Alloy meets the AoP — "Relational thinking" at work

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

2009 (last update: Nov. 2013)
Braga, Portugal

Model driven engineering

Model driven engineering (**MDE**) is a voluminous area of work, full of approaches, acronyms, notations.

UML has taken the lead in *unifying* such notations, but it is too **informal** to be accepted as a reference (formal) approach.

Model-oriented formal methods — eg. **VDM** [3], **Z** [6] — solve this informality problem at a high-cost: many people find it hard to understand models written in maths (cf. maths illiteracy if not mathphobic behaviour).

Alloy [2] offers a good compromise — it is formal in a light-weight manner.

Alloy

What **Alloy** offers

- A unified approach to **modeling** based on the notion of a **relation — "everything is a relation"** in Alloy.
- A minimal syntax (centered upon relational composition) with an object-oriented flavour which captures much of what otherwise would demand for **UML+OCL**.
- A **pointfree** subset.
- A model-checker for model assertions (counter-examples within scope).

Alloy

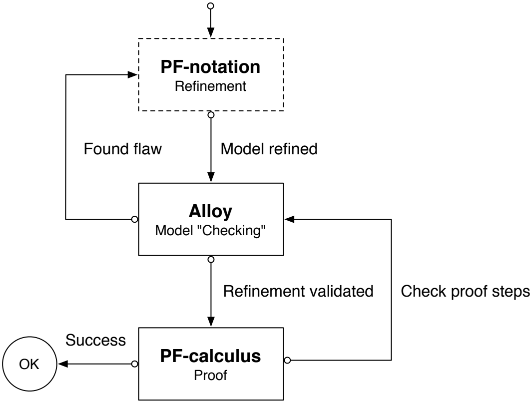
What **Alloy does not** offer

- Complete calculus for deduction (proof theory)
- Strong type checking

Opportunities

- Enrich the standard Alloy *modus operandi* with **relational algebra** (vulg. **AoP** [1], algebra of programming) calculational proofs
- Follow an Alloy-centric **design method** for high assurance model-oriented design

Life-cycle



Source: [5]

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	
P4	0	

parent	P1	P2	P3	P4		
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 $Person \xleftarrow{!^\circ} 1$.

By the way

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, $V \subseteq\ !^\circ$

Every vector $1 \xleftarrow{V} A$ can be uniquely represented by **coreflexive** (diagonal) $\delta\, V$. Conversely, every coreflexive Φ can be represented by vector $! \cdot \Phi$. This arises from:

$$\Phi \subseteq \Psi \quad \Leftrightarrow \quad ! \cdot \Phi \subseteq\ ! \cdot \Psi \tag{161}$$

— recall exercise 60.

When "everything is a relation"

In Alloy:

- Sets are relations of arity 1 (ie. **vectors**) , eg.

$$Person = \{(P1), (P2), (P3), (P4)\}$$

- Scalars are relations with size 1, eg. $me = \{(P1)\}$
- Relations are first order, but there are multi-ary relations.
- However, **Alloy** relations are not *n*-ary in the usual sense: instead of thinking of $R \in 2^{A \times B \times C}$ as a set of triples (there is no such thing as *tupling* in Alloy), think of R in terms of *currying*:

$$R \in (B \to C)^A$$

(More about this later.)

Kleene algebra flavour

Basic operators:

.	*composition*
+	*union*
^	*transitive closure*
*	*transitive-reflexive closure*

(There is no explicit recursion is **Alloy**.) Other relational operators:

~	*converse*
++	*override*
&	*intersection*
-	*difference*
->	*Cartesian product*
<:	*domain restriction*
:>	*range restriction*

Semantic rules

Semantic rules for the *at most* ordering, intersection, union and converse:

$$
\begin{aligned}
[\![R \;\; \mathrm{in} \;\; S]\!] &= [\![R]\!] \subseteq [\![S]\!] && (162)\\
[\![R \,\&\, S]\!] &= [\![R]\!] \cap [\![S]\!] && (163)\\
[\![R + S]\!] &= [\![R]\!] \cup [\![S]\!] && (164)\\
[\![\tilde{\ } R]\!] &= [\![R]\!]^{\circ} && (165)
\end{aligned}
$$

Basic facts:

$$
\begin{aligned}
[\![\mathit{no} \;\; R]\!] &= [\![R]\!] \subseteq \bot && (166)\\
[\![\mathit{some} \;\; R]\!] &= [\![R]\!] \supset \bot && (167)\\
[\![\mathit{lone} \;\; R]\!] &= \langle \exists\, a, b \;::\; [\![R]\!] \subseteq \underline{b} \cdot \underline{a}^{\circ} \rangle && (168)
\end{aligned}
$$

Semantic rules

Alloy's syntax for $\top$ makes types explicit:

$$[\![A \to B]\!] \;=\; [\![B]\!] \cdot \top \cdot [\![A]\!] \tag{169}$$

Types A and B are sets which, in our semantics, will be captured by *coreflexives*.

In general, given a set $s : A$, we have the semantic rule

$$[\![s]\!] \;=\; A \xleftarrow{\Phi_s} A \tag{170}$$

The largest such s is A itself, represented by the largest such coreflexive: the identity id_A.

