallel composite without communication*)* $\mathbf{Q} \times \mathbf{R}$ *is the weighted automaton which has set of states* $Q \times R$*, left interface* $A \times C$*, right interface* $B \times D$*, top interface* $X \times Z$*, bottom interface* $Y \times W$*, sequential interface functions* $\gamma_{0,\mathbf{Q}} \times \gamma_{0,\mathbf{R}}$*,* $\gamma_{1,\mathbf{Q}} \times \gamma_{1,\mathbf{R}}$ *and transition matrices*

$$(\mathsf{Q} \times \mathsf{R})_{(a,c),(b,d)} = \mathsf{Q}_{a,b} \otimes \mathsf{R}_{c,d}.[3]$$

If the automata are positive weighted then $\varepsilon_{A\times C} = (\varepsilon_A, \varepsilon_C)$*,* $\varepsilon_{B\times D} = (\varepsilon_B, \varepsilon_D)$*.*

This just says that the weight of a transition from (q, r) to (q', r') with left signal (a, c) and right signal (b, d) is the product of the weight of the transition $q \to q'$ with signals a and b, and the weight of the transition $r \to r'$ with signals c and d.

Sometimes we will say simply *parallel composition* to intend parallel composition without communication.

Proposition 1.2.7. $(\mathcal{W}, \times)$ *is a commutative monoid.*

[3] We denote by the symbol $\otimes$ the usual tensor product of matrices.

Proof. Given weighted automata $\mathbf{Q}^X_{Y;A,B}, \mathbf{R}^Z_{W;C,D}, \mathbf{S}^U_{V;E,F}$, we can define obvious isomorphisms $\mathbf{Q}\times\mathbf{R}\to\mathbf{R}\times\mathbf{Q}$ and $(\mathbf{Q}\times\mathbf{R})\times\mathbf{S}\to\mathbf{Q}\times(\mathbf{R}\times\mathbf{S})$ analogous to those defined for the sum of weighted automata. The unit of $(\mathcal{W},\times)$ is the weighted automata $\mathbf{I}^I_{I;I,I}$, where $I=\{\star\}$, with transition matrix [1].
□

Lemma 1.2.8. *If* $\mathbf{Q}$ *and* $\mathbf{R}$ *are positive weighted automata then so is* $\mathbf{Q}\times\mathbf{R}$ *and*

$$\mathbf{N}(\mathbf{Q}\times\mathbf{R})=\mathbf{N}(\mathbf{Q})\times\mathbf{N}(\mathbf{R}).$$

Hence if $\mathbf{Q}$ *and* $\mathbf{R}$ *are Markov automata then so is* $\mathbf{Q}\times\mathbf{R}$.

Proof.

$$\begin{aligned}\left[\mathsf{N}(\mathsf{Q}\times\mathsf{R})_{(a,c),(b,d)}\right]_{(q,r),(q',r')} &= \frac{[\mathsf{Q}_{a,b}]_{q,q'}\,[\mathsf{R}_{c,d}]_{r,r'}}{\sum_{q',r'}\sum_{(a,c),(b,d)}[\mathsf{Q}_{a,b}]_{q,q'}\,[\mathsf{R}_{c,d}]_{r,r'}}\\ &= \frac{[\mathsf{Q}_{a,b}]_{q,q'}\,[\mathsf{R}_{c,d}]_{r,r'}}{\sum_{q',(a,b)}[\mathsf{Q}_{a,b}]_{q,q'}\sum_{r',(c,d)}[\mathsf{R}_{c,d}]_{r,r'}}\\ &= \left[\mathsf{N}(\mathsf{Q})_{(a,b)}\right]_{q,q'}\left[\mathsf{N}(\mathsf{R})_{(c,d)}\right]_{r,r'}\\ &= \left[(\mathsf{N}(\mathsf{Q})\times\mathsf{N}(\mathsf{R}))_{(a,c),(b,d)}\right]_{(q,r),(q',r')}.\end{aligned}$$

For the second part notice that if $\mathbf{Q}$ and $\mathbf{R}$ are Markov then

$$\mathbf{Q}\times\mathbf{R}=\mathbf{N}(\mathbf{Q})\times\mathbf{N}(\mathbf{R})=\mathbf{N}(\mathbf{Q}\times\mathbf{R})$$

which implies that $\mathbf{Q}\times\mathbf{R}$ is Markov.
□

Lemma 1.2.9. *If* $\mathbf{Q}$ *and* $\mathbf{R}$ *are Markov automata then* $(\mathbf{Q}\times\mathbf{R})^k=\mathbf{Q}^k\times\mathbf{R}^k$.

Parallel with communication

Definition 1.2.10. *Given weighted automata* $\mathbf{Q}^X_{Y;A,B}$ *and* $\mathbf{R}^Z_{W;B,C}$ *the* communicating parallel composite of weighted automata $\mathbf{Q}\times_{\mathbf{B}}\mathbf{R}$ *(or sometimes more briefly* $\mathbf{Q}||\mathbf{R}$*) has set of states* $Q\times R$*, left interface* A*, right interface* C*, top and bottom interfaces* $X\times Z$*,* $Y\times W$*, sequential interface functions* $\gamma_{0,\mathbf{Q}}\times\gamma_{0,\mathbf{R}}$*,* $\gamma_{1,\mathbf{Q}}\times\gamma_{1,\mathbf{R}}$ *and transition matrices*

$$(\mathsf{Q}\times_{\mathsf{B}}\mathsf{R})_{a,c}=\sum_{b\in B}\mathsf{Q}_{a,b}\otimes\mathsf{R}_{b,c}.$$

In communicating parallel composition a state is a pair of states, a transition is a pair of transitions, one of each automaton, which synchronize on the common parallel interface. The weights are multiplied.

The communication of the parts of a parallel composition is anonymous, in the sense that each automaton has a precisely interface and it has no knowledge of the automata with which it is communicating.

From a circuit theory point of view, parallel composition without communication and communicating parallel composition correspond, respectively, to the parallel and series operations of circuit components.

Lemma 1.2.11. *(*$\mathbf{Q}||\mathbf{R})\ ||\ \mathbf{S} = \mathbf{Q}\ ||\ (\mathbf{R}||\mathbf{S})$.

Proof. This follows from the fact that $\otimes$ is associative.
□

It is easy to see that $\mathbf{Q}||\mathbf{R}$ is not necessarily Markov even when both $\mathbf{Q}$ and $\mathbf{R}$ are. The reason is that the communication in the communicating parallel composite reduces the number of possible transitions, so that we must normalize to get (conditional) probabilities. However in a multiple composition it is only necessary to normalize at the end, because of the following lemma.

Lemma 1.2.12. *If* $\mathbf{Q}$ *and* $\mathbf{R}$ *are positive weighted automata then so is* $\mathbf{Q}||\mathbf{R}$ *and* $\mathbf{N}(\mathbf{N}(\mathbf{Q})||\mathbf{N}(\mathbf{R})) = \mathbf{N}(\mathbf{Q}||\mathbf{R})$.

Proof.

$$\begin{aligned}
\left[(\mathsf{NQ}||\mathsf{NR})_{a,c}\right]_{(q,r),(q',r')} &= \sum_{b\in B} \left[\mathsf{NQ}_{a,b}\right]_{q,q'} \otimes \left[\mathsf{NR}_{b,c}\right]_{r,r'} \\
&= \sum_{b\in B} \frac{\left[\mathsf{Q}_{a,b}\right]_{q,q'}}{\sum_{q'}\sum_{a,b}\left[\mathsf{Q}_{a,b}\right]_{q,q'}} \cdot \frac{\left[\mathsf{R}_{b,c}\right]_{r,r'}}{\sum_{r'}\sum_{b,c}\left[\mathsf{R}_{b,c}\right]_{r,r''}} \\
&= \frac{1}{\sum_{q',r'}(\sum_{a,b}\left[\mathsf{Q}_{a,b}\right]_{q,q'}\sum_{b,c}\left[\mathsf{R}_{b,c}\right]_{r,r'})} \sum_{b\in B}\left[\mathsf{Q}_{a,b}\right]_{q,q'}\left[\mathsf{R}_{b,c}\right]_{r,r'} . \\
&= c_{q,r}\left[(\mathsf{Q}||\mathsf{R})_{a,c}\right]_{(q,r),(q',r')}, \text{ where} \\
c_{q,r} &= \frac{1}{\sum_{q',r'}(\sum_{a,b}\left[\mathsf{Q}_{a,b}\right]_{q,q'}\sum_{b,c}\left[\mathsf{R}_{b,c}\right]_{r,r''})} \text{ depends only on } q,r.
\end{aligned}$$

Hence by the lemma 1.1.7 above

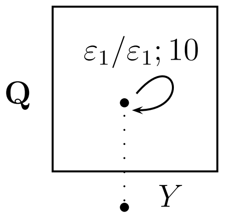
$$\mathbf{N}(\mathbf{NQ}||\mathbf{NR}) = \mathbf{N}(\mathbf{Q}||\mathbf{R}).$$

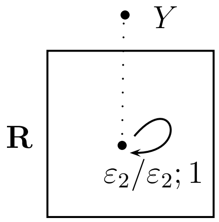
□

Remark 1.2.13. The lemmas 1.2.8 and 1.2.12 mentioned above mean that normalization is compositional with respect to parallel operations. Thus, in an expression involving only parallel operations we can indifferently normalize step by step or only at the end of our calculation.
This is *not* true for sequential operations. We will see in chapter 4 that in many probability problems a wrong use of normalization leads to paradoxes. In fact if we normalize at the beginning of the calculation we often loose the relative weight given to transitions in different states.

In the following example we show that normalization is not compositional with respect to sequential composition.

Example 1.2.14. Consider the weighted automaton **Q** and **R** as in the following pictures:





Let **U** be the automaton $\mathbf{N}(\mathbf{Q} +_Y \mathbf{R})$. Then **U** is the weighted automaton with one state and transition matrices

$$\mathsf{U}_{\varepsilon_1,\varepsilon_1} = \begin{bmatrix}\frac{10}{11}\end{bmatrix} \quad \mathsf{U}_{\varepsilon_2,\varepsilon_2} = \begin{bmatrix}\frac{1}{11}\end{bmatrix}$$

Let **V** be the automaton $\mathbf{N}(\mathbf{N}(\mathbf{Q}) +_Y \mathbf{N}(\mathbf{R}))$. Then **V** has one state as **U** but transition matrices:

$$\mathsf{V}_{\varepsilon_1,\varepsilon_1} = \begin{bmatrix}\frac{1}{2}\end{bmatrix} \quad \mathsf{V}_{\varepsilon_2,\varepsilon_2} = \begin{bmatrix}\frac{1}{2}\end{bmatrix}$$

If we normalize before sequential composition we loose the relative weight of each transition.

Definition 1.2.15. *If* **Q** *and* **R** *are Markov automata,* **Q** *with left interface A and right interface B,* **R** *with left interface B and right interface C then the series composite of Markov automata* $\mathbf{Q} \cdot \mathbf{R}$ *is defined to be* $\mathbf{Q} \cdot \mathbf{R} = \mathbf{N}(\mathbf{Q} || \mathbf{R})$.

Theorem 1.2.16. $(\mathbf{Q} \cdot \mathbf{R}) \cdot \mathbf{S} = \mathbf{Q} \cdot (\mathbf{R} \cdot \mathbf{S})$.

Proof.

$$\begin{aligned}(\mathbf{Q}\cdot\mathbf{R})\cdot\mathbf{S} &= \mathbf{N}(\mathbf{N}(\mathbf{Q}||\mathbf{R})||\mathbf{S})\\ &= \mathbf{N}(\mathbf{N}(\mathbf{Q}||\mathbf{R})||\mathbf{N}(\mathbf{S})) \text{ since } \mathbf{S} \text{ is Markov}\\ &= \mathbf{N}((\mathbf{Q}||\mathbf{R})||\mathbf{S}) = \mathbf{N}(\mathbf{Q}||(\mathbf{R}||\mathbf{S})) \text{ by 1.2.11 and 1.2.12}\\ &= \mathbf{N}(\mathbf{N}(\mathbf{Q})||\mathbf{N}(\mathbf{R}||\mathbf{S})) \text{ by 1.2.12}\\ &= \mathbf{N}(\mathbf{Q}||\mathbf{N}(\mathbf{R}||\mathbf{S})) = \mathbf{Q}\cdot(\mathbf{R}\cdot\mathbf{S}) \text{ since } \mathbf{Q} \text{ is Markov.}\end{aligned}$$

□

Theorem 1.2.17. *If* $\mathbf{Q}$ *and* $\mathbf{R}$ *are Markov automata then*

$$(\mathbf{Q}\times\mathbf{R})^k = \mathbf{Q}^k \times \mathbf{R}^k$$

Proof.

$$\begin{aligned}&(\mathsf{Q}\times\mathsf{R})^k_{(u,u'),(v,v')}\\ &= (\mathsf{Q}_{a_1,b_1}\otimes\mathsf{R}_{a'_1,b'_1})(\mathsf{Q}_{a_2,b_2}\otimes\mathsf{R}_{a'_2,b'_2})\cdots(\mathsf{Q}_{a_k,b_k}\otimes\mathsf{R}_{a'_k,b'_k})\\ &= (\mathsf{Q}_{a_1,b_1}\otimes\mathsf{Q}_{a_2,b_2}\otimes\cdots\otimes\mathsf{Q}_{a_k,b_k})(\mathsf{R}_{a'_1,b'_1}\otimes\mathsf{R}_{a'_2,b'_2}\otimes\cdots\otimes\mathsf{R}_{a'_k,b'_k})\\ &= (\mathsf{Q}^k\times\mathsf{R}^k)_{(u,u'),(v,v')}.\end{aligned}$$

□

Remark. It is not the case that if $\mathbf{Q}$, $\mathbf{R}$ are Markov then $(\mathbf{Q}\cdot\mathbf{R})^k = \mathbf{Q}^k\cdot\mathbf{R}^k$. The reason is that normalizing length k steps in a weighted automaton is not the same as considering k step paths in the normalization of the automaton.

Parallel constants

Definition 1.2.18. *Given a relation* $\rho \subset A\times B$ *we define a weighted automaton* $\mathbf{Par}(\rho)$ *as follows: it has one state* $*$ *say. Top and bottom interfaces have one element. The transition matrices* $[\mathsf{Par}(\rho)_{a,b}]$ *are* 1×1 *matrices, that is, real numbers. Then* $\mathsf{Par}(\rho)_{a,b} = 1$ *if* ρ *relates* a *and* b, *and* $\mathsf{Par}(\rho)_{a,b} = 0$ *otherwise. If* $(\varepsilon_A, \varepsilon_B) \in \rho$ *then* $\mathbf{Par}(\rho)$ *is also positive weighted.*

Parallel connectors Some special cases, all described in [40], have particular importance and are called *parallel connectors* or *wires*:

(i) the automaton corresponding to the identity function 1_A, considered as a relation on $A\times A$ is also called $\mathbf{1}_A$;

(ii) the automaton corresponding to the diagonal function $\Delta_A : A \to A\times A$ (considered as a relation) is called $\mathbf{\Delta}_A$ (or briefly $\mathbf{\Delta}$); the automaton corresponding to the opposite relation of Δ_A is called $\mathbf{\Delta}^o_A$ (or $\mathbf{\Delta}^o$);

(iii) the automaton corresponding to the function $twist_{A,B} : A \times B \to B \times A$ is again called $\mathbf{twist}_{A,B}$;

(iv) the automaton corresponding to the projection function $A \to \{*\}$ is called $\mathbf{p}_A$ and its opposite $\mathbf{p}^o_A$.

The role of parallel wires is to *equate signals*.

Parallel codiagonal The automaton corresponding to the function $\nabla_A : A + A \to A$ is called the *parallel codiagonal*, and is denoted ∇ where is no confusion[4]. The automaton corresponding to the opposite relation is written ∇^o.

1.2.3 Some derived operations

In this section we define some important operations derived from the canonical operations and the constants defined above. Their meaning will became more clear looking ahead at the graphical representation in the next section.

Local sum

Given weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Z_{W;A,B}$ the local sum $\mathbf{Q} +_{\mathbf{loc}} \mathbf{R}$ is defined to be $\nabla^o_A||(\mathbf{Q}+\mathbf{R})||\nabla_A$. It has top and bottom interfaces $X+Z$ and $Y+W$, and left and right interfaces A and B.

Local sequential composition

Given weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Y_{Z;A,B}$ the local sequential composite $\mathbf{Q} +_{Y,loc} \mathbf{R}$ (denoted briefly $\bullet$) is defined to be

$$\nabla^o_A||(\mathbf{Q} +_Y \mathbf{R})||\nabla_A.$$

It has top and bottom interfaces X and Z, and left and right interfaces A and B.

Sequential feedback

Given weighted automata $\mathbf{Q}^{X+Z}_{Y+Z;A,B}$ sequential feedback $\mathsf{Sfb}_Z(\mathbf{Q}^{X+Z}_{Y+Z;A,B})$ is defined to be

$$(\mathbf{1}_X + \mathbf{i}_Z) \bullet (\mathbf{1}_X + \nabla^o_Z) \bullet (\mathbf{Q} + \mathbf{1}_Z) \bullet (\mathbf{1}_Y + \nabla_Z) \bullet (\mathbf{1}_Y + \mathbf{i}^o_Z).$$

[4] We use the symbol ∇ to denote also the (sequential) codiagonal (see p. 25).

Parallel feedback

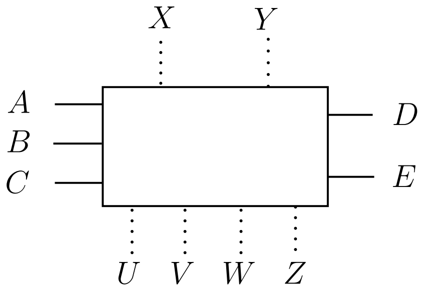
Given weighted automata $\mathbf{Q}^{X}_{Y;A\times C,B\times C}$ parallel feedback $\mathsf{Pfb}_C(\mathbf{Q}^{X}_{Y;A+C,B+C})$ is defined to be

$$(\mathbf{1}_A \times \mathbf{p}^o_C)||(\mathbf{1}_A \times \mathbf{\Delta}_C)||(\mathbf{Q} \times \mathbf{1}_C)||(\mathbf{1}_B \times \mathbf{\Delta}^o_C)||(\mathbf{1}_B \times \mathbf{p}_C).$$

1.3 Graphical representation of expressions of weighted automata

Not only do weighted automata have a graphical representation, as seen above, but so also do expressions of automata, as described in [40]. We extend the representation given in that paper to the combination of sequential and parallel operations and constants. We will see in section 3.1 that the graphical representation of single automata is actually a special case of this new representation of expressions, modulo the equations satisfied by wires (the Frobenius and separable algebra equations first introduces in [12], see also [57]).

In general an expression will be represented by a diagram of the following sort:

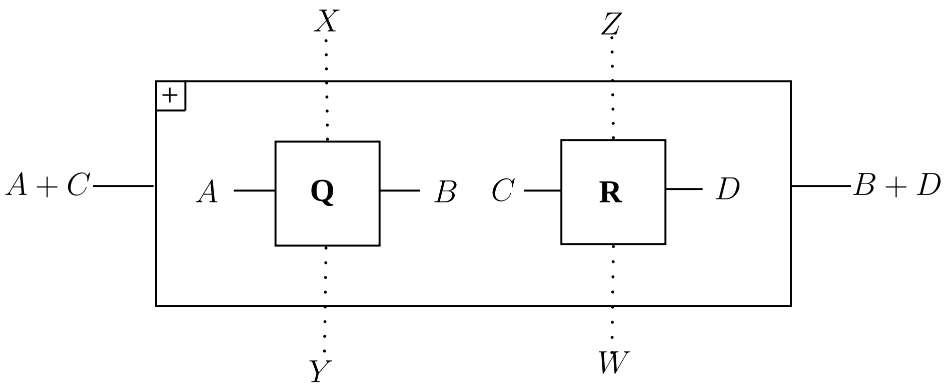


The multiple lines on the left and right hand sides correspond to parallel interfaces which are products of sets. For example, the component has left interface $A \times B \times C$. Instead the multiple lines on the top and bottom correspond to sequential interfaces which are disjoint sums of sets, so the top interface is $X + Y + Z$.

1.3.1 Operations and constants

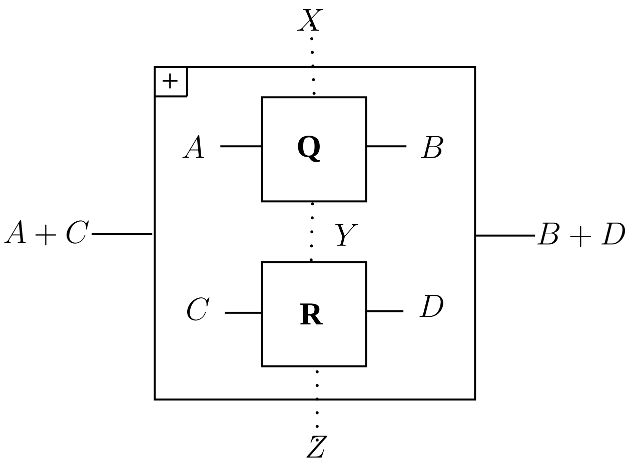
Sum

The sum $\mathbf{Q}^{X}_{Y;A,B} + \mathbf{R}^{Z}_{W;C,D}$ is represented as:



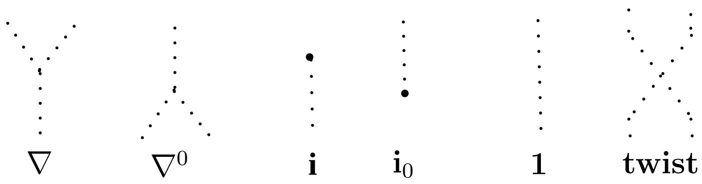
Sequential composition

The sequential composite $\mathbf{Q}^X_{Y;A,B} +_Y \mathbf{R}^Y_{Z;C,D}$ is represented as:



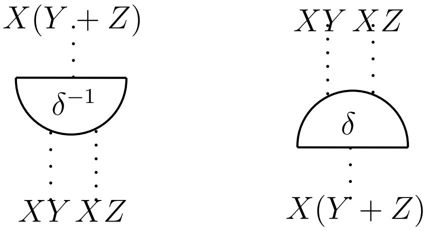
Sequential connectors

The various sequential connectors are represented as:



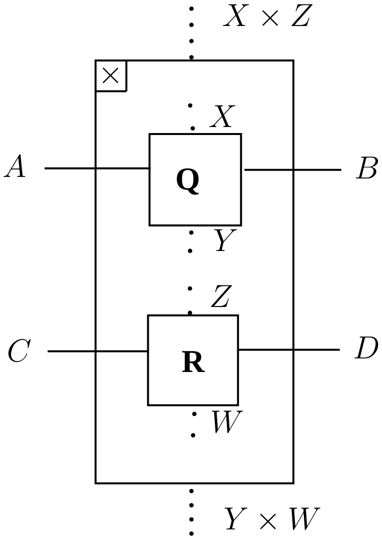
Distributive law

The distributive law $\delta^{-1} : X \times (Y+Z) \to X \times Y + X \times Z$ and its opposite are represented as:



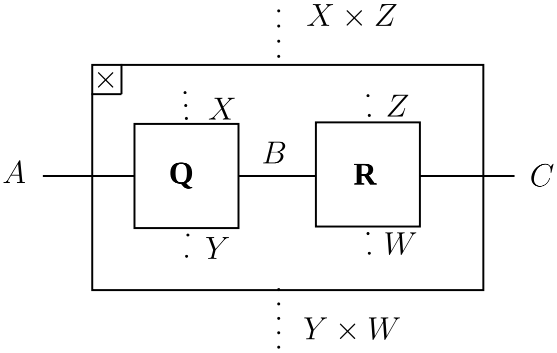
Product

The product $\mathbf{Q}^{X}_{Y;A,B} \times \mathbf{R}^{Z}_{W;C,D}$ is represented as:



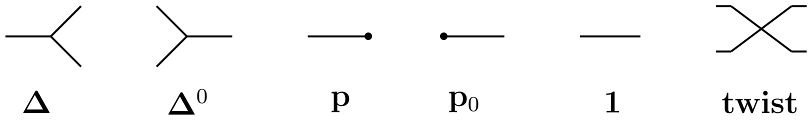
Parallel with communication

The parallel with communication $\mathbf{Q}^{X}_{Y;A,B} \times_B \mathbf{R}^{Z}_{W;B,C}$ is represented as:



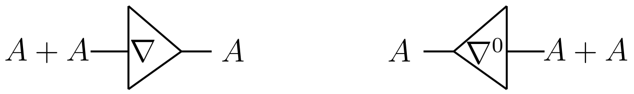
Parallel connectors

The various parallel connectors are represented as:



Parallel codiagonal

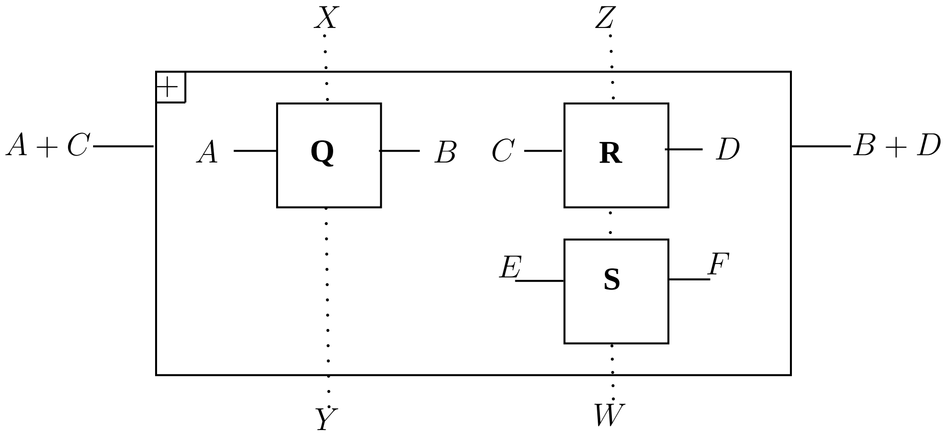
The parallel codiagonal $\nabla : A + A \to A$ and its opposite ∇^o are represented as:



1.3.2 Some derived operations and constants

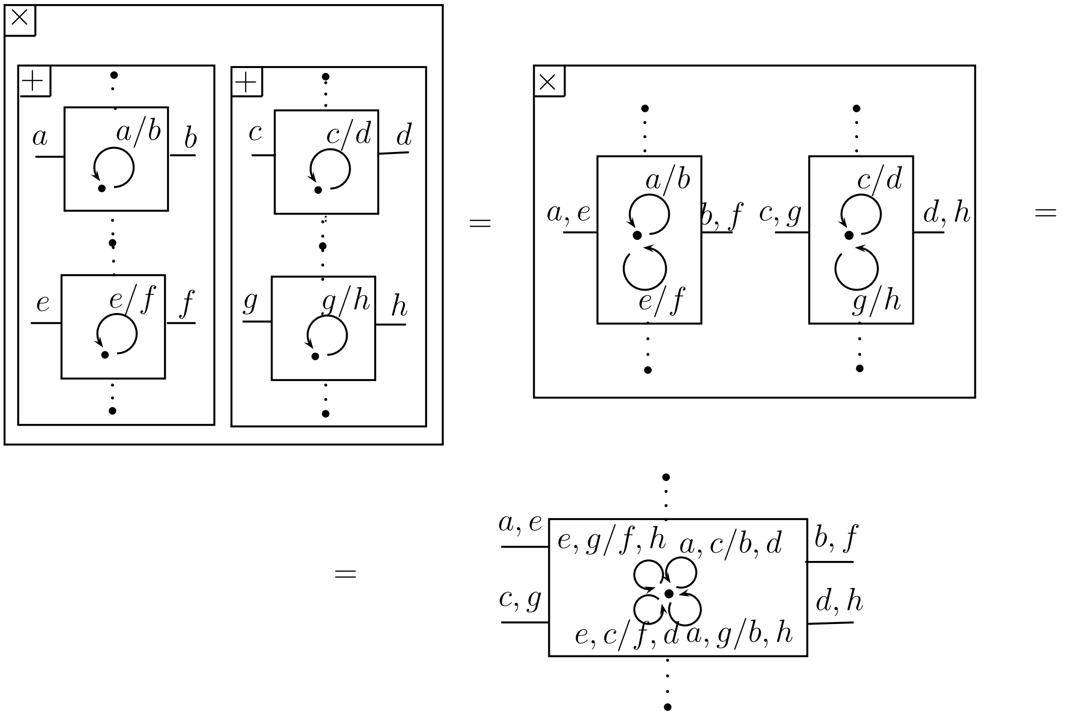
Repeated sequential and repeated parallel operations

When representing repeated sequential operations, frequently we omit all but the last bounding rectangle, as in the following picture:

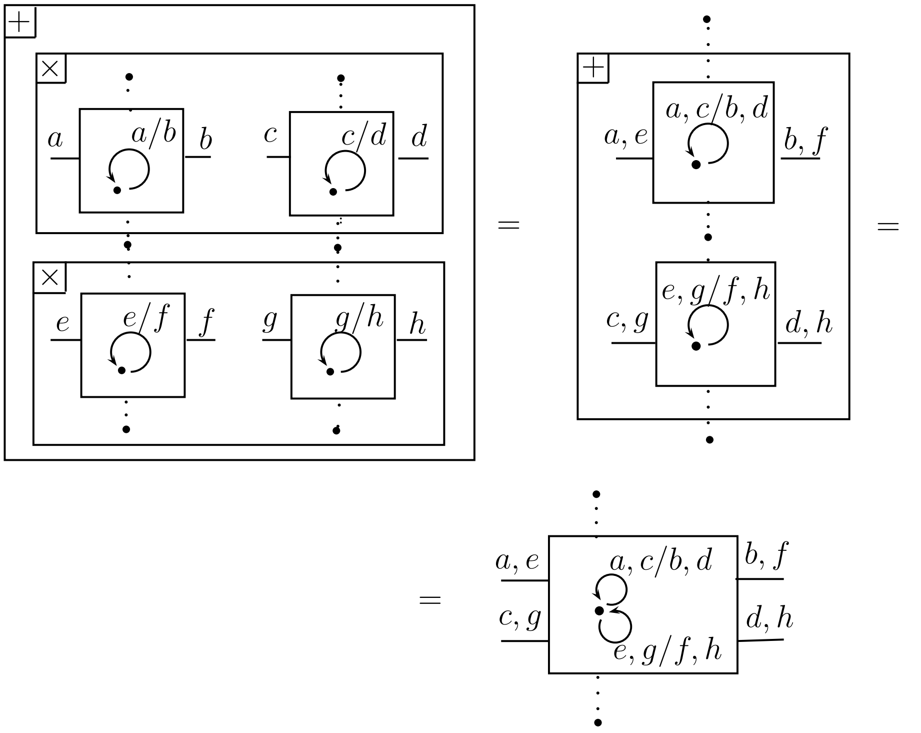


With care this does not lead to ambiguity - different interpretations lead to at worst isomorphic automata. We do the same with repeated parallel operations. It is not possible to remove bounding rectangles for mixed repeated sequential and parallel operations without serious ambiguity.

