Component models in typed linear algebra

Luis S. Barbosa
(joint work with J.N. Oliveira)

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY
INESCTEC

Universidade do Minho

IFIP WG 1.3 Meeting

Rome, March 16, 2013

Starting point

A calculus of state-based components building on a generic approach to transition systems, described by coalgebras

$$Q \to \mathbf{F}Q$$

where Q is a set of states and $\mathbf{F}Q$ captures the future behaviour of the system, according to evolution "pattern" $\mathbf{F}$

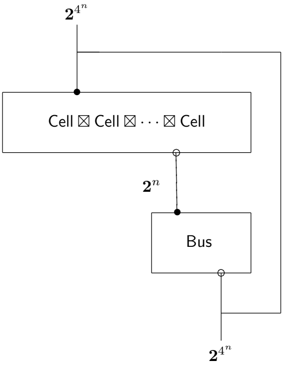
Examples:

- Mealy machines — $\mathbf{F}Q = \mathbf{B}(Q \times O)^I$
- Moore machines — $\mathbf{F}Q = (\mathbf{B}Q)^I \times O$

for I, O input / output types, and $\mathbf{B}$ a **behaviour** (strong) monad — e.g. **maybe** $(- + \mathbf{1})$, **powerset** $(\mathcal{P})$, **distribution** $(\mathcal{D})$, etc.

Starting point

The component calculus



$$\mathsf{GameLife} \;=\; ((\mathsf{Cell} \boxtimes \mathsf{Cell} \boxtimes \cdots \boxtimes \mathsf{Cell})\,;\,\mathsf{Bus})\,\Lsh$$

$$\begin{array}{rl}
\textit{lax} & (p \boxtimes p')\,;(q \boxtimes q') \;\sim\; (p\,;q) \boxtimes (p'\,;q') \\
 & \mathsf{copy}_{K \boxtimes K'} \;\sim\; \mathsf{copy}_K \boxtimes \mathsf{copy}_{K'} \\
\textit{functions} & \ulcorner f \urcorner \boxtimes \ulcorner g \urcorner \;\sim\; \ulcorner f \times g \urcorner \\
\textit{assoc} & (p \boxtimes q) \boxtimes r \;\sim\; (p \boxtimes (q \boxtimes r))[\mathsf{a}, \mathsf{a}^\circ] \\
\textit{id} & \mathsf{idle} \boxtimes p \;\sim\; p[\mathsf{r}, \mathsf{r}^\circ] \\
\textit{zero} & \mathsf{nil} \boxtimes p \;\sim\; \mathsf{nil}[\mathsf{zl}, \mathsf{zl}^\circ] \\
\textit{comm} & p \boxtimes q \;\sim\; (q \boxtimes p)[\mathsf{s}, \mathsf{s}] \quad \text{if } \mathsf{B} \text{ is } \textit{commutative}
\end{array}$$

Motivation: going quantitative

From: **may it happen?**
... to: **how often / how costly / how ... will it happen?**

- In particular, can propagation of **faults** be predicted (**calculated**) rather than simulated?

cf, calculating fault **propagation** *in* **functional programs** *([Oliveira'12] in the context of the QAIS project, 2012-15)*

Background

Vast literature, e.g.,

- **Probabilistic program semantics** — [Kozen 79]
- **Weighted automata** — [Buchholz 08, Droste & Gastin 09]
- **Probabilistic automata** — [Larsen & Skou 91]
- **Coalgebraic approaches** — [Sokolova 05]

  In particular, a recent paper

  [Bonchi et al 12] — A coalgebraic perspective on linear weighted automata — Information and Computation, 211:77–105.

  combines coalgebraic reasoning with linear algebra.

But there is a **price to pay**: functors need to handle quantities explicitly while states become vectors and coalgebras become linear maps

Our aim

- to obtain the same quantitative effect in component modelling while retaining the simplicity of the original (qualitative) coalgebra approach

keep weighting and quantification implicit rather than explicit

i.e., change to a typed linear algebra and hide weight calculations by matrix operations

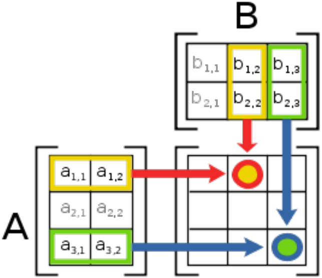
Typed is the keyword ...

- **Functions** — functional programming, an advanced type discipline: typing $f : A \to B$ well accepted.

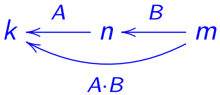
- **Relations** — ubiquitous (eg. graphs) but still under the atavistic *set of pairs* interpretation. Thus $R \subseteq A \times B$ widespread, compared to $A \xrightarrow{R} B$.