Semantic rules

Restricted to binary relations, "dot join" is (forward) binary relation composition:

$$[\![S.R]\!] = [\![R]\!] \cdot [\![S]\!] \qquad C \xleftarrow{[\![R]\!]} B \xleftarrow{[\![S]\!]} A \tag{171}$$

Dot join can be used in Alloy between relations which are not binary, eg. sets (unary relations, or **vectors**). We have the following semantic rule in the first case,

$$[\![s.R]\!] = \underbrace{\rho\,([\![R]\!] \cdot [\![s]\!])}_{sp(R,s)} \qquad C \xleftarrow{[\![s.R]\!]} C \xleftarrow{[\![R]\!]} B \xleftarrow{[\![s]\!]} B \tag{172}$$

for R binary and s a set (unary relation).

Semantic rules

Expression $sp(R, s)$ under the brace provides an explanation for this kind of composition: it yields the *strongest post-condition* ensured by R once pre-conditioned by s.

Thanks to

$$[\![R.s]\!] \;=\; [\![s.\tilde{}R]\!] \tag{173}$$

[2] one has

$$[\![R.s]\!] = \delta\,([\![s]\!] \cdot [\![R]\!]) \tag{174}$$

where δ is the domain combinator.

Semantic rules

In case R is a function f and s is a scalar x (that is, a singleton vector), then $[\![x.f]\!]$ is the scalar $f\ x$.

Functions are declared using a suitable **multiplicity keyword**, eg.

```
sig A { f : one B }
```

Thus,

```
sig A { f : one B, g : one C }
```

declares $\langle f, g \rangle$, etc.

The table in the next slide gives the semantics of Alloy's **multiplicities** in relation algebra.

Multiplicities in Alloy + taxonomy

<table>
<tr><td colspan="3"><code>A lone -> B</code>
injective</td><td colspan="3"><code>A -> some B</code>
entire</td><td colspan="3"><code>A -> lone B</code>
simple</td><td colspan="3"><code>A some -> B</code>
surjective</td></tr>
<tr><td colspan="4"><code>A lone -> some B</code>
representation</td><td colspan="4"><code>A -> one B</code>
function</td><td colspan="4"><code>A some -> lone B</code>
abstraction</td></tr>
<tr><td colspan="6"><code>A lone -> one B</code>
injection</td><td colspan="6"><code>A some -> one B</code>
surjection</td></tr>
<tr><td colspan="12"><code>A one -> one B</code>
bijection</td></tr>
</table>

(courtesy of Alcino Cunha)

Recalling terminology

Topmost criteria:

binary relation

injective entire simple surjective

Definitions:

	Reflexive	*Coreflexive*
$\ker R$	entire R	injective R
$\mathrm{img}\, R$	surjective R	simple R

$$\ker R = R^{\circ} \cdot R$$

$$\mathrm{img}\, R = R \cdot R^{\circ}$$

Exercise

Exercise 66: Tell which of the rules (166), (167), (168) could have been written with right-hand side $\top \subseteq \top \cdot [\![R]\!] \cdot \top$.

□

Exercise 67: The assertion in the following fragment of Alloy,

```
sig A { f : one B }
sig B {}

assert GC {
    all x: set A, y: set B | x.f in y <=> x in f.y
}
```

captures a "shunting rule" valid in such a language. Resort to the semantic rules given above to prove the validity of this assertion.

□

Modelling example

A conference model (adapted from Alcino Cunha): one has **papers** written by **people**, ie. the Alloy

```
sig Artigo {
autores : some Pessoa
}
```

which declares entire relation $Artigo \xrightarrow{autores} Pessoa$, and a **state** which evolves by letting papers be submitted, reviewed and (possibly) accepted:

```
sig State {
    submetido : set Artigo,
    aceite : set Artigo,
    nota : Artigo -> Pessoa -> lone Nota
}
```

Example

For each state s, $s.submetido$ and $s.aceite$ are sets, which our semantics encodes by coreflexives $Artigo \xleftarrow{[\![s.submetido]\!]} Artigo$ and $Artigo \xleftarrow{[\![s.aceite]\!]} Artigo$. We will use the obvious abbreviations given in the following diagram:

$$A \xleftarrow[Ac,Sb]{} A \xrightarrow[Aut]{} P$$

that is:

- $A = [\![Artigo]\!]$, $P = [\![Pessoa]\!]$
- $Aut = [\![autores]\!]$ (entire),
- $Ac = [\![s.aceite]\!]$, $Sb = [\![s.submetido]\!]$ (coreflexives).

What about $[\![s.nota]\!]$?

More on relational types

The Alloy type for $[\![s.nota]\!]$ is

```
Artigo -> Pessoa -> lone Nota
```

What is the semantics of types of the form $A \rightarrow B \rightarrow \cdots \rightarrow C$?

This questions deserves some pondering on relational types. Given types A, B, we write $A \rightarrow B$ to denote the set of all relations from A to B.

Let $B^A \subseteq A \rightarrow B$ denote the set of all **functions** in such a type. It's well-known that binary predicates are in bijection with binary relations, $2^{A \times B} \cong A \rightarrow B$ and that the well-known **curry** / **uncurry** isomorphism $(C^B)^A \cong C^{A \times B}$ holds.

More on relational types

These can be used to show that any of the relation types $(A \times B) \to C$, $A \to (B \times C)$ or $(B \to C)^A$ are isomorphic.

Thus, every relation R of the first type is in 1-to-1 correspondence with a function f of the third type such as

$$c\ R(a, b) \quad \Leftrightarrow \quad c(f\ a)b$$

since $f\ a$ is a relation of type $B \to C$.

This is how n-ary relations in Alloy should be interpreted: they are (higher-order) **functions** which yield (n-1)-ary relations as outputs and so on. For instance, $[\![a.(s.nota)]\!]$ is of type $Pessoa \to Nota$.

In the sequel we will represent such relations in uncurried format, as in the next example.