Example 1.3.1. Consider the following example in which all the transitions have weight 1:



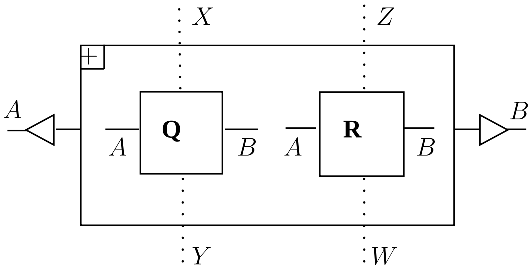
If we calculate instead first the products and then the sequential composition we obtain:

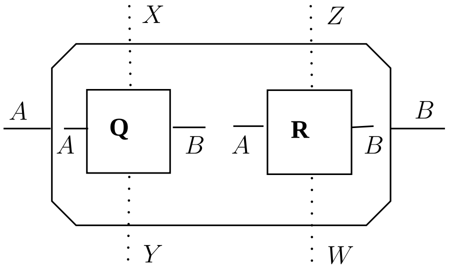


The two results are not isomorphic.

Local sum

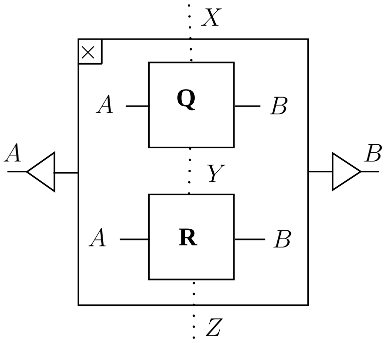
The local sum $\mathbf{Q}^X_{Y;A,B} +_{loc} \mathbf{R}^Z_{W;A,B}$ is represented by the first diagram below which includes the parallel codiagonals, but also, since it is such a common derived operation, more briefly by the second diagram below:

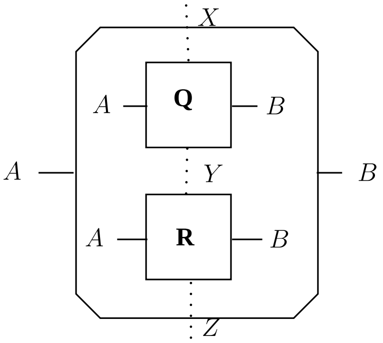




Local sequential

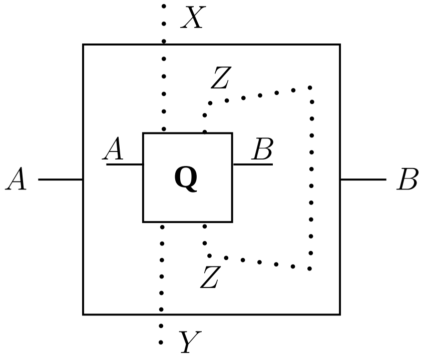
The local sequential $\mathbf{Q}^X_{Y;A,B} +_{Y,loc} \mathbf{R}^Y_{Z;A,B}$ is represented by the first diagram below which includes the parallel codiagonals, but also, since it is such a common derived operation, more briefly by the second diagram below:





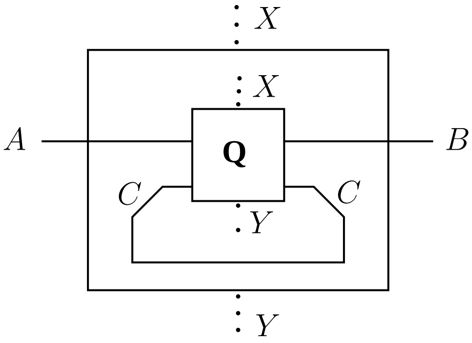
Sequential feedback

The sequential feedback $\mathsf{Sfb}_Z(\mathbf{Q}^{X+Z}_{Y+Z;A,B})$ is represented by the diagram:



Parallel feedback

The parallel feedback $\mathsf{Pfb}_C(\mathbf{Q}^X_{Y;A\times C,B\times C})$ is represented by the diagram:



1.4 The Dining Philosophers system

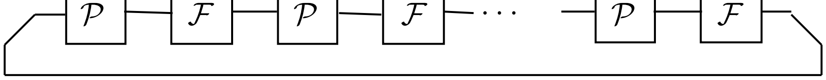
In this section we show how to describe a system of n Dining Philosophers (with 12^n states), and we observe that Perron-Frobenius theory yields a proof that the probability of reaching deadlock tends to one as the number of steps goes to infinity.

The model of the Dining Philosophers problem we consider is an expression in the algebra, involving also the automata **Phil** and **Fork**. The system of n Dining Philosophers is

$$\mathbf{DP}_n = \mathbf{Pfb}_A(\mathbf{Phil}||\mathbf{Fork}||\mathbf{Phil}||\mathbf{Fork}||\cdots||\mathbf{Phil}||\mathbf{Fork}),$$

where in this expression there are n philosophers and n forks.

We may represent this system by the following diagram, where we abbreviate **Phil** to $\mathcal{P}$ and **Fork** to $\mathcal{F}$.



Let us examine the case when $n = 2$ with initial state $(1,1,1,1)$. Let $\mathbf{Q}$ be the reachable part of $\mathbf{DP}_2$. The states reachable from the initial state are $q_1 = (1,1,1,1)$, $q_2 = (1,3,3,2)$, $q_3 = (3,2,1,3)$, $q_4 = (1,1,4,2)$, $q_5 = (4,2,1,1)$, $q_6 = (1,3,2,1)$, $q_7 = (2,1,1,3)$, $q_8 = (2,3,2,3)$ (q_8 is the unique deadlock state). The single matrix of the automaton Q, using this ordering of the states, is

$$\begin{bmatrix}
\frac{1}{4} & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} \\
0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$

Calculating powers of this matrix[5] we see that the probability of reaching deadlock from the initial state in 2 steps is $\frac{23}{48}$, in 3 steps is $\frac{341}{576}$, and in 4 steps is $\frac{4415}{6912}$.

1.4.1 The probability of deadlock

The idea of this section is to apply Perron-Frobenius theory (see, for example [30]) to the Dining Philosopher automaton. However, for convenience, we give the details of the proof of the case we need, without refering to the general theorem.

Definition 1.4.1. *Consider a Markov automaton* $\mathbf{Q}$ *with input and output sets being one element sets* $\{\varepsilon\}$. *A state* q *is called a* deadlock *if the only transition out of* q *with positive probability is a transition from* q *to* q *(the probability of the transition must necessarily be* 1*).*

Theorem 1.4.2. *Consider a Markov automaton* $\mathbf{Q}$ *with interfaces being one element sets, with an initial state* q_0. *Suppose that:*

(i) $\mathbf{Q}$ *has precisely one reachable deadlock state,*

(ii) for each reachable state, not a deadlock, there is a path with non-zero probability to q_0, *and*

(iii) for each reachable state q *there is a transition with non-zero probability to itself.*

Then the probability of reaching a deadlock from the initial state in k *steps tends to* 1 *as* k *tends to infinity.*

[5] For Maple calculations of the probability of reaching deadlock in the Dining Philosopher system see the appendix A.

Proof. Let $\mathbf{R} = \mathbf{Reach}(\mathbf{Q}, q_0)$. Suppose $\mathbf{R}$ has m states. Then in writing the matrix R we choose to put the deadlock last, so that R has the form

$$\mathsf{R} = \begin{bmatrix} \mathsf{S} & \mathsf{T} \\ 0 & 1 \end{bmatrix}$$

where S is $(m-1) \times (m-1)$ and T is $(m-1) \times 1$. Now

$$\mathsf{R}^k = \begin{bmatrix} \mathsf{S}^k & \mathsf{T}_k \\ 0 & 1 \end{bmatrix}$$

for some matrix T_k. Condition (i) implies that there is a path with positive probability (a positive path) from any non-deadlock state to any other in $\mathbf{R}$. Condition (ii) implies that if there is a positive path of length l between two states then there is also a positive path of all lengths greater than l. These two facts imply that there is a k_0 such that from any non-deadlock state to any other state there is a positive path of length k_0. For this k_0 the matrix $T_{k_0}$is *strictly positive*. This means that the row sums of S^{k_0} are strictly less than 1. But the eigenvalues of a matrix are dominated in absolute value by the maximum of the absolute row sums (the sums of absolute values of the row elements). Hence the eigenvalues of S^{k_0} and hence of S all have absolute value less than 1. But by considering the Jordan canonical form of a matrix whose eigenvalues all have absolute values less than 1 it is easy to see that S^k tends to 0 as k tends to infinity. Hence T_k tends to the column vector all of whose entries are 1. Hence the probability of reaching the deadlock from any of the other states in k steps tends to 1 as k tends to infinity.
□

Corollary 1.4.3. *In the Dining Philosopher problem* $\mathbf{DP}_n$ *with* q_0 *being the state* $(1, 1, \cdots, 1)$ *the probability of reaching a deadlock from the initial state in* k *steps tends to* 1 *as* k *tends to infinity.*

Proof. We just need to verify the conditions of the theorem for the Dining Philosopher problem. It is straightforward to check that the state

$$(2, 3, 2, 3, \cdots, 2, 3)$$

in which the philosophers are all in state 2 and the forks in state 3 is a reachable deadlock. It is clear that in any state q there is a positive transition to q, since each component has silent moves in each state. We need only check that for any reachable state other than this deadlock that there is a positive path to the initial state. Consider the states f_1, f_2 of two forks adjacent modulo n, and the state p of the philosopher between these two forks.

Examining the positive paths possible in two adjacent forks and the corresponding philosopher we see that the reachable configurations are limited to (a) $f_1 = 1$, $p = 1$, $f_2 = 1$, (b) $f_1 = 1$, $p = 1$, $f_2 = 3$, (c) $f_1 = 1$, $p = 4$, $f_2 = 2$, (d) $f_1 = 2$, $p = 1$, $f_2 = 1$, (e) $f_1 = 2$, $p = 1$, $f_2 = 3$, (f) $f_1 = 3$, $p = 2$, $f_2 = 1$, (g) $f_1 = 3$, $p = 2$, $f_2 = 3$, (h) $f_1 = 3$, $p = 3$, $f_2 = 2$, (i) $f_1 = 2$, $p = 4$, $f_2 = 2$. We will show that in states other than the deadlock or the inital state there is a transition of the system which increases the number of forks in state 1. Notice that in a reachable state the states of adjacent forks determine the state of the philosopher between. Consider the possible configurations of fork states. We need not consider cases all forks are in state 1 (initial), or all in state 3 (the known deadlock). Given two adjacent forks in states $3, 2$ there are transitions which only involve this philosopher and the two forks (apart from null signals) which result in one of the forks returning to state 1 (the philosopher puts down a fork that he holds). This is also the case when two adjacent forks are in states $1, 2$ or $2, 2$ or $3, 1$. But in a circular arrangement other than all 1's or all 3's one of the pairs $1, 2$ or $2, 2$ or $3, 1$ or $3, 2$ must occur.
□

Remark 1.4.4. Notice that in the proof described above we did not use the specific positive probabilities of the actions of the philosophers and forks. Hence the result is true with any positive probabilities replacing the specific ones we gave in the description of the philosopher and fork. In fact, different philosophers and forks may have different probabilities without affecting the conclusion of the corollary[6].

Remark 1.4.5. There are many non-compositional algorithms for determining properties like deadlock in other automata models originating in [55, 66] (see for example [3]).

[6] For Maple calculations of the probability of reaching deadlock in $\mathbf{DP}_n$ with different choices of weights see the appendix A.

Chapter 2

Systems with parallel and sequential interfaces

This chapter presents in detail a mathematical model based on spans and cospans of graphs which permits a compositional description of systems with both sequential and parallel communication. The presentation of the model is very abstract but it serves as a guide to describe very concrete situations.

We develop further the algebra for the sequential and parallel composition of systems introduced in the two papers [39], [41]. Whereas those papers dealt with the finite state control, here we add data structures. As in [41] the sequential composition is a cospan composition and the parallel a span composition. The algebra of parallel operations and constants has the structure of a well-supported compact closed category (briefly *wscc category*) - a symmetric monoidal category for which every object has a separable algebra structure ([12]). Similarly, the algebra of sequential operations and constants has a structure of a wscc category.

There is a strong analogy between our notion of system with parallel and sequential interfaces and the definition of weighted automata introduced in chapter 1. First of all the operations are similar, except for the sum which is different. In fact, in weighted and Markov automata model we have choosen to define a slightly different sum in order to simplify the description of concrete problems rather than defining a sum analogous to the sum of systems. The algebra of weighted and Markov automata with the simplified sum is not a wscc category with respect to sequential operations and constants whereas for parallel operations and constants it is. In the abstract model of systems we prefer the more complicated definition of sum in order two have two symmetric monoidal structures.

Further, in the cospan-span model for each transition there is a set of arrows attached which is the analogous of the weighting of a transition in weighted and Markov automata.

The plan of the chapter is as follows. We begin with the basic concepts of span and cospan of graphs that are necessary to understand the following sections. Then we introduce the abstract notion of a system with sequential and parallel interfaces. In section 2.3 we make simplifying assumptions arriving in the section 2.4 with an algebra that is, in effect, an implementable programming language of systems. The reader should be aware that the word *system* has an increasingly specific meaning in successive sections of the chapter. The motive for proceeding in this way is to show that what may appear as arbitrary and unmotivated in section 2.4 actually arises in a natural way from general considerations. In addition, for some applications a different set of simplifying assumptions may be more appropriate. Finally, as examples of programs in the language we indicate how sequential programming, classical concurrency examples, hierarchy and change of geometry may be expressed.

An important element of this chapter is the matrix calculus which arises from the fact that categories of spans in an extensive category [11] have direct sums. It allows an explicit relation between programs with data types, and finite automata which express the control structure of the program.

Another important element is the role of the distributive law in various roles, including flattening hierarchy.

In this chapter we concentrate on the operations of the algebra, and its expressivity, rather than the equations satisfied. Theoretical considerations behind this model include [59], [57],[11], [12],[13] and [43].

2.1 Spans and cospans of graphs

We introduce the categories of spans and cospans of graphs in order to have the necessary mathematical notions to define systems with parallel and sequential interfaces.

By a graph G we mean here a set of states $states(G)$, a set of transitions $transitions(G)$ and two functions $source, target : transitions(G) \to states(G)$ which specify the source state and target state of a transition. We say that a graph G is *finite* if $states(G)$ and $transitions(G)$ are finite sets.

Given two finite graphs A and B, a *span from A to B* is a diagram

$$\begin{tikzcd} A & G \arrow[l, "\partial_0^G"'] \arrow[r, "\partial_1^G"] & B \end{tikzcd}$$

in the category of graphs and morphisms between them. We say that two spans $\partial_0^G : A \leftarrow G \rightarrow B : \partial_1^G$ and $\partial_0^H : A \leftarrow H \rightarrow B : \partial_1^H$ are *isomorphic* if there exists an isomorphism of graphs $\phi : G \rightarrow H$ such that the following diagram is commutative:

$$\begin{tikzcd}
A & G \arrow[l, "\partial_0^G"'] \arrow[r, "\partial_1^G"] \arrow[d, "\phi"] & B \\
& H \arrow[lu, "\partial_0^H"] \arrow[ru, "\partial_1^H"'] &
\end{tikzcd}$$

The composition of spans is by pullback: given two spans

$$\begin{tikzcd}
& G \arrow[ld, "\partial_0^G"'] \arrow[rd, "\partial_1^G"] & & H \arrow[ld, "\partial_0^H"'] \arrow[rd, "\partial_1^H"] & \\
A & & B & & C
\end{tikzcd}$$

the composite is $\partial_0^G p : A \leftarrow G \times_B H \rightarrow C : \partial_1^G q$ where $G \times_B H, p, q$ is a pullback of ∂_1^G, ∂_0^H.

$$\begin{tikzcd}
& & G \times_B H \arrow[ld, "p"'] \arrow[rd, "q"] & & \\
& G \arrow[ld, "\partial_0^G"'] \arrow[rd, "\partial_1^G"] & & H \arrow[ld, "\partial_0^H"'] \arrow[rd, "\partial_1^H"] & \\
A & & B & & C
\end{tikzcd}$$

Since span composition is unique up to an isomorphism of the central graph, we will always consider spans up to isomorphisms. For simplicity we will often denote the arrows of a span as ∂_0 and ∂_1 without specifying the center G. We denote by *Span(Graph)* the category whose objects are finite graphs and whose arrows are isomorphism classes of spans of graphs.

We denote by *Cospan(Graph)* the dual category of $Span(Graph)$. Thus, a *cospan* is a diagram $\gamma_0 : X \rightarrow G \leftarrow Y : \gamma_1$. Composition of cospans is by pushout and an arrow of $Cospan(Graph)$ is an isomorphism class of cospans.

2.2 Systems with sequential and parallel interfaces

We represent systems by (possibly infinite) graphs of states and transitions, to which we will be adding extra structure.

2.2.1 Sequential interfaces

In order to compose sequentially one system with another both systems must have appropriate interfaces. The idea comes from the sequential composition of automata, which occurs for example in Kleene's theorem: certain states (final states) of one automata are identified with certain states (initial states) of another. Here we replace initial and final states by graph morphisms into the graph of the system.

Definition 2.2.1. A system with sequential interfaces *is a cospan* $\gamma_0 : X \rightarrow G \leftarrow Y : \gamma_1$ *of graphs. The graph* G *is the graph of the system;* X *and* Y *are the graphs of the interfaces. We write this also as* G_Y^X*. Composition of systems is by pushout. The category of systems with sequential interfaces is* $Cospan(Graph)$*. A behaviour of* G_Y^X *is a path in the central graph* G*.*

As we said above, in speaking of the *category* of cospans we should consider cospans only up to an isomorphism of the central graph of the cospan. In practice we will always consider representative cospans, and any equation we state will be true only up to isomorphism. The same proviso should be applied to our discussion later of spans, and systems.

2.2.2 Parallel interfaces

Similarly, to compose in communicating parallel two systems each system must have a parallel interface. The idea here comes, for example, from circuits. A circuit component has a physical boundary and transitions of the circuit component produce transitions on the physical boundary. Joining two circuit components, the transitions of the resulting system are restricted by the fact that the transitions on the common boundary must be equal. We describe the relation between transitions of the system G and the transitions on a boundary A by a graph morphism $G \rightarrow A$. To obtain a category when we compose we require that a system has two parallel interfaces.

Definition 2.2.2. A system with parallel interfaces *is a span* $\partial_0 : A \leftarrow G \rightarrow B : \partial_1$ *of graphs. The graph* G *is the graph of the system;* A *and* B *are the graphs of the interfaces. We write this also as* $G_{A,B}$*.*

Composition of systems is by pullback. The category of systems with sequential interfaces is $Span(Graph)$*. A behaviour of* $G_{A,B}$ *is a path in the central graph* G*.*

2.2.3 Combined sequential and parallel interfaces

We can now combine the two kinds of interfaces just defined to obtain a system that allows both sequential and parallel composition.

Definition 2.2.3. *A* system with sequential and parallel interfaces *consists of a commutative diagram of graphs and graph morphisms*

$$\begin{tikzcd}
G_0 \arrow[d, "\gamma_0"'] & X \arrow[l, "\partial_0"'] \arrow[r, "\partial_1"] \arrow[d, "\gamma_0"'] & G_1 \arrow[d, "\gamma_0"'] \\
A & G \arrow[l, "\partial_0"'] \arrow[r, "\partial_1"] & B \\
G_2 \arrow[u, "\gamma_1"] & Y \arrow[l, "\partial_0"'] \arrow[r, "\partial_1"] \arrow[u, "\gamma_1"] & G_3 \arrow[u, "\gamma_1"]
\end{tikzcd}$$

or more briefly, when we are not emphasizing the corner graphs, as

$$\begin{tikzcd}
\bullet \arrow[d] & X \arrow[l] \arrow[r] \arrow[d] & \bullet \arrow[d] \\
A & G \arrow[l] \arrow[r] & B \\
\bullet \arrow[u] & Y \arrow[l] \arrow[r] \arrow[u] & \bullet \arrow[u]
\end{tikzcd}$$

We denote such a system very briefly as $G^X_{Y;A,B}$, or even G^X_Y or $G_{A,B}$ or even just G, depending on the context. A behaviour of $G^X_{Y;A,B}$ is a path in the central graph G. Another useful notation is as follows: given an object O in the diagram we denote the four adjacent objects by $O^{\rightarrow}, O^{\leftarrow}, O^{\downarrow}$ and $O^{\uparrow}$; for example, $G^{\leftarrow\uparrow} = G^{\uparrow\leftarrow} = G_0$.

Such a system may be regarded in two ways:

(i) as three systems with parallel interfaces, the first X_{G_0,G_1} and third Y_{G_2,G_3} being sequential interfaces to the second $G_{A,B}$;

(ii) as three systems with sequential interfaces, two ($A^{G_0}_{G_2}$, $B^{G_1}_{G_3}$) being parallel interfaces to the other (G^X_Y).

The point is that to compose in parallel a system with sequential interfaces requires that the sequential interfaces also have parallel interfaces. It is not necessary that the parallel interfaces themselves have parallel interfaces, since interfaces are identified, not composed, in the composition. A similar remark applies to sequential composition. Notice that for simplicity we have used the same symbols γ_0, γ_1 for all the sequential interface morphisms and similarly ∂_0, ∂_1 for all the parallel interface morphisms.

Operations on systems

We define now operations on systems with parallel and sequential interfaces analogous to those defined for weighted automata.

Definition 2.2.4. *Two systems* $G^X_{Y;A,B}$ *and* $H^Z_{W;B,C}$ *admit a composition by pullback, the* communicating parallel *(or horizontal)* composition*, denoted* $G^X_{Y;A,B} \times_B H^Z_{W;B,C}$ *(or briefly* $G^X_{Y;A,B} || H^Z_{W;B,C}$*).*

By the associativity of pullback it follows the associativity of horizontal composition.

Proposition 2.2.5. *The communicating parallel composition of systems is associative.*

Of course, certain corner graphs of G and H are required to be the same. This applies also in the next definition.

Definition 2.2.6. *Two systems* $G^X_{Y;A,B}$ *and* $K^Y_{Z;D,E}$ *admit a composition by pushout, the* sequential *(or vertical)* composition*, denoted* $G^X_{Y;A,B} +_Y K^Y_{Z;D,E}$ *(or briefly* $G^X_{Y;A,B} \circ K^Y_{Z;D,E}$*).*

Proposition 2.2.7. *The sequential composition of systems of associative.*

It is important to note that, given four systems $G^X_{Y;A,B}$, $H^U_{V;B,C}$, $K^Y_{Z;D,E}$, $L^V_{W;E,F}$ in the following configuration

$$\begin{tikzcd}
\bullet \arrow[d] & X \arrow[l] \arrow[r] \arrow[d] & \bullet \arrow[d] & U \arrow[l] \arrow[r] \arrow[d] & \bullet \arrow[d] \\
A & G \arrow[l] \arrow[r] & B & H \arrow[l] \arrow[r] & C \\
\bullet \arrow[u] \arrow[d] & Y \arrow[l] \arrow[r] \arrow[u] \arrow[d] & \bullet \arrow[u] \arrow[d] & V \arrow[l] \arrow[r] \arrow[u] \arrow[d] & \bullet \arrow[u] \arrow[d] \\
D & K \arrow[l] \arrow[r] & E & L \arrow[l] \arrow[r] & F \\
\bullet \arrow[u] & Z \arrow[l] \arrow[r] \arrow[u] & \bullet \arrow[u] & W \arrow[l] \arrow[r] \arrow[u] & \bullet \arrow[u]
\end{tikzcd}$$

there is an obvious comparison map

$$(G^X_{Y;A,B} || H^U_{V;B,C}) +_{Y \times V} (K^Y_{Z;D,E} || L^V_{W;E,F}) \to (G^X_{Y;A,B} +_Y K^Y_{Z;D,E}) || (H^U_{V;B,C} +_V L^V_{W;E,F}),$$

satisfying appropriate (lax monoidal) coherence equations, which however is *not in general* an isomorphism. This reflects the fact that the left-hand expression involves more synchronization than the right.