- **Matrices** — key concept in mathematics as a whole, many tools (eg. MATLAB, MATHEMATICA) but still "untyped" — explicit **dimension** checking required.

Matrices as arrows



Given a **semiring** $(\mathbb{S}; +, \times, 0, 1)$ matrix **composition** $A \cdot B$ obeys to the **typing rule**



such that

$$r(A \cdot B)c \quad = \quad \langle \sum x \;::\; (rAx) \times (xBc) \rangle \qquad (1)$$

where $\sum$ is the finite iteration over n of the $+$ operation of $\mathbb{S}$.

Typed linear algebra

- **objects** are matrix dimensions and whose
- **morphisms** ($m \xleftarrow{M} n$, $n \xleftarrow{N} k$, etc) are the matrices themselves.

Strictly speaking, there is one such category per matrix cell-level algebra.

Notation:

- write rAc for the (r, c)-th cell of matrix A
- $Mat_{\mathbb{S}}$ denotes the category, parametric on **semiring** $\mathbb{S}$

Typed linear algebra

Type checking:

> *For matrices A and B* **of the same type** $n \longleftarrow m$ *, we can extend cell level algebra to matrix level, eg. by* **adding** *and* **multiplying** *matrices (Hadamard product),*
>
> $$A + B \quad , \quad A \times B$$

The underlying type system is **polymorphic** and type inference proceeds by **unification**, as in programming languages.

For instance, the **identity matrix**

$$n \xleftarrow{id_n} n \quad = \quad \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{n \times n}$$

is polymorphic on type n.

Converse

Given matrix $n \xleftarrow{M} m$, notation $m \xleftarrow{M^\circ} n$ denotes its **converse**.
(M° is M changed by transposition)

$$id_n \cdot M = M = M \cdot id_m \qquad (2)$$

$$(M^\circ)^\circ = M \qquad (3)$$

$$(M \cdot N)^\circ = N^\circ \cdot M^\circ \qquad (4)$$

Typed linear algebra

Abelian structure

$$M + 0 = M = 0 + M \tag{5}$$
$$M \cdot 0 = 0 = 0 \cdot M \tag{6}$$

Bilinearity — composition is bilinear relative to $+$:

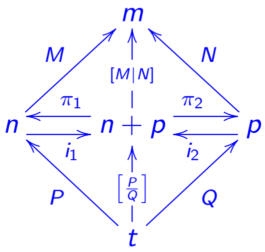
$$M \cdot (N + P) = M \cdot N + M \cdot C \tag{7}$$
$$(N + P) \cdot M = N \cdot M + P \cdot M \tag{8}$$

Biproducts — products and coproducts together enabling **block** algebra — the whole story goes back to MacLane & Birkhoff; see also recent thesis [Macedo 12] for applications

(Polymorphic) block combinators

Two ways of putting matrices together to build larger ones:

- $X = [M|N]$ — M and N side by side ("junc")
- $X = \left[\frac{P}{Q}\right]$ — P on top of Q ("split").



cf $\pi_1 = [id_m|0]$, $\imath_1 = \left[\frac{id_m}{0}\right]$ and $P + Q = [i_1 \cdot P|i_2 \cdot Q]$

Blocked linear algebra

Rich set of laws, for instance **divide-and-conquer**,

$$[A|B] \cdot \left[\frac{C}{D}\right] \quad = \quad A \cdot C + B \cdot D \tag{9}$$

two **“fusion”-laws**,

$$C \cdot [A|B] \quad = \quad [C \cdot A|C \cdot B] \tag{10}$$

$$\left[\frac{A}{B}\right] \cdot C \quad = \quad \left[\frac{A \cdot C}{B \cdot C}\right] \tag{11}$$

structural equality,

$$\left[\frac{A}{B}\right] = \left[\frac{C}{D}\right] \quad \Leftrightarrow \quad A = C \wedge B = D \tag{12}$$

— all offered for free from **biproducts**.

Vectors

Vectors are special cases of matrices in which one of the types is 1, for instance

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \qquad \text{and} \qquad w = \begin{bmatrix} w_1 & \ldots & w_n \end{bmatrix}$$

Column vector v is of type $m \longleftarrow 1$ (m rows, one column) and **row** vector w is of type $1 \longleftarrow n$ (one row, n columns).

