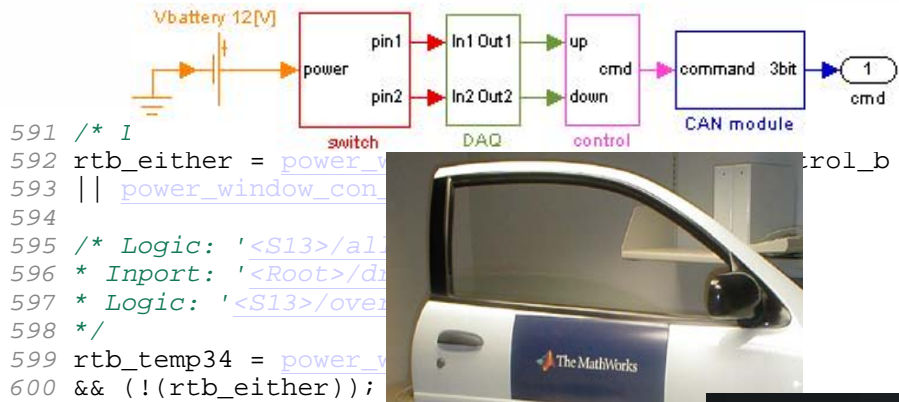
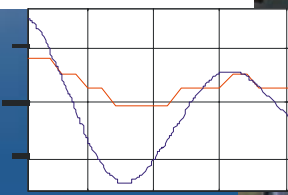


Model-Based Design – An overview and research



Pieter J. Mosterman
pieter.mosterman@mathworks.com

Senior Research Scientist
The MathWorks, Inc.



Introduction

- Model-Based Design
 - Exploit computational models
 - Increasingly adopted in industry
- Modeling at an enterprise level
 - Many different modeling formalisms
 - Relate and combine models
 - Different parts of a system
 - Different design stages of a system
- Challenges
 - Efficiently manage models, formalisms and levels of abstraction
 - Efficiently evaluate dynamics of different computational semantics

Agenda

- **Model-Based Design**
- Computer Automated Multiparadigm Modeling
 - Application
- Hybrid Dynamic Systems
- Summary

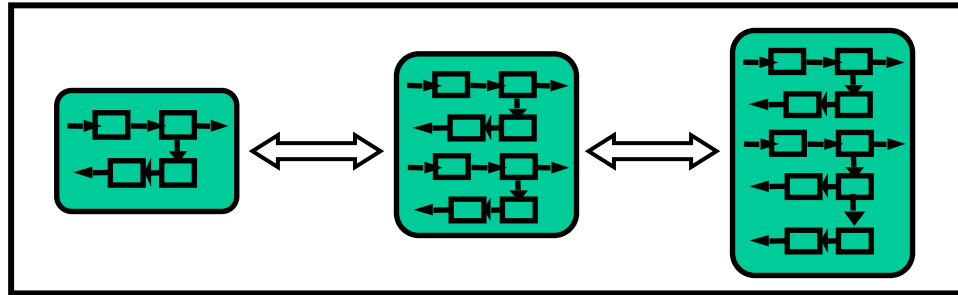
Model-Based Design

Requirements and Specifications



Excel
Word
Scenario diagrams

↕
Functionality



Block diagrams
Statecharts
Hybrid Automata
Bond graphs

↕
Implementation



C/C++
Java
HDL
ASM

↕
Test and Verification

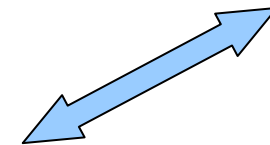
