Natural Semantics Based Tools for Semantic Web With Application to Product Models

Thesis Proposal

Adrian Pop[1]
Programming Environments Laboratory (PELAB)
Department of Computer and Information Science (IDA)
Linköping University (LiU)

1 Background

Our research is a mix of several computer science areas:

- Compilers and Debuggers for Programming Languages
- Semantic Web and Description Logics
- Integrated Product Design using Modeling and Simulation with Modelica

We apply results from programming languages in the Semantic Web area and we would like to apply the languages and tools of the Semantic Web area in the Product Design and Modeling/Simulation area. The thesis research is part of several research projects, which are based in different research areas.

In the following we will briefly describe each area in no particular order. We emphasize on the connection between our research and the specific area.

1.1 Integrated Product Design

In the area of model-driven product design using modeling and simulation we focus on the integration of Modelica language [6, 20] with conceptual modeling [15] tools. The research is part of the *System Engineering & Computational System Design (SECD) ProViking project*.

Modelica is an object-oriented language used for modeling of large and heterogeneous physical systems. For modeling with Modelica, commercial software products such as MathModelica [5] or Dymola [3] have been developed. Also open-source projects like the OpenModelica Project [21] are available.

Designing products is a complex process. Highly integrated tools are essential to help a designer to work efficiently. Designing a product includes early design phase product concept modeling and evaluation, physical modeling and simulation and finally the physical product realization. For physical modeling and simulation available tools provide advanced functionality. However, the integration of such tools with conceptual modeling tools is a resource consuming process that today requires large amounts of manual, and error prone work. Also, the number of physical models available to the designer in the product concept design phase is typically quite large. This has an impact on the selection of the best set of component choices for detailed product concept simulation.

To address these issues we have developed a framework for product development based on an XML meta-model [30] of Modelica and its representation in an Modelica Database [31].

To provide flexibility of the product design framework we addressed the composition and transformation of Modelica models in the COMPOST framework [32].

1.2 Compilers and Debuggers for Programming Languages

From this area we consider formal semantics of programming languages. In particular the work on Natural Semantics [24, 29], which is a formalism for specifying many aspects of programming languages i.e. type systems, dynamic semantics,

[1] URL: http://www.ida.liu.se/~adrpo
Email: adrpo@ida.liu.se

translational semantics, static semantics, etc. Natural Semantics is an operational semantics derived from the Plotkin [22] structural operational semantics combined with the sequent calculus for natural deduction.

Relational Meta-Language (RML) [28] is a practical language for writing natural semantics specifications. The RML language is compiled to highly efficient C code by the rml2c compiler. In this way large parts of a compiler can be automatically generated from their Natural Semantics specifications.

RML was successfully used in our laboratory for specifying and generating compilers from Natural Semantics for Java, Modelica and other languages.

However, in this thesis we are interested in applying Natural Semantics to different languages (OWL Lite/DL). Modification of the existing RML language and compiler may be needed.

In this area our research work extended the Relational Meta-Language (RML) tool [28] with tracing and debugging facilities.

As a crash course in Natural Semantics and Relational Meta-Language (RML) we give an example of a small expression (Exp) language and its realization in Natural Semantics and RML.

A specification in Natural Semantics has two parts: declaration of syntactic and semantic objects involved followed by groups of inference rules. In our example language we have expressions build from numbers. The abstract syntax of this language is declared in the following way:

integers:

$v \in Int$

expressions:

$$e \in Exp ::= v \mid e1 + e2 \mid e1 - e2 \mid e1 * e2 \mid e1 / e2 \mid -e$$

The inference rules for our language are bundled together in a judgment $e => v$ in the following way:

$$(1)\quad v => v$$

$$(2)\quad \frac{e1 => v1 \;\; e2 => v2 \;\; v3 = v1 + v2}{e1 + e2 => v3}$$

In RML, the Natural Semantics specification presented above can be represented by the following source code:

```
(* file exp1.rml *)
module exp1:
(* Abstract syntax of the language Exp1 *)
  datatype Exp   =  INTconst of int
                 |  ADDop    of Exp * Exp
                 |  SUBop    of Exp * Exp
                 |  MULop    of Exp * Exp
                 |  DIVop    of Exp * Exp
                 |  NEGop    of Exp

  relation eval:  Exp => int
end
(* Evaluation semantics of Exp1 *)
relation eval: Exp => int  =

 axiom eval( INTconst(ival) )
    => ival   (* eval of an integer node *)
              (* is the integer itself *)

 (* Evaluation of an addition node PLUSop
  * is v3, if v3 is the result of adding
  * the evaluated results of its children
  * e1 and e2. Subtraction, multiplication,
  * division operators have similar specs.
  *)

 rule  eval(e1) => v1  &
       eval(e2) => v2  &
       int_add(v1,v2) => v3
       -----------------------
        eval( ADDop(e1,e2) ) => v3
   ...
end (* eval *)
```

The proof-theoretic interpretation is assigned to this specification. We interpret inference rules as recipes for constructing proofs. If we wish to prove that there is a value v such that $1 + 2 => v$ holds for this specification. To prove this proposition we need an inference rule that has a conclusion, which can be instantiated (matched) to the proposition. The only proposition that matches is proposition 2.

$$\frac{1 => v1 \;\; 2 => v2 \;\; v = v1 + v2}{1 + 2 => v}$$

To prove further we need to apply proposition 1 several times and we reach the conclusion.

1.3 Semantic Web and Description Logics

Semantic Web [10, 11] is a new wave of research that provides a common framework that allows data to be shared and reused between applications. Semantic Web is trying to give the data on the web a well-defined meaning in order to allow both machines and humans to process it [16]. In order to

achieve such goal the Semantic Web has a layered architecture as in Figure 1.

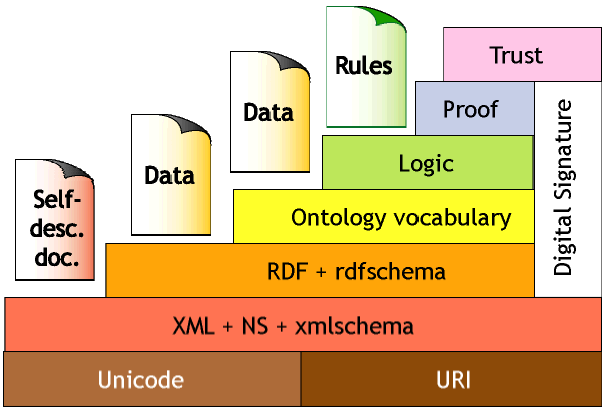

Figure 1: The Semantic Web layering

At the bottom in top of Unicode and Uniform Resource Identifiers (URI) is XML, namespaces (NS) and XML-Schema. XML specifies a term list with no relations. On top of XML comes Resource Description Framework (RDF) [9] language to define a simple datamodel for objects the relations between them. RDF Vocabulary Description Language (RDFS or RDF schema) [8] is a vocabulary for describing properties and classes of RDF resources. The Ontology layer uses languages like the Web Ontology Language OWL [13] to add more vocabulary for describing properties and classes, typing of properties, relations between classes, cardinality constraints, etc. OWL consists of three sublanguages that provide increased expressiveness with different computational properties [14]:

- *OWL Lite* provides classification hierarchies and very simple constraints
- *OWL DL* provides the maximum possible expressiveness that still has computational completeness and decidability. OWL DL has a correspondence with Description Logics [1].
- *OWL Full* offers maximum expressiveness with no computational guarantees.

On top of these ontology languages rules and logic are present to add application behavior.

Description Logics [1] are a family of formalisms for representing and reasoning with knowledge. Description Logics is used to represent data and knowledge of the relations between individual objects and their grouping into classes. The Description Logics *reasoners* [7, 23] make deductions from a knowledge base of such description of classes and individuals. These deductions are targeted to detect inconsistencies, to classify (organize) the classes into sub-class hierarchies and to classify individuals under appropriate concepts. The literature on Description Logics and Semantic Web is huge and we do not wish to enter into more detail here. We will provide more insight in the Related Work section and also in the Methodology section on what work we are based upon and how do we extend this work.

