Semantic Type Annotation

ActionWebs Meeting Presentation

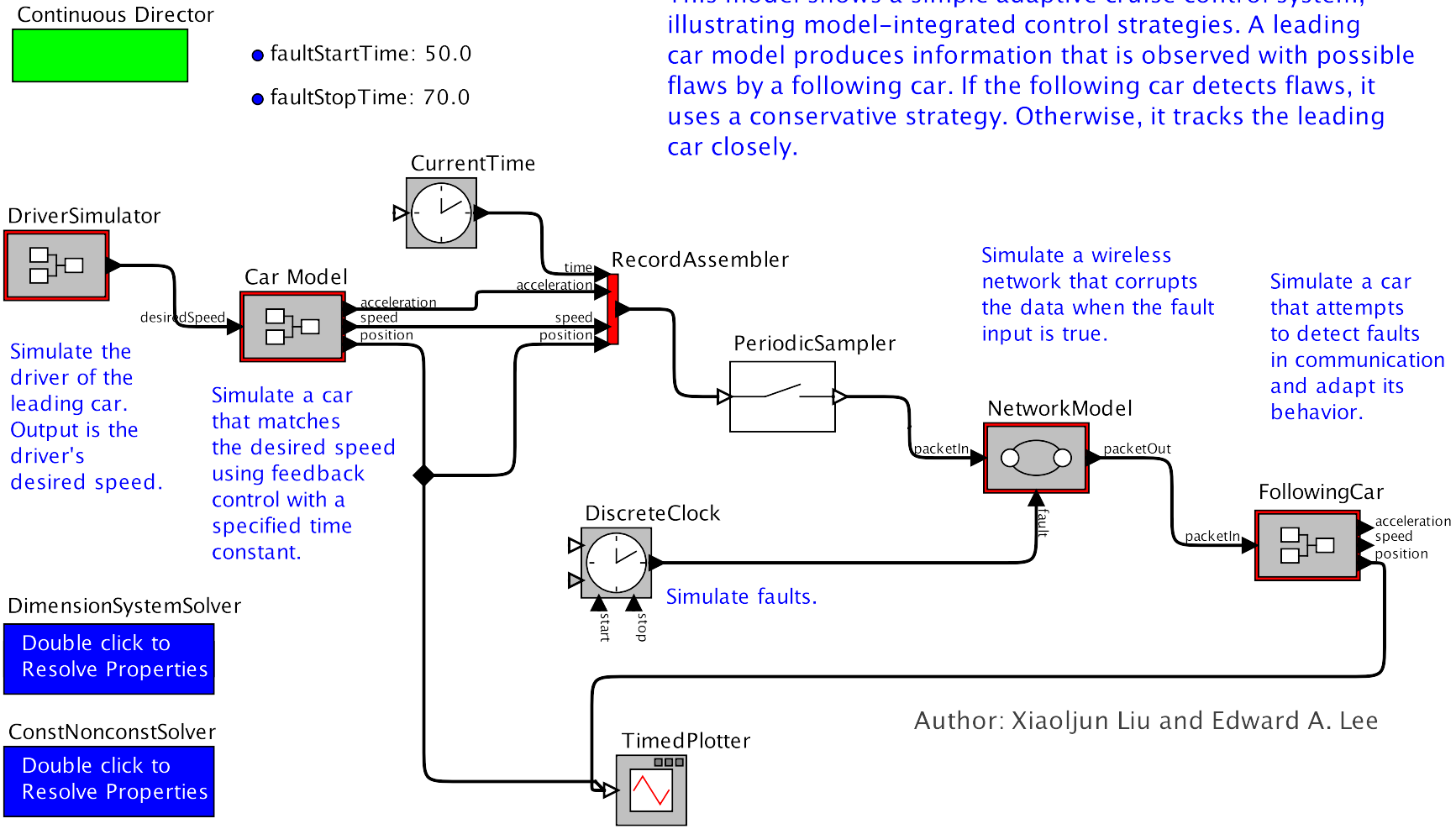
Ben Lickly[1] Man-Kit Leung[1] Thomas Mandl[2] Edward A. Lee[1]
Elizabeth Latronico[2] Charles Shelton[2] Stavros Tripakis[1]

[1]University of California, Berkeley

[2]Bosch Research, LLC

December 17, 2009

Actor-oriented Models

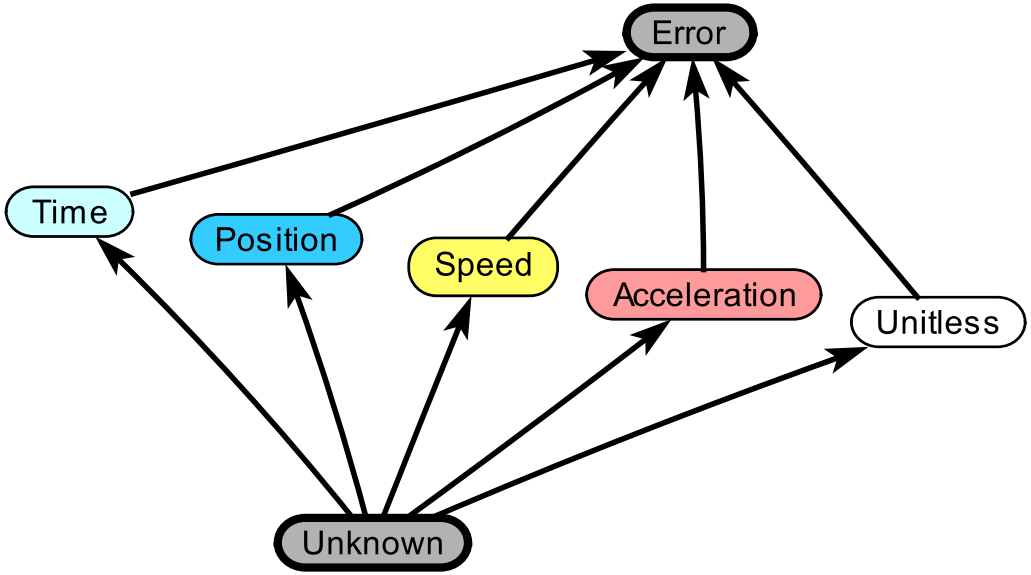


What Questions Do We Want to Ask?

