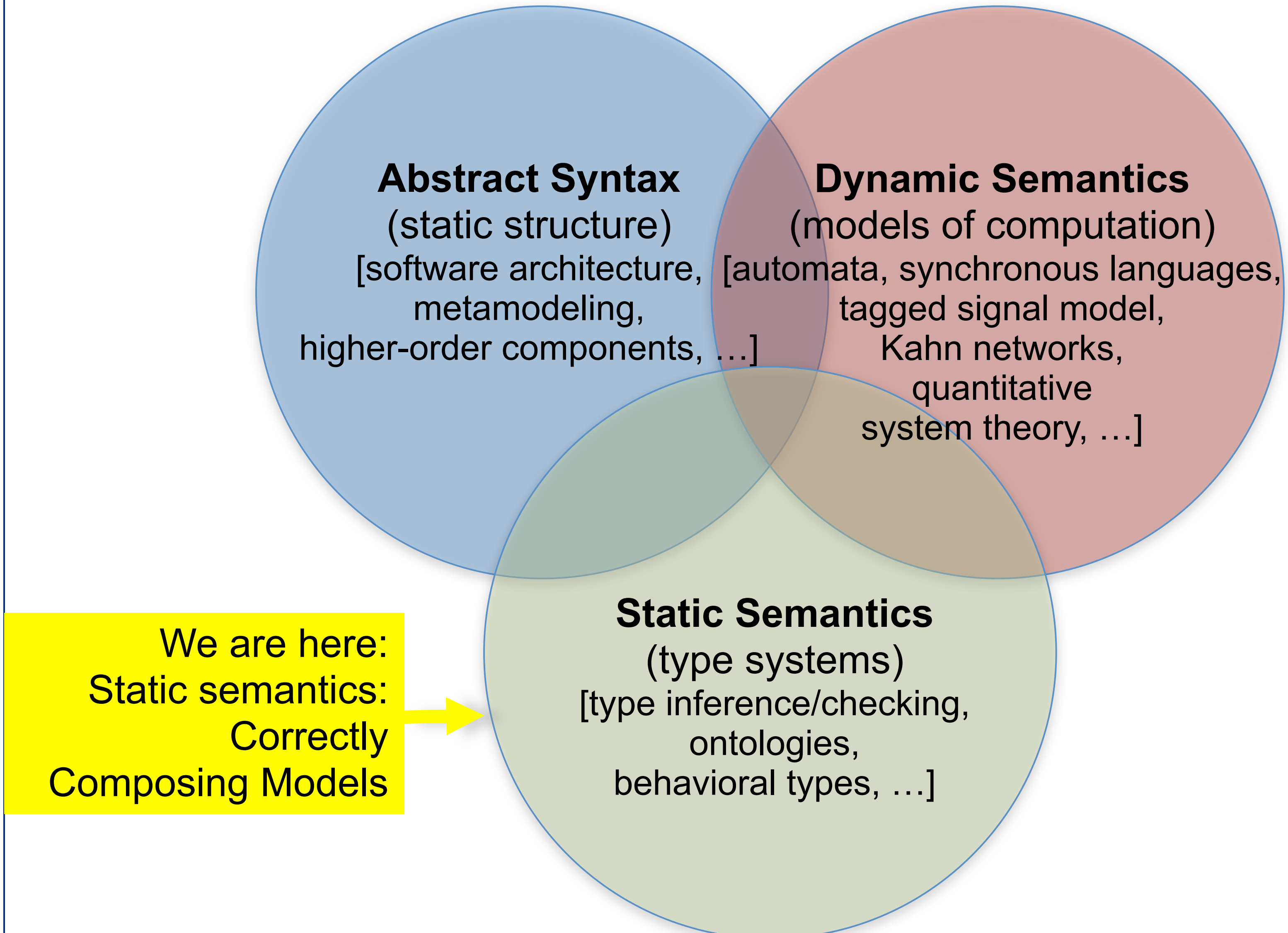
A Practical Ontology Framework for Static Model Analysis

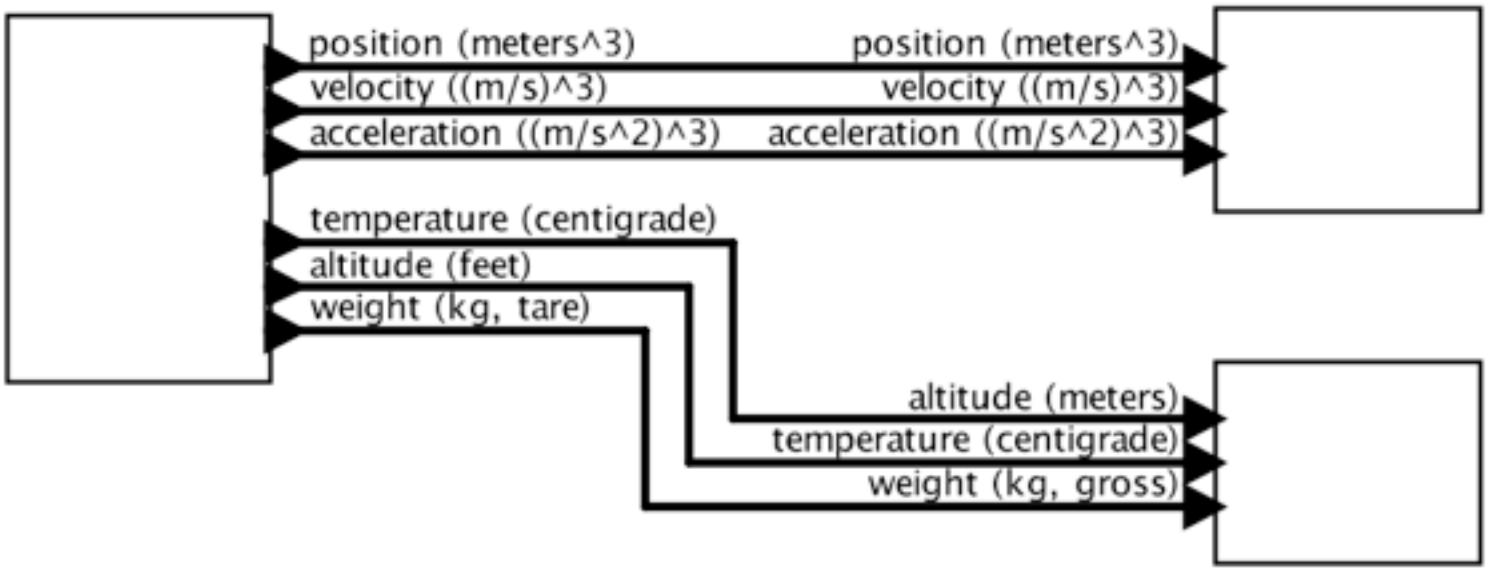
Ben Lickly
Charles Shelton
Elizabeth Latronico
Edward A. Lee