We are involved in this area with two projects:

1. *Semantic Web For Products (SWEB)*. In this project, our contribution will provide pragmatic techniques and methodologies for consistency checking of ontologies and their corresponding documents.
2. *European Network of Excellence on "Reasoning on the Web with Rules and Semantics (REWERSE)"*. Our contribution in this project comprises composition and typing of rule-based languages for the web.

2 Problem Formulation

The focus of the thesis as *short-term goal* is to provide lightweight, practical and efficient tool implementations for Semantic Web languages. The toolbox will include tools for consistency checking of both ontologies and the data sets (documents) that are conform to such ontologies.

The *long-term goal* of this thesis is to integrate all these technologies into a framework for model-driven product design and development.

2.1 Detailed problem description

Short-term goal of the thesis. In the Semantic Web for Products (SWEBPROD) project we employ Semantic Web technologies for product development. A key issue in this project is the development of efficient lightweight tools for ontology checking and processing. Our approach in building such tools is to develop a Natural Semantics specification of OWL Lite/DL (Description Logics) in the Relational Meta-Language (RML). This specification will be then

compiled to executable format using the Relational Meta-Language (RML) compiler. Also, such specifications can be debugged using the existing RML debugger.

A benefit of representing various reasoning tasks using Natural Semantics is to have a clear proof-theoretic view of how and from where certain knowledge was inferred from the existing knowledge base. The proof tree of each reasoning task can be made available to the users by using the output from the RML debugger.

Long-term goal of the thesis. The future research in this thesis will focus on integrating Semantic Web technologies with Product Design and also with Modeling and Simulation tools. This will facilitate model interchange between various modeling and simulation tools and between product design tools. Also, the use of already defined vocabularies (vocabularies) for physical, mathematical, biological and chemical domains could be use when designing models.

3 Relevance

The main importance of providing such tools is to automate and integrate methodologies and techniques from various area of computer science.

In the area of Semantic Web having tools based on Natural Semantics specification will provide a framework for:

- Experimentation with different semantics and different algorithms for specific reasoning tasks
- Proof-theoretic (deductive) explanations for the variety of inferences performed by the reasoning tasks

The benefits of using Semantic Web technologies in product design tools or in modeling and simulation tools are several:

- Support for library designers (classification of classes, coherence checking, etc)
- Knowledge management through the design phases (easy accessible data in machine accessible form can be made available or can be used in the various design phases of the product development)
- Declarative query languages can be used to search for specific models needed in the conceptual design of products.
- Software information systems (SIS) could be built to facilitate model understanding and information finding.

4 Related Work

As related work in the area of Description Logic and Semantic Web we use and extend several research results: The idea of having a proof explanation of the reasoning tasks has its root in the work of McGuinness and Borgida [25-27]. The implementation in RML of the Natural Semantics specifications of Description Logics will be adapted from [17]. Also, in order to proof our concept we will perform comparisons with existing OWL implementations [7].

There are few systems implemented that compile or interpret Natural Semantics. One of these systems is Centaur with its implementation of Natural Semantics called Typol [19]. This system is translating the inference rules to Prolog and is several order of magnitude slower than RML [28].

In the area of Product Design our framework has similarities with Schemebuilder [18]. However our work is more oriented towards the design of advanced complex products that require systems engineering, and targeted to the simulation modeling language Modelica, which to our knowledge has more expressive power in the areas of our research, than many tools for systems engineering that are currently widely used. For details on Systems Engineering, see [4].

Our RML-debugger employs source-code instrumentation, which is the only portable and elegant alternative for an optimizing compiler like RML. Similar approach is used in debugging Standard ML [33].

5 Results, research contributions

In this section we present our research contributions as preliminary results and the expected results for the future.

5.1 Preliminary results

The preliminary result consists of several articles, which we briefly present in the next sections. Also,

we describe work not yet published and work in progress.

5.1.1 ModelicaXML as an alternative representation for Modelica

Adrian Pop, Peter Fritzson. *ModelicaXML: A Modelica XML representation with Applications*, in *International Modelica Conference*, 3-4 November, 2003, Linköping, Sweden.