Example 2.2.8. Consider the systems

$$\begin{tikzcd}
0 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[d] \\
1 & G \arrow[l] \arrow[r] & 1 & H \arrow[l] \arrow[r] & 1 \\
0 \arrow[u] \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 1 \arrow[u] \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 0 \arrow[u] \arrow[d] \\
1 & K \arrow[l] \arrow[r] & 1 & L \arrow[l] \arrow[r] & 1 \\
0 \arrow[u] & 0 \arrow[l] \arrow[r] \arrow[u] & 0 \arrow[u] & 0 \arrow[l] \arrow[r] \arrow[u] & 0 \arrow[u]
\end{tikzcd}$$

where all the arrows are obvious. If we calculate first the pullbacks and then the pushout we obtain:

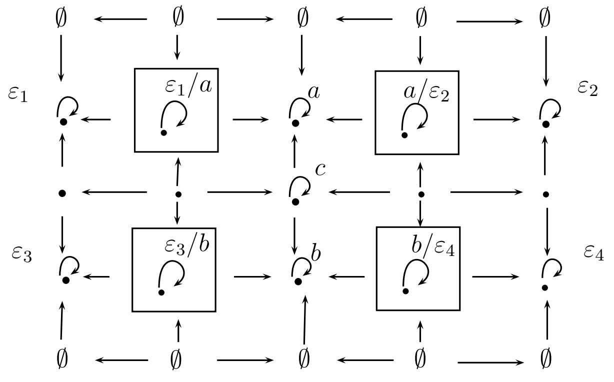
$$\begin{tikzcd}
0 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[d] \\
1 & G\times H \arrow[l] \arrow[r] & 1 \\
0 \arrow[u] \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 0 \arrow[u] \arrow[d] \\
1 & K\times L \arrow[l] \arrow[r] & 1 \\
0 \arrow[u] & 0 \arrow[l] \arrow[r] \arrow[u] & 0 \arrow[u]
\end{tikzcd}
=
\begin{tikzcd}
0 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[d] \\
1 & (G\times H)+(K\times L) \arrow[l] \arrow[r] & 1 \\
0 \arrow[u] & 0 \arrow[l] \arrow[r] \arrow[u] & 0 \arrow[u]
\end{tikzcd}$$

On the other hand, if we calculate first sequential compositions we have:

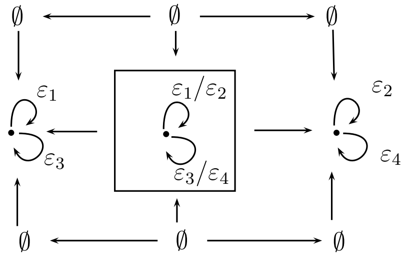
$$\begin{tikzcd}
0 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[d] \\
1 & G+K \arrow[l] \arrow[r] & 1 & H+L \arrow[l] \arrow[r] & 1 \\
0 \arrow[u] & 0 \arrow[l] \arrow[r] \arrow[u] & 0 \arrow[u] & 0 \arrow[l] \arrow[r] \arrow[u] & 0 \arrow[u]
\end{tikzcd}
=
\begin{tikzcd}
0 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[d] \\
1 & (G+K)\times(H+L) \arrow[l] \arrow[r] & 1 \\
0 \arrow[u] & 0 \arrow[l] \arrow[r] \arrow[u] & 0 \arrow[u]
\end{tikzcd}$$

The comparison map $(G \times H) + (K \times L) \to (G + K) \times (H + L)$ is not an isomorphism.

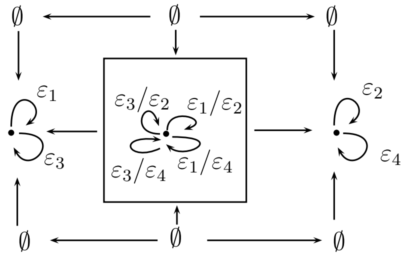
Example 2.2.9. Consider the following systems:



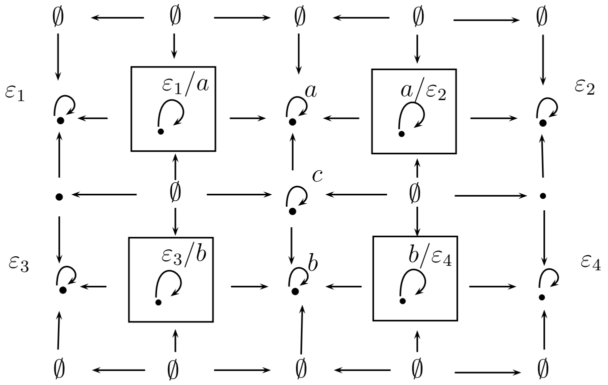
If we calculate first the parallel compositions and then the pushout we obtain:



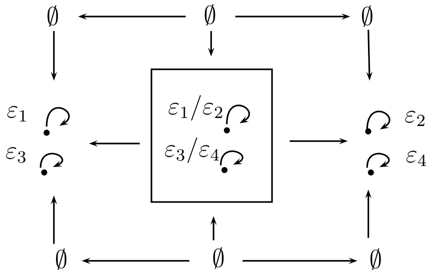
Calculating first the pushouts the composite is:



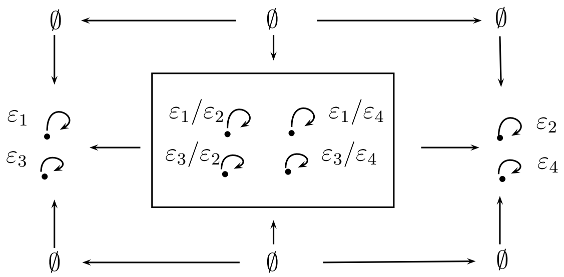
Example 2.2.10. Consider now the systems:



Calculating first the parallel compositions we have:



With the second calculation we obtain:



Definition 2.2.11. *The* product *(or* parallel composition without communication*)* $G^X_{Y;A,B} \times H^Z_{W;C,D}$ *of two systems* $G^X_{Y;A,B}$, $H^Z_{W;C,D}$ *is formed by taking the product of all the objects and arrows in* G *with the corresponding objects and arrows in the* H*; briefly*

$$\begin{tikzcd}
\bullet\times\bullet & X\times Z \arrow[l] \arrow[r] \arrow[d] & \bullet\times\bullet \arrow[d] \\
A\times C \arrow[from=1-1, d] & G\times H \arrow[l] \arrow[r] & B\times D \\
\bullet\times\bullet \arrow[u] & Y\times W \arrow[l] \arrow[r] \arrow[u] & \bullet\times\bullet \arrow[u]
\end{tikzcd}$$

Definition 2.2.12. *The* sum $G^X_{Y;A,B} + H^Z_{W;C,D}$ *of two systems* $G^X_{Y;A,B}$, $H^Z_{W;C,D}$ *is formed by taking the sum of all the objects and arrows in* G *with the corresponding objects and arrows in the* H*; briefly*

$$\begin{tikzcd}
\bullet+\bullet & X+Z \arrow[l] \arrow[r] \arrow[d] & \bullet+\bullet \arrow[d] \\
A+C \arrow[from=1-1, d] & G+H \arrow[l] \arrow[r] & B+D \\
\bullet+\bullet \arrow[u] & Y+W \arrow[l] \arrow[r] \arrow[u] & \bullet+\bullet \arrow[u]
\end{tikzcd}$$

The last part of the algebra of systems consists of a number of constants.

Definition 2.2.13. *The constants of the algebra are systems constructed from the constants of the distributive category structure of Sets regarded as discrete graphs ([68],[47],[26]).*

When we describe in a later section a programming language there will be of course also as constants the operations of data types; the particular language we describe has the natural numbers together with predecessor and successor.

2.2.4 The wscc structure on span and cospan categories

As seen above, a system may be regarded indifferently as a cospan of spans of graphs or as a span of cospans of graphs. Thus, the category of systems is a subcategory[1] of the categories $Cospan(Span(Graph))$ and $Span(Cospan(Graph))$ and it has the same *well supported compact closed structures* (briefly, *wscc structures*) defined on span and cospan categories. In fact, it is well-known ([57, 58]) that the category $Cospan(\mathbf{E})$, where $\mathbf{E}$ is a category with finite colimits, is a wscc category, that is a symmetric monoidal category in which every object has a *separable algebra structure* ([12]). This means that for each object A of $\mathbf{E}$ there exist four arrows[2]

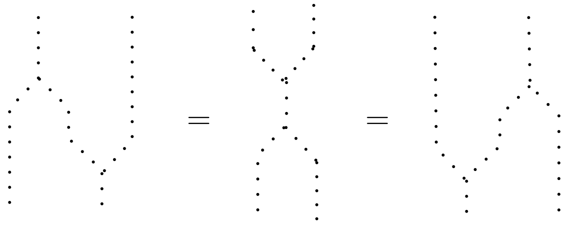
$$i_A : O \to A, \quad \nabla_A : A + A \to A, \quad i_A^o : A \to O, \quad \nabla_A^o : A \to A + A$$

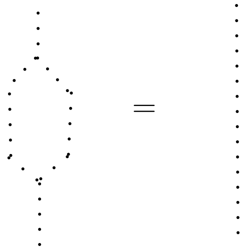
such that (A, ∇_A, i_A) is a commutative monoid, (A, ∇_A^o, i_A^o) is a cocommutative comonoid and the following axioms hold:

(i) $(1_A + \nabla_A) \circ (\nabla_A^o + 1_A) = \nabla_A^o \circ \nabla_A = (\nabla_A^o + 1_A) \circ (1_A + \nabla_A)$ (Frobenius equations),

(ii) $\nabla_A \circ \nabla_A^o = 1_A$.

The axioms are represented as[3]:





Also the dual category $Span(\mathbf{E})$, where $\mathbf{E}$ is a category with finite limits, is wscc. For each object A of $Span(\mathbf{E})$ there are four arrows[4]

$$p_A : I \to A, \quad \Delta_A : A \times A \to A, \quad p_A^o : A \to I, \quad \Delta_A^o : A \to A \times A$$

such that (A, Δ_A, p_A) is a commutative monoid, (A, Δ_A^o, p_A^o) is a cocommutative comonoid and the following axioms hold:

[1] For simplifying assumptions see section 2.3.

[2] These constants are the analogous of sequential constants defined for weighted automata (see p. 25).

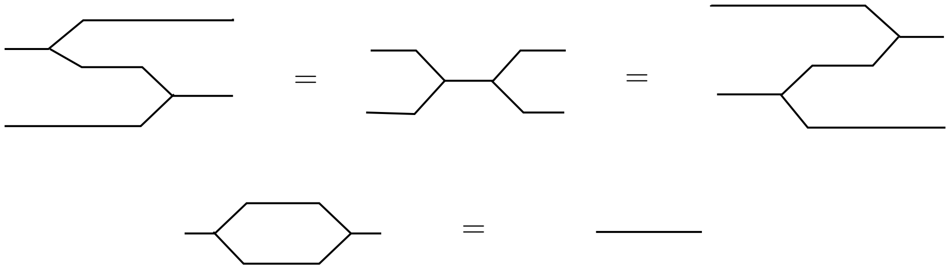
[3] See the graphical representation of sequential wires, p. 33.

[4] Compare these arrows with parallel constants defined on page p. 30.

(i) $(1_A \times \Delta_A)||(\Delta^o_A \times 1_A) = \Delta^o_A||\Delta_A = (\Delta^o_A \times 1_A)||(1_A \times \Delta_A)$ (Frobenius equations),

(ii) $\Delta_A||\Delta^o_A = 1_A$.

Graphically:



2.3 Simplifying Assumptions

We introduce a number of simplifying assumptions with the aim of arriving at a implementable programming language for systems. As we do this we will be considering also certain important derived operations of the algebra.

2.3.1 Simplifying the interfaces

Assumption 1. *We assume from now on that in a system with sequential and parallel interfaces G as described above the corner graphs $G^{\leftarrow\uparrow}$, $G^{\rightarrow\uparrow}$, $G^{\leftarrow\downarrow}$, $G^{\rightarrow\downarrow}$ each have* one state and no transitions*, that the graphs A, B each have* one state*, and that the graphs X, Y have* no transitions*.*

The idea is that in many cases the sequential interface consists only of states with no transitions, whereas the parallel interfaces are "stateless", that is, consist of transitions and one state. The assumptions are appropriate for message passing communication but not for systems in which there is communication by shared variables, since this requires that the parallel interfaces have state. It is not difficult to make assumptions for this type of communication but we prefer here to make the simpler assumption.

Given the assumption we may ignore the corner graphs of a system so that it consists of five graphs G, A, B, X, Y and the four graph morphisms

$$\begin{tikzcd}
& X \arrow[d, "\gamma_0"'] & \\
A & G \arrow[l, "\partial_0"'] \arrow[r, "\partial_1"] & B \\
& Y \arrow[u, "\gamma_1"] &
\end{tikzcd}.$$

Since the single states of A and B need not have a name, we may sometimes confuse A and B with $transitions(A)$ and $transitions(B)$ respectively. We may think of A, B, X, Y as sets, and of A and B as labels for the transitions in G (the graph morphisms ∂_0, ∂_1 providing the labelling).

As a consequence of the simple form of the corner graphs of a system we have the following result.

Proposition 2.3.1. *The* communicating parallel composite $G^X_{Y;A,B} \times_B H^Z_{W;B,C}$ *(briefly $G^X_{Y;A,B}||H^Z_{W;B,C}$) of two systems has top sequential interface $X \times Z$, and bottom sequential interface $Y \times W$; we can summarize this by the formula $G^X_{Y;A,B}||H^Z_{W;B,C} = (G||H)^{X\times Z}_{Y\times W;A,C}$.*
The sequential composite $G^X_{Y;A,B} +_Y H^Y_{Z;C,D}$ *has left parallel interface the graph with one vertex and transitions $transitions(A) + transitions(C)$ which we denote with some* abuse of notation *as $A + C$, and similarly right parallel interface $B + D$; we can summarize this by the formula $G^X_{Y;A,B} +_Y H^Y_{Z;C,D} = (G +_Y H)^X_{Z;(A+C),(B+D)}$.*
Trivially, $G^X_{Y;A,B} \times H^Z_{W;C,D} = (G \times H)^{X\times Z}_{Y\times W;A\times C,B\times D}$.

Notice that the class of systems we are considering is closed under sequential and communicating parallel composition and product, but is *not* closed under the operation of sum since the resulting system will have parallel interfaces with two states, not one.

We now introduce two derived operations similar to the sequential composite and the sum, but which are local in the sense that the parallel interfaces are fixed. Intuitively they are sequential operations within a fixed parallel protocol.

Definition 2.3.2. *The* local sequential composition $G^X_{Y;A,B} +_{Y,loc} H^Y_{Z;A,B}$ *(or more briefly $G^X_{Y;A,B} \bullet H^Y_{Z;A,B}$) of two systems $G^X_{Y;A,B}$, $H^Y_{Z;A,B}$, is formed from*

$G^X_{Y;A,B} +_Y H^Y_{Z;A,B}$ *by composing with appropriate codiagonals as follows:*

$$\begin{tikzcd}
\bullet \arrow[d] & \bullet \arrow[l, "1"'] \arrow[r, "1"] \arrow[d] & \bullet \arrow[d] & X \arrow[l] \arrow[r] \arrow[d] & \bullet \arrow[d] & \bullet \arrow[l, "1"'] \arrow[r, "1"] \arrow[d] & \bullet \arrow[d] \\
A & A+A \arrow[l, "\nabla"'] \arrow[r, "1"] & A+A & G+_Y H \arrow[l] \arrow[r] & B+B & B+B \arrow[l, "1"'] \arrow[r, "\nabla"] & B \\
\bullet \arrow[u] & \bullet \arrow[l, "1"'] \arrow[r, "1"] \arrow[u] & \bullet \arrow[u] & Z \arrow[l] \arrow[r] \arrow[u] & \bullet \arrow[u] & \bullet \arrow[l, "1"'] \arrow[r, "1"] \arrow[u] & \bullet \arrow[u]
\end{tikzcd}$$

where the codiagonals $\nabla : A + A \to A, \nabla : B + B \to B$ *are codiagonals on transitions, but the identity on the single state.*

Definition 2.3.3. *The* local sum $G^X_{Y;A,B} +_{loc} H^Z_{W;A,B}$ *of two systems* $G^X_{Y;A,B}$, $H^Z_{W;C,D}$, *is formed from* $G^X_{Y;A,B} + H^Z_{W;C,D}$ *by composing with appropriate codiagonals as follows:*

$$\begin{tikzcd}
\bullet \arrow[d] & \bullet+\bullet \arrow[l, "\nabla"'] \arrow[r, "1"] \arrow[d] & \bullet+\bullet \arrow[d] & X+Z \arrow[l] \arrow[r] \arrow[d] & \bullet+\bullet \arrow[d] & \bullet+\bullet \arrow[l, "1"'] \arrow[r, "\nabla"] \arrow[d] & \bullet \arrow[d] \\
A & A+A \arrow[l, "\nabla"'] \arrow[r, "1"] & A+A & G+H \arrow[l] \arrow[r] & B+B & B+B \arrow[l, "1"'] \arrow[r, "\nabla"] & B \\
\bullet \arrow[u] & \bullet+\bullet \arrow[l, "\nabla"'] \arrow[r, "1"] \arrow[u] & \bullet+\bullet \arrow[u] & Y+W \arrow[l] \arrow[r] \arrow[u] & \bullet+\bullet \arrow[u] & \bullet+\bullet \arrow[l, "1"'] \arrow[r, "\nabla"] \arrow[u] & \bullet \arrow[u]
\end{tikzcd}$$

Clearly, $G^X_{Y;A,B} +_{loc} H^Z_{W;A,B} = (G +_{loc} H)^{X+Z}_{Y+W;A,B}$.

Now the class of systems we are now considering is closed under the operations of parallel and sequential composition, product, local sequential composition and local sum.

2.3.2 Finiteness assumptions

In general, pushouts and pullbacks of infinite graphs are not implementable. We need to make some finiteness assumptions.

Assumption 2. *We assume that in system* $G^A_{B;X,Y}$ *that* A *and* B *have a* finite number of transitions.

This assumption means that the pullbacks in the parallel composition are implementable. A further consequence of this assumption is that the transitions of the graph G decompose as a disjoint union

$$transitions(G) = \biguplus_{a \in A, b \in B} transitions(G)_{a,b}$$

where $transitions(G)_{a,b}$ is the set of transitions labelled by $a \in A, b \in B$. Denote by $G_{a,b}$ the graph with the same states as G but with transitions $transitions(G)_{a,b}$.

The next assumption will have the effect that our systems have a finite state automata as control structure. Usually finite state automata are presented as recognizers of regular languages [35]. However the original work of McCulloch and Pitts [50] introduced automata as systems with thresholds, that is systems with infinite state spaces which decomposed into finite sums. Our finiteness assumptions are of this nature.

Assumption 3. *We assume that the set of states of the graphs G, $G^{\uparrow}$, $G_{\downarrow}$ are given as a finite disjoint sums*: $states(G) = U_1 + U_2 + \cdots + U_m$, $states(G^{\uparrow}) = X_1 + X_2 + \cdots + X_k$, $states(G_{\downarrow}) = Y_1 + Y_2 + \cdots + Y_l$.

The first effect of this is that each of the graphs $G_{a,b}$ $(a \in A, b \in B)$ breaks up as a matrix of *spans of sets.*

To see this notice that a graph G is just an endomorphism

$$source, target : transitions(G) \to states(G)$$

in $Span(Sets)$. Further the category $Span(Sets)$ has direct sums, the direct sum of U and V being $U+V$ with injections the functions $i_U : U \to U+V$, $i_V : V \to U+V$ considered as spans, and projections the same functions but now considered as the opposite spans $i_U^{op} : U+V \to U$, $i_V^{op} : U+V \to V$. The commutative monoid structure on $Span(Sets)(U,V)$ is given by sum and the empty span. Since a graph is just an endomorphism in $Span(Sets)$ a graph G whose state set is $U+V$ may be represented as a 2×2 matrix of spans $\begin{pmatrix} G_{U,U} & G_{U,V} \\ G_{V,U} & G_{V,V} \end{pmatrix}$ where for example $G_{U,V} = i_V^{op} G i_U$. Further

$$G = i_U G_{U,U} i_U^{op} + i_V G_{U,V} i_U^{op} + i_U G_{V,U} i_V^{op} + i_V G_{V,V} i_V^{op}.$$

Generalizing this to the case in which the states break up into a disjoint sum of n subsets Assumption 3 implies that each of the graphs $G_{a,b}$ may be represented as a $k \times k$ matrix of spans, the i,jth entry of which we will denote G_{a,b,U_i,U_j}, or even $G_{a,b,i,j}$. It has a simple meaning: G_{a,b,U_i,U_j} *is the set of transitions of G labelled a, b whose sources lie in U_i and whose targets lie in U_j. The projections of the span G_{a,b,U_i,U_j} are the projections onto the sources and targets.*

It is easy also to expand the matrix to include the functions $\gamma_0 : X \to G$, $\gamma_1 : X \to G$. The resulting matrix has columns indexed by $X_1, X_2, \cdots, X_k$, $U_1, U_2, \cdots, U_l$ and rows indexed by $Y_1, Y_2, \cdots, Y_l$, $U_1, U_2, \cdots, U_m$. As an example when $k = l = m = 2$ the matrix has the form

$$\begin{array}{c||cc|cc|}
G_{a,b} & X_1 & X_2 & U_1 & U_2 \\
\hline\hline
Y_1 & 0 & 0 & G_{a,b,U_1,Y_1} & G_{a,b,U_2,Y_1} \\
Y_2 & 0 & 0 & G_{a,b,U_1,Y_2} & G_{a,b,U_2,Y_2} \\
\hline
U_1 & G_{a,b,X_1,U_1} & G_{a,b,X_2,U_1} & G_{a,b,U_1,U_1} & G_{a,b,U_2,U_1} \\
U_2 & G_{a,b,X_1,U_1} & G_{a,b,X_2,U_2} & G_{a,b,U_1,U_2} & G_{a,b,U_2,U_2} \\
\hline
\end{array}$$

where 0 denotes the empty span.

Example 2.3.4. The predecessor function $\mathit{pred} : N \to N + 1$ which returns an error if the argument is 0 but otherwise decrements, may be considered as a system with trivial parallel interfaces, top sequential interface N bottom sequential interface $N + 1$ and central graph having states $N + N + 1$, transitions N and $\mathit{source} : N \to N + (N + 1) = \mathit{inj}_N$, $\mathit{target} : N \to N + (N + 1) = \mathit{inj}_{(N+1)}$. (This is the usual picture of a function as a graph on the disjoint union of the domain and codomain, with edges relating domain elements and their images,) We call this system *pred*. The matrix is

$$\begin{array}{c||c|ccc|}
\mathit{pred} & N & N & N & 1 \\
\hline\hline
N & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
\hline
N & 1 & 0 & 0 & 0 \\
N & 0 & \mathit{pred}_{N,N} & 0 & 0 \\
1 & 0 & \mathit{pred}_{N,1} & 0 & 0 \\
\hline
\end{array}$$

where 0 denotes the empty span and 1 denotes the identity span. The span $\mathit{pred}_{N,1}$ is the partial function which returns error on zero, and the span $\mathit{pred}_{N,N}$ is the partial function returning $n - 1$ for $n > 0$.

We describe next a derived operation which is a minor modification of the parallel composition, in order to simplify the matrix version of the parallel composition. The mathematical fact behind the derived operation is this: in a symmetric monoidal category with direct sums, in which the tensor product distributes over the direct sums, if two arrows are represented as matrices, then via distributivity isomorphisms the matrix of the tensor product of two arrows is a tensor product of the matrices of the arrows. The precise distributivity isomorphism needs to be specified since there are many possible, resulting in different ordering of the rows and columns of the tensor product matrix.

Definition 2.3.5. Distributed parallel.

Given systems $G^{X_1+\cdots+X_k}_{Y_1+\cdots+Y_l;A,B}$, $H^{Z_1+\cdots+Z_{k'}}_{W_1+\cdots+W_{l'};B,C}$ *the communicating parallel composite* $G\times_B H$ *(briefly* $G||H$*) has left interface* A*, right interface* C*, top interface* $(X_1+\cdots+X_k)\times(Z_1+\cdots+Z_{k'})$ *and bottom interface* $(Y_1+\cdots+Y_l)\times(W_1+\cdots+W_{l'})$*. Composing on the top and bottom interfaces with distributivity isomorphisms we obtain a system with left interface* A*, right interface* C*, top interface* $X_1\times Z_1+X_2\times Z_1+\cdots+X_k\times Z_{k'}$ *and bottom interface* $Y_1\times W_1+Y_2\times W_1+\cdots+Y_l\times W_{l'}$ *The set of states of* $G||H$ *may similarly be distributed to have the form* $U_1\times V_1+\cdots+U_m\times V_{m'}$*. We will, with an abuse of notation, denote this resulting system also as* $G||H$*.*

Definition 2.3.6. Distributed product.

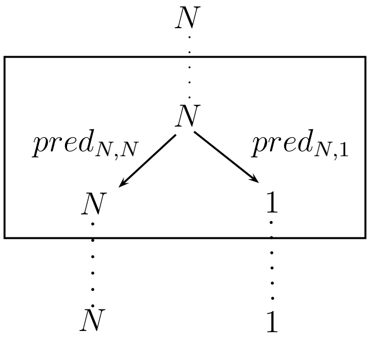
Given systems $G^{X_1+\cdots+X_k}_{Y_1+\cdots+Y_l;A,B}$, $H^{Z_1+\cdots+Z_{k'}}_{W_1+\cdots+W_{l'};C,D}$ *the product* $G\times H$ *has left interface* $A\times C$*, right interface* $B\times D$*, top interface* $(X_1+\cdots+X_k)\times(Z_1+\cdots+Z_{k'})$ *and bottom interface* $(Y_1+\cdots+Y_l)\times(W_1+\cdots+W_{l'})$*. Composing on the top and bottom interfaces with distributivity isomorphisms we obtain a system with left interface* $A\times C$*, right interface* $B\times D$*, top interface* $X_1\times Z_1+X_2\times Z_1+\cdots+X_k\times Z_{k'}$ *and bottom interface* $Y_1\times W_1+Y_2\times W_1+\cdots+Y_l\times W_{l'}$ *The set of states of* $G\times H$ *may similarly be distributed to have the form* $U_1\times V_1+\cdots+U_m\times V_{m'}$*. We will, with an abuse of notation, denote this resulting system also as* $G\times H$*.*

The last assumption we make has the consequence that the pushout in sequential composition is done a the level of control, not of data, and is therefore implementable.

