AlloyInEcore: Embedding of First-Order Relational Logic into Meta-Object Facility for Automated Model Reasoning

Ferhat Erata  
UNIT Information Technologies R&D, Turkey

Ivan Kurtev  
Altran Netherlands, the Netherlands

Arda Goknil  
University of Luxembourg, Luxembourg

Bedir Tekinerdogan  
Wageningen University, the Netherlands

ABSTRACT

We present AlloyInEcore, a tool for specifying metamodels with their static semantics to facilitate automated formal reasoning on models. Software development projects require that software systems be specified in various models (e.g., requirements models, architecture models, test models, and source code). It is crucial to reason about those models to ensure the correct and complete system specifications. AlloyInEcore allows the user to specify metamodels with their static semantics, while, using the semantics, it automatically detects inconsistent models, and completes partial models. It has been evaluated on three industrial case studies in the automotive domain (https://modelwriter.github.io/AlloyInEcore/).

CCS CONCEPTS

• Software and its engineering → Specification languages; Semantics; Formal methods;

KEYWORDS

Formal Reasoning; Modeling; Relational Logic; Alloy; KodKod

ACM Reference Format:
Ferhat Erata, Arda Goknil, Ivan Kurtev, and Bedir Tekinerdogan. 2018. AlloyInEcore: Embedding of First-Order Relational Logic into Meta-Object Facility for Automated Model Reasoning. In Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE ’18), November 4–9, 2018, Lake Buena Vista, FL, USA. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3256024.3264588

1 INTRODUCTION

Model Driven Engineering (MDE) is becoming a crucial practice in industry due to the increasing complexity of software systems that warrant better support for managing development artifacts [17]. In MDE, software is developed by successively transforming abstract models to more concrete ones. Each model conforms to its metamodel, an artefact usually created using Ecore [7], a de facto industry standard for metamodeling and an example of implementation of the Meta-Object Facility (MOF) [11] which describes the

means to create and manipulate models and metamodels. An important challenge in MDE is providing ability of automated, formal reasoning on models, e.g., checking model consistency and completing partial models [13, 16].

We present a tool, AlloyInEcore, which allows specification of metamodels with their static semantics and facilitates multiple forms of automated, formal reasoning on models. AlloyInEcore is targeted at environments that require integration and reasoning on heterogeneous models. Such environments are often encountered within the context of our research [18, 19] in collaboration with Ford-Otosan [12]. The key idea behind AlloyInEcore is that the static semantics of an Ecore metamodel can be specified within a simple first-order logic of sets and relations to support reasoning on models conforming to the metamodel.

Alloy [20] is a declarative modeling language based on first-order relational logic. It has been explored by the MDE community for the purpose of analyzing UML/OCL models [1, 6, 26]. Most of the existing tools and approaches use a transformation of UML/OCL models to Alloy, which, however, does not support directly some important concepts like multiple inheritance, generic types, and type parameters due to the fundamental differences between UML/OCL and Alloy notations [6, 30]. In the case of dealing with various models in different abstraction levels, it is required to enable the specification of such concepts, and herewith multiple forms of model reasoning. To do so, AlloyInEcore provides the following major features: (i) Alloy-like notation embedded into Ecore to specify the static semantics of metamodels based on First-order Logic (FOL), relational operators, and transitive closure; (ii) direct translation of Ecore metamodels into the language of Kodkod [29], an efficient SAT-based constraint solver for FOL with relational algebra. In this way we avoid the problems related to the differences between Alloy and Ecore. Our tool performs two major model reasoning tasks: completing partial models and detecting inconsistent model parts.

2 RELATED WORK

Several formal analysis methods and specification languages have been proposed relying on modern SAT-solvers, SMT solvers and theorem provers (e.g., Formula [21] using Z3 SMT-solver [24], Clafer [2] using Alloy along with Choco CSP solver [22], and Alloy using KodKod [29] that relies on SAT solvers like Minisat [8]).

A number of MDE solutions and tools provide automated model reasoning using existing formal analysis methods and specification languages based on constraint logic programming (e.g., [4, 5]), SAT-based model finders (e.g., [1, 6, 26, 27]), and SMT solvers (e.g., [15, 25]). However, to the best of our knowledge, none of
them provides a method that embeds FOL augmented with relational calculus into MOF/Ecore to specify the static metamodel semantics with the support for partial models, composition, cardinality constraints, multiple inheritance, generic types, and type parameters. For instance, Anastakis et al. [1] advocate to transform UML/OCL specifications to Alloy for automated model reasoning. The proposed transformation does not support multiple inheritance, generic types, and type parameters because of the differences between UML/OCL and Alloy notations. Lightning [14] is a tool-supported approach for defining some aspects of Domain-Specific Languages (e.g., abstract syntax and semantics) entirely in Alloy. In our approach, language designers can use AlloyInEcore to specify the abstract syntax of a language as an Ecore metamodel enriched with embedded Alloy-like statements. USE [23] is a tool for analyzing models expressed in UML and OCL. Similarly to our approach, USE translates models into relational logic and relies on the Kodkod library.

