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

CUGS thesis proposal

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

Outline

- Introduction
- Research Track
- Thesis goals
  - short term goal
  - long term goal
- Thesis Plan
- Conclusions

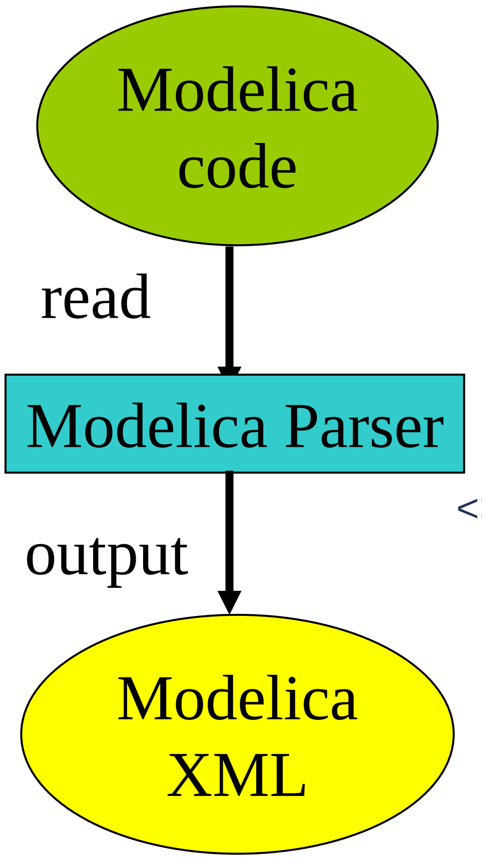
- The research combines several computer science areas
  - Compilers, Debuggers, Compiler generation for high level declarative programming languages (Natural Semantics)
  - Semantic Web (Description Logics)
  - Integrated product design using Modeling and Simulation with Modelica
- Involvement in Research Projects
  - SWEBPROD (Semantic Web for Products)
  - REWERSE (Reasoning on the web with rules and semantics)
  - SECD (Systems Engineering & Computational System Design)

- short term goal
  - practical tool implementation for Semantic Web languages using Natural Semantics
- long term goal
  - adapt and integrate Semantic Web technologies into a framework for model-driven product design and development

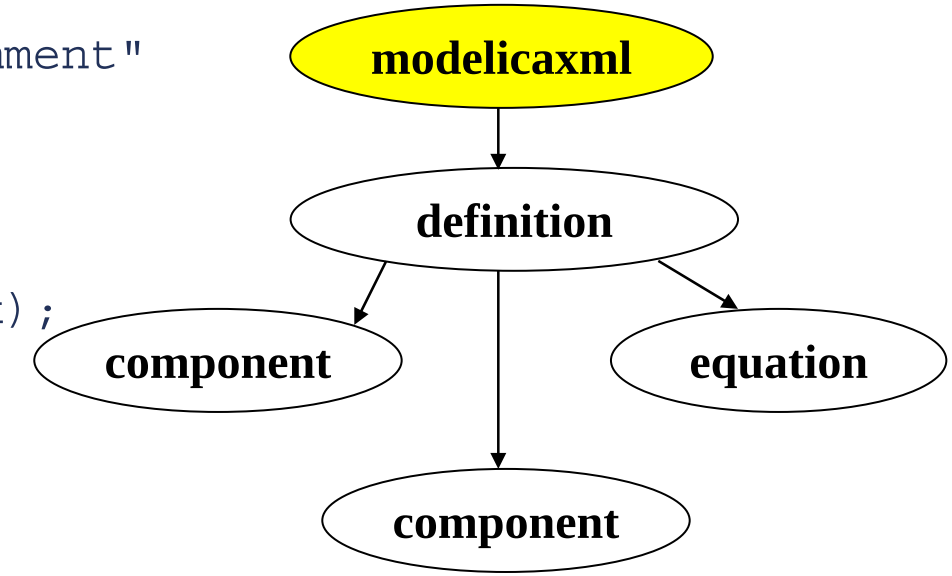
- Preliminary results
  - Adrian Pop, Peter Fritzson, *ModelicaXML: A ModelicaXML representation with Applications*, International Modelica Conference, 2003
  - Adrian Pop, Ilie Savga, Uwe Assmann, Peter Fritzson, *Composition and XML dialects: A ModelicaXML case study*, Software Composition Workshop, 2004
  - Adrian Pop, Olof Johansson, Peter Fritzson, *An Integrated Framework for Model-Driven Product Design and Development using Modelica*, Conference on Simulation and Modeling, 2004
- Systems
  - a Relational Meta-Language (RML) debugger
  - ModelicaXML toolbox

