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

Ferhat Erata[1] Wolfgang Ahrendt[2,3] Gerardo Schneider[2,3]

[1]UNIT Information Technologies Ltd., Izmir, Turkey
[2]Chalmers University of Technology, Gothenburg, Sweden
[3]University of Gohenburg, Gothenburg, Sweden

41st ACM/IEEE International Conference on Software Engineering, Montréal, Canada
25-31 May, 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

ITEA 3

EUREKA Σ!

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

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

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

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

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)

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

$$
\begin{array}{rl}
\mathit{expr} ::= & \\
\quad \mathit{var} & \text{(variable)} \\
\quad \mid \mathit{expr} = \mathit{expr} & \text{(equality)} \\
\quad \mid \sim \mathit{expr} & \text{(transpose)} \\
\quad \mid \hat{}\mathit{expr} & \text{(clousure)} \\
\quad \mid \mathit{expr} + \mathit{expr} & \text{(union)} \\
\quad \mid \mathit{expr} \ \& \ \mathit{expr} & \text{(intersection)} \\
\quad \mid \mathit{expr} - \mathit{expr} & \text{(difference)} \\
\quad \mid \mathit{expr} \cdot \mathit{expr} & \text{(join)} \\
\quad \mid \mathit{expr} \rightarrow \mathit{expr} & \text{(product)} \\
\quad \mid \{\mathit{varDecls} \mid \mathit{formula}\} & \text{(comprehension)} \\
\quad \mid \textbf{univ} & \text{(universal set)} \\
\quad \mid \textbf{none} & \text{(empty set)} \\
\quad \mid \textbf{iden} & \text{(identity)}
\end{array}
$$

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

Relations

$$
\begin{aligned}
\mathsf{Name} &= \{(\mathsf{N0}),(\mathsf{N1}),(\mathsf{N2})\} \\
\mathsf{Alias} &= \{(\mathsf{A0})\} \\
\mathsf{alias} &= \{(\mathsf{N2},\mathsf{A0})\} \\
\mathsf{Address} &= \{(\mathsf{D0}),(\mathsf{D1}),(\mathsf{D2})\} \\
\mathsf{addr} &= \{(\mathsf{N0},\mathsf{D0}),(\mathsf{A0},\mathsf{D2})\}
\end{aligned}
$$

Queries

$$
\begin{aligned}
\mathsf{Name.addr} &= \{(\mathsf{N0}),(\mathsf{N1}),(\mathsf{N2})\}.\{(\mathsf{N0},\mathsf{D0}),(\mathsf{A0},\mathsf{D2})\} \\
&= \{(\mathsf{D0})\} \\
\{(\mathsf{N1})\}.\mathsf{addr} &= \{\ \} \\
\mathsf{Address.\sim addr} &= \{(\mathsf{D0}),(\mathsf{D1}),(\mathsf{D2})\}.\{(\mathsf{D0},\mathsf{N0}),(\mathsf{D2},\mathsf{A0})\} \\
&= \{(\mathsf{N0}),\ (\mathsf{A0})\}
\end{aligned}
$$

Relations

```
Name    = {(N0),(N1),(N2)}
Alias   = {(A0)}
alias   = {(N2,A0)}
Address = {(D0),(D1),(D2)}
addr    = {(N0,D0),(A0,D2)}
```

