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"

(in *"Relativity Theory of Electrons and Protons"*, 1936).

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

I agree — standard **matrix notation** is clumsy by **modern computer science** standards.

Prelude

First of all, don't refer to it as **multiplication** but rather as **composition**, as it generalizes **function** composition

$$(f \cdot g)\, x \quad = \quad f\, (g\, x)$$

and **relation** composition,

$$y\, (R \cdot S)\, x \quad \Leftrightarrow \quad \langle \exists\, z \;::\; y\, R\, z \wedge z\, S\, x \rangle$$

compare with

$$y\, (M \cdot N)\, x \quad = \quad \langle \sum z \;::\; (y\, R\, z) \times (z\, S\, x) \rangle$$

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

In my view, any modelling tool should live peacefully with this *function* $\rightarrow$ *relation* $\rightarrow$ *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:

$$
\begin{aligned}
&Person = \{(P1),(P2),(P3),(P4)\}\\
&parent = \{(P1,P2),(P1,P3),(P2,P4)\}\\
&me = \{(P1)\}\\
&me.parent = \{(P2),(P3)\}\\
&me.parent.parent = \{(P4)\}\\
&Person.parent = \{(P2),(P3),(P4)\}
\end{aligned}
$$

Relations are Boolean matrices

The same in matrix form:

					1	
				P1	1	me
				P2	0	
				P3	0	
parent	P1	P2	P3	P4	0	
P1	0	0	0	0	0	me.parent
P2	1	0	0	0	1	
P3	1	0	0	0	1	
P4	0	1	0	0	0	
P1	0	0	0	0	0	me.parent.parent
P2	1	0	0	0	0	
P3	1	0	0	0	0	
P4	0	1	0	0	1	

Note how me, $me.parent$ etc are all at most $Person \xleftarrow{!^{\circ}} 1$.

Functions are Boolean matrices

A relation $B \xleftarrow{V} A$ is said to be a **vector** if either A or B are the singleton type 1.

Relation $1 \xleftarrow{V} A$ is said to be a **row**-vector; clearly, $V \subseteq\ !$

Relation $B \xleftarrow{V} 1$ is said to be a **column**-vector; clearly, $V \subseteq\ !^\circ$

Functions are Boolean matrices f such that

$$! \cdot f = ! \tag{1}$$

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

NB: mind the two **polymorphic** copies of $! : A \to 1$, the everywhere-1 function.

Probabilistic functions

$$\begin{tikzcd} \textit{Function} \arrow[d] \\ \textit{Probabilistic function} \arrow[d, "\lfloor\cdot\rfloor"'] \\ \textit{Relation} \arrow[u, bend right, "\lceil\cdot\rceil"'] \arrow[d] \\ \textit{Matrix} \end{tikzcd}$$

$$f = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$g = \begin{bmatrix} 1 & 0 & 0.5 & 0.1 \\ 0 & 0.7 & 0 & 0.9 \\ 0 & 0.3 & 0.5 & 0 \end{bmatrix}$$

$$R = \lfloor g \rfloor = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Both f and g satisfy (1) ; $\lceil R \rceil = \begin{bmatrix} 1 & 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix}$

Probabilistic functions vs relations

Galois connection

$$\lfloor f \rfloor \subseteq R \Leftrightarrow f \leqslant \lceil R \rceil \tag{2}$$

such that

$$\lfloor \lceil R \rceil \rfloor = R$$

— that is, $\lceil \_ \rceil$ is **injective** and $\lfloor \_ \rfloor$ is **surjective**.