Special matrices

- The **bottom** matrix $n \xleftarrow{0} m$ — wholly filled with 0s
- The **top** matrix $n \xleftarrow{1} m$ — wholly filled with 1s
- The **identity** matrix $n \xleftarrow{id} n$ — diagonal of 1s
- The **bang** (row) vector $1 \xleftarrow{!} m$ — wholly filled with 1s

Thus, (typewise) **bang** matrices are special cases of **top** matrices:

$$1 \xleftarrow{1} m \quad = \quad !$$

Also note that, on type $1 \longleftarrow 1$:

$$1 \ = \ ! \ = \ id$$

Type generalization

As is standard is relational mathematics, matrix types can be generalized from numeric dimensions ($n, m \in \mathbb{N}_0$) to arbitrary denumerable types (X, Y), taking **disjoint union** $X + Y$ for $m + n$, Cartesian product $X \times Y$ for mn, etc.

In this setting, a **function** $B \xleftarrow{f} A$ will be represented in $Mat_{\mathbb{S}}$ by a (Boolean) matrix $B \xleftarrow{[\![f]\!]} A$ such that

$$b[\![f]\!]a \quad \triangleq \quad (b =_{\mathbb{S}} f\ a)$$

Thus

$$! \cdot [\![f]\!] \;=\; !$$

Weighted Mealy machines as $Mat_{\mathbb{S}}$ arrows

A **weighted Mealy machine** $M = (I, O, Q, \alpha, \gamma)$ consists of

- input and output **alphabets** I, O, respectively
- finite set of **states** Q
- $\gamma : Q \to \mathbb{S}$ — **weighted** vector of **seed** (initial) states
- $\alpha : Q \to (\mathbb{S}^{Q \times O})^I$ such that $\alpha(p)(i)(q, o)$ is the cost of a **transition** from p to q triggered by input i and producing output o: $p \xrightarrow{i/o} q$ (0 if no such transition).

If weights are trivial, the definition boils down to

$$(Q, \alpha : Q \to (Q \times O)^I, \gamma : \mathcal{P}Q)$$

i.e., a (seeded) coalgebra for functor $\mathbf{F}X = (X \times O)^I$ in Set.

Probabilistic Mealy machines as $Mat_{\mathbb{S}}$ arrows

For a **probabilistic** Mealy machine make:

- $\mathbb{S}$ the interval $[0,1]$ in $\mathbb{R}$
- α is such that $! \cdot \alpha \leq !$. I.e., $! \cdot \alpha$ is a $(0,1)$-vector (because $! \cdot M$ adds all columns of M).

- Wherever $! \cdot \alpha = !$ the machine is **total** and α is a **column stochastic** matrix, or **probabilistic function**
- For $I = 1$, the definition boils down to a **probabilistic automata** A **weighted finite automaton** $W = (I, Q, \alpha, \gamma)$ where
  - $\gamma : Q \rightarrow \mathbb{S}$ — weight functions for leaving a state
  - $\alpha : I \rightarrow \mathbb{S}^{Q \times Q}$ such that $\mu(a)(p, q)$ is the cost of **transition** $p \xrightarrow{a} q$ (0 if no such transition).

Weighted Mealy machines as $Mat_{\mathbb{S}}$ arrows

- $\gamma : Q \to \mathbb{S}$ is encoded as $Mat_{\mathbb{S}}$ vector $Q \longrightarrow 1$

$$1 \; \gamma \; q \quad \triangleq \quad \gamma(q) \tag{13}$$

- The matrix encoding of $\alpha : Q \to (\mathbb{S}^{Q \times O})^I$ can be regarded as either of type $Q \times I \longrightarrow Q \times O$ or $Q \longrightarrow I \times Q \times O$, as these types are isomorphic in $Mat_{\mathbb{S}}$.

Putting α and γ together into a $Mat_{\mathbb{S}}$ **coalgebra**

$$Q \xrightarrow{M = \left[\frac{\alpha}{\gamma}\right]} (I \times Q \times O) + 1$$

for functor

$$\mathbf{F}X = (id \otimes X \otimes id) \oplus id$$

Weighted Mealy machines as $Mat_{\mathbb{S}}$ arrows

$$\mathbf{F}X = (id \otimes X \otimes id) \oplus id$$

where $\otimes$ is **Kronecker** product and $\oplus$ is **direct sum**

absorption

$$(C \oplus D) \cdot \left[\frac{A}{B}\right] = \left[\frac{C \cdot A}{D \cdot B}\right] \tag{14}$$

fusion

$$\left[\frac{M}{N}\right] \otimes C = \left[\frac{M \otimes C}{N \otimes C}\right] \tag{15}$$

pointwise Kronecker

$$(y, x)(M \otimes N)(b, a) = (yMb) \times (xNa) \tag{16}$$

Weighted Mealy homomorphisms in $Mat_{\mathbb{S}}$

Let us now see how the **typed LA** encoding of **WA** regains the **simplicity** of the original, **qualitative** starting point.

