RAIRO-Theor. Inf. Appl. 45 (2011) 117–142
DOI: 10.1051/ita/2011015 Available online at:
www.rairo-ita.org

THE COMPOSITIONAL CONSTRUCTION OF MARKOV PROCESSES II

L. de Francesco Albasini[1], N. Sabadini[2] and R.F.C. Walters[2]

Abstract. We add sequential operations to the categorical algebra of weighted and Markov automata introduced in [L. de Francesco Albasini, N. Sabadini and R.F.C. Walters, `arXiv:0909.4136`]. The extra expressiveness of the algebra permits the description of hierarchical systems, and ones with evolving geometry. We make a comparison with the probabilistic automata of Lynch *et al.* [*SIAM J. Comput.* **37** (2007) 977–1013].

Mathematics Subject Classification. 18B20, 60J10, 18D10, 05C22.

1. Introduction

In [9] we introduced a notion of Markov automaton, together with parallel operations which permit the compositional description of Markov processes. We illustrated by showing how to describe a system of n dining philosophers (with 12^n states), and we observed that Perron-Frobenius theory yields a proof that the probability of reaching deadlock tends to one as the number of steps goes to infinity. In this paper we add sequential operations to the algebra (and the necessary structure to support them) following analogous developments in [10,19]. The extra operations add considerable expressiveness to the algebra since the sequential and parallel operations may be alternated to permit the description of hierarchical systems, and ones with evolving geometry. We illustrate our algebra by describing a system called Sofia's Birthday Party, originally introduced in [19], and a probabilistic fork bomb.

Keywords and phrases. Categorical algebra, Markov process, weighted automaton, hierarchical, distributed.

[1] Dipartimento di Fisica e Matematica, Università degli Studi dell'Insubria, via Valleggio 11, 22100 Como, Italy; `luisa.dfa@libero.it`

[2] Dipartimento di Informatica e Comunicazione, Università degli Studi dell'Insubria, via Carloni 78, 22100 Como, Italy; `{nicoletta.sabadini};{robert.walters}@uninsubria.it`

The introduction of *sequential operations* to the algebra of [9] requires extra structure to be added to the automata, namely sequential interfaces. Further, in [10] weighted automata (where the weighting of a transition is a non-negative real number) 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 [11].

To see how hierarchical and mobile systems may be modelled in this algebra, using the combined sequential and parallel operations, consider a set of automata S and let us denote the set of automata given as expressions in terms of parallel operations in the automata of 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 only one transition. Then any automaton has a representation in $\Sigma(E)$. The dining philosopher problem of [9] is described as an element of $\Pi\Sigma(E)$, that is of communicating sequential systems. An element of $\Sigma\Pi\Sigma(E)$ is one in which the parallel structure may evolve, and so on. The system Sofia's Birthday Party is in $\Pi\Sigma\Sigma(E)$ but illustrates also the form of systems of type $\Pi\Sigma\Pi\Sigma(E)$. The version of a fork bomb which we describe belongs to $\Sigma(\Pi\Sigma)^n$. There is a close relation between this aspect of the algebra and the state charts of [13].

1.1. Comparison with other models

There is a huge literature on probabilistic and weighted automata, transducers, and process calculi (see for example, [4,11,15,22,25,27,29]). However the model of [9] and of this paper is distinguished from the others in the following ways:

(i) In many other probabilistic automata models [1,25] 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 is instead *generative* in the sense of [31] 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 [31] is perhaps slightly different since they also speak of "pushing buttons".

(ii) The actual origins of this paper are the work of Eilenberg [12] on weighted transducers – our operations are variations on those introduced there – modified by further category theoretic experience, beginning, for example, with [5,18]. We view the automata and the operations on them rather differently from [12]. 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 [10] 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 [23] and CSP [16], with interleaving semantics and underlying broadcast topology. Instead, our algebra models truly concurrent systems with explicit network topologies. We have shown in [8] 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. Recently, directly inspired by our work, Sobocinski [28] has introduced our wire calculus, and our parallel composition, into process 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 in this paper 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 [10]. 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 [17], and hence has close relations with the work of Stefanescu [30] on network algebras, and the theory of fix point operators studied by many authors including Bloom and Ésik [3] – connections are described in [14]. The diagrams we introduce are related to those of topological field theory described, for example, in [20], and also those of the diagrammatic approach to quantum mechanics described for example in [6,7].

(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 in this paper 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 *et al.* in [22,27] fit naturally in our context and we give a simplified description of their behaviour in terms of the operations of our algebra.

