Relational Reasoning via SMT Solving

Aboubakr Achraf El Ghazi
Mana Taghdiri

Karlsruhe Institute of Technology, Germany

FM 2011, Limerick, Ireland

June 22, 2011

Motivation

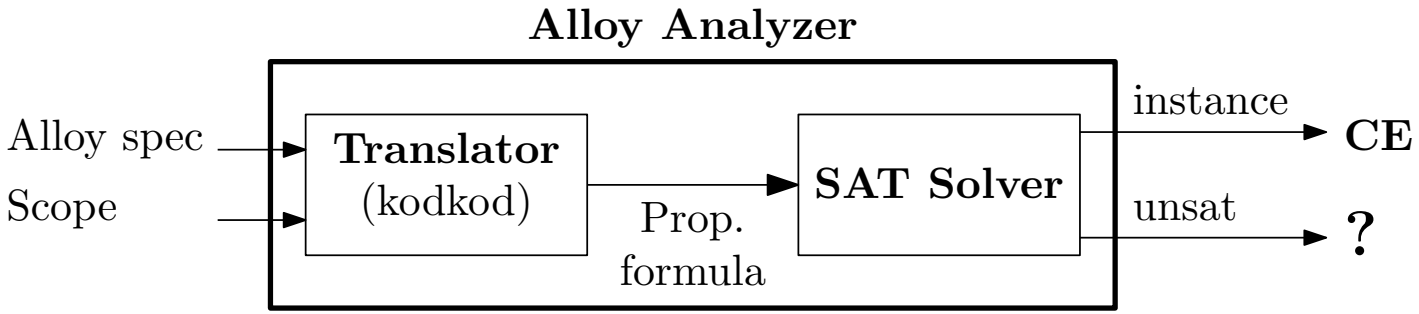
- Alloy is a widely-used modeling language
  - Stand-alone framework for checking high-level system designs
    - Network protocols
    - File system policies
    - Schedulers
  - Intermediate language for checking programs
    - Functional properties of Java programs
    - Test case generation
    - Specification extraction
  - Engine for generating counterexamples
    - For theorem provers
- Taught in about 30 universities
- Used in industry (AT&T, Telcordia, etc.)

The Alloy language

- Declarative modeling language
- Simple, uniform semantics
  - Everything is a relation
  - Semantics equivalent to first-order relational logic
- Expressive, familiar syntax
  - Set and relational operators, first-order quantifiers, transitive closure, linear integer arithmetic
  - Concise formulation of rich properties
  - Syntax similar to an OO language
- Analyzable
  - The Alloy Analyzer is fully automatic
  - Looks for an instance violating a given assertion

The Alloy Analyzer (AA)

- Performs bounded analysis
  - Requires a user-provided scope
  - Reduction to a satisfiability problem (SAT)
  - Enables automation



- Shortcomings
  - Can never prove an assertion correct, even for the simplest models
  - Limited support for numerical expressions

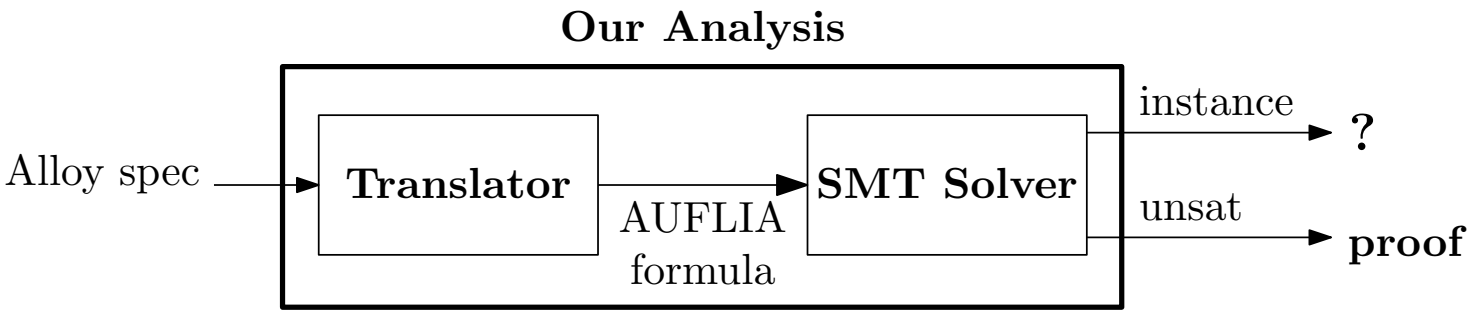
▷ Need for a fully automatic, proof-capable engine!

