Using SMT Solvers in Finding Finite Models and Cores for Relational Logic

Ferhat Erata[1,2] Ruzica Piskac[1]

[1]Yale University, Computer Science, New Haven, CT, USA
[2]UNIT Information Technologies Ltd., Izmir, Turkey

The 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Tallinn, Estonia
26–30 August, 2019

Information Technology for European Advancement (ITEA)

ITEA-ModelWriter: Synchronized Document Engineering Platform

`https://itea3.org/project/modelwriter.html`

ITEA-ASSUME: Affordable Safe & Secure Mobility Evolution

`https://itea3.org/project/assume.html`

ITEA-XIVT: eXcellence In Variant Testing

`https://itea3.org/project/xivt.html`

ITEA3

EUREKA
innovation across borders

European Cooperation in Science and Technology (COST)

IC1404 Multi-Paradigm Modelling for Cyber-Physical Systems

`http://www.cost.eu/COST_Actions/ict/IC1404`

IC1402 Runtime Verification beyond Monitoring

`http://www.cost.eu/COST_Actions/ict/IC1402`

Outline

1. First-order Relational Logic
   - Applications of Alloy
   - Alloy Demonstration
2. Research Road-map
3. Relational Specification
   - Universe and Bounds
   - Constraints
   - Outcome
4. Evaluation

Applications of Alloy

- Access Control and Security Policies.
- Feature Modeling and Analysis
- Domain Specific Languages and Modeling.
- Testing and Automated Test Case Generation
- Software Architecture
- Configuration and Reconfiguration, Data Structure Repair
- Program verification.
- Databases.
- Model-Driven Development.
- Network Protocols
- Requirements

Front-end

Universe and Bounds

$$
\begin{aligned}
problem &::= universe\ \ relDecl^{*}\, formula^{*} \\
universe &::= \{atom^{*}\} \\
relDecl &::= relation\ :_{arity}\ [constant, constant] \\
constant &::= \{tuple^{*}\} \\
tuple &::= \langle atom^{*} \rangle \\
arity &::= positiveinteger \\
relation &::= identifier \\
atom &::= identifier
\end{aligned}
$$

$$
\begin{array}{llr}
\textit{formula} ::= & & \\
 & \textit{expr} \subset \textit{expr} & \text{(subset)} \\
 | & \textit{expr} = \textit{expr} & \text{(equality)} \\
 | & \textbf{some } \textit{expr} & \text{(at least one)} \\
 | & \textbf{one } \textit{expr} & \text{(exactly one)} \\
 | & \textbf{lone } \textit{expr} & \text{(at most one)} \\
 | & \textbf{no } \textit{expr} & \text{(empty)} \\
 | & \neg \textit{formula} & \text{(negation)} \\
 | & \textit{formula} \wedge \textit{formula} & \text{(conjuction)} \\
 | & \textit{formula} \vee \textit{formula} & \text{(disjunction)} \\
 | & \textit{formula} \Rightarrow \textit{formula} & \text{(implication)} \\
 | & \textit{formula} \Leftrightarrow \textit{formula} & \text{(biimplication)} \\
 | & (\forall \mid \exists \mid \exists! \mid \nexists)\ \textit{varDecls} \mid \textit{formula} & \text{(universal)} \\
 | & \textit{intexpr} \ \{ < \mid \leq \mid = \mid > \mid \geq \} \ \textit{intexpr} & \text{(comparison)}
\end{array}
$$

formula ::=

expr **in** *expr* (subset)

| *expr* = *expr* (equality)

| **some** *expr* (at least one)

| **one** *expr* (exactly one)

| **lone** *expr* (at most one)

| **no** *expr* (empty)

| !*formula* (negation)

| *formula* **and** *formula* (conjuction)

| *formula* **or** *formula* (disjunction)

| *formula* **implies** *formula* (implication)

| *formula* **iff** *formula* (biimplication)

| (**all** | **some** | **one** | **no**) *varDecls* | *formula* (universal)

| *intexpr* $\{ < \mid \leq \mid = \mid > \mid \geq \}$ *intexpr* (comparison)