- *Declarative language*
  - Equations and mathematical functions allow acausal modeling, high level specification, increased correctness
- *Multi-domain modeling*
  - Combine electrical, mechanical, thermodynamic, hydraulic, biological, control, event, real-time, etc...
- *Everything is a class*
  - Strongly typed object-oriented language with a general class concept, Java & Matlab like syntax
- *Visual component programming*
  - Hierarchical system architecture capabilities
- *Efficient, non-proprietary*
  - Efficiency comparable to C; advanced equation compilation, e.g. 300 000 equations

ModelicaXML




```
class Test "comment"
   Real x;
   Real xdot;
 equation
   xdot = der(x);
end Test;
```




```
<modelicaxml>
  <definition ident= "Test"
              comment="comment">
    <component ident="x" type="Real"
               visibility="public" />
    <component ident="xdot" type="Real"
               visibility="public" />
    <equation>...</equation>
  </definition>
</modelicaxml>
```

ModelicaXML

- Advantages
  - 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) from practically any programming language
  - Analysis of Modelica models (model checkers and validators)
  - Pretty printing (un-parsing)
  - Translation between Modelica and other modeling languages (interchange)
  - Query and transformation of Modelica models
  - Certain models could be translated to and from the Unified Modeling Language (UML)
- Shortcommings
  - XML can represent only structure, no semantics
- Initial ideas on using Semantic Web
  - to represent some of the Modelica semantics

ModelicaXML composition and transformation

- **Why the need for Modelica composition and transformation?**
  - Interoperability between existing modeling languages or CAD tools and Modelica
  - Automatic generation of different version of models from product specifications. Choosing best design based on automatic simulation.
  - Automatic configuration of models using external sources (XML, databases, files)
  - Protection of intellectual property through obfuscation
  - Fine grain support for library developers

- The information in the current web:
  - has meaning for human only
  - is not machine processable
- Semantic Web brings:
  - semi-structured information
  - means to add more than structure (semantics/constrains) on data
  - languages: XML, XMLSchema, RDF, RDFS, OWL
  - reasoning and inferences services (Description Logics): subsumption, classification, coherence checking, etc
  - integration and reuse of knowledge by using shared ontologies

ModelicaXML and Semantic Web

- The benefit of using Semantic Web languages for Modelica
  - Models could be automaticaly translated between modeling tools
  - Software information systems (SIS) could more easily be constructed for Modelica, facilitating model understanding and information finding
  - Model consistency could be checked

- Based on
  - Gordon Plotkin's Structural Operational Semantics (SOS)
  - Gentzen's Sequent Calculus for Natural Deduction.
- "Natural Semantics" (NS)
  - term by Gilles Kahn
  - formalism widely used for specifications of:
    - type systems
    - programming languages

Natural Semantics - Syntax

$$\frac{H_1 \vdash T_1 : R_1 \quad \ldots \quad H_n \vdash T_n : R_n}{H \vdash T : R} \quad \text{if } \langle\text{cond}\rangle$$

- Hi are hypotheses (environments)
- Ti are terms (pieces of abstract syntax)
- Ri are results (types, run-time values, changed environments)
- Hj |- Tj : Rj are sequents
- Premises or preconditions are above the line
- Conclusion is below the line
- Condition on the side if exists must be satisfied

Natural Semantics vs Relational Meta-Language (RML)

RML has the same visual syntax as NS

```
rule    <cond>
        RelName1(H1,T1) => R1 & ...
        RelNameN(Hn,Tn) => Rn &
        ------------------------------
        RelName(H, T) => R
```

RML language properties

- Separation of input and output arguments/results
- Statically strongly typed
- Polymorphic type inference
- Efficient compilation of pattern-matching

RML debugger

- based on source code instrumentation
- some support from the runtime system

Short term goal

- reasoning tools for Semantic Web languages (OWL Lite/DL)
  - implementation in RML of Natural Semantics specifications for Description Logics reasoning tasks
  - use the RML debugger to output explanation of such tasks
- possible problems:
  - scalability
  - RML has some limitations (formal arguments to relations, number of constructors in datatypes)
- why?
  - to have our own reasoning toolbox and to be able to experiment with alternative semantics and means to express the dynamic semantics implemented in RML

