Towards a probabilistic Alloy

DALI/TRUST Kick-off Workshop

University of Minho, September 19-20, 2016

J.N. Oliveira & M.A. Cunha

INESC TEC & University of Minho
Prelude

Sir Arthur Eddington (1882-1944):

"I cannot believe that anything so ugly as multiplication of matrices is an essential part of the scheme of nature"


Serious warning to mathematicians and physicists — notation should be beautiful :-)

I agree — standard matrix notation is clumsy by modern computer science standards.
First of all, don’t refer to it as *multiplication* but rather as *composition*, as it generalizes *function* composition

\[(f \cdot g) x = f (g x)\]

and *relation* composition,

\[y (R \cdot S) x \iff \exists z :: y R z \land z S x\]

compare with

\[y (M \cdot N) x = \langle \sum z :: (y R z) \times (z S x) \rangle\]

where the infix \(y M z\) linking to relational notation is intentional.

In my view, any modelling tool should live peacefully with this *function → relation → matrix* evolution in *expressiveness*. 
The evolution

Determinism (*functions*):
- Functional programming (FP)
- Imperative programming (if restricted)

Non-determinism (*relations*):
- Logic programming
- Relational modelling

Probabilism (*matrices*):
- Quantum programming
- Probabilistic modelling
Alloy

Relational composition:

- The Swiss army knife of Alloy
- It subsumes **function application** and **“field selection”**
- Encourages a **navigational** (point-free) style based on pattern \( x.(R.S) \).
- Example:

  \[
  \begin{align*}
  Person &= \{ (P1),(P2),(P3),(P4) \} \\
  parent &= \{ (P1,P2),(P1,P3),(P2,P4) \} \\
  me &= \{ (P1) \} \\
  me.parent &= \{ (P2),(P3) \} \\
  me.parent.parent &= \{ (P4) \} \\
  Person.parent &= \{ (P2),(P3),(P4) \}
  \end{align*}
  \]
Relations are Boolean matrices

The same in matrix form:

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Note how *me*, *me.parent* etc are all at most \( \text{Person} \leftarrow 1 \).
Functions are Boolean matrices

A relation $B \leftarrow^V A$ is said to be a vector if either $A$ or $B$ are the singleton type $1$.

Relation $1 \leftarrow^V A$ is said to be a row-vector; clearly, $V \subseteq !$.

Relation $B \leftarrow^V 1$ is said to be a column-vector; clearly, $V \subseteq !°$.

Functions are Boolean matrices $f$ such that

$$! \cdot f = !$$

(1)

for instance $[1 \ 1] \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1 \ 1 \ 1]$.

NB: mind the two polymorphic copies of $!: A \rightarrow 1$, the everywhere-$1$ function.
Probabilistic functions

Both \( f \) and \( g \) satisfy (1); 
\[
\begin{bmatrix}
1 & 0 & 0.5 & 0.5 \\
0 & 0.5 & 0 & 0.5 \\
0 & 0.5 & 0.5 & 0
\end{bmatrix}
\]
Probabilistic functions vs relations

Galois connection

\[ [f] \subseteq R \Leftrightarrow f \leq [R] \]  \hspace{1cm} (2)

such that

\[ [[R]] = R \]

— that is, \([\_]\) is injective and \([-\] is surjective.

This enables us to regard the latter (supports) as a relational abstract interpretation of probabilistic functions (PF).

The preorder \((\leq)\) should rank PFs according to (lack of) uniformity.
Monoidal categories

Every stage in the hierarchy forms a **monoidal category** whose **composition** has been given already, and whose **tensor** is based on **pairing**: 

\[ M \otimes N = (M \cdot \text{fst}) \triangleright (N \cdot \text{snd}) \]  

(3)

Pairing is a **weak product** for matrices (PFs included):

\[ k = M \triangleright N \Rightarrow \begin{cases} \text{fst} \cdot k = M \\ \text{snd} \cdot k = N \end{cases} \]

For **pure** functions this becomes a **full** categorial product.\(^1\)

\[^1\text{See e.g. (Murta and Oliveira, 2015; Oliveira and Micaldi, 2016).}\]
Control a *sprinkler* to wet the *grass* in case it does not *rain*.