Example — continued

We define

$$[\![s.nota]\!] = A \times P \xrightarrow{Nt} N$$

under the above abbreviations and

- $N = [\![Nota]\!]$,
- $Nt = [\![s.nota]\!]$ (simple)

The model also declares another coreflexive on P(eople),

```
some sig Comissao in Pessoa {}
```

telling which people are in the reviewing committee, which we will denote by $P \xleftarrow{Com} P$.

Example — model

Altogether, we have diagram

$$
\begin{tikzcd}
P & P \arrow[l, "Com"'] & & & \\
& & & A \times P \arrow[ull, "\pi_2"'] \arrow[dll, "\pi_1"] \arrow[ll, phantom, "\neq"] \arrow[r, "Nt"] & \mathbb{N} \\
A & A \arrow[l, "Ac{,}Sb"] \arrow[uu, "Aut"] & & &
\end{tikzcd}
$$

where the $\neq$ signals a non-commutative triangle.

Example — invariants

The following invariants capture in Alloy the requirements of the problem:

```
fact Invariante {
    all s : State {
        s.nota in s.submetido -> Comissao -> Nota
        all a : Artigo | no a.(s.nota).Nota & a.autores
        ((s.nota).last).Pessoa in s.aceite
        all a : s.aceite | some a.(s.nota)
    }
}
```

The first one ensures that revisions submissions can only be made by committee members.

Example — invariants

Following (169), type

```
s.submetido -> Comissao -> Nota
```

corresponds to $N \xleftarrow{\top \cdot (Sb \times Com)} A \times P$ once uncurried, whereby the first invariant becomes

$$Nt \subseteq \top \cdot (Sb \times Com)$$

which could be written alternatively as

$$\delta\, Nt \subseteq Sb \times Com$$

thanks to the universal-property of the domain operator (126).

Example — invariants

The second invariant, which ensures that no author can be a reviewer of any of her/his papers,

```
all a : Artigo | no a.(s.nota).Nota &
a.autores
```

converts to:

$$
\begin{aligned}
& [\![no\ a.(s.nota).Nota\ \&\ a.autores]\!] \\
\Leftrightarrow \quad & \{\ (174)\ ;\ (163)\ \} \\
& \delta\,([\![Nota]\!] \cdot [\![a.(s.nota)]\!]) \cap [\![a.autores]\!] \subseteq \bot \\
\Leftrightarrow \quad & \{\ [\![Nota]\!] = id\ ;\ (172)\ \} \\
& \delta\,[\![a.(s.nota)]\!]) \cap \rho\,([\![autores]\!] \cdot [\![a]\!]) \subseteq \bot
\end{aligned}
$$

Example — invariants

The universal quantification can be avoided by defining a relation of the same type as *autores*, relating papers with their reviewers:

$$Rv = \quad P \xleftarrow{\pi_2 \cdot (\delta\ Nt) \cdot \pi_1^\circ} A$$

where $p(Rv)a$ means "person p has reviewed paper a". Thus $Rv \cap Aut \subseteq \bot$ must hold — the same as

$$Rv \subseteq (Aut \Rightarrow \bot)$$

where $Aut \Rightarrow \bot$ means the "negation of Aut", as we shall later see.

NB: recal from (95) that, in general, $b(R \Rightarrow S)a$ means $\neg(bRa) \vee bSa$.

Example — invariants

The semantics of the third Alloy invariant

```
((s.nota).last).Pessoa in s.aceite
```

won't be considered for the moment because it calls for a relational operator we have nor yet seen (relation **division**, coming up soon).

Finally, the fourth invariant

```
all a : s.aceite | some a.(s.nota)
```

enforcing that accepted papers have at least one mark easily converts to

$$Ac \subseteq \top \cdot Nt \cdot \pi_1^\circ$$

Example — invariants diagram

Altogether, the three invariants can be drawn as commuting rectangles as follows:

$$
\begin{tikzcd}
 & A \times P \arrow[r, "\top"] \arrow[dr, phantom, "\supseteq"] & N \\
A \times P \arrow[d, "\pi_2"'] \arrow[dr, phantom, "\rotatebox{-90}{$\subseteq$}"] & A \times P \arrow[l, "\delta\, Nt"'] \arrow[r, "Nt"] \arrow[u, "Sb \times Com"] \arrow[dr, phantom, "\rotatebox{90}{$\subseteq$}"] & N \arrow[u, "id"'] \arrow[d, "\top"] \\
P & A \arrow[l, "Aut \Rightarrow \bot"] \arrow[r, "Ac"'] \arrow[u, "\pi_1^\circ"'] & A
\end{tikzcd}
$$

Example — property

Model checking of property

```
check Propriedade {
all s : State | s.aceite in s.submetido
} for 6 but 1 State
```

(read: "only submitted papers can be accepted") finds no counter-examples.

This happens because this property, $Ac \subseteq Sb$ — equivalent to

$$Ac \subseteq \top \cdot Sb$$

(why?) — holds (next slide).