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

The preorder $(\leqslant)$ 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**:

$$M \otimes N = (M \cdot fst) \triangledown (N \cdot snd) \tag{3}$$

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

$$k = M \triangledown N \Rightarrow \begin{cases} fst \cdot k = M \\ snd \cdot k = N \end{cases}$$

$$\begin{tikzcd} A & A \times B \arrow[l, "fst"'] \arrow[r, "snd"] & B \\ & C \arrow[ul, "M"] \arrow[u, "M \triangledown N"] \arrow[ur, "N"'] & \end{tikzcd}$$

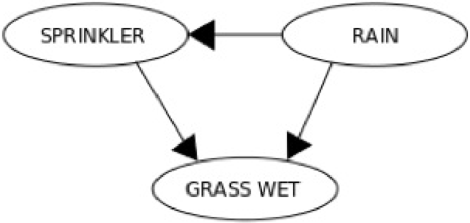
For **pure** functions this becomes a **full** categorial product.[1]

[1] See e.g. (Murta and Oliveira, 2015; Oliveira and Miraldo, 2016).

Illustration

Example adapted from

[https://en.wikipedia.org/wiki/Bayesian_network]



Control a **sprinkler** to wet the **grass** in case it does not **rain**.

Functional (deterministic) model

$S = R = G = 2$

$sprinkler : R \to S$
$sprinkler\ r = \neg\ r$

$grass : S \times R \to G$
$grass\ (s, r) = s \vee r$

$rain \in \{0, 1\}$

$$
\begin{tikzcd}
G \times (S \times R) \\
S \times R \arrow[u, "grass \triangledown id"'] \\
R \arrow[u, "sprinkler \triangledown id"'] \\
1 \arrow[u, "rain"']
\end{tikzcd}
$$

Grass always wet:

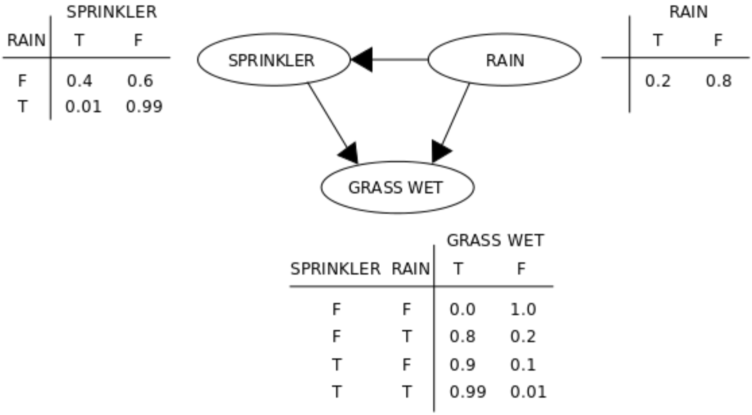
$$grass\ (sprinkler\ r, r) = \neg\ r \vee r = \mathrm{T}$$

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

$$G \times (S \times R) \xleftarrow{state} 1 \ = (grass \triangledown id) \cdot (sprinkler \triangledown id) \cdot rain$$

Bayesian networks

Previous model is not realistic — the picture actually found on Wikipedia is:



Bayesian network (probabilistic model)

Let

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

$$\begin{tikzcd} G \times (S \times R) \\ S \times R \arrow[u, "grass \triangledown id"'] \\ R \arrow[u, "sprinkler \triangledown id"'] \\ 1 \arrow[u, "rain"'] \end{tikzcd}$$

The "same" state arrow

$$G \times (S \times R) \xleftarrow{state} 1 = (grass \triangledown id) \cdot (sprinkler \triangledown id) \cdot rain$$

but over a different **category** (next slide).

Bayesian network (probabilistic model)

$$G \times (S \times R) \xleftarrow{state} 1 \quad =$$

G	S	R	
dry	off	*no*	0.4800
		yes	0.0396
	on	*no*	0.0320
		yes	0.0000
wet	off	*no*	0.0000
		yes	0.1584
	on	*no*	0.2880
		yes	0.0020

Moreover, we can define

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

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

etc. to obtain e.g. $\mathrm{P}_{state}(grass\_wet) = grass\_wet \cdot state = 44.84\%$.