Approach

- Analyze Alloy models using an SMT solver
- Satisfiability modulo theories (AUFLIA logic)
  - Arrays
  - Uninterpreted functions with equality
  - Linear integer arithmetic
  - First-order quantifiers
    - Quantifiers, in general, make the logic undecidable
    - Recent SMT solvers handle quantifiers much better
    - In the AUFLIA-Division of SMTComp 2010, Z3 solved 194 out of 206 benchmarks
- Benefits
  - No type finitization $\Rightarrow$ Can prove valid assertions
  - Fully automatic
  - Full support of linear integer arithmetic

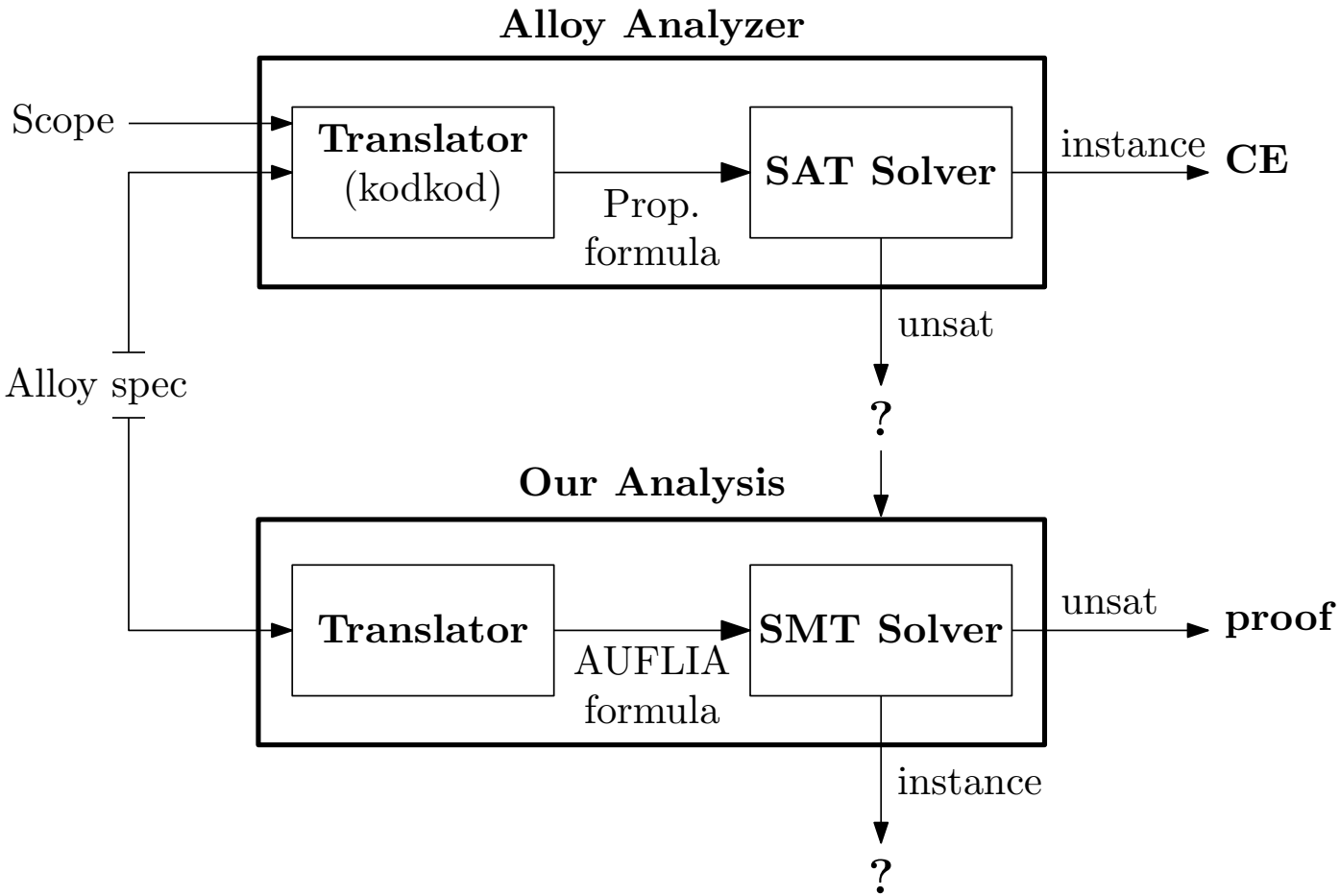
Approach

- Each Alloy construct is converted using a first-order SMT axiom
  - Type system
  - Relational operations
  - Transitive closure
  - Set cardinality
- Our target logic is undecidable