Example adapted from [https://en.wikipedia.org/wiki/Bayesian_network](https://en.wikipedia.org/wiki/Bayesian_network)
Functional (deterministic) model

\[ S = R = G = 2 \]

\[ \text{sprinkler} : R \rightarrow S \]
\[ \text{sprinkler} \ r = \neg \ r \]

\[ \text{grass} : S \times R \rightarrow G \]
\[ \text{grass} (s, r) = s \lor r \]

\[ \text{rain} \in \{0, 1\} \]

Grass always wet:

\[ \text{grass} (\text{sprinkler} \ r, r) = \neg \ r \lor r = \top \]

Altogether, two possible states \[ \{(1, (1, 0)), (1, (0, 1))\} \] of type:

\[ G \times (S \times R) \xleftarrow{\text{state}} 1 = (\text{grass} \downarrow \text{id}) \cdot (\text{sprinkler} \downarrow \text{id}) \cdot \text{rain} \]
Bayesian networks

Previous model is not realistic — the picture actually found on Wikipedia is:
Bayesian network (probabilistic model)

Let

\[ S = R = G = 2 \]
\[ S \xleftarrow{\text{sprinkler}} R = \begin{bmatrix} 0.60 & 0.99 \\ 0.40 & 0.01 \end{bmatrix} \]
\[ R \xleftarrow{\text{rain}} 1 = \begin{bmatrix} 0.80 \\ 0.20 \end{bmatrix} \]
\[ G \xleftarrow{\text{grass}} S \times R = \begin{bmatrix} 1.00 & 0.20 & 0.10 & 0.01 \\ 0 & 0.80 & 0.90 & 0.99 \end{bmatrix} \]

The “same” state arrow

\[ G \times (S \times R) \xleftarrow{\text{state}} 1 = (g_{\text{grass}} \triangleleft \text{id}) \cdot (s_{\text{sprinkler}} \triangleleft \text{id}) \cdot \text{rain} \]

but over a different category (next slide).
Bayesian network (probabilistic model)

\[
G \times (S \times R) \xleftarrow{\text{state}} 1 =
\]

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>off</td>
<td>no</td>
<td>0.4800</td>
</tr>
<tr>
<td></td>
<td>dry</td>
<td>yes</td>
<td>0.0396</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>no</td>
<td>0.0320</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>off</td>
<td>no</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>wet</td>
<td>yes</td>
<td>0.1584</td>
</tr>
<tr>
<td></td>
<td>on</td>
<td>no</td>
<td>0.2880</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes</td>
<td>0.0020</td>
</tr>
</tbody>
</table>

Moreover, we can define

\[
1 \xleftarrow{\text{grass\_wet}} G \times (S \times R) = [0 \ 1] \cdot \text{fst}
\]

\[
1 \xleftarrow{\text{raining}} G \times (S \times R) = [0 \ 1] \cdot \text{snd} \cdot \text{snd}
\]

etc. to obtain e.g. \( P_{\text{state}}(\text{grass\_wet}) = \text{grass\_wet} \cdot \text{state} = 44.84\% \).
Bayesian network querying

Conditional probabilities over state distribution $\delta$:

$$P_\delta(a \mid b) = \frac{(a \times b) \cdot \delta}{b \cdot \delta} \quad \text{where} \quad 1 \leftarrow^{a,b} S \leftarrow^{\delta} 1$$

Boolean vectors $a$ and $b$ describe event sets.