- Types
- Units
- Dimensions
- Constant / Non-constant
- Product configuration
- Code ownership
- Code Maturity Level
- .....

What Questions Do We Want to Ask?

- Types
- Units
- Dimensions
- Constant / Non-constant
- Product configuration
- Code ownership
- Code Maturity Level
- .....



Approach

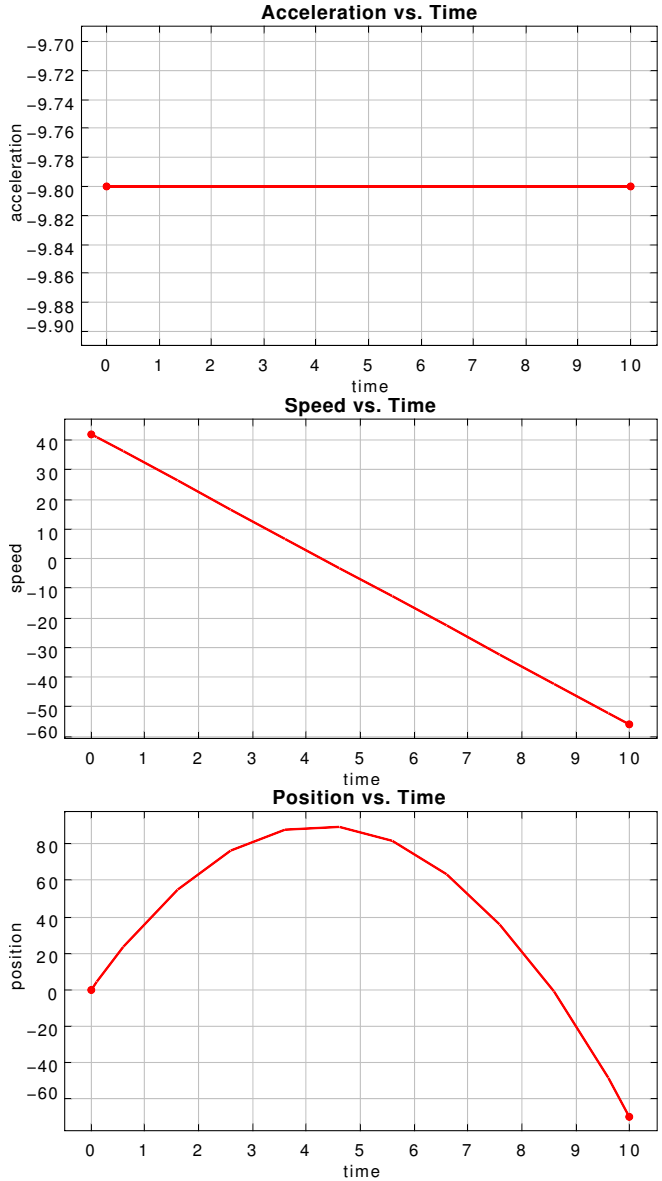
Ptolemy II: Our modeling and simulation environment

Pthomas: A framework on top of Ptolemy II, allowing analysis of these semantic properties

Types of analysis:

- Inference
- Validity Checking

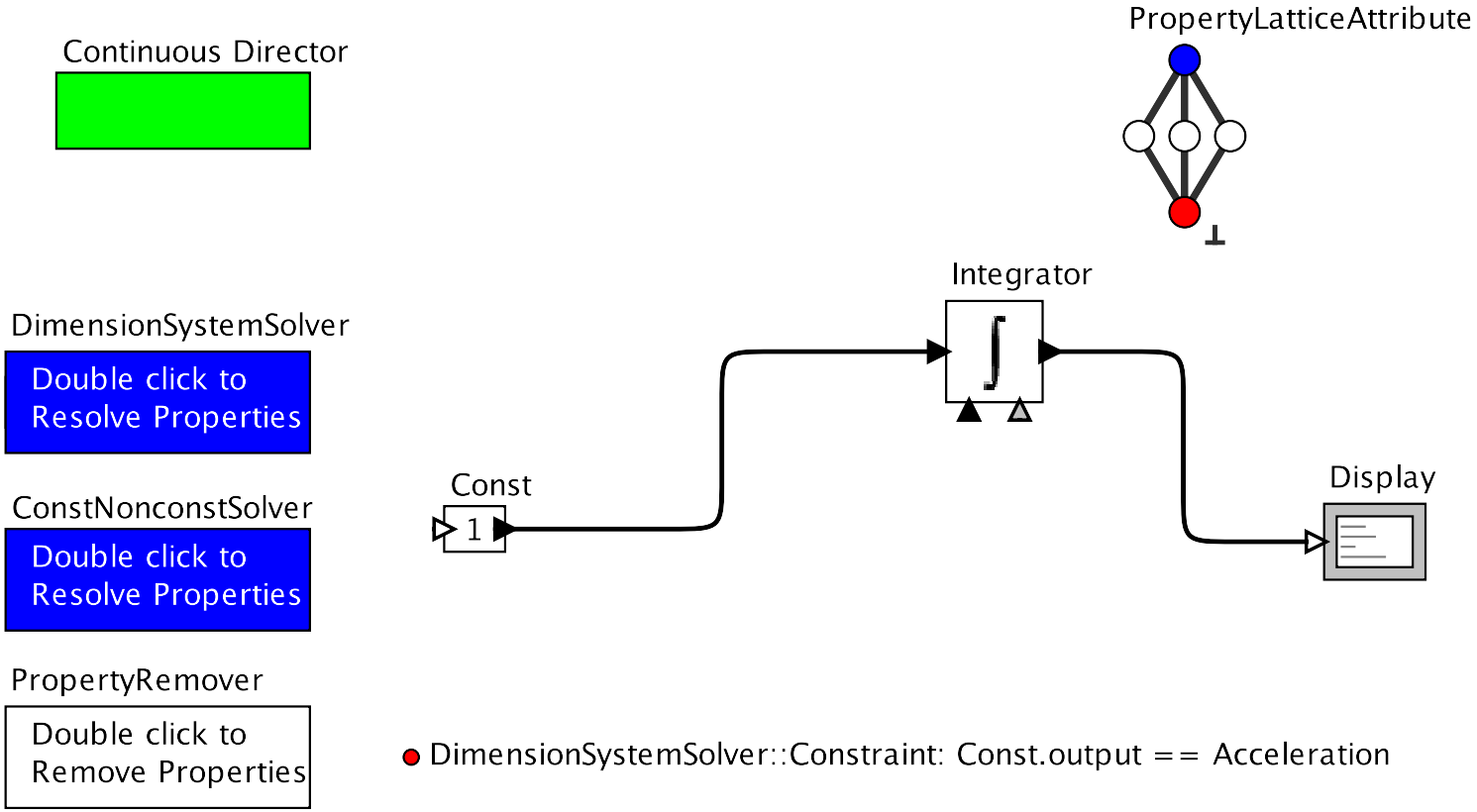
Calculus Review



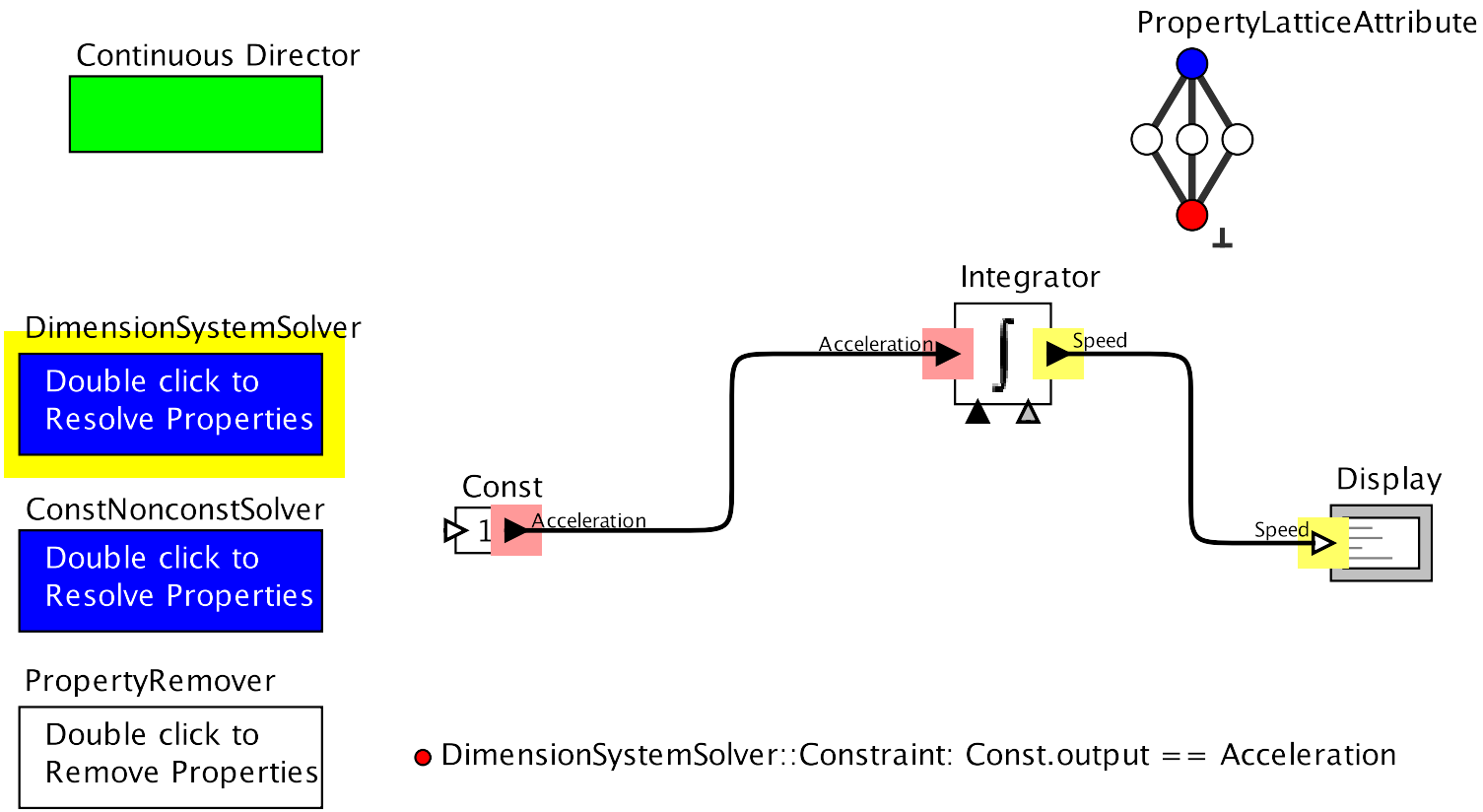
$$\int acceleration(t)\,dt = speed(t)$$

$$\int speed(t)\,dt = position(t)$$

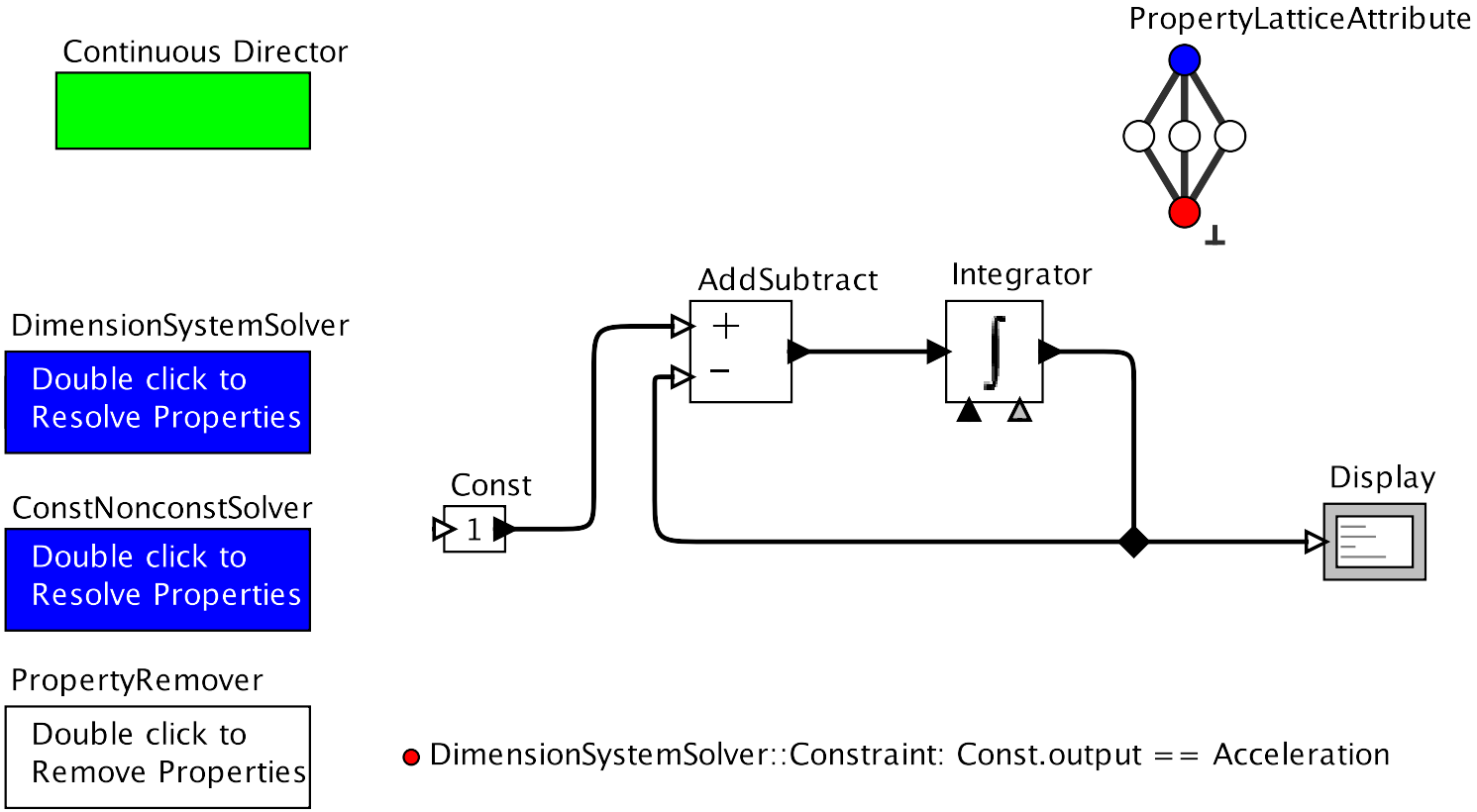
Demo


Continuous Director
PropertyLatticeAttribute
⊥
Integrator
DimensionSystemSolver
Double click to Resolve Properties
Const
1
Display
ConstNonconstSolver
Double click to Resolve Properties
PropertyRemover