Assumption 4. *We assume that in the matrix of the system* $G^A_{B;X,Y}$ *that the entries involving the sequential interfaces are either the identity span* 1 *or the empty span* 0*.*

Automaton representation

Of course the matrix for $G^X_{Y,;A,B}$ has a geometric representation as *a labelled automaton*, with top sequential interfaces $X_1,X_2,\cdots,X_k$, bottom sequential interfaces $Y_1,Y_2,\cdots,Y_l$, and vertices which are labelled by the sets U_i and for each $a\in A, b\in B$ edges from U_i to U_j labelled G_{a,b,U_i,U_j}. As usual we will omit edges labelled with empty spans. This representation has advantages both technical and conceptual, but is less easy to typeset. We give one example, namely the automaton representation of the predecessor system described above, which however has trivial parallel interfaces. We will see further examples in section 2.5.



2.4 The programming language Cospan-Span

The idea of this section is to restate the notion of system we have developed, and describe the operations on systems. The reader should compare the notions described here with those described in [41] where finite state systems were considered. We describe the programming language at the same time as its semantics. The programs are the expressions in the operations and constants; an execution of a program is a path in the graph described by the expression.

2.4.1 Systems

Definition 2.4.1. *A* system G *consists of*

(i) two finite sets A, B called the left and right parallel interfaces on G;

(ii) two families of possibly infinite sets $X = X_1, X_2, \cdots, X_k$ and $Y = Y_1, Y_2, \cdots, Y_l$ called the top and bottom sequential interfaces;

(iii) a family of possibly infinite sets $U = U_1, U_2, \cdots, U_m$ which together constitute the internal state space of G;

(iv) two functions $\varphi : \{1, 2, \cdots, k\} \to \{1, 2, \cdots m\}$ and $\psi : \{1, 2, \cdots, l\} \to \{1, 2, \cdots m\}$ called the inclusions of the sequential interfaces, with the properties that $X_i = U_{\varphi(i)}$ and $Y_i = U_{\psi(i)}$;

(v) a family of spans of sets $G_{a,b,i,j} : U_i \to U_j$ ($a \in A$, $b \in B$, $i \in \{1, 2, \cdots, m\}$, $j \in \{1, 2, \cdots, m\}$) which together constitute a family of graphs $G_{a,b}$ ($a \in A, b \in B$) each with vertex set $U_1 + U_2 + \cdots + U_m$.

The graph $G_{a,b}$ is the graph of transitions of the system when the signals a, b occur on the parallel interfaces. *We denote the system as $G^X_{Y;A,B}(U)$.*

It is easy to see that this is the essential concrete content of the notion of system developed in the previous section.

2.4.2 Operations on systems, and constants

In the following we denote families by giving a typical element.

Definition 2.4.2. *The* product *of two systems $G^X_{Y;A,B}(U)$, $H^Z_{W;C,D}(V)$, denoted $G{\times}H$, has left and right interfaces $A{\times}C$, $B{\times}D$, top interface $\{X_i{\times}Z_j\}$, bottom interface $\{Y_i \times W_j\}$, internal state space $\{U_i \times V_j\}$, inclusions of sequential interfaces $\varphi_{G\times H} = \varphi_G \times \varphi_H$ and $\psi_{G\times H} = \psi_G \times \psi_H$, and finally the spans*

$$(G \times H)_{(a,c),(b,d),(i_1,j_1),(i_2,j_2)} = G_{a,b,i_1,i_2} \times H_{c,d,j_1,j_2}.$$

Ignoring the sequential interfaces, the matrix of the distributed product is just the tensor product of the matrices of the components.

Definition 2.4.3. *The* communicating parallel composition *of two systems $G^X_{Y;A,B}(U)$, $H^Z_{W;B,C}(V)$, denoted $G||H$, has left and right interfaces A, C, top interface $\{X_i{\times}Z_j\}$, bottom interface $\{Y_i{\times}W_j\}$, internal state space $\{U_i{\times}V_j\}$, inclusions of sequential interfaces $\varphi_{G\times H} = \varphi_G \times \varphi_H$ and $\psi_{G\times H} = \psi_G \times \psi_H$, and finally the spans*

$$(G||H)_{a,c,(i_1,j_1),(i_2,j_2)} = \sum_b (G_{a,b,i_1,i_2} \times H_{b,c,j_1,j_2}).$$

Definition 2.4.4. *The* sequential composite *of two systems $G^X_{Y;A,B}(U)$, $H^Y_{Z;C,D}(V)$, denoted $G{+_Y}H$, has left and right interfaces $A{+}C$, $B{+}D$, top interface $\{X_i\}$, bottom interface $\{Z_i\}$, internal state space $(\{U_i\} + \{V_j\})/(U_{\psi_G(i)} \sim V_{\varphi_H(i)})$, inclusions of sequential interfaces φ_G and ψ_H, and finally the spans*

$$(G +_Y H)_{p,q,[W_i],[W_j]} = \sum_{U\in[W_i],U'\in[W_j]} G_{p,q,U,U'} + \sum_{V\in[W_i],V'\in[W_j]} H_{p,q,V,V'}$$

where $p \in A{+}C$, $q \in B{+}D$, $W, W' \in \{U_i\}{+}\{V_j\}$, $[W]$ denotes the equivalence class of W.

Definition 2.4.5. *The* local sequential *of two systems $G^X_{Y;A,B}(U)$, $H^Y_{Z;A,B}(V)$, denoted $G{+_Y}H$ (briefly $G \bullet H$), has left and right interfaces A, B, top interface $\{X_i\}$, bottom interface $\{Z_i\}$, internal state space $\{U_i\} + \{V_j\}/(U_{\psi_G(i)} \sim V_{\varphi_H(i)})$, inclusions of sequential interfaces φ_G and ψ_H, and finally the spans*

$$(G \bullet H)_{p,q,[W_i],[W_j]} = \sum_{U\in[W_i],U'\in[W_j]} G_{p,q,U,U'} + \sum_{V\in[W_i],V'\in[W_j]} H_{p,q,V,V'}$$

where $p \in A, q \in B$, $W, W' \in \{U_i\} + \{V_j\}$, and $[W]$ denotes the equivalence class of W.

Definition 2.4.6. *The* local sum *of two systems $G^X_{Y;A,B}(U)$, $H^Z_{W;A,B}(V)$, denoted $G +_{loc} H$, has left and right interfaces A, B, top interface $\{X_i\}+\{Z_j\}$, bottom interface $\{Y_i\} + \{W_j\}$, internal state space $(\{U_i\} + \{V_j\})$, inclusions of sequential interfaces $\varphi_G + \varphi_H$ and $\psi_G + \psi_H$, and finally the spans*

$$(G +_{loc} H)_{p,q,[U_i],[U_j]} = G_{p,q,U_i,U_j}$$

and

$$(G +_{loc} H)_{p,q,[V_i],[V_j]} = G_{p,q,V_i,V_j},$$

where $p \in A, q \in B$ and all remaining spans are empty.

2.4.3 Programs

In our view programming languages should be presented by first describing an algebra of systems. Then programs are elements of the free algebra of the same type, generated by some basic systems. The meaning of the program is then the evaluation in the concrete algebra. The programs of the Cospan-Span language are expressions in the operations and constants of the algebra described above, and the following basic systems: $pred^N_{N+1}$, $succ^{N+1}_N$ (defined similarly to $pred^N_{N+1}$). The evaluation of a program is a system; a behaviour is a path in the central graph of the system.

2.5 Concluding remarks

If one examines the previous investigations in this project it will be clear that many matters discussed at the level of finite state control may now be lifted to include also data.

2.5.1 Turing completeness

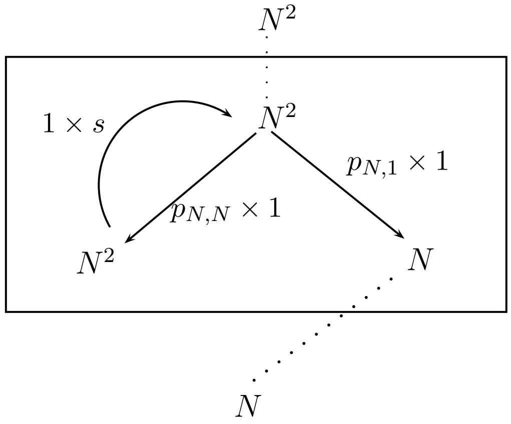
It is not difficult to relate the Elgot automata introduced in [68], [38],[42] to the algebra of cospans of graphs. It was shown in [60] that Elgot automata based on the elementary operations of predecessor and successor for natural numbers are Turing complete, and hence also the algebra described in this chapter. We give an example which illustrates sequential programming in Cospan-Span. All the systems in the following have trivial parallel interfaces. We use the following constants definable from distributive category operations, considered as systems with trivial parallel interfaces in which the central graph has no transitions (in which case a system reduces to a cospan of sets):

- $\eta_X = 0 \to X \xleftarrow{\nabla} X + X$,
- $\varepsilon_X = X + X \xrightarrow{\nabla} X \leftarrow 0$,
- $\nabla_X = X + X \xrightarrow{\nabla} X \leftarrow X$,
- $1_X = X \xrightarrow{1} X \xleftarrow{1} X$.

Example 2.5.1. The following is a program which, commencing in a state of the top sequential interface, computes addition of two natural numbers, terminating in the lower interface:

$$(\eta_{N^2} + 1_{N^2}) \bullet (1_{N^2} + \nabla) \bullet (1 + p \times 1_N) \bullet (1_{N^2} + 1_N \times s + 1_N) \bullet (\varepsilon_{N^2} + 1_N).$$

The system described by the program is:



where $p_{N,1}, p_{N,N}$ are the partial functions arising from the predecessor function

$$\begin{aligned} p: \ & N \to 1 + N \\ & n \mapsto p(n) := \begin{cases} \star & \text{if } n = 0 \\ n - 1 & \text{if } n > 0 \end{cases} \end{aligned}$$

and

$$\begin{aligned} s: \ & N \to N, \\ & n \mapsto s(n) := n + 1 \end{aligned}$$

is the successor function.

For instance, if the initial state is $(2, 3)$ the program computes

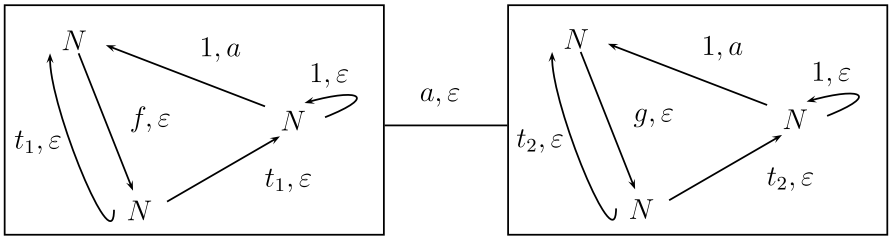
$$(2,3) \xrightarrow{p_{N,N} \times 1} (1,3) \xrightarrow{1 \times s} (1,4) \xrightarrow{p_{N,N} \times 1} (0,4) \xrightarrow{1 \times s} (0,5) \xrightarrow{p_{N,1} \times 1} (\star, 5) \cong 5$$

and returns 5.

2.5.2 Problems of concurrency

In [41],[42], [40] the authors explain how classical problems of concurrency may be modelled in $Span(Graph)$ algebra, at the level of finite state abstraction, which is the appropriate level for controlling many properties. In this chapter we show that these descriptions may be extended to include also operations on the data types.

We give a simple example of a parallel composite of two systems P and Q. P has trivial left interface, and right interface $\{\varepsilon, a\}$ whereas Q has trivial right interface and left interface $\{\varepsilon, a\}$. The combined system may be represented by the diagram (analogous to those [40]), in which the first part of a label is the span of sets, and the second part is the label on the parallel interface. The left system is P and the right Q.



The system P repeatedly applies f and then a test t_1 until the test results false, and then P may idle, eventually (in the Italian sense) synchronizing with Q on the signal a. After this P repeats the whole sequence. Q does the same, but with a different function g and a different test t_2, and seeks to synchronize with P.

Each of P and Q may be described by a Cospan-Span program in a similar way to the addition program above.

2.5.3 Hierarchy

There is an obvious relevance to hierarchical systems of the fact that systems in this algebra may be constructed by repeated parallel and sequential operations, with analogies to state charts. For an example of hierarchical system see section 3.3.

2.5.4 Change of geometry

In [42] the change of geometry is described using sequential operations on parallel systems. However in that paper only the local sequential composition is considered, whereas in this chapter we have a general sequential operation, which allows change of geometry with a change of parallel protocol.

Chapter 3

Hierarchical and mobile systems

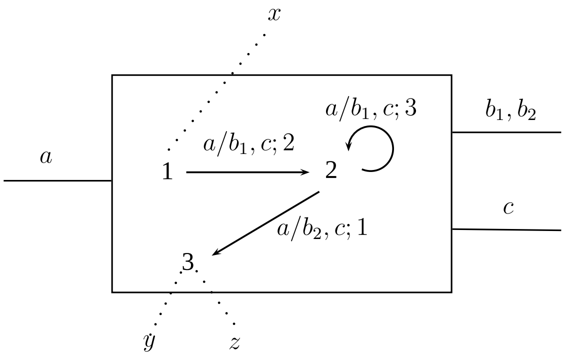
In this chapter we explain with examples how the algebra of Markov automata with both parallel and sequential interfaces can be used to describe hierarchical systems and ones with evolving geometry.

The plan of the chapter is as follows. In the first section we show that any automaton may be described as a sequential expression of automata with two states and a single transition. Then, we describe a variant of Dining Philosophers system called Sofia's birthday party, originally introduced in [42]. In the third section we describe a Fork Bomb. Finally, we describe in our model the finite probabilistic automata of Segala and Lynch [61, 49] and their behaviours.

3.1 Any automaton is in $\Sigma(E)$

Given a set S of automata, we denote with $\Pi(S)$ the set of automata which can be described as parallel composites of elements in S, and we denote with $\Sigma(S)$ the set of automata which can be described as sequential composites of elements in S.

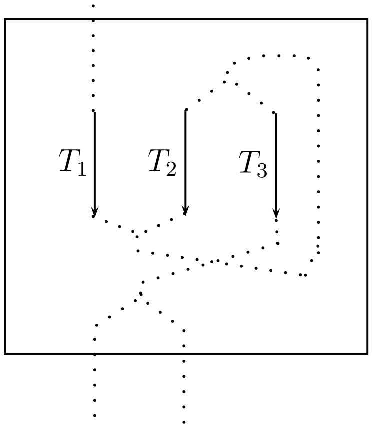
Let E be the set of automata with two states and a single transition. Then any automaton is an element of $\Sigma(E)$. To see this consider the following Markov automata:



Let T_1, T_2, T_3 be three single transition as follows:

- T_1 has the single transition labelled on the left by a and the right by (b_1, c) with weight 2,
- T_2 has the single transition labelled on the left by a and the right by (b_1, c) with weight 3,
- T_3 has the single transition labelled on the left by a and the right by (b_2, c) with weight 1,

then the following diagram shows how the automaton may be given as $T_1+T_2+T_3$ composed sequentially with sequential wires:



Remark 3.1.1. It is clear from this result that the pictures of automata are actually a special case of our designs for expressions of automata ([57]).

Example 3.1.2. The Dining Philosopher system described in section 1.4 is in $\Pi(\Sigma(E))$, that is a communicating parallel system.

3.2 Sofia's birthday party

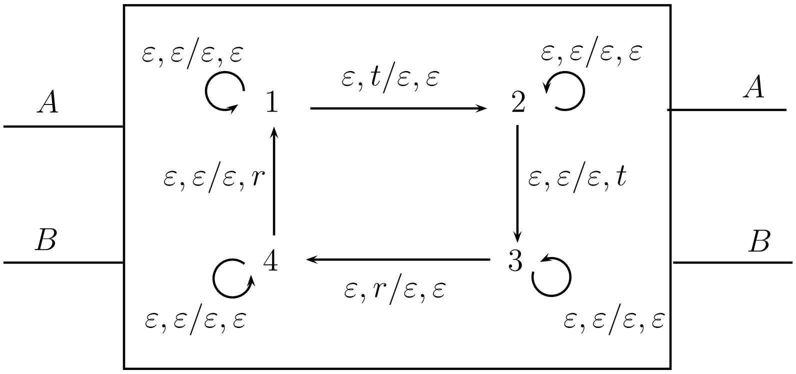
The example we would like to describe is a variant of the Dining Philosopher Problem which we call *Sofia's birthday party.* Instead of a circle of philosophers around a table with as many forks, we consider a circle of seats around a table separated by forks on the table. Then there are a number of children (not greater than the number of seats). The protocol of each child is the same as that of a philosopher. However in addition, if a child is not holding a fork, and the seat to the right is empty, the child may change seats - the food may be better there.

To simplify the problem we will assume that all transitions have weight 0 *or* 1, *so the transitions of components we mention will all have weight* 1.

To describe this we need six automata – a child $\mathcal{C}$, an empty seat $\mathcal{E}$, a fork $\mathcal{F}$, two transition elements $\mathcal{L}$ and $\mathcal{R}$, and the identity 1_A of A (a wire). The interface sets involved are $A = \{x, \varepsilon\}$ and $B = \{\varepsilon, t, r\}$.

The transition elements have left and right interfaces $A \times B$. The graph of the of the transition element $\mathcal{L}$ has two vertices p and q and one labelled edges $x, \varepsilon/\varepsilon, \varepsilon : q \to p$. Its top interface is $Q = \{q\}$, and its bottom interface is $P = \{p\}$. The graph of the transition element $\mathcal{R}$ also has two vertices p and q, and has one labelled edges $\varepsilon, \varepsilon/x, \varepsilon : q \to p$. Its top interface is also $Q = \{q\}$, and its bottom interface is $P = \{p\}$. The empty seat $\mathcal{E}$ has left and right interfaces $A \times B$. The graph of the empty seat has one vertex e and one labelled edge $\varepsilon, \varepsilon/\varepsilon, \varepsilon : e \to e$. Its top interface is P and its bottom interface is Q. The functions γ_0, γ_1 are uniquely defined.

The child $\mathcal{C}$ has labelled graph as follows:



$\varepsilon, \varepsilon/\varepsilon, \varepsilon$
$\varepsilon, t/\varepsilon, \varepsilon$
$\varepsilon, \varepsilon/\varepsilon, \varepsilon$
A
1
2
A
$\varepsilon, \varepsilon/\varepsilon, r$
$\varepsilon, \varepsilon/\varepsilon, t$
B
4
3
B
$\varepsilon, r/\varepsilon, \varepsilon$
$\varepsilon, \varepsilon/\varepsilon, \varepsilon$
$\varepsilon, \varepsilon/\varepsilon, \varepsilon$

The states have the following interpretation: in state 1 the child has no forks; in state 2 it has a fork in its left hand; in state 3 it has both forks (and can eat); in state 4 it has returned it left fork. The child's top interface is P and its bottom interface is Q. The function γ_0 takes p to 1; the function γ_1 takes q to 1.

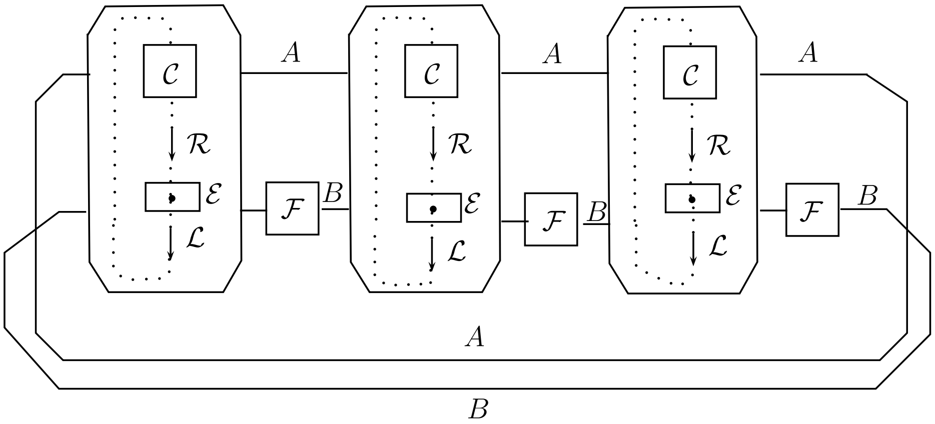
The fork $\mathcal{F}$ is as in the Dining Philosopher system (but with all transitions weighted 1).

Let $\mathcal{S} = \mathsf{Sfb}_P(C \bullet R \bullet E \bullet L)$. This automaton has the following interpretation – it can either be a child (on a seat) or an empty seat. The transition elements $\mathcal{R}$ and $\mathcal{L}$ allow the seat to become occupied or vacated. It is straightforward to see that this automaton is a positive weighted automaton.

Then Sofia's birthday party is given by the normalization of expression

$$\mathsf{Pfb}_{A\times B}(\mathcal{S}||(1_A \times \mathcal{F})||\mathcal{S}||(1_A \times \mathcal{F})||...||\mathcal{S}||(1_X \times \mathcal{F})).$$

This automaton has the behaviour as informally described above. Its diagrammatic representation is:



Notice that though Sofia's birthday party belongs to $\Pi\Sigma\Sigma(E)$ slight variations of the system belong instead to $\Pi\Sigma\Pi\Sigma(E)$, for example the system where more than one child may occupy a seat (communicating there). If the system starts in a state with as many children as seats - then movement is impossible and the system is equivalent to the Dining Philosophers.

Let us look at a particular case of Sofia's birthday party in more detail. Consider the system with three seats, and two children. There are 36 states reachable from initial state $(5, 1, 1, 1, 1, 1)$, where 5 is the state in which the seat is empty, and there are 141 transitions.

The states are

$$\begin{array}{llll}
(5,1,1,1,1,1) & (5,1,1,3,2,1) & (5,3,2,1,1,1) & (5,3,2,3,2,1) \\
(1,1,1,1,5,1) & (1,3,2,1,5,1) & (5,1,1,3,3,2) & (5,3,2,3,3,2) \\
(5,3,3,2,1,1) & (1,3,3,2,5,1) & (1,1,5,1,1,1) & (2,1,1,1,5,3) \\
(2,1,5,1,1,3) & (2,3,2,1,5,3) & (2,3,3,2,5,3) & (5,1,1,1,4,2) \\
(5,3,2,1,4,2) & (5,1,4,2,1,1) & (1,1,4,2,5,1) & (2,1,4,2,5,3) \\
(1,1,5,3,2,1) & (2,1,5,3,2,3) & (3,2,1,1,5,3) & (3,2,5,1,1,3) \\
(3,2,5,3,2,3) & (5,3,3,2,4,2) & (3,2,4,2,5,3) & (1,1,5,3,3,2) \\
(4,2,1,1,5,1) & (4,2,5,1,1,1) & (4,2,5,3,2,1) & (5,1,4,2,4,2) \\
(4,2,4,2,5,1) & (1,1,5,1,4,2) & (4,2,5,3,3,2) & (4,2,5,1,4,2).
\end{array}$$

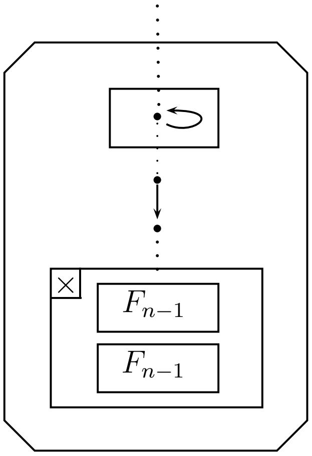
Then in 12 of these states there is a child eating: only one may eat at a time. It is straightforward to calculate the 36×36 Markov matrix and iterating show that the probability of a child eating after 1 step from initial state $(5,1,1,1,1,1)$ is 0, after 2 steps is $\frac{19}{60}$, after 3 steps is $\frac{98}{225}$, after 4 steps is $\frac{49133}{108000}$, after 5 steps is $\frac{1473023}{3240000}$ and after 100 steps is 0.3768058221.

3.3 A fork bomb

We describe in this section a system in $\Sigma(\Pi\Sigma)^n$, namely a probabilistic version of a fork bomb; that is, a process which may duplicate, and each of its descendants may duplicate also, hence leading to an exponential growth in the number of processes.

In order to consider a finite example, and to model the fact that resources are in practice limited, we consider a process F_n which, with equal probability, may decide to idle, or to produce two parallel processes F_{n-1}, each of which may, with equal probability, in turn choose to idle or produce two processes F_{n-2}, and so on. Processes F_0 simply idle.

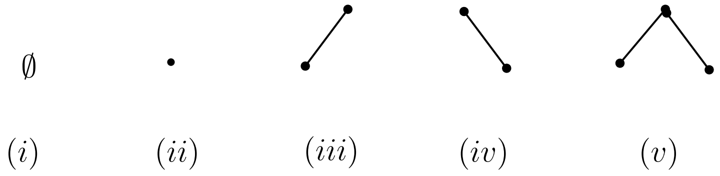
We would like to model such a system in our algebra, and to calculate the probability of arriving in the situation of having 2^n idling processes after k steps. The simplest version, in which there is no communication between the processes, and hence all parallel interfaces may be taken to be trivial, is shown the following picture, which may easily be converted into an expression in our algebra:



Of course the full expression for the system involves substituting for F_{n-1} and then F_{n-2} et cetera. Notice that if f_n is the number of states in F_n then f_n satisfies the recurrence relation

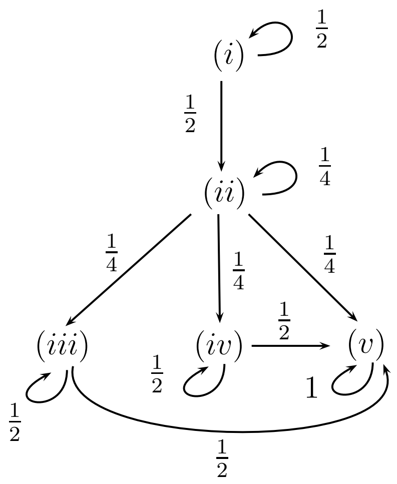
$$f_n = 1 + f_{n-1}^2, \quad f_0 = 1,$$

and hence $f_0 = 1$, $f_1 = 2$, $f_2 = 5$, $f_3 = 26$, $\cdots$. It is straightforward to see that f_n is the number of binary trees of depth at most n. (The recursive equation for the set B of all finite binary trees is $B \cong 1 + B \times B$; for an interesting observation of Lawvere on this equation see [48, 7].) In fact the states of F_n may be identified with such trees. Each state is a tuple of initial states of a number of processes, which have been created after a number of bifurcations. The trees encode the sequence of bifurcations which have occurred: vertices indicate bifurcations and edges indicate which of the two child process has bifurcated next. As an example, the trees of depth at most 2 are:



The initial state of F_2 (no bifurcations) corresponds to (i) the empty tree; the pair of initial states of $F_1 \times F_1$ corresponds to (ii) the one vertex tree (one bifurcation); the triple of initial states of $(F_0 \times F_0) \times F_1$ (after a bifurcation then a further bifurcation to the left) corresponds to the binary tree (iii); the triple of initial states of $F_1 \times (F_0 \times F_0)$ corresponds to the tree (iv); the quadruple of initial states of $(F_0 \times F_0) \times (F_0 \times F_0)$ corresponds to tree (v).