Long term goal

- integrating Semantic Web technologies with Product Design and Modeling and Simulation tools
  - model interchange
  - use of already defined vocabularies (taxonomies) and ontologies in the product design process
  - facilitating several tasks in the product development management
    - consistency checking (documents, components, forms, etc)
    - searching and information retrieval (large distributed libraries)
    - composition and interoperability
    - traceability (from requirements to design to product)
    - comparison (version management etc)

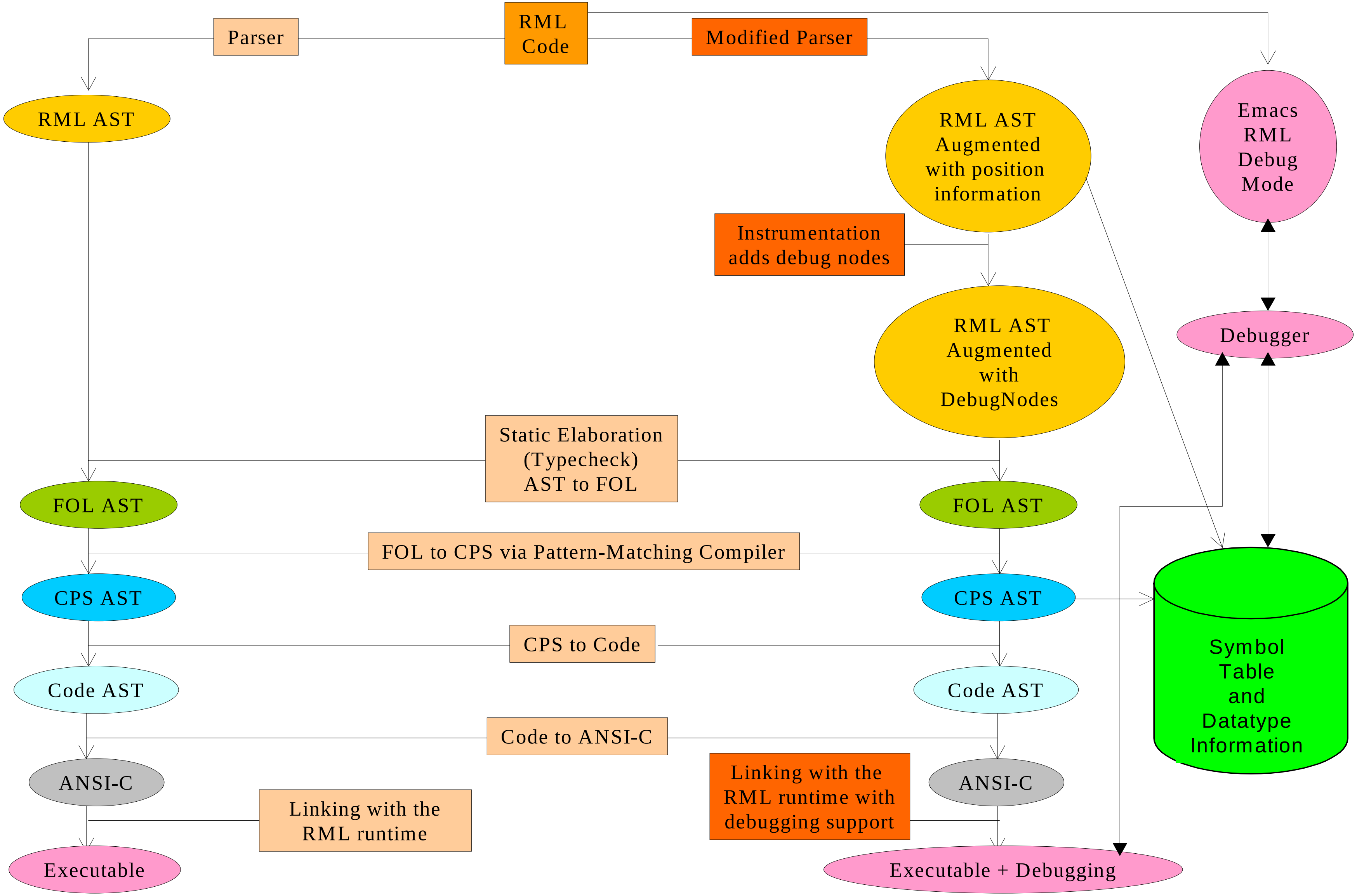
Thesis Plan

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

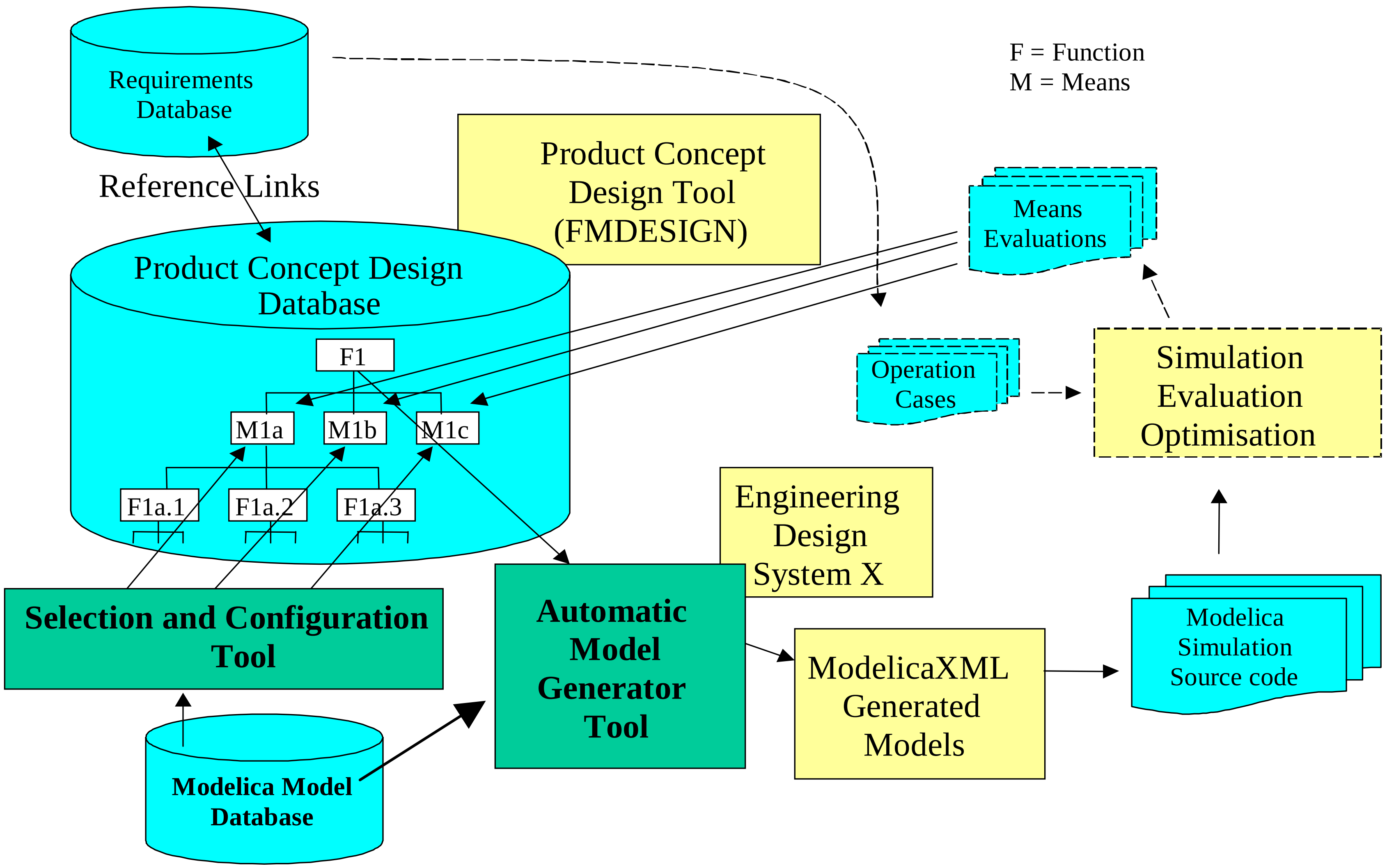
Thank you!

Questions?