Recall

$$S \times (G \times R) \leftarrow^{\text{sprinkler} \downarrow\text{(grass} \cdot \text{sprinkler} \downarrow \text{id}) \downarrow \text{id)}} R$$

Forwards: $P_\delta(\text{grass_wet} \mid \text{raining}) = 80.19\%$

Backwards: $P_\delta(\text{raining} \mid \text{grass_wet}) = 35.77\%$
By the way

**Bayes theorem:**

\[
P(a \mid b) = P(b \mid a) \frac{P(a)}{P(b)}
\]  

(5)

cf. (assuming \( \delta : 1 \rightarrow S \)):

\[
P_\delta(a \mid b) = P_\delta(b \mid a) \frac{P_\delta(a)}{P_\delta(b)}
\]

\[
\Leftrightarrow \quad \{ \text{trivial} \}
\]

\[
P_\delta(a \mid b) P_\delta(b) = P_\delta(b \mid a) P_\delta(a)
\]

\[
\Leftrightarrow \quad \{ \text{(4) twice} \}
\]

\[
(a \times b) \cdot \delta = (b \times a) \cdot \delta
\]
Towards probabilistic contracts

$P_\delta(raining \mid grass\_wet) = 35.77\%$ — *backwards* reasoning — is suggestive of (weakest) **precondition** validation — in a sense, it tells how important raining is as **cause** for the grass to be wet (**effect**).

Note that probabilistic function

$$f : R \rightarrow S \times G$$

$$f = sprinkler \downarrow (grass \cdot (sprinkler \downarrow id))$$

that is,

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>off</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dry</td>
<td>0.384</td>
<td>0.196</td>
</tr>
<tr>
<td>wet</td>
<td>0.216</td>
<td>0.794</td>
</tr>
<tr>
<td><strong>on</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dry</td>
<td>0.256</td>
<td>0.002</td>
</tr>
<tr>
<td>wet</td>
<td>0.144</td>
<td>0.008</td>
</tr>
</tbody>
</table>

describes a system that **reacts** to raining.
Towards probabilistic contracts

In what sense — measure? — can we say that some \( f \) satisfies the contract

\[
\text{grass}_\text{wet} \xleftarrow{f} \text{raining}
\]  

(6)

and what does (6) mean?

Back to \textbf{pure} function \( f : Y \leftarrow X \):

\[
q \xleftarrow{f} p \iff \langle \forall x : x \in X : p x \Rightarrow q (f x) \rangle
\]

or, if you wish,

\[
q \xleftarrow{f} p \iff \neg \langle \exists x : x \in X : p x \land \neg q (f x) \rangle
\]
Towards probabilistic contracts

That is, model checking $q \leftarrow^f p$ means finding those $x \in X$ that violate the contract.

In a probabilistic setting, such $x \in X$ are captured by a distribution $\delta : 1 \rightarrow X$.

Term $q(f \cdot x)$ will then correspond to scalar $1 \leftarrow^q f \cdot x$ — a probability.

But first we have to regard $f$ as a (kind of) probabilistic relation, as in the Bayesian network above — we need to have access to the I/O behaviour of $f$. 
Towards probabilistic contracts

Recall that a probabilistic function \( f : A \rightarrow B \) (PF) is half way between pure functions and relations: \( \text{!} \cdot f = \text{!} \) holds and their support \( \lfloor f \rfloor \) is a relation of the same type \( A \rightarrow B \).

Below we will take PF

\[
\begin{array}{c|ccc}
  & a_1 & a_2 & a_3 \\
\hline
b_1 & 0.7 & 0.01 & 1 \\
\hline
b_2 & 0.3 & 0.99 & 0 \\
\end{array}
\]

as example, with support

\[
\begin{array}{c|ccc}
  & a_1 & a_2 & a_3 \\
\hline
b_1 & 1 & 1 & 1 \\
\hline
b_2 & 1 & 1 & 0 \\
\end{array}
\]

a relation which can be mapped back to the PF

\[
\begin{array}{c|ccc}
  & a_1 & a_2 & a_3 \\
\hline
b_1 & 0.5 & 0.5 & 1 \\
\hline
b_2 & 0.5 & 0.5 & 0 \\
\end{array}
\]

as seen before.
Towards probabilistic contracts

The I/O behaviour of \( f \)
knowing the distribution \( \delta \)
of the inputs is given by \( \gamma \)
in
\[
\begin{array}{ccc}
\delta & \downarrow \gamma & 1 \\
B \times A & \leftarrow f \circ \text{id} & \rightarrow A \\
\end{array}
\]