A calculation in our algebra of the states and transitions of F_2, with their probabilities, yields the following Markov automaton:



from which it is possible to calculate that the probability of reaching the state (v) in which there are 4 idling processes of type F_0 is greater than 99% after 12 steps.

3.4 The probabilistic automata of Segala and Lynch

We indicate how the finite probabilistic automata of Segala and Lynch [61, 49] (which we shall now refer to as Segala-Lynch automata) and their finite behaviours can be described in our model. Segala-Lynch automata model systems with non-deterministic choice of probabilistic actions. A behaviour consists of probabilistic transition of the non-determinism. The parallel operation is based on Hoare synchronization. Properties of interest are relative to a class of schedulers.

We show that Segala-Lynch automata are equivalent to a certain subclass of ours, and we then show that their behaviours (so-called finite probabilistic executions) are, in our model, the reachable part of a composition of a scheduling automaton and a weighted automaton. We show also that Hoare synchronization may be described by an expression in our algebra.

3.4.1 Segala-Lynch automata

A finite Segala-Lynch automaton [49] consists of a finite set Q of states (with a specified initial state q_0), a finite alphabet of actions A (divided into internal and external actions, though for simplicity we ignore the internal actions here), and a finite set of *probabilistic transitions*. By a probabilistic transition we mean a triple $(q, a, \sum_{q' \in Q} p_{q'} q')$, where $a \in A$, $q \in Q$, and $p_{q'}$ are non-negative real numbers with $\sum_{q' \in Q} p_{q'} = 1$ (the sum $\sum_{q'} p_{q'} q'$ is a formal sum). The idea is that the probabilistic transition is labelled a and goes from q to a distribution of states instead of a single state. We write such a transition also as $q \xrightarrow{a} \sum_{q' \in Q} p_{q'} q'$.

Given such a Segala-Lynch automaton we construct a weighted automaton as follows: state set Q, top sequential interface q_0, bottom sequential interface $\emptyset$, left parallel interface $A \cup \{\varepsilon\}$ and right parallel interface T the set of (names of) probabilistic transitions of the Segala-Lynch automaton augmented by the symbol ε. The set T contains the information necessary to schedule the Segala-Lynch automaton. Finally the matrices of the weighted automaton are defined as follows: if $t = (q, a, \sum_{q' \in Q} p_{q'} q')$ then $[\mathsf{Q}_{a,t}]_{q,q'} = p_{q'}$ all other values being 0; further $\mathsf{Q}_{\varepsilon,\varepsilon}$ is the identity matrix and all other matrices are zero.

We call the resulting weighted automaton an SL automaton, and it is clear that from the SL automaton one may recover the Segala-Lynch automaton, since the non-zero matrices of the SL automaton are exactly the transition matrices of single probabilistic transitions.

3.4.2 Behaviour of Segala-Lynch automata

The type of scheduler that Segala and Lynch have in mind is one which at any moment remembers the previous states and actions of the scheduling, but not which probabilistic transitions have occurred, and on that basis makes a probabilistic choice of which of the probabilistic transaction enabled in the current state to schedule next.

We will describe this in terms of SL automata and our algebra.

For simplicity we will describe first a more general type of scheduler, one which also remembers the probabilistic transitions carried out. Such a scheduler is a weighted automaton with the following properties:

(i) the left interface is T, the right interface is trivial;

(ii) the graph (of transitions with non-zero weighting) is a finite tree with root the initial state;

(iii) there are no ε labelled transitions,

(iv) out of any state there is at most one transition with non-zero weight with a given label $t \in T$.

That is, in each state the scheduler decides on a weight for the different probabilistic transitions to execute and passes to a new state which remembers which of the probabilistic transitions was chosen. It is not difficult, though a little messy, to modify this notion so that the scheduler remembers only the action taken rather than the whole probabilistic transition: equate states in the general scheduler which arise from the same action in a given state. Such a scheduler we call an SL scheduler. Now consider the parallel (in our sense) of a SL automaton with an SL scheduler. The normalized reachable part of this composite is easily seen to be a finite probabilistic execution of the Segala-Lynch automaton (and all probabilistic executions arise in this way). Normalization is necessary because the scheduler may attempt to schedule a probabilistic transition in a state in which the transition is not enabled.

3.4.3 The parallel composition of Segala-Lynch automata

We describe how CSP synchronization of weighted automata may be modelled in our algebra. Consider two positive weighted automata $\mathbf{Q}_{A+B,T}$ and $\mathbf{Q}'_{B+C,T'}$ with A, B and C pairwise disjoint, and with A and C containing ε's. To define the CSP parallel composition of $\mathbf{Q}$ and $\mathbf{Q}'$ we need a special component $\mathbf{M}_{A+B+C,(A+B)\times(B+C)}$ with one state and non-zero 1×1 matrices being $\mathsf{M}_{a,(a,\varepsilon)} = 1$, $\mathsf{M}_{c,(\varepsilon,c)} = 1$, $\mathsf{M}_{b,(b,b)} = 1$ and $\mathsf{M}_{\varepsilon,(\varepsilon,\varepsilon)} = 1$, where $a \in A, b \in B, c \in C$.

The CSP parallel composite is then

$$\mathbf{M}_{A+B+C,(A+B)\times(B+C)}||(\mathbf{Q}_{A+B,T} \times \mathbf{Q}'_{B+C,T'}).$$

When applied to the SL automata derived from Segala-Lynch automata it is routine to check that this gives the SL automaton of their parallel composite as Segala-Lynch automata.

Chapter 4

Conditional probability

It is well-known that the notion of conditional probability plays a prominent role in probability theory. The aim of this chapter is the introduction of a notion of conditional probability in the algebra of weighted and Markov automata. We prove results analogous to *Bayes' theorem* and *Total probability theorem* and we describe some famous problems of conditional probability in this algebra.

In our view, the occurrence in a system of an event with a certain probability associated, which we represent as a transition in a weighted automaton, induces a signal on the two parallel interfaces of the automaton. The observation of one of these signals on the right or on the left interface changes the probabilities of the system. Mathematically, the observation of a signal a on an parallel interface A is represented by the communicating parallel composition between a weighted automaton called *observer of a signal a on A* and the system.

A benefit of this approach is the clear distinction between states and transitions and the fact that a probability of an event is always associated to a specific state q into the state space of an automaton. Further, we show how many paradoxes of conditional probability arise from the fact that normalization is not compositional to sequential operations.

4.1 Classical probability theory

Classically (Kolmogorov axioms), given a probability space $(\Omega, \mathcal{F}, P)$, where Ω is a set of outcomes called *sample space*, $\mathcal{F}$ is a σ-algebra on Ω and P is a probability measure on $\mathcal{F}$, and given an event D of non-zero probability with respect to P, we define the *probability P of an event C given D*, denoted by

$P(C|D)$, with the formula

$$P(C|D) = \frac{P(C \cap D)}{P(D)}$$

where the event $C \cap D$ is the intersection of the events C and D. The most important fact about conditional probability is Bayes' formula given by the following

Theorem 4.1.1. *(Bayes' theorem)*
Let $(\Omega, \mathcal{F}, P)$ be probability space. Then, for each events C, D in Ω such that $P(C) \neq 0$ and $P(D) \neq 0$

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

Another important result known as *Total probability theorem* is the following

Theorem 4.1.2. *(Total probability theorem)*
Let $(\Omega, \mathcal{F}, P)$ be probability space. Then, for each event D in Ω, given a partition[1] *$\{C_i\}_{i \in I}$ of events in Ω with non-zero probability*

$$P(D) = \sum_{i \in I} P(D|C_i)P(C_i)$$

By Total probability theorem we obtain the following variant of Bayes' formula which is very useful in many concrete problems:

$$P(C_k|D) = \frac{P(D|C_k)P(C_k)}{\sum_{i \in I} P(D|C_i)P(C_i)}$$

where $k \in I$.

4.2 Conditional probability for weighted automata

To our purpose it is useful to generalize the definition of normalization of weighted positive automata given in the first chapter[2] to the case of weighted automata not necessarily positive. We give the following

[1] A partition of a sample space Ω is a family $\{C_i\}_{i \in I}$ of events in Ω such that $C_i \cap C_j = \emptyset$ for $i \neq j$ and $\cup_{i \in I} C_i = \Omega$.

[2] See p. 18.

Definition 4.2.1. *Given a* weighted automaton $\mathbf{Q}^X_{Y;A,B}$,*we define the normalized automaton* $\mathbf{N}(\mathbf{Q})$ *as the weighted automaton with the same interfaces and states, but with*

$$[\mathsf{N}(\mathsf{Q})_{a,b}]_{q,q'} = \begin{cases} \frac{[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'\in Q}[\mathcal{Q}]_{q,q'}} = \frac{[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'\in Q}\sum_{a\in A,b\in B}[\mathsf{Q}_{a,b}]_{q,q'}} & \textit{if } \sum_{q'}[\mathcal{Q}]_{q,q'} \neq 0 \\ 0 & \textit{otherwise} \end{cases}$$

where $\mathcal{Q}$ *is the total matrix*[3] *of* $\mathbf{Q}^X_{Y;A,B}$.

Definition 4.2.2. *Given a* weighted automaton $\mathbf{Q}^X_{Y;A,B}$, *we define the* probability of a signal $b \in B$ on the right interface B in a state $q \in Q$ *in the weighted automaton* $\mathbf{Q}^X_{Y;A,B}$, *denoted by* $P^{\mathbf{Q}}_q(b)$, *with the formula*

$$P^{\mathbf{Q}}_q(b) = \sum_{q'\in Q}\sum_{a\in A}[\mathsf{N}(\mathsf{Q})_{a,b}]_{q,q'}$$

In a similar way we define $P^{\mathbf{Q}}_q(a)$ *with* $a \in A$.

In other words, $P^{\mathbf{Q}}_q(b)$ is the normalized sum of all the weights of transitions out of q with right label b. It is clear from the definition that if there is at least one non-zero transition out of q then

$$P^{\mathbf{Q}}_q(b) = \frac{\sum_{q',a}[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q',a,b}[\mathsf{Q}_{a,b}]_{q,q'}}$$

Definition 4.2.3. *Given a* weighted automaton $\mathbf{Q}^X_{Y;A,B}$, *and a signal* $a \in A$, *we define* observer of the signal a on the interface A *as the weighted automaton* $\mathbf{O}_a$ *with state space* $\{0,1\}$, *parallel interfaces* $\{\varepsilon\}$, A, *trivial sequential interfaces, and transition matrix*

$$\mathsf{O}_{\varepsilon,a} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

The other transition matrices are zero matrices.
We define the automaton $\mathbf{Q}$ given a, *denoted by* $\mathbf{Q}|a$, *as*

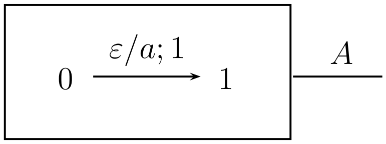
$$\mathbf{Q}|a = \mathbf{O}_a \times_A \mathbf{Q}$$

Given a signal $b \in B$, *we define* the probability of b given a *in a state* $q \in Q$ *in the weighted automaton* $\mathbf{Q}$, *denoted by* $P^{\mathbf{Q}}_q(b|a)$, *as*

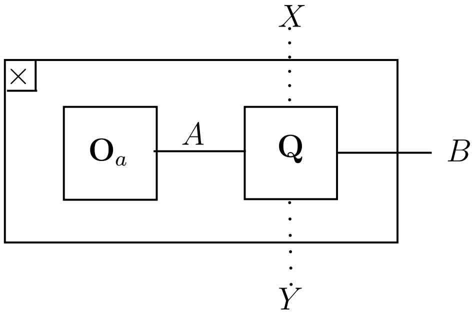
$$P^{\mathbf{Q}}_q(b|a) = P^{\mathbf{Q}|a}_{(0,q)}(b)$$

[3] For the definition of total matrix see p. 16.

The automaton $\mathbf{O}_a$ is described by the diagram:



The automaton $\mathbf{Q}|a$ is the communicating parallel composite:



Remark 4.2.4. Conditional probability $P_q^{\mathbf{Q}}(b|a)$ is defined also when $P_q^{\mathbf{Q}}(a) = 0$.

Now we extend the notion of probability of a signal on an interface to the case of two signals, one on each parallel interface.

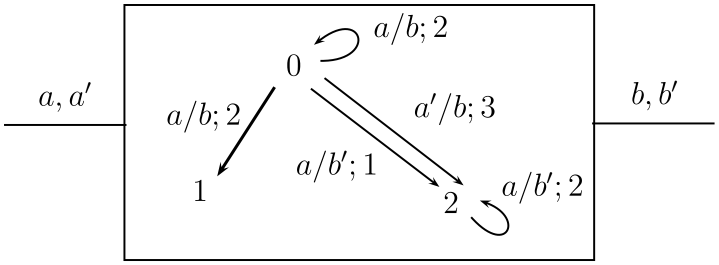
Definition 4.2.5. *Given a* weighted automaton $\mathbf{Q}^X_{Y;A,B}$*, we define the* probability of the signals $a \in A$ on the left interface and $b \in B$ on the right interface in a state $q \in Q$ *in the weighted automaton* $\mathbf{Q}^X_{Y;A,B}$*, denoted by* $P_q^{\mathbf{Q}}(a,b)$*, with the formula*

$$P_q^{\mathbf{Q}}(a,b) = \sum_{q' \in Q} [\mathbf{N}(\mathsf{Q}_{a,b})]_{q,q'}$$

Clearly, if there is at least one non-zero transition out of q then

$$P_q^{\mathbf{Q}}(a,b) = \frac{\sum_{q'} \left[\mathsf{Q}_{a,b}\right]_{q,q'}}{\sum_{q',a,b} \left[\mathsf{Q}_{a,b}\right]_{q,q'}}$$

Example 4.2.6. Consider the weighted automaton $\mathbf{Q}$ represented by the following picture:



Then $P^{\mathbf{Q}}_0(b) = \frac{7}{8}$, $P^{\mathbf{Q}}_0(b|a) = \frac{4}{5}$, $P^{\mathbf{Q}}_0(a,b) = \frac{1}{2}$, $P^{\mathbf{Q}}_2(b) = 0$, $P^{\mathbf{Q}}_2(a|b) = 0$.

The following result is an analogous of Bayes' theorem for the algebra of weighted automata.

Theorem 4.2.7. *(Bayes' theorem for weighted automata)*
Let $\mathbf{Q}^X_{Y;A,B}$ be a weighted automaton. Then, for each $a \in A$, $b \in B$ and $q \in Q$

$$P^{\mathbf{Q}}_q(a,b) = P^{\mathbf{Q}}_q(b|a)P^{\mathbf{Q}}_q(a)$$

Proof. Let $\mathbf{Q}^X_{Y;A,B}$ be a weighted automaton. Let $a \in A$, $b \in B$, $q \in Q$. If $P^{\mathbf{Q}}_q(a) = 0$ then also $P^{\mathbf{Q}}_q(a,b) = 0$ and the theorem is true.

Let $P^{\mathbf{Q}}_q(a) \neq 0$. Given $x \in \{0,1\}$, $q' \in Q$ we have that

$$\begin{aligned}
[(\mathbf{Q}|a)_{\varepsilon,b}]_{(0,q),(x,q')} &= [(\mathbf{O}_a)_{\varepsilon,a} \otimes \mathbf{Q}_{a,b}]_{(0,q),(x,q')} \\
&= [(\mathbf{O}_a)_{\varepsilon,a}]_{0,x} \cdot [\mathbf{Q}_{a,b}]_{q,q'} \\
&= \begin{cases} [\mathbf{Q}_{a,b}]_{q,q'} & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{4.1}$$

By (4.1) and by $P^{\mathbf{Q}}_q(a) \neq 0$ it follows that

$$\sum_{(x,q')\in\{0,1\}\times Q} \sum_{b\in B} [(\mathbf{Q}|a)_{\varepsilon,b}]_{(0,q),(x,q')} = \sum_{q'\in Q} \sum_{b\in B} [\mathbf{Q}_{a,b}]_{q,q'} \neq 0 \tag{4.2}$$

Then

$$\begin{aligned}
P^{\mathbf{Q}}_q(b|a) &= P^{\mathbf{Q}|a}_{(0,q)}(b) \\
&= \sum_{(x,q')\in\{0,1\}\times Q} [\mathsf{N}(\mathsf{Q}|a)_{\varepsilon,b}]_{(0,q),(x,q')} \\
&= \frac{\sum_{(x,q')} [(\mathbf{Q}|a)_{\varepsilon,b}]_{(0,q),(x,q')}}{\sum_{(x,q')} \sum_{b\in B} [(\mathbf{Q}|a)_{\varepsilon,b}]_{(0,q),(x,q')}} \quad \text{by (4.2)} \\
&= \frac{\sum_{q'\in Q} [\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'\in Q} \sum_{b\in B} [\mathsf{Q}_{a,b}]_{q,q'}} \quad \text{by (4.1)}
\end{aligned}$$

Hence, since $P^{\mathbf{Q}}_q(a) \neq 0$ implies $\sum_{q'\in Q}[\mathcal{Q}]_{q,q'} \neq 0$, it follows that:

$$\begin{aligned}
P^{\mathbf{Q}}_q(b|a)P^{\mathbf{Q}}_q(a) &= \frac{\sum_{q'\in Q} [\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'\in Q} \sum_{b\in B} [\mathsf{Q}_{a,b}]_{q,q'}} \cdot \frac{\sum_{q'\in Q} \sum_{b\in B} [\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q'\in Q}[\mathcal{Q}]_{q,q'}} \\
&= P^{\mathbf{Q}}_q(a,b)
\end{aligned}$$

□

Corollary 4.2.8. *In a weighted automaton $\mathbf{Q}^X_{Y;A,B}$ for each $a \in A$, $b \in B$ and $q \in Q$*

$$P^{\mathbf{Q}}_q(b|a)P^{\mathbf{Q}}_q(a) = P^{\mathbf{Q}}_q(a|b)P^{\mathbf{Q}}_q(b)$$

Theorem 4.2.9. *(Total probability theorem for weighted automata) Let $\mathbf{Q}^X_{Y;A,B}$ be a weighted automaton. Then, for each $b \in B$ and $q \in Q$*

$$P^{\mathbf{Q}}_q(b) = \sum_{a \in A} P^{\mathbf{Q}}_q(b|a)P^{\mathbf{Q}}_q(a)$$

Proof. If $\sum_{q' \in Q}[\mathcal{Q}]_{q,q'} = 0$ then $P^{\mathbf{Q}}_q(b) = 0$, for each $a \in A$ $P^{\mathbf{Q}}_q(a) = 0$ and the theorem is true.

Let $\sum_{q' \in Q}[\mathcal{Q}]_{q,q'} \neq 0$.

$$\begin{aligned} P^{\mathbf{Q}}_q(b) &= \frac{\sum_{q' \in Q}\sum_{a \in A}[\mathsf{Q}_{a,b}]_{q,q'}}{\sum_{q' \in Q}[\mathcal{Q}]_{q,q'}} \\ &= \sum_{a \in A} P^{\mathbf{Q}}_q(a,b) \\ &= \sum_{a \in A} P^{\mathbf{Q}}_q(b|a)P^{\mathbf{Q}}_q(a) \quad \text{by Bayes' theorem} \end{aligned}$$

□

We introduce now the definition of the probability of a path $v = (b_1, b_2, \cdots, b_k)$ on an interface B.

Definition 4.2.10. *Given a* weighted automaton $\mathbf{Q}^X_{Y;A,B}$ *and a natural number k, we define the* probability of a path $v = (b_1, b_2, \cdots, b_k) \in B^k$ on B in a state q of Q *as*

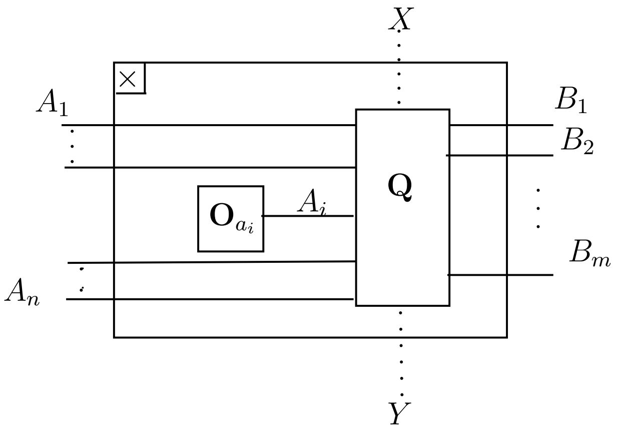
$$P^{\mathbf{Q}}_q(v) = P^{\mathbf{Q}^k}_q(v)$$

In a similar way we define $P^{\mathbf{Q}}_q(u)$ with $u \in A^k$.

$P^{\mathbf{Q}}_q(v)$ is defined as the probability of the signal v in q in the power automaton $\mathbf{Q}^k$.

Remark 4.2.11. It is obvious by the definition of probability of a path $v \in B^k$ on B in a weighted automaton $\mathbf{Q}^X_{Y;A,B}$ that we can extend the notion of conditional probability defining also the probability $P^{\mathbf{Q}}_q(v|u)$ of a path $v \in B^k$ in a state q given a path $u \in A^k$ of the same length. Thus, Bayes' theorem and Total probability theorem can be extended to probabilities of paths.

Remark 4.2.12. It is interesting to note that it is also possible to give a more general definition of the probability of a signal in a given state considering weighted automata with products $A_1 \times \cdots \times A_n$, $B_1 \times \cdots \times B_m$ as parallel interfaces. So, given a weighted automaton $\mathbf{Q}^X_{Y;A,B}$ where $A = A_1 \times \cdots \times A_n$, we can define the probability of a *signal* $a_i \in A_i$*, for a given* $i \in \{1, \ldots, n\}$*, in a state* $q \in Q$ *on the i-th wire of* A as the normalized sum of all the weights with left label $(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$ with $a_k \in A_k$, $k = 1, \ldots, (i-1), (i+1), \ldots, n$.
An intuitive definition of the weighted automaton $\mathbf{Q}$ given a_i on the i-th wire of A is suggested by the following picture:



4.3 Examples

4.3.1 Checking a fair coin

One box contains 20 fair coins and 5 coins rigged so that the probability of getting heads is $\frac{3}{4}$. Alice selects randomly a coin from the box and tosses it four times. She observes the following outcomes:

'head', 'head', 'tail', 'head'.

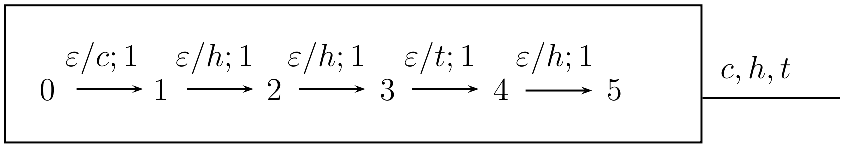
What is the probability that the selected coin is rigged given the observation?

To describe the problem we need two weighted automata - **Alice** and **Box**, representing the observer Alice and the box respectively. **Alice** has state space $\{0, 1, 2, 3, 4, 5\}$, parallel interfaces $\{\varepsilon\}$, $\{c, h, t\}$ and trivial sequential interfaces. The non zero transition matrices are:

$$\mathsf{Alice}_{\varepsilon,c} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathsf{Alice}_{\varepsilon,h} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \ \mathsf{Alice}_{\varepsilon,t} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Alice has labelled graph as follows:



It means that when Alice chooses a coin a signal c occurs on the non trivial parallel interface. She does not know if the selected coin is fair or not. She observes with probability 1 the outcomes h, h, t, h (h is 'head' and t 'tail').

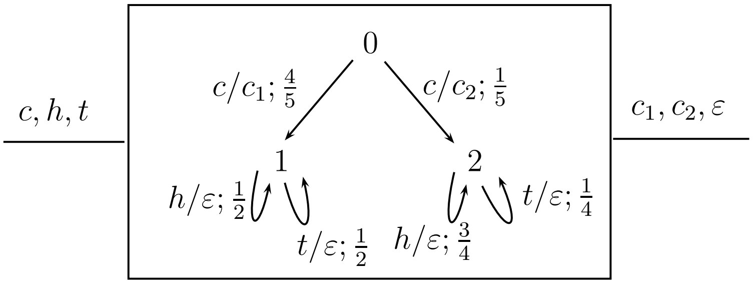
The automaton **Box** has state space $\{0, 1, 2\}$, parallel interfaces $\{c, h, t\}$ and $\{c_1, c_2, \varepsilon\}$, and trivial sequential interfaces. The transition matrices of **Box** are:

$$\mathsf{Box}_{c,c_1} = \begin{bmatrix} 0 & \frac{4}{5} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ \mathsf{Box}_{c,c_2} = \begin{bmatrix} 0 & 0 & \frac{1}{5} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

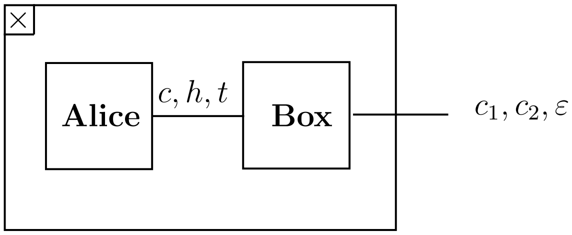
$$\mathsf{Box}_{h,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{3}{4} \end{bmatrix}, \ \mathsf{Box}_{t,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix}.$$

The other eight transition matrices are zero matrices.

Box is represented by the following picture:


c, h, t
0
$c/c_1; \frac{4}{5}$
$c/c_2; \frac{1}{5}$
1
2
$h/\varepsilon; \frac{1}{2}$
$t/\varepsilon; \frac{1}{2}$
$h/\varepsilon; \frac{3}{4}$
$t/\varepsilon; \frac{1}{4}$
c_1, c_2, ε

The system **Box** *given* the observation of Alice is the communicating parallel composite **Alice**||**Box**:



The required probability is:

$$\begin{aligned}
P_0^{\mathbf{Box}}(c_2\varepsilon\varepsilon\varepsilon\varepsilon|chhth) &= \frac{P_0^{\mathbf{Box}}(chhth, c_2\varepsilon\varepsilon\varepsilon\varepsilon)}{P_0^{\mathbf{Box}}(chhth)} \\
&= \frac{P_0^{\mathbf{Box}}(chhth, c_2\varepsilon\varepsilon\varepsilon\varepsilon)}{P_0^{\mathbf{Box}}(chhth, c_2\varepsilon\varepsilon\varepsilon\varepsilon) + P_0^{\mathbf{Box}}(chhth, c_1\varepsilon\varepsilon\varepsilon\varepsilon)} \\
&= \frac{\frac{1}{5}\,(\frac{3}{4})^3\,\frac{1}{4}}{\frac{1}{5}\,(\frac{3}{4})^3\,\frac{1}{4}\,+\,\frac{4}{5}\,(\frac{1}{2})^4} \cong 0.297
\end{aligned}$$

The observation of Alice, that is the path *chhth*, *changes* the probability of the path $c_2\varepsilon\varepsilon\varepsilon\varepsilon$.

