# 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

Linköpings universitet

## Equation-Based Object-Oriented (EOO) Languages



**Domain-Specific Language (DSL)**

- **Primarily domain:** Modeling of physical systems
- **Multiple physical domains:** e.g., mechanical, electrical, hydraulic

- **Modelica**
- **VHDL-AMS**
- **gPROMS**

**Models and Objects**

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

**Equation-Based Object-Oriented (EOO)**

**Acausality**

- **At the equation-level**
  u = R * i
- **At the object connection level**

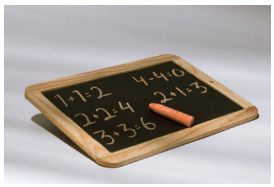
# Agenda

David Broman
david.broman@liu.se

**Part I**
**Expressiveness**

**Part II**
**Extensibility**

**Part III**
**Formalization**







# Expressiveness

David Broman
david.broman@liu.se

**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

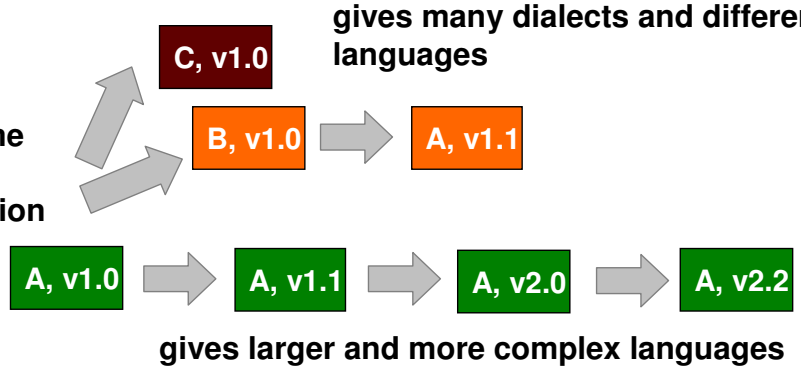
David Broman
david.broman@liu.se

## Extensibility – mechanisms to add new language features

**Uses**

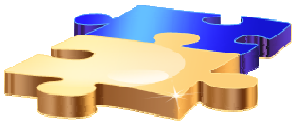
- Simulation
- Optimization
- Code generation for real-time
- Model export
- Grey-box system identification

etc.




gives many dialects and different languages

gives larger and more complex languages







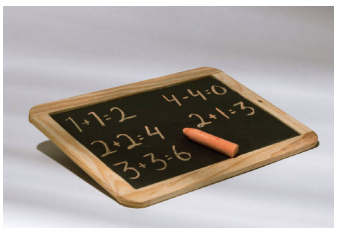
# Formalization

David Broman
david.broman@liu.se

## Formalization – precise semantics "meaning" of the language

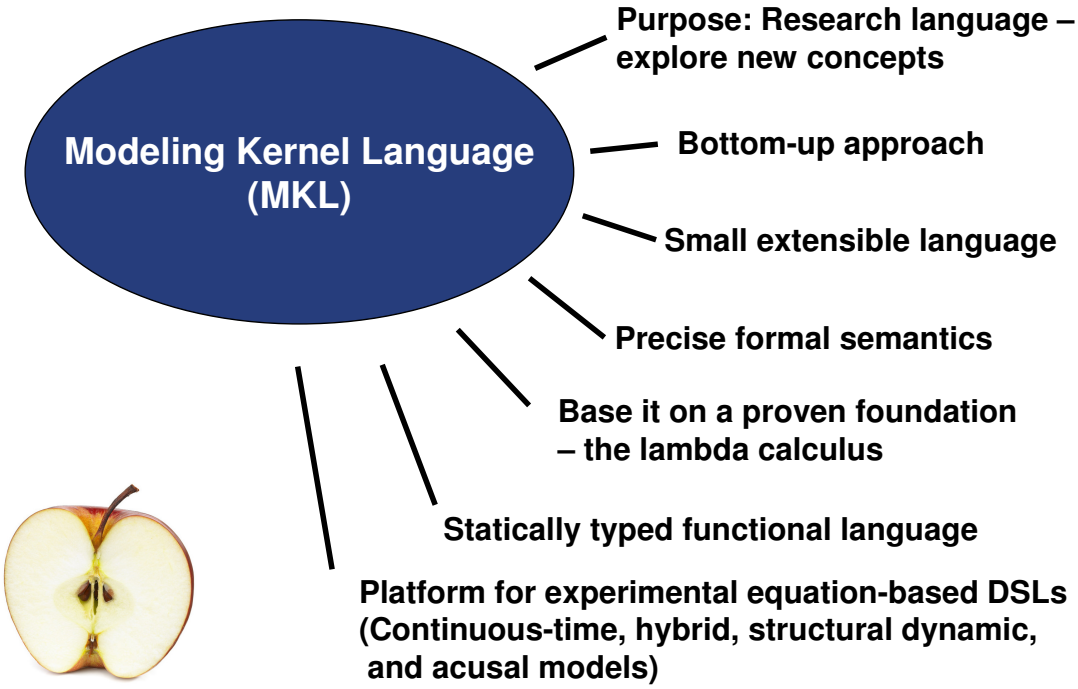
**Language Specifications of state-of-the-art are informally defined**