Double click to Remove Properties
DimensionSystemSolver::Constraint: Const.output == Acceleration

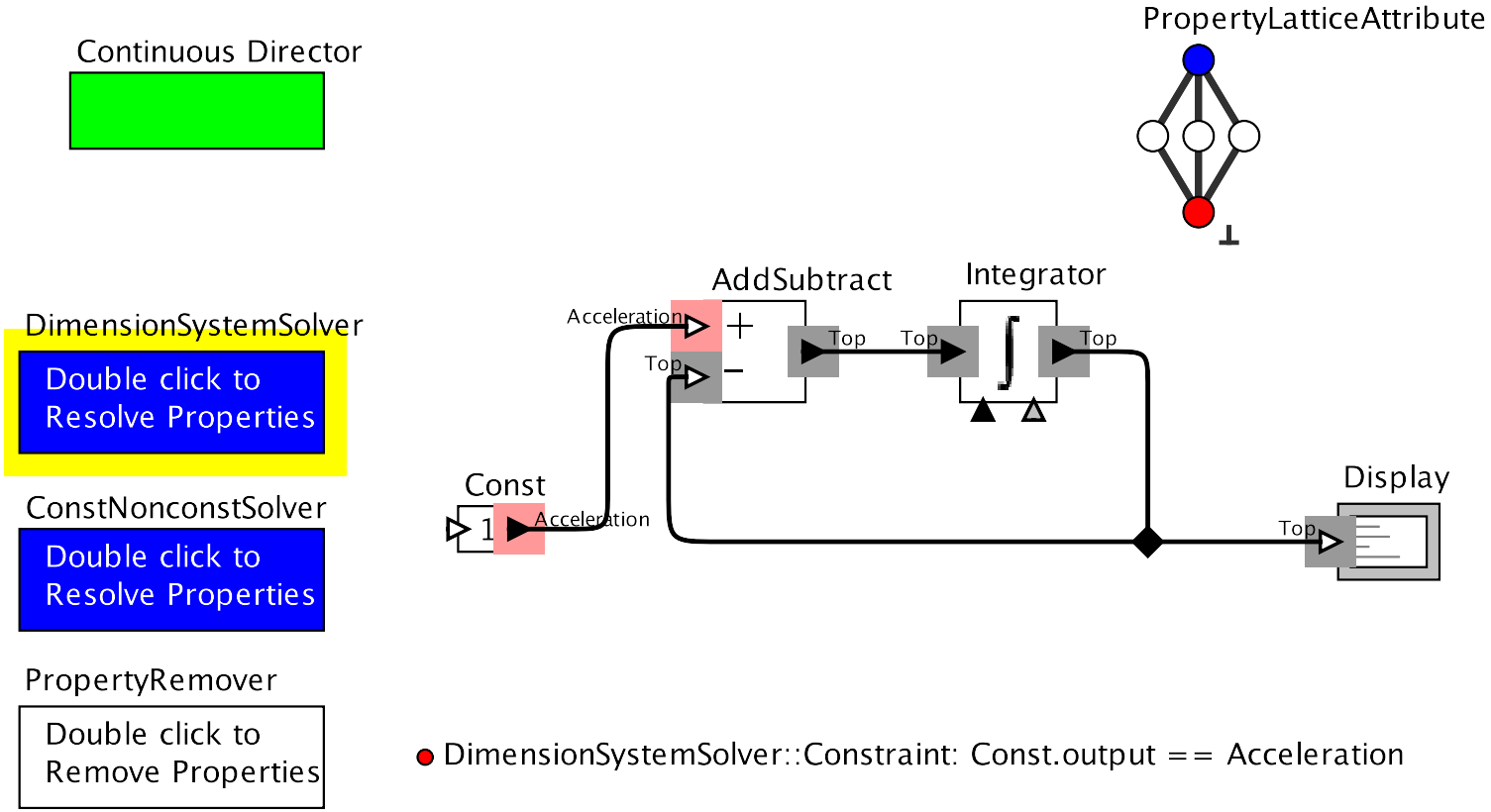
Demo


Continuous Director
PropertyLatticeAttribute
⊥
Integrator
DimensionSystemSolver
Double click to Resolve Properties
Acceleration
Speed
Const
1
Acceleration
Display
Speed
ConstNonconstSolver
Double click to Resolve Properties
PropertyRemover
Double click to Remove Properties
DimensionSystemSolver::Constraint: Const.output == Acceleration