Example — proof

$$
\begin{array}{ll}
 & Ac \subseteq \top \cdot Sb \\
\Leftarrow & \quad \{ \; Ac \subseteq \top \cdot Nt \cdot \pi_1^\circ \text{ (fourth invariant) ; shunting } \} \\
 & \top \cdot Nt \subseteq \top \cdot Sb \cdot \pi_1 \\
\Leftarrow & \quad \{ \; Nt \subseteq \top \cdot (Sb \times Com) \text{ (first invariant) ; } \top \cdot \top = \top \; \} \\
 & \top \cdot (Sb \times Com) \subseteq \top \cdot Sb \cdot \pi_1 \\
\Leftarrow & \quad \{ \; \text{free theorem: } \pi_1 \cdot (R \times S) \subseteq R \cdot \pi_1 \; \} \\
 & \top \cdot (Sb \times Com) \subseteq \top \cdot \pi_1 \cdot (Sb \times Com) \\
\Leftrightarrow & \quad \{ \; \text{since } \top \cdot \pi_1 = \top \; \} \\
 & \textsc{True}
\end{array}
$$

Example — methods

Every model has a **state** and **methods** changing the state.

Typically, such methods can be of the following kinds:

- **C**reate — create a new state
- **R**ead — read the state
- **U**pdate — change the state (eg. make it "larger")
- **D**elete — delete information from the state (make it "smaller")

This is the well-known **CRUD**e interface to object manipulation in state based software systems.

How free are we to "invent" a **CRUD**e interface for our models?

Example — methods

Since the **state** is made of **relations**, one may predict how the evolution of such relations interferes with the model invariants.

For instance, in our model we have three relations which can evolve — Sb, Ac and Nt. Looking at the invariants,

$$Nt \subseteq \top \cdot (Sb \times Com) \tag{175}$$

$$\pi_2 \cdot (\delta\, Nt) \cdot \pi_1^\circ \cap Aut \subseteq \bot \tag{176}$$

$$Ac \subseteq \top \cdot Nt \cdot \pi_1^\circ \tag{177}$$

the following rules apply:

> *Relations on the upper side can always grow bigger; relations on the lower side can always go smaller; other situations call for* ***contracts*** *(pre-conditions).*

Example — methods

Clearly:

- relation Sb can always grow bigger (no problem in accepting more submissions)
- relation Ac can always get smaller (eg. deciding not to accept a paper after all [1])

This leaves out a most important relation, Nt, which has to grow somehow, otherwise no papers will ever be accepted ($Nt = \bot$ entails $Ac = \bot$).

Think of a method which adds new marks to Nt, $Nt' = Nt \cup New$, where (type checking) New is of the same type as Nt. (In Alloy: `s'.nota = s.nota + new`)

[1] But please note that we are ignoring one invariant for the time being...

Example — contracts

We need contracts ensuring (175) and (176). Our aim is to find appropriate (weakest) pre-conditions, one invariant at a time:

$$
\begin{array}{ll}
& \underbrace{Nt' \subseteq \top \cdot (Sb \times Com)}_{\text{(175) for } Nt'} \\
\Leftrightarrow & \quad \{ \ Nt' = Nt \cup New; \text{ universal-}\cup \text{ (65) } \ \} \\
& Nt \subseteq \top \cdot (Sb \times Com) \ \wedge \ New \subseteq \top \cdot (Sb \times Com) \\
\Leftrightarrow & \quad \{ \ \text{definition} \ \} \\
& (175) \wedge \underbrace{New \subseteq \top \cdot (Sb \times Com)}_{\text{WP for (175)}}
\end{array}
$$

The contract therefore is: new *marks* can only be assigned by committee members to submitted papers.

Example — contracts

Next we address (176):

$$\underbrace{\pi_2 \cdot (\delta\, Nt') \cdot \pi_1^\circ \cap Aut \subseteq \bot}_{\text{(176) for } Nt'}$$

$$\Leftrightarrow \qquad \{\ Nt' = Nt \cup New\text{; domain and composition distribute over } \cup\ \}$$

$$(\pi_2 \cdot (\delta\, Nt) \cdot \pi_1^\circ \cup \pi_2 \cdot (\delta\, New) \cdot \pi_1^\circ) \cap Aut \subseteq \bot$$

$$\Leftrightarrow \qquad \{\ \cap \text{ distributes over } \cup\ \}$$

$$(\pi_2 \cdot (\delta\, Nt) \cdot \pi_1^\circ \cap Aut) \cup (\pi_2 \cdot (\delta\, New) \cdot \pi_1^\circ \cap Aut) \subseteq \bot$$

$$\Leftrightarrow \qquad \{\ \text{universal-}\cup\ (65)\ \}$$

$$(176)\ \wedge \underbrace{\pi_2 \cdot (\delta\, New) \cdot \pi_1^\circ \cap Aut \subseteq \bot}_{\text{WP for (176)}}$$

Example — contracts

Suppose now that we we want to refine the method which ranks papers to a one-at-a-time fashion, that is, New in $Nt' = Nt \cup New$ is made of a paper a, a reviewer p and a mark n.

In our (uncurried) model this is captured by

$$New = \underline{n} \cdot \langle \underline{a}, \underline{p} \rangle^{\circ}$$

In Alloy, this is written `a -> p -> n` (curried notation).

Exercise 68: Check that $\underline{n} \cdot \langle \underline{a}, \underline{p} \rangle^{\circ} = \{(n, (a, p))\}$.

□

Below we instantiate the generic contracts calculated above for this situation.