Bayesian network querying

Conditional probabilities over state distribution δ:

$$\mathrm{P}_\delta(a \mid b) = \frac{(a \times b) \cdot \delta}{b \cdot \delta} \quad \textit{where} \quad 1 \xleftarrow{a,b} S \xleftarrow{\delta} 1 \tag{4}$$

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

Recall

$$\begin{tikzcd}
S \times (G \times R) \arrow[d, "\mathit{grass_wet}, \mathit{raining}"'] & & R \arrow[ll, "\mathit{sprinkler}^{\triangledown}((\mathit{grass} \cdot (\mathit{sprinkler}^{\triangledown}\, \mathit{id}))^{\triangledown}\, \mathit{id})"'] \\
1 & & 1 \arrow[ll, "\frac{(\mathit{grass_wet} \times \mathit{raining}) \cdot \delta}{\mathit{grass_wet} \cdot \delta}"] \arrow[u, "\mathit{rain}"'] \arrow[ull, "\delta"']
\end{tikzcd}$$

Forwards: $\mathrm{P}_\delta(\mathit{grass\_wet} \mid \mathit{raining}) = 80.19\%$

Backwards: $\mathrm{P}_\delta(\mathit{raining} \mid \mathit{grass\_wet}) = 35.77\%$

By the way

Bayes theorem:

$$\mathrm{P}(a \mid b) = \mathrm{P}(b \mid a) \, \frac{\mathrm{P}(a)}{\mathrm{P}(b)} \tag{5}$$

cf. (assuming $\delta : 1 \to S$):

$$\mathrm{P}_\delta(a \mid b) = \mathrm{P}_\delta(b \mid a) \, \frac{\mathrm{P}_\delta(a)}{\mathrm{P}_\delta(b)}$$

$\Leftrightarrow$ { trivial }

$$\mathrm{P}_\delta(a \mid b) \, \mathrm{P}_\delta(b) = \mathrm{P}_\delta(b \mid a) \, \mathrm{P}_\delta(a)$$

$\Leftrightarrow$ { (4) twice }

$$(a \times b) \cdot \delta = (b \times a) \cdot \delta$$

$\square$

Towards probabilistic contracts

$\mathrm{P}_\delta(raining \mid grass\_wet) = 35.77\%$ — *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

$$f : R \to S \times G$$
$$f = sprinkler \triangledown (grass \cdot (sprinkler \triangledown id))$$

that is,

$$f = \begin{array}{cc|cc} & & no & yes \\ \hline \text{off} & dry & 0.384 & 0.196 \\ & wet & 0.216 & 0.794 \\ \text{on} & dry & 0.256 & 0.002 \\ & wet & 0.144 & 0.008 \end{array}$$

describes a system that **reacts** to raining.

Towards probabilistic contracts

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

$$grass\_wet \xleftarrow{f} raining \tag{6}$$

and what does (6) mean?

Back to **pure** function $f : Y \leftarrow X$:

$$q \xleftarrow{f} p \quad \Leftrightarrow \quad \langle \forall\ x \ :\ x \in X :\ p\ x \Rightarrow q\ (f\ x) \rangle$$

or, if you wish,

$$q \xleftarrow{f} p \quad \Leftrightarrow \quad \neg\ \langle \exists\ x \ :\ x \in X :\ p\ x \wedge \neg\ q\ (f\ x) \rangle$$

Towards probabilistic contracts

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

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

Term $q\ (f\ x)$ will then correspond to scalar $1 \xleftarrow{q \cdot f \cdot x} 1$ — 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 $f : A \to B$ (PF) is half way between pure functions and relations: $! \cdot f = !$ holds and their support $\lfloor f \rfloor$ is a relation of the same type $A \to B$.

Below we will take PF

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

as example, with support

$$\lfloor f \rfloor = \begin{array}{c|ccc} & a_1 & a_2 & a_3 \\ \hline b_1 & 1 & 1 & 1 \\ b_2 & 1 & 1 & 0 \end{array}$$

a relation which can be mapped back to the PF

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

as seen before.