Queries

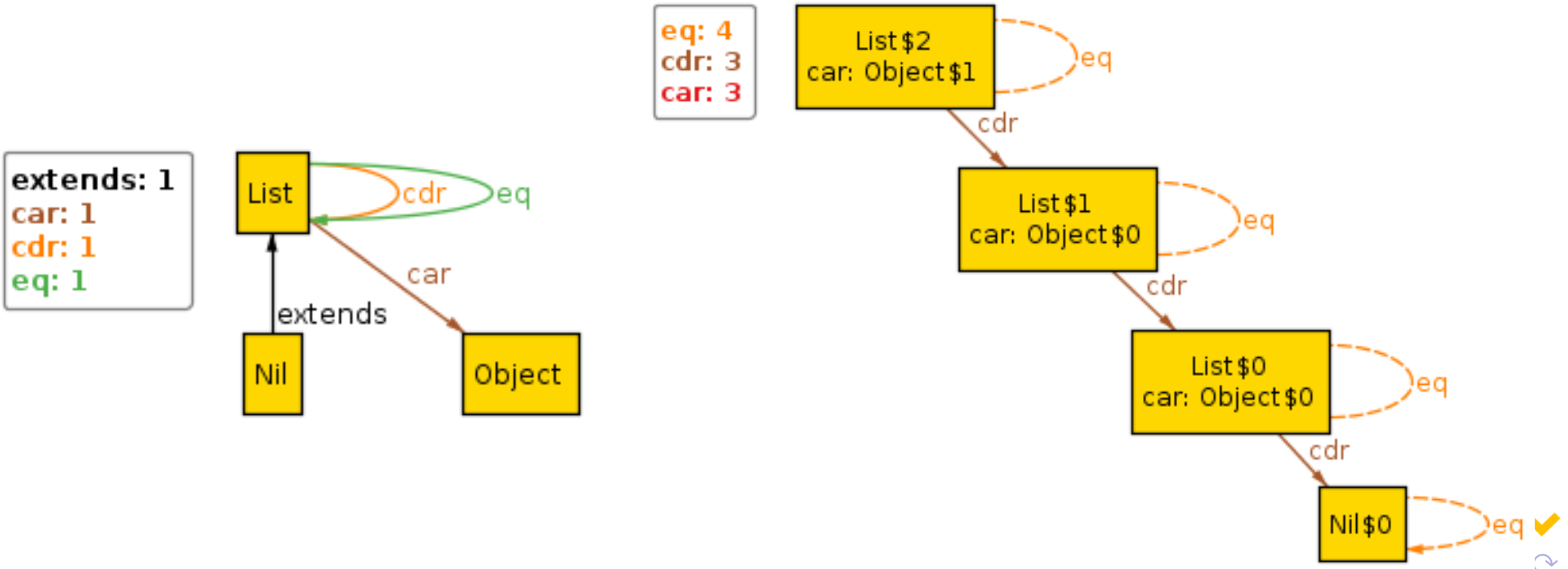
```
addr + alias     = {(N0,D0),(A0,D2),(N2,A0)}
^(addr + alias)  = {(N0,D0),(A0,D2),(N2,A0),(N2,D2)}

Name.^(addr + alias) = {(D0),(D2)}
({(N2)}.alias).addr  = {(D2)}
```

Alloy Demonstration

A Lisp-like List

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



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

KodKod Java API → CVC4+AX

KodKod Java API → KodKod Model Finder ✓

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.