Example — contracts

WP for (175):

$$
\begin{array}{ll}
 & New \subseteq \top \cdot (Sb \times Com) \\
\Leftrightarrow & \quad \{ \ New = \underline{n} \cdot \langle \underline{a}, \underline{p} \rangle^\circ \ \} \\
 & \underline{n} \cdot \langle \underline{a}, \underline{p} \rangle^\circ \subseteq \top \cdot (Sb \times Com) \\
\Leftrightarrow & \quad \{ \ \text{shunting} \ \} \\
 & \underline{n} \subseteq \top \cdot (Sb \times Com) \cdot \langle \underline{a}, \underline{p} \rangle \\
\Leftrightarrow & \quad \{ \ \times\text{-absorption} \ \} \\
 & \underline{n} \subseteq \top \cdot \langle Sb \cdot \underline{a}, Com \cdot \underline{p} \rangle \\
\Leftrightarrow & \quad \{ \ \text{going pointwise} \ \} \\
 & a \in [\![s.submetido]\!] \wedge p \in [\![Comissao]\!]
\end{array}
$$

Example — contracts

WP for (176):

$$
\begin{array}{ll}
& \pi_2 \cdot (\delta\ New) \cdot \pi_1^\circ \cap Aut \subseteq \bot \\
\Leftrightarrow & \quad \{\ New = \underline{n} \cdot \langle \underline{a}, \underline{p} \rangle^\circ\ \} \\
& \pi_2 \cdot (\delta\,(\underline{n} \cdot \langle \underline{a}, \underline{p} \rangle^\circ)) \cdot \pi_1^\circ \cap Aut \subseteq \bot \\
\Leftrightarrow & \quad \{\ \text{domain of composition and converse}\ \} \\
& \pi_2 \cdot (\rho\,(\langle \underline{a}, \underline{p} \rangle)) \cdot \pi_1^\circ \cap Aut \subseteq \bot \\
\Leftrightarrow & \quad \{\ \langle \underline{a}, \underline{p} \rangle \text{ is simple ; converses}\ \} \\
& \pi_2 \cdot \langle \underline{a}, \underline{p} \rangle \cdot (\pi_1 \cdot \langle \underline{a}, \underline{p} \rangle)^\circ \cap Aut \subseteq \bot \\
\Leftrightarrow & \quad \{\ \times\text{-cancellation}\ \} \\
& \underline{p} \cdot \underline{a}^\circ \cap Aut \subseteq \bot \\
\Leftrightarrow & \quad \{\ \text{introducing variables}\ \} \\
& \neg (p\ Aut\ a)
\end{array}
$$

Example — contracts

This corresponds to the Alloy `p not in a.autores`. The two calculated conditions can in fact be found in the proposed version of the method:

```
pred rever [a : Artigo, p : Pessoa, n : Nota, s,s' : State]
   // pre
   a in s.submetido
   p in Comissao
   p not in a.autores
   no p.(a.(s.nota))
   // pos
   s'.nota = s.nota + a->p->n
   n in last implies
      s'.aceite = s.aceite + a else s'.aceite = s.aceite
   s'.submetido = s.submetido
```

Exercise 69: The pre-condition of method `rever` includes yet another condition. Guess where this arises from.

□

Exercises

Exercise 70: Define a method which accepts papers, $Ac' = Ac \cup New$, and calculate the corresponding contract entiled by the invariants of the model.

$\square$

Exercise 71: Derive the Alloy code for the contract of the previous exercise for $New = \underline{a} \cdot \underline{a}^{\circ}$, that is, for the method which accepts one paper a at a time.

$\square$

Exercises

Exercise 72: The original Alloy model enforces Nt simple, cf. `nota : Artigo -> Pessoa -> lone Nota`; that is, no reviewer can assign more than one mark to a given paper. Simplicity of Nt is therefore another invariant "hidden in the notation". Resort to the the union-simplicity rule (160) to calculate the contract to impose on method $Nt' = Nt \cup New$ with respect to this requirement.

□

Exercise 73: Recall the diagram of the *starving* invariant of problem Propositio de homine et capra et lvpo:

$$\begin{tikzcd} Being \arrow[d, "where"'] \arrow[dr, phantom, "\subseteq"] & Being \arrow[l, "CanEat"'] \arrow[d, "Farmer"] \\ Bank & Being \arrow[l, "where"] \end{tikzcd}$$

Write the same in Alloy syntax.

□

2nd case study — Verified FSystem (VFS)

A real-life case study:

- **VSR** (Verified Software Repository) initiative
- **VFS** (Verified File System) on Flash Memory — challenge put forward by Rajeev Joshi and Gerard Holzmann (NASA JPL) [4]
- Two levels — POSIX level and (NAND) flash level
- Working document: **Intel ® Flash File System Core Reference Guide** (Oct. 2004) is POSIX aware.

2nd case study — Verified FSystem (VFS)

The problem (sample):

File System API Reference

intel®

4.6 FS_DeleteFileDir

Deletes a single file/directory from the media

Syntax

```
FFS_Status    FS_DeleteFileDir (
        mOS_char     *full_path,
        UINT8        static_info_type );
```

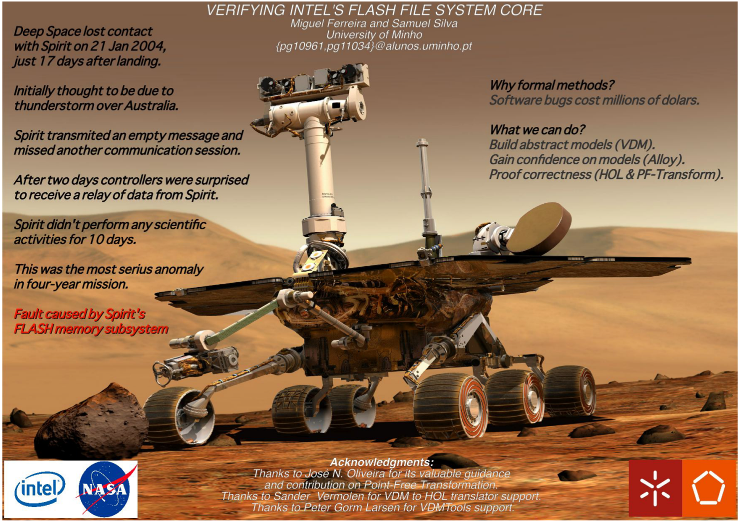
Parameters

Parameter	Description
*full_path	(IN) This is the full path of the filename for the file or directory to be deleted.
static_info_type	(IN) This tells whether this function is called to delete a file or a directory.