3 TOOL OVERVIEW

AlloyInEcore is a generalization of our previous tool Tarski [9, 10], which reasons on trace links in traceability models using configurable trace link semantics. Fig. 1 presents an overview of our tool. In Step 1, the user specifies an Ecore metamodel and its static semantics expressed in FOL augmented with the relational calculus [28] embedded in Ecore. To do so, AlloyInEcore natively supports Alloy in Ecore with a custom Eclipse editor.

Once the user specifies the metamodel and its semantics, AlloyInEcore allows creation of instance model(s) conforming to the metamodel (Step 2). After the model is created, the tool proceeds to Step 3 using automated model reasoning. In the following, we elaborate each step using the theory of lists as a running example.

3.1 Specification of Metamodels and Semantics

As the first step, the user specifies a metamodel and its semantics in our Alloy-like notation embedded into Ecore. The user can create the metamodel using any graphical, textual, or tree-based Ecore model editor including our AlloyInEcore editor. Fig. 2 gives the theory of lists metamodel in the Ecore graphical editor.

The user uses our editor to specify the static semantics of the metamodel (see Fig. 3). The keywords in blue and brown are introduced by AlloyInEcore for specifying the metamodel semantics, while the ones in red are the Ecore keywords for defining the metamodel itself. Some of the AlloyInEcore keywords (e.g., ghost, model, and nullable) were borrowed from JML [3].
two lists are the same (‘a.car = b.car’ in Line 13), the subsequent List instances are equal (‘a.cdr in b.cdr eq’ in Line 14), and the List instances are of the same type (‘a.class = b.class’ in Line 14). The invariant in Line 16 guarantees that each Vehicle instance is in at least one list (see the some keyword). The Nil class represents the empty list (Line 19). The one keyword makes the Nil class a singleton set, which means there can be only one Nil instance in a model. A Nil instance has neither the car nor the cdr property (Lines 20-21), while a Non-nil List instance has both car and cdr (Line 22). A Nil instance is always a subsequent list of any List instance (Line 23). The singleton class Memory holds the Vehicle and List classes (see the compiles keyword in Lines 27-28). It is important to note that ‘List<? extends Vehicle>’ represents the List instances of any subclass of Vehicle (Line 28). Each Vehicle instance has a unique name (Line 33) and there is always exactly one Vehicle instance with the name “Ford F-150 XLT” (Line 34). There are two types of vehicles: NonEnginedVehicle and EnginedVehicle (Line 37-38). TruckList and CarList are lists of EnginedVehicles (Lines 40-41), while BicycleList is a list of NonEnginedVehicle (Line 42).

3.2 Specification of Models
The user can use any graphical, textual, or tree-based Ecore model editor to specify models conforming to the metamodel (Step 2 in Fig. 1). Before creating any model, AlloyInEcore automatically checks if the user can specify at least one valid model that conforms to the metamodel and its static semantics. The user may have specified some contradicting invariants where it is not possible to create a valid model. AlloyInEcore automatically identifies the contradictions in the metamodel specification and notifies the user.

3.3 Automated Reasoning on Models
Model completion and consistency checking aim at deriving new instances and relations in the given model, and determining model parts violating the metamodel semantics, respectively. These two activities are processed as a single reasoning activity because they use the same reasoning machinery. The consistency checking can be considered as part of model completion because a partial model is completed only if it is consistent.

3.3.1 Checking Model Consistency. AlloyInEcore takes a model and its metamodel as input, and automatically identifies, using the static metamodel semantics, inconsistent model parts as output. AlloyInEcore provides an explanation of the inconsistency by giving all the instances and relations causing the inconsistency. Fig. 4 gives three AlloyInEcore panes for an example inconsistent model of the theory of lists. The first pane in Fig. 4 gives the inconsistent model, while, in the second pane, AlloyInEcore highlights part of the metamodel semantics causing the inconsistency. Although the cdr property of the List class is given acyclic (see the highlighted acyclic keyword in Line 10), the red colored cdr in TruckList$0 is referring to the instance itself. The third pane in Fig. 4 gives the first order relational logic formula that corresponds to the acyclic keyword for further explanation of the inconsistency.