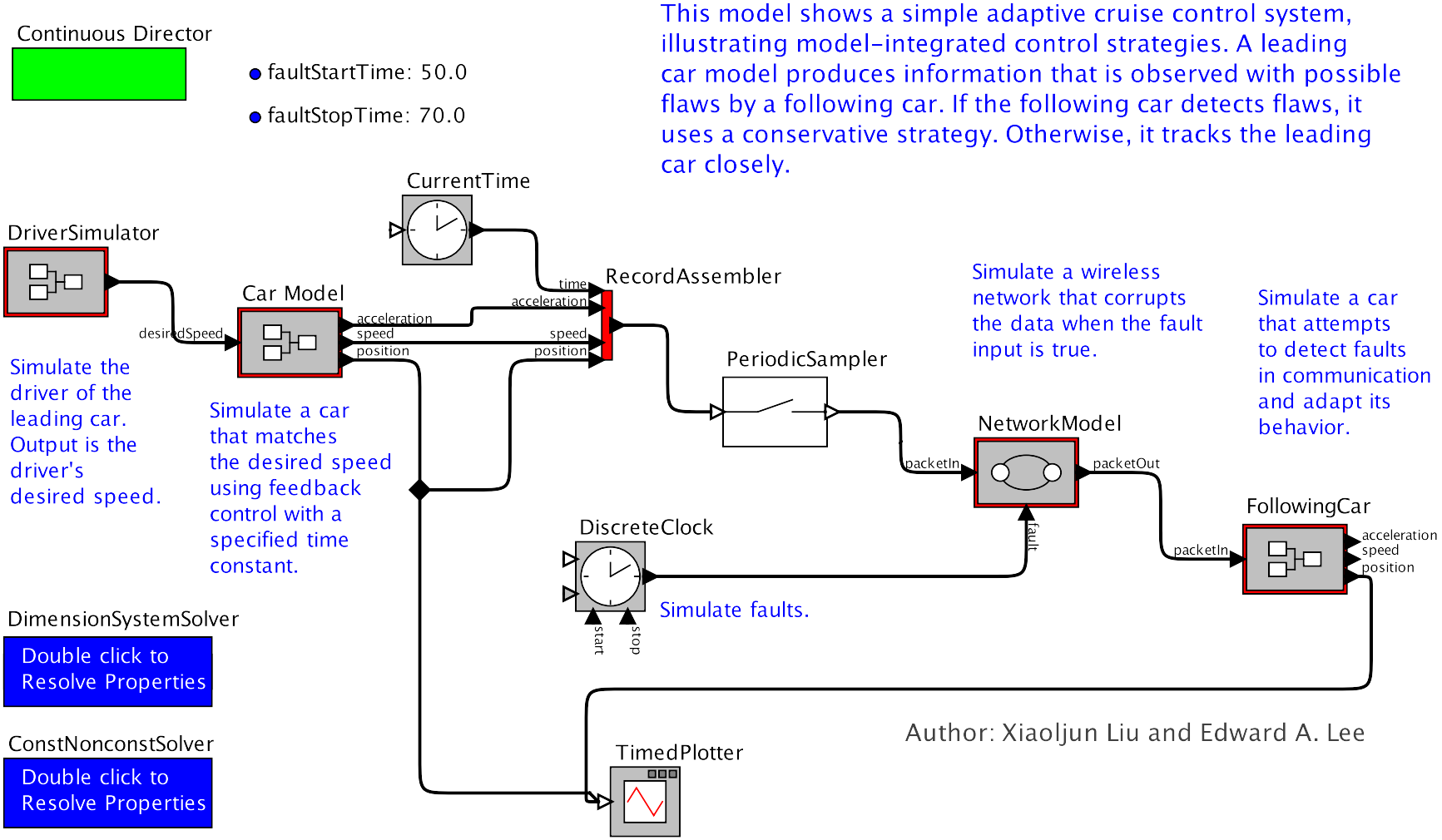
Demo


Continuous Director
PropertyLatticeAttribute
⊥
AddSubtract
+
−
Integrator
DimensionSystemSolver
Double click to Resolve Properties
Const
1
Display
ConstNonconstSolver
Double click to Resolve Properties
PropertyRemover
Double click to Remove Properties
DimensionSystemSolver::Constraint: Const.output == Acceleration