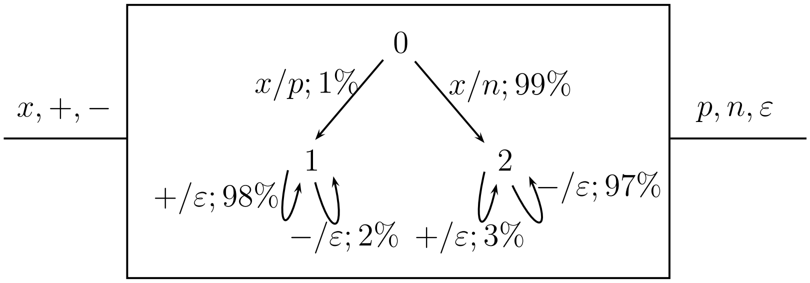
Remark 4.3.1. If we want to use the terminology of classical probability theory, we define C_1 as the event ”a fair coin is choosen”, C_2 as the event ”a non fair coin is choosen” and O as the event ”Alice obtains head, head, tail, head”. Then, $P(C_2) = \frac{1}{5}$ is the prior probability to select a rigged coin. The required probability is the posterior probability $P(C_2|O)$ that we calculate with the formula

$$P(C_2|O) = \frac{P(O|C_2)P(C_2)}{P(O|C_2)P(C_2) + P(O|C_1)P(C_1)}$$

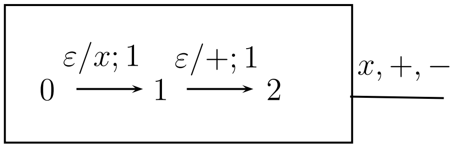
4.3.2 False positives

We want now to consider a diagnostic test to detect a specific disease with a low rate incidence: we know that 1 person in 100 is infected. We know also that if a subject is infected then the error rate of the test, i.e. a negative test result with a positive subject, is 2/100. On the other hand, the test is wrong 2 times in 100 within non infected population. What is the probability that a subject with a positive test result is actually infected?

We model the sampled population with a weighted automaton **P** with state space $\{0, 1, 2\}$, left interface $\{x, +, -\}$ and right interface $\{p, n, \varepsilon\}$.



An observer is a weighted automaton **O** with state space $\{0, 1, 2\}$, left interface $\{\varepsilon\}$, right interface $\{x, +, -\}$ represented by the following picture:



The system is given by the communicating parallel composite **O**||**P**. The probability of false positives is:

$$P_0(n\varepsilon|x+) = \frac{0.99 \cdot 0.03}{0.99 \cdot 0.03 \; + \; 0.01 \cdot 0.98} \cong 0.752$$

At first glance this outcome could seem to be counter-intuitive. The probability that a positive test result indicates a positive subject is not given only by the accuracy of the test, but also depends on the characteristics of the sampled population (by Bayes' theorem). When the incidence is lower than the test is false positive rate, even tests that have a very low chance of giving a false positive in an individual case will give more false than true positives overall.

The key concept of Bayes' theorem is that the true rates of false positives and false negatives are not a function of the accuracy of the test alone, but also the actual rate or frequency of occurrence within the test population.

4.3.3 Two boys and girls problem

Here we want to explain how to solve a problem proposed by Gary Foshee, a puzzle designer from Issaquah, Washigton, during a talk at the *Ninth Gathering* 4 *Gardner* (March 24 – 28, 2010, Atlanta, [4]). Foshee's question, which we call *Two boys and girls problem*, belongs to a well-known set of problems which generated long discussions between mathematicians and many variants[4].

The problem we consider is the following:

if someone with two children tells you that at least one of whom is a boy born Tuesday, what is the probability that both children are boys?

We assume that the sex of each child is independent of the sex of the other, the probabilities to have a boy or a girl are equal and that the births are equally distributed during the week.

We consider first two simpler versions of this problem in which there is no mention of the day of birth.

Case 1: declaration

Consider the following problem:

1. *If someone with two children* tells *you that at least one of whom is a boy, what is the probability that that both children are boys?*

The simplifying assumption mentioned above about boys and girls birthrate means that given a parent with two children the following four combinations in birth order are equally likely: *Boy-Boy, Boy-Girl, Girl-Boy, Girl-Girl*. In the case of 2 boys a parent declares to have (at least) one boy with weight 1, in the case of 2 girls the weight of a declaration that there is a girl is 1, but in the other two cases there is a half chance that the declaration is "boy", and a half chance that the declaration is "girl".

To describe the problem we need a weighted automaton $\mathbf{Q}_1$ with state space $\{1, 2, 3, 4\}$, left parallel interface $\{x, b, g\}$, right interface $\{\varepsilon, b_0, b_1, b_2\}$,

[4] One of the most famous formulations of the problem was introduced by Martin Gardner with the name of *Two children problem* in a *Scientific American* column in 1959 ([28]) and then was republished in his book [27].

both sequential interfaces $\emptyset \subseteq \{1,2,3,4\}$, and transition matrices

$$(\mathsf{Q}_1)_{x,b_0} = \begin{bmatrix} 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

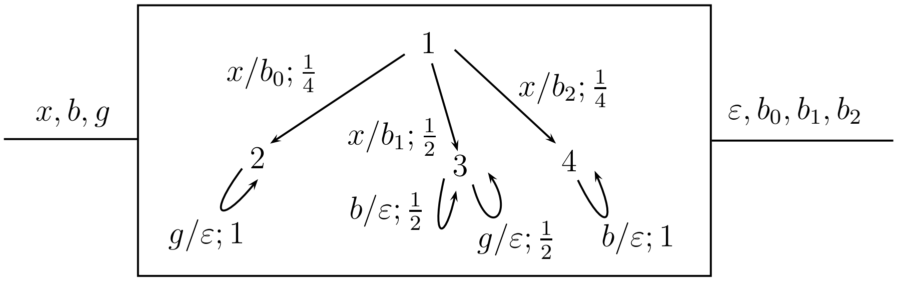
$$(\mathsf{Q}_1)_{x,b_1} = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ (\mathsf{Q}_1)_{x,b_2} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$(\mathsf{Q}_1)_{g,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ (\mathsf{Q}_1)_{b,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The other transition matrices are zero matrices.

The intention behind these matrices is as follows: in state 1 the system does a transition to state 2 with probability $\frac{1}{4}$ labelled x/b_0 (*a parent with no boys is choosen*), a transition to state 3 with probability $\frac{1}{2}$ labelled x/b_1 (*a parent with exactly one boy is choosen*) and a transition to state 4 with probability $\frac{1}{4}$ labelled x/b_2 (*a parent with* 2 *boys is choosen*); in state 2 the system does a transition to state 2 with probability 1 labelled g/ε (*the parent says to have a girl*); in state 3 the system does a transition to state 3 with probability $\frac{1}{2}$ labelled g/ε (*the parent says to have a girl*) and a transition to state 3 with probability $\frac{1}{2}$ labelled b/ε (*the parent says to have a boy*); in state 4 the system does a transition to state 4 with probability 1 labelled b/ε (*the parent says to have a boy*).

We put this information in the following picture:



The required probability is:

$$P_1(b_2\varepsilon|xb) = \frac{\frac{1}{4}}{\frac{1}{4} + \frac{1}{2}\frac{1}{2}} = \frac{1}{2}$$

The information about a child given by a declaration does not change the probability that the other child is a boy.

Case 2: a question

Consider now a slightly different problem:

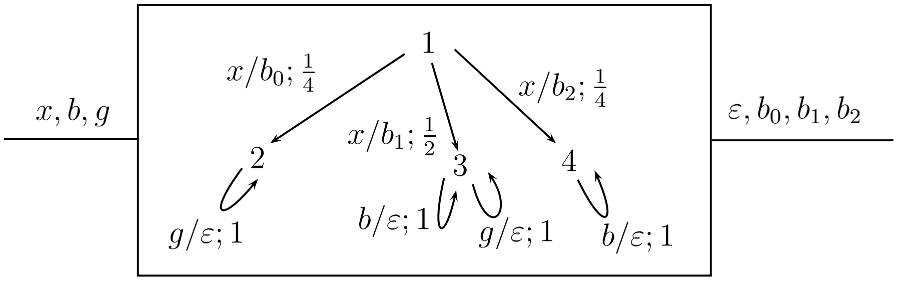
2. *if you* ask *to someone with two children if he has at least a boy and he answer "yes", then what is the probability that both children are boys?*

In this case a parent with 2 boys gives a positive answer to the question if he has a boy with the same weight of a parent with only 1 boy. Similarly, a parent with 2 girls gives a positive answer to the question if he has a girl with the same weight of a parent with only 1 girl.

The system is a weighted automaton Q_2 with the same state space, interfaces and transition matrices of Q_1 except for the following matrices:

$$(\mathsf{Q}_2)_{g,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ (\mathsf{Q}_2)_{b,\varepsilon} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Q_2 is represented by the following picture:



The weights of the transitions in state 3 (*parent with exactly one boy*) to state 3 and in state 4 (*parent with* 2 *boys*) to state 4 labelled b/ε (*positive answer to the question about a boy*) are the same. Similarly, the transition from state 3 to state 3 and the transition from state 4 to state 4 labelled g/ε (*positive answer to the question about a girl*) have the same weights.

The required probability is:

$$P_1(b_2\varepsilon|xb) = \frac{\frac{1}{4}}{\frac{1}{4}+\frac{1}{2}} = \frac{1}{3}$$

A positive answer about a boy changes the probability to have 2 boys from $\frac{1}{4}$ (before the question) to $\frac{1}{3}$ (after the positive answer).

The day of birth

Consider now the original version of *Two boys and girls problem* given above. The simplifying assumption about the birthrate of boys and girls during the week means that the probability to have a boy born on Tuesday is the same to have a boy, i.e. $\frac{1}{2}$. As in case 1, the weight of the transition in state 4 labelled b/ε (*declaration to have a boy born on Tuesday*) to state 4 is 1. Similarly, the weight of the transition in state 3 with the same label to state 3 is $\frac{1}{2}$. Calculations analogous to those described above show that the required conditional probability is $\frac{1}{2}$. The information given about a child, including the day of birth, does not change the probability that the second child is a boy.

Remark 4.3.2. Consider the problem

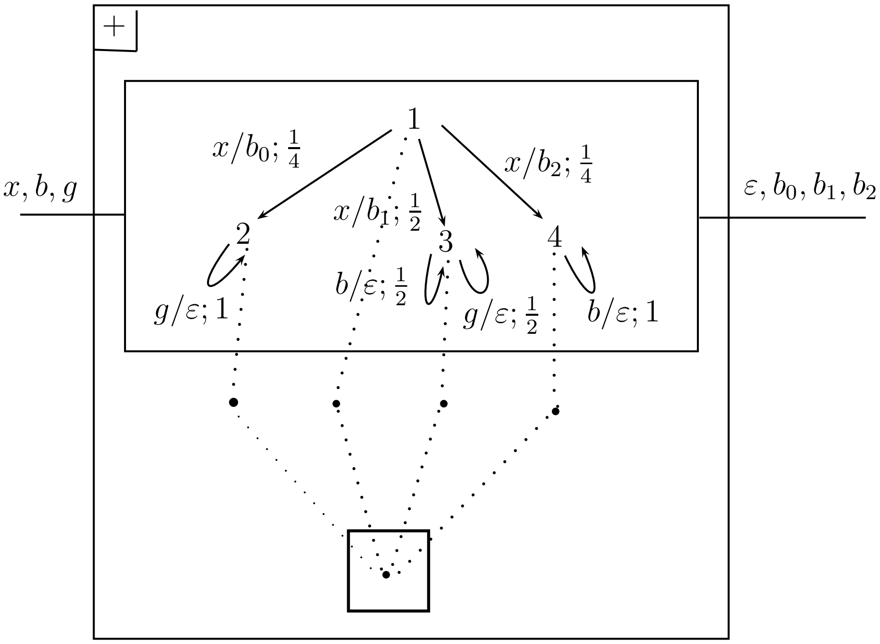
> *if you* ask *to someone with two children if he has at least a boy born on Tuesday and he answer "yes", then what is the probability that both children are boys?*

Then, calculations analogous to those of case 2 give the answer $\frac{13}{27}$.

4.4 Concluding remarks

The problems described in this chapter show that a wrong interpretation of conditional probability leads often to paradoxical answers. Many of these paradoxes arise from considering normalized probabilities, rather than weights of actions.

In classical probability theory, which is based on set theory rather than automata, probabilities are not explicitly attached to a specific state and the systems described are *abstractions* of weighted automata. Mathematically, the abstraction of a weighted automaton $\mathbf{Q}$ is the sequential operation consisting in identifying all the states of $\mathbf{Q}$ like in the following example:



Many of the paradoxes arise when we normalize *before* the abstraction because normalization is not compositional with respect to sequential operations[5](including abstraction).

[5] See remark 1.2.13 p. 29.

Chapter 5

Quantum automata

This chapter is devoted to another extension of the span graphs algebra ([40]) closely related to weighted and Markov automata model described in chapter 1. Here our purpose is the compositional description of quantum protocols. In this extension, introduced in [19], we consider doubly indexed families $\varphi_{a,b}$ $(a \in A, b \in B)$ of *operators* on finite dimensional vector spaces, the main operation being $\Sigma_{b\in B}\varphi_{a,b} \otimes \psi_{b,c}$, $(a \in A,\ c \in C)$.

We distinguish two kind of components: quantum automata and classical finite state automata. The benefit of this approach is that the inclusion of explicit components of finite state classical control adds conceptual clarity and precision to quantum protocols.

An important thing to note is that in order to decompose the above system into parts, the algebra contains much more than just classical and quantum components - it is only in combination that the qubits form quantum components.

Another point of interest is that span graphs algebra has been used for compositional model checking (in which non-determinism and the state explosion are considered the main problem) whereas here our extension is used for quantum computing (in which linearity and the expanded state space are the cited advantage).

We define a $\mathbb{C}$-automaton $\mathbf{Q}$ with a given set A of "signals on the left interface", and set B of "signals on the right interface" to consist of a finite dimensional complex vector space V and a family of linear transformations $\varphi_{a,b} : V \to V$ $(a \in A, b \in B)$. A quantum $\mathbb{C}$-automaton is one in which the space V has the extra structure of an hermitian inner product, and in which the linear transformations are unitary transformations or orthogonal projections. A classical $\mathbb{C}$-automaton is one with the extra structure that the space V is of the form $\mathbb{C}^X$ for a given finite set X and for which the matrices of the linear transformations are zero-one matrices induced by binary relations on X.

As example we explain how the teleportation protocol of [6] may be modelled in our algebra and we prove its correctness.

5.1 $\mathbb{C}$-automata

Definition 5.1.1. *Consider two finite alphabets A and B. A* $\mathbb{C}$-automaton $\mathbf{Q}$ with left interface A and right interface B *consists of a finite dimensional complex vector space V of states, and an $A \times B$ indexed family $\varphi = \varphi_{a,b(a \in A, b \in B)}$ of linear transformations from V to V.*

The idea of a $\mathbb{C}$-automaton is the same of Markov automata: the transitions that occur induce *signals* on the two interfaces of the automaton, which signals are represented by letters in the alphabets. For examples see a later section.

Example 5.1.2. Let V be a 2-dimensional complex vector space with a basis $\{v_1, v_2\}$. Consider the alphabets $A = \{a, a'\}$, $B = \{b\}$ and the linear transformations $\varphi_{a,b}, \varphi_{a',b} : V \to V$ whose representative matrices with respect to the given basis are:

$$\varphi_{a,b} = \begin{bmatrix} 2i & 0 \\ 4 & i \end{bmatrix}, \ \varphi_{a',b} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Then $\mathbf{Q}$ with state space V and linear transformations $\{\varphi_{a,b}, \varphi_{a',b}\}$ is a $\mathbb{C}$-automaton with left interface A and right interface B. A state of $\mathbf{Q}$ is a linear combination $c_1 v_1 + c_2 v_2$ with $c_1, c_2 \in \mathbb{C}$.

Definition 5.1.3. *A $\mathbb{C}$-automaton $\mathbf{Q}$ with the extra structure that the space V is endowed with an hermitian inner product $< \mid >$ and for which the linear transformations are either unitary or orthogonal projections is called a* quantum $\mathbb{C}$-automaton.

Definition 5.1.4. *A $\mathbb{C}$-automaton $\mathbf{Q}$ with the extra structure that the space V is $\mathbb{C}^X$ for a given finite set X, and for which the linear transformations $\varphi_{a,b}$ are of the form $\varphi_{a,b}(e_x) = \sum_y c_y e_y$ with c_y is 0 or 1 $(x \in X)$ (where $\{e_x\}_{(x \in X)}$ is the standard basis of $\mathbb{C}^X$ defined by $e_x(y) = 1$ if $y = x$, and 0 otherwise) is called a* classical (finite state) $\mathbb{C}$ -automaton. *Note: we will often write just x instead of e_x for a basis element.*

The definition of classical $\mathbb{C}$-automaton requires that every element of the matrices of the linear transformations written with respect to the standard basis are 0 or 1.

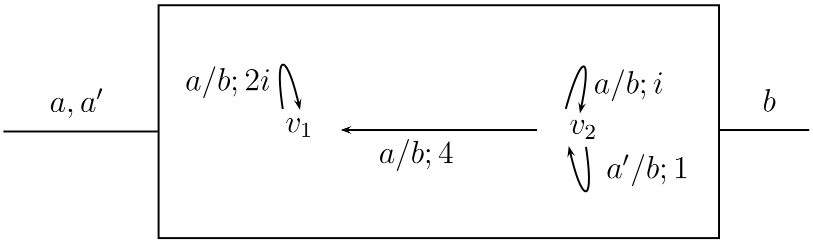
Definition 5.1.5. *Consider a $\mathbb{C}$-automaton* $\mathbf{Q}$ *with interfaces A and B. A* behaviour *of length k of* $\mathbf{Q}$ *consists of a two words of length k, one $w_1 = a_1a_2 \cdots a_k$ in A^* and the other $w_2 = b_1b_2 \cdots b_k$ in B^* and a sequence of vectors*

$$\mathbf{x}_0, \mathbf{x}_1 = \varphi_{a_1,b_1}(\mathbf{x}_0), \mathbf{x}_2 = \varphi_{a_2,b_2}(\mathbf{x}_1), \cdots, \mathbf{x}_k = \varphi_{a_k,b_k}(\mathbf{x}_{k-1}).$$

5.2 Graphical representation

There is an intuitive graphical notation of classical and quantum $\mathbb{C}$-automata analogous to that used for Markov automata.

Example 5.2.1. The $\mathbb{C}$-automaton $\mathbf{Q}$ of example 5.1.2 is represented by the following picture:



Given a chosen basis for the state space of an automaton (or, more generally, a decomposition of the state space as a direct sum) we may compress the description of an automaton with interfaces A and B, which requires $A \times B$ matrices of scalars (or, more generally, matrices of linear transformations), into a single labelled graph, like the ones introduced in [40]. Further, expressions of automata in this algebra may be drawn as "tensor diagrams" also as in [40]. We indicate both of these matters by describing some examples.

5.2.1 Qubits

Qubit automata are a $\mathbb{C}$-automata with state space $\mathbb{C}^2$ which singly, or combined, form quantum automata. We will describe three particular qubit automata which will need for our discussion of teleportation. One of the qubit automata is a quantum automaton; the others will be combined to form a 2 qubit quantum automaton.

Qubit Q_1

Consider the alphabets $A_1 = \{\varepsilon, c, h, m_0, m_1\}$ and $B_1 = \{\varepsilon, \neg\}$. Then $\mathbf{Q}_1$ is the automaton with left interface A_1 and right interface B_1, state space $\mathbb{C}^2$

and transition matrices

$$\begin{aligned}
\varphi_{\varepsilon,\varepsilon} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \varphi_{c,\neg} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\
\varphi_{c,\varepsilon} &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \ \varphi_{h,\varepsilon} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\
\varphi_{m_0,\varepsilon} &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \ \varphi_{m_1,\varepsilon} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned}$$

The other four transition matrices are zero matrices.

The intention behind these matrices is as follows: $\mathbf{Q}_1$ may do a transition labelled ε, ε (*idle transition*); $\mathbf{Q}_1$ may receive a signal h and perform a transition determined by the unitary Hadamard matrix; $\mathbf{Q}_1$ may receive a signal c (*do* $\mathrm{C}_{\mathrm{not}}$) and if it is in state 1 pass on a signal $\neg$ with the intention to perform a *not* on another qubit; the signal m_0 means that a *measurement* with result 0 has occurred on $\mathbf{Q}_1$; the signal m_1 means that a *measurement* with result 1 has occurred on $\mathbf{Q}_1$. All this information may be put in the following diagram, noting that

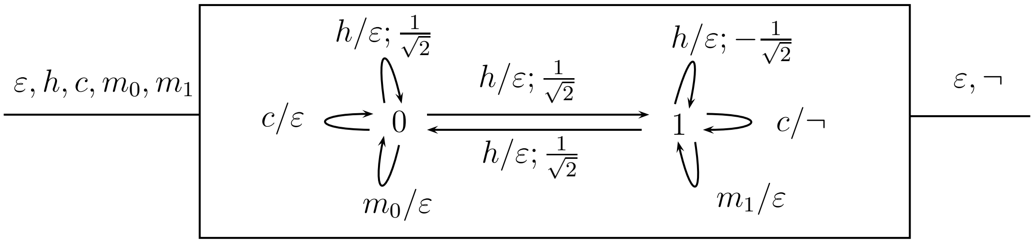
(i) the basis elements of $\mathbb{C}^2$ are called 0 and 1, and occur in the diagram as vertices,

(ii) labels of transitions indicate which matrix is involved,

(iii) the absence of an edge from i to j means that the i,jth element of the matrix is 0,

(iv) we have in any case omitted loops labelled ε, ε,

(v) we have included the value of the matrix element only when it is not 1.


$\varepsilon, h, c, m_0, m_1$
$h/\varepsilon; \frac{1}{\sqrt{2}}$
$h/\varepsilon; -\frac{1}{\sqrt{2}}$
$\varepsilon, \neg$
$h/\varepsilon; \frac{1}{\sqrt{2}}$
c/ε
0
1
$c/\neg$
$h/\varepsilon; \frac{1}{\sqrt{2}}$
m_0/ε
m_1/ε

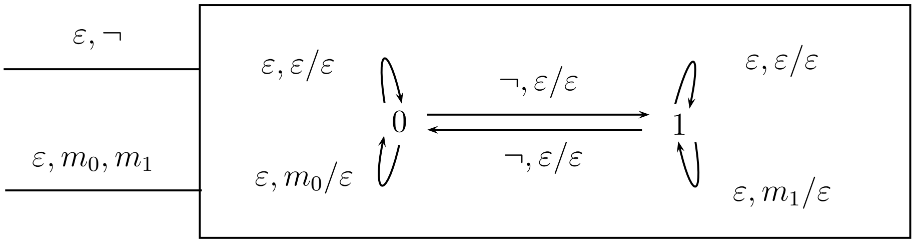
Qubit Q_2

Consider the alphabets $A_2 = \{\varepsilon, \neg\} \times \{\varepsilon, m_0, m_1\} = A_{21} \times A_{22} = B_1 \times A_{22}$ and $B_2 = \{\varepsilon\}$. Then $\mathbf{Q}_2$ is the automaton with left interface A_2 and right interface B_2, state space $\mathbb{C}^2$ and transition matrices

$$\varphi_{(\varepsilon,\varepsilon),\varepsilon} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \varphi_{(\neg,\varepsilon),\varepsilon} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$\varphi_{(\varepsilon,m_0),\varepsilon} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \ \varphi_{(\varepsilon,m_1),\varepsilon} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

The remaining matrices are zero.

The intention behind these matrices is as follows: $\mathbf{Q}_2$ may do a transition labelled ε, ε (*idle transition*); $\mathbf{Q}_2$ may receive a signal $\neg$ and perform a *not* transition; the signal m_0 means that a *measurement* with result 0 has occurred on $\mathbf{Q}_2$; the signal m_1 means that a *measurement* with result 1 has occurred on $\mathbf{Q}_2$.



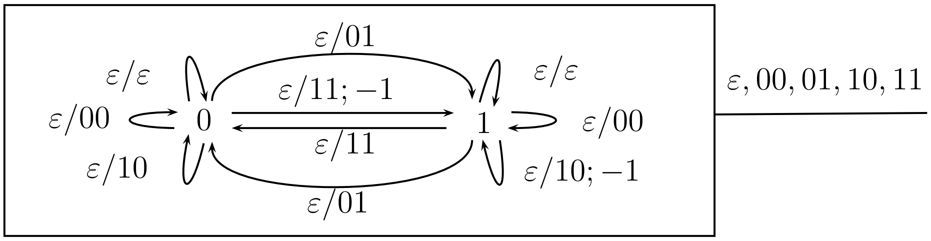
$\varepsilon, \neg$
ε, m_0, m_1
$\varepsilon, \varepsilon/\varepsilon$
$\neg, \varepsilon/\varepsilon$
$\varepsilon, \varepsilon/\varepsilon$
0
1
$\varepsilon, m_0/\varepsilon$
$\neg, \varepsilon/\varepsilon$
$\varepsilon, m_1/\varepsilon$

Qubit Q_3

Consider the alphabets $A_3 = \{\varepsilon\}$, and $B_3 = \{\varepsilon, 00, 01, 10, 11\}$. Then $\mathbf{Q}_3$ is the automaton with left interface A_3 and right interface B_3, state space $\mathbb{C}^2$ and transition matrices

$$\varphi_{\varepsilon,\varepsilon} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$\varphi_{\varepsilon,00} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \varphi_{\varepsilon,10} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$
$$\varphi_{\varepsilon,01} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \ \varphi_{\varepsilon,11} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

The intention behind these matrices is as follows: $\mathbf{Q}_3$ may do a transition labelled ε, ε (*idle transition*); $\mathbf{Q}_3$ may receive one of four signal $00, 01, 10, 11$ and perform the given unitary transformations.



5.2.2 Alice and Bob

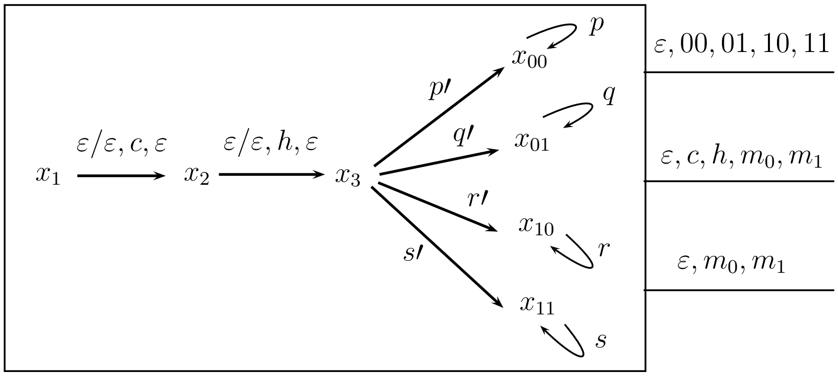
We now describe two classical $\mathbb{C}$-automata **Alice** and **Bob** which represent, respectively, the sender and the receiver of teleportation.