3.3.2 Completing Partial Models. If the given model is consistent, AlloyInEcore automatically deduces new instances and relations in the input model using the static metamodel semantics. The model is completed only if it is consistent and not an exact model (i.e., a
We presented a tool that enables the specification of metamodel semantics in AlloyInEcore, including executable files and a screencast covering its use. AlloyInEcore has been implemented as an Eclipse plug-in. We use it to evaluate the effectiveness of our tool in specifying metamodel semantics in AlloyInEcore (the 1st, 2nd and 3rd columns in Table 1).

### Table 1: Number of Classes, Properties, Invariants, Models and Completed & Inconsistent Parts in the Case Studies

| Metaclasses | Properties | Invariants | Model Elements | Comple. Elements | Inconsist. Elements |
|-------------|------------|------------|----------------|------------------|---------------------|
| #1          | 3          | 7          | 42             | 116              | 480                 | 5                   |
| #2          | 5          | 8          | 6              | 51               | 114                 | 1                   |
| #3          | 15         | 12         | 10             | 135              | 432                 | 2                   |

To evaluate the output, we had semi-structured interviews with the engineers. All the completed model parts and the identified inconsistencies were confirmed by the engineers to be correct (the 4th and 5th columns). The engineers considered the automated reasoning on models to be highly valuable. The Alloy-like notation embedded into Ecore was sufficient and easy for the engineers to specify the metamodel semantics in the case studies. For the largest model (the 3rd row), it took 126 secs to perform the reasoning.

### 5 IMPLEMENTATION & AVAILABILITY

AlloyInEcore has been implemented as an Eclipse plug-in. We use Kodkod [29] to perform automated reasoning on models based on the metamodel semantics. AlloyInEcore translates the input metamodel and semantic specification into a first order relational formula. It also translates the input model into a Universe and Bounds in Kodkod. Kodkod translates the formula and the bounds into a Boolean satisfiability (SAT) problem to invoke an off-the-shelf SAT solver. If the SAT solver finds a SAT solution to the problem, Kodkod translates that SAT solution into a solution to the formula from which AlloyInEcore derives the completed model.

AlloyInEcore is approximately 31K lines of Java code, excluding comments and third-party libraries. Additional details about AlloyInEcore, including executable files and a screencast covering motivations, are available on the tool’s website at:

[https://modelwriter.github.io/AlloyInEcore/](https://modelwriter.github.io/AlloyInEcore/)

### 6 CONCLUSION

We presented a tool that enables the specification of metamodel semantics for automated model reasoning. The key characteristics of our tool are (1) enabling the user to specify metamodels and their semantics in a single environment, (2) identifying inconsistent model parts, and (3) completing partial models. The tool has been evaluated over three industrial case studies.
APPENDICES

We provide three appendices for the paper. In Appendix A, we provide source code, tool and data availability details. In Appendix B, we provide the simplified and readable version of the grammar of the AlloyInEcore’s recognizer. Finally in Appendix C, we present the log file representing the full translation of the use case, Theory of List, running on the AlloyInEcore.

A AVAILABILITY & OPEN SOURCE LICENSE

Source Codes, Screencast and Datasets. The source codes files and datasets of AlloyInEcore are publicly available for download and use at the project website. A screencast and the installation steps for AlloyInEcore are also available at the same website and can be found at:

https://modelwriter.github.io/AlloyInEcore/

AlloyInEcore is being developed under Work Package 3 and 5 within ASSUME [19] project, labeled by the European Union’s EUREKA Cluster programme ITEA (Information Technology for European Advancement). Further details about the project can be found at:

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

B THE GRAMMAR OF THE FRONT-END

The simplified version of the grammar that parses AlloyInEcore models is presented in the following.

```
package ::= package identifier { package* classifier* invariant* }
classifier ::= class | dataType | enum
class ::= abstract? cardinality? class identifier template
        extends type? bound? { feature* invariant* }
feature ::= attribute | reference
attribute ::= qualifier* attribute cardinality? identifier
        : type multiplicity { id? derived? }
reference ::= qualifier* property cardinality? identifier
        : type multiplicity { derived? composes? props* }
mult ::= [ constant .. constant? ] ( [ ] )
type ::= identifier (< argument ( argument? > )?
argument ::= type | wildcard
wildcard ::= ? ( ( extends | super | type ) )?
template ::= < parameter , parameter? >
parameter ::= identifier ( extends type ( & type? ) )?
        bound ::= [ constant, constant ]
invariant ::= formula
qualifier ::= model | ghost
cardinality ::= one | lone | some | no
```

