A Practical Ontology Framework for Static Model Analysis

Ben Lickly
Charles Shelton
Elizabeth Latronico
Edward A. Lee

EMSOFT 2011
October 9-14, 2011
UC Berkeley & Bosch Research
A Taxonomy of Modeling Issues

Abstract Syntax
(static structure)
[software architecture, metamodeling, higher-order components, …]

Dynamic Semantics
(models of computation)
[automata, synchronous languages, tagged signal model, Kahn networks, quantitative system theory, …]

Static Semantics
(type systems)
[type inference/checking, ontologies, behavioral types, …]

We are here:
Static semantics: Correctly Composing Models
Model Composition Errors
Model Composition Errors

Transposition error
Model Composition Errors

Transposition error

Units error

- position (meters^3)
- velocity ((m/s)^3)
- acceleration ((m/s^2)^3)
- temperature (centigrade)
- altitude (feet)
- weight (kg, tare)

- altitude (meters)
- temperature (centigrade)
- weight (kg, gross)
Model Composition Errors

- Units error
- Semantics error
- Transposition error
Our Goals

- Detect such interfacing errors
- Minimize the manual annotations required (use inference)
- Improve communication in engineering teams by augmenting interface definitions with semantic information
Outline

1. Existing (Finite) Ontology Analysis
2. Value-parametrized Ontologies
3. Recursive Ontologies
Components in a model (e.g. parameters, ports, messages, fields in a packet, etc.) can have properties drawn from a lattice.

Components in a model (e.g. actors) can impose constraints on property relationships.

The type system infrastructure can infer concepts and detect errors.

Here is a lattice representing a simple dimension ontology.
Inferring Concepts

User-defined constraints added (in as few places as possible)
Inferring Concepts

User-defined constraints added (in as few places as possible)

Relationships between lattice elements are constraints imposed by the Integrator component.

FuelDimensionSystemSolver

Double click to Apply Ontology

- $x: 4.0$
- `fuelDimensionSystem::constraint3: x >= Flow`

User-defined constraints added (in as few places as possible)
Inferring Concepts

- Dimensionless
- Level
- Flow
- Unknown

FuelDimensionSystemSolver

Double click to Apply Ontology

x: 4.0

fuelDimensionSystem::constraint3: x >= Flow
Inferring Errors

FuelDimensionSystemSolver

- x: 4.0
- fuelDimensionSystem::constraint3: x >= Level
Inferring Errors

FuelDimensionSystemSolver

Double click to Apply Ontology

- \( x: 4.0 \)
- \( \text{fuelDimensionSystem::constraint3}: x \geq \text{Level} \)
More Complex Model: Cooperative Cruise Control

This model shows a simple adaptive cruise control system, illustrating model-integrated control strategies. A leading car model produces information that is observed with possible flaws by a following car. If the following car detects flaws, it uses a conservative strategy. Otherwise, it tracks the leading car closely.

Example from “Scalable Semantic Annotation Using Lattice-Based Ontologies” (MODELS 2009)
More Complex Model: Cooperative Cruise Control
More Complex Model: Cooperative Cruise Control

This model shows a simple adaptive cruise control system, illustrating model-integrated control strategies. A leading car model produces information that is observed with possible flaws by a following car. If the following car detects flaws, it uses a conservative strategy. Otherwise, it tracks the leading car closely.
More Complex Model: Cooperative Cruise Control

Car simulator. This model takes as input a desired speed and implements a simple proportional controller with the specified time constant to achieve that speed. It outputs the acceleration, speed, and position of the car.
Why is this insufficient?
Units Errors

- Position (meters^1)
- Velocity ((m/s)^1)
- Acceleration ((m/s^2)^1)
- Temperature (Celsius)
- Altitude (feet)
- Weight (kg, tare)

Units error

- Position (meters^1)
- Velocity ((m/s)^1)
- Acceleration ((m/s^2)^1)
- Temperature (Celsius)
- Altitude (meters)
- Weight (kg, gross)
Definitions

Least Upper Bound/\( LUB(S) \): The least \( x \), if it exists, such that
\[
\forall s \in S, \ x \geq s.
\]

Greatest Lower Bound/\( GLB(S) \): The greatest \( x \), if it exists, such that
\[
\forall s \in S, \ x \leq s.
\]
Definitions

**Complete Lattice:** Partially ordered set in which all subsets have a *LUB* and *GLB*.
Value-parametrized Concepts

Concepts with values can be useful.
Value-dependent Constraints

\[
\text{AddSubtract}
\]

\[
\begin{align*}
\text{out} & \geq f_{\text{sub}}(x, y) \\
f_{\text{sub}}(x, y) &= \\
&= \begin{cases} 
\text{Unused} & \text{if } x = \text{Unused} \text{ or } y = \text{Unused} \\
\text{Constant}(c_x - c_y) & \text{if } x = \text{Constant}(c_x) \\
& \quad \text{and } y = \text{Constant}(c_y) \\
\text{Nonconst} & \text{otherwise}
\end{cases}
\end{align*}
\]
Constant Propagation Analysis

Ramp

Const

Const2

Const3

MultiplyDivide

AddSubtract

TimedPlotter
Full Unit Systems
Units Library

Divides units into *base dimensions* and *derived dimensions*.

Contains commonly used units of mass, time, length, force, etc.
Derived Concepts

- Dimensionless
- Time
- Position
- Velocity
- Acceleration

<table>
<thead>
<tr>
<th>dimensionArray:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Dimension = &quot;LengthConcept&quot;, Exponent = 1}, {Dimension = &quot;TimeConcept&quot;, Exponent = -2}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LengthConcept:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TimeConcept:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m_per_sec2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{LengthConcept = &quot;m&quot;, TimeConcept = &quot;sec&quot;, &quot;sec&quot;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cm_per_sec2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{LengthConcept = &quot;cm&quot;, TimeConcept = &quot;sec&quot;, &quot;sec&quot;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ft_per_sec2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{LengthConcept = &quot;ft&quot;, TimeConcept = &quot;sec&quot;, &quot;sec&quot;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kph_per_sec:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{LengthConcept = &quot;km&quot;, TimeConcept = &quot;hr&quot;, &quot;sec&quot;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mph_per_sec:</th>
</tr>
</thead>
<tbody>
<tr>
<td>{LengthConcept = &quot;mi&quot;, TimeConcept = &quot;hr&quot;, &quot;sec&quot;}</td>
</tr>
</tbody>
</table>
Automatically Inferred Constraints

Multiply/divide and related constraints need not be specified for these unit systems.

\[ \text{oldPosition} + (\text{currentTime} - \text{oldTime}) \times \text{oldSpeed} \]
Semantics Errors

- Transposition error
- Semantics error
Domain-specific Units

![Diagram showing the relationship between Conflict, OilTemperature, and AtmosphericTemperature]
Record Types

record = {
  label1 : data1;
  label2 : data2;
  ...
}

Representing the concept of a record can be difficult.
Recursive Ontologies

Recursive lattices can express structured data types.
Related Work

1. Constraint Satisfiability (Rehof and Mogensen)
   • Efficient inference algorithm

2. Abstract Interpretation (Cousot and Cousot)
   • Analysis of static semantics using complete lattices.

3. Existing systems for unit analysis, including those for Ada, SCADE, SystemC, and C++
Conclusion

Our analysis framework:

1. Efficiently infers unspecified concepts throughout large models.
2. Includes general mechanisms for infinite lattices.
3. Specifically includes useful features for unit systems.
Questions?

EMSOFT, October 2011 36  Lickly, Shelton, Latronico & Lee