Error Codes/Return Values

FFS_StatusSuccess	Success
FFS_StatusNotInitialized	Failure
FFS_StatusInvalidPath	Failure
FFS_StatusInvalidTarget	Failure
FFS_StatusFileStillOpen	Failure

2nd case study — Verified FSystem (VFS)



VFS in Alloy (simplified)

The system:

```
sig System {
  fileStore: Path -> lone File,
  table: FileHandle -> lone OpenFileInfo
}
```

Paths:

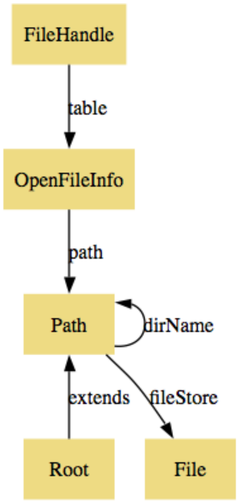
```
sig Path {
  dirName: one Path
}
```

The root is a path:

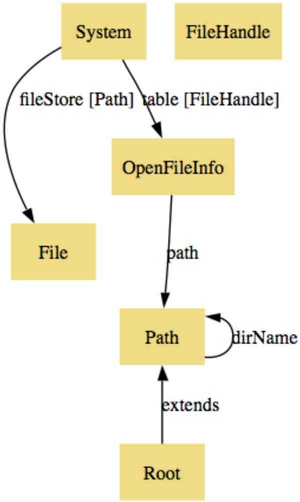
```
one sig Root extends Path {
}
```

Alloy diagrams for FSystem

Simplified:



Complete:



Binary relation semantics

Meaning of signatures:

```
sig Path {
  dirName: one Path
}
```

declares function $\mathit{Path} \xrightarrow{\mathit{dirName}} \mathit{Path}$.

```
sig System {
  fileStore: Path -> lone File,
}
```

declares simple relation $\mathit{System} \times \mathit{Path} \xrightharpoonup{\mathit{fileStore}} \mathit{File}$.
(**NB**: We often use harpoon arrows $\rightharpoonup$ for singling out simple relations.)

Binary relation semantics

- Since (as we have seen)

$$(A \times B) \rightharpoonup C \quad \cong \quad (B \rightharpoonup C)^A$$

$fileStore$ can be alternatively regarded as a function in $(Path \rightharpoonup File)^{System}$, that is, for $s : System$,

$$Path \xrightarrow{s.fileStore} File$$

- Thus the "navigation-styled" notation of **Alloy**: $p.(s.fileStore)$ means the file accessible from path p in file system s.
- Similarly, line `table: FileHandle -> lone OpenFileInfo` in the model declares

$$FileHandle \xrightarrow{s.table} OpenFileInfo$$

From Alloy to relational diagrams

We draw

$$
\begin{tikzcd}
\color{red}{OpenFileDescriptor} \arrow[d, "path"'] & & \color{red}{FileHandle} \arrow[ll, "\color{red}{table\ s}"'] \\
\color{red}{Path} \arrow[rr, "\color{red}{fileStore\ s}"'] & & \color{red}{File} \\
Path \arrow[u, "dirName"] & &
\end{tikzcd}
$$

where

- *table s* and *fileStore s* are simple relations
- the other arrows depict functions

(Diagram to be completed soon.)

Model constraints

Referential integrity:

Non-existing files cannot be opened:

```
pred ri[s: System]{
    all h: FileHandle, o: h.(s.table) |
                  some (o.path).(s.fileStore)
}
```

Paths closure:

Mother directories exist and are indeed directories:

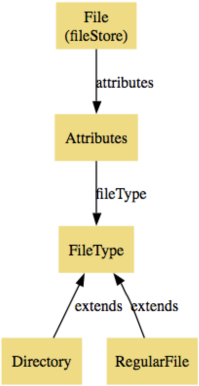
```
pred pc[s: System]{
  all p: Path |
        some p.(s.fileStore) =>
        (some d: (p.dirName).(s.fileStore) |
                         d.fileType=Directory)
}
```

2nd part of Alloy FSystem model

```
sig File {
    attributes: one Attributes
}

sig Attributes{
    fileType: one FileType
}

abstract sig FileType {}
one sig RegularFile extends FileType {}
one sig Directory extends FileType {}
```



Updated binary relational diagram

$$
\begin{tikzcd}
\mathit{OpenFileDescriptor} \arrow[d, "\mathit{path}"'] & \mathit{FileHandle} \arrow[l, "\mathit{table}\ s"'] & \\
\mathit{Path} \arrow[r, "\mathit{fileStore}\ s"'] & \mathit{File} \arrow[r, "\mathit{attributes}"] & \mathit{Attributes} \arrow[d, "\mathit{fileType}"] \\
\mathit{Path} \arrow[u, "\mathit{dirName}"] & & \mathit{FileType}
\end{tikzcd}
$$

where

- $\mathit{table}\ s$, $\mathit{fileStore}\ s$ are simple relations
- all the other arrows depict functions