In this paper we present a Modelica meta-model. This meta-model is an alternative representation of the Modelica language in XML format. ModelicaXML is the structure of the Modelica language after the parsing phase. We also have a first look at using Semantic Web languages to express some of the Modelica semantics.

This representation provides more functionality than a typical C++ class library implementing an AST representation of Modelica:

- Declarative query languages for XML can be used to query the XML representation.
- The XML representation can be accessed via standard interfaces like Document Object Model (DOM) [2] from practically any programming language.

The usages of the ModelicaXML representation for Modelica models, combined with the power of general XML tools, ease the implementation of tasks like:

- Analysis of Modelica programs (model checkers and validators).
- Pretty printing (un-parsing).
- Translation between Modelica and other modeling languages (interchange).
- Query and transformation of Modelica models.

Although ModelicaXML captures the structured representation of Modelica source code, the semantics of the Modelica language cannot be expressed without implementing specific XML-based tools. To address this issue we have investigated the benefits of using languages developed in the Semantic Web Community [11]. We believe that using such technology for Modelica models would enable several applications in the future:

- Models could be automatically translated between modeling tools.
- Models could become autonomous (active documents) if they are packaged together with the operational semantics from the compiler, and therefore, they could be simulated in a normal browser.
- Software information systems (SIS) could more easily be constructed for Modelica, facilitating model understanding and information finding. We consider adapting the approach described in [34] to construct such a SIS for Modelica.
- Model consistency could be checked using already implemented Description Logic (DL) reasoners (i.e. Fact or Fact++ [23]) or our implementation. Using our implementation will give us the freedom to experiment with more language constructs and constraints.
- Certain models could be translated to and from the Unified Modeling Language (UML) [12].

This preliminary work fits perfectly in the thesis frame, being the middleware of our Product Design framework.

5.1.2 Composition and transformation of Modelica by using its ModelicaXML meta-model

Adrian Pop, Ilie Savga, Uwe Assmann and Peter Fritzson. *Composition of XML dialects: A ModelicaXML case study*, in *Software Composition Workshop 2004, affiliated with ETAPS 2004*,3 April, 2004, Barcelona.

This paper investigates how software composition and transformation can be applied to domain specific languages used today in modeling and simulation of physical systems. More specifically, we address the composition and transformation of the Modelica language. The composition targets the ModelicaXML dialect, which is the XML representation of the Modelica language. By extending the COMPOST concrete composition layer with a component model for Modelica, we provide composition and transformation of Modelica.

Transformation and composition of Modelica models allows easy, automatic change of models to fit context. Also, entire systems can be

automatically generated, configured and simulated. Such result gives the framework for product design a high flexibility and scalability.

5.1.3 An Integrated framework for model-driven product design and development using Modelica

Adrian Pop, Olof Johansson and Peter Fritzson. *An Integrated Framework for Model-Driven Product Design and Development using Modelica*, in *Conference on Simulation and Modeling*,23-24 Septemeber, 2004, Copenhagen, submited.

This paper presents our work in the area of model-driven product development processes. The focus is on the integration of product design tools with modeling and simulation tools. The goal is to provide automatic generation of models from product specifications using a highly integrated set of tools. Also, we provide the designer with the possibility of selecting the best design choice, verified through (automatic) simulation of different implementation alternatives of the same product model. To have a flexible interaction among various tools of the framework an XML representation of the Modelica modeling language called ModelicaXML is used. For efficient search in a large base of simulation models the Modelica Database was designed.

As future work we want to explore the use of ontologies for product concept design and for the classification of the available component libraries. For this purpose the languages developed by the Semantic Web [11] community will be used. Research efforts based on this standard are integrating experience of many promising research areas, for instance declarative rules, which still lack a vendor neutral exchange formats for industrial applications. The semantic web standard lacks important functionality for quality assurance and other necessary functionality, which today is implemented in commercial products, but will open up for sharing of important research results with industry in collaborative environments.

This framework is our test-bed for experimenting novel techniques and methodologies in conceptual design.