Relational Meta-Language debugger



Integrated Product Development

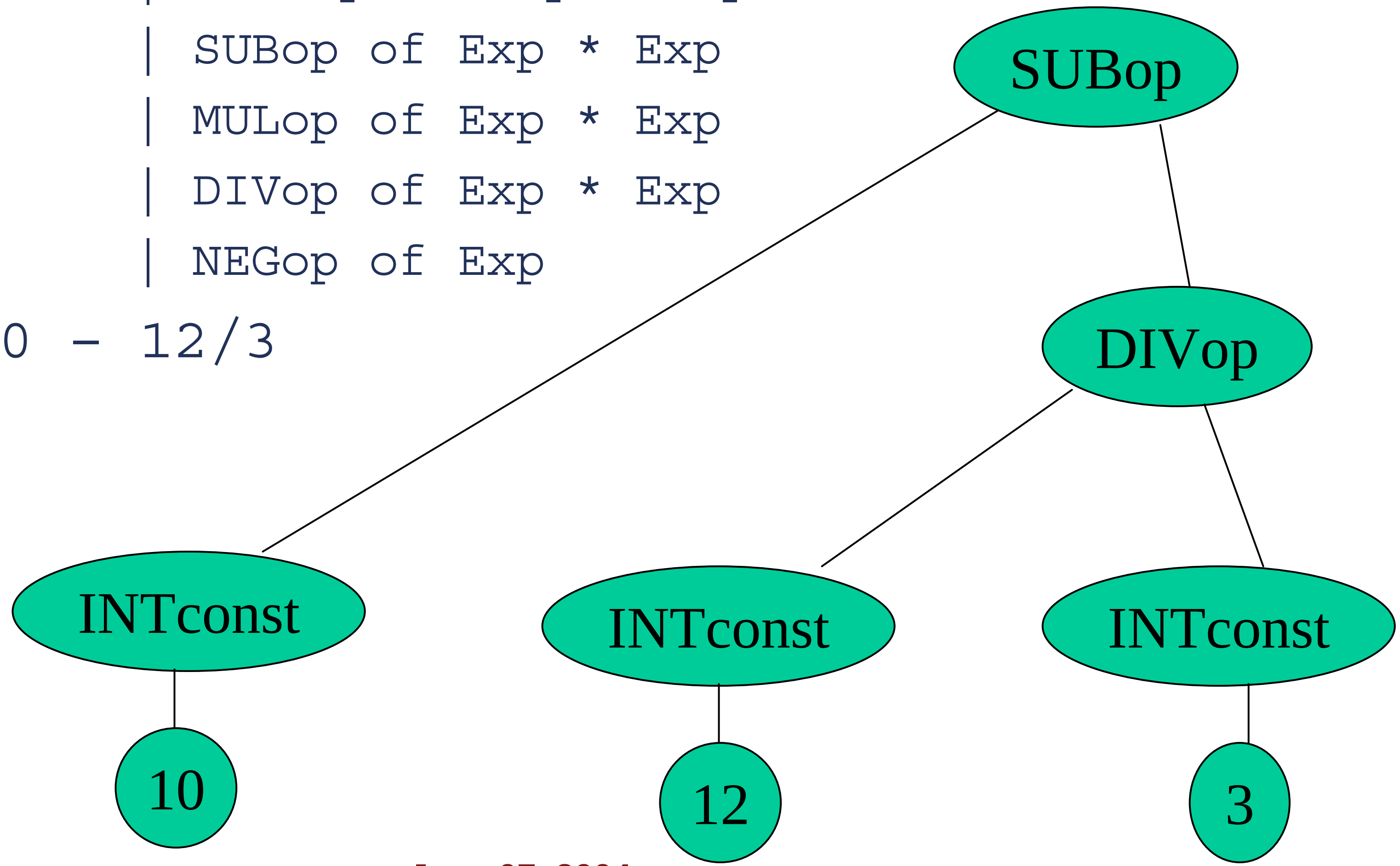


RML example: the Exp language

- Abstract syntax

```
datatype Exp = INTconst of int
             | PLUSop of Exp * Exp
             | SUBop of Exp * Exp
             | MULop of Exp * Exp
             | DIVop of Exp * Exp
             | NEGop of Exp
```

```
Exp: 10 - 12/3
```



RML example: the Exp language

- Relation eval

```
relation eval: Exp => int =
 axiom eval(INTconst(ival)) => ival
 rule eval(e1) => v1 & eval(e2) => v2 & int_add(v1,v2) => v3
      ------------------------------------------------------
      eval (PLUSop(e1,e2)) => v3
 rule eval(e1) => v1 & eval(e2) => v2 & int_sub(v1,v2) => v3
      -----------------------------------------------------
      eval (SUBop(e1,e2)) => v3
 rule eval(e1) => v1 & eval(e2) => v2 & int_mul(v1,v2) => v3
      -----------------------------------------------------
      eval (MULop(e1,e2)) => v3
 rule eval(e1) => v1 & eval(e2) => v2 & int_div(v1,v2) => v3
      -----------------------------------------------------
      eval (DIVop(e1,e2)) => v3
 rule eval(e) => v1 & int_neg(v1) => v2
      -----------------------------------------------------
      eval (NEGop(e)) => v2
end
```