- 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?

David Broman
david.broman@liu.se










David Broman
david.broman@liu.se

# Part I

## Expressiveness

# Expressiveness - HOAM

David Broman
david.broman@liu.se

## 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.

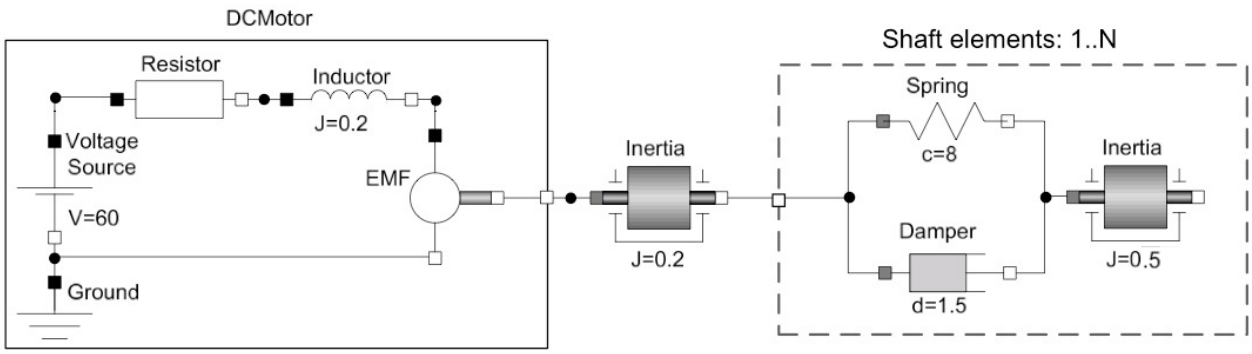




# HOAM – Example

David Broman
david.broman@liu.se

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



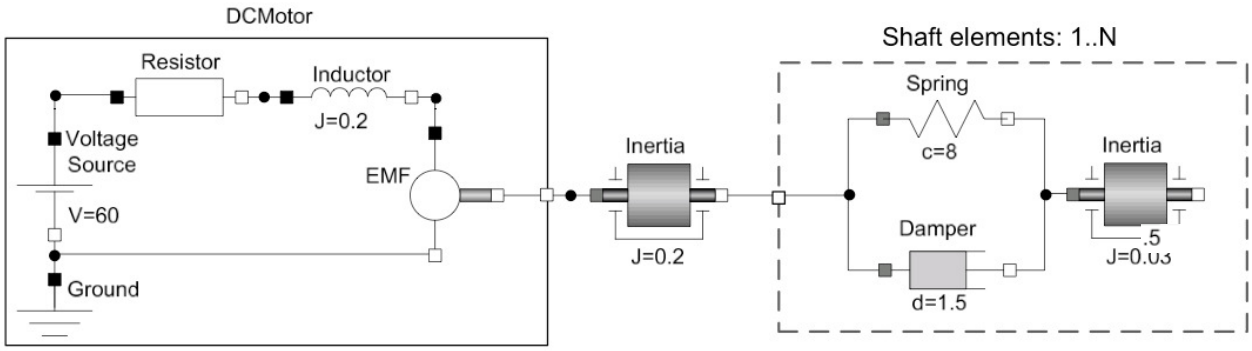

```
let ShaftElement flangeA:Rotational -> flangeB:Rotational ->
                 Equations =
  let r1:Rotational in
  Spring 8. flangeA r1;
  Damper 1.5 flangeA r1;
  Inertia 0.5 r1 flangeB
```