$$
\begin{array}{llr}
\textit{expr} ::= & & \\
& \textit{var} & \text{(variable)} \\
| & \textit{expr} = \textit{expr} & \text{(equality)} \\
| & \sim \textit{expr} & \text{(transpose)} \\
| & \hat{}\,\textit{expr} & \text{(clousure)} \\
| & \textit{expr} \cup \textit{expr} & \text{(union)} \\
| & \textit{expr} \cap \textit{expr} & \text{(intersection)} \\
| & \textit{expr} \setminus \textit{expr} & \text{(difference)} \\
| & \textit{expr} \fatsemi \textit{expr} & \text{(join)} \\
| & \textit{expr} \times \textit{expr} & \text{(product)} \\
| & \{\textit{varDecls} \mid \textit{formula}\} & \text{(comprehension)} \\
| & \textbf{univ} & \text{(universal set)} \\
| & \textbf{none} & \text{(empty set)} \\
| & \textbf{iden} & \text{(identity)}
\end{array}
$$

$expr ::=$

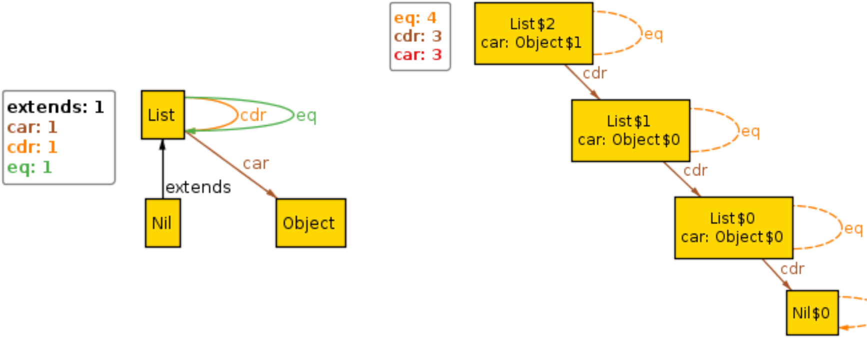
var	(variable)
$\mid expr = expr$	(equality)
$\mid \sim expr$	(transpose)
$\mid \hat{}expr$	(clousure)
$\mid expr + expr$	(union)
$\mid expr \;\&\; expr$	(intersection)
$\mid expr - expr$	(difference)
$\mid expr \cdot expr$	(join)
$\mid expr \rightarrow expr$	(product)
$\mid \{varDecls \mid formula\}$	(comprehension)
$\mid$ **univ**	(universal set)
$\mid$ **none**	(empty set)
$\mid$ **iden**	(identity)

$$
\begin{aligned}
&\mathit{intexpr} ::= \\
&\qquad \mathit{integer} && \text{(literal)} \\
&\qquad \mid \#\mathit{expr} && \text{(cardinality)} \\
&\qquad \mid \textbf{sum}\ (\mathit{expr}) && \text{(sum)} \\
&\qquad \mid \mathit{intexpr}\ \{+ \mid - \mid \times \mid \div\}\ \mathit{intexpr} && \text{(arithmetic)} \\
&\mathit{varDecls} ::= (\mathit{variable} : \mathit{expr})^* \\
&\mathit{variable} ::= \mathit{identifier}
\end{aligned}
$$

Alloy Demonstration

A Lisp-like List

datatype List = Nil | Cons **of** Element ∗ List



Research Road-map

Alloy Benchmarks
Proving Properties vs.
Finding counterexamples.
Producing Model, and Core.
Producing Proofs.
Fixed Bitwidth Arithmetic vs.
Linear Integer Arithmetic.

Alloy Analyzer → KodKod Java API →

Mace-Style Model Finder

CVC4+AX

KodKod Model Finder

SMT Solvers

Triggers?

Vampire

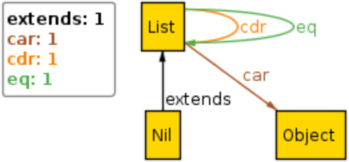
Understand the effect of following features on the performance:

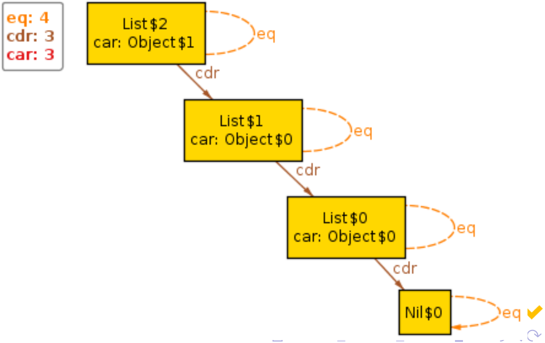
1. Partial Models.
2. Decidable Fragments, Satisfiability with Finite Extension,
2. Binary, Ternary, N-ary Relations.
3. Hierarchical Types.
4. Nested Relational Joins.
5. Transitive Closure.
6. Nested Quantifiers.
7. Set Cardinality.
8. Arithmetic operations.
9. Ordered Relations.