Alice

Let $X = \{x_1, x_2, x_3, x_{00}, x_{01}, x_{10}, x_{11}\}$. Then **Alice** is the classical $\mathbb{C}$-automaton with state space $\mathbb{C}^X$ with left interface $A_{Alice} = \{\varepsilon\}$ and right interface

$$\begin{aligned} B_{Alice} &= \{\varepsilon, 00, 01, 10, 11\} \times \{\varepsilon, c, h, m_0, m_1\} \times \{\varepsilon, m_0, m_1\} \\ &= B_{Alice,1} \times A_1 \times A_{22}. \end{aligned}$$

and transformations as indicated in the diagram


p
x_{00}
$\varepsilon, 00, 01, 10, 11$
$p\prime$
q
$\varepsilon/\varepsilon, c, \varepsilon$
$\varepsilon/\varepsilon, h, \varepsilon$
$q\prime$
x_{01}
$\varepsilon, c, h, m_0, m_1$
x_1
x_2
x_3
$r\prime$
x_{10}
r
$s\prime$
ε, m_0, m_1
x_{11}
s

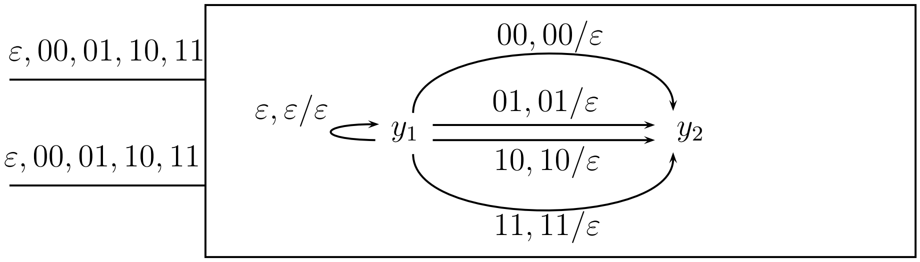
where

$$\begin{aligned} p' &= \varepsilon/\varepsilon, m_0, m_0, \qquad & p &= \varepsilon/00, \varepsilon, \varepsilon, \\ q' &= \varepsilon/\varepsilon, m_0, m_1, \qquad & q &= \varepsilon/01, \varepsilon, \varepsilon, \\ r' &= \varepsilon/\varepsilon, m_1, m_0, \qquad & r &= \varepsilon/10, \varepsilon, \varepsilon, \\ s' &= \varepsilon/\varepsilon, m_1, m_1, \qquad & s &= \varepsilon/11, \varepsilon, \varepsilon. \end{aligned}$$

Bob

Let $Y = \{y_1, y_2\}$. Then **Bob** is the classical $\mathbb{C}$-automaton with state space $\mathbb{C}^Y$ with left interface $A_{Bob} = \{\varepsilon, 00, 01, 10, 11\} \times \{\varepsilon, 00, 01, 10, 11\}$ and right interface $B_{Bob} = \{\varepsilon\}$ and transformations relative to the standard basis e_{y_1}, e_{y_2} having the following non-zero elements:

$$\varphi_{(\varepsilon,\varepsilon),\varepsilon}(e_{y_1}) = e_{y_1},$$
$$\varphi_{(00,00),\varepsilon}(e_{y_1}) = e_{y_2}, \varphi_{(01,01),\varepsilon}(e_{y_1}) = e_{y_2},$$
$$\varphi_{(10,10),\varepsilon}(e_{y_1}) = e_{y_2}, \varphi_{(11,11),\varepsilon}(e_{y_1}) = e_{y_2}.$$


$\varepsilon, 00, 01, 10, 11$
$\varepsilon, 00, 01, 10, 11$
$\varepsilon, \varepsilon/\varepsilon$
y_1
y_2
$00, 00/\varepsilon$
$01, 01/\varepsilon$
$10, 10/\varepsilon$
$11, 11/\varepsilon$

5.3 The algebra of $\mathbb{C}$-automata

Now we define operations on $\mathbb{C}$-automata analogous (in a precise sense) to those defined for Markov automata.

Definition 5.3.1. *Given a $\mathbb{C}$-automata $\mathbf{Q}$ with left and right interfaces A and B, state space V, and family of transformations φ, and $\mathbf{R}$ with interfaces C and D, state space W, transformations ψ, the* parallel composite $\mathbf{Q} \times \mathbf{R}$ *is the $\mathbb{C}$-automaton which has state space $V \otimes W$, left interface $A \times C$, right interface $B \times D$, and transformations*

$$(\varphi \times \psi)_{(a,c),(b,d)} = \varphi_{a,b} \otimes \psi_{c,d}.$$

Definition 5.3.2. *Given $\mathbb{C}$-automata $\mathbf{Q}$ with left and right interfaces A and B, state space V, and family of transformations φ, and $\mathbf{R}$ with interfaces B and C, state space W, and family of transformations ψ the* series (communicating parallel) composite of $\mathbb{C}$-automata $\mathbf{Q}\times_{\mathbf{B}}\mathbf{R}$ *(or more briefly $\mathbf{Q}||\mathbf{R}$) has state space $V \otimes W$, left interface A, right interface C, and transition maps*

$$(\varphi \times_B \psi)_{a,c} = \sum_{b \in B} \phi_{a,b} \otimes \psi_{b,c}.$$

Notice that when the state spaces of the $\mathbf{C}$-automata have direct sum decompositions, and hence the operators have matrix representations, the tensor products in the above definitions may be calculated (via distributivity isomorphisms) using tensor products of matrices. This gives a way of calculating the operations analogous to the operations on automata in [40].

Definition 5.3.3. *Given a relation $\rho \subset A \times B$ we define a $\mathbb{C}$-automaton* $\mathbf{Par}(\rho)$ *as follows: it has state space $\mathbb{C}$. The transition matrices $\rho_{a,b}$ are 1×1 matrices, that is, complex numbers. Then $\rho_{a,b} = 1$ if ρ relates a and b, and $\rho_{a,b} = 0$ otherwise.*

Some special cases, analogous to those defined for Markov automata, have particular importance and are called *parallel connectors* or *wires*:

(i) the automaton corresponding to the identity function 1_A, considered as a relation on $A \times A$ is called $\mathbf{1}_A$;

(ii) the automaton corresponding to the diagonal function $\Delta_A : A \to A \times A$ (considered as a relation) is called $\boldsymbol{\Delta}_A$; the automaton corresponding to the opposite relation of Δ_A is called $\boldsymbol{\Delta}_A^o$.

(iii) the automaton corresponding to the function $twist_{A,B} : A \times B \to B \times A$ is called $\mathbf{twist}_{A,B}$.

(iv) the automaton corresponding to the projection function $A \to \{*\}$ is called $\mathbf{p}_A$ and its opposite $\mathbf{p}_A^o$.

(v) the automaton corresponding to the relation $\eta_A = \{(*, (a, a)); a \in A\} \subset \{*\} \times (A \times A)$ is called η_A; the automaton corresponding to the opposite of η_A is called ε_A.

5.3.1 The teleportation protocol

An essential aspect of teleportation protocol is the existence of a pair of qubits in an *entangled* state, that is in a state that cannot be decomposed into separate states of each of the two qubits. Formally, we have the following

Definition 5.3.4. *Given two qubits* $\mathbf{Q}$ *with interfaces A,B, and* $\mathbf{R}$ *with interfaces C,D, we say that a state z of* $\mathbf{Q} \times \mathbf{R}$[1] *is* entangled *if there do not exist $\alpha_i, \beta_i \in \mathbb{C}, i = 1, 2$ such that*

$$z = (\alpha_1 0 + \beta_1 1) \otimes (\alpha_2 0 + \beta_2 1)$$

Entanglement implies that a measurement of one qubit affects a measurement of the other. Famous examples of entangled states are the *Bell states.*

[1] If z is a state of $\mathbf{Q} \times \mathbf{R}$ then $z \in \mathbb{C}^2 \otimes \mathbb{C}^2$ and $z = \sum_{i,j=0,1} \gamma_{i,j} i \otimes j$ with $\gamma_{i,j} \in \mathbb{C}, i, j = 0, 1$.

Definition 5.3.5. *Given two qubits* $\mathbf{Q}$ *and* $\mathbf{R}$*, we call* Bell states *(or* EPR states*) the following states of* $\mathbf{Q} \times \mathbf{R}$*:*

- $Bell_1 = \frac{1}{\sqrt{2}}(0 \otimes 0 + 1 \otimes 1)$,
- $Bell_2 = \frac{1}{\sqrt{2}}(0 \otimes 1 + 1 \otimes 0)$,
- $Bell_3 = \frac{1}{\sqrt{2}}(0 \otimes 0 - 1 \otimes 1)$,
- $Bell_4 = \frac{1}{\sqrt{2}}(0 \otimes 1 - 1 \otimes 0)$.

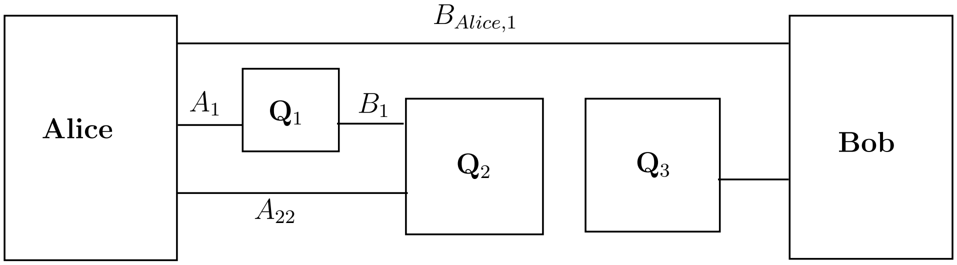
The protocol TP

Now the model of the teleportation protocol we consider is an expression in the algebra, involving also the automata $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3$, **Alice**, and **Bob**. The protocol is

$$\mathbf{TP} = \mathbf{Alice}||(1_{A_3} \times ((\mathbf{Q}_1 \times 1_{A_{22}})||\mathbf{Q}_2))||(1_{A_3} \times \mathbf{Q}_3)||\mathbf{Bob}.$$

Notice that $(\mathbf{Q}_1 \times 1_{A_{22}})||\mathbf{Q}_2$ and $\mathbf{Q}_3$ are quantum $\mathbb{C}$-automata.

As explained in [40], we may represent this system by the following diagram:



The behaviour of TP

Consider the following initial state of **TP**

$$x_1 \otimes (\alpha 0 + \beta 1) \otimes \frac{1}{\sqrt{2}}(0 \otimes 0 + 1 \otimes 1) \otimes y_1;$$

that is that state of $\mathbf{Q}_1$ is arbitrary and $\mathbf{Q}_2$ and $\mathbf{Q}_3$ are in the Bell state $Bell_1$. Since the combined system **TP** is closed it consists of a single linear transformation θ acting on the state space $\mathbb{C}^X \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^Y$. A behaviour consists of a sequence of applications of θ to the initial state. However, in view of the construction of θ from parts, we may give a more explicit description of behaviours beginning in this initial state. In the following calculation it is critical that $\mathbb{C}^X$ and C^Y break up into a direct sums $\mathbb{C} \oplus \mathbb{C} \oplus \cdots \oplus \mathbb{C}$ so that, using the distributive law of tensor over direct sum, **Alice** and **Bob** can do different actions on the qubits in different summands. This is entirely analogous to the use of sums and the distributive law in sequential programming, in particular in defining *if then else* [26],[68].

Simplifying the notation, writing for example 00 instead of $0 \otimes 0$, a four step behaviour is:

$$\begin{aligned}
&x_1 \otimes (\alpha 0 + \beta 1) \otimes \frac{1}{\sqrt{2}}(00 + 11) \otimes y_1 \\
&\mapsto x_2 \otimes \frac{1}{\sqrt{2}}(\alpha 000 + \alpha 011 + \beta 110 + \beta 101) \otimes y_1 \\
&\mapsto \frac{1}{2} x_3 \otimes (\alpha(0+1)00 + \alpha(0+1)11 + \beta(0-1)10 + \beta(0-1)01) \otimes y_1 \\
&= \frac{1}{2} x_3 \otimes (\alpha(000 + 100 + 011 + 111) + \beta(010 - 110 + 001 - 101)) \otimes y_1 \\
&\mapsto \frac{1}{2}(x_{00} \otimes (\alpha 000 + \beta 001) \otimes y_1 + x_{01} \otimes (\alpha 011 + \beta 010) \otimes y_1 + \\
&x_{10} \otimes (\alpha 100 - \beta 101) \otimes y_1 + x_{11} \otimes (\alpha 111 - \beta 110) \otimes y_1) \\
&\mapsto \frac{1}{2}(x_{00} \otimes (\alpha 000 + \beta 001) \otimes y_2 + x_{01} \otimes (\alpha 010 + \beta 011) \otimes y_2 + \\
&x_{10} \otimes (\alpha 100 + \beta 101) \otimes y_2 + x_{11} \otimes (\alpha 110 + \beta 111) \otimes y_2) \\
&= \frac{1}{2}(x_{00} \otimes 00 + x_{01} \otimes 01 + x_{10} \otimes 10 + x_{11} \otimes 11) \otimes (\alpha 0 + \beta 1) \otimes y_2.
\end{aligned}$$

5.3.2 The algebra of automata: further work

There is clearly much more to develop about the algebraic structure. We mention only that for each object A the constants $\mathbf{\Delta}_A$, $\mathbf{\Delta}_A^o$ satisfy the Frobenius equations[2] [12], namely that

$$(\mathbf{1}_A \times \mathbf{\Delta}_A^o)||(\mathbf{\Delta}_A \times \mathbf{1}_A) = \mathbf{\Delta}_A^o||\mathbf{\Delta}_A = (\mathbf{\Delta}_A \times \mathbf{1}_A)||(\mathbf{1}_A \times \mathbf{\Delta}_A^o).$$

and the equation

$$\mathbf{\Delta}_A||\mathbf{\Delta}_A^o = \mathbf{1}_A$$

Notice that relations on X also exist as closed classical automata with state space $\mathbb{C}^X$ and there the Frobenius equations are also satisfied, which fact has been used in axiomatizing classical data in [14].

There is another sense in which the algebra is incomplete. We have not described the relation between our diagrammatic representation, which

[2] See section 2.2.4 p. 50.

concern parallel operations, and those of Coecke, Selinger and others in which the diagrams represent flow of data, that is, involve sequential operations. We hope to apply the ideas of [41], [20] to study this relation.

Appendix A

Calculating $\mathbf{DP}_n$ with Maple

The use of a symbolic computation system may be extremely useful to evaluate expressions in the algebra of weighted or quantum automata. Here we want to compute the total matrix of the n Dining Philosopher system $\mathbf{DP}_n$ described in chapter 1 using a set of Maple procedures[1]. Since $\mathbf{DP}_n$ is an element of $\Pi(\Sigma(E))$, namely a communicating parallel system of sequential automata, in the following sections we will consider only operations on weighted automata with trivial sequential interfaces.

A.1 Weighted automata

We denote a state of a weighted automaton by an integer index `[n]`. A transition is a tuple

```
[[a],[b],[n],[m],p]
```

where `a` and `b` are the left and right labels respectively, `n` the domain, `m` the codomain and `p` the weight of the transition. Then, a weighted automaton with trivial sequential interfaces consists of a list

```
sys(state(),trans())
```

where

- `state()` is a list `[1],...,[n]` of indexes of states,
- `trans()` is a list `[t1],...,[tm]` of transitions.

Example A.1.1. The automaton **Phil** in the Dining Philosopher system[2] is represented in the following way:

[1] The procedures we describe were written in Maple 5.

[2] For the definition of **Phil** and **Fork** in the Dining Philosophers system see section 1.1.2.

```
> phil:=sys(stat([1],[2],[3],[4]),trans(
  [[e],[e],[1],[1],1/2],
  [[e],[e],[2],[2],1/2],
  [[e],[e],[3],[3],1/2],
  [[e],[e],[4],[4],1/2],
  [[t],[e],[1],[2],1/2],
  [[e],[t],[2],[3],1/2],
  [[r],[e],[3],[4],1/2],
  [[e],[r],[4],[1],1/2]
  )):
```

Similarly, **Fork** is:

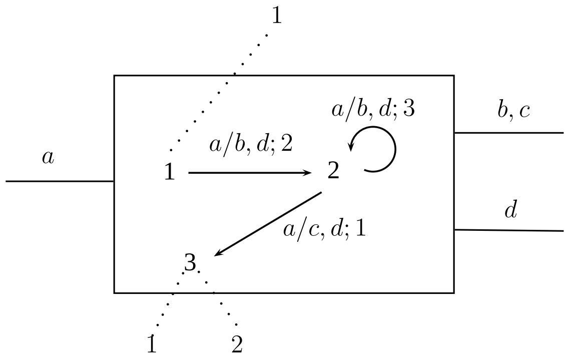
```
> fork:=sys(stat([1],[2],[3]),trans(
  [[e],[e],[1],[1],1/3],
  [[e],[e],[2],[2],1/2],
  [[e],[e],[3],[3],1/2],
  [[t],[e],[1],[3],1/3],
  [[r],[e],[3],[1],1/2],
  [[e],[t],[1],[2],1/3],
  [[e],[r],[2],[1],1/2])):
```

More in general, a weighted automaton $\mathbf{Q}^{X}_{Y;A,B}$ may be represented as a list

```
sys(state(),trans(),incond(),outcond())
```

where `state()` and `trans()` are lists of states and transitions respectively as described above, and `incond()` and `outcond()` are lists of integers representing the images of the in and out condition functions into the state space.

Example A.1.2. Consider the weighted automaton **Q** represented by the following picture:



Then, **Q** is:

```
> Q:=sys(
  stat([1],[2],[3]),
  trans(
  [[a],[b,d],[1],[2],2],
  [[a],[b,d],[2],[2],3],
  [[a],[c,d],[2],[3],1],
  ),
  incond(1),
  outcond(3,3)
  ):
```

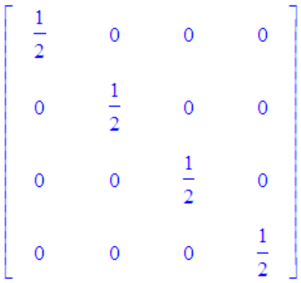
Since the automata we consider have trivial sequential interfaces, for simplicity in the following sections we will ignore the in and out conditions.

A.2 Transition matrices

We define now the procedure `transmatr` which returns the transition matrix of a weighted automaton `sys1` with left label `llab` and right label `rlab`. `transmatr` recalls the auxiliary procedures `ord`, which returns the index position of a state, and `eqlist` which returns *true* when two given lists are equal.

```
> ord:=proc(sys,state)
  local n,k,i;
  n:=nops(op(1,sys));
  k:=0;
  for i from 1 to n do
  if op(i,op(1,sys))=state then
  k:=i;
  fi;
  od;
  RETURN(k);
  end:

> eqlist:=proc(list1,list2)
  local n,m,i;
  n:=nops(list1);
  m:=nops(list2);
  i:=1;
  if not(n=m) then RETURN(false);fi;
  for i from 1 to n do
  if not(op(i,list1)=op(i,list2)) then RETURN(false);fi;
  i:=i+1;
  od;
  RETURN(true);
  end:

> transmatr:=proc(sys,llab,rlab)
  local n,t,l,A,i,h,k,p,q;
  n:=nops(op(1,sys)); # no of states of sys
  t:=op(2,sys); #transition vector
  l:=nops(t);#no of transitions
  A:=matrix(n,n,0);
  for i from 1 to l do;
  if ( eqlist(op(1,op(i,t)),llab) ) and
  (eqlist(op(2,op(i,t)),rlab) ) then
  h:=op(3,op(i,t));
  k:=op(4,op(i,t));
  p:=ord(sys,h);
  q:=ord(sys,k);
  A[p,q]:=A[p,q]+op(5,op(i,t));
  fi;
  od;
  RETURN(evalm(A));
  end:
```

Example A.2.1. The command `transmatr(phil,[e],[e]);` returns the following matrix:



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

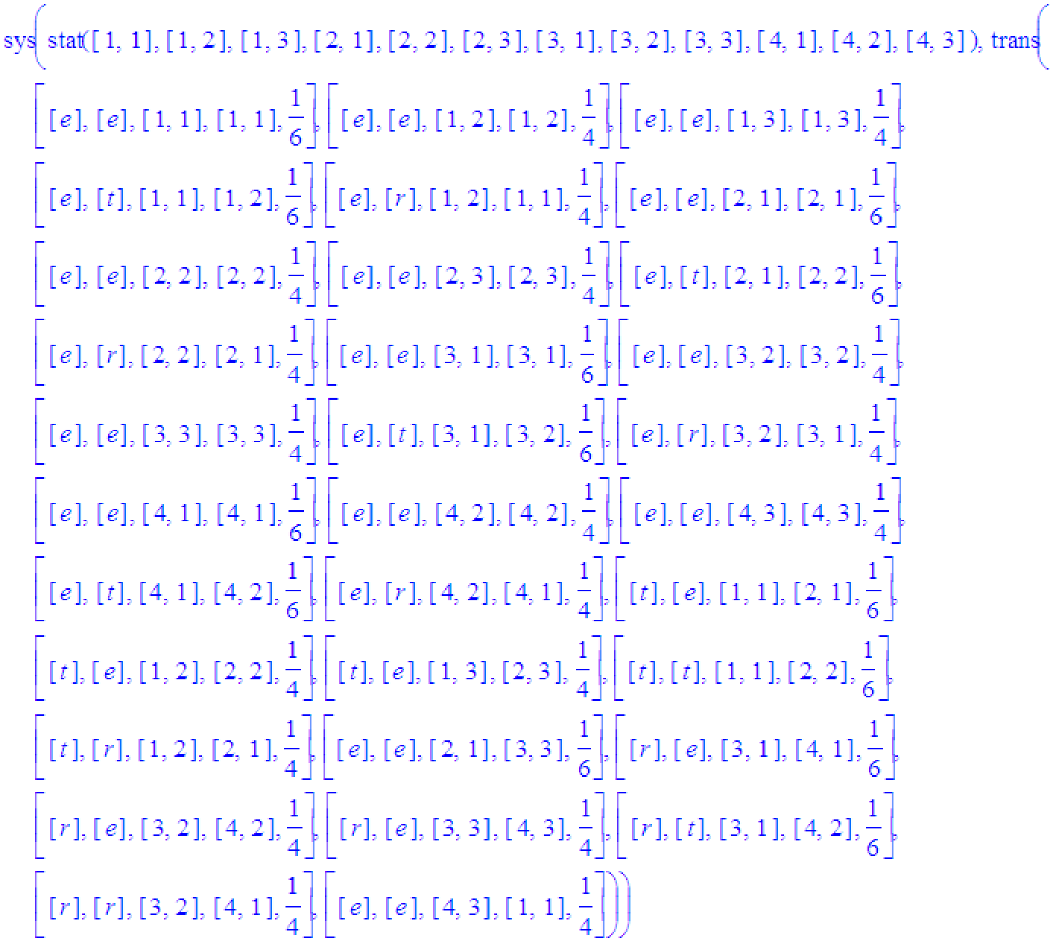
A.3 Operations

A.3.1 Series composition

Given two weighted automata sys1, sys2, the procedure ser(sys1,sys2) returns the series composite of sys1 and sys2.

```
> serstat:=proc(sys1,sys2) local l,i,j,n1,n2;
  n1:=nops(op(1,sys1));
  n2:=nops(op(1,sys2));
  l:=stat();
  for i from 1 to n1 do;
  for j from 1 to n2 do;
  l:=stat(op(l),[op(op(i,op(1,sys1))),op(op(j,op(1,sys2)))]);
  od;od;
  RETURN(l);
  end:

> sertrans:=proc(sys1,sys2) local i,j,l,n1,n2,t1,t2;
  n1:=nops(op(2,sys1));
  n2:=nops(op(2,sys2));
  l:=trans();
  t1:=op(2,sys1);
  t2:=op(2,sys2);
  for i from 1 to n1 do;
  for j from 1 to n2 do;
  if op(2,op(i,t1))=op(1,op(j,t2)) then
  l:=trans(
  op(l),
  [ op(1,op(i,t1)),op(2,op(j,t2)),
  [op(op(3,op(i,t1))),op(op(3,op(j,t2)))],
  [op(op(4,op(i,t1))),op(op(4,op(j,t2)))],
  op(5,op(i,t1))*op(5,op(j,t2))  ]);
  fi;
  od;od;
  RETURN(l);
  end:

> ser:=proc(sys1,sys2);
  RETURN(sys(serstat(sys1,sys2),sertrans(sys1,sys2)));
  end:
```

Example A.3.1. The command `ser(phil,fork)` returns the following automaton:



$$
\begin{gathered}
sys\Big(stat([1,1],[1,2],[1,3],[2,1],[2,2],[2,3],[3,1],[3,2],[3,3],[4,1],[4,2],[4,3]),\ trans\Big(\\
\left[[e],[e],[1,1],[1,1],\tfrac{1}{6}\right],\left[[e],[e],[1,2],[1,2],\tfrac{1}{4}\right],\left[[e],[e],[1,3],[1,3],\tfrac{1}{4}\right],\\
\left[[e],[t],[1,1],[1,2],\tfrac{1}{6}\right],\left[[e],[r],[1,2],[1,1],\tfrac{1}{4}\right],\left[[e],[e],[2,1],[2,1],\tfrac{1}{6}\right],\\
\left[[e],[e],[2,2],[2,2],\tfrac{1}{4}\right],\left[[e],[e],[2,3],[2,3],\tfrac{1}{4}\right],\left[[e],[t],[2,1],[2,2],\tfrac{1}{6}\right],\\
\left[[e],[r],[2,2],[2,1],\tfrac{1}{4}\right],\left[[e],[e],[3,1],[3,1],\tfrac{1}{6}\right],\left[[e],[e],[3,2],[3,2],\tfrac{1}{4}\right],\\
\left[[e],[e],[3,3],[3,3],\tfrac{1}{4}\right],\left[[e],[t],[3,1],[3,2],\tfrac{1}{6}\right],\left[[e],[r],[3,2],[3,1],\tfrac{1}{4}\right],\\
\left[[e],[e],[4,1],[4,1],\tfrac{1}{6}\right],\left[[e],[e],[4,2],[4,2],\tfrac{1}{4}\right],\left[[e],[e],[4,3],[4,3],\tfrac{1}{4}\right],\\
\left[[e],[t],[4,1],[4,2],\tfrac{1}{6}\right],\left[[e],[r],[4,2],[4,1],\tfrac{1}{4}\right],\left[[t],[e],[1,1],[2,1],\tfrac{1}{6}\right],\\
\left[[t],[e],[1,2],[2,2],\tfrac{1}{4}\right],\left[[t],[e],[1,3],[2,3],\tfrac{1}{4}\right],\left[[t],[t],[1,1],[2,2],\tfrac{1}{6}\right],\\
\left[[t],[r],[1,2],[2,1],\tfrac{1}{4}\right],\left[[e],[e],[2,1],[3,3],\tfrac{1}{6}\right],\left[[r],[e],[3,1],[4,1],\tfrac{1}{6}\right],\\
\left[[r],[e],[3,2],[4,2],\tfrac{1}{4}\right],\left[[r],[e],[3,3],[4,3],\tfrac{1}{4}\right],\left[[r],[t],[3,1],[4,2],\tfrac{1}{6}\right],\\
\left[[r],[r],[3,2],[4,1],\tfrac{1}{4}\right],\left[[e],[e],[4,3],[1,1],\tfrac{1}{4}\right]\Big)\Big)
\end{gathered}
$$