**One shaft element is created by standard components.**

# HOAM – Example

David Broman
david.broman@liu.se

**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**





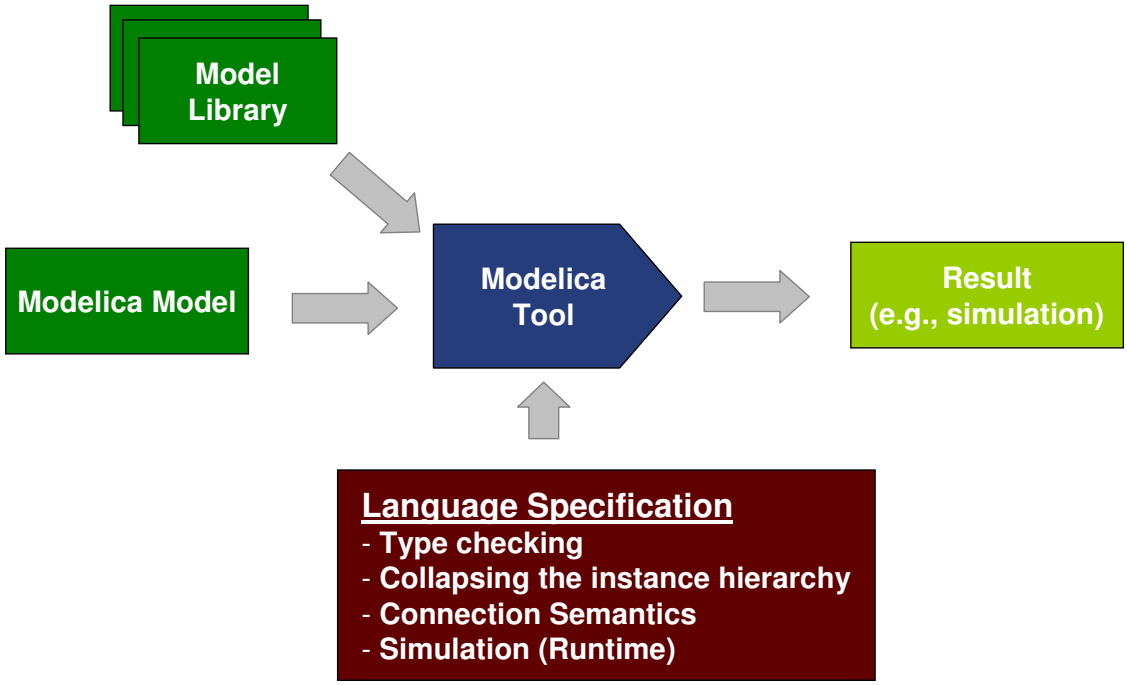
David Broman
david.broman@liu.se

# Part II

## Extensibility

# Modelica Environment

David Broman