(vi) In our model, the communicating parallel composition involves conditional probability. 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 4.3.3 of [27] argues against such a parallel composition.

2. 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 [10] differ 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*.

The reader is advised in reading the following definition to keep in mind the explicit example in Section 2.1.1. The main intuition is that in order to develop a compositional calculus of automata networks each component must be an open system with communication interfaces. Conventionally we distinguish these parallel interfaces into left and right interfaces A and B respectively, and it will be crucial in the following that A and B may be products of sets. The transitions between states that are traditionally in automata theory labelled in an alphabet Σ are here instead labelled by pairs of elements in $A \times B$ representing the effect of the transition on the parallel interfaces. Technically we make the definitions in terms of matrices rather than automata since this follows the tradition in the theory of Markov chains.

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 2.1. 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, $\gamma_0 : X \to Q$, and $\gamma_1 : Y \to Q$. 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.

We will use a brief notation for the automata $\mathbf{Q}$ indicating its interfaces, namely $\mathbf{Q}^{X}_{Y;A,B}$. We shall 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.

Definition 2.2. A weighted automaton $\mathbf{Q}$ is *positive* if the parallel interfaces A and B contain special elements, 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.

For a positive weighted automaton the total matrix has strictly positive row sums.

Definition 2.3. 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 satisfying the extra condition that the row sums of the total matrix $\mathcal{Q}$ are all 1. That is, for all $q \in Q$

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

For a Markov automaton 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 a *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 2.1.

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

Definition 2.4. Consider a weighted automaton $\mathbf{Q}$ with parallel interfaces A and B. A *behaviour* of length k of $\mathbf{Q}$ consists of two words of length k, one $u = a_1a_2\dots a_k$ in A^* and the other $v = b_1b_2\dots b_k$ in B^* and a sequence of non-negative row vectors

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

Notice that, even for Markov automata, x_i is *not* generally a distribution of states even if x_0 is; for example often x_i is the row vector 0.

Definition 2.5. 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

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

It is obvious that a weighted automaton $\mathbf{Q}$ is Markov if and only if $\mathbf{Q} = \mathbf{N}(\mathbf{Q})$.

Remark 2.6. It is sometimes useful to define the normalization of any weighted automaton, not necessarily positive, by permitting zero rows in the transition matrices. The normalization is then not necessarily Markov.

Definition 2.7. 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 $u = (a_1, a_2, \ldots, a_k) \in A^k$ and $v = (b_1, b_2, \ldots, b_k) \in B^k$ then

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

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

If $\mathbf{Q}$ is a weighted automaton and $u = (a_1, a_2, \ldots, a_k) \in A^k$, $v = (b_1, b_2, \ldots, b_k) \in B^k$, 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.

2.1. Graphical representation of weighted automata

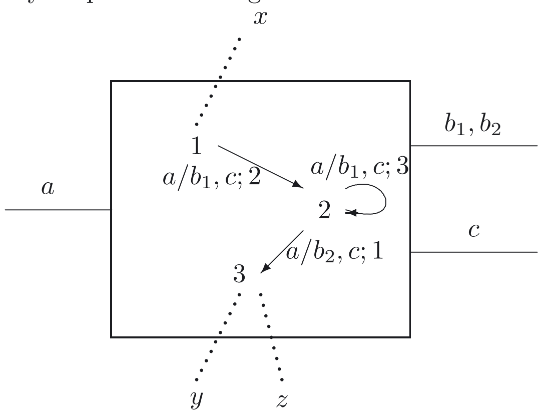
Although the definitions above are mathematically straightforward, in practice a graphical notation is more intuitive. We may compress the description of an automaton with parallel interfaces A and B, which requires $A \times B$ matrices, into a single labelled graph, like the ones introduced in [18]. We indicate by describing some examples.

2.1.1. *An example*

Consider the automaton with parallel interfaces $\{a\}$, $\{b_1, b_2\} \times \{c\}$, 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.

2.1.2. *A philosopher*

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

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

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

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

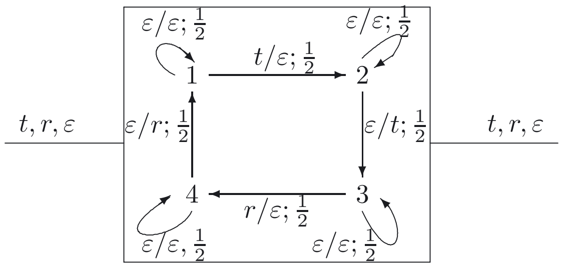
The other four transition matrices are zero matrices.