Example (\( f \) as before):

\[
\begin{array}{ccc}
\delta & |
\hline
A & 0.1 \\
a_1 & 0.1 \\
a_2 & 0.2 \\
a_3 & 0.7 \\
\end{array}
\]

\[
\begin{array}{ccc}
B \times A & |
\hline
(b_1, a_1) & 0.070 \\
(b_1, a_2) & 0.002 \\
(b_1, a_3) & 0.300 \\
(b_2, a_1) & 0.030 \\
(b_2, a_2) & 0.198 \\
(b_2, a_3) & 0 \\
\end{array}
\]

Then

\[
\gamma =
\]

\[
\begin{array}{ccc}
\delta & |
\hline
A & 0.1 \\
a_1 & 0.1 \\
a_2 & 0.2 \\
a_3 & 0.7 \\
\end{array}
\]

\[
\begin{array}{ccc}
B \times A & |
\hline
(b_1, a_1) & 0.070 \\
(b_1, a_2) & 0.002 \\
(b_1, a_3) & 0.300 \\
(b_2, a_1) & 0.030 \\
(b_2, a_2) & 0.198 \\
(b_2, a_3) & 0 \\
\end{array}
\]


Measuring probabilistic contracts

Let us define:

\[
\llbracket q \leftarrow f \ p \rrbracket_\delta = P_\gamma(q \cdot \text{fst} \mid p \cdot \text{snd})
\]  

where \( \gamma = (f \lor \text{id}) \cdot \delta \) — check the following diagram:

In the next slide we show that

\[
\llbracket q \leftarrow f \ p \rrbracket_\delta = (q \boxdot p) \cdot (f \lor \text{id}) \cdot \delta
\]

where \( p \boxdot q \) abbreviates \( p \cdot \text{fst} \times q \cdot \text{snd} \).
Measuring probabilistic contracts

Spelling out the meaning of $q \leftarrow^f p$ given distribution $\delta$ of inputs:

$$\llbracket q \leftarrow^f p \rrbracket_\delta$$

$$= \begin{cases} \text{definition (7)} \end{cases}$$

$$P_{(f \uparrow id) \cdot \delta}(q \cdot fst \mid p \cdot snd)$$

$$= \begin{cases} \text{definition (4)} \end{cases}$$

$$\frac{(q \boxdot p) \cdot (f \uparrow id) \cdot \delta}{p \cdot snd \cdot (f \uparrow id) \cdot \delta}$$

$$= \begin{cases} \text{go pointwise} \end{cases}$$

$$\langle \sum b, a : q b \land p a : (\delta a) (b f a) \rangle$$

$$P_\delta(p)$$

Case $p = \text{true}$ simplifies (a lot!) to $\llbracket q \leftarrow^f \text{true} \rrbracket_\delta = q \cdot f \cdot \delta$. 
Measuring probabilistic contracts

Example, recalling

\[
\begin{array}{c|ccc}
   & a_1 & a_2 & a_3 \\
 b_1 & 0.7 & 0.01 & 1 \\
 b_2 & 0.3 & 0.99 & 0 \\
\end{array}
\quad \text{and} \quad
\begin{array}{c|c}
   & a_1 & 0.1 \\
 a_1 & 0.1 \\
 a_2 & 0.2 \\
 a_3 & 0.7 \\
\end{array}
\]

Then, for instance,

\[
\{ b_2 \} \overset{f}{\prec} \{ a_1, a_2 \} = 76\%
\]

\[
\{ b_2 \} \overset{f}{\prec} \{ a_3 \} = 0\%
\]

\[
\{ b_2 \} \overset{f}{\prec} \text{true} = 22.8\%
\]

\[
\text{true} \overset{f}{\prec} \{ a_1, a_2 \} = 100\%
\]

etc
Measuring probabilistic contracts

I didn’t make a full sanity check of the definition, but some expected laws are easy to check, cf. e.g.