david.broman@liu.se

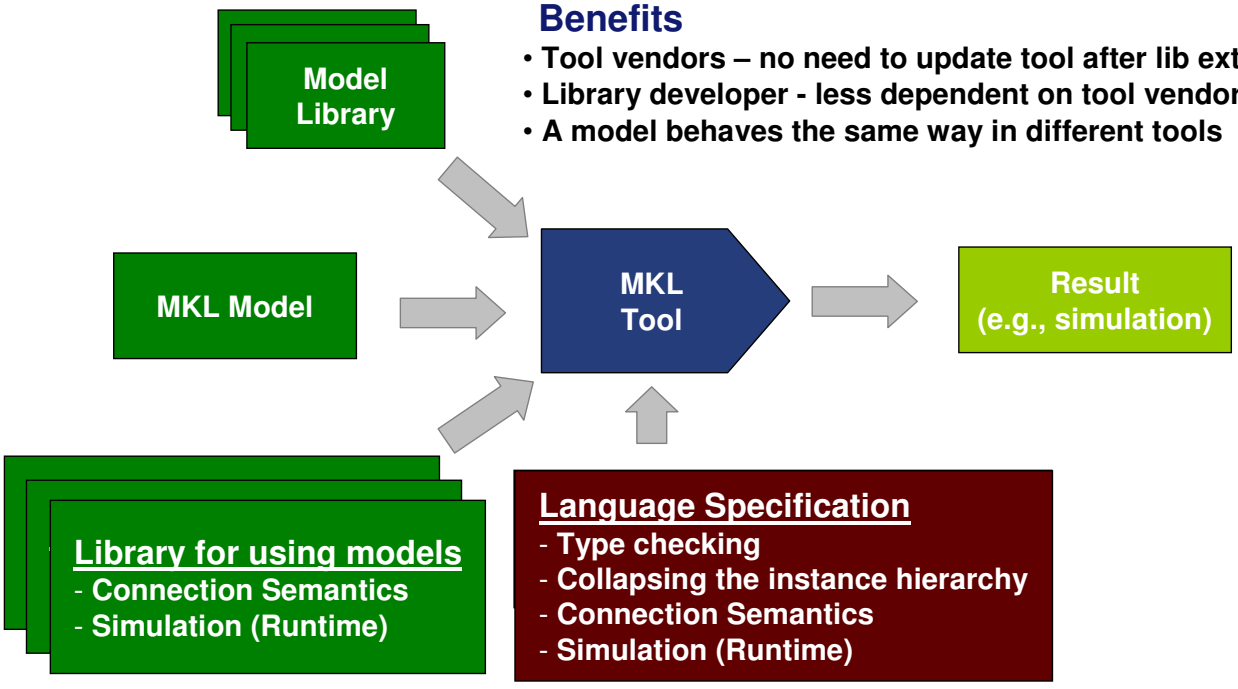






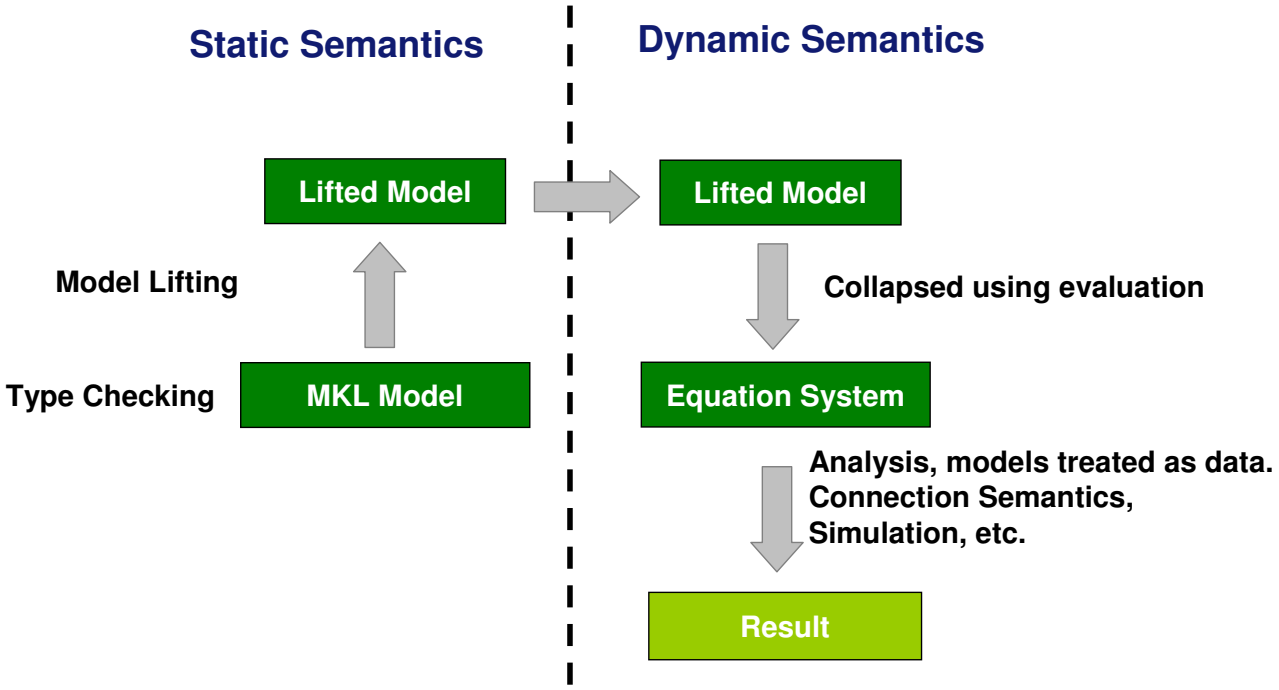

# MKL Environment

David Broman
david.broman@liu.se



# Intensional Analysis and Model Lifting

David Broman
david.broman@liu.se