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 it does a transition to state 2 with probability $\frac{1}{2}$ labelled t, ε (*take the left fork*); in state 2 it does a transition to state 3 with probability $\frac{1}{2}$ labelled ε, t (*take the right fork*); in state 3 it does a transition to state 4 with probability $\frac{1}{2}$ labelled r, ε (*release the left fork*); and in state 4 it 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:


$\varepsilon/\varepsilon;\frac{1}{2}$
$\varepsilon/\varepsilon;\frac{1}{2}$
$t/\varepsilon;\frac{1}{2}$
1
2
t,r,ε
$\varepsilon/r;\frac{1}{2}$
$\varepsilon/t;\frac{1}{2}$
t,r,ε
4
3
$r/\varepsilon;\frac{1}{2}$
$\varepsilon/\varepsilon,\frac{1}{2}$
$\varepsilon/\varepsilon;\frac{1}{2}$

2.1.3. *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

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

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

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

The other four transition matrices are zero.

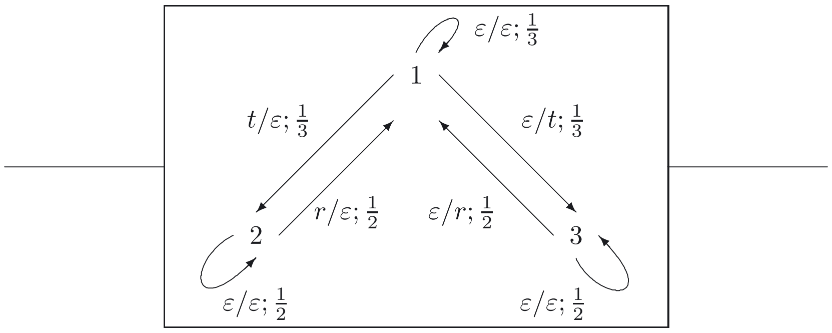
Fork is a Markov automaton since its total matrix is

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

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

3. The algebra of weighted automata: operations

Now we define operations on weighted automata analogous (in a precise sense) to those defined in [18,19].

3.1. Sequential operations

3.1.1. *Sum*

Definition 3.1. 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 interfaces $A+C$, 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.

3.1.2. *Sequential composition*

Definition 3.2. Given weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Y_{Z;C,D}$, the *sequential composite of weighted automata* $\mathbf{Q}{+_Y}\mathbf{R}$ 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

$$\gamma_0 = X \xrightarrow{\gamma_0} Q \rightarrow Q+R \rightarrow (Q+R)/\sim, \; \gamma_1 = Z \xrightarrow{\gamma_1} R \rightarrow Q+R \rightarrow (Q+R)/\sim .$$

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.

3.1.3. *Sequential constants*

Definition 3.3. Given a relation $\rho \subset X \times Y$ and two sets A, B we define a weighted automaton $\mathbf{Seq}(\rho)$ as follows: it has as state space the set of equivalence classes of $X+Y$ under the equivalence relation $\sim$ generated by ρ. The parallel interfaces are A and B, and all matrices are zero 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. Notice that any function $f : X \to Y$ induces a relation on $X+Y$, namely x is related to $f(x)$ for each $x \in X$, and hence any function induces a weighted automaton as above.

Sequential connectors. Some special cases have particular importance and are called *sequential connectors* or *wires*: (i) the automaton corresponding to the identity function $1_X : X \to X$ is also called 1_X; (ii) the automaton corresponding to the codiagonal function $\nabla : X+X \to X$ is called ∇; the automaton corresponding to the opposite relation of ∇ is called ∇^o; (iii) the automaton corresponding to the function $twist : X+Y \to Y+X$ is called $twist_{X,Y}$; (iv) the automaton corresponding to the function $\emptyset \subseteq X$ is called i; the automaton corresponding to the opposite relation of the function $\emptyset \subseteq X$ is called i^o. Notice that we have overloaded 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.

3.2. Parallel operations

3.2.1. *Parallel composition*

Definition 3.4. Given weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Z_{W;C,D}$ the *parallel composite* $\mathbf{Q} \times \mathbf{R}$ is the weighted automaton which has set of states $Q \times R$, left interfaces $A \times C$, 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,

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

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. The following simple lemmas were proved in Section 3 of [9].

Lemma 3.5. *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* $\mathbf{Q} \times \mathbf{R}$.

Lemma 3.6. *If* $\mathbf{Q}$ *and* $\mathbf{R}$ *are Markov automata then* $(\mathbf{Q} \times \mathbf{R})^k = \mathbf{Q}^k \times \mathbf{R}^k$.