EMSOFT 2011
October 9-14, 2011
UC Berkeley & Bosch Research

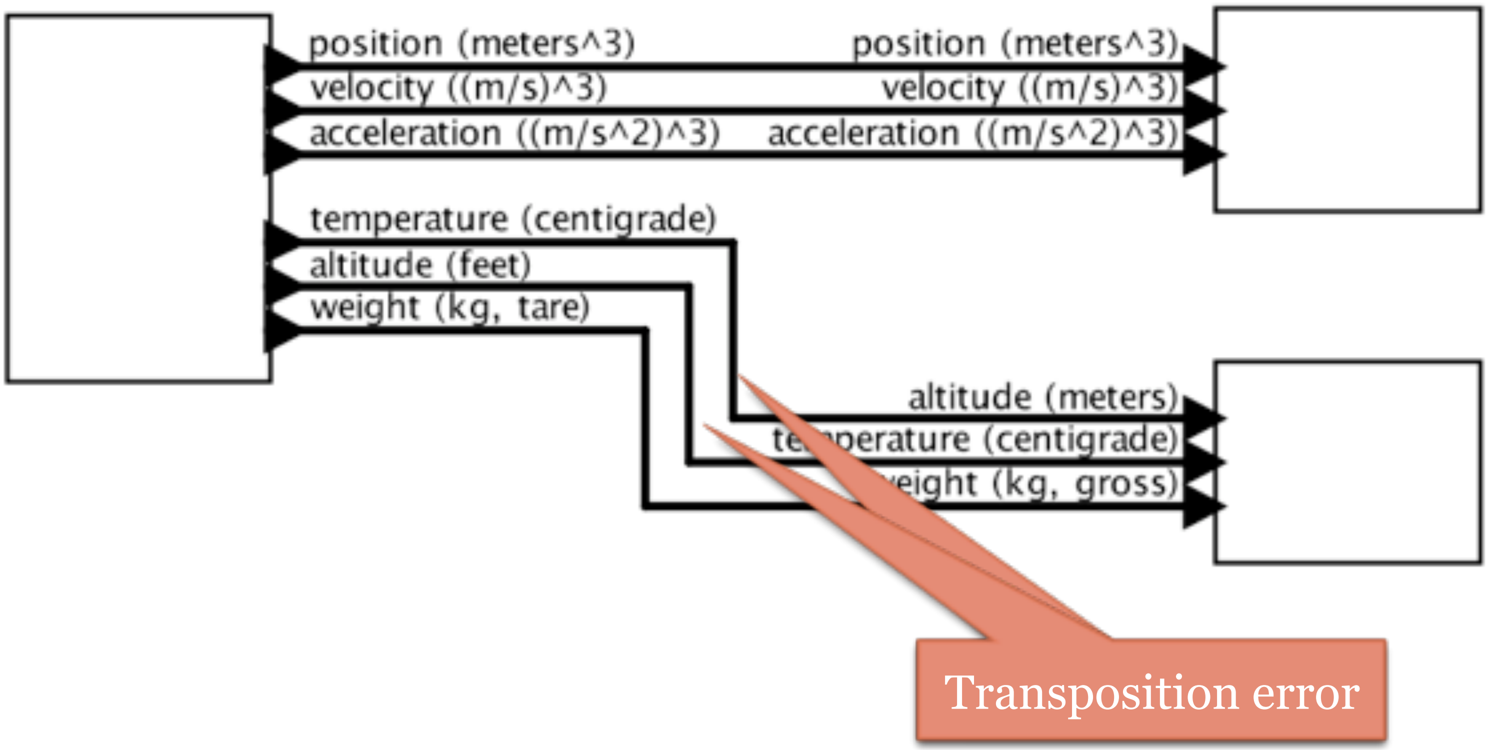
A Taxonomy of Modeling Issues



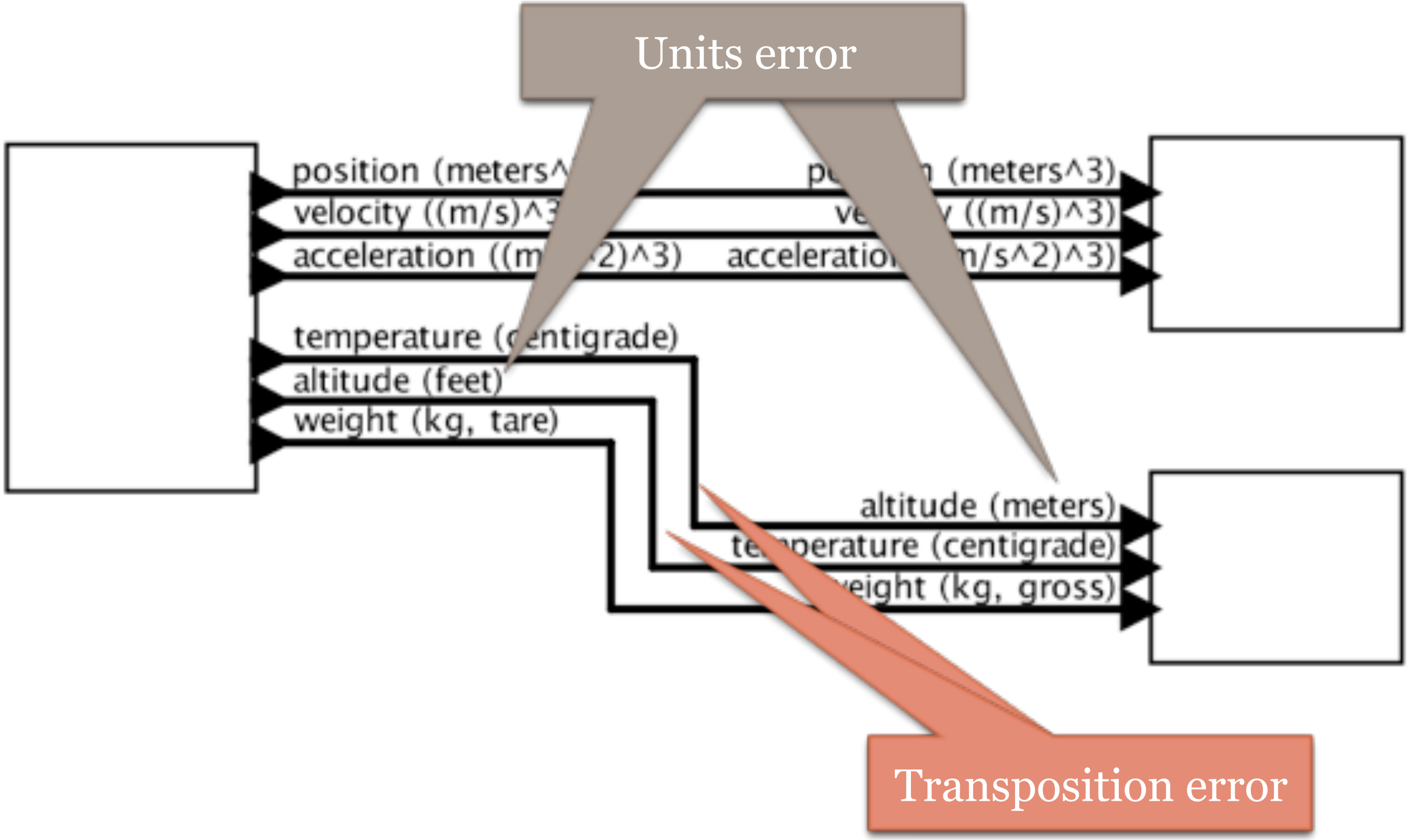
Model Composition Errors



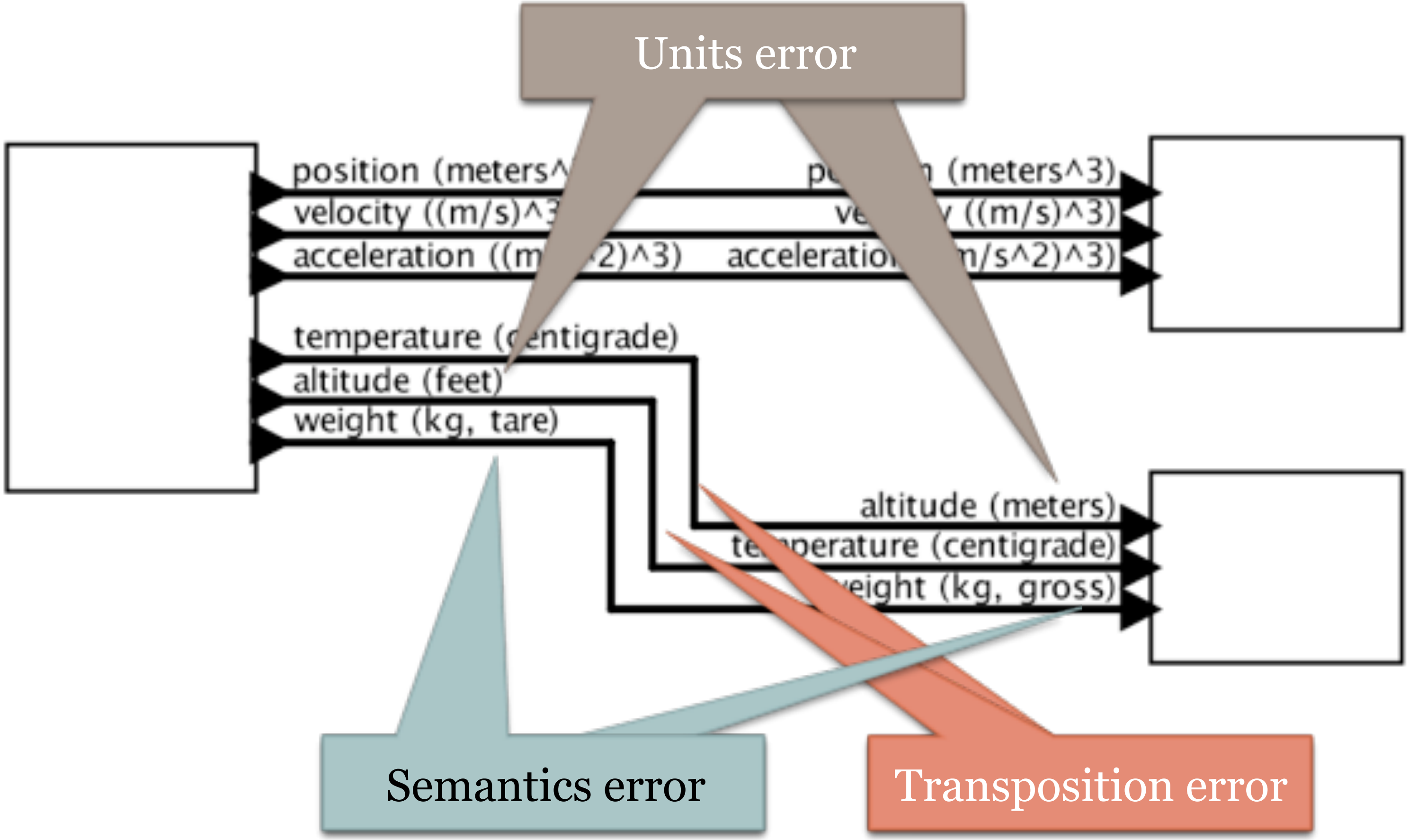
Model Composition Errors



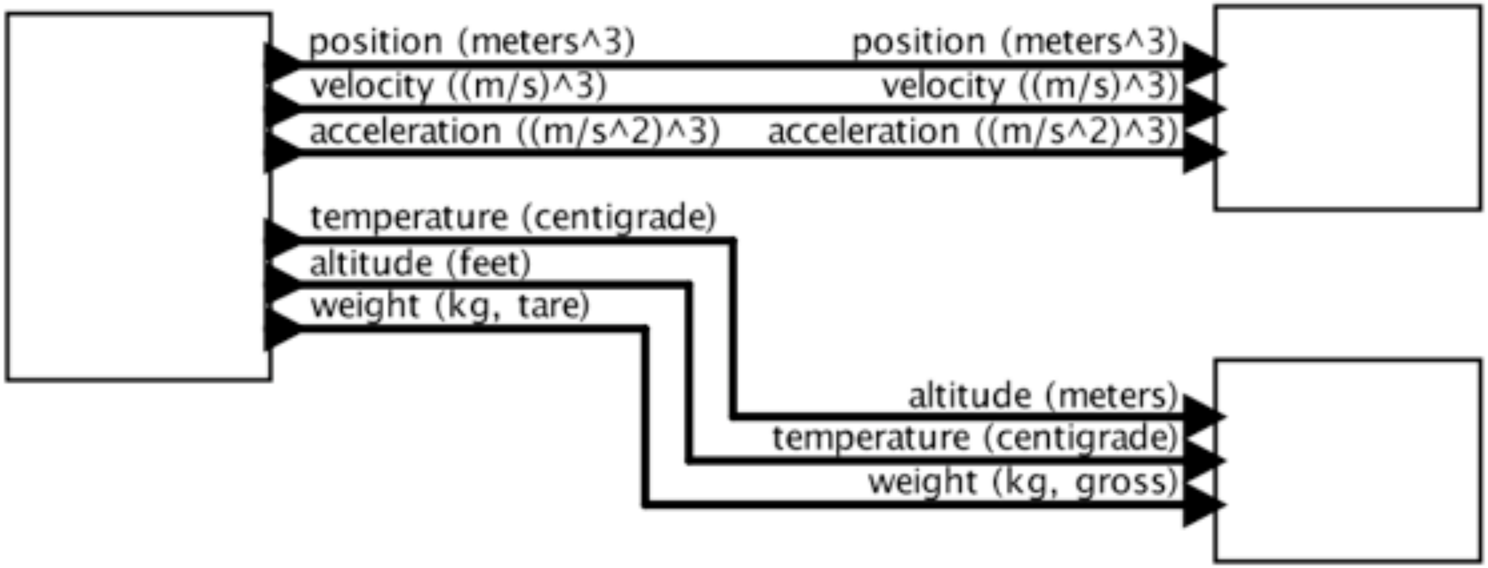
Model Composition Errors



Model Composition Errors



Our Goals

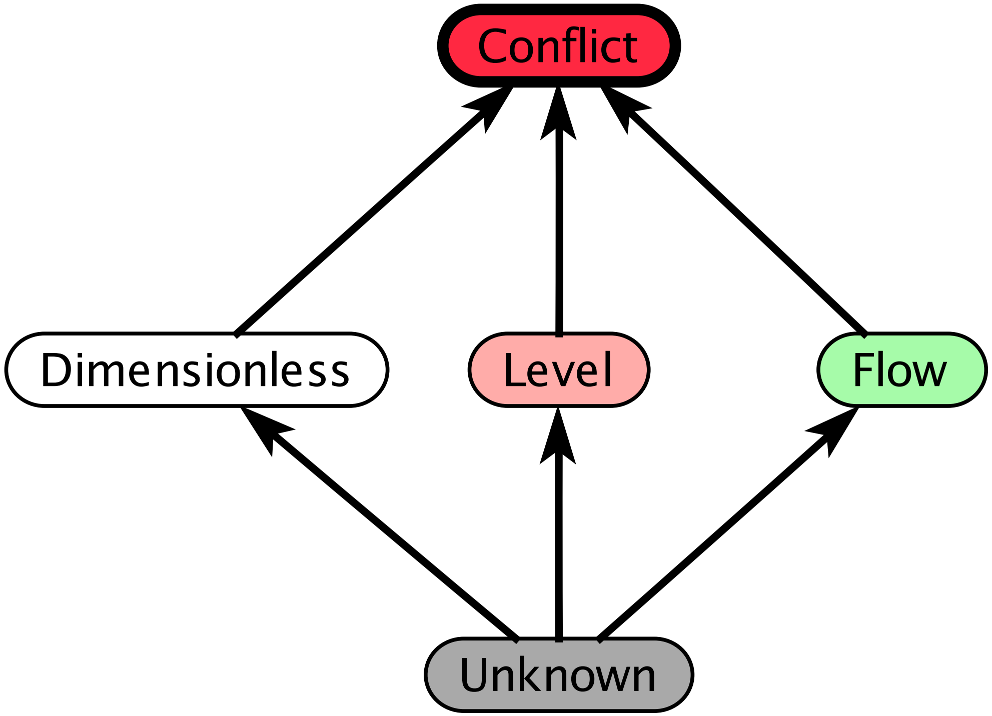


- Detect such interfacing errors
- Minimize the manual annotations required (use inference)
- Improve communication in engineering teams by augmenting interface definitions with semantic information

Outline

1. Existing (Finite) Ontology Analysis
2. Value-parametrized Ontologies
3. Recursive Ontologies

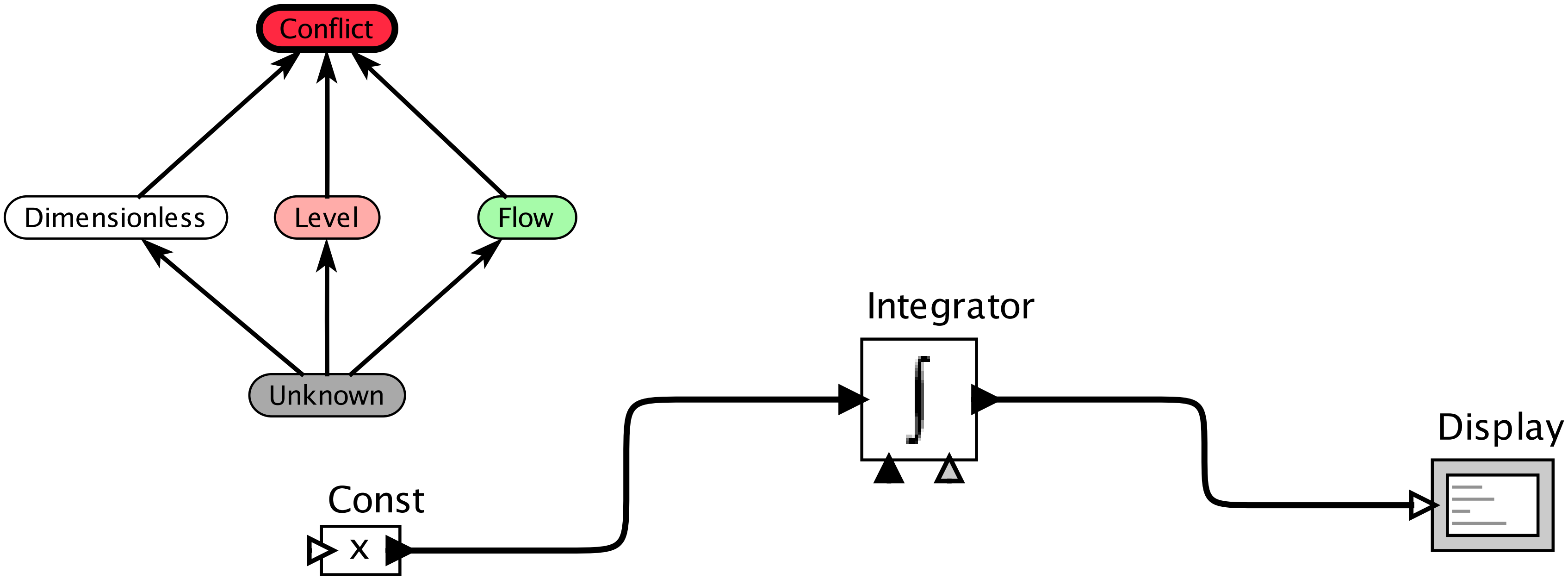
Domain-Specific Ontologies



Here is a *lattice* representing a simple dimension ontology.