KodKod Walktrough

A Lisp-like List

datatype List = Nil | Cons **of** Element ∗ List





Universe

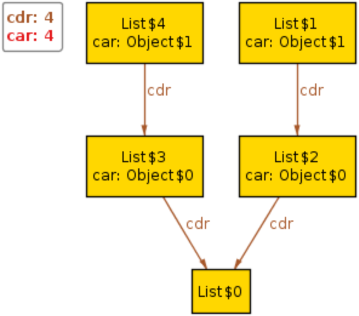
$\{o_0,\ o_1,\ l_0,\ l_1,\ l_2,\ l_3,\ l_4,\ l_5\}$

Bounds

$$\begin{aligned}
&List &&:_1 [\{\langle l_0\rangle, \langle l_1\rangle, \langle l_2\rangle, \langle l_3\rangle, \langle l_4\rangle, \langle l_5\rangle\}] \\
&Object &&:_1 [\{\langle o_0\rangle, \langle o_1\rangle\}] \\
&Nil &&:_1 [\{\}, \{\langle l_0\rangle, \langle l_1\rangle, \langle l_2\rangle, \langle l_3\rangle, \langle l_4\rangle, \langle l_5\rangle\}] \\
&car &&:_2 [\{\langle l_4, o_1\rangle, \langle l_3, o_0\rangle, \langle l_2, o_0\rangle, \langle l_1, o_1\rangle\}, \\
& && \quad \{\langle x, y\rangle \mid x : List \wedge y : Object\}] \\
&cdr &&:_2 [\{\langle l_4, l_3\rangle, \langle l_3, l_0\rangle, \langle l_2, l_0\rangle, \langle l_1, l_2\rangle\}, \\
& && \quad \{\langle x, y\rangle \mid x : List \wedge y : List\}] \\
&eq &&:_2 [\{\}, \{\langle x, y\rangle \mid x : List \wedge y : List\}]
\end{aligned}$$

Universe

$\{o_0, o_1, l_0, l_1, l_2, l_3, l_4, l_5\}$



Universe

$\{o_0, o_1, l_0, l_1, l_2, l_3, l_4, l_5\}$

KodKod API

```
String List0 = "List0"; String List1 = "List1";
String List2 = "List2"; String List3 = "List3";
String List4 = "List4"; String List5 = "List5";
String Object0 = "Object0";
String Object1 = "Object1";

Universe universe = new Universe(List0, List1,
List2, List3, List4, List5, Object0, Object1);
```

Translation

```
(declare-datatypes () ((univ (Object!1) (Object!1)
           (List!0) (List!1) ... (List!4) (List!5))))

(declare-fun Object (univ) Bool)
(declare-fun List (univ) Bool)
...
(declare-fun eq (univ univ) Bool)

(assert (Object Object0))
(assert (Object Object1))
(assert (List List0))
...
(assert (cdr List1 List2))
```

Axioms

1. *Nil* is a *List*.
2. *Nil* is a singleton.
3. *Nil* list has neither *car* nor *cdr*.
4. A Non-nil *List* has some *car* and *cdr*.
5. *Nil* is always reachable from any *List*.
6. Two lists are equal iff the objects they point to are same and the *Lists* they point are equal.
7. *car* relation is a partial function.