5.1.4 A debugger for Relational Meta-Language

We have seen in the first section of this thesis proposal a small example of RML specification.

To provide debugging, the RML debugger takes such specification (actually the AST of RML) and adds instrumentation code in the left and right hand side of a goal (term). This instrumentation code is responsible for stopping is a breakpoint is set and for displaying bounded variables in the goal about to be executed. In this way, complete proof trees of the inference rules that fired can be collected. Here is not the place to enter into more detail. We intend to publish a paper with a more detailed view on the RML debugger and the techniques/methodologies used in the implementation.

5.2 Expected results

Experimenting with existing and novel reasoning techniques will improve the process management sector by providing consistency checking, searching and information retrieval facilities, composition and interoperability, traceability and comparison for all kind of documents.

As a long-term result we foresee the integration of reasoning tools in a knowledge-based platform for product development.

6 Methodology

Strategy of approaching the problem:
In order to automatically build the tools that will perform reasoning tasks on ontologies we have to have to look into what reasoning services are available in Description Logics (OWL Lite/DL) systems. The basic reasoning tasks that a Description Logics (OWL Lite/DL) system supports are: subsumtion, incoherence checking, disjointness, equivalence, classification, instance checking, retrieval, and knowledge base consistency. All these reasoning tasks can be represented as rules in Natural Semantics as described in [17, 25, 26]. We expect to encounter some difficulties in adapting such Natural Semantics specification to RML. To overcome such difficulties the RML compiler will be adapted.

The RML debugger will be used to output traces of proof-trees of the description logics (actually OWL Lite/DL) reasoning tasks.

The whole idea of having our own reasoning tools is to provide more constraints than the actual OWL languages encompass, by implementing them directly in Natural Semantics.

Validity of the conclusion:
To validate our results we will compare our approach with existing approaches [7, 23]. Also, the tools will be validated in use-cases provided by industrial partners involved in the research projects.

7 Project Plan

In the first part of our research we focused on developing an alternative meta-model for Modelica (ModelicaXML). Also, we used this meta-model to experiment with transformation and composition of Modelica. As a case study we integrated the ModelicaXML meta-model and Modelica Database model into a framework for product design and development.

The next part of the research will have a slight change of focus towards Semantic Web. We need more insight and knowledge in the area to build useful tools and methodologies that later can be used in the research projects we are involved.

7.1 Detailed Plan

Date	Task
2002-01	The beginning of PhD studies
2003-08	The ModelicaXML meta-model for Modelica (paper accepted)
2004-03	Composition and transformation of XML dialects: A ModelicaXML case study (paper accepted)
2004-05	Release of the first version of RML debugger (work in progress)
2004-05	An integrated framework for model-driven product design and development using Modelica (paper submitted)
2004-06	RML prototype of basic reasoning tasks in OWL Lite
2004-08	Evaluation of the RML prototype and improvements (also improvements of RML debugger based on feedback from the OpenModelica project)
2004-10	Article on using RML to perform reasoning
2004-12	Lic. thesis
2005-03	Integration of our toolbox with the work of the partners involved in current research projects.
2005-06	Research on novel methodologies to improve product design.
2006-05	Experimenting with these new methodologies in our framework for product design.
2007-01	Thesis

8 References

1. Description Logics Website. *Description Logics*, http://dl.kr.org/.
2. World Wide Web Consortium (W3C). *Document Object Model (DOM)*, http://www.w3.org/DOM/.
3. Dynasim. *Dymola*, http://www.dynasim.se/.
4. INCOSE. *International Council on System Engineering*, http://www.incose.org.
5. MathCore. *MathModelica*, http://www.mathcore.se/.
6. *Modelica: A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification version 2.1*, Modelica Association, 2003,
7. World Wide Web Consortium (W3C). *OWL Implementations*, http://www.w3.org/2001/sw/WebOnt/impls.
8. World Wide Web Consortium (W3C). *RDF Vocabulary Description Language (RDFS/RDF-Schema)*, http://www.w3.org/TR/rdf-schema/.
9. Word Wide Web Consortium (W3C). *Resource Description Framework (RDF)*, http://www.w3c.org/RDF.
10. World Wide Web Consortium (W3C). *Semantic Web*, http://www.w3.org/2001/sw/.
11. *Semantic Web Community Portal*, http://www.semanticweb.org/.
12. UML Website. *Unified Modeling Language (UML)*, http://www.uml.org/.
13. Word Wide Web Consortium (W3C). *Web Ontology Language (OWL)*,