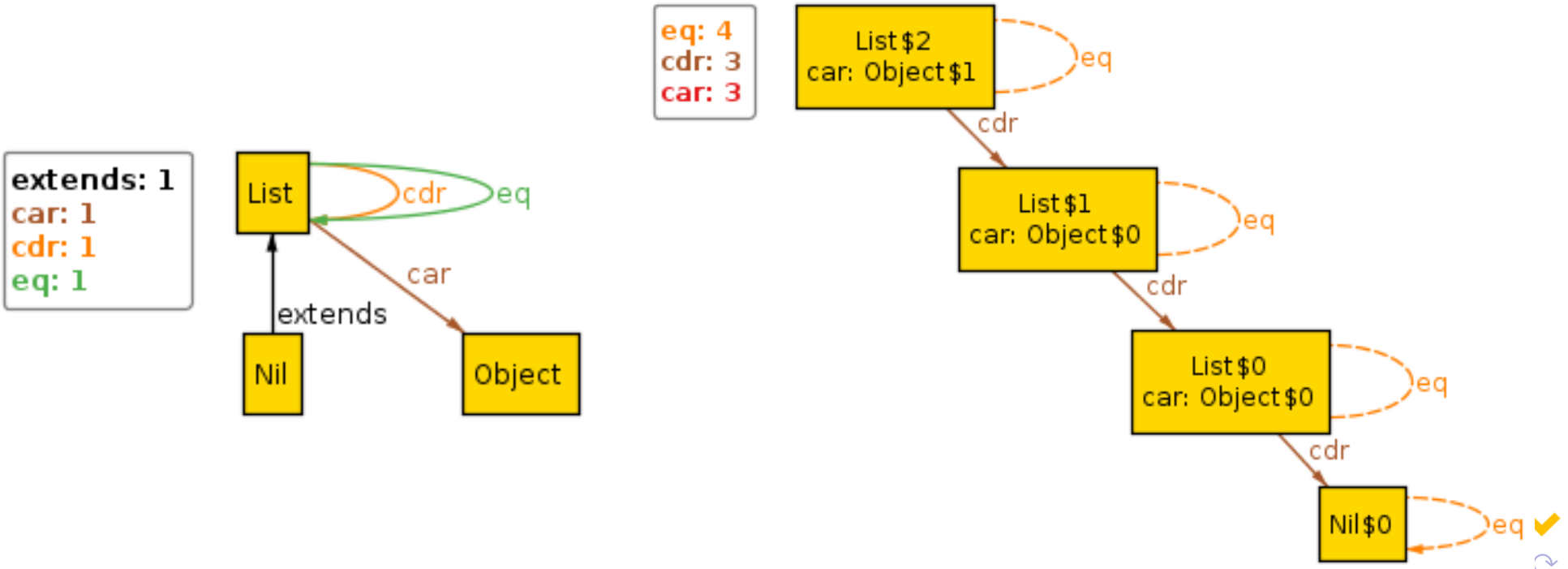
KodKod Java API → SMT Solvers ✓ Triggers?

KodKod Java API → Vampire

KodKod Walktrough

A Lisp-like List

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



Universe

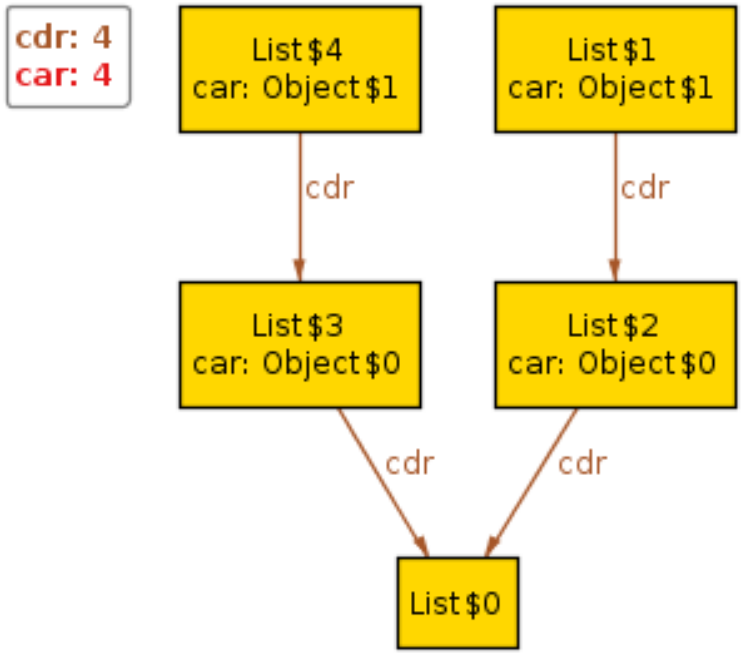
$\{o_0,\ o_1,\ l_0,\ l_1,\ l_2,\ l_3,\ l_4,\ l_5\}$

Bounds

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

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

```
1  String List0 = "List0"; String List1 = "List1";
2  String List2 = "List2"; String List3 = "List3";
3  String List4 = "List4"; String List5 = "List5";
4  String Object0 = "Object0";
5  String Object1 = "Object1";
6
7  Universe universe = new Universe(List0, List1,
8  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

