Translating Alloy Specifications to the Point-free Style

Nuno Macedo

Departamento de Informática
Universidade do Minho
Braga, Portugal

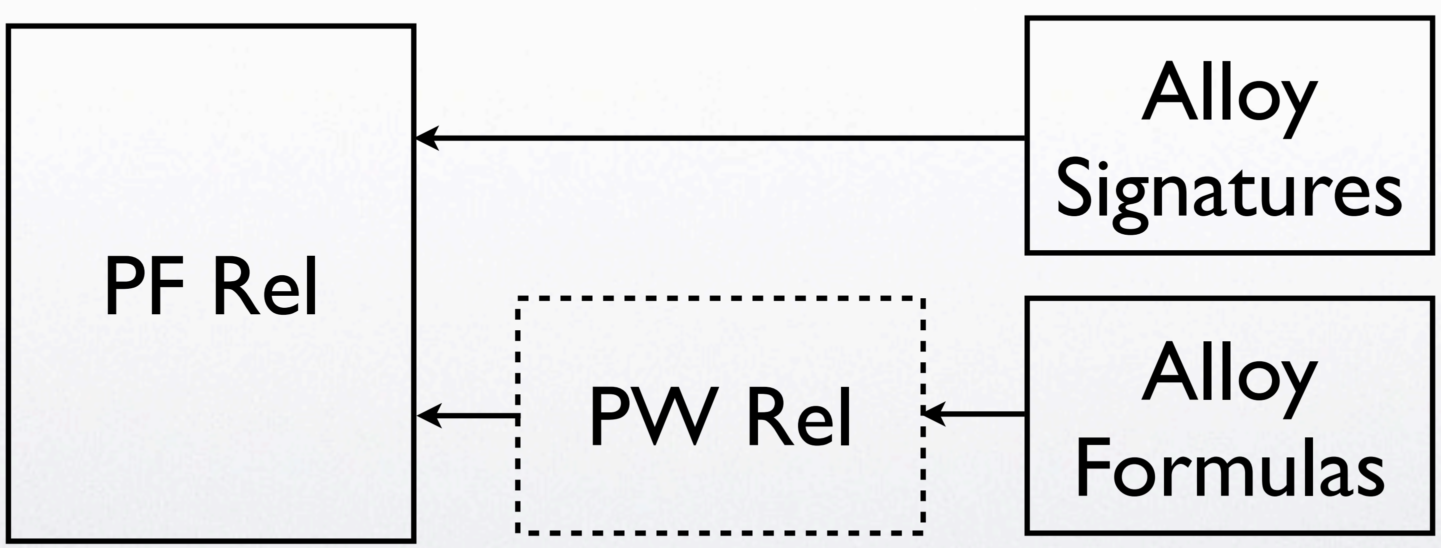
June 22nd, 2011

Motivation

- *Alloy* provides a tool for automatic *bounded* verification (the *Alloy Analyzer*);
- Sometimes however, *unbounded* verification is necessary;
- Alloy's logic is a *relational*, so relational frameworks are natural choices;
- The *point-free* (PF) style provides simple enough formulas for manipulation and analysis.

Objectives

- A complete translation of Alloy models to a PF relational framework is proposed.



Alloy

- State-based modeling language;
- Simple language, based on simple mathematical notations;
- Characteristics of object modeling;
- Automatic bounded verification.

Calculus of Relations

- *Relational Logic*:
  - First-order logic (FOL) enhanced with relational operators (composition, meet, join,...)

$$\langle \forall a, b :: a \, R \, b \Rightarrow a \, S \, b \rangle$$

- *Relation Algebras (RAs):*
  - FOL without variables

$$R \subseteq S$$

  - Equivalent to FOL with 3 quantified variables.

PF Relational Logic

- Fork Algebras (FAs) were created to overcome the lack of expressiveness of RAs;
- Introduces *pairs* and a new operator *fork:*

$$c\,R\,a \wedge b\,S\,a \equiv (c,b)\,\langle R,S\rangle\,a$$

- Equivalent to FOL;
- Can be seen as an untyped version of the categorical relational calculus commonly used.