A **homomorphism** *between weighted Mealy machines M and M' is a function h making the following $Mat_{\mathbb{S}}$-diagram commutes,*

$$\begin{array}{ccc} I \times Q \times O + 1 & \xleftarrow{\;\;M\;\;} & Q \\ \Big\downarrow{\scriptstyle (id \otimes h \otimes id) \oplus id} & & \Big\downarrow{\scriptstyle h} \\ I \times Q' \times O + 1 & \xleftarrow[\;\;M'\;\;]{} & Q' \end{array} \tag{17}$$

Weighted Mealy homomorphisms in $Mat_{\mathbb{S}}$

In cross-checking that this indeed is the usual, quantified definition, we will resort to two **rules of thumb**,

$$y(f \cdot N)x = \langle \sum z : y = f(z) : zNx \rangle \quad (18)$$

$$y(g^{\circ} \cdot N \cdot f)x = (g(y))N(f(x)) \quad (19)$$

where N is an arbitrary matrix and f, g are functional matrices.

These rules generalize similar equalities in **relation algebra**.

Weighted Mealy homomorphisms in $Mat_{\mathbb{S}}$

Let us calculate:

$$
\begin{array}{ll}
& (\mathbf{F}h) \cdot M \;=\; M' \cdot h \\
\Leftrightarrow & \quad \{ \text{ unfold } \mathbf{F}h \text{ , } M \text{ and } M' \text{ } \} \\
& ((id \otimes h \otimes id) \oplus id) \cdot \left[\frac{\alpha}{\gamma}\right] \;=\; \left[\frac{\alpha'}{\gamma'}\right] \cdot h \\
\Leftrightarrow & \quad \{ \text{ absorption (14), identity (2) and fusion (11) } \} \\
& \left[\frac{(id \otimes h \otimes id) \cdot \alpha}{\gamma}\right] \;=\; \left[\frac{\alpha' \cdot h}{\gamma' \cdot h}\right] \\
\Leftrightarrow & \quad \{ \text{ equality (12) } \} \\
& \left\{ \begin{array}{l} (id \otimes h \otimes id) \cdot \alpha = \alpha' \cdot h \\ \gamma = \gamma' \cdot h \end{array} \right.
\end{array}
\tag{20}
$$

Weighted Mealy homomorphisms in $Mat_{\mathbb{S}}$

Next we unfold $(id \otimes h \otimes id) \cdot \alpha = \alpha' \cdot h$ by extensional equality

$$
\begin{array}{ll}
& (i, q', o)((id \otimes h \otimes id) \cdot \alpha)q = (i, q', o)(\alpha' \cdot h)q \\
\Leftrightarrow & \quad \{ \text{ (19) on the rhs, since } h \text{ is a function } \} \\
& (i, q, o)((id \otimes h \otimes id) \cdot \alpha)q = (i, q', o)\alpha'(h(q)) \\
\Leftrightarrow & \quad \{ \text{ (18) on the lhs, since } id \otimes h \otimes id \text{ is a function too } \} \\
& \langle \sum (i', p, o') \ : \ (i, q', o) = (id \otimes h \otimes id)(i', p, o') : \ (i', p, o')\alpha q \rangle \\
& = \ (i, q', o)\mu'(h(q)) \\
\Leftrightarrow & \quad \{ \text{ simplifying } \} \\
& \langle \sum \ p \ : \ q' = h(p) : \ (i, p, o)\alpha q \rangle = (i, q', o)\alpha'(h(q))
\end{array}
$$

Weighted Mealy homomorphisms in $Mat_{\mathbb{S}}$

Finally, writing $p \xleftarrow{i/o} q$ for the weight of the corresponding transition:

$$\langle \sum p \;:\; q' = h(p) : \; p \xleftarrow{i/o} q \,\rangle \;=\; q' \xleftarrow{i/o} h(q)$$

In words:

> *the weight associated to transition* $q' \xleftarrow{i/o} h(q)$ *in the target automaton accumulates the weights of all transitions* $p \xleftarrow{i/o} q$ *in the source automaton for all* p *which* h *maps to* q'*.*

Unfolding $\gamma = \gamma' \cdot h$ will yield the expected $\gamma(q) = \gamma'(h(q))$.

Weighted behaviour

- In Set the final coalgebra for $\mathbf{F}X = (X \times O)^I$ is

$$out : O^{I^+} \rightarrow (O^{I^+} \times O)^I$$
$$out(f)(i) = (\lambda s.\, f(i : s), f[i])$$

- Functions $f : I^+ \rightarrow O$ are the behaviours generated by Mealy machines. A weighted behaviour associates a weight in $\mathbb{S}$ to each of them.
- Seed conditions have to be put into the picture as well.