Towards probabilistic contracts

The I/O behaviour of f knowing the distribution δ of the inputs is given by γ in

$$\begin{tikzcd} & & 1 \arrow[dll, "\gamma"'] \arrow[d, "\delta"] \\ B \times A & & A \arrow[ll, "f \triangledown id"'] \end{tikzcd}$$

Example (f as before):

$$\delta = \begin{array}{c|c} A & \\ \hline a_1 & 0.1 \\ a_2 & 0.2 \\ a_3 & 0.7 \end{array}$$

Then

$$\gamma = \begin{array}{c|r} B \times A & \\ \hline (b_1, a_1) & 0.070 \\ (b_1, a_2) & 0.002 \\ (b_1, a_3) & 0.300 \\ (b_2, a_1) & 0.030 \\ (b_2, a_2) & 0.198 \\ (b_2, a_3) & 0 \end{array}$$

Measuring probabilistic contracts

Let us define:

$$[\![\, q \xleftarrow{f} p \,]\!]_\delta = \mathrm{P}_\gamma(q \cdot fst \mid p \cdot snd) \tag{7}$$

where $\gamma = (f \triangledown id) \cdot \delta$ — check the following diagram:

$$\begin{tikzcd}
& 1 \arrow[dl, "\gamma"'] \arrow[d, "\delta"] \\
B \times A \arrow[d, "fst"'] \arrow[r, bend right, "snd"'] & A \arrow[l, "f \triangledown id"'] \arrow[dl, bend left, "f"] \arrow[d, bend left, "p"] \\
B \arrow[r, bend right, "q"'] & 1
\end{tikzcd}$$

In the next slide we show that

$$[\![\, q \xleftarrow{f} p \,]\!]_\delta = \frac{(q \boxtimes p) \cdot (f \triangledown id) \cdot \delta}{p \cdot \delta} \tag{8}$$

where $p \boxtimes q$ abbreviates $p \cdot fst \times q \cdot snd$.

Measuring probabilistic contracts

Spelling out the meaning of $q \xleftarrow{f} p$ given distribution δ of inputs:

$$
\begin{array}{ll}
& [\![\, q \xleftarrow{f} p \,]\!]_{\delta} \\
= & \qquad \{\ \text{definition (7)}\ \} \\
& \mathrm{P}_{(f \triangledown id)\cdot\delta}(q \cdot fst \mid p \cdot snd) \\
= & \qquad \{\ \text{definition (4)}\ \} \\
& \dfrac{(q \boxtimes p)\cdot(f \triangledown id)\cdot\delta}{p \cdot snd \cdot (f \triangledown id)\cdot\delta} \\
= & \qquad \{\ \text{go pointwise}\ \} \\
& \dfrac{\langle \sum\ b, a\ :\ q\ b \wedge p\ a :\ (\delta\ a)\ (b\ f\ a)\rangle}{\mathrm{P}_{\delta}(p)}
\end{array}
$$

Case $p = true$ simplifies (a lot!) to $[\![\, q \xleftarrow{f} true \,]\!]_{\delta} = q \cdot f \cdot \delta$.

Measuring probabilistic contracts

Example, recalling

$$f = \begin{array}{c|ccc} & a_1 & a_2 & a_3 \\ \hline b_1 & 0.7 & 0.01 & 1 \\ b_2 & 0.3 & 0.99 & 0 \end{array} \quad \textit{and} \quad \delta = \begin{array}{c|c} & \\ \hline a_1 & 0.1 \\ a_2 & 0.2 \\ a_3 & 0.7 \end{array}$$

Then, for instance,

$$\{b_2\} \xleftarrow{f} \{a_1, a_2\} = 76\%$$

$$\{b_2\} \xleftarrow{f} \{a_3\} = 0\%$$

$$\{b_2\} \xleftarrow{f} \mathit{true} = 22.8\%$$

$$\mathit{true} \xleftarrow{f} \{a_1, a_2\} = 100\%$$

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.