- 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.

Inferring Concepts



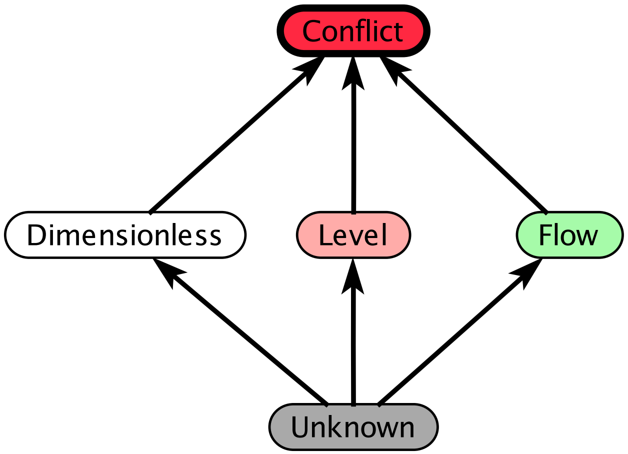
FuelDimensionSystemSolver

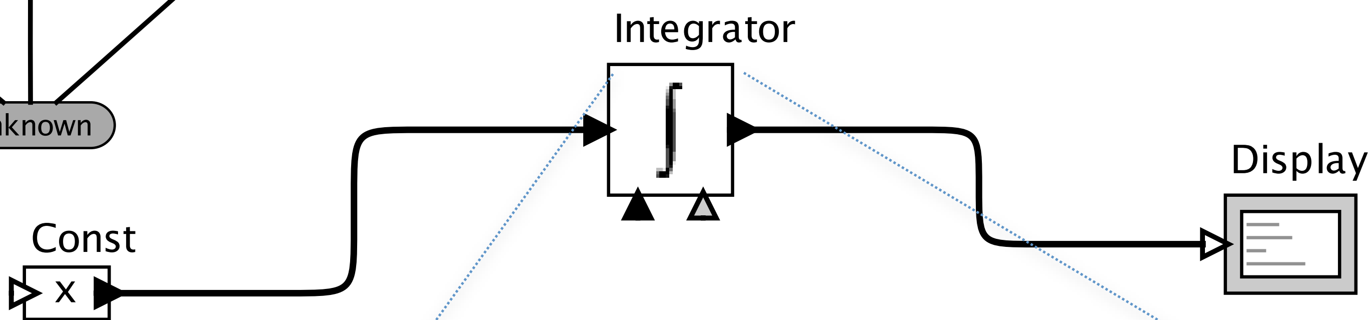
Double click to Apply Ontology

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

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

Inferring Concepts





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

FuelDimensionSystemSolver

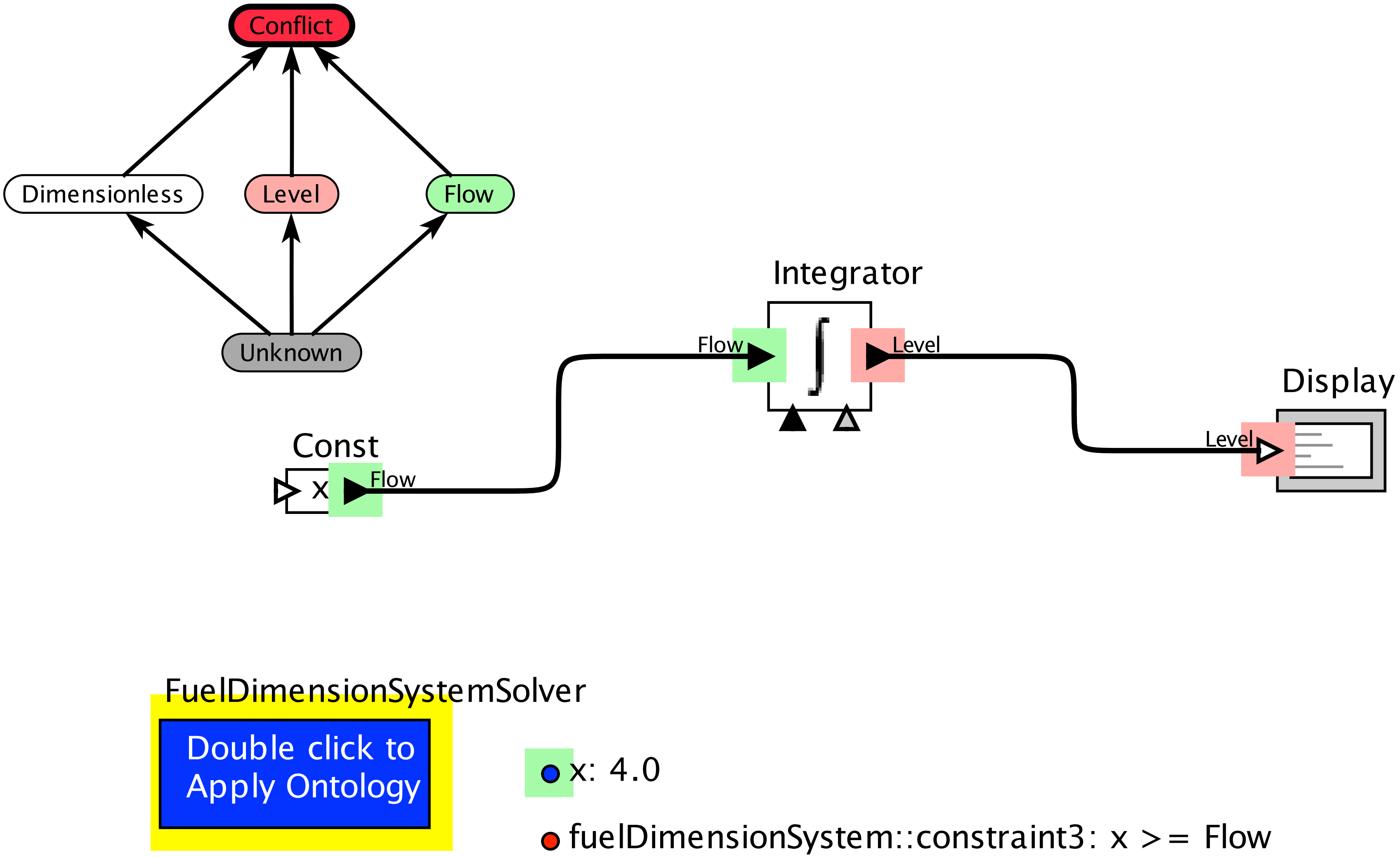
Double click to Apply Ontology

x: 4.0

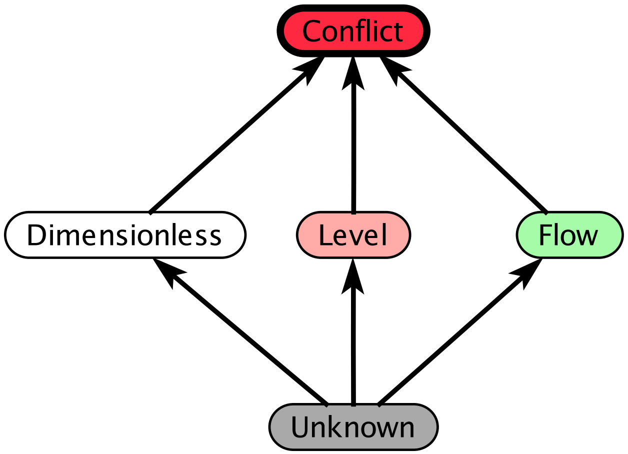
fuelDimensionSystem::constraint3: x >= Flow

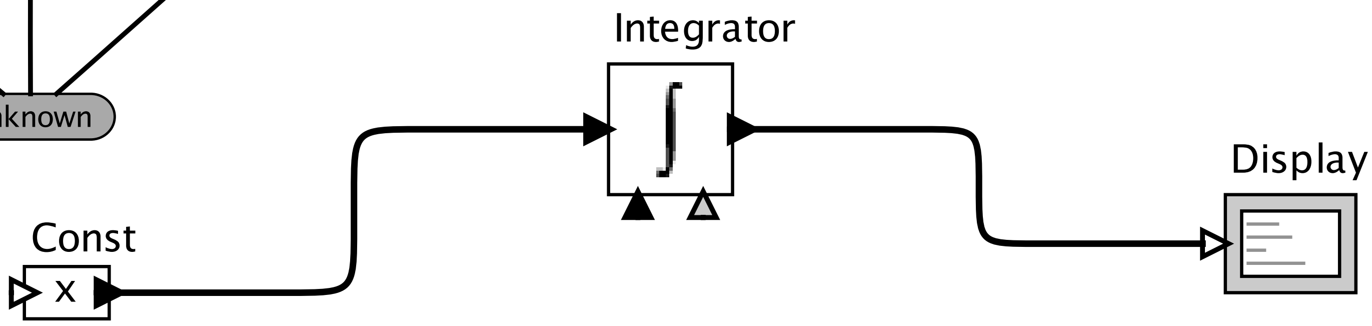
User-defined constraints added (in as few places as possible)