Demo

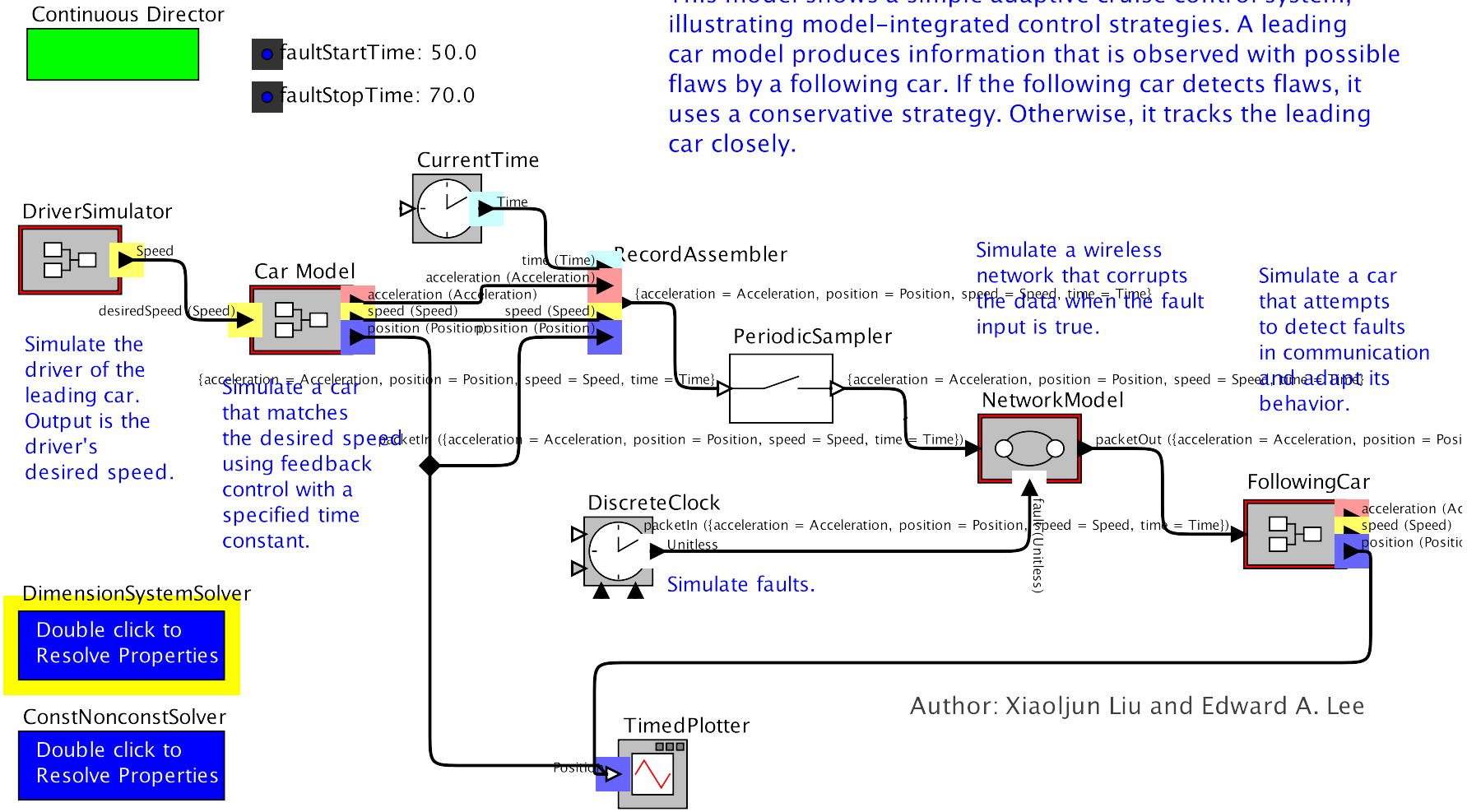

Continuous Director
PropertyLatticeAttribute
⊥
DimensionSystemSolver
Double click to Resolve Properties
ConstNonconstSolver
Double click to Resolve Properties
PropertyRemover
Double click to Remove Properties
AddSubtract
Integrator
Acceleration
Top
+
−
Top
Top
Const
1
Acceleration
Display
Top
DimensionSystemSolver::Constraint: Const.output == Acceleration