props ::= cyclic | transitive | reflexive | irreflexive | symmetric
        | asymmetric | antisymmetric | total | functional | surjective
        | injective | bijective | complete | bijection | preorder
        | equivalence | partialorder | totalorder

```
formula ::= 
    expr ⊂ expr
    | expr = expr
    | some expr
    | one expr
    | lone expr
    | no expr
    | ¬ formula
    | formula ∧ formula
    | formula ∨ formula
    | formula ⇒ formula
    | ∀ varDecs | formula
    | ∃ varDecs | formula
    | intexpr { = | < | > } intexpr

expr ::= var
    | expr = expr
    | ~ expr
    | “expr
    | expr ∪ expr
    | expr ∩ expr
    | expr \ expr
    | expr · expr
    | expr → expr
    | formula ? expr :: expr
    | ( varDecs | formula )
    | π(expr, intexpr*)
    | int2expr (intexpr)
    | univ

intexpr ::= 
    integer
    | expr
    | sum ( e )
    | intexpr { + | _ | × | ÷ } intexpr

varDecs ::= ( variable : expr )
variable ::= identifier
```
C TRANSLATION OF THE THEORY OF LIST

In Figure 3, theoryoflist.ecore file and the partial instance given in Figure 4 are being translated into KodKod API calls and then the result is interpreted to generate the Figure 5. The following listings are taken from AlloyInEcore’s log file running this scenario and showing how AlloyInEcore constructs a KodKod problem from given metamodel and partial model to detect inconsistency (UNSAT) and to complete partial model (SAT).

1. The universe of Theory of List instance (cf. Listing 1)
2. Bounds for Unary Relations (cf. Listing 2)
3. Bounds for Internal Binary Relations (cf. Listing 3)
4. Bounds for User’s Binary Relations (cf. Listing 4)
5. Generated Formulas (cf. Listing 5)
6. The Outcome and the statistics (cf. Listing 6)
7. The Generated Model (cf. Listings 7 and 8)
AlloyInEcore: Embedding of First-Order Relational Logic into ... ESEC/FSE ’18, November 4–9, 2018, Lake Buena Vista, FL, USA

Listing 1: Universe

| Universe Contents |
|-------------------|
| 1 | [EnginedVehicle84, EnginedVehicle184, TruckList729, Vehicle_3, TruckList848, TruckList400, Nil, EString_1, TruckList610, EString_3, TruckList, 3, 5, Vehicle, 1, EnginedVehicle, List_6, CarList, List, Object, NonEnginedVehicle, BicycleList, Memory, EnginedVehicle284, FORD F-150 XLT, String_0, CarList493, EInt, 4, 2, EString, List_5, Memory46] |

Listing 2: Bounds for Unary Relations

| Relation Bounds |
|-----------------|
| 2 | relation bounds: |
| 3 | BicycleList: [[]], [[]], [[]], [[]] |
| 4 | Class: [[Nil], [TruckList], [Vehicle], [EnginedVehicle], [CarList], [List], [Object], [NonEnginedVehicle], [BicycleList], [Memory], [EInt], [EString]] |
| 5 | Nil<EnginedVehicle >=: [[]], [[]], [[]] |
| 6 | Class<EnginedVehicle >=: [[EnginedVehicle]] |
| 7 | EnginedVehicle: [[EnginedVehicle84], [EnginedVehicle184], [EnginedVehicle284]], [[EnginedVehicle84], [EnginedVehicle184], [Vehicle_3], [EnginedVehicle284]] |
| 8 | Class<Vehicle >=: [[Vehicle]] |
| 9 | Nil<NonEnginedVehicle >=: [[List_6], [List_5]] |
| 10 | List<? extends Vehicle >=: [[TruckList729], [TruckList848], [TruckList400], [TruckList610], [CarList493]], [[TruckList729], [TruckList848], [TruckList400], [TruckList610], [List_6], [CarList493], [List_5]] |
| 11 | T r u c k L i s t : [T r u c k L i s t 7 2 9 , T r u c k L i s t 8 4 8 , T r u c k L i s t 4 0 0 , T r u c k L i s t 6 1 0 , C a r L i s t 4 9 3 , L i s t _ 5 ], [T r u c k L i s t 7 2 9 , T r u c k L i s t 8 4 8 , T r u c k L i s t 4 0 0 , T r u c k L i s t 6 1 0 , L i s t _ 6 , C a r L i s t 4 9 3 , L i s t _ 5 ]] |
| 12 | Class<CarList >=: [[CarList]] |
| 13 | Class<Nil >=: [[Nil]] |
| 14 | "FORD F-150 XLT": [[FORD F-150 XLT]] |
| 15 | Memory: [[Memory46]] |
| 16 | List<EnginedVehicle >=: [[TruckList729], [TruckList848], [TruckList400], [TruckList610], [CarList493]], [[TruckList729], [TruckList848], [TruckList400], [TruckList610], [List_6], [CarList493], [List_5]] |
| 17 | Class<List >=: [[List]] |
| 18 | Vehicle: [[EnginedVehicle84], [EnginedVehicle184], [EnginedVehicle284]], [[EnginedVehicle84], [EnginedVehicle184], [Vehicle_3], [EnginedVehicle284]] |
| 19 | Class<BicycleList >=: [[BicycleList]] |
| 20 | EString: [[EString_1], [EString_3], [FORD F-150 XLT]], [[EString_1], [EString_3], [FORD F-150 XLT], [String_0]] |
| 21 | Class<EInt >=: [[EInt]] |
| 22 | List<NonEnginedVehicle >=: [[List_6], [List_5]] |
| 23 | Class<EString >=: [[EString]] |
| 24 | Object: [[EnginedVehicle84], [EnginedVehicle184], [TruckList729], [TruckList848], [TruckList400], [TruckList610], [EnginedVehicle284], [CarList493], [List_5]], [[EnginedVehicle84], [EnginedVehicle184], [TruckList729], [Vehicle_3], [TruckList848], [TruckList400], [TruckList610], [List_6], [EnginedVehicle284], [CarList493], [List_5]] |
| 25 | Class<TruckList >=: [[TruckList]] |
| 26 | NonEnginedVehicle: [[]], [[]], [[]], [[]], [[]], [[]], [[]], [[]] |
| 27 | CarList: [[CarList493], [List_6], [CarList493], [List_5]] |
| 28 | EInt: [[3], [5], [1], [4], [2]] |
| 29 | TruckList: [[TruckList729], [TruckList848], [TruckList400], [TruckList610]], [[TruckList729], [TruckList848], [TruckList400], [TruckList610], [List_6], [List_5]] |
| 30 | Class<NonEnginedVehicle >=: [[NonEnginedVehicle]] |
| 31 | Nil: [[]], [[]], [[]], [[]] |
| 32 | Class<Memory >=: [[Memory]] |
| 33 | Class<Object >=: [[Object]] |
Listing 3: Lower and Upper Bounds for Internal Binary Relations