N-ary Relations

- Alloy allows relations of any arity;
- Unary relations are represented by correflexives (fragments of the identity);
- N-ary relations are "uncurried" to binary relations with the domain as a tuple:

$$A \to B \to C \rightsquigarrow A \times B \to C$$

N-ary Operators

- New operators to manipulate n-ary relations:
  - N-ary composition:

$$(a,b)\, R \bullet S\, c \equiv \langle \exists k :: a\, R\, k \wedge (k,b)\, S\, c \rangle$$

  - Rotate:

$$(a,b)\, \overrightarrow{R}\, c \equiv (c,a)\, R\, b$$

- When dealing with binary relations, they collapse to binary composition and converse;
- Possess some interesting properties, similar to their binary counterparts.

Formula Translation

- Marcelo Frias, Carlos Pombo and Nazareno Aguirre, *An equational calculus for Alloy*;
- Automatically translates (only) Alloy *formulas* to FA;
- Resulting formulas are extremely complex:

$$\textbf{all}\ \mathrm{a} : \mathrm{A}\ |\ \textbf{some}\ \mathrm{c} : \mathrm{C}\ |\ \mathrm{c}\ \textbf{in}\ \mathrm{r{\cdot}a}$$

$$\downarrow$$

$$\overline{\overline{\overline{\overline{\top \cdot \rho(\langle id, \pi_2 \cdot \phi \rangle)} \cdot \langle \pi_1 \cdot \pi_1, \pi_2 \cdot \pi_1, \pi_2, \top \rangle \cdot \langle \pi_1, \pi_2, \top \rangle} \cdot \langle id, \top \rangle} \cdot \top} = \top$$

$$\phi = id \times \top \cap \overline{\delta(\pi_2 \cdot \pi_1 \cap \pi_2) \cap \overline{\rho(id \times R \cdot \delta(\pi_2 \cdot \pi_1 \cap \pi_2))} \cap id \times \top}$$

Formula Translation

- Operations on n-ary relations can not be directly translated to FA in an efficient way;
- Formulas will be translated in two steps:
  - Alloy to FOL: fully expands Alloy formulas to their PW definition;
  - FOL to FA: mechanic PW to PF translation, enhanced with heuristics.

Formula Translation

- The main idea of the default translation is to "push" all variables to a single tuple, e.g.:

$$y\,R\,x_i \rightsquigarrow y\,(R \cdot \Pi_i^n)\,(x_1, \ldots, x_n)$$

- Which can be automatically removed in the end:

$$y\,R\,(x_1, \ldots, x_n) \rightsquigarrow y\,R \cdot \langle id, \top \rangle\,(x_1, \ldots, x_{n-1})$$

- However, by further enhancing the translation with heuristic rules, we obtained extremely simple formulas.

Formula Translation

all a,b : A | (**some** c : C | c = r·a && c = r·b) ⇒ a = b

$$\langle \forall a, b \in A :: \langle \exists c \in C :: a\,R\,c \wedge b\,R\,c \rangle \Rightarrow a = b \rangle$$

$$\top \subseteq \overline{((\top \cdot (\pi_1 \cdot \pi_2 \cap R \cdot \pi_2) \cap \top \cdot (\pi_2 \cdot \pi_2 \cap R \cdot \pi_2)) \cdot id \nabla \top \cap \overline{\top \cdot (\pi_1 \cap \pi_2)}) \cdot id \nabla \top \cdot \top}$$

$$r \cdot r^{\circ} \subseteq id$$

Type System

- Alloy's type system is very loose, allowing the combination of any types (with some restrictions on the arities);
- By encoding them as correflexives, we are able to easily define the hierarchy of signatures and check if types of expressions match;
- A binary relation of type $R :: A \to B$ induces the fact

$$R \subseteq \Phi_A \cdot \top \cdot \Phi_B$$

Signature Translation

- Using the same technique, we are able to encode the multiplicities defined in Alloy signatures:

Signatures

Sig A	Property
`set`	$true$
`lone`	$\Phi_A \cdot \top \cdot \Phi_A \subseteq id$
`some`	$\top \subseteq \top \cdot \Phi_A \cdot \top$
`one`	$\Phi_A \cdot \top \cdot \Phi_A \subseteq id \wedge \top \subseteq \top \cdot \Phi_A \cdot \top$

Relations

ith field	Property
`set`	$true$
`lone`	$\overset{(\lvert R\rvert-i)\rightarrow\circ}{R} \cdot \overset{(\lvert R\rvert-i)\rightarrow}{R} \subseteq id$
`some`	$id \subseteq \overset{(\lvert R\rvert-i)\rightarrow}{R} \cdot \overset{(\lvert R\rvert-i)\rightarrow\circ}{R}$
`one`	$\overset{(\lvert R\rvert-i)\rightarrow\circ}{R} \cdot \overset{(\lvert R\rvert-i)\rightarrow}{R} \subseteq id \wedge id \subseteq \overset{(\lvert R\rvert-i)\rightarrow}{R} \cdot \overset{(\lvert R\rvert-i)\rightarrow\circ}{R}$

- When dealing with binary relations, collapses to the typical taxonomy (total, injective, surjective...).

Example

Alloy model

```
abstract sig Person {}
sig Student, Professor extends Person {}
sig Course {
  lecturer : some Professor,
  depends : set Course
}
sig University {
  enrolled : set Student,
  courses  : Student -> Course
}
pred inv[u : University] {
  (u.courses).Course in u.enrolled
  all s : Student |
     (s.(u.courses)).*depends in s.(u.courses)
}
pred enroll[u, u' : University, s : Student] {
  u'.enrolled = u.enrolled + s
  u'.courses = u.courses
}
assert {
  all u,u':University,s:Student |
     inv[u] and enroll[u,u',s] => inv[u']
}
```

FA model

Signature facts

$id = \Phi_{Person} \cup \Phi_{Course} \cup \Phi_{University}$

$\Phi_{Student} \cup \Phi_{Professor} \subseteq \Phi_{Person} \wedge \Phi_{Student} \cap \Phi_{Professor} = \perp$

$lecturer \subseteq \Phi_{Course} \cdot \top \cdot \Phi_{Professor}$

$enrolled \subseteq \Phi_{University} \cdot \top \cdot \Phi_{Student}$

$courses \subseteq \Phi_{University} \cdot \top \cdot \Phi_{Student} \times \Phi_{Course}$

$depends \subseteq \Phi_{Course} \cdot \top \cdot \Phi_{Course}$

$id \subseteq lecturer \cdot lecturer^\circ$

Assertion

$$(\Phi_U \times \Phi_U \times \Phi_S) \cap c_1/(e_1 \cdot \pi_1) \cap (c_1 \cdot (\Phi_S \times d^{*\circ}))/c_1$$
$$\cap$$
$$c_1/c_2 \cap c_2/c_1 \cap e_1/e_2 \cap e_2 \cdot \pi_2 \cdot \pi_2 \cap e_2/(e_1 \cup id_3)$$
$$\subseteq$$
$$c_2/(e_2 \cdot \pi_1) \cap (c_2 \cdot (\Phi_S \times d^{*\circ}))/c_2$$

Conclusions

- Complete and automatic translation of Alloy models;
- Due to the simplicity, it is suitable for manual verification;
- Automatic verification is also possible, e. g., Prover9 automatically verified the previous example;
- Complexity increases with the number of n-ary relations, which are common in Alloy.

Translating Alloy Specifications to the Point-free Style

Nuno Macedo

Departamento de Informática
Universidade do Minho
Braga, Portugal

June 22nd, 2011