http://www.w3.org/TR/2003/CR-owl-features-20030818/.
14. World Wide Web Consortium (W3C). *Web Ontology Language (OWL) Overview*, http://www.w3.org/TR/owl-features/.
15. Mogens Myrup Andreasen. *Machine Design Methods Based on a Systematic Approach (Syntesemetoder på systemgrundlag)*, Lund Technical University, Lund, Sweden, 1980.
16. Tim Berners-Lee, James Hendler and Ora Lassila. *The Semantic Web*, *Scientific American*, May 2001.
17. Alex Borgida. *From Type Systems to Knowledge Representation: Natural Semantics Specifications for Description Logics.,* International Journal of Intelligent and Cooperative Information Systems, March 1992: p. 93-126.
18. R.H. Bracewell, D.A.Bradley. *Schemebuilder, A Design Aid for Conceptual Stages of Product Design*, in *International Conference on Engineering Design, IECD'93* 1993, The Hague.
19. Thierry Despeyroux. *TYPOL: a formalism to implement natural semantics*, INRIA, Sophia Antipolis, 1988.
20. Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, Wiley-IEEE Press, 2003, http://www.mathcore.com/drmodelica.
21. Peter Fritzson, Peter Aronsson, Peter Bunus, Vadim Engelson, Levon Saldamli, Henrik Johansson and Andreas Karström. *The Open Source Modelica Project*, in *Proceedings of The 2th International Modelica Conference*,March 18-19, 2002, Munich, Germany.
22. Plotkin G.D. *A structural approach to operational semantics*, Århus University, Århus, 1981.
23. Ian Horrocks. *The FaCT System*, http://www.cs.man.ac.uk/~horrocks/FaCT/.
24. Gilles Kahn. *Natural Semantics*, in *Programming of Future Generation Computers*, Fuchi K. and Niva M., Editors, 1988, Elsevier Science Publishers: North Holland. p. 237-258.
25. Deborah L. McGuinness. *Explaining Reasoning in Description Logics*, Rutgers University Thesis, New Brunswick, 1996.
26. Deborah L. McGuinness, Alex Borgida. *Explaining Subsumption in Description Logics*, in *Fourteenth International Joint Conference on Artificial Intelligence* August 1995.
27. Deborah L. McGuinness, Paulo Pinheiro da Silva. *Infrastructure for Web Explanations*, in *2nd International Semantic Web Conference (ISWC2003)*,October, 2003, USA: Springer.
28. Mikael Pettersson. *Compiling Natural Semantics*, Lecture Notes in Computer Science (LNCS) 1549, Springer-Verlag, 1999, http://www.ida.liu.se/~pelab/rml.
29. Mikael Pettersson. *Compiling Natural Semantics*, *Department of Computer and Information Science*, Linköping University, Linköping, Dissertation No. 413, 1995.
30. Adrian Pop, Peter Fritzson. *ModelicaXML: A Modelica XML representation with Applications*, in *International Modelica Conference*,3-4 November, 2003, Linköping, Sweden.
31. Adrian Pop, Olof Johansson and Peter Fritzson. *An Integrated Framework for Model-Driven Product Design and Development using Modelica*, in *Conference on Simulation and Modeling*,23-24 Septemeber, 2004, Copenhagen, submited.
32. Adrian Pop, Ilie Savga, Uwe Assmann and Peter Fritzson. *Composition of XML dialects: A ModelicaXML case study*, in *Software Composition Workshop 2004, affiliated with ETAPS 2004*,3 April, 2004, Barcelona.
33. Andrew Tolmach, Andrew W. Appel. *A debugger for Standard ML,* Journal of Functional Programming, 1995, **5**(2).
34. Christopher Welty. *An Integrated Representation for Software Development and Discovery*, 1996.