Weighted behaviour

The function $B_W : Q \to \mathbb{S}^{O^{I^+}}$ which associates to each state in Q of M its weighted behaviour is encoded into a $Mat_{\mathbb{S}}$ matrix of type $Q \longrightarrow O^{I^+}$, ie. the **F**-homomorphism

$$\begin{array}{ccc} I \times Q \times O + 1 & \xleftarrow{\quad M \quad} & Q \\ \Big\downarrow {\scriptstyle (id \otimes B_W \otimes id) \oplus id} & & \Big\downarrow {\scriptstyle B_W} \\ I \times O^{I^+} \times O + 1 & \xleftarrow[\quad M_\nu \quad]{} & O^{I^+} \end{array}$$

where

$$M_\nu = \left[\frac{\alpha_\nu}{!} \right]$$

$$(i, \lambda\, s.\, f(i : s), f[i])\ \alpha_\nu\ q$$

Weighted behaviour

What does homomorphism B_W mean?

$$M_\nu \cdot B_W = ((id \otimes B_W \otimes id) \oplus id) \cdot M$$

$$\left[\frac{\alpha_\nu}{!}\right] \cdot B_W = ((id \otimes B_W \otimes id) \oplus id) \cdot \left[\frac{\alpha}{\gamma}\right]$$

$\Leftrightarrow$ { fusion (11) and absorption (14) }

$$\left[\frac{\alpha_\nu \cdot B_W}{! \cdot B_W}\right] = \left[\frac{(id \otimes B_W \otimes id) \cdot \alpha}{\gamma}\right]$$

$\Leftrightarrow$ { equality (12) }

$$\begin{cases} \alpha_\nu \cdot B_W = (id \otimes B_W \otimes id) \cdot \alpha \\ ! \cdot B_W = \gamma \end{cases}$$

Weighted behaviour

$$\boxed{! \cdot B_W = \gamma}$$

$$
\begin{array}{ll}
 & 1\,(! \cdot B_W)\,q \;=\; 1\,\gamma\,q \\
\Leftrightarrow & \quad \{\ \text{composition; } ! \text{ and } \gamma \text{ are functions}\ \} \\
 & \langle \sum\ z\ :\ 1 = !(z)\ :\ z\,B_W\,q \rangle \;=\; \gamma(q) \\
\Leftrightarrow & \quad \{\ \text{simplifying}\ \} \\
 & \langle \sum\ z\ ::\ z\,B_W\,q \rangle \;=\; \gamma(q)
\end{array}
$$

i.e., the weight of an initial state q is the sum of all weights all behaviours generated from q.

Weighted behaviour

$$\boxed{\alpha_\nu \cdot B_W = (id \otimes B_W \otimes id) \cdot \alpha}$$

Let's start by unfolding $(id \otimes B_W \otimes id) \cdot \alpha$:

$$
\begin{array}{ll}
& (i, f, o)\,((id \otimes B_W \otimes id) \cdot \alpha)\, q \\
= & \{ \text{ matrix composition } \} \\
& \langle \sum\ i', q', o'\ ::\ (i, f, o)(id \otimes B_W \otimes id)(i', q', o') \rangle \times (i', q', o')\alpha\, q \\
= & \{ \text{ abbreviate } (i, q', o)\alpha\, q \text{ to } q' \xleftarrow{i/o} q \ \} \\
& \langle \sum\ q'\ ::\ f\, B_W\, q' \times q' \xleftarrow{i/o} q \rangle
\end{array}
$$

Weighted behaviour

$$(i, f, o)\,(\alpha_\nu \cdot B_W)\,q \;=\; \langle \sum\; q' \;::\; f\,B_W\,q' \;\times\; q' \xleftarrow{i/o} q\;\rangle$$

$\Leftrightarrow$ { matrix composition; α_ν is Boolean }

$$\langle \sum\; g \;:\; (i, f, o)\,\alpha_\nu\,g \;:\; g\,B_W\,q \rangle$$

$$=\; \langle \sum\; q' \;::\; f\,B_W\,q' \;\times\; q' \xleftarrow{i/o} q\;\rangle$$

$\Leftrightarrow$ { one-point rule }

$$(i, o, f)\,\alpha_\nu\,g \;\times\; g\,B_W\,q \;=\; \langle \sum\; q' \;::\; f\,B_W\,q' \;\times\; q' \xleftarrow{i/o} q\;\rangle$$