Demo

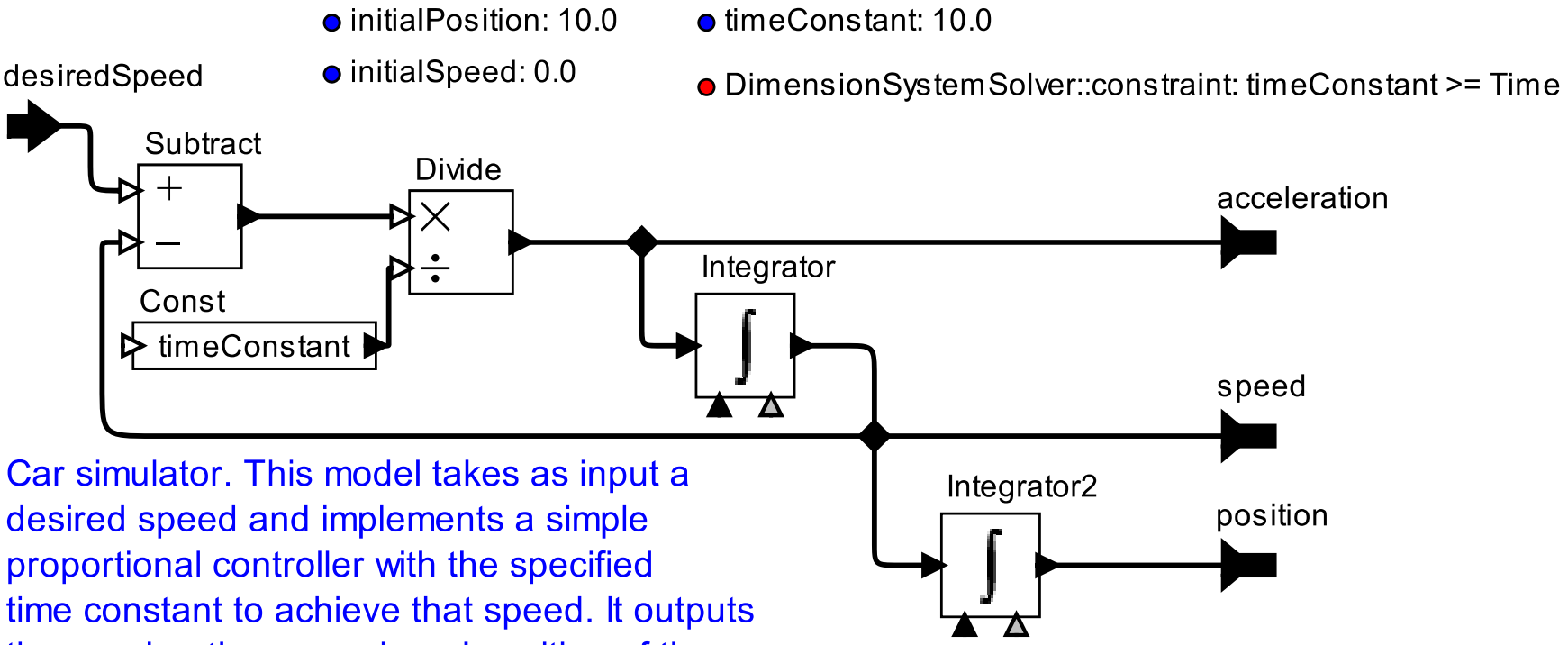

Continuous Director
faultStartTime: 50.0
faultStopTime: 70.0
This model shows a simple adaptive cruise control system, illustrating model-integrated control strategies. A leading car model produces information that is observed with possible flaws by a following car. If the following car detects flaws, it uses a conservative strategy. Otherwise, it tracks the leading car closely.
CurrentTime
DriverSimulator
Car Model
desiredSpeed
acceleration
speed
position
time
acceleration
speed
position
RecordAssembler
PeriodicSampler
Simulate the driver of the leading car. Output is the driver's desired speed.
Simulate a car that matches the desired speed using feedback control with a specified time constant.
Simulate a wireless network that corrupts the data when the fault input is true.
Simulate a car that attempts to detect faults in communication and adapt its behavior.
NetworkModel
packetIn
packetOut
fault
FollowingCar
packetIn
acceleration
speed
position
DiscreteClock
start
stop
Simulate faults.
DimensionSystemSolver
Double click to Resolve Properties
ConstNonconstSolver
Double click to Resolve Properties
TimedPlotter
Author: Xiaoljun Liu and Edward A. Lee

Demo

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.

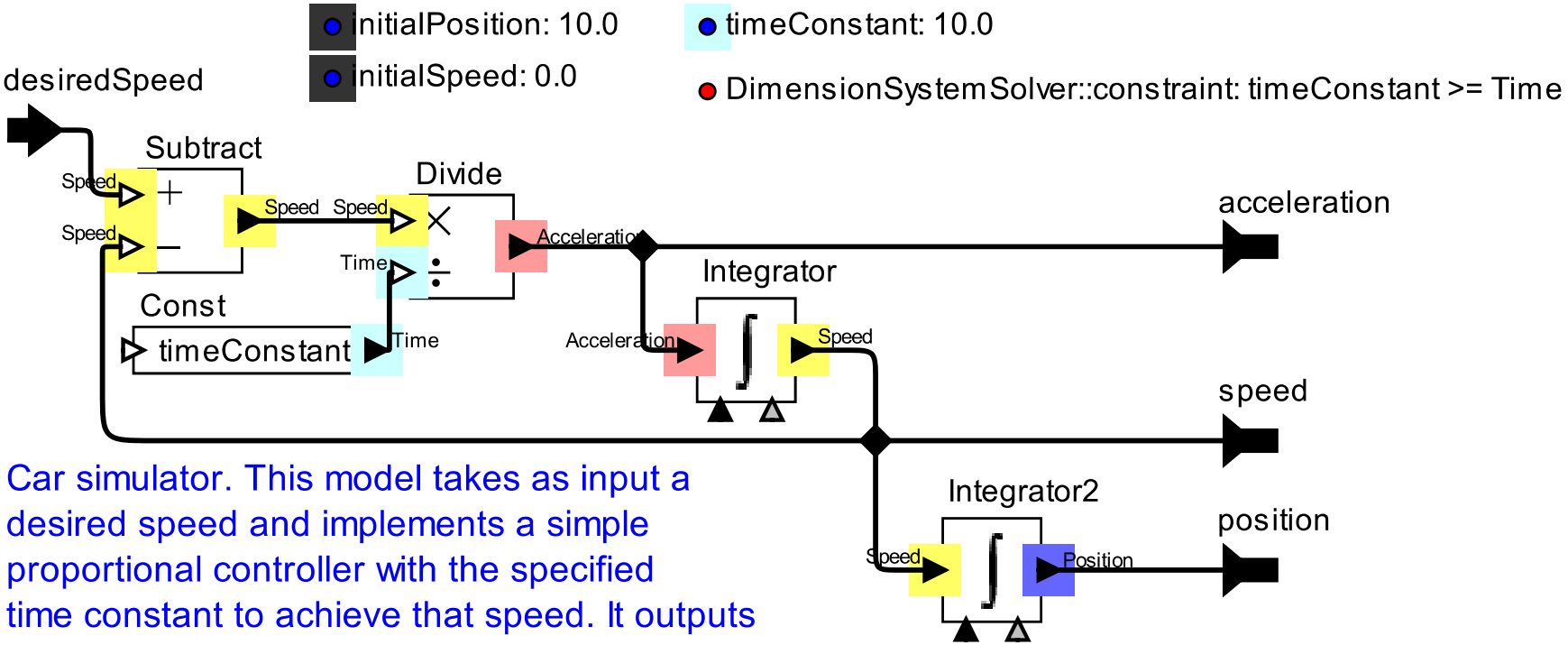


Demo



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.

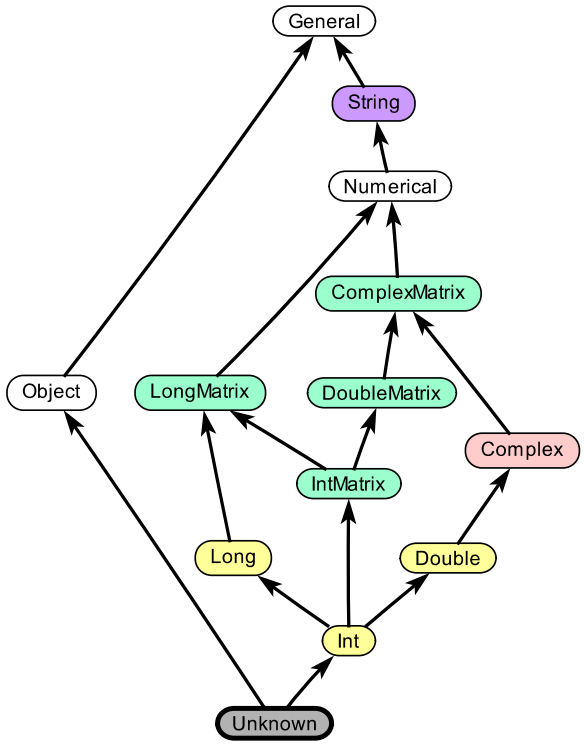
Demo



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.