Inferring Concepts



Inferring Errors



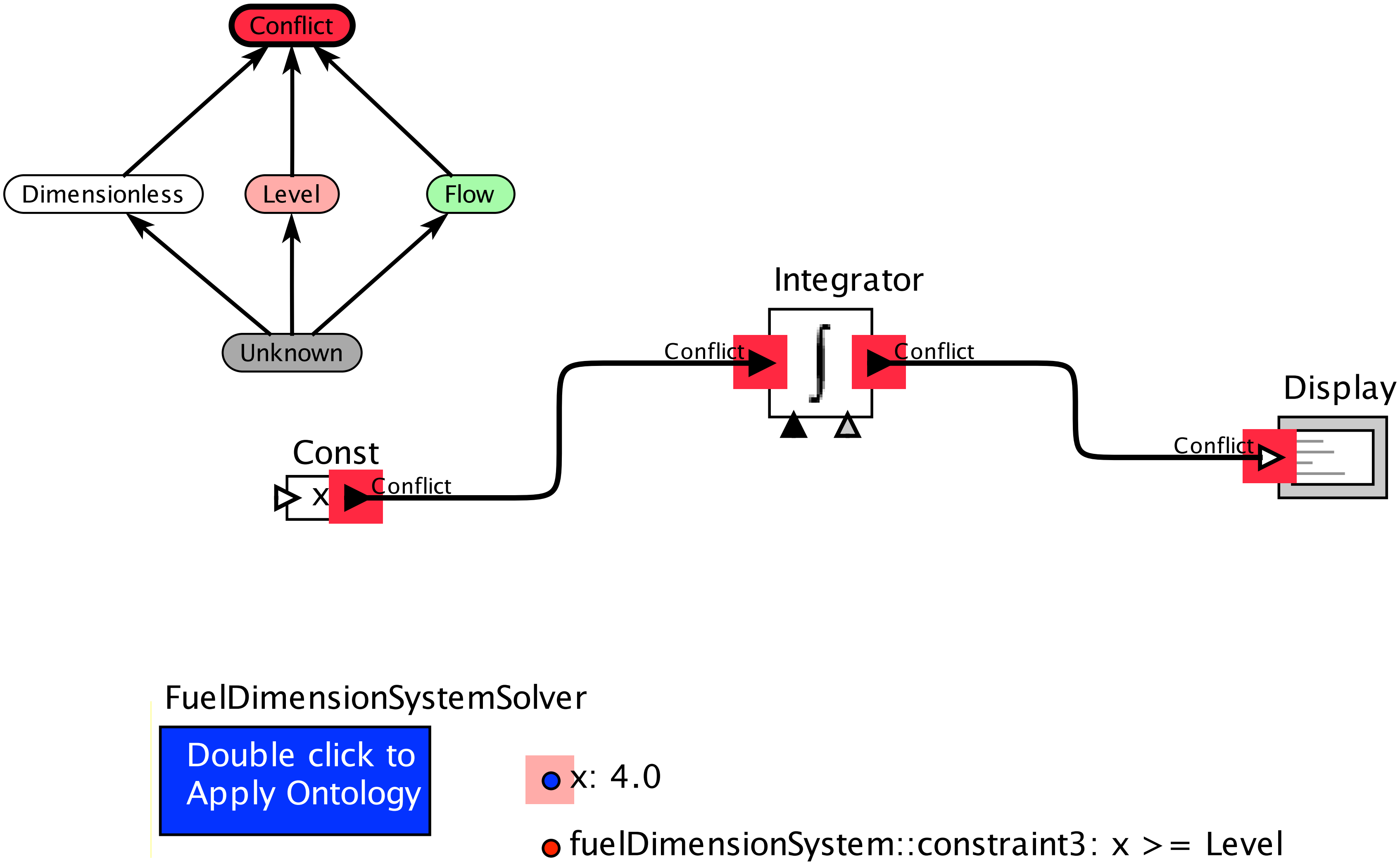


FuelDimensionSystemSolver

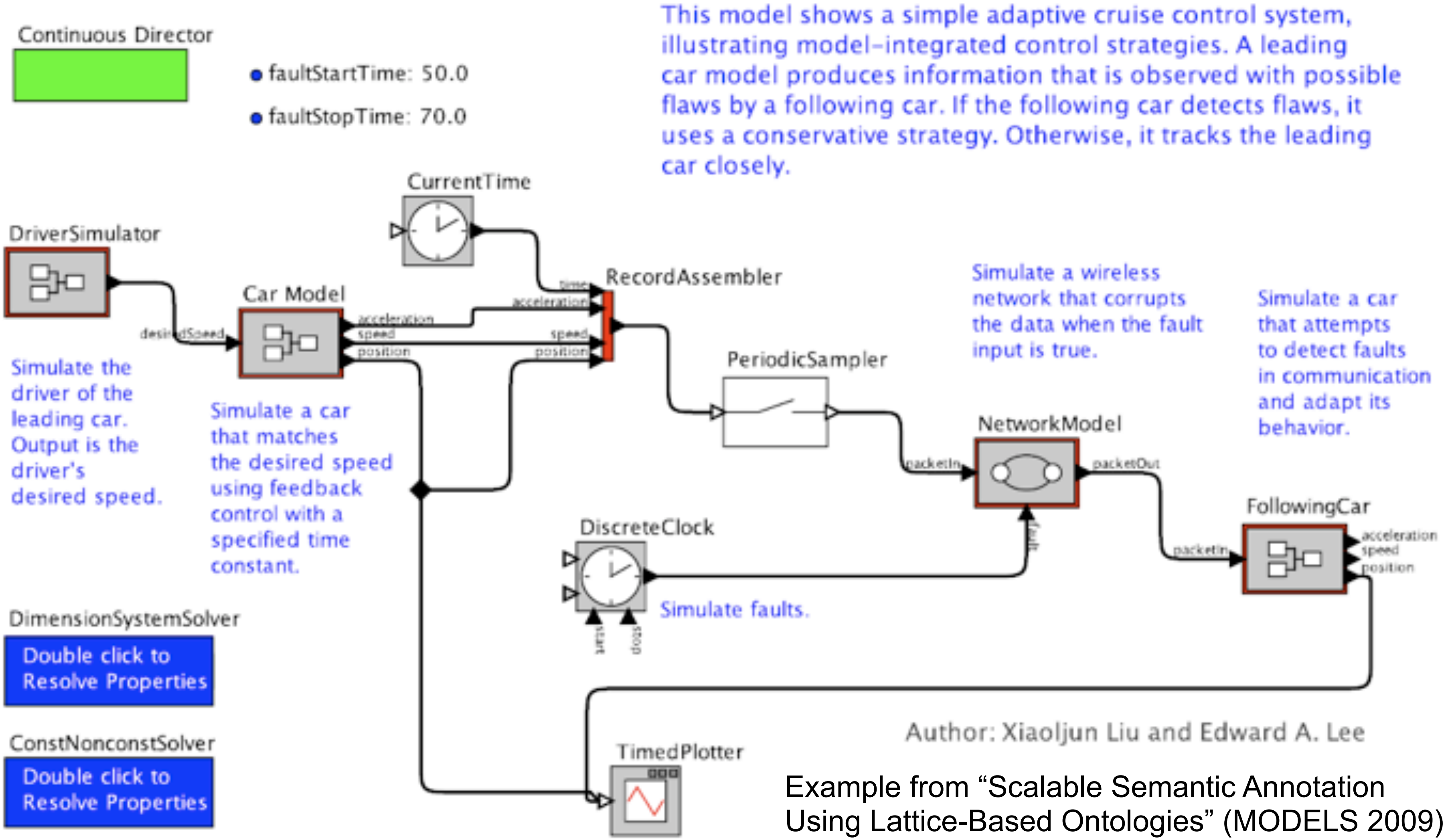
Double click to Apply Ontology

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

Inferring Errors

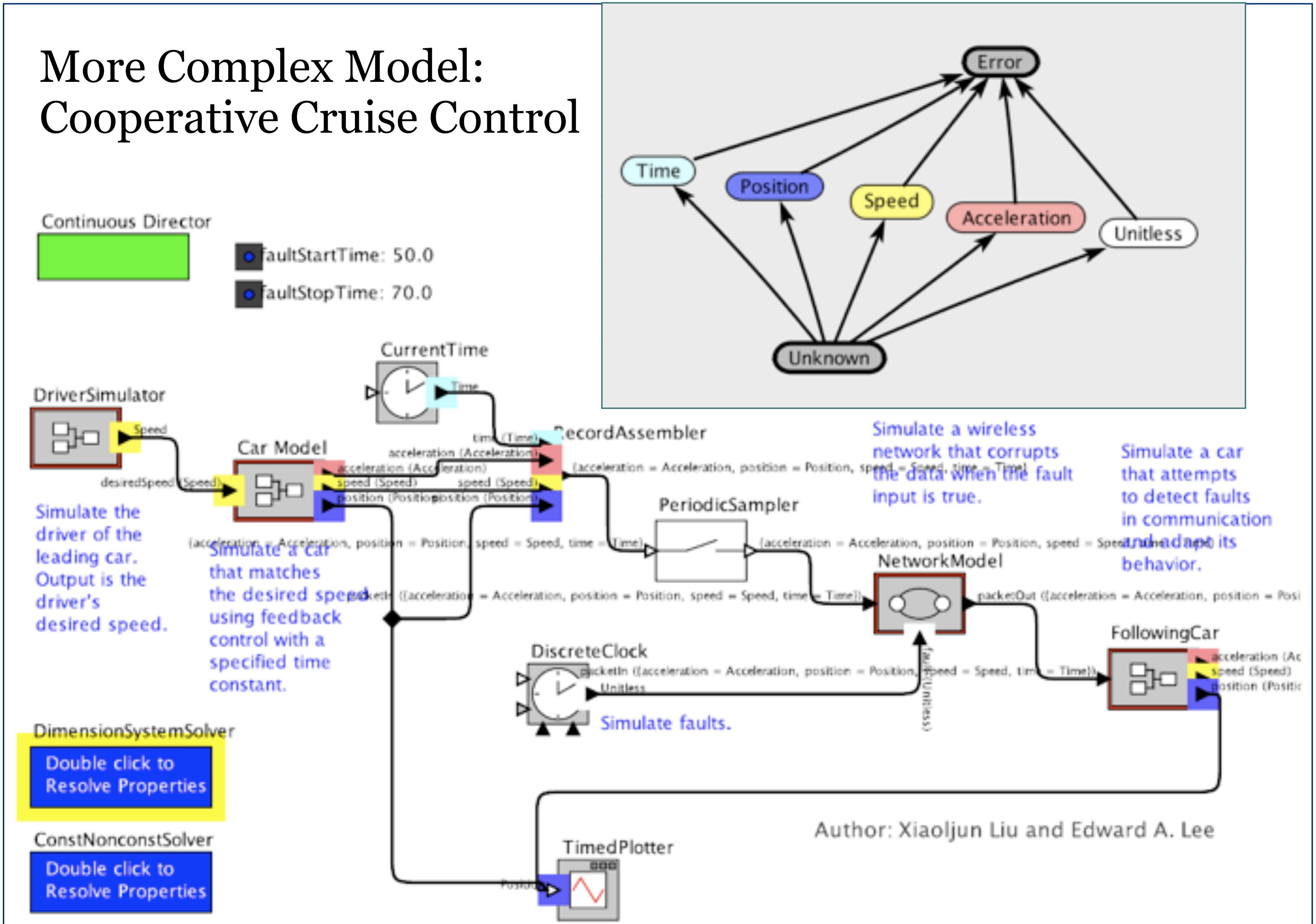


More Complex Model: Cooperative Cruise Control

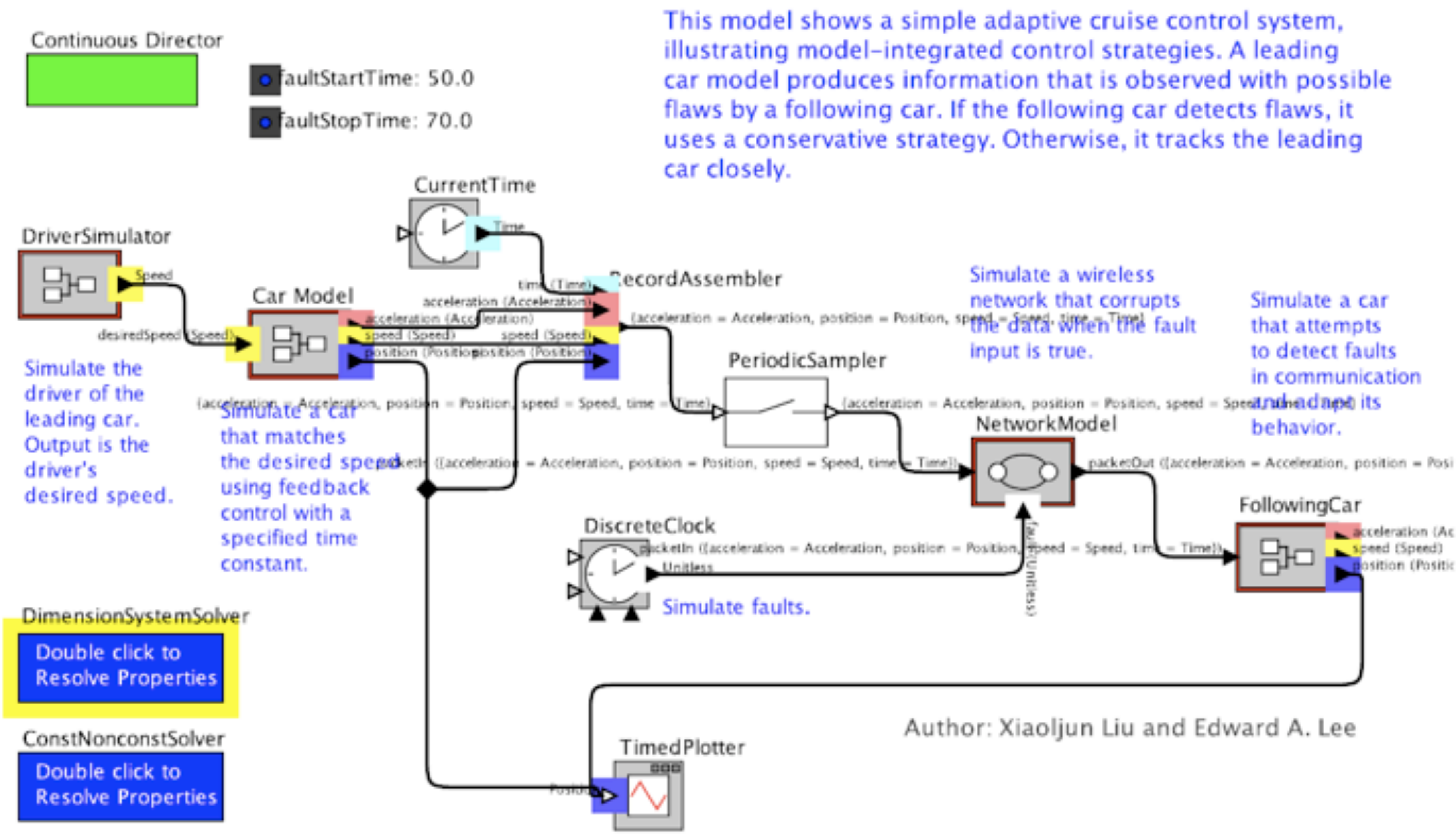


Example from “Scalable Semantic Annotation Using Lattice-Based Ontologies” (MODELS 2009)