3.2.2. *Parallel with communication*

Definition 3.7. Given weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Z_{W;B,C}$ the *communicating parallel composite of weighted automata* $\mathbf{Q}\times_B\mathbf{R}$ (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

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

The following simple lemma was proved in Section 3 of [9].

Lemma 3.8. *If* $\mathbf{Q}$ *and* $\mathbf{R}$ *are positive weighted automata then so is* $\mathbf{Q}||\mathbf{R}$ *and* $\mathbf{N}(\mathbf{N}(\mathbf{Q})||\mathbf{N}(\mathbf{R})) = \mathbf{N}(\mathbf{Q}||\mathbf{R})$.

3.2.3. *Parallel constants*

Definition 3.9. 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 $\mathsf{Par}(\rho)$ is also positive weighted.

Parallel connectors. Some special cases, all described in [18], 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 1_A; (ii) the automaton corresponding to the diagonal function $\Delta : A \to A \times A$ (considered as a relation) is called Δ_A; the automaton corresponding to the opposite relation of Δ is called Δ^o_A; (iii) the automaton corresponding to the function $twist : A \times B \to B \times A$ is again called $twist_{A,B}$; (iv) the automaton corresponding to the projection function $A \to \{*\}$ is called p and its opposite p^o.

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

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

3.3. Some derived operations

3.3.1. *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 $\nabla^o_A||(\mathbf{Q}+\mathbf{R})||\nabla_B$. It has top and bottom interfaces $X+Z$ and $Y+W$, and left and right interfaces A and B.

3.3.2. *Local sequential composition*

Given weighted automata $\mathbf{Q}^X_{Y;A,B}$ and $\mathbf{R}^Y_{Z;A,B}$ the local sequential composite $\mathbf{Q}+_{\mathbf{Y},\mathbf{loc}}\mathbf{R}$, also denoted briefly as $\mathbf{Q}\,;\mathbf{R}$, is defined to be $\nabla^o_A||(\mathbf{Q}+_Y\mathbf{R})||\nabla_B$. It has top and bottom interfaces X and Z, and left and right interfaces A and B.

3.3.3. *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

$$(1_X+_{loc}i_Z)\,;(1_X+_{loc}\nabla^o_Z)\,;(\mathbf{Q}+_{loc}1_Z)\,;(1_Y+_{loc}\nabla_Z)\,;(1_Y+_{loc}i^o_Z).$$

This formula is easier to understand looking ahead at the graphical representation in the next section.

3.3.4. *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

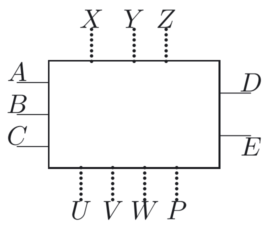
$$(1_A\times p^o_C)||(1_A\times\Delta_C)||(\mathbf{Q}\times 1_C)||(1_B\times\Delta^o_C)||(1_B\times p_C).$$

This formula is also easier to understand looking ahead to the graphical representation in the next section.

3.4. 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 [18]. We extend the representation given in that paper to the combination of sequential and parallel operations and constants. We will see in Section 4.1 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 introduced in [5], see also [26]).

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

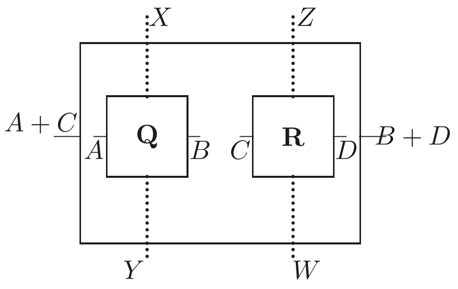


The multiple lines on the left and right hand sides correspond to parallel interfaces which are products of sets. For example, the component has left interface $A \times B \times C$. Instead the multiple lines on the top and bottom correspond to sequential interfaces which are disjoint sums of sets,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