34  extends: [[[Vehicle, EnginedVehicle], [Vehicle, NonEnginedVehicle], [List, Nil], [List, TruckList], [List, CarList], [List, BicycleList], [Object, Nil], [Object, TruckList], [Object, Vehicle], [Object, EnginedVehicle], [Object, CarList], [Object, List], [Object, NonEnginedVehicle], [Object, BicycleList]]

35  super: [[[Nil, List], [Nil, Object], [TruckList, List], [TruckList, Object], [Vehicle, Object], [EnginedVehicle, Vehicle], [EnginedVehicle, Object], [CarList, List], [CarList, Object], [List, Object], [NonEnginedVehicle, Vehicle], [NonEnginedVehicle, Object], [BicycleList, List], [BicycleList, Object]]

36  lists: [[[Memory46, TruckList729], [Memory46, TruckList848], [Memory46, TruckList400], [Memory46, TruckList610], [Memory46, CarList493]], [[Memory46, TruckList729], [Memory46, TruckList848], [Memory46, TruckList400], [Memory46, TruckList610], [Memory46, List_, 6], [Memory46, CarList493], [Memory46, List_, 5]]

37  class: [[[EnginedVehicle84, Vehicle], [EnginedVehicle84, EnginedVehicle], [EnginedVehicle84, Object], [EnginedVehicle84, EngineedVehicle], [EngineedVehicle84, Vehicle], [EnginedVehicle84, EngineedVehicle], [EngineedVehicle84, Object], [TruckList729, TruckList], [TruckList729, List], [TruckList729, Object], [TruckList848, TruckList], [TruckList848, List], [TruckList848, Object], [TruckList400, TruckList], [TruckList400, List], [TruckList400, Object], [EString_1, EString], [TruckList610, TruckList], [TruckList610, List], [TruckList610, Object], [EString_3, EString], [3, EInt], [5, EInt], [1, EInt], [EngineedVehicle284, Vehicle], [EngineedVehicle284, EngineedVehicle], [EngineedVehicle284, Object], [FORD F−150 XLT, EString], [CarList493, CarList], [CarList493, List], [CarList493, Object], [4, EInt], [2, EInt], [List_, 5, List], [List_, 5, Object], [Memory46, Memory]], [[EnginedVehicle84, Vehicle], [EnginedVehicle84, EngineedVehicle], [EngineedVehicle84, Vehicle], [EnginedVehicle84, EngineedVehicle], [EngineedVehicle84, Object], [TruckList729, TruckList], [TruckList729, List], [TruckList729, Object], [Vehicle_3, EnginedVehicle], [Vehicle_3, Object], [Vehicle_3, Object], [Vehicle_3, NonEnginedVehicle], [TruckList848, TruckList], [TruckList848, List], [TruckList848, Object], [TruckList400, TruckList], [TruckList400, List], [TruckList400, Object], [EString_1, EString], [TruckList610, TruckList], [TruckList610, List], [TruckList610, Object], [EString_3, EString], [3, EInt], [5, EInt], [1, EInt], [List_, 6, Nil], [List_, 6, TruckList], [List_, 6, CarList], [List_, 6, List], [List_, 6, Object], [List_, 6, BicycleList], [EngineedVehicle284, Vehicle], [EngineedVehicle284, EngineedVehicle], [EngineedVehicle284, Object], [FORD F−150 XLT, EString], [String_0, EString], [CarList493, CarList], [CarList493, List], [CarList493, Object], [4, EInt], [2, EInt], [List_, 5, Nil], [List_, 5, TruckList], [List_, 5, CarList], [List_, 5, List], [List_, 5, Object], [List_, 5, BicycleList], [Memory46, Memory]]}
AlloyInEcore: Embedding of First-Order Relational Logic into ... ESEC/FSE ’18, November 4–9, 2018, Lake Buena Vista, FL, USA