$\Leftrightarrow$ { $f = \lambda\,s.\,g(i : s)$, $o = g[i]$ because $(i, o, f)\,\alpha_\nu\,g$ }

$$g\,B_W\,q \;=\; \langle \sum\; q' \;::\; (\lambda\,s.\,g(i : s))\,B_W\,q' \;\times\; q' \xleftarrow{i/g[i]} q\;\rangle$$

Weighted behaviour

Summing up

$$\begin{cases} \alpha_\nu \cdot B_W = (id \otimes B_W \otimes id) \cdot \alpha \\ ! \cdot B_W = \gamma \end{cases}$$

$\Leftrightarrow$ { just computed, going index-wise }

$$\begin{cases} g\, B_W\, q \;=\; \langle \sum\; q' \;::\; (\lambda\, s.\, g(i:s))\, B_W\, q' \;\times\; q' \xleftarrow{i/g[i]} q \;\rangle \\ \langle \sum\; z \;::\; z\, B_W\, q \rangle \;=\; \gamma(q) \end{cases}$$

In words:

- (seed rule) Each initial state q generates a number of possible behaviours; the sum of their weights equals the weight of q.
- (generation rule) A behaviour $(\lambda\, s.\, g(i:s))$ is generated from all states reachable from a state generating g by accepting input i and outputting $g[i]$, accumulating the weights.

Weighted bisimulations in $Mat_{\mathbb{S}}$

Strategy

- Start from an equivalence relation K over Q and define the quotient Q/K
- Check whether, whenever states $p, p' \in Q$ evolve under the same label to the same equivalence class $[q] \in Q/K$, are related by pKp', to conclude they are observational equivalent and K is a bisimulation.

... to be framed in $Mat_{\mathbb{S}}$

Weighted bisimulations in $Mat_{\mathbb{S}}$

General construction [Oliveira,12]: Equivalence relation K is a **bisimulation** for a $\mathbf{F}$-machine M iff any surjection h, such that $K = h^{\circ} \cdot h$, is a homomorphism $M/K \xleftarrow{h} M$:

$$\mathbf{F}h \cdot M = (M/K) \cdot h$$

$$\Leftrightarrow \qquad \{ \text{ definition of } M/K \ \}$$

$$\mathbf{F}h \cdot M = \mathbf{F}h \cdot M \cdot h^{\bullet} \cdot h$$

$$\Leftrightarrow \qquad \{ \text{ making } K_{\bullet} = h^{\bullet} \cdot h \ \}$$

$$\mathbf{F}K \cdot M = \mathbf{F}K \cdot M \cdot K_{\bullet}$$

i.e., $\mathbf{F}K \cdot W$ is invariant wrt the "weighted equivalence" $K_{\bullet}$.

Weighted bisimulations in $Mat_{\mathbb{S}}$

For Mealy machines

$$\mathbf{F}K \cdot M = \mathbf{F}K \cdot M \cdot K_{\bullet}$$

boils down to the index-wise formulation

$$\boxed{\langle \forall\ p, p', q, i, o\ :\ p\,K\,p'\ :\ [q]_K \xleftarrow{i/o} p\ =\ [q]_K \xleftarrow{i/o} p'\ \rangle}$$

where

$$p_1\ K_{\bullet}\ p_2\ \ =\ \ (h(p_1))(h \cdot h^{\circ})^{-1}(h(p_2))$$

Diagonal $(h \cdot h^{\circ})^{-1}$ represents the *weight vector [which] is well known in stochastic modeling* [Buchholz 08].

Lessons from this exercise

Much still to be done! — but time already to wrap up with the main points:

- Shift from **qualitative** to **quantitative** methods may proceed in two ways:
  - Extend original definitions in the **same** category
    or
  - Stay with original definitions but **change** the category
- $Mat_{\mathbb{S}}$ appears to be a suitable choice for calculating with (simple) weighted (probabilistic) automata.

Back to the component calculus

Non deterministic components live in two *universes* related by an adjunction:

- one is **“for calculating”**
- the other **“for programming”** (with the underlying **monad**)

$$f = \Lambda R \quad \Leftrightarrow \quad \langle \forall\ b, a \ :: \ b\ R\ a \Leftrightarrow b \in f\ a \rangle$$

that is,

$$A \to \mathcal{P}B \ \underset{\Lambda}{\overset{(\in\cdot)}{\cong}} \ A \to_{Rel} B$$

Back to the component calculus

In **probabilistic** components outputs become distributions,

$$A \to \mathcal{D}B \quad \cong \quad A \to_{LS} B$$

$$M = [\![f]\!] \quad \Leftrightarrow \quad \langle \forall\ b, a\ ::\ M(b,a) = (f\ a) b \rangle$$

where $\mathcal{D}B$ is the B-distribution **monad**

$$\mathcal{D}B \quad = \quad \{ \mu \in [0,1]^B \mid \sum_{b \in B} \mu\ b = 1 \}$$

and LS denotes the **category** of **left-stochastic** matrices (columns in such matrices add up to 1).

