PROJECT PROPOSAL- UML MODELLING IN PROLOG

Anudeep Medishetti - 112675646

November 20, 2019

1 ABSTRACT

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. UML provides a number of diagrams to describe the modeling system from different perspectives, which contain overlapping information about the systems. The paper "**Verifying Static Aspects of UML models using Prolog**" describes various types of consistency UML class diagrams has to maintain and also proposes an approach for consistency checking of UML class diagrams and object diagrams using PROLOG. In this proposal, I defined the methods to extend the details mentioned in the paper and implement them to run UML verification in PROLOG.

2 INTRODUCTION

PAPER OUTLINE link

As mentioned above, the paper "Verifying Static Aspects of UML models using Prolog" describes about static aspects of UML Modelling and how to verify them.
The following are the inferences from the paper.

1. Consistency checking in UML models is essential to ensure system stability and also to refrain design errors flow to the coding phase.
2. There are various types of consistency checks namely vertical consistency, horizontal consistency, semantic consistency and syntactical consistency.
3. The paper checks horizontal, syntactical and semantic consistency.
4. All these consistency checks can be performed in PROLOG, since the problem of inconsistencies in UML models can be converted into problem of contradictions in PROLOG.
5. We use Essential Meta Object Facility, a subset of Meta Object Facility(MOF) to define the meta-models.
6. The syntax of class diagrams is a structure M={class, attribute, operation, association, rolename, multiplicity}, where details on each item is described below.

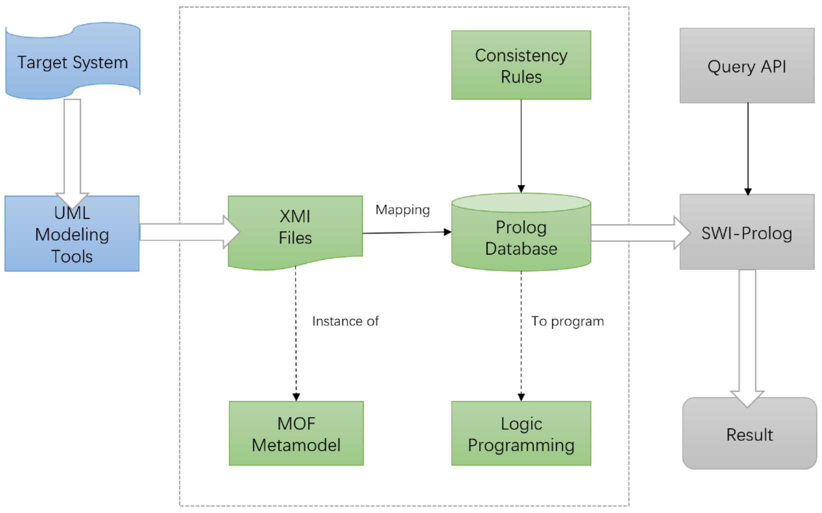

Figure 1: Approach

Components of EMOF Metamodel converted to PROLOG

As mentioned above, to start with this verification, we have to store all the components of EMOF model in PROLOG. For example, class in EMOF metamodel is defined as class/3 clause in PROLOG. It includes an identifier, name and a boolean to specify whether it is abstract.

All other components are similarly defined as shown below.

1. ***class(classid, classname, isAbstract)***
2. ***attribute(attrid, attrname, attrtype, classid)***
3. ***operation(opid, opname, [parameters], classid)***
4. ***parameter(pid, pname, ptype)***
5. ***Association(associd, classA, classB, assocType)***
6. ***rolename(roleid, nameA, nameB, associd)***
7. ***multiplicity(multiid, classid, lowval, upval, associd)***
8. ***generalization(genid, subid, superid)***

Prolog functions for some of the above clauses are defined in the paper.

Object Diagrams to PROLOG

Object diagrams, as known, show a complete/partial view of the structure of a modelled system at a specific time. And these diagrams focuses on the set of objects and attribute values, and the links between these objects at a particular time.

$(M) = \{object; link; attrval\}$

object $->$ set of objects

links $->$ set of links connecting objects.

arrival $->$ set of functions assigning attribute values to each object.

And **domain** of a class is defined as a set of object identifiers that are the instances of the class.

$domain(c) = \bigcup \{\ objects(c') \mid c' \ \varepsilon \ class \wedge c' \prec c.\}$

Validation as per the paper

The paper describes various methods for UML model - static aspect consistency checking. Some of them are.

1. Classes and associations should have unique names.
2. Names of the attributes are unique to one class.
3. There are no direct or indirect cycles in the generalization relationship.
4. If class c1 is a sub class of class c2, the domain of c1 is a subset of the domain of c2. The set of objects connected to its subclasses should also be disjoint

$$\begin{array}{l} \forall c_1, c_2 \in class, c_1 < c_2 \rightarrow domain\,(c_1) \subseteq domain\,(e_2)\,. \\ \forall \epsilon class, domain(c) = \bigcup_{c_i \in sub(c)} domain(c_i) \end{array} \tag{1}$$

3 NEXT STEPS

I am planning to implement the equivalent of above shown approach in XSB, by following the steps as mentioned below.

1. Create EMOF equivalent clauses in XSB.
2. For each component in EMOF models, I will try to define all the rules and verify each rule based on user input.
3. I will try to implement an interactive system by "Reading the input from a user for UML model diagrams, by giving certain instructions."
4. I will try to read the input and check, whether it satisfies the validation criteria for model consistency, based on the rules defined in clauses.
5. To validate, whether the PROLOG implementation for verifying static UML aspects is correct, I will try to find some accepted external methodology for UML model consistency checking.
6. After trying out to implement the static aspect validation in PROLOG, we can even try to extend the model to verify the dynamic aspects for the same.