Listing 4: Lower and Upper Bounds for User’s Binary Relations

```
38  cdr: [[[TruckList729, TruckList400], [TruckList848, TruckList400], [TruckList610, TruckList729],
          [CarList493, TruckList848]], [[TruckList729, TruckList729], [TruckList729, TruckList848],
          [TruckList729, TruckList400], [TruckList729, TruckList610], [TruckList729, List_6],
          [TruckList729, CarList493], [TruckList729, List_5], [TruckList848, TruckList729],
          [TruckList848, TruckList848], [TruckList848, TruckList400], [TruckList848, TruckList610],
          [TruckList848, List_6], [TruckList848, CarList493], [TruckList848, List_5], [TruckList400,
          TruckList729], [TruckList400, TruckList848], [TruckList400, TruckList400], [TruckList400,
          TruckList610], [TruckList400, TruckList610], [List_6], [CarList493, TruckList400], [TruckList400,
          TruckList400], [TruckList400, TruckList400], [TruckList400, TruckList610], [TruckList610,
          TruckList610], [TruckList610, TruckList610], [TruckList610, List_5], [TruckList610, CarList493],
          [TruckList610, List_5], [List_6, TruckList729], [List_6, TruckList848], [List_6, TruckList400],
          [List_6, TruckList610], [List_6, List_6], [CarList493, List_5], [CarList493, TruckList729],
          [CarList493, TruckList848], [CarList493, TruckList400], [CarList493, CarList493], [List_5,
          TruckList729], [List_5, TruckList848], [List_5, TruckList400], [List_5, TruckList610], [List_5,
          List_6], [List_5, CarList493], [List_5, List_5]]
39  car: [[[TruckList729, EngineVehicle84], [TruckList848, EngineVehicle84], [TruckList400,
          EngineVehicle84]], [TruckList610, EngineVehicle84], [CarList493, EngineVehicle84]],
          [[TruckList729, EngineVehicle84], [TruckList729, EngineVehicle84], [TruckList729,
          EngineVehicle84], [TruckList729, EngineVehicle84], [TruckList729, Vehicle_3], [TruckList729,
          EngineVehicle84], [TruckList848, EngineVehicle84], [TruckList848, EngineVehicle84],
          [TruckList848, EngineVehicle84], [TruckList848, Vehicle_3], [TruckList848, EngineVehicle84],
          [TruckList400, EngineVehicle84], [TruckList400, EngineVehicle84], [TruckList400, Vehicle_3],
          [TruckList400, EngineVehicle84], [TruckList610, EngineVehicle84], [TruckList610,
          EngineVehicle84], [TruckList610, EngineVehicle84], [TruckList610, EngineVehicle84], [List_6,
          EngineVehicle84], [List_6, EngineVehicle84], [List_6, Vehicle_3], [List_6, EngineVehicle84],
          [CarList493, EngineVehicle84], [CarList493, EngineVehicle84], [CarList493, EngineVehicle84],
          [CarList493, Vehicle_3], [CarList493, EngineVehicle84], [List_5, EngineVehicle84], [List_5,
          EngineVehicle84], [List_5, Vehicle_3], [List_5, EngineVehicle84]]
40  eq: [[], [[TruckList729, TruckList729], [TruckList729, TruckList848], [TruckList729, TruckList400],
          [TruckList729, TruckList610], [TruckList729, TruckList610], [List_6], [TruckList729, CarList493],
          [TruckList729, List_5], [TruckList848, TruckList729], [TruckList848, TruckList848],
          [TruckList848, TruckList400], [TruckList848, TruckList610], [TruckList848, List_6],
          [TruckList848, CarList493], [TruckList848, List_5], [TruckList400, TruckList729],
          [TruckList400, TruckList848], [TruckList400, TruckList400], [TruckList400, TruckList610],
          [TruckList400, List_6], [TruckList400, CarList493], [TruckList400, List_5], [TruckList400,
          TruckList729], [TruckList610, TruckList848], [TruckList610, TruckList400], [TruckList610,
          TruckList610], [List_6, TruckList729], [List_6, TruckList848], [List_6, TruckList400],
          [List_6, TruckList610], [List_6, List_6], [CarList493, List_5], [CarList493, TruckList729],
          [CarList493, TruckList848], [CarList493, TruckList400], [CarList493, TruckList610],
          [CarList493, List_6], [CarList493, CarList493], [CarList493, List_5], [List_5, TruckList729],
          [List_5, TruckList848], [List_5, TruckList400], [List_5, TruckList610], [List_5, List_6],
          [List_5, CarList493], [List_5, List_5]]
41  name: [[[EngineVehicle84, EString_3], [EngineVehicle184, EString_1]], [[EngineVehicle84,
          EString_1], [EngineVehicle84, EString_3], [EngineVehicle84, FORD F-150 XLT],
          [EngineVehicle84, String_0], [EngineVehicle184, EString_1], [EngineVehicle184, EString_3],
          [EngineVehicle184, FORD F-150 XLT], [EngineVehicle184, String_0], [Vehicle_3, EString_1],
          [Vehicle_3, EString_3], [FORD F-150 XLT], [Vehicle_3, String_0], [EngineVehicle284,
          EString_1], [EngineVehicle284, EString_3], [EngineVehicle284, FORD F-150 XLT],
          [EngineVehicle284, String_0]]
42  vehicles: [[[Memory46, EngineVehicle84], [Memory46, EngineVehicle84], [Memory46,
          EngineVehicle84]], [[[Memory46, EngineVehicle84], [Memory46, EngineVehicle84], [Memory46,
          EngineVehicle84]]]
Listing 5: Generated Formulas

