Extensible Modeling Languages
Utilizing Libraries for Model Creation, Use, and Domain-Specific Extensions

5th MODPROD Workshop on Model-Based Product Development
February 8, 2011

David Broman
Department of Computer and Information Science
Linköping University, Sweden
david.broman@ida.liu.se

Equation-Based Object-Oriented (EOO) Languages

Domain-Specific Language (DSL)
- Primarily domain: Modeling of physical systems
- Multiple physical domains: e.g., mechanical, electrical, hydraulic

Models and Objects
- Object in e.g., Java, C++: object = data + methods
- Objects in EOO languages: object = data + equations

Acausality
- At the equation-level
  \[ u = R \cdot i \]
- At the object connection level

- Modelica
- VHDL-AMS
- gPROMS

Part I
Expressiveness

Part II
Extensibility

Part III
Formalization
Expressiveness – ease and possibility of expressing complex models or tasks

Language versions:
- $A$, v1.0 → $A$, v1.1 → $A$, v2.0 → $A$, v2.2

Standard library versions:
- $L$, v1.0 → $L$, v1.1 → $L$, v2.0 → $L$, v2.2
Extensibility – mechanisms to add new language features

**Uses**
- Simulation
- Optimization
- Code generation for real-time
- Model export
- Grey-box system identification etc.

---

### Formalization

**Formalization – precise semantics “meaning” of the language**

**Language Specifications of state-of-the-art are informally defined**

- hard to interpret unambiguously when developing compilers
- hard to reason about when extending the language
- hard to formalize e.g. Modelica due to size and complexity
What is MKL?

Modeling Kernel Language (MKL)

- Purpose: Research language – explore new concepts
- Bottom-up approach
- Small extensible language
- Precise formal semantics
- Base it on a proven foundation – the lambda calculus
- Statically typed functional language

Platform for experimental equation-based DSLs (Continuous-time, hybrid, structural dynamic, and acusal models)

Part I
Expressiveness

Part II
Extensibility

Part III
Formalization
Higher-Order Acusal Models (HOAM)

Higher-Order Functions
I.e. first class citizens, can be passed around as any value

+  Acausal Models
Models in EOO languages, composing DAEs and other interconnected models.

=  Higher-Order Acausal Models
I.e., first class acausal models.

HOAM – Example

Example of a mechatronic system with a DC motor and a flexible shaft

let ShaftElement flangeA:Rotational -> flangeB:Rotational ->
    Equations =
    let r1:Rotational in
    Spring 8. flangeA r1;
    Damper 1.5 flangeA r1;
    Inertia 0.5 r1 flangeB

One shaft element is created by standard components.
Example of a mechatronic system with a DC motor and a flexible shaft

let MechSys =
    let r1:Rotational in
    let r2:Rotational in
    let r3:Rotational in
    DCMotor r1;
    Inertia 0.2 r1 r2;
    (serializeRotational 120 ShaftElement) r2 r3

Higher-order function that can compose any mechanical component in series
### Modelica Environment

- **Model Library**

- **Modelica Model**

- **Modelica Tool**

- **Result** (e.g., simulation)

**Language Specification**
- Type checking
- Collapsing the instance hierarchy
- Connection Semantics
- Simulation (Runtime)

### MKL Environment

- **Model Library**

- **MKL Model**

- **MKL Tool**

- **Result** (e.g., simulation)

**Library for using models**
- Connection Semantics
- Simulation (Runtime)

**Language Specification**
- Type checking
- Collapsing the instance hierarchy
- Connection Semantics
- Simulation (Runtime)

**Benefits**
- Tool vendors – no need to update tool after lib ext.
- Library developer - less dependent on tool vendors
- A model behaves the same way in different tools
Part I
Expressiveness

Part II
Extensibility

Part III
Formalization

Intensional Analysis and Model Lifting

Static Semantics

Dynamic Semantics

Lifted Model

Model Lifting

Type Checking

MKL Model

Lifted Model

Collapsed using evaluation

Equation System

Analysis, models treated as data. Connection Semantics, Simulation, etc.

Result

Part III

Formalization
Formalization of Semantics

Part I
Expressiveness

Part II
Extensibility

Part III
Formalization

How do we verify our solution?

Prototype Implementation

Symbol Table

Type Safety Proof

Two main lemmas
• Progress
• Preservation
Conclusions

Expressiveness (HOAM)

Extensibility (Library Approach)

Formalization (Operational Semantics)

Modeling Kernel Language (MKL)

Thanks for listening!

Part I
Expressiveness

Part II
Extensibility

Part III
Formalization