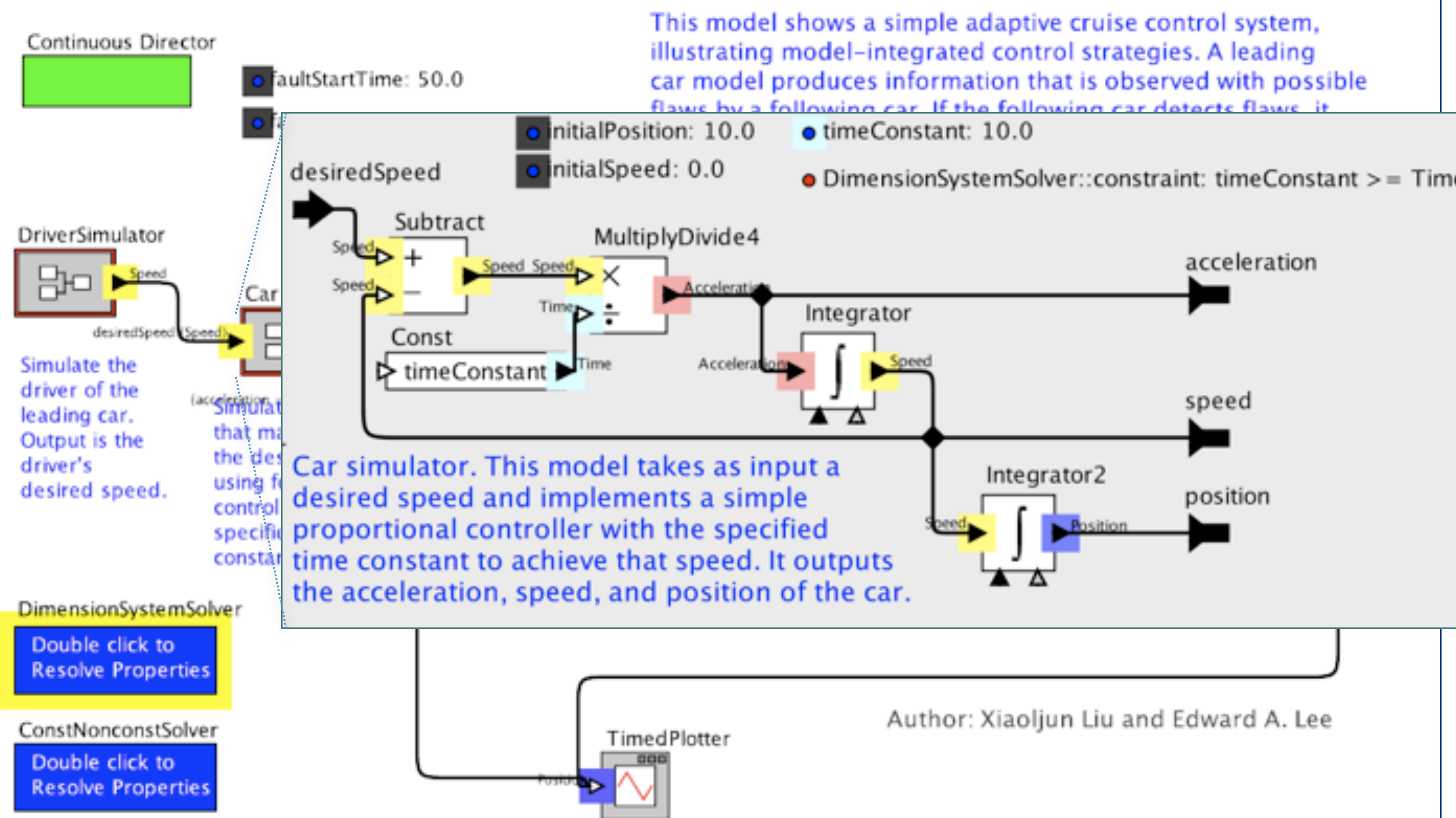
More Complex Model: Cooperative Cruise Control



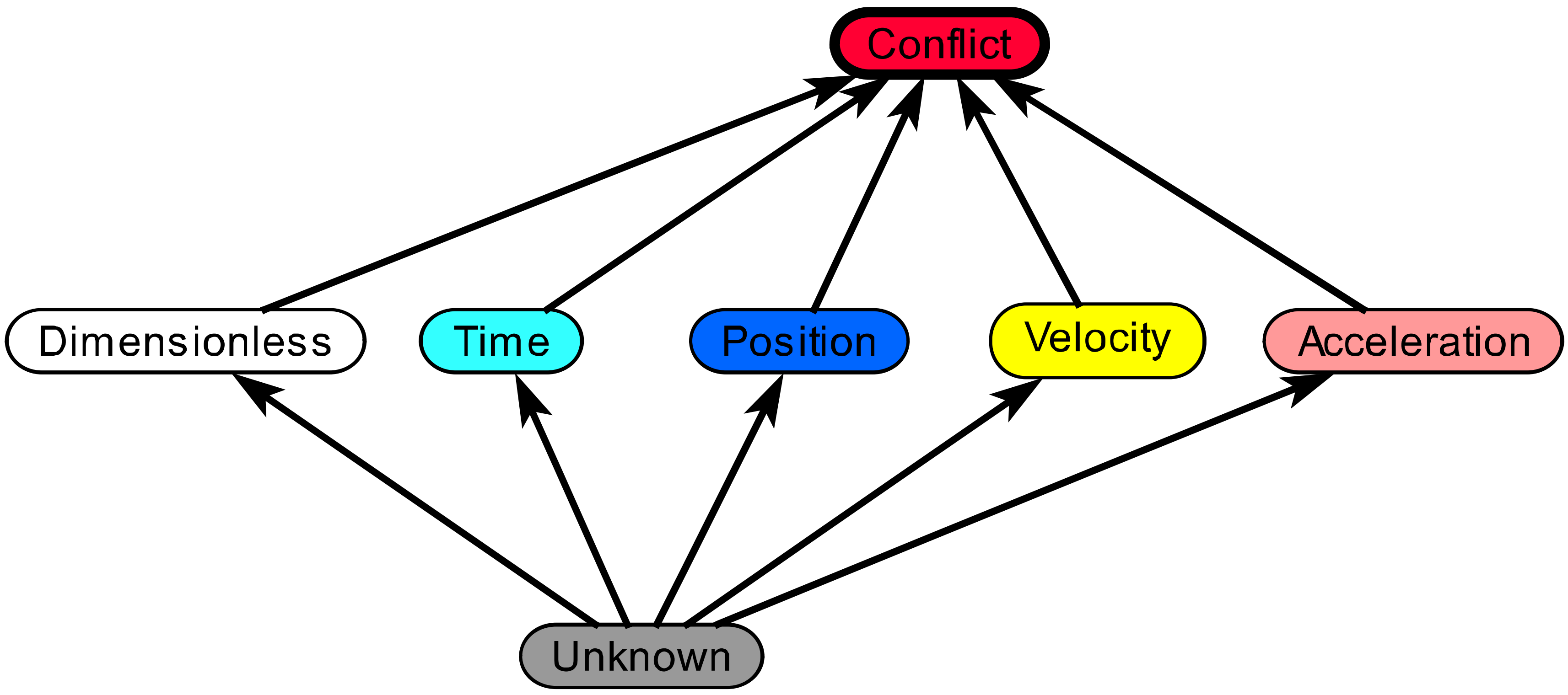
More Complex Model: Cooperative Cruise Control



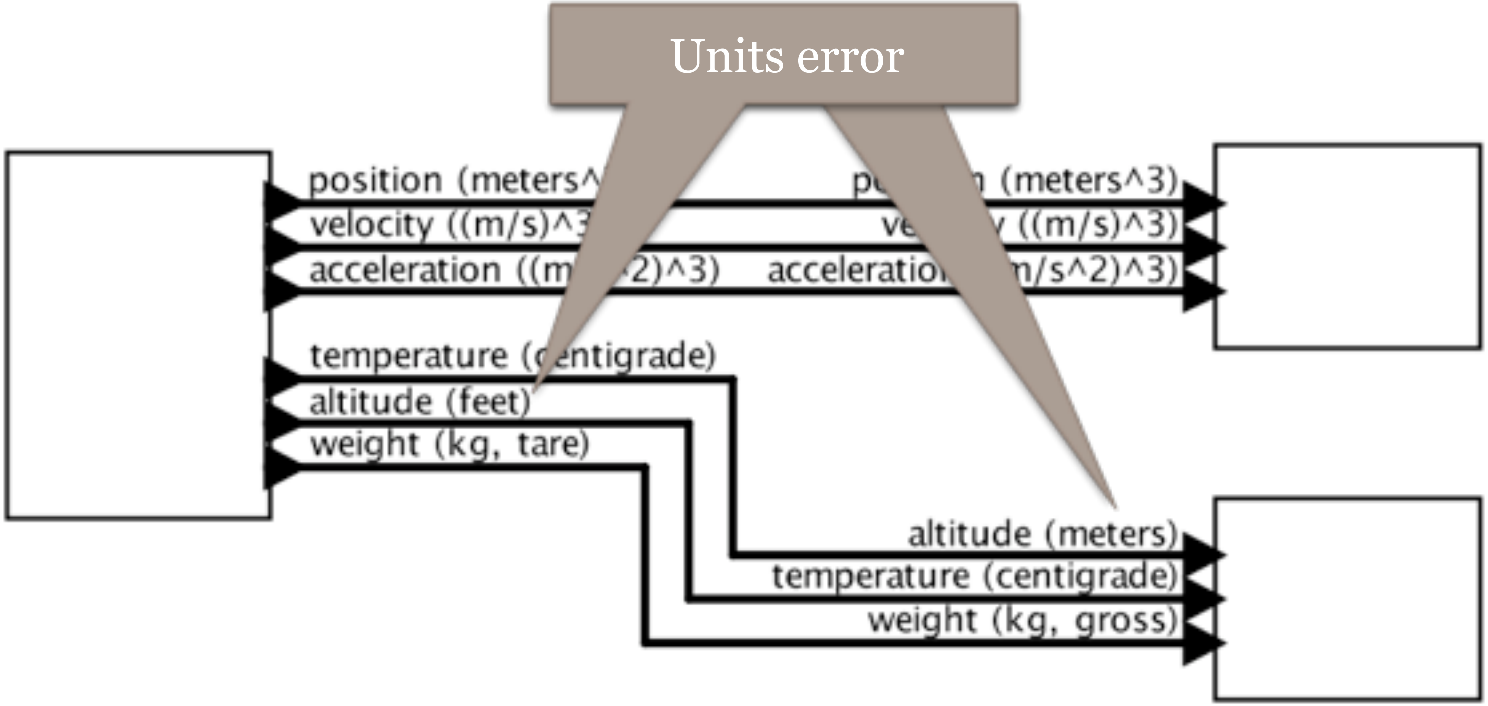
More Complex Model: Cooperative Cruise Control



Why is this insufficient?



Units Errors



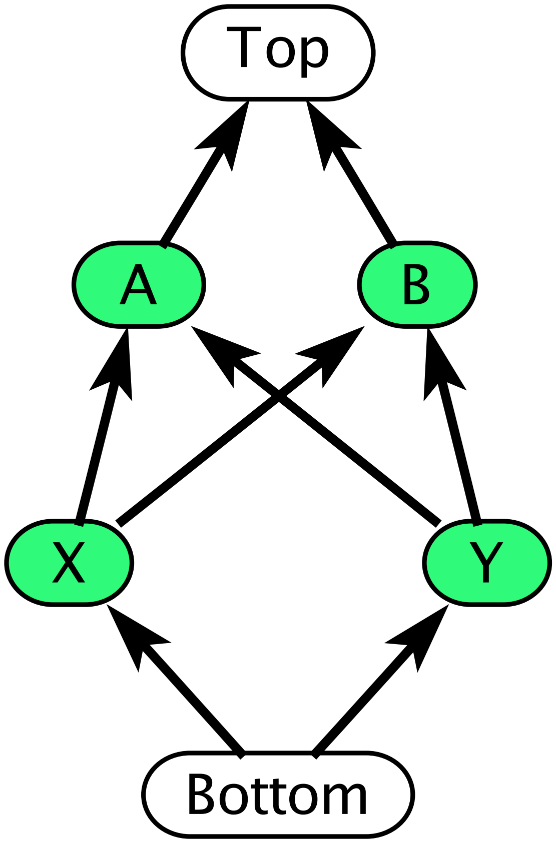
Definitions

Least Upper Bound/ LUB(S): The least x, if it exists, such that

$$\forall s \in S,\ x \geq s.$$

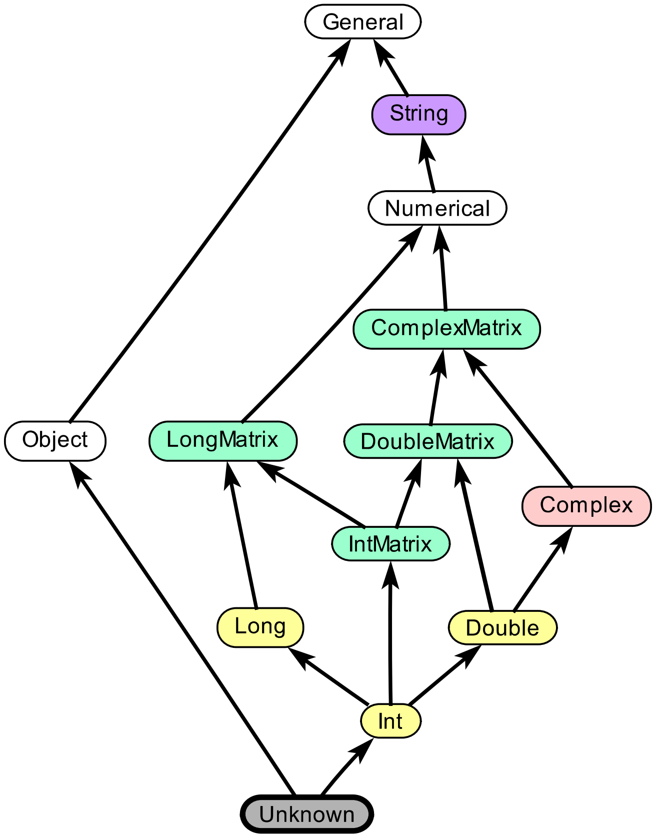
Greatest Lower Bound/GLB(S): The greatest x, if it exists, such that