```
43 List = (List <? extends Vehicle > + Nil)
44 (all n: univ | Class<Nil> in (n.class) => n in Nil)
45 no (Memory & Object)
46 Vehicle = (EnginedVehicle + NonEnginedVehicle)
47 (all e: univ | Class<EnginedVehicle> in (e.class) => e in EnginedVehicle)
48 (Memory.(lists + vehicles)) = (Memory + Object)
49 (all m: Memory.lists | lone (m.-(lists + vehicles)))
50 no (NonEnginedVehicle & EnginedVehicle)
51 (all l: List | lone (l.car))
52 Object = (Vehicle + List)
53 cdr in ((List<EnginedVehicle> -> List<EnginedVehicle>) + (List<NonEnginedVehicle> -> List<NonEnginedVehicle>))
54 (all l: List | lone (l.cdr))
55 (all a: Vehicle, b: Vehicle | !(a = b) => !((a.name) = (b.name)))
56 no (Nil.cdr)
57 (all o: univ | Class<Object> in (o.class) => o in Object)
58 no (CarList & Nil<EnginedVehicle>)
59 List<? extends Vehicle> = (List<EnginedVehicle> + List<NonEnginedVehicle>)
60 (all b: univ | Class<BicycleList> in (b.class) => b in BicycleList)
61 (all m: Memory.vehicles | lone (m.- (lists + vehicles)))
62 no (TruckList & Nil<EnginedVehicle>)
63 List<EnginedVehicle> = CarList + TruckList + Nil<EnginedVehicle>
64 one Nil
65 no (BicycleList & Nil<NonEnginedVehicle>)
66 no (List<EnginedVehicle> & List<NonEnginedVehicle>)
67 no (Nil.car)
68 all a: List, b: List | a in b.eq <= (a.car = b.car and a.cdr) in ((b.cdr).eq and a.class = b.class)
69 (all l: univ | Class<List> in (l.class) <= l in List)
70 (all l: List | 1 in (l.eq))
71 some vehicles
72 (all l: List - Nil | some (l.cdr) and some (l.car))
73 Nil = (Nil<NonEnginedVehicle> + Nil<EnginedVehicle>)
74 (all e: univ | Class<Elnt> in (e.class) <= e in Elnt)
75 eq in ((List<NonEnginedVehicle> -> List<NonEnginedVehicle>) + (List<EnginedVehicle> -> List<Engine...)
76 (all t: univ | Class<TruckList> in (t.class) <= t in TruckList)
77 (all l: List | !(1 in (l.cdr)))
78 lists in (Memory -> List<? extends Vehicle>)
79 no (Vehicle & List)
80 one Memory
81 vehicles in (Memory -> Vehicle)
82 (all v: univ | Class<Vehicle> in (v.class) <= v in Vehicle)
83 (all c: univ | Class<CarList> in (c.class) <= c in CarList)
84 List<NonEnginedVehicle> = (BicycleList + Nil<NonEnginedVehicle>)
85 (all l: List | Nil in (l.cdr))
86 (all v: Object - List | some (v.car))
87 no (Nil<EnginedVehicle> & Nil<NonEnginedVehicle>)
88 (all n: univ | Class<NonEnginedVehicle> in (n.class) <= n in NonEnginedVehicle)
89 no (TruckList & CarList)
90 some lists
91 (all e: univ | Class<EString> in (e.class) <= e in EString)
92 car in ((List<NonEnginedVehicle> -> NonEnginedVehicle) + (List<EnginedVehicle> -> EnginedVehicle))
93 name in (Vehicle -> EString)
94 (some v: Vehicle | all v': Vehicle | (v'.name) = "FORD F-150 XLT" <= v' = v)
95 (all m: univ | Class<Memory> in (m.class) <= m in Memory)
96 (all v: Vehicle | one (v.name))
```
Listing 6: The Outcome and the statistics of the reasoning process

---OUTCOME---
SATISFIABLE