$$
\begin{array}{ll}
& [\![\, true \xleftarrow{f} p \,]\!]_\delta \\
= & \qquad \{ \text{ unfold definition, } true = \top \text{ (1s everywhere) } \} \\
& \dfrac{(\top \times p \cdot snd) \cdot (f \triangledown id) \cdot \delta}{p \cdot snd \cdot (f \triangledown id) \cdot \delta} \\
= & \qquad \{ \text{ Hadamard product: } \top \times M = M \ \} \\
& \dfrac{p \cdot snd \cdot (f \triangledown id) \cdot \delta}{p \cdot snd \cdot (f \triangledown id) \cdot \delta} \\
= & \qquad \{ \text{ trivia } \} \\
& 1 \\
\Box &
\end{array}
$$

Measuring probabilistic contracts

However,

$$[\![\, p \xleftarrow{g} \mathit{false} \,]\!]_\delta = \frac{(p \cdot \mathit{fst} \times \bot \cdot \mathit{snd}) \cdot (g \triangledown \mathit{id}) \cdot \delta}{\bot \cdot \delta} = \frac{0}{0}$$

is mathematically undetermined (inconsistency).

Sequential rule — my guess is something like:

$$[\![\, q \xleftarrow{f \cdot g} r \,]\!]_\delta \geqslant [\![\, q \xleftarrow{f} p \,]\!]_{g \cdot \delta} \times [\![\, p \xleftarrow{g} r \,]\!]_\delta$$

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 $p\ a \wedge \neg\ q\ (f\ a)$ holds.

Assume f **probabilistic** in "Alloy++" **:-)**. Then:

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

would mean finding δ 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 **multisets** — see e.g. *Relational Modeling and Reasoning with Multisets and Multirelations in Alloy* (Sun et al., 2016).

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* 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) (...)*

<u>TRUST</u> IS THE RIGHT PROJECT TO COME BACK TO THIS TOPIC!

Literature

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).

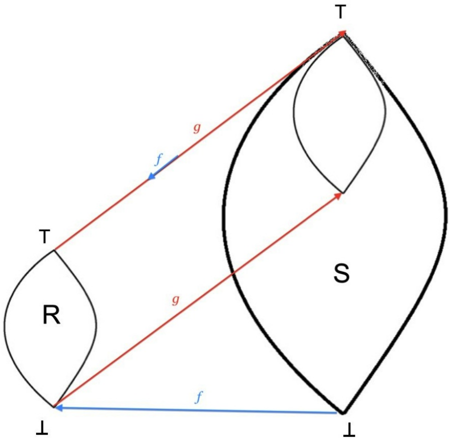
Afterthought

(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

F. Gretz, N. Jansen, B.L. Kaminski, J.-P. Katoen, A. McIver, and F. Olmedo. Conditioning in probabilistic programming. *CoRR*, abs/1504.00198, 2015. URL http://arxiv.org/abs/1504.00198.

A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer-Verlag, 2005. ISBN 0387401156.

D. Murta and J.N. Oliveira. A study of risk-aware program transformation. *SCP*, 110:51–77, 2015.

A.V. Nori, C.-K. Hur, S.K. Rajamani, and S. Samuel. R2: an efficient MCMC sampler for probabilistic programs. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 2476–2482, 2014. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8192.

J.N. Oliveira. Towards a linear algebra of programming. *FAoC*, 24 (4-6):433–458, 2012.

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

— a practical approach to state-based system calculi. *JLAMP*, 85(4):449–474, 2016.

Peiyuan Sun, Zinovy Diskin, Michał Antkiewicz, and Krzysztof Czarnecki. Relational Modeling and Reasoning with Multisets and Multirelations in Alloy. (GSDLAB-TR 2016-01-22), 2016.