Approach

- Our engine complements the Alloy Analyzer



Example – a simple file system

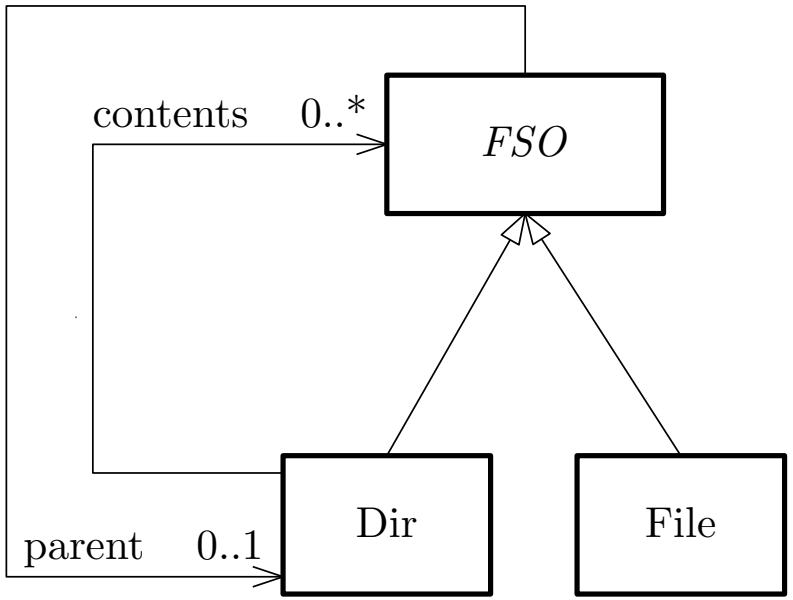
Alloy Model:

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
fact {
 contents = ~parent
 all d:Dir| not(d in d.^contents)
 all d:Dir| #(d.contents) <= 5
}
assert oneLocation {
 all o:FSO, lone d:FSO|
  o in d.contents
}
check oneLocation for 8
```

Example – declarations

Alloy Model:

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
fact {
 contents = ~parent
 all d:Dir| not(d in d.^contents)
 all d:Dir| #(d.contents) <= 5
}
assert oneLocation {
 all o:FSO, lone d:FSO|
  o in d.contents
}
check oneLocation for 8
```



Example – type hierarchy

- A sort for each top-level Alloy type
- A membership function for each Alloy type
- An axiom to enforce subtyping

Alloy Model:

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
```

Z3 Axioms:

```
(declare-sort FSO)

(declare-fun isFSO (FSO) Bool)
(declare-fun isDir (FSO) Bool)
(declare-fun isFile (FSO) Bool)

(assert (forall (f FSO)
 (=> (isDir f) (isFSO f))))
(assert (forall (f FSO)
 (=> (isFile f) (isFSO f))))
```

Example – type hierarchy

- Abstract types are the union of their subtypes
- Extension types are disjoint

Alloy Model:

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
```

Z3 Axioms:

```
(assert (forall (f FSO)
 (=> (isFSO f)
  (or (isDir f)(isFile f)))))

(assert (forall (f FSO)
 (not
  (and (isDir f)(isFile f)))))
```

Example – relations

- Membership function and type-enforcing axioms for each relation
- Uninterpreted functions to enforce multiplicity keywords

Alloy Model:

parent: FSO -> lone Dir

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
```

Z3 Axioms:

parent: FSO -> FSO -> Bool

```
(declare-fun parent (FSO FSO) Bool)
(declare-fun contents (FSO FSO) Bool)
(assert (forall (f FSO)(g FSO)
 (=> (parent f g)
     (and (isFSO f)(isDir g)))))
(assert (forall (f FSO)(g FSO)
 (=> (contents f g)
     (and (isDir f)(isFSO f)))))
(declare-fun oneParent (FSO) FSO)
(assert (forall (f FSO)(g FSO)
 (=> (parent f g)
     (= g (oneParent f)))))
```

Example – relations

- Membership function and type-enforcing axioms for each relation
- Uninterpreted functions to enforce multiplicity keywords

Alloy Model:

parent: FSO -> lone Dir

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
```