---STATS---
p cnf 2646 6717
primary variables: 178
translation time: 149 ms
solving time: 10 ms

Listing 7: The Generated Model (Unary Relations)

---MODEL---
Nil <V> : List_5
List <E> : TruckList610, TruckList400, List_5, CarList493, List_6, TruckList848, TruckList729
Class <Nil> : Nil
CarList : CarList493
Class <EInt> : EInt
Vehicle : EnginedVehicle184, EnginedVehicle284, EnginedVehicle84
List <EnginedVehicle> : TruckList610, TruckList400, List_5, CarList493, List_6, TruckList848, TruckList729
Class <Vehicle> : Vehicle
NonEnginedVehicle :
Class <Memory> : Memory
Object : TruckList610, TruckList400, List_5, EnginedVehicle184, EnginedVehicle284, CarList493, List_6, EnginedVehicle84, TruckList848, TruckList729
TruckList : TruckList610, TruckList400, List_6, TruckList848, TruckList729
Nil <NonEnginedVehicle> :
EString : EString_1, EString_3, FORD F-150 XLT
EInt : 2, 1, 4, 3, 5
Class <List> : List
Class <Object> : Object
Class : Nil, EString, NonEnginedVehicle, CarList, Vehicle, BicycleList, EInt, List, Memory, EnginedVehicle, Object, TruckList
EnginedVehicle : EnginedVehicle184, EnginedVehicle284, EnginedVehicle84
Class <CarList> : CarList
Class <TruckList> : TruckList
List <V extends Vehicle> : TruckList610, TruckList400, List_5, CarList493, List_6, TruckList848, TruckList729
Nil <EnginedVehicle> : List_5
List <NonEnginedVehicle> :
BicycleList :
"FORD F-150 XLT": FORD F-150 XLT
Class <EString> : EString
Class <NonEnginedVehicle> : NonEnginedVehicle
Class <EnginedVehicle> : EnginedVehicle
Class <BicycleList> : BicycleList
Memory : Memory46
Listing 8: The Generated Model (Binary Relations)

137  eq (TruckList610, TruckList610)
138  eq (TruckList848, TruckList729)
139  eq (TruckList729, TruckList729)
140  eq (TruckList400, TruckList400)
141  eq (TruckList729, TruckList848)
142  eq (List_5, List_5)
143  eq (CarList493, CarList493)
144  eq (List_6, List_6)
145  eq (TruckList848, TruckList848)
146  car (TruckList400, EnginedVehicle184)
147  car (List_6, EnginedVehicle284)
148  car (TruckList610, EnginedVehicle184)
149  car (CarList493, EnginedVehicle184)
150  car (TruckList848, EnginedVehicle84)
151  car (TruckList729, EnginedVehicle84)
152  vehicles (Memory46, EnginedVehicle184)
153  vehicles (Memory46, EnginedVehicle84)
154  vehicles (Memory46, EnginedVehicle284)
155  lists (Memory46, CarList493)
156  lists (Memory46, List_5)
157  lists (Memory46, List_6)
158  lists (Memory46, TruckList610)
159  lists (Memory46, TruckList400)
160  lists (Memory46, TruckList729)
161  lists (Memory46, TruckList848)
162  name (EnginedVehicle284, FORD F-150 XLT)
163  name (EnginedVehicle184, EString_1)
164  name (EnginedVehicle84, EString_3)
165  cdr (List_6, List_5)
166  cdr (TruckList848, TruckList400)
167  cdr (TruckList729, TruckList400)
168  cdr (TruckList400, List_6)
169  cdr (TruckList610, TruckList729)
170  cdr (CarList493, TruckList848)