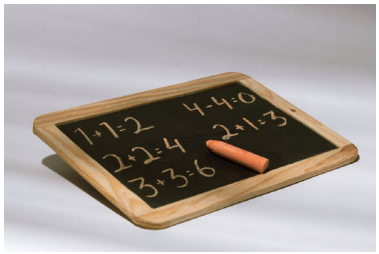





David Broman
david.broman@liu.se

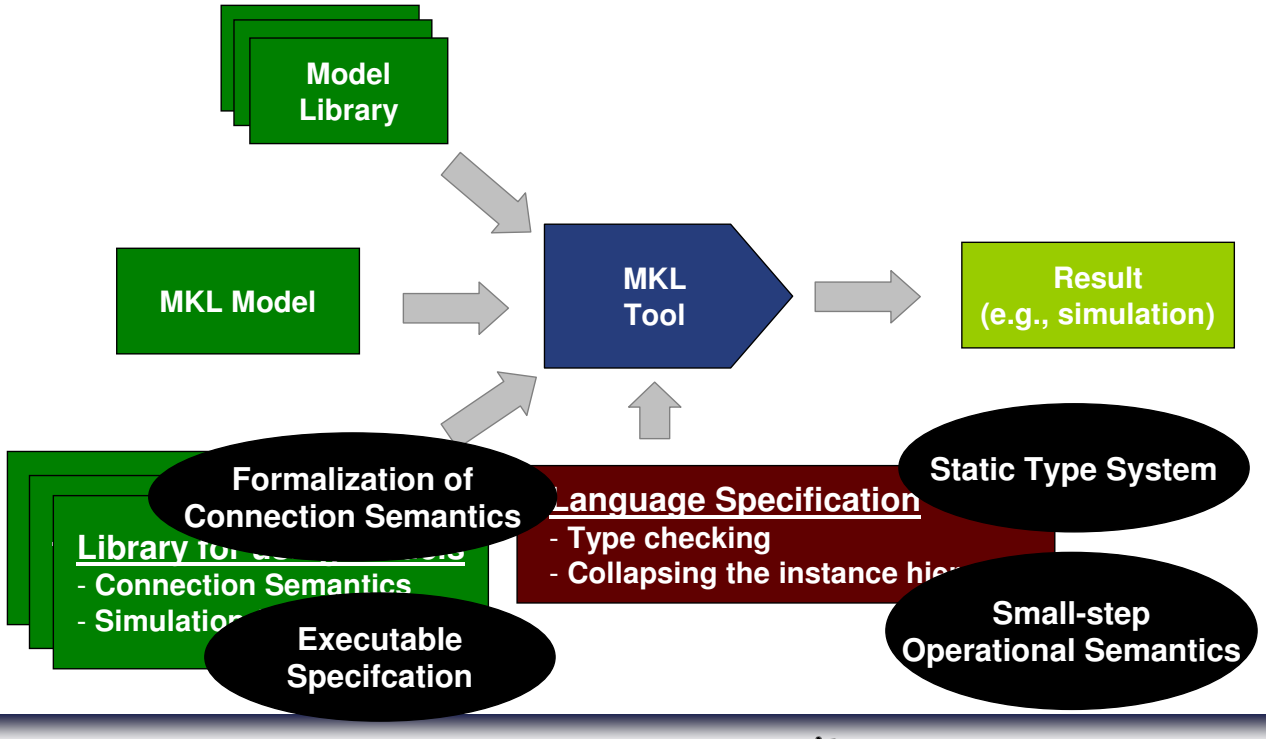
# Part III

## Formalization



# Formalization of Semantics

David Broman
david.broman@liu.se



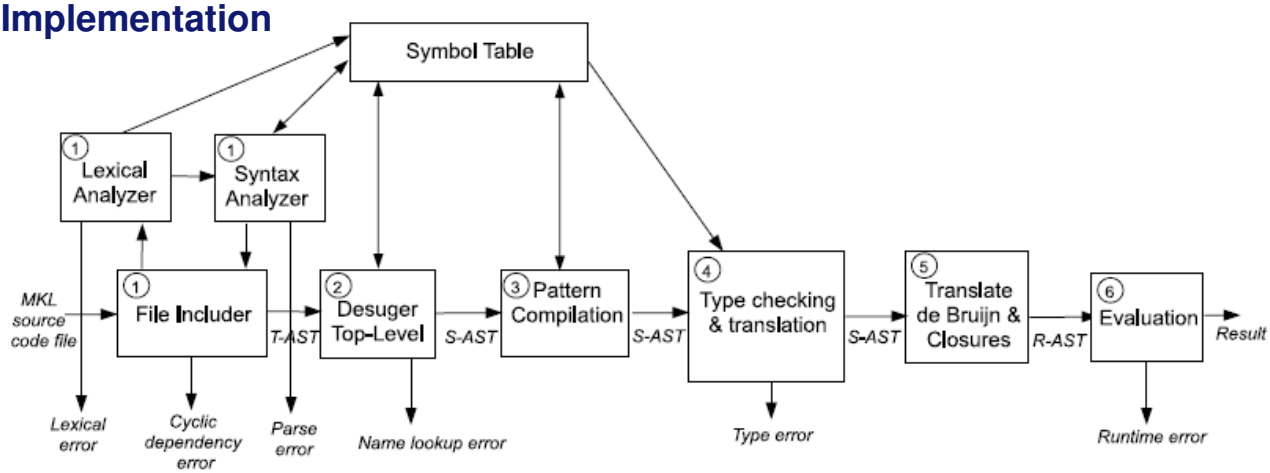





# How do we verify our solution?

David Broman