Z3 Axioms:

parent: FSO -> FSO -> Bool

```
(declare-fun parent (FSO FSO) Bool)
(declare-fun contents (FSO FSO) Bool)
(assert (forall (f FSO)(g FSO)
 (=> (parent f g)
     (and (isFSO f)(isDir g)))))
(assert (forall (f FSO)(g FSO)
 (=> (contents f g)
     (and (isDir f)(isFSO f)))))
(declare-fun oneParent (FSO) FSO)
(assert (forall (f FSO)(g FSO)
 (=> (parent f g)
     (= g (oneParent f)))))
```

Example – relations

- Membership function and type-enforcing axioms for each relation
- Uninterpreted functions to enforce multiplicity keywords

Alloy Model:

```
parent: FSO -> lone Dir
```

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
```

Z3 Axioms:

```
parent: FSO -> FSO -> Bool
```

```
(declare-fun parent (FSO FSO) Bool)
(declare-fun contents (FSO FSO) Bool)
(assert (forall (f FSO)(g FSO)
 (=> (parent f g)
     (and (isFSO f)(isDir g)))))
(assert (forall (f FSO)(g FSO)
 (=> (contents f g)
     (and (isDir f)(isFSO f)))))
(declare-fun oneParent (FSO) FSO)
(assert (forall (f FSO)(g FSO)
 (=> (parent f g)
     (= g (oneParent f)))))
```

Example – facts

- Low-level axiomatization based on membership functions

```
(contents in ~parent) and
(~parent in contents)
```

Alloy Model:

```
fact {
 contents = ~parent
 all d:Dir|not(d in d.^contents)
 all d:Dir| #(d.contents) <= 5
}
```

Z3 Axioms:

```
(assert (and
 (forall (x FSO)(y FSO)
  (=> (contents x y)
      (parent y x)))
 (forall (x FSO)(y FSO)
  (=> (parent x y)
      (contents y x)))
))
```

Example – transitive closure

- Integer based iterative computation

$r : T \to T,\ n = size(T)$

$\hat{}\texttt{r} = \texttt{r} \cup \texttt{r.r} \cup \ldots \cup r^{(n)}$

Alloy Model:

```
fact {
 contents = ~parent
 all d:Dir|
   not(d in d.^contents)
 all d:Dir|
   #(d.contents) <= 5
}
```

$\texttt{itrR(i)} = \texttt{r} \cup \texttt{r.r} \cup \ldots \cup r^{(i)}$

$\texttt{tr} = \texttt{r} \cup \texttt{r.r} \cup \ldots$

Z3 Axioms:

$$itrR : Nat \times FSO \times FSO \to Bool$$
$$tr : FSO \times FSO \to Bool$$

$$\forall x, y : FSO, itrR(1, x, y) = r(x, y)$$
$$\forall i : Int, x, y : FSO,$$
$$(i > 1) \implies itrR(i, x, y) = (itrR(i-1, x, y) \vee$$
$$\exists z : FSO, itrR(i-1, x, z) \wedge r(z, y))$$

$$\forall x, y : FSO, tr(x, y) = (\exists i : Int, itrR(i, x, y))$$

Example – Integer expressions