$$\forall s \in S,\ x \leq s.$$



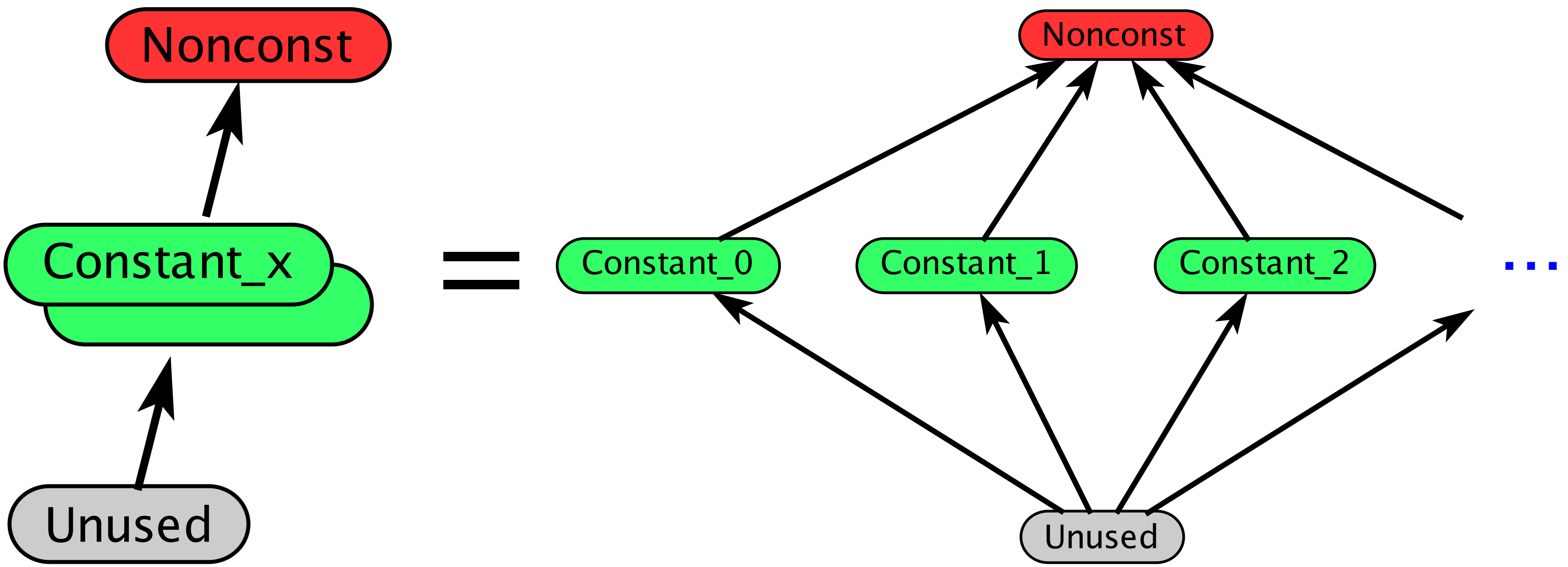
Definitions

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

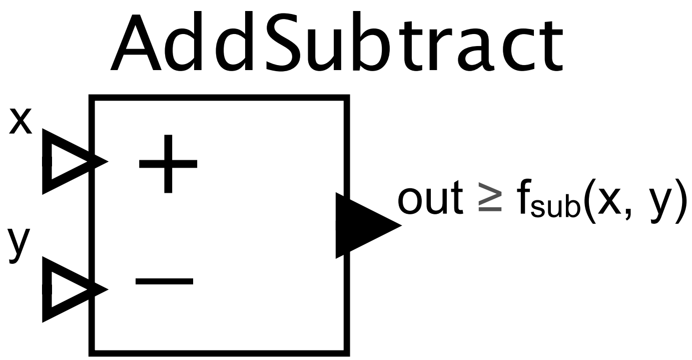


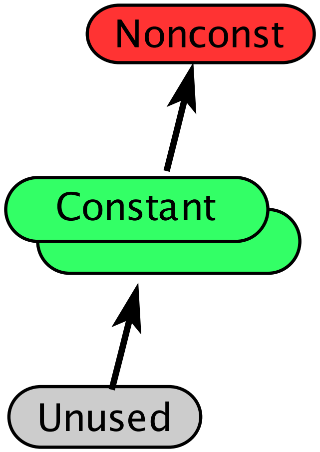
Value-parametrized Concepts

Concepts with values can be useful.



Value-dependent Constraints

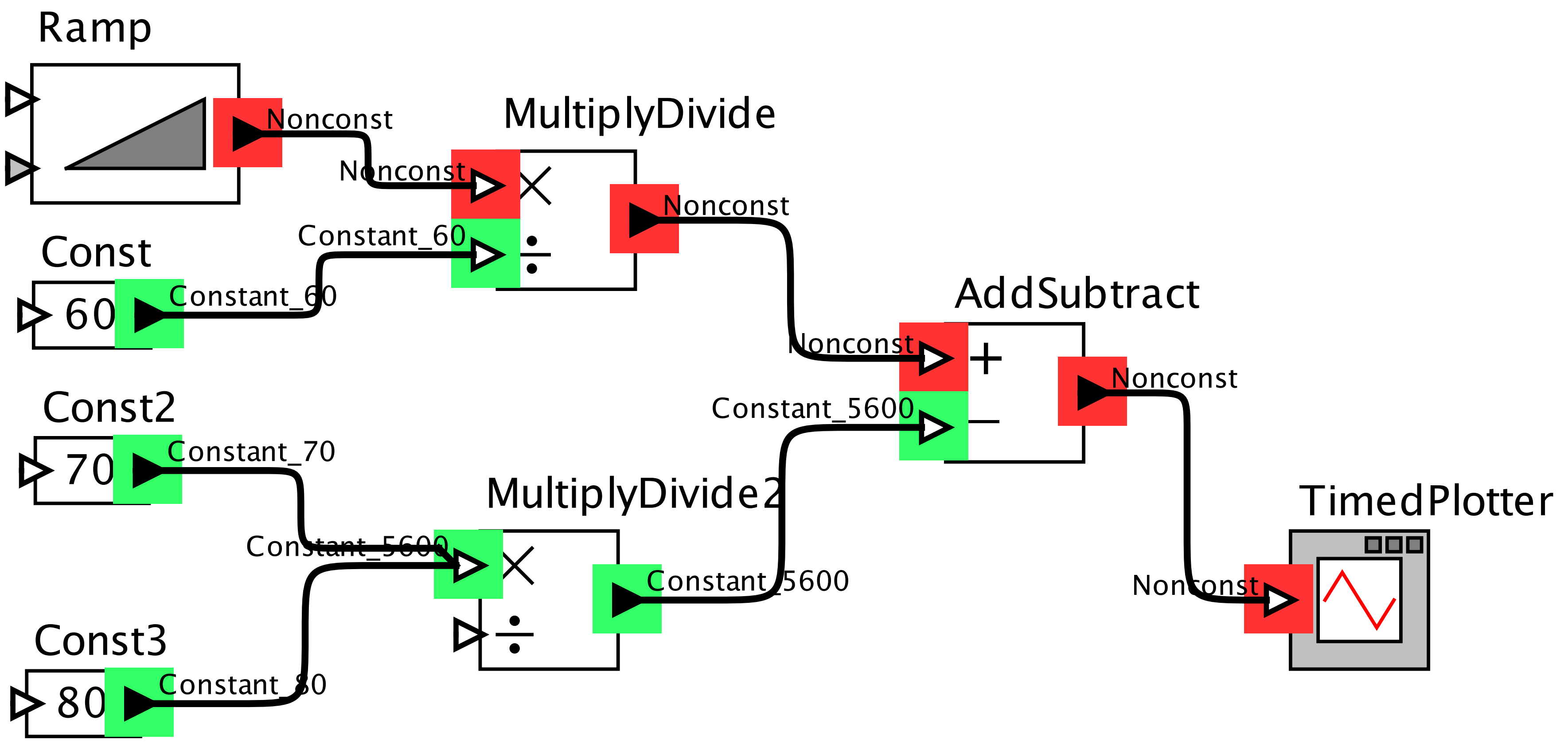




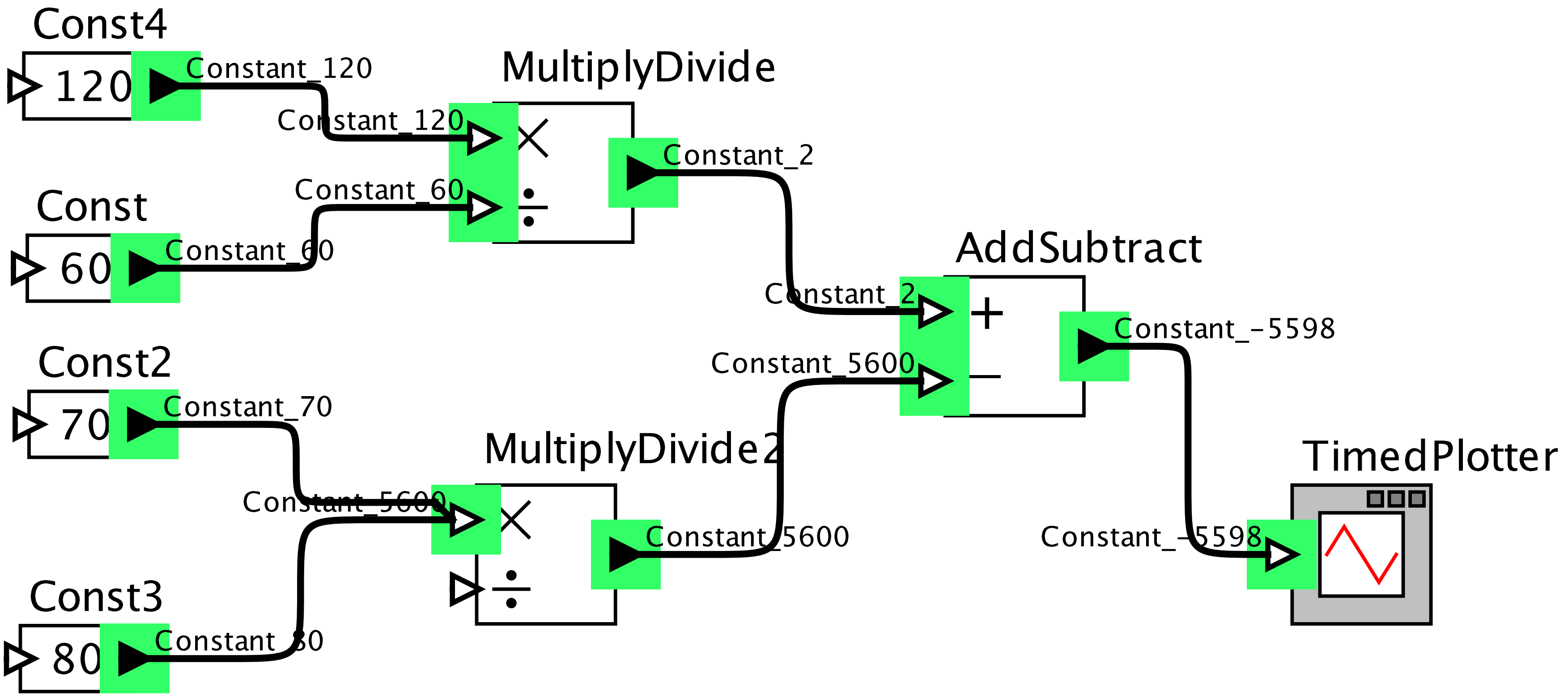
$f_{sub}(x, y) =$

$$\begin{cases} \text{Unused} & \text{if } x = \text{Unused or } y = \text{Unused} \\ \text{Constant}(c_x - c_y) & \text{if } x = \text{Constant}(c_x) \\ & \text{ and } y = \text{Constant}(c_y) \\ \text{Nonconst} & \text{otherwise} \end{cases}$$