Background

- Lattice
  A partially ordered set in which every pair of elements has both a least upper bound and a greatest lower bound.
- Monotonic Function
  A function f for which
  $\vec{x_1} \leq \vec{x_2} \implies f(\vec{x_1}) \leq f(\vec{x_2})$



Problem Statement

Given:

$$\text{Lattice:} \quad P \tag{1}$$

$$\text{Constants \& Variables:} \quad p_1, p_2, \cdots \in P \tag{2}$$

$$\text{Constraints of the form:} \quad f(p_1, p_2, \cdots) \leq p_n \qquad (f \text{ monotonic}) \tag{3}$$

is there a satisfying assignment to variables?

Problem Statement

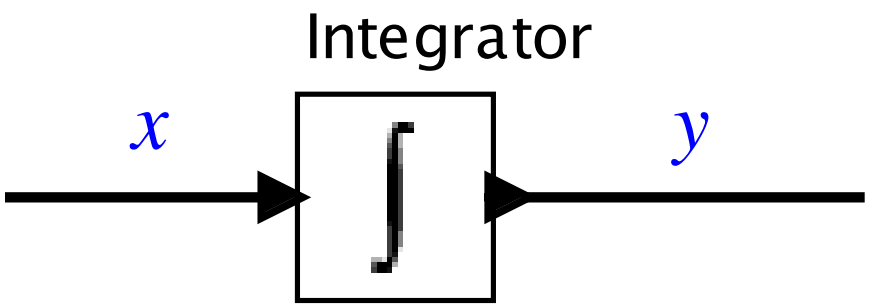
Given:

$$\begin{aligned} \text{Lattice:} \quad & P && (1) \\ \text{Constants \& Variables:} \quad & p_1, p_2, \cdots \in P && (2) \\ \text{Constraints of the form:} \quad & f(p_1, p_2, \cdots) \leq p_n \qquad (f \text{ monotonic}) && (3) \end{aligned}$$

is there a satisfying assignment to variables?

This problem has a linear time algorithm!
(Rehof and Mogensen, 1996)

Writing Actor Constraints

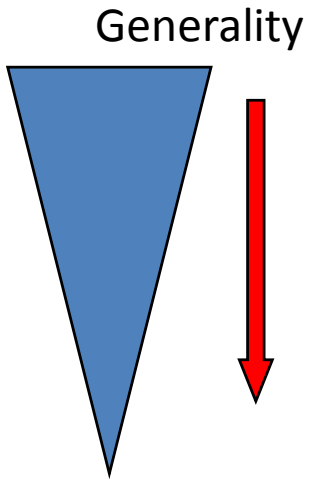


Component	Elements	Constraints
Integrator	input x, output y	$f_I(p_y) \leq p_x$ $f_O(p_x) \leq p_y$

$$f_I(p_y) = \begin{cases} \text{Undef.} & \text{if } p_y = \text{Undef.} \\ \text{Speed} & \text{if } p_y = \text{Pos.} \\ \text{Accel.} & \text{if } p_y = \text{Speed} \\ \text{Unitless} & \text{if } p_y = \text{Time} \\ \text{Error} & \text{otherwise} \end{cases} \qquad f_O(p_x) = \begin{cases} \text{Undef.} & \text{if } p_x = \text{Undef.} \\ \text{Pos.} & \text{if } p_x = \text{Speed} \\ \text{Speed} & \text{if } p_x = \text{Accel.} \\ \text{Time} & \text{if } p_x = \text{Unitless} \\ \text{Error} & \text{otherwise} \end{cases}$$

User interface

Problem: $|C| \propto |M|$



1. **Default Constraints**
   - Set globally by the property solver (actors, connections, etc.)
2. **Actor-specific Constraints**
   - Uses an *adapter pattern* for actors
3. **Instance-specific Constraints**
   - Specified through model annotations

Future work

- Handling of conflicts
  - How can we present useful information in case of a type error? ✓
- Unit system
  - Unit conversion, Default units, Aliasing
- Extension to infinite lattices

Conclusion

Our framework (Pthomas) for analyzing model properties:

1. Requires minimal user specification (lattice and constraints)
2. Infers unspecified properties
3. Catches and reports design errors
4. Scales up efficiently to large models

Relational Constraint Problem (RCP)

$$RCP : (P, C)$$

P is a partially ordered set, C is a set constraints of the form:

$$r(P_x, P_y, \ldots)$$

where r is a relation (e.g. $=, +, \times, \leq$).

A *solution* is a satisfying assignment to property variables.

Definite Monotone Function Problem (DMFP)

Special case of RCP

$$DMFP : (P, C_F)$$

P is a lattice, C_F is a set of *definite* inequalities:

$$F(P_y, P_z, \ldots) \leq P_x$$

where F is a monotonic function.

Here, there is a unique *least fixed point* (LFP) solution.

Related work

- Constraint Satisfiability (Rehof and Mogensen)
  - Linear time algorithm for monotone function problem
- Hindley-Milner Type Theory
  - Sound, incomplete static check of programs before run time.
- Web Ontology Language (OWL), Eclipse Modeling Framework (EMF), Object Constraint Language (OCL)
  - Similarities: Ontology frameworks (concepts and relationships)
  - Differences: Expressiveness vs. Efficiency

Algorithm D (Rehof and Mogensen, 1996)

Pseudocode

$C_{var} \leftarrow \{\tau \leq A \in C : A \text{ a variable}\}$;
$C_{const} \leftarrow \{\tau \leq A \in C : A \text{ a constant}\}$;
$p(\beta) = \text{Undef. for all variables } \beta$;
while *there are unsatisfied constraints in* C_{var} **do**
  Let $\tau \leq \beta$ be one such constraint ;
  $\beta \leftarrow \beta \vee \tau$;
end
if *there are unsatisfied constraints in* C_{const} **then**
  Fail: There is no solution ;
end

For a finite lattice, this algorithm takes $O(height(L) \times |C|)$.

Conflicts

- What is a conflict?
  - Unsatisfiable constraints
- How is it detected?

$C_{const} \leftarrow \{\tau \leq A \in C : A \text{ a constant}\}$;

...

if *there are unsatisfied constraints in* C_{const} **then**

  Fail: There is no solution ;

end