3.5. Operations and constants

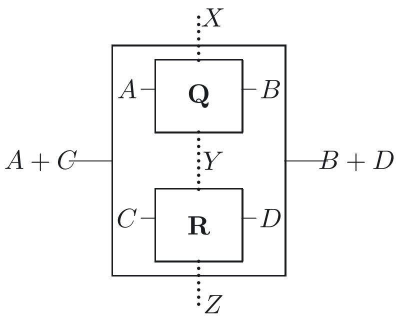
3.5.1. *Sum*

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



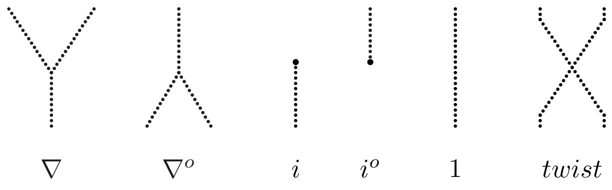
3.5.2. *Sequential composition*

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



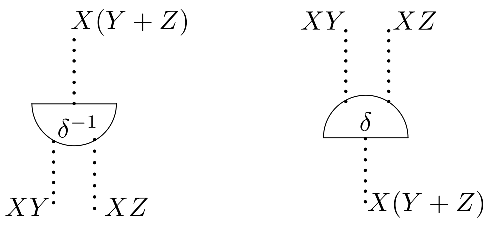
3.5.3. *Sequential connectors*

The various sequential connectors are represented as:



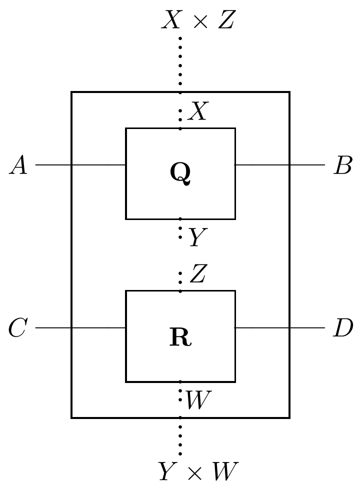
3.5.4. *Distributive law*

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



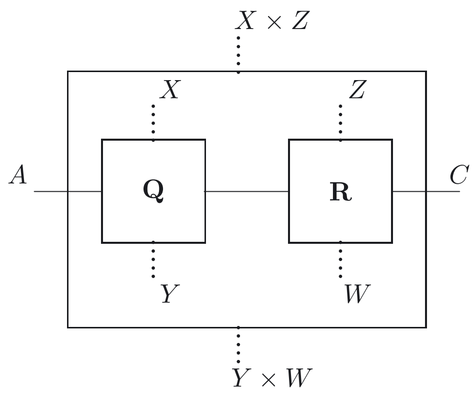
3.5.5. *Product*

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



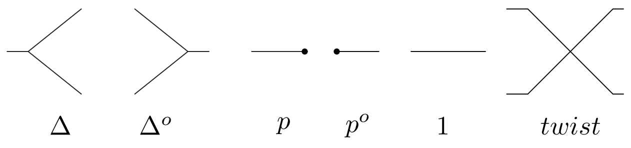
3.5.6. *Parallel with communication*

The parallel with communication $\mathbf{Q}^X_{Y;A,B}||\mathbf{R}^Z_{W;B,C}$ is represented as:



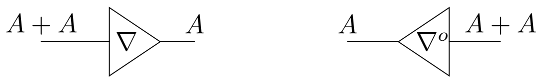
3.5.7. *Parallel connectors*

The various parallel connectors are represented as:



3.5.8. *Parallel codiagonal*

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



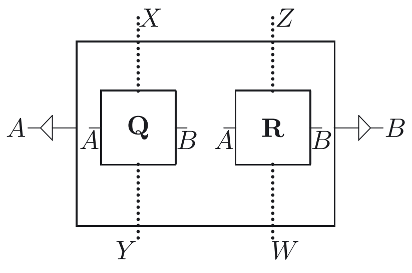
3.6. Some derived operations and constants

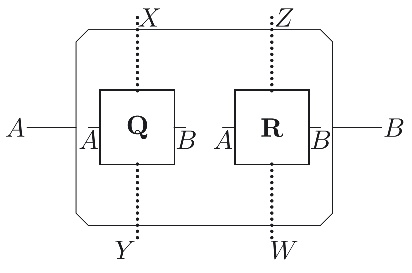
3.6.1. *Repeated sequential and repeated parallel operations*

When representing repeated sequential operations, frequently we omit all but the last bounding rectangle. 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 and parallel operations without serious ambiguity.