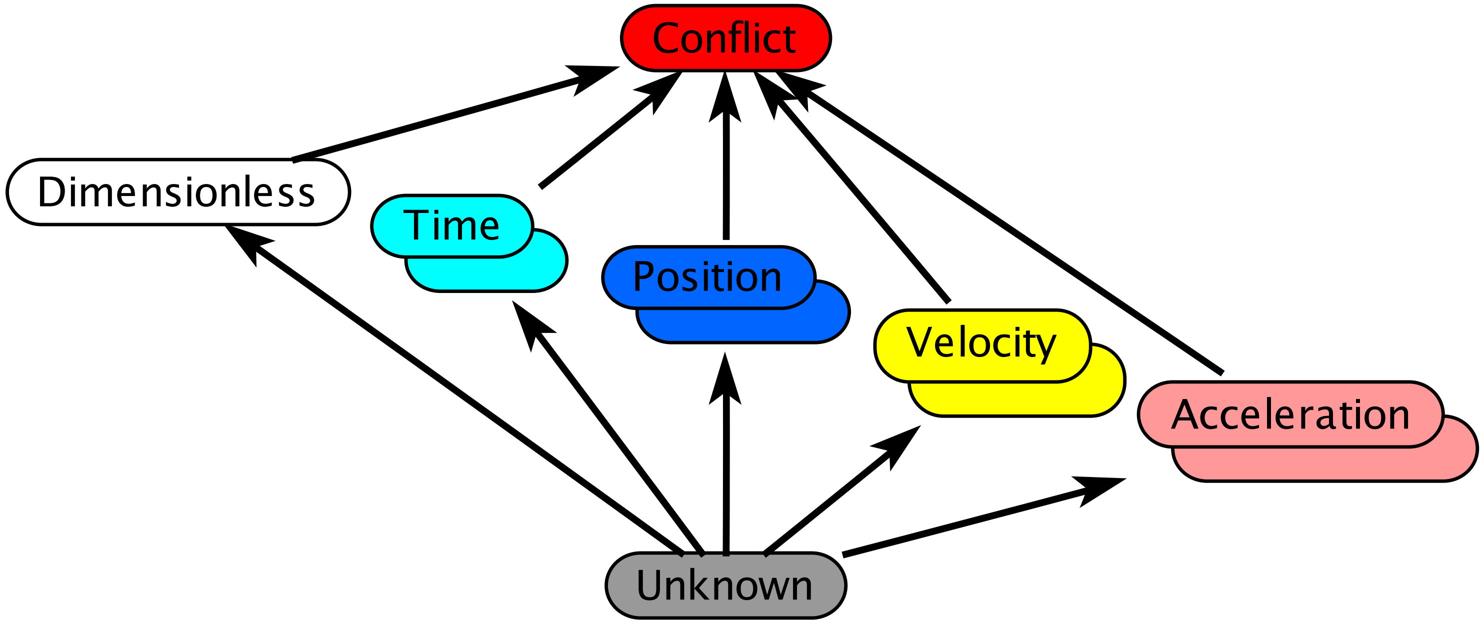
Constant Propagation Analysis



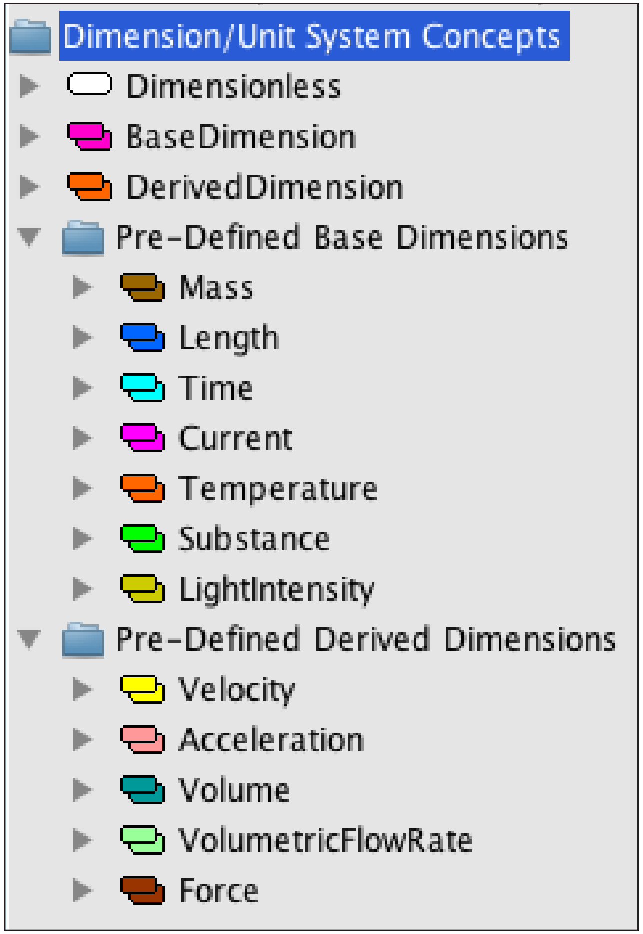
Constant Propagation Analysis



Full Unit Systems



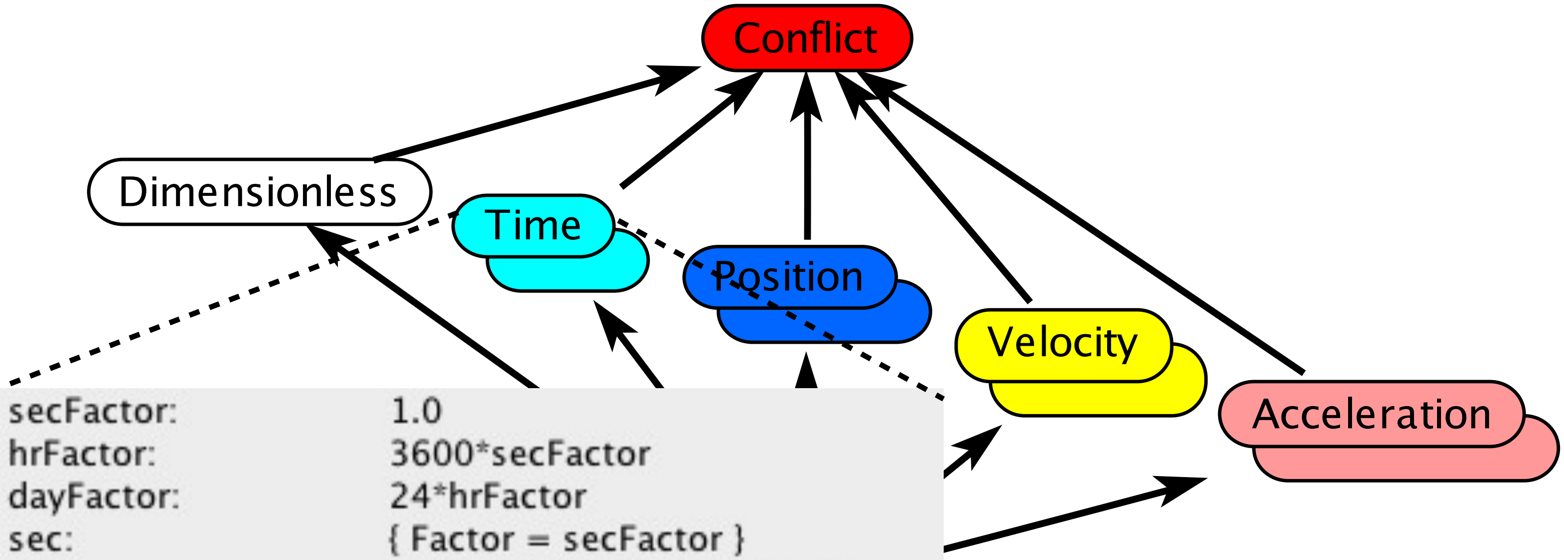
Units Library



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

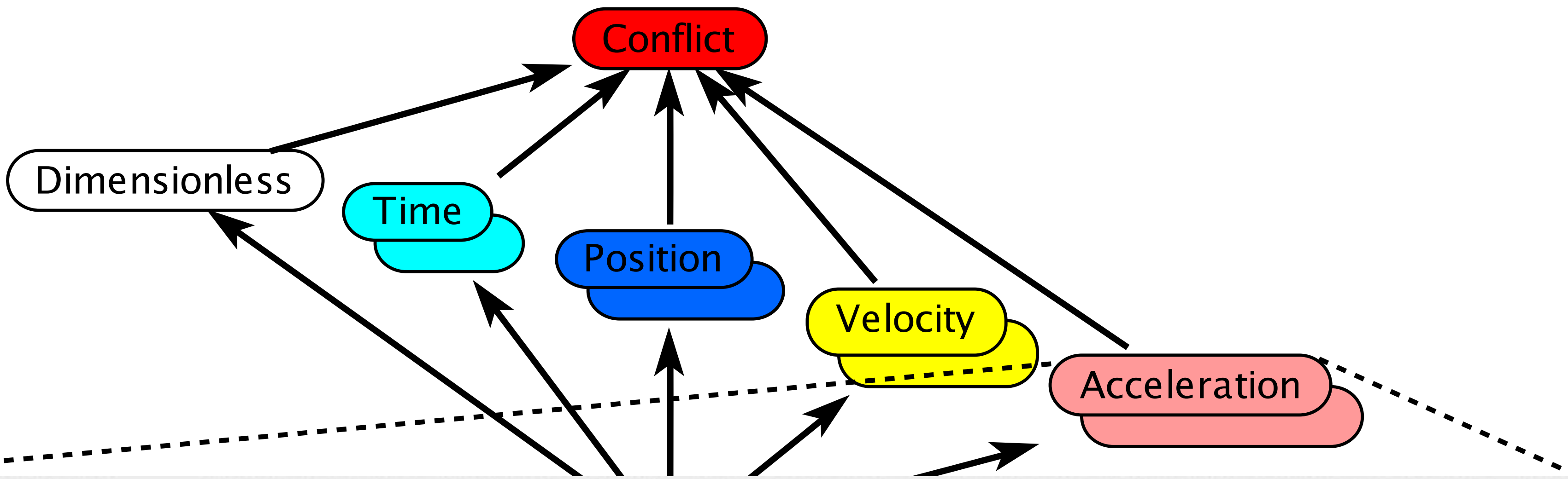
Contains commonly used units of mass, time, length, force, etc.

Base Concepts



secFactor:	1.0
hrFactor:	3600*secFactor
dayFactor:	24*hrFactor
sec:	{ Factor = secFactor }
ms:	{ Factor = 0.001*secFactor }
us:	{ Factor = 1E-06*secFactor }
ns:	{ Factor = 1E-09*secFactor }
minute:	{ Factor = 60*secFactor }
hr:	{ Factor = hrFactor }
day:	{ Factor = dayFactor }
yrCalendar:	{ Factor = 365.2425*dayFactor }
yrSidereal:	{ Factor = 31558150*secFactor }
yrTropical:	{ Factor = 31556930*secFactor }