```
fact {
 contents = ~parent
 all d:Dir| not(d in d.^contents)
 all d:Dir| #(d.contents) <= 5
}
```

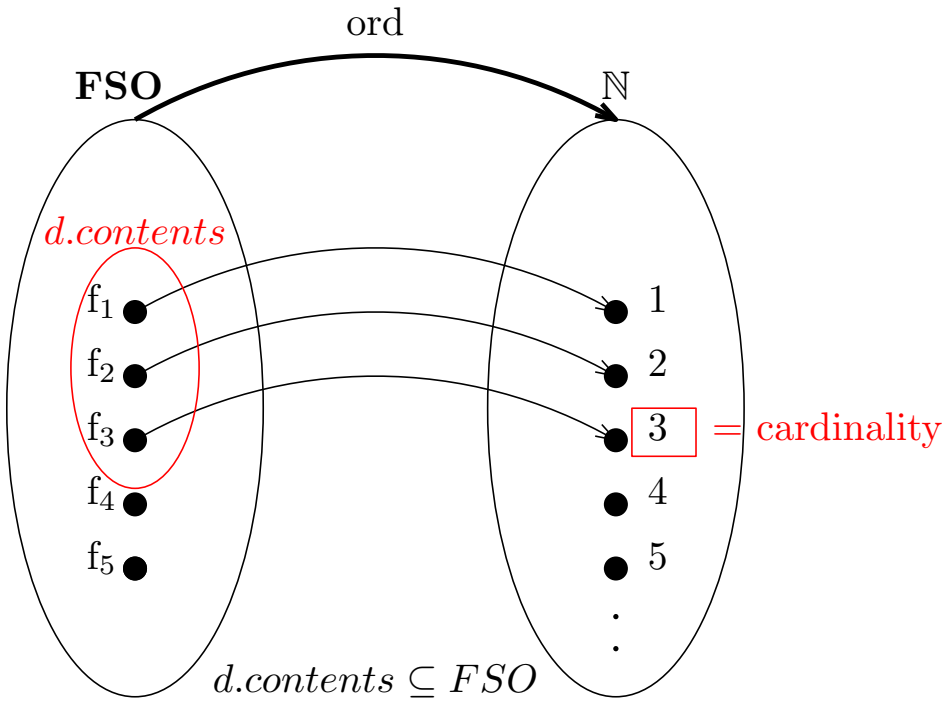
- Integer type in Alloy mapped to the integer type in SMT
- We use unbounded integer semantics – *deviation from Alloy*
  - Alloy integer arithmetic performed wrt a given bitwidth
  - $2 + 2 > 2$ does not hold in Alloy wrt a bitwidth of 3
- Challenging if Alloy's cardinality and comprehension expressions are involved
  - `(sum d: Dir| #d.contents) = 7`

Example – cardinality

d.contents $\subseteq$ *FSO* and contains a free variable *d*

Alloy Model:

```
fact {
 contents = ~parent
 all d:Dir|
   not(d in d.^contents)
 all d:Dir|
   #(d.contents) <= 5
}
```



Example – cardinality

- (Triggered) SMT axiomatization for cardinality

#FSO

Alloy Model:

```
fact {
 contents = ~parent
 all d:Dir|
   not(d in d.^contents)
 all d:Dir|
   #(d.contents) <= 5
}
```

Z3 Axioms:

$$crd : Nat$$
$$ord : FSO \rightarrow Nat$$
$$inv : Nat \rightarrow FSO$$

$$crd = 0 \Leftrightarrow \forall f : FSO, \neg isFSO(f) \quad (1)$$
$$inv = ord^{-1} \quad (2)$$
$$\forall f : FSO \mid isFSO(f) \Rightarrow 1 \leq ord(f) \leq crd \quad (3)$$
$$\forall i : Nat \mid 1 \leq i \leq crd \Rightarrow isFSO(inv(i)) \quad (4)$$

- Patterns used for Axiom 4

Example – cardinality

- (Triggered) SMT axiomatization for cardinality

$d.contents \subseteq FSO$ and contains a free variable d

Alloy Model:

```
fact {
 contents = ~parent
 all d:Dir|
   not(d in d.^contents)
 all d:Dir|
   #(d.contents) <= 5
}
```

Z3 Axioms:

$exp = d.contents$

$$
\begin{aligned}
&crd : Dir \rightarrow Nat \\
&ord : Dir \rightarrow FSO \rightarrow Nat \\
&inv : Dir \rightarrow Nat \rightarrow FSO \\
&\forall d : Dir \mid crd(d) = 0 \Leftrightarrow exp = \emptyset && (1) \\
&inv = ord^{-1} && (2) \\
&\forall d : Dir \mid \forall f : FSO \mid \\
&\quad f \in exp \Rightarrow 1 \leq ord(d,f) \leq crd(d) && (3) \\
&\forall d : Dir \mid \forall i : Nat \mid \\
&\quad 1 \leq i \leq crd(d) \Rightarrow inv(d,i) \in exp && (4)
\end{aligned}
$$

- Patterns used for Axiom 4

Example – assertions

- Negation of the assertion is conjoined with the other formulas

Alloy Model:

```
assert oneLocation {
 all o:FSO, lone d:FSO|
  o in d.contents
}
```

Z3 Axioms:

```
(assert (not (forall (o FSO)
 (forall (d1 FSO)(d2 FSO)
  (=> (and (contents d1 o)
           (contents d2 o))
      (= d1 d2))
))))
```

Example – proof

Alloy Model:

```
abstract sig FSO {
 parent: lone Dir
}
sig Dir extends FSO {
 contents: set FSO
}
sig File extends FSO {}
fact {
 contents = ~parent
 all d:Dir| not(d in d.^contents)
 all d:Dir| #(d.contents) <= 5
}
assert oneLocation {
 all o:FSO, lone d:FSO|
  o in d.contents
}
check oneLocation for 8
```

- No type finitization
- The SMT formula looks like $F_1 \wedge F_2 \wedge \cdots \wedge F_n \wedge \neg A$
- Unsatisfiability means correctness of the assertion
- The example assertion is proven correct by Z3

Evaluation

		Alloy Analyzer		Our Analysis by Z3	
Problem	Assertion	Scope	Time (sec)	Time (sec)	Result
address book	delUndoesAdd	31	80.91	0.00	proved
	addIdempotent	31	112.66	0.01	proved
COM	theorem1	14	175.46	0.00	proved
	theorem2	14	177.97	0.00	proved
	theorem3	14	168.51	0.00	proved
	theorem4a	14	174.89	0.00	proved
	theorem4b	14	166.68	0.00	proved
abstract memory	writeRead	44	179.44	0.00	proved
	writeIdempotent	29	98.67	0.03	proved
media assets	hidePreservesInv	87	86.03	0.00	proved
	pasteAffectsHidden	29	138.34	0.00	proved
mark sweep	soundness1	9	81.52	0.12	false CE
	soundness2	8	28.84	0.11	false CE
	completeness	7	32.52	0.14	false CE
nQueen	solCondition	73	173.51	0.05	proved
address book	addLocal	3	0.05	0.10	sound CE
media assets	cutPaste	3	0.19	0.06	sount CE
own grandpa	ownGrandpa	4	0.01	0.12	sound CE
nQueen	15Queens	15	4.95	13.53	sound CE
handshake	puzzle	10	2.47	time out	N/A

Related work

- Proving Alloy assertions
  - Dynamite [Frias, Pombo, Moscato, 2007] uses PVS
  - Prioni [Arkoudas, Khurshid, Marinov, Rinard, 2003] uses Athena
  - Based on interactive theorem provers
  - No support of integer or cardinality expressions
- Language constructs
  - Transitive closure [Lev-ami, Immerman, Reps, Sagiv, 2005] for program verification – based on heuristics
  - Set cardinality [Suter, Steiger, Kuncak, 2011] – a decision procedure based on integer arithmetic constraints – handles simple sets
  - comprehension expressions [Leino, Monahan, 2009] defined for an integer range – useful after computing cardinality

Conclusions

- A fully automatic engine capable of proving Alloy assertions
  - Via axiomatization of (core) Alloy in SMT-lib
  - Proof not guaranteed
  - Complements the current Alloy Analyzer
- Useful for invalid assertions too
  - Valid counterexamples produced in several experiments

- Future directions
  - Transitive closure is still a challenge – *negative context*
  - Complex cardinality expressions time out
  - Support of Alloy library: traces, sequences, etc.

Example – triggerd cardinality

- SMT instantiation is grounded term based
- Need to enforce the instantiation of axioms for all $1 \leq i \leq crd(d)$
  - Sufficient to be done for Axiom 4 only

$$\forall d : Dir \mid \forall i : Nat \mid 1 \leq i \leq crd(d) \Rightarrow inv(d, i) \in exp$$

- A trigger schema

$trg : Dir \times Int \rightarrow Bool$

(**assert** (**forall** (d Dir) (=> (< 0 (crd d)) (trg d 1)))) (1)

(**assert** (**forall** (d Dir) (i **Int**)
  (=> (**and** (<= 1 i) (< i (crd d))) (trg d (+ i 1))) **:pat** {(trg d i)})) (2)

(**assert** (**forall** (d Dir) (i **Int**)
  (=> (**and** (<= 1 i) (<=i ($crdd$))) (exp (inv d i))) **:pat** {(trg d i)})) (3)