3.6.2. *Local sum*

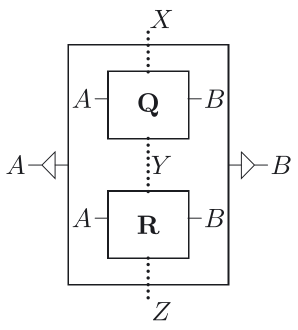
The local sum $\mathbf{Q}^{X}_{Y;A,B} +_{loc} \mathbf{R}^{Z}_{W;A,B}$ is represented by the first diagram below which includes the parallel codiagonals, but also, since it is such a common derived operation, more briefly by the second diagram below:

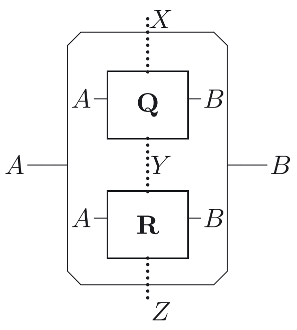




3.6.3. *Local sequential*

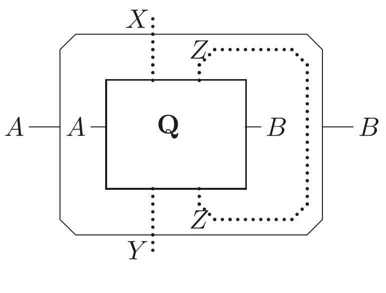
The local sequential $\mathbf{Q}^{X}_{Y;A,B} +_{Y,loc} \mathbf{R}^{Y}_{Z;A,B}$ is represented by the first diagram below which includes the parallel codiagonals, but also, since it is such a common derived operation, more briefly by the second diagram below:





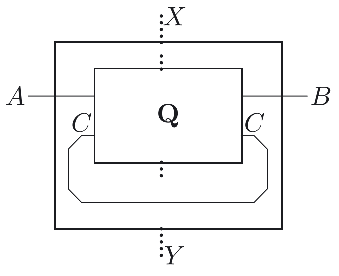
3.6.4. *Sequential feedback*

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



3.6.5. *Parallel feedback*

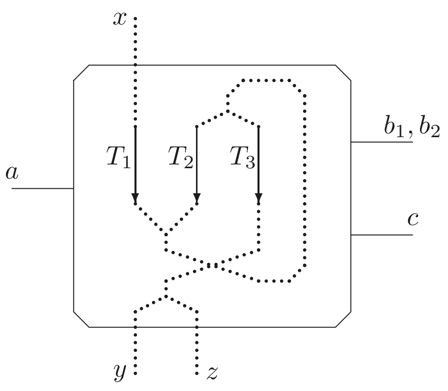
The parallel feedback $\mathsf{Pfb}_C(\mathbf{Q}^X_{Y;A\times C,B\times C})$ is represented by the diagram:



4. Examples

4.1. Any automaton is in $\Sigma(E)$

If E is the set of automata with two states with a single transition, then any automaton may be given as a sequential expression of elements in E (see [26]). We illustrate by considering the first example of Section 2.1. Let T_1, T_2, T_3 be the three single transition automata 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, and 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 $T_1 +_{\mathbf{loc}} T_2 +_{\mathbf{loc}} T_3$ composed sequentially with sequential wires:



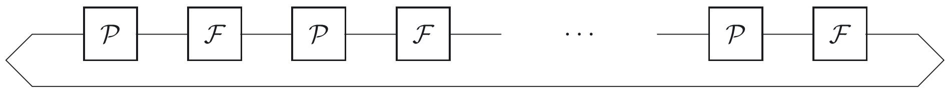
4.2. The dining philosophers system

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

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

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

The system is represented 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 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 $\frac{4415}{6912}$.

4.2.1. *The probability of deadlock*

We describe here briefly the result of [9] in which we apply Perron-Frobenius theory to the problem of deadlock in the Dining Philosopher problem.

Definition 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).

Proposition 4.2 (Perron-Frobenius)**.** *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.

A sketch of the proof of this proposition was given in Section 4 of [9], as well as a proof of the following corollary.

Corollary 4.3. *In the dining philosopher problem $\mathbf{DP}_n$ with q_0 being the state $(1,1,\ldots,1)$ the unique reachable deadlock is $(3,2,3,2,\ldots,3,2)$. The initial state is reachable from all other reachable states. Hence the probability of reaching a deadlock from the initial state in k steps tends to 1 as k tends to infinity.*

Remark. The corollary does not depend on the specific positive probabilities of the actions of the philosophers and forks. Hence the result is true with any positive probabilities. In fact, different philosophers and forks may have different probabilities without affecting the conclusion of the corollary.

Remark. The reader should note that there has been considerable study of (non-compositional) algorithms for determining properties like deadlock in other automata models (see [1]) originating in [24,32].