\[
\llbracket \text{true } \xleftarrow{f} p \rrbracket_\delta = \begin{cases} \text{unfold definition, true } = \top \text{ (1s everywhere)} \end{cases}
\]

\[
(\top \times p \cdot \text{snd}) \cdot (f \triangledown \text{id}) \cdot \delta
\]

\[
p \cdot \text{snd} \cdot (f \triangledown \text{id}) \cdot \delta
\]

\[
\begin{cases} \text{Hadamard product: } \top \times M = M \end{cases}
\]

\[
p \cdot \text{snd} \cdot (f \triangledown \text{id}) \cdot \delta
\]

\[
p \cdot \text{snd} \cdot (f \triangledown \text{id}) \cdot \delta
\]

\[
\begin{cases} \text{trivia} \end{cases}
\]

1

□
Measuring probabilistic contracts

However,

\[
\begin{align*}
\begin{bmatrix} p \leftarrow g \text{ false} \end{bmatrix}_\delta &= \frac{(p \cdot \text{fst} \times \bot \cdot \text{snd}) \cdot (g \circ id) \cdot \delta}{\bot \cdot \delta} = \frac{0}{0}
\end{align*}
\]

is mathematically undetermined (inconsistency).

Sequential rule — my guess is something like:

\[
\begin{align*}
\begin{bmatrix} q \leftarrow f \cdot g \text{ r} \end{bmatrix}_\delta &\geq \begin{bmatrix} q \leftarrow f \text{ p} \end{bmatrix}_{g \cdot \delta} \times \begin{bmatrix} p \leftarrow g \text{ r} \end{bmatrix}_\delta
\end{align*}
\]

But be warned that, in the setting of McIver and Morgan (2005), probabilistic Hoare triples are not compositional — will our definition above remedy this?

(Future work.)
Model checking probabilistic contracts

In Alloy, given

```
assert contract { all a:A | p[a] => q[a.f] }
```

executing

```
check contract for ... A
```

means finding \( a \) such that \( p\ a \land \neg q\ (f\ a) \) holds.

Assume \( f \) **probabilistic** in ”Alloy++” :-). Then:

```
check contract >= 80% for ... A
```

would mean finding \( \delta \) such that \( [ q \xleftarrow{f} p ]_\delta < 0.8 \).
Probabilistic Alloy?

Doable?

Hope so. (Free Alloy run-time matrices from the Boolean dictatorship.)

There have been experiments with Alloy over multiset — see e.g. Relational Modeling and Reasoning with Multisets and Multirelations in Alloy \cite{sun16}.

At home: Alcino + JNO had some ideas about a quantitative Alloy, several years ago... Weighted relations... Automated analysis with an SMV solver...
Probabilistic Alloy?

Excerpt from our unpublished (incomplete) note:

Syntactically we propose no extensions to the Alloy language. On the contrary, we propose to reduce it by removing arithmetic operators and the \textit{sum} quantifier.

Semantically, the meaning of Alloy’s “dot-join” is relaxed to numeric matrix-matrix and matrix-vector operations. (It already is so, but restricted to Boolean 0 and 1. Within 0s and 1s normal multiplication implements intersection and union is normal addition minus intersection (Oliveira, 2012) (...)

\textbf{Trust is the right project to come back to this topic!}
There is work on the literature about **conditioning** in probabilistic programming, see e.g. (Gretz et al., 2015), (McIver and Morgan, 2005), which can be of help.

Also algorithms that use ideas from program analysis in probabilistic programming, see e.g. (Nori et al., 2014).
Afterthought

(The drawing again!)

Recall matrix supports.

Can the current Alloy relational engine help in finding the counter-example distributions, as a kind of AI?

\( f = \lfloor - \rfloor, \ g = \lceil - \rceil \)

etc.
References


J.N. Oliveira and V.C. Miraldo. “Keep definition, change category”