Constraints: still missing

Updating diagram with constraints

Complete diagram, where $\underline{\mathit{Directory}}$ is the "everywhere-*Directory*" constant function:

$$
\begin{tikzcd}
\mathit{OpenFileDescriptor} \arrow[d, "\mathit{path}"'] & & \mathit{FileHandle} \arrow[ll, "\mathit{table}\ \ s"'] \arrow[d, "\top"] \arrow[dll, phantom, "\subseteq"] & \\
\mathit{Path} \arrow[rr, shift right, "\mathit{fileStore}\ \ s"'] & & \mathit{File} \arrow[ll, shift right, "(\mathit{fileStore}\ \ s)^{\circ}"'] \arrow[r, "\mathit{attributes}"] \arrow[d, phantom, "\cup\!|"] & \mathit{Attributes} \arrow[d, "\mathit{fileType}"] \\
\mathit{Path} \arrow[u, "\mathit{dirName}"] \arrow[rr, "\mathit{fileStore}\ \ s"'] & & \mathit{File} \arrow[r, "\underline{\mathit{Directory}}"'] & \mathit{FileType}
\end{tikzcd}
$$

Constraints:

- Top rectangle is the PF-transform of *ri* (referential integrity)
- Bottom rectangle is the PF-transform of *pc* (path closure)

Constraints in symbols

Referential integrity:

$$ri(s) \quad \triangleq \quad path \cdot (table\ s) \subseteq (fileStore\ s)^{\circ} \cdot \top \qquad (178)$$

which is equivalent to

$$ri(s) \quad \triangleq \quad \rho\,(path \cdot (table\ s)) \subseteq \delta\,(fileStore\ s)$$

since $\rho\,R = \delta\,R^{\circ}$. PF version (178) nicely encodes into **Alloy** syntax

```
pred riPF[s: System]{
    s.table.path in (FileHandle->File).~(s.fileStore)
}
```

thanks to its emphasis on **composition**.

Constraints in symbols

Paths closure:

$$pc\ N \triangleq \underline{Directory} \cdot N \subseteq fileType \cdot attributes \cdot N \cdot dirName \quad (179)$$

where N abbreviates $s.fileStore$, recall

$$\begin{tikzcd}
Path \arrow[rr, "N"'] & & File \arrow[r, "attributes"] & Attributes \arrow[d, "fileType"] \\
Path \arrow[u, "dirName"] \arrow[rr, "N"'] & & File \arrow[u, phantom, "\rotatebox{90}{$\subseteq$}"] \arrow[r, "\underline{Directory}"'] & FileType
\end{tikzcd}$$

Again thanks to emphasis on **composition**, this is easily encoded in PF-Alloy:

```
pred pcPF[s: System]{
   s.fileStore.(File->Directory) in
       dirName.(s.fileStore).attributes.fileType
}
```

Monotonicity analysis

From

$$ri(s) \quad \triangleq \quad path \cdot (table\ s) \subseteq (fileStore\ s)^{\circ} \cdot \top$$

one infers:

- *table s* can always go smaller (eg. by closing files)
- *fileStore s* can always go larger (eg. by creating new files)

On the other hand ($N = fileStore\ s$),

$$pc\ N \triangleq \underline{Directory} \cdot N \subseteq fileType \cdot attributes \cdot N \cdot dirName$$

calls for contracts (in general).

Exercise

Exercise 74: Consider the following examples of file system operations:

- **edit** an existing file without changing its attributes
- **open** a file for editing
- **create** a file in the current directory
- **rename** an existing file system object (file or directory)

Tell which operations call for contracts with respect to the two invariants *ri* and *pc*.

□

VFS CRUD and its contracts

Example: Consider the operation which removes file system objects, as modeled in Alloy:

```
pred delete[s',s: System, sp: set Path]{
        s'.table = s.table
        s'.fileStore = (univ-sp) <: s.fileStore
}
```

that is,

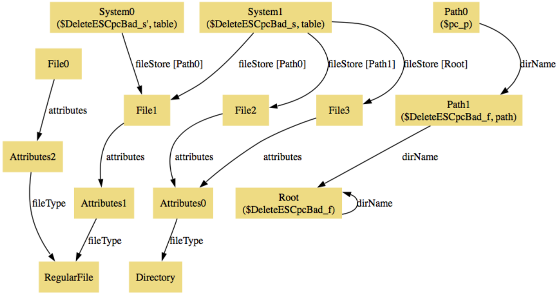
$$delete\ sp\ (M, N) \quad \triangleq \quad (M, N \cdot \Phi_{(\notin sp)}) \tag{180}$$

where M (resp. N) abbreviates $s.table$ (resp. $s.fileStore$) and $\Phi_{(\notin sp)}$ is the coreflexive associated to the complement of sp.

Intuitively

Intuitively, *delete* will put at risk

- the *ri* constraint once we decide to delete file system objects which are open;
- the *pc* constraint once we decide to delete directories with children.



(Model-checking in **Alloy** will easily spot these flaws, as checked above by a counter-example for the latter situation.)

Calculation

We want to calculate the weakest pre-condition (contract) for each constraint to be maintained.

For this we will recall the following properties of relational algebra: shunting (54),

$$h \cdot R \subseteq S \;\Leftrightarrow\; R \subseteq h^{\circ} \cdot S$$

pre-restriction (117),

$$R \cdot \Phi \;=\; R \cap \top \cdot \Phi$$

and

$$f \cdot R \subseteq \top \cdot S \;\Leftrightarrow\; R \subseteq \top \cdot S \tag{181}$$

Exercise 75: Prove (181). Can this equivalence be generalized?