```
1  Relation List = Relation.unary("List");
2  Relation car = Relation.binary("car");
3  Variable l = Variable.unary("l");
4  Formula f1 = l.join(car).lone()
5                  .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

```
6  Relation Nil = Relation.unary("Nil");
7  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

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

Alloy

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

SMTLIB

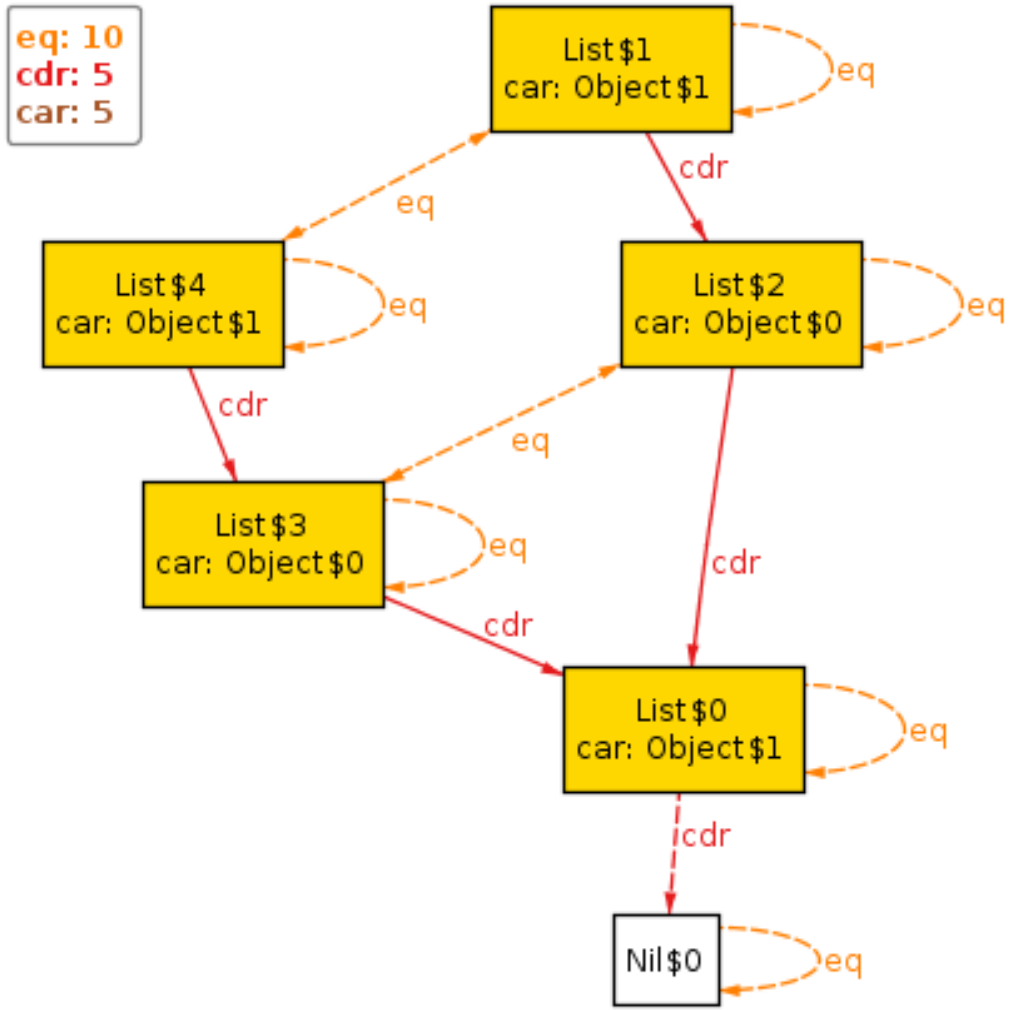
```
(forall ((l univ))
  (=> (and (List l) (not (Nil l)))
      (and
        (exists ((x!1 univ)) (cdr l x!1))
        (exists ((x!1 univ)) (car l x!1)))))
```

Outcome

SAT

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

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