david.broman@liu.se

## Prototype Implementation



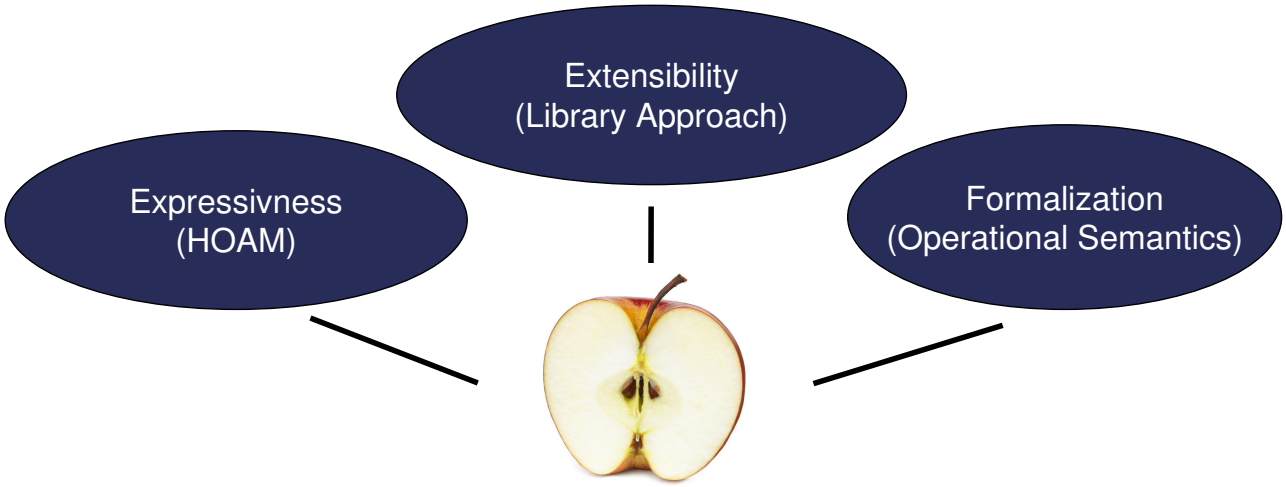

## Type Safety Proof

**Two main lemmas**

- **Progress**
- **Preservation**

# Conclusions

David Broman
david.broman@liu.se




**Modeling Kernel Language (MKL)**

## Thanks for listening!