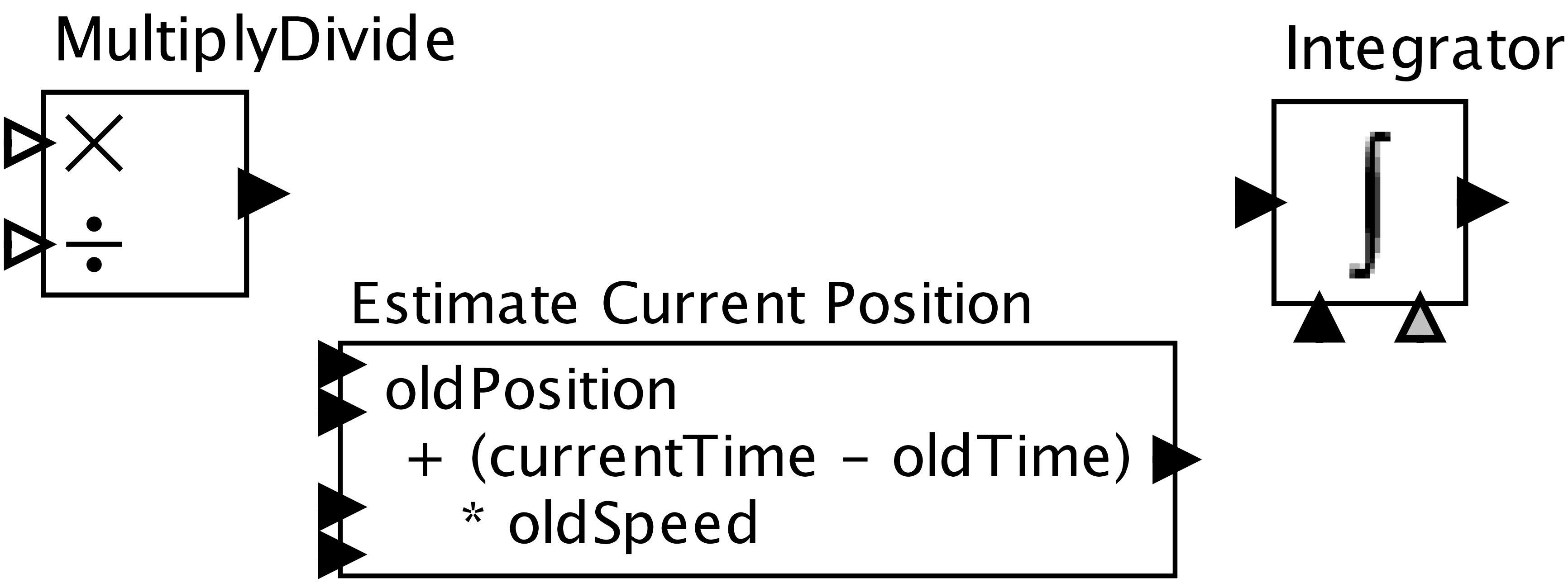
Derived Concepts



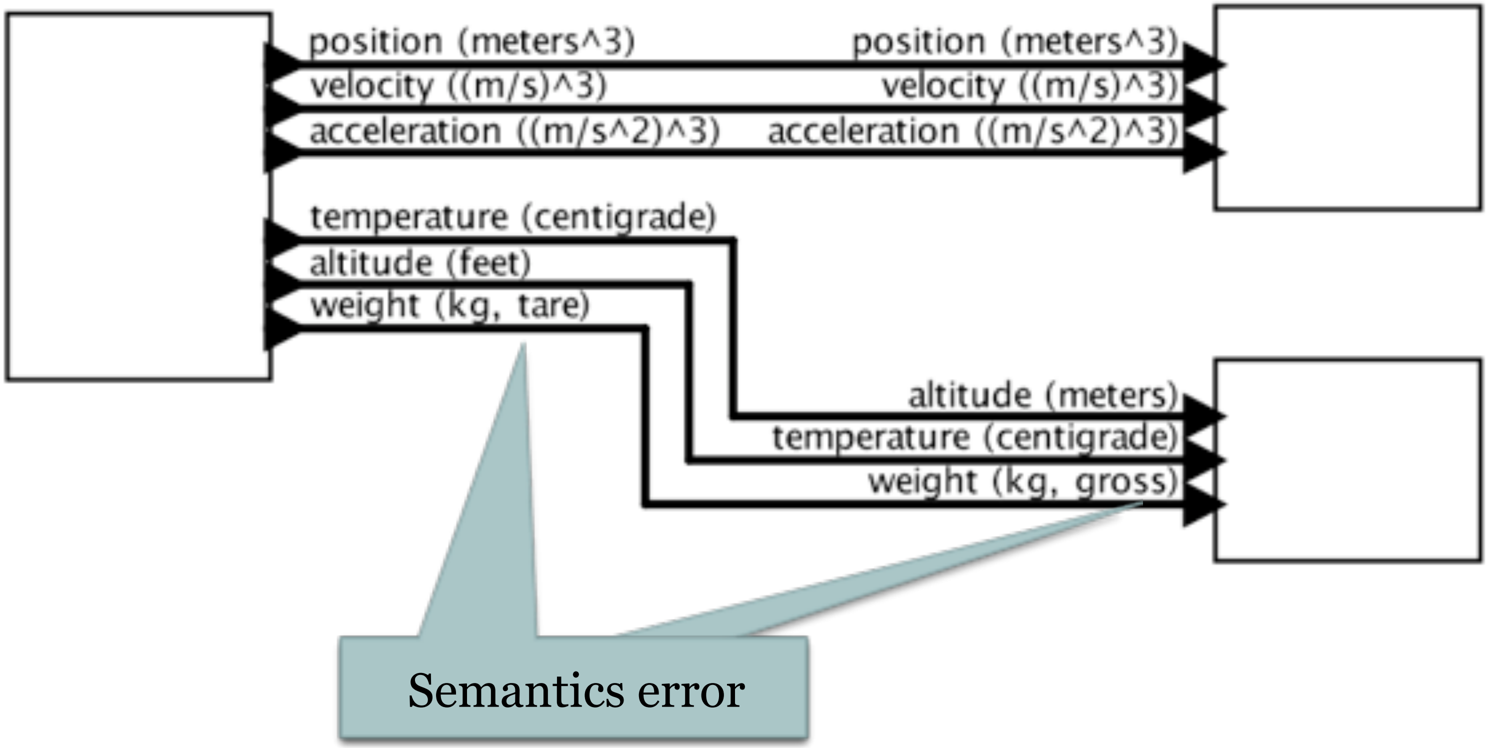
dimensionArray:	{ {Dimension = "LengthConcept", Exponent = 1}, {Dimension = "TimeConcept", Exponent = -2} }
LengthConcept:	Position
TimeConcept:	Time
m_per_sec2:	{ LengthConcept = {"m"}, TimeConcept = {"sec", "sec"} }
cm_per_sec2:	{ LengthConcept = {"cm"}, TimeConcept = {"sec", "sec"} }
ft_per_sec2:	{ LengthConcept = {"ft"}, TimeConcept = {"sec", "sec"} }
kph_per_sec:	{ LengthConcept = {"km"}, TimeConcept = {"hr", "sec"} }
mph_per_sec:	{ LengthConcept = {"mi"}, TimeConcept = {"hr", "sec"} }

Automatically Inferred Constraints

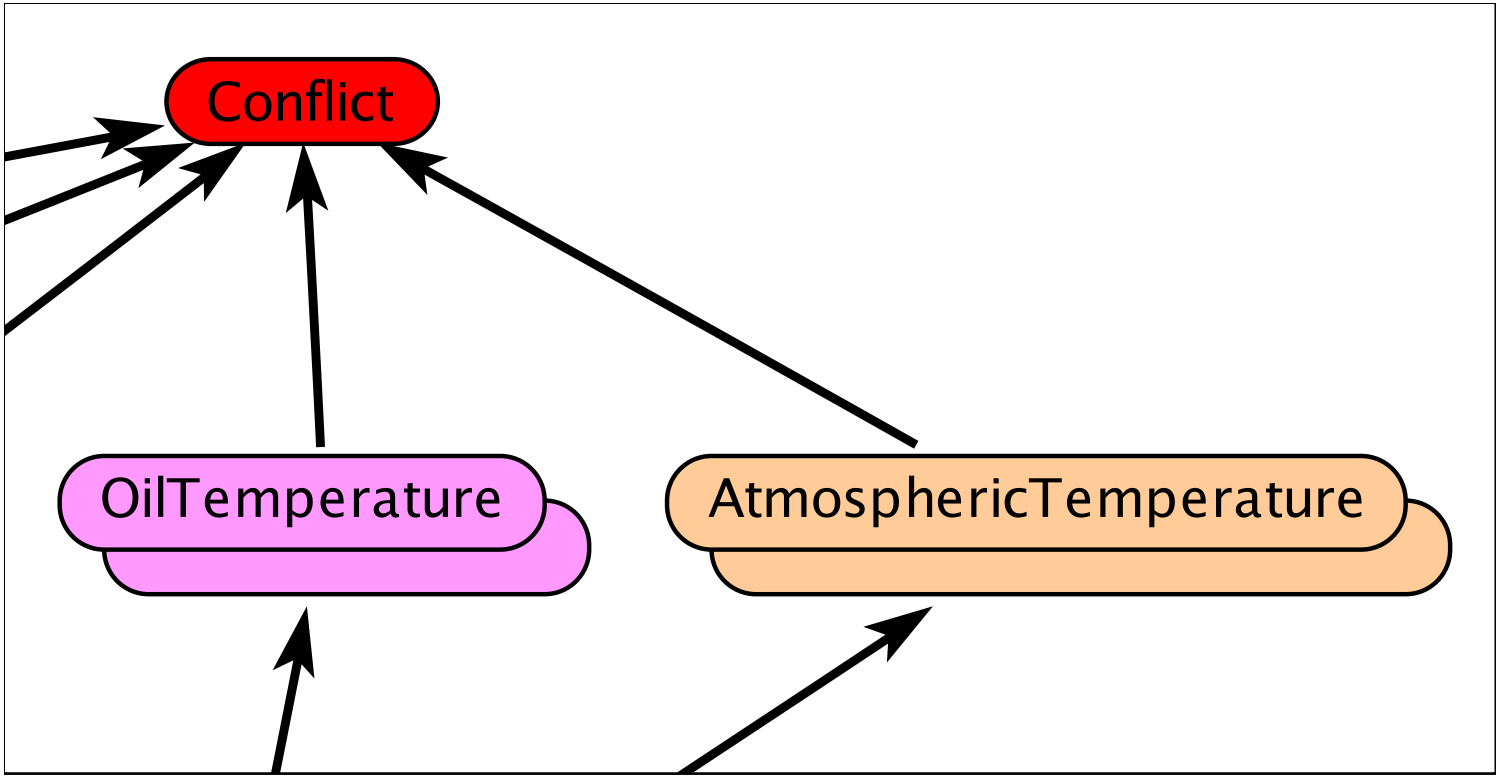
Multiply/divide and related constraints need not be specified for these unit systems.



Semantics Errors



Domain-specific Units



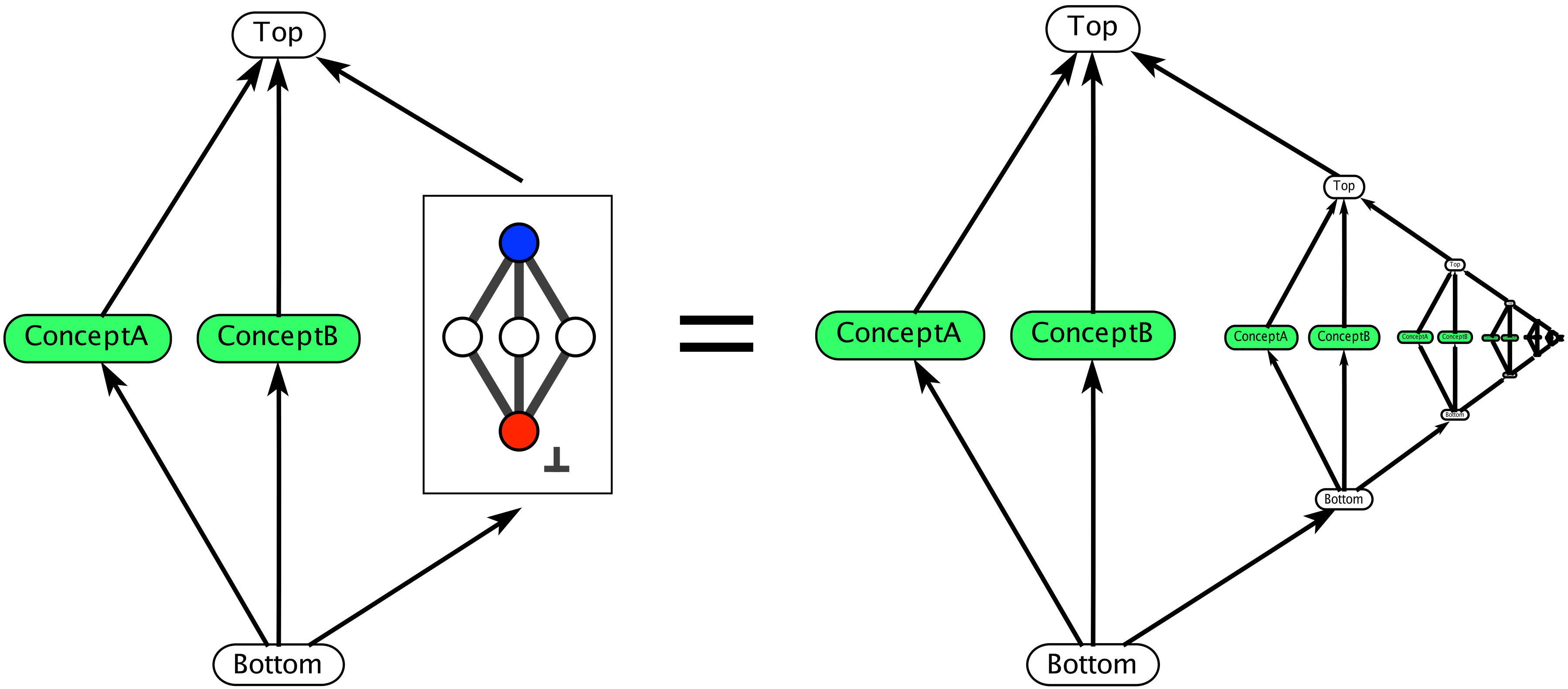
Record Types

```
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.

Thanks!

Questions?