Axioms

1. $Nil \subseteq List$
2. $\texttt{one}\ Nil$
3. $\texttt{no}\,(Nil.cdr \cup Nil.car)$
4. $\forall l : List - Nil \mid \texttt{some}\,(l.cdr) \wedge \texttt{some}\,(l.car)$
5. $\forall l : List \mid Nil \subseteq (l.^{*}cdr)$
6. $\forall a, b : List \mid a \subseteq b.eq\ \texttt{iff}$
   $(a.car = b.car) \wedge (a.cdr \subseteq (b.cdr).eq)$
7. $\forall l : List \mid \texttt{lone}\,(l.car)$

(constraints)

Alloy

```
(all l: one List | lone (l.car))
```

KodKod API

```
Relation List = Relation.unary("List");
Relation car = Relation.binary("car");
Variable l = Variable.unary("l");
Formula f1 = l.join(car).lone()
                 .forAll(l.oneOf(List));
```

Alloy

```
(all l: one List | lone (l.car))
```

SMTLIB

```
(forall ((l univ))
  (=> (List l)
    (forall ((x!1 univ) (x!2 univ))
      (=> (and (cdr l x!1) (cdr l x!2))
      (= x!1 x!2))))))
```

Alloy

```
(one Nil)
```

KodKod API

```
Relation Nil = Relation.unary("Nil");
Formula f2 = Nil.one();
```

Alloy

```
(one Nil)
```

SMTLIB

```
(and (exists ((x!0 univ) (Nil x!0))
     (forall ((x!0 univ) (x!1 univ))
       (=> (and (Nil x!0) (Nil x!1))
           (and (= x!0 x!1))))))
```

Alloy

```
(all l: one (List - Nil) |
    (some (l.cdr) and some (l.car)))
```

KodKod API

```
Relation car = Relation.binary("cdr");
Formula f3 = l.join(cdr).some()
          .and(l.join(car).some())
            .forAll(
               l.oneOf(List.difference(Nil)));
```

Alloy

```
(all l: one (List - Nil) |
    (some (l.cdr) and some (l.car)))
```

SMTLIB

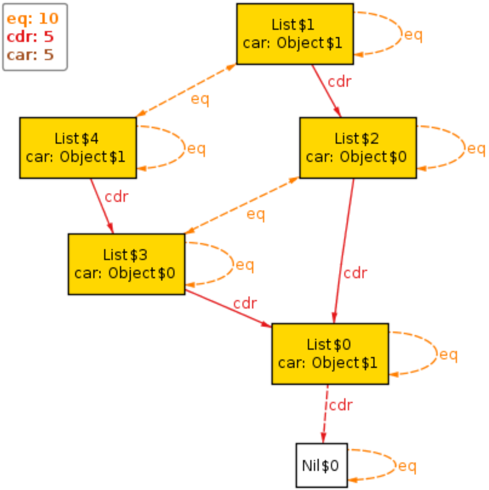
```
(forall ((l univ))
  (=> (and (List l) (not (Nil l)))
      (and
        (exists ((x!1 univ)) (cdr l x!1))
        (exists ((x!1 univ)) (car l x!1)))))
```

Outcome

SAT

$$
\begin{aligned}
List &\mapsto \{\langle l_0\rangle, \langle l_1\rangle, \langle l_2\rangle, \langle l_3\rangle, \langle l_4\rangle, \langle l_5\rangle\}\\
Object &\mapsto \{\langle o_0\rangle, \langle o_1\rangle\}\\
Nil &\mapsto \{\langle l_5\rangle\}\\
car &\mapsto \{\langle l_4, o_1\rangle, \langle l_3, o_0\rangle, \langle l_2, o_0\rangle, \langle l_1, o_1\rangle, \langle l_0, o_1\rangle\} \qquad \text{(model)}\\
cdr &\mapsto \{\langle l_4, l_3\rangle, \langle l_3, l_0\rangle, \langle l_2, l_0\rangle, \langle l_1, l_2\rangle, \langle l_0, l_5\rangle\}\\
eq &\mapsto \{\langle l_5, l_5\rangle, \langle l_4, l_4\rangle, \langle l_3, l_3\rangle, \langle l_2, l_2\rangle, \langle l_1, l_1\rangle,\\
&\quad \langle l_0, l_0\rangle, \langle l_4, l_1\rangle, \langle l_1, l_4\rangle, \langle l_3, l_4\rangle, \langle l_2, l_3\rangle\}
\end{aligned}
$$

Outcome


eq: 10
cdr: 5
car: 5
List$1
car: Object$1
List$4
car: Object$1
List$2
car: Object$0
List$3
car: Object$0
List$0
car: Object$1
Nil$0
eq
cdr

Comparison with Z3's MBQI