□

Contract calculation – *ri*

$$
\begin{array}{ll}
 & ri(M, N \cdot \Phi_{(\notin S)}) \\
\Leftrightarrow & \qquad \{ \text{ (178) } \} \\
 & path \cdot M \subseteq (N \cdot \Phi_{(\notin S)})^{\circ} \cdot \top \\
\Leftrightarrow & \qquad \{ \text{ converses (45,111) } \} \\
 & path \cdot M \subseteq \Phi_{(\notin S)} \cdot N^{\circ} \cdot \top \\
\Leftrightarrow & \qquad \{ \text{ (118) } \} \\
 & path \cdot M \subseteq N^{\circ} \cdot \top \cap \Phi_{(\notin S)} \cdot \top \\
\Leftrightarrow & \qquad \{ \text{ $\cap$-universal (64) } \} \\
 & path \cdot M \subseteq N^{\circ} \cdot \top \wedge path \cdot M \subseteq \Phi_{(\notin S)} \cdot \top \\
\Leftrightarrow & \qquad \{ \text{ (178) ; shunting (54) } \} \\
 & ri(M, N) \ \wedge \ \underbrace{M \subseteq path^{\circ} \cdot \Phi_{(\notin S)} \cdot \top}_{wp}
\end{array}
$$

Contract calculation – *ri*

The obtained weakest pre-condition *wp* converts back to the pointwise

$$\langle \forall\ b\ :\ b \in rng\ M :\ path\ b \notin S \rangle$$

which instantiates to

$$\langle \forall\ b\ :\ b \in rng\ M :\ path\ b \neq p \rangle$$

for $S := \{p\}$. We are done as far invariant *ri* is concerned.

Exercise 76: Encode the calculated contract (weakest pre-condition) in Alloy.

□

Contract calculation – *pc*

For improved readability, we introduce abbreviations $ft := fileType \cdot attributes$ and $d := \underline{Directory}$ in

$$
\begin{array}{ll}
 & pc(delete\ S\ (M, N)) \\
\Leftrightarrow & \quad \{ \text{ (180) and (179) } \} \\
 & d \cdot (N \cdot \Phi_{(\notin S)}) \subseteq ft \cdot (N \cdot \Phi_{(\notin S)}) \cdot dirName \\
\Leftrightarrow & \quad \{ \text{ shunting (54) } \} \\
 & d \cdot N \cdot \Phi_{(\notin S)} \cdot dirName^{\circ} \subseteq ft \cdot N \cdot \Phi_{(\notin S)} \\
\Leftrightarrow & \quad \{ \text{ (117) } \} \\
 & d \cdot N \cdot \Phi_{(\notin S)} \cdot dirName^{\circ} \subseteq ft \cdot N \cap \top \cdot \Phi_{(\notin S)} \\
\Leftrightarrow & \quad \{ \cap\text{-universal ; shunting } \}
\end{array}
$$

Contract calculation – *pc*

$$\left\{\begin{array}{l} d \cdot N \cdot \Phi_{(\notin S)} \subseteq ft \cdot N \cdot dirName \\ d \cdot N \cdot \Phi_{(\notin S)} \subseteq \top \cdot \Phi_{(\notin S)} \cdot dirName \end{array}\right.$$

$$\Leftrightarrow \qquad \{ \ \top \text{ absorbs } d \text{ (181)} \ \}$$

$$\left\{\begin{array}{l} \underbrace{d \cdot N \cdot \Phi_{(\notin S)} \subseteq ft \cdot N \cdot dirName}_{\text{weaker than } pc(N)} \\ \underbrace{N \cdot \Phi_{(\notin S)} \subseteq \top \cdot \Phi_{(\notin S)} \cdot dirName}_{wp} \end{array}\right.$$

Back to points, *wp* is:

$$\langle \forall\ q\ :\ q \in dom\ N \wedge q \notin S :\ dirName\ q \notin S \rangle$$

$$\Leftrightarrow \qquad \{ \ \text{predicate logic} \ \}$$

$$\langle \forall\ q\ :\ q \in dom\ N \wedge (dirName\ q) \in S :\ q \in S \rangle$$

Ensuring paths closure

In words:

> *if the parent directory of existing path q is marked for deletion than so must be q.*

Translating the calculated contract back to Alloy:

```
pred pre_delete[s: System, sp: set Path]{
     all q: Path |
        (some q.(s.fileStore) &&
              q.dirName in sp) => q in sp
}
```

Exercises

Exercise 77: Recalling exercise 74, calculate the contract required by the operation

$$open\ K\ (M, N) \triangleq (M \cup K, N)$$

□

Exercise 78: Specify the POSIX `mkdir` operation and calculate its contract.

□

R. Bird and O. de Moor.
Algebra of Programming.
Series in Computer Science. Prentice-Hall International, 1997.

D. Jackson.
Software Abstractions: Logic, Language, and Analysis.
The MIT Press, Cambridge Mass., 2012.
Revised edition, ISBN 0-262-01715-2.

C.B. Jones.
Systematic Software Development Using VDM.
Series in Computer Science. Prentice-Hall International, 1986.

R. Joshi and G.J. Holzmann.
A mini challenge: build a verifiable filesystem.
Formal Asp. Comput., 19(2):269–272, 2007.

J.N. Oliveira and M.A. Ferreira.
Alloy meets the algebra of programming: a case study, 2012.
To appear in IEEE Transactions on Software Engineering.

J.M. Spivey.
The Z Notation — A Reference Manual.
Series in Computer Science. Prentice-Hall International, 1989.
C.A.R. Hoare (series editor).