A.3.2 Parallel feedback

The following procedure computes the parallel feedback of a system `sys1`:

```
> feedback:=proc(sys1) local i,n,n1,n2,l;
  l:=trans();
  n:=nops(op(2,sys1));#no of transitions
  n1:=nops(op(1,op(1,op(2,sys1))));#no of labels on left
  n2:=nops(op(2,op(1,op(2,sys1))));#no of labels on right
  for i from 1 to n do;
  if op(n1,op(1,op(i,op(2,sys1))))=op(n2,op(2,op(i,op(2,sys1))))
  then l:=trans(op(l),[[op(1..n1-1,op(1,op(i,op(2,sys1))))],
  [op(1..n2-1,op(2,op(i,op(2,sys1))))],op(3,op(i,op(2,sys1))),
  op(4,op(i,op(2,sys1))),op(5,op(i,op(2,sys1)))]);
  fi;od;
  RETURN(sys(op(1,sys1),l));
  end:
```

A.3.3 Reachability

The procedure `reach(x,sys1)` computes the reachable part of a weighted automaton `sys1` with initial state `x` and recalls some auxiliary procedures.

```
> reachinone:=proc(x,sys) local i,l,t;
  #returns list reachable from state x plus x
  l:=stat(x);
  for i from 1 to nops(op(2,sys)) do;
  t:=op(i,op(2,sys));#i th transition
  if op(3,t)=x then l:=adjoin(op(4,t),l);
  fi;od;
  RETURN(l);
  end:

> reachinonelist:=proc(list,sys) local i, l;
  #returns the list of  states reachable in one step
  #from the list l of states plus states in l;
  l:=list;
  for i from 1 to nops(list) do;
  l:=adjoinlist(reachinone(op(i,l),sys),l);
  od;end:

> reachstat:=proc(x,sys) local l,i,m,n;
  l:=stat(x);n:=-1;m:=0;
  i:=0;while(n<m) do;
  l:=reachinonelist(l,sys);i:=i+1;
  n:=m;
  m:=nops(l);
  od;
  RETURN(l);
  end:

> reachtrans:=proc(x,sys1) local i,j,m,n,ls,lt;
  #returns the transitions reachable from x
  ls:=reachstat(x,sys1);
  lt:=trans();
  m:=nops(op(2,sys1));#number of transitions
  n:=nops(ls);#number of reachable states
  for i from 1 to m do;
  for j from 1 to n do;
  if op(3,op(i,op(2,sys1)))=op(j,ls)
  then lt:=adjoin(op(i,op(2,sys1)),lt);fi;
  od;
  od;
  RETURN(trans(op(lt)));
  end:

> reach:=proc(x,sys1);
  RETURN(sys(reachstat(x,sys1),reachtrans(x,sys1)));
  end:
```

A.3.4 Normalization

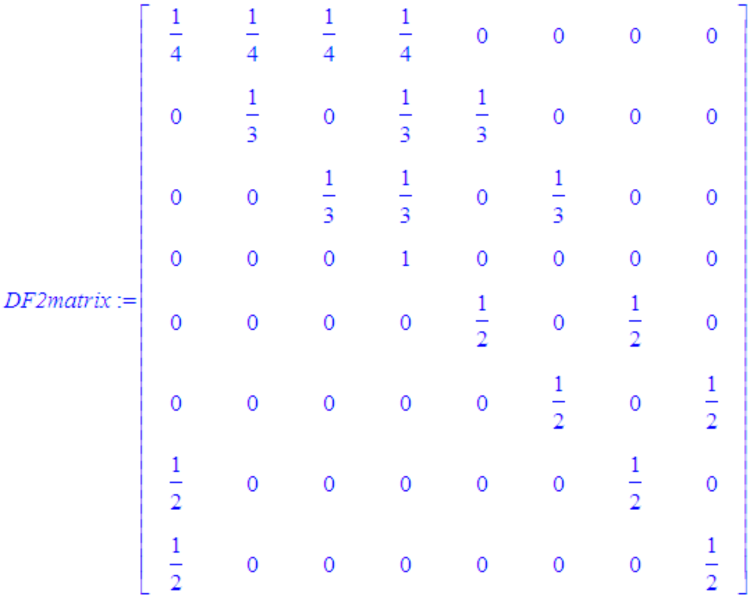
Given a real matrix `A`, `normalize(A)` returns the normalized matrix of `A`.

```
> rowsum:=proc(A,i) local j,x,y;
  x:=0;
  for j from 1 to linalg[coldim](A) do;
  x:=x+A[i,j];od;
  RETURN(x);
  end:

> normalizerow:=proc(A,i) local j,x;
  x:=rowsum(A,i);
  for j from 1 to linalg[coldim](A) do;
  A[i,j]:=A[i,j]/x;
  od;
  RETURN(A);
  end:

> normalize:=proc(A) local i;
  for i from 1 to linalg[rowdim](A) do;
  normalizerow(A,i);
  od;
  RETURN(A);
  end:
```

A.4 Examples

The transition matrix of **DP**$_n$ with $n = 2$ is given by:

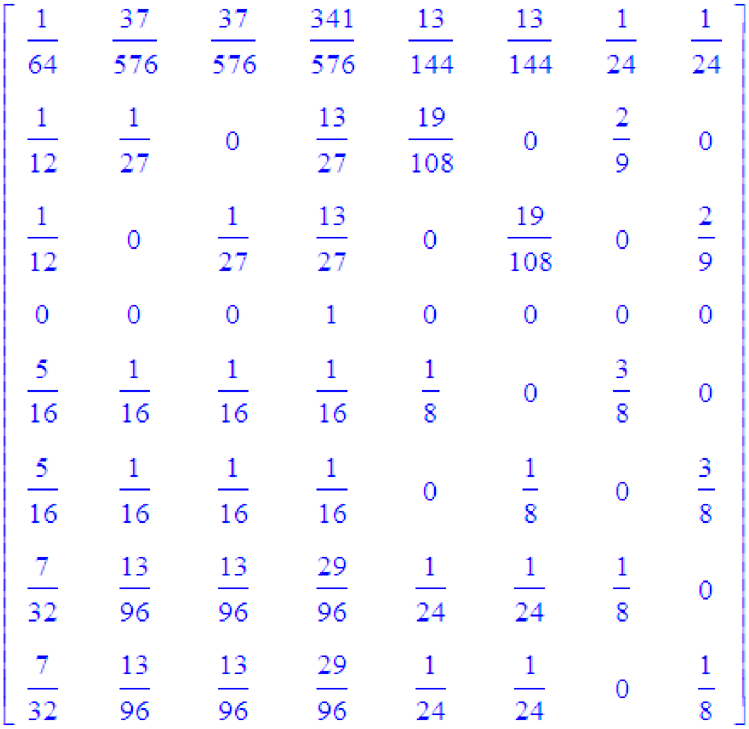
```
> DF2:=reach([1,1,1,1],feedback(ser(ser(phil,fork),ser(phil,fork)))):
> DF2matrix:=normalize(transmatr(DF2,[],[]));
```



$$DF2matrix := \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

After 3 steps the probability of reaching deadlock (fourth column of the matrix) from the initial state `[1,1,1,1]` (first row) is $\frac{341}{576}$:

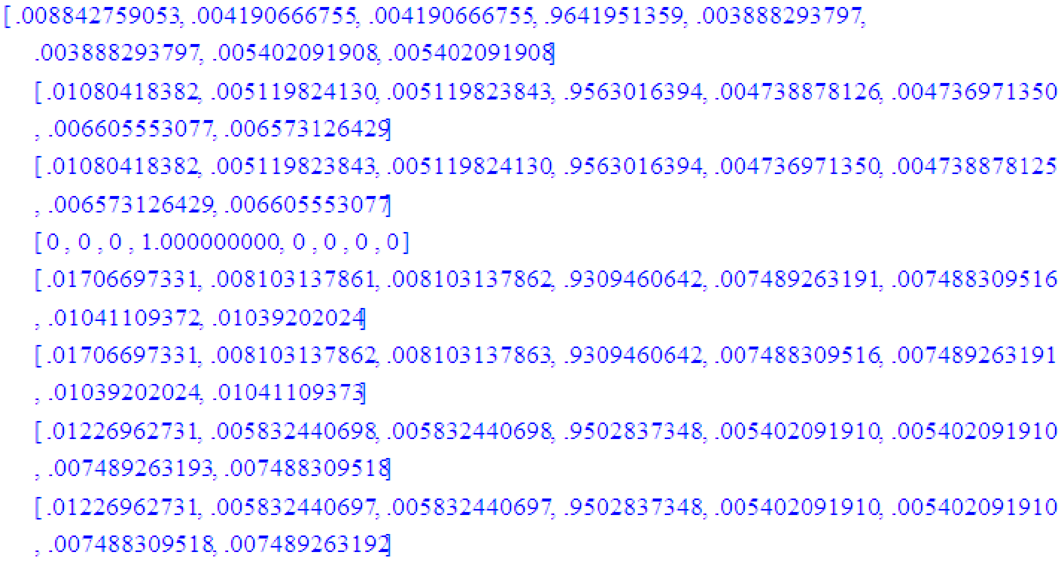
```
> evalm(DF2matrix^3);
```



$$\begin{bmatrix} \frac{1}{64} & \frac{37}{576} & \frac{37}{576} & \frac{341}{576} & \frac{13}{144} & \frac{13}{144} & \frac{1}{24} & \frac{1}{24} \\ \frac{1}{12} & \frac{1}{27} & 0 & \frac{13}{27} & \frac{19}{108} & 0 & \frac{2}{9} & 0 \\ \frac{1}{12} & 0 & \frac{1}{27} & \frac{13}{27} & 0 & \frac{19}{108} & 0 & \frac{2}{9} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{5}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{8} & 0 & \frac{3}{8} & 0 \\ \frac{5}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & 0 & \frac{1}{8} & 0 & \frac{3}{8} \\ \frac{7}{32} & \frac{13}{96} & \frac{13}{96} & \frac{29}{96} & \frac{1}{24} & \frac{1}{24} & \frac{1}{8} & 0 \\ \frac{7}{32} & \frac{13}{96} & \frac{13}{96} & \frac{29}{96} & \frac{1}{24} & \frac{1}{24} & 0 & \frac{1}{8} \end{bmatrix}$$

After 20 steps the probability of deadlock is 0.9641951359:

```
> evalf(evalm(DF2matrix^20),10);
```



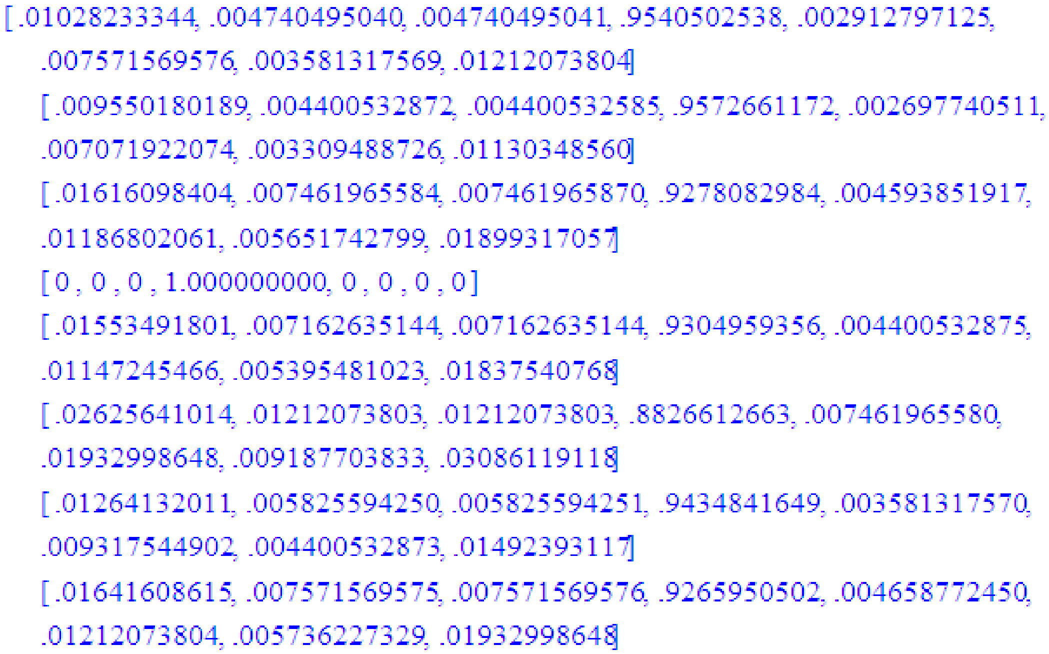
```
[.008842759053, .004190666755, .004190666755, .9641951359, .003888293797,
    .003888293797, .005402091908, .005402091908]
    [.01080418382, .005119824130, .005119823843, .9563016394, .004738878126, .004736971350
    , .006605553077, .006573126429]
    [.01080418382, .005119823843, .005119824130, .9563016394, .004736971350, .004738878125
    , .006573126429, .006605553077]
    [0 , 0 , 0 , 1.000000000, 0 , 0 , 0 , 0]
    [.01706697331, .008103137861, .008103137862, .9309460642, .007489263191, .007488309516
    , .01041109372, .01039202024]
    [.01706697331, .008103137862, .008103137863, .9309460642, .007488309516, .007489263191
    , .01039202024, .01041109373]
    [.01226962731, .005832440698, .005832440698, .9502837348, .005402091910, .005402091910
    , .007489263193, .007488309518]
    [.01226962731, .005832440697, .005832440697, .9502837348, .005402091910, .005402091910
    , .007488309518, .007489263192]
```

In an analogous way, we can calculate $\mathbf{DP}_n$ for $n > 2$. For instance, for $n = 3$ we can calculate:

```
> DF3:=reach([1,1,1,1,1,1],feedback(ser(ser(ser(phil,fork),
  ser(phil,fork)),ser(phil,fork)))):
> DF3matrix:=normalize(transmatr(DF3,[],[])):
```

Then, we change the weights in $\mathbf{DP}_2$ and we compute the transition matrix of the system after 20 steps:

```
> phil1:=sys(stat([1],[2],[3],[4]),trans(
  [[e],[e],[1],[1],1],
  [[e],[e],[2],[2],3],
  [[e],[e],[3],[3],.5],
  [[e],[e],[4],[4],.5],
  [[t],[e],[1],[2],1],
  [[e],[t],[2],[3],3],
  [[r],[e],[3],[4],.5],
  [[e],[r],[4],[1],.5]
  )):
> fork1:=sys(stat([1],[2],[3]),trans(
  [[e],[e],[1],[1],.5],
  [[e],[e],[2],[2],.5],
  [[e],[e],[3],[3],1],
  [[t],[e],[1],[3],.5],
  [[r],[e],[3],[1],.5],
  [[e],[t],[1],[2],.5],
  [[e],[r],[2],[1],1])):
> phil2:=sys(stat([1],[2],[3],[4]),trans(
  [[e],[e],[1],[1],1],
  [[e],[e],[2],[2],3],
  [[e],[e],[3],[3],5],
  [[e],[e],[4],[4],.5],
  [[t],[e],[1],[2],1],
  [[e],[t],[2],[3],3],
  [[r],[e],[3],[4],5],
  [[e],[r],[4],[1],.5]
  )):
> fork2:=sys(stat([1],[2],[3]),trans(
  [[e],[e],[1],[1],5],
  [[e],[e],[2],[2],2],
  [[e],[e],[3],[3],1],
  [[t],[e],[1],[3],5],
  [[r],[e],[3],[1],2],
  [[e],[t],[1],[2],5],
  [[e],[r],[2],[1],1])):
> DF2a:=reach([1,1,1,1],feedback(ser(ser(phil1,fork1),
  ser(phil2,fork2)))):
> DF2amatrix:=normalize(transmatr(DF2a,[],[])):
> evalm(DF2amatrix^20):
```



```
[.01028233344, .004740495040, .004740495041, .9540502538, .002912797125,
   .007571569576, .003581317569, .01212073804]
   [.009550180189, .004400532872, .004400532585, .9572661172, .002697740511,
   .007071922074, .003309488726, .01130348560]
   [.01616098404, .007461965584, .007461965870, .9278082984, .004593851917,
   .01186802061, .005651742799, .01899317057]
   [0 , 0 , 0 , 1.000000000, 0 , 0 , 0 , 0]
   [.01553491801, .007162635144, .007162635144, .9304959356, .004400532875,
   .01147245466, .005395481023, .01837540768]
   [.02625641014, .01212073803, .01212073803, .8826612663, .007461965580,
   .01932998648, .009187703833, .03086119118]
   [.01264132011, .005825594250, .005825594251, .9434841649, .003581317570,
   .009317544902, .004400532873, .01492393117]
   [.01641608615, .007571569575, .007571569576, .9265950502, .004658772450,
   .01212073804, .005736227329, .01932998648]
```

With other similar calculations we note that the probability of reaching the deadlock state grows with increasing numbers of steps independently of the choices of weights[3].

[3] See remark 1.4.4 p. 41.

Bibliography

[1] S. Abramsky and B. Coecke. A Categorical Semantics of Quantum Protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 415–425. IEEE Computer Society, 2004.

[2] A. Arnold. *Finite transition systems: semantics of communicating systems*. Prentice Hall International (UK), 1994.

[3] C. Baier, M. Grösser, and F. Ciesinski. Model checking linear-time properties of probabilistic systems. *Monographs in Theoretical Computer Science*, pages 519–596, 2009.

[4] A. Bellos. Magic numbers: A meeting of mathemagical tricksters. *New Scientist*, 2762:44–49, 2010.

[5] J. Benabou. Introduction to Bicategories. *Springer Lecture Notes in Mathematics*, 1967.

[6] C.H. Bennett, G. Brassard, C. Crépeau adn R. Jozsa, A. Peres, and W. K. Wootters. Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels. *Phys. Rev. Lett.*, 70:1895–1899, 1993.

[7] A. Blass. Seven trees in one. *Journal of Pure and Applied Algebra*, 103:1–21, 1995.

[8] S.L. Bloom and Z.Ésik. *Iteration Theories: The Equational Logic of Iterative Processes*. EATCS Monographs on Theoretical Computer Science. Springer–Verlag, 1993.

[9] R. Blute, A. Edalat, and P. Panangaden. Bisimulation for Labelled Markov Processes. *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, pages 95–106, 1997.

[10] A. Carboni, G. M. Kelly, R. F. C Walters, and R.J. Wood. Cartesian Bicategories II. *Theory and Applications of Categories*, 19(6):93–124, 2008.

[11] A. Carboni, S. Lack, and R.F.C. Walters. Introduction to extensive and distributive categories. *Journal of Pure and Applied Algebra*, 84:145–158, 1993.

[12] A. Carboni and R.F.C. Walters. Cartesian bicategories I. *Journal of Pure and Applied Algebra*, 49:11–32, 1987.

[13] J.R.B. Cockett and S. Lack. The extensive completion of a distributive category. *Theory and Applications of Categories*, 8:541–554, 2001.

[14] B. Coecke, E. O Paquette, and D. Pavlovic. Classical and Quantum Structures. Technical Report RR-08-02, OUCL, 2008.

[15] B. Coecke and D. Pavlovic. Quantum measurements without sums. In G. Chen, L. Kauffman, and S. Lamonaco, editors, *Mathematics of Quantum Computing and Technology.* Taylor and Francis, 2006.

[16] B. Coecke and S. Perdrix. Environment and classical channels in categorical quantum mechanics. *ArXiv:1004.1598*, 2010.

[17] L. de Francesco Albasini, N. Sabadini, and R.F.C. Walters. The parallel composition of processes. *ART 2008, Analysing Reduction systems using Transition systems*, pages 111–121, 2008. (Also arXiv:0904.3961).

[18] L. de Francesco Albasini, N. Sabadini, and R.F.C. Walters. Systems with discrete geometry. *ART 2008, Analysing Reduction systems using Transition systems*, pages 122–131, 2008.

[19] L. de Francesco Albasini, N. Sabadini, and R.F.C. Walters. An algebra of automata which includes both classical and quantum entities. *To appear in Proceedings 6th QPL workshop, Quantum Physics and Logic, ENTCS*, 2009. (Preliminary version arXiv:0901.4754, 2009).

[20] L. de Francesco Albasini, N. Sabadini, and R.F.C. Walters. Cospans and spans of graphs: a categorical algebra for the sequential and parallel composition of discrete systems. *ArXiv:0909.4136*, 2009.

[21] L. de Francesco Albasini, N. Sabadini, and R.F.C. Walters. The compositional construction of Markov processes. *Applied Categorical Structures*, 2010. Presented to International Conference in Category Theory,

University of Cape Town, 29 June - 4 July, 2009. (Preliminary version arXiv:0901.2434, 2009).

[22] L. de Francesco Albasini, N. Sabadini, and R.F.C. Walters. The compositional construction of Markov processes II. *To appear in RAIRO-Theoretical Informatics and applications*, 2010. Presented to ICTCS '09, 11th Italian Conference on Theoretical Computer Science, Cremona, Italy. (Preliminary version arXiv:1005.0949, 2010).

[23] L. de Francesco Albasini, N. Sabadini, and R.F.C. Walters. Weighted automata and CospanSpan. Presented to International Conference in Category Theory, Università di Genova, available as: http://ct2010.disi.unige.it/slides/Walters_CT2010.pdf, June 20-26 2010.

[24] M. Droste, W. Kuich, and H. Vogler(Eds.). *Handbook of Weighted Automata*. EATCS Monographs in Theoretical Computer Science. EATCS, 2009.

[25] S. Eilenberg. *Automata, Languages, and Machines (Volume A)*. Academic Press, San Diego, CA, 1974.

[26] C.C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium 1973: proceedings of the Logic Colloquium, Bristol, July 1973*, volume 80, pages 175–230. North Holland Publishers, 1975.

[27] M. Gardner. *The Second Scientific American Book of Mathematical Puzzles and Diversions*. University of Chicago Press, Chicago. First appeared in 1961, Simon and Schuster, NY, 1987.

[28] M. Gardner. Mathematical games, Jan. 1957–Dec. 1981. A column in Scientific American; all 15 volumes of the column are on 1 CD-ROM, available from MAA.

[29] R. J. Van Glabbeek, S. A. Smolka, and B. Steffen. Reactive, Generative and Stratified Models of Probabilistic Processes. *Information and Computation*, 121(1):59–80, 1995.

[30] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer, 2001.

[31] D. Harel. Statecharts: a visual formalism for complex systems. *The Science of Computer Programming*, 8:231–274, 1987.

[32] M. Hasegawa. On traced monoidal closed categories. *Mathematical Structure in Computer Science*, 2009.

[33] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[34] C. A. R. Hoare, editor. *Communicating Sequential Processes.* Prentice-Hall, New York, 1985.

[35] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Second edition, Addison Wesley, 2001.

[36] A. Joyal and R. Street. Braided tensor categories. *Advances in Mathematics*, 102(1):20–78, 1993.

[37] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(03):447–468, 1996.

[38] P. Katis, N. Sabadini, and R.F.C. Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, 115:141–178, 1997.

[39] P. Katis, N. Sabadini, and R.F.C. Walters. Representing P/T nets in Span(Graph). *Proceedings AMAST '97*, 1349:307–321, 1997.

[40] P. Katis, N. Sabadini, and R.F.C. Walters. Span(Graph): A categorical algebra of transition systems. In *Proc. AMAST '97*, volume 1349 of *LNCS*, pages 307–321. Springer Verlag, 1997.

[41] P. Katis, N. Sabadini, and R.F.C. Walters. A formalisation of the IWIM Model. In A. Porto and G.-C. Roman, editors, *COORDINATION 2000*, volume 1906 of *LNCS*, pages 267–283. Springer Verlag, 2000.

[42] P. Katis, N. Sabadini, and R.F.C. Walters. On the algebra of systems with feedback and boundary. *Rendiconti del Circolo Matematico di Palermo Serie II, Suppl. 63*, pages 123–156, 2000.

[43] P. Katis, N. Sabadini, and R.F.C. Walters. Feedback, trace and fixed-point semantics. *Theoret. Informatics Appl. 36*, pages 181–194, 2002.

[44] G.M. Kelly and M. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, 1980.

[45] J. Kock. *Frobenius algebras and 2D topological Quantum Field Theories.* Cambridge University Press, 2004.

[46] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *Proc. of the 38th Annual Symposium on Foundations of Computer Science*, pages 66–75, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

[47] F.W. Lawvere. Functorial Semantics of Elementary Theories. *Journal of Symbolic Logic*, 31:294–295, 1966.

[48] F.W. Lawvere. Some remarks on the future of category theory. In *Proceedings Category Theory 1990*, volume 1488, pages 1–13. Springer Verlag, 1991.

[49] Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. Observing Branching Structure through Probabilistic Contexts. *SIAM J. Comput.*, 37(4):977–1013, 2007.

[50] W.S. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[51] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer Berlin / Heidelberg, 1980.

[52] Mehryar Mohri. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of WeightedAutomata*, EATCS Monographs in Theoretical Computer Science, pages 213–254. Springer, 2009.

[53] C. Moore and J. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237:275–306, 2000.

[54] R. Penrose. Applications of negative dimensional tensors. *Combinatorial Mathematics and its Applications*, pages 221–244, 1971.

[55] A. Pnueli and L. Zuck. Probabilistic Verification by Tableaux. In *Proceedings LICS'86*, pages 322–331, 1986.

[56] M. O. Rabin. Probabilistic Automata. *Information andControl*, 6:230–245, 1963.

[57] R. Rosebrugh, N. Sabadini, and R.F.C. Walters. Generic commutative separable algebras and cospans of graphs. *Theory and Applications of Categories*, 15:264–177, 2005.

[58] R. Rosebrugh, N. Sabadini, and R.F.C. Walters. Calculating colimits and limits compositionally. *presented to Category Theory 2007, Carvoeiro, Portugal, 18th June 2007*, 2007.

[59] R. Rosebrugh, N. Sabadini, and R.F.C. Walters. Calculating colimits compositionally. *Montanari Festschrift*, 5065:581–592, 2008. (Also arXiv:0712.2525).

[60] N. Sabadini, S. Vigna, and R.F.C. Walters. A note on recursive functions. *Mathematical Structures in Computer Science*, 6:127–139, 1996.

[61] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, May 1995. Also, Technical Report MIT/LCS/TR-676.

[62] P. Selinger. Towards a quantum programming language. *Mathematical. Structures in Comp. Sci.*, 14(4):527–586, 2004.

[63] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In *Handbook of weighted automata*, volume 2925 of *EATCS Monographs in Theoretical Computer Science*, pages 1–43. Springer Verlag, 2004.

[64] G. Stefanescu. *Network Algebra.* Springer-Verlag, 2000.

[65] V. Turaev. *Quantum Invariants of Knots and 3-Manifolds.* De Gruyter, 2 edition, 2010.

[66] M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings FOCS'85*, pages 327–338, 1985.

[67] S. Vigna. On the Relations between Distributive Computability and the BS. *Model. Theor. Comput. Sci.*, 162(1):5–21, 1996.

[68] R.F.C. Walters. *Categories and Computer Science.* Carslaw Publications 1991, Cambridge University Press 1989, 1992.

[69] R.F.C. Walters and R.J. Wood. Frobenius Objects in Cartesian Bicategories. *Theory and Applications of Categories*, 20:25–47, 2008. (Also arXiv:0708.1925).

[70] W. Zielonka. Notes on Finite Asynchronous Automata. *Informatique Théorique et Applications (ITA)*, 21(2):99–135, 1987.