Towards a linear algebra of components

The smooth interplay between functions, relations and matrices provides the ground for

- re-interpreting the component calculus in LS (composition as multiplication)
- introducing faults in both components and their glue: the calculation of their propagation along an architecture comes for free

Towards a linear algebra of components

... but much remains to be done

- coping with both measurable and **unmeasurable** non-determinism: characterize the adjoint categories required by the various forms in which both appear combined in the literature — see eg. the **taxonomy** given by [Sokolova 05]
- going ahead of finite support and discrete distributions

Annex

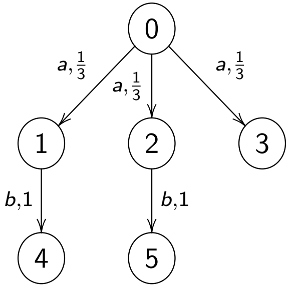
Annex:

computing weighted bisimulation

(details in [Oliveira 12])

Annex

Motivation (with a probabilistic automata)



		Q					
Q	A	0	1	2	3	4	5
0	a	0	0	0	0	0	0
0	b	0	0	0	0	0	0
1	a	0.3	0	0	0	0	0
1	b	0	0	0	0	0	0
2	a	0.3	0	0	0	0	0
2	b	0	0	0	0	0	0
3	a	0.3	0	0	0	0	0
3	b	0	0	0	0	0	0
4	a	0	0	0	0	0	0
4	b	0	1	0	0	0	0
5	a	0	0	0	0	0	0
5	b	0	0	1	0	0	0

Matrix α is type $Q \times A \longleftarrow Q$, for $Q = \{0, ..., 5\}$ and $A = \{a, b\}$.

Annex

Is equivalence relation

		Q					
		0	1	2	3	4	5
Q	0	1	0	0	0	0	0
	1	0	1	1	0	0	0
	2	0	1	1	0	0	0
	3	0	0	0	1	0	0
	4	0	0	0	0	1	1
	5	0	0	0	0	1	1

a bisimulation? It has four classes which can be represented by a quotient automaton using a suitable homomorphism h.

Annex

Candidate **surjective** homomorphism $Q' \xleftarrow{h} Q$:

		Q					
		0	1	2	3	4	5
Q'	0	1	0	0	0	0	0
	I	0	1	1	0	0	0
	II	0	0	0	1	0	0
	III	0	0	0	0	1	1

Its **kernel** $K = Q \xleftarrow{h^\circ \cdot h} Q$ is the given equivalence:

		Q					
		0	1	2	3	4	5
Q	0	1	0	0	0	0	0
	1	0	1	1	0	0	0
	2	0	1	1	0	0	0
	3	0	0	0	1	0	0
	4	0	0	0	0	1	1
	5	0	0	0	0	1	1

Annex

Building $M' = M/K$ (below we focus on α, α' only).

First attempt:

$$M' = M/K = (\mathbf{F}h) \cdot M \cdot h^{\circ}$$

that is

$$\alpha' = \alpha/K = (h \otimes id) \cdot \alpha \cdot h^{\circ}$$

		Q'			
Q'	A	0	I	II	III
0	a	0	0	0	0
0	b	0	0	0	0
I	a	2/3	0	0	0
I	b	0	0	0	0
II	a	1/3	0	0	0
II	b	0	0	0	0
III	a	0	0	0	0
III	b	0	2	0	0

Annex

It doesn't work because, in $Mat_{\mathbb{S}}$, h° is not a "true" converse of h: the **image** $h \cdot h^{\circ} \neq id$ is a **diagonal** counting "how much non-injective" h is, cf.

Q' \ Q'	0	I	II	III
0	1	0	0	0
I	0	2	0	0
II	0	0	1	0
III	0	0	0	2

However, **surjective** function h has inverses such as, eg. $h^{\bullet} = h^{\circ} \cdot (h \cdot h^{\circ})^{-1}$, obtained by straightforward **inversion** of diagonal $h \cdot h^{\circ}$:

Q \ Q'	0	I	II	III
0	1	0	0	0
1	0	1/2	0	0
2	0	1/2	0	0
3	0	0	1	0
4	0	0	0	1/2
5	0	0	0	1/2

Annex

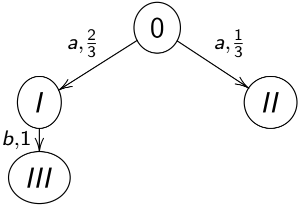
Second attempt:

$$M' = M/K = (\mathbf{F}h) \cdot M \cdot h^{\bullet}$$

that is (aside)

$$\alpha' = \alpha/K = (h \otimes id) \cdot \alpha \cdot h^{\bullet}$$

which leads to automaton



		Q'			
Q'	A	0	I	II	III
0	a	0	0	0	0
0	b	0	0	0	0
I	a	2/3	0	0	0
I	b	0	0	0	0
II	a	1/3	0	0	0
II	b	0	0	0	0
III	a	0	0	0	0
III	b	0	1	0	0

(Clearly, $h^{\bullet} \cdot h = K$ for injective h)

Annex

Definition. Equivalence relation K is a **bisimulation** for M iff any surjection h, such that $K = h^\circ \cdot h$, is a homomorphism $M/K \xleftarrow{h} M$:

$$\mathbf{F}h \cdot M = (M/K) \cdot h$$

$$\Leftrightarrow \quad \{ \text{ definition of } M/K \ \}$$

$$\mathbf{F}h \cdot M = \mathbf{F}h \cdot M \cdot h^\bullet \cdot h$$

$$\Leftrightarrow \quad \{ \text{ making } K_\bullet = h^\bullet \cdot h \ \}$$

$$\boxed{\mathbf{F}K \cdot M = \mathbf{F}K \cdot M \cdot K_\bullet}$$

Annex

Noting that $\mathbf{F}K$ is an equivalence relation (as K is so and $\mathbf{F}$ is a functor) and unfolding the invariant $\mathbf{F}K \cdot W$, for α:

$$\begin{aligned}
& (q,a)((K \otimes id) \cdot \mu)p \\
= & \quad \{ \ \text{composition rule (1)} \ \} \\
& \langle \sum q', a' \ :: \ (q,a)(K \otimes id)(q',a') \times ((q',a')\alpha(p) \rangle \\
= & \quad \{ \ \text{Kronecker (1) ; term } K \otimes id \text{ is Boolean} \ \} \\
& \langle \sum q', a' \ :: \ (qKq') \times (a = a') \times ((q',a')\alpha(p) \rangle \\
= & \quad \{ \ \text{let } [q]_K \text{ denote the equivalence class of } q \ \} \\
& \langle \sum q' \ : \ q' \in [q]_K : \ \ q' \xleftarrow{a} p \ \rangle
\end{aligned}$$

Annex

- In words:

$$\langle \sum \; q' \;:\; q' \in [q]_K :\; q' \xleftarrow{a} p \,\rangle$$

is the accumulated cost (probability) of transitions within the same equivalence class, which is invariant for equivalent initial states

Now turn attention to

$$(q, a)(\mathbf{F}K \cdot \alpha \cdot K_\bullet)p \;=\; \langle \sum \; p' \;::\; (q, a)(\mathbf{F}K \cdot \alpha)p' \times p' K_\bullet \; p \rangle$$

The weighted equivalence term is such that

$$p' K_\bullet \; p \;=\; \frac{1}{|p|_K} p' K \; p$$

where $|p|_K$ is the cardinal of equivalence class $[p]_K$.

Annex

Thus

$$(q,a)(\mathbf{F}K \cdot \alpha \cdot K_\bullet)p \quad = \quad \frac{1}{|p|_K}\langle \sum \; p' \; : \; p' \in [p]_K : \; (q,a)(\mathbf{F}K \cdot \alpha)p' \rangle$$

whose RHS unfolds into:

$$\frac{1}{|p|_K}\langle \sum \; p' \; : \; p' \in [p]_K : \; \langle \sum \; q'' \; : \; q'' \in [q]_K : \; q'' \xleftarrow{a} p' \; \rangle \rangle$$

In summary:

$$\langle \sum \; q' \; : \; q' \in [q]_K : \; q' \xleftarrow{a} p \; \rangle = \frac{1}{|p|_K}\langle \sum \; p', q'' \; : \; p' \in [p]_K \wedge q'' \in [q]_K : \; q'' \xleftarrow{a} p' \; \rangle$$

Annex

The following notation abbreviation will help: for R, S subsets of Q,

$$S \xleftarrow{a} R \quad = \quad \langle \sum \; p, q \; : \; p \in R \wedge q \in S : \;\; q \xleftarrow{a} p \,\rangle$$

Then equivalence K is a bisimulation once

$$[q]_K \xleftarrow{a} p \quad = \quad \frac{1}{|p|_K} \times (\, [q]_K \xleftarrow{a} [p]_K \,)$$

holds.