5. Sofia's birthday party

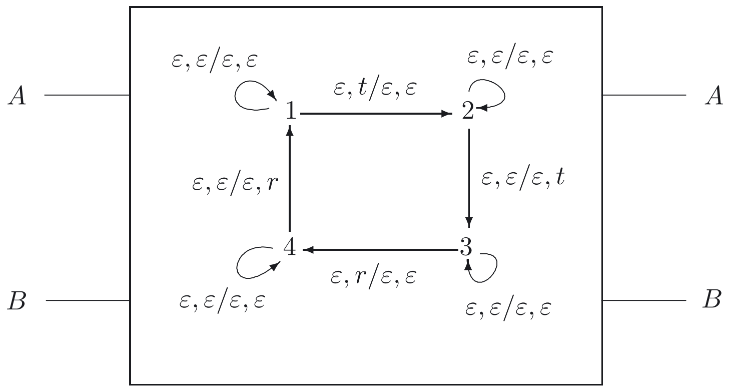
The example we would like to describe is a variant of the Dining Philosopher Problem which we call Sofia's Birthday Party. Instead of a circle of philosophers around a table with as many forks, we consider a circle of seats around a table separated by forks on the table. Then there are a number of children (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 transition element $\mathcal{L}$ has two vertices p and q and one labelled edge $x, \varepsilon/\varepsilon, \varepsilon : q \to p$. 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 edge $\varepsilon, \varepsilon/x, \varepsilon : q \to p$. 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:


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

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.

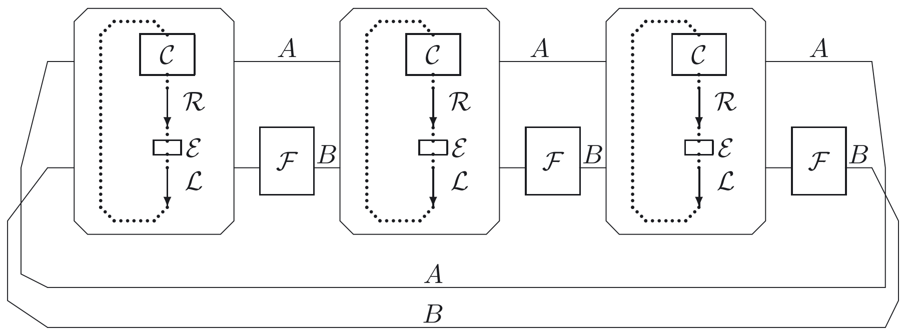
The fork $\mathcal{F}$ is as in the dining philosopher system (but with all transitions weighted 1).

Let $\mathcal{S} = \mathsf{Sfb}_P(C\,;R\,;E\,;L)$. 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

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

This automaton has the behaviour as informally described above. Its diagrammatic representation (in the case of three seats) 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

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

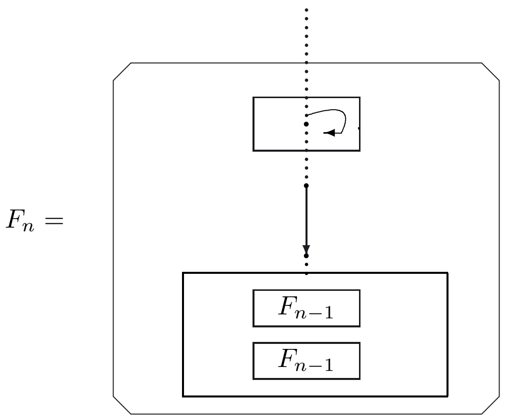
Then in 12 of these states there is a child eating: only one may eat at a time. It is straightforward to calculate the 36×36 Markov matrix and iterating show that the probability of a child eating after 1 step from initial state $(5, 1, 1, 1, 1, 1)$ is 0, after 2 steps is $\frac{19}{60}$, after 3 steps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

6. 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.

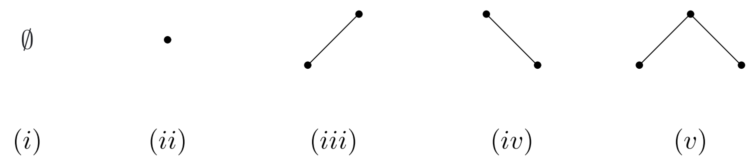
We would like to model such a system in our algebra, and to calculate the probability of arriving in the situation of having 2^n idling processes after k steps. The simplest version, in which there is no communication between the processes, and hence all parallel interfaces may be taken to be trivial,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

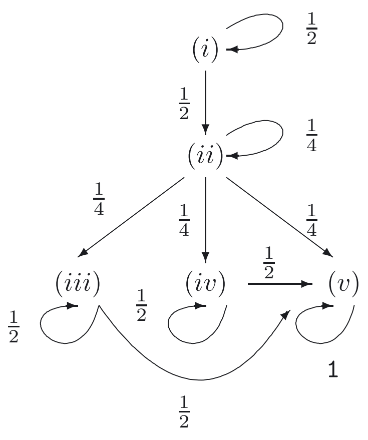
$$f_n = 1 + f_{n-1}^2, \quad f_0 = 1,$$

and hence $f_0 = 1$, $f_1 = 2$, $f_2 = 5$, $f_3 = 27$, ... 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 [2,21]). 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.

7. The probabilistic automata of Segala and Lynch

We indicate how the finite probabilistic automata of Segala and Lynch [22,27] (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 scheduling 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.

7.1. Segala-Lynch automata

A finite Segala-Lynch automaton [22] 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.

7.2. 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, and (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.

7.3. The parallel composition of Segala-Lynch automata

We describe how CSP synchronization of weighted automata may be modelled in our algebra (see also [8]). 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 Segal-Lynch automata.

References

[1] C. Baier, M. Grösser and F. Ciesinski, *Model checking linear-time properties of probabilistic systems.* Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science (2009), 519–596.

[2] A. Blass, Seven trees in one. *J. Pure Appl. Algebra* **103** (1991) 1–21.

[3] S.L. Bloom and Z. Ésik, *Iteration Theories.* EATCS Monographs on Theoretical Computer Science (1993).

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

[5] A. Carboni and R.F.C. Walters, Cartesian bicategories I. *J. Pure Appl. Algebra* **49** (1987) 11–32.

[6] B. Coecke and D. Pavlovic, Quantum measurements without sums, in *Mathematics of Quantum Computing and Technology.* G. Chen, L. Kauffman and S. Lamonaco, Eds. Chapman & Hall (2007), 567–604.

[7] B. Coecke and S. Perdrix, *Environment and classical channels in categorical quantum mechanics.* `arXiv:1004.1598` (2010).

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

[9] L. de Francesco Albasini, N. Sabadini and R.F.C. Walters, The compositional construction of Markov processes. `arXiv:0901.2434`.

[10] 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`.

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

[12] S. Eilenberg, *Automata, Languages, and Machines A.* Academic Press, New York (1974).

[13] D. Harel, Statecharts: a visual formalism for complex systems. *Sci. Comput. Program.* **8** (1987) 231–274.

[14] M. Hasegawa, On traced monoidal closed categories. *Math. Struct. Comput. Sci.* **19** (2009) 217–244.

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

[16] C.A.R. Hoare, *Communicating Sequential Processes.* Prentice Hall (1985).

[17] A. Joyal, R. Street and D. Verity, Traced monoidal categories. *Math. Proc. Cambridge Philos. Soc.* **119** (1996) 447–468.

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

[19] P. Katis, N. Sabadini and R.F.C. Walters, A formalisation of the IWIM Model. A. Porto and G.-C. Roman, Eds., in *Proc. Coordination 2000. Lecture Notes in Computer Science* **1906** (2000) 267–283.

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

[21] F.W. Lawvere, Some remarks on the future of category theory, *Proceedings Category Theory 1990. Lecture Notes in Mathematics* **1488** (1991) 1–13.

[22] N.A. Lynch, R. Segala and F.W. Vaandrager, Observing branching structure through probabilistic contexts. *SIAM J. Comput.* **37** (2007) 977–1013.

[23] R. Milner, *A Calculus of Communicating Systems*. Springer Verlag (1980).

[24] A. Pnueli and L. Zuck, Probabilistic verification by tableaux, in *Proceedings LICS'86*. IEEE Computer Society Press (1986), 322–331.

[25] M.O. Rabin, Probabilistic Automata. *Inform. Control* **6** (1963) 230–245.

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

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

[28] P. Sobociński, A non-interleaving process calculus for multi-party synchronisation, *Proceedings ICE '09* (2009).

[29] A. Sokolova and E.P. de Vink, *Probabilistic automata: system types, parallel composition and comparison. Lecture Notes in Computer Science* **2925** (2004) 1–43.

[30] Gh. Stefanescu, *Network Algebra*. Springer Verlag (2000).

[31] R.J. van Glabbeek, S.A. Smolka and B. Steffen, *Reactive, generative and stratified models of probabilistic processes*. Inform. Comput. (1995).

[32] M. Vardi, Automatic verification of probabilistic concurrent finite-state programs, in *Proceedings FOCS'85*. IEEE Computer Society Press (1985), 327–338.

Communicated by A. Cherubini.
Received February 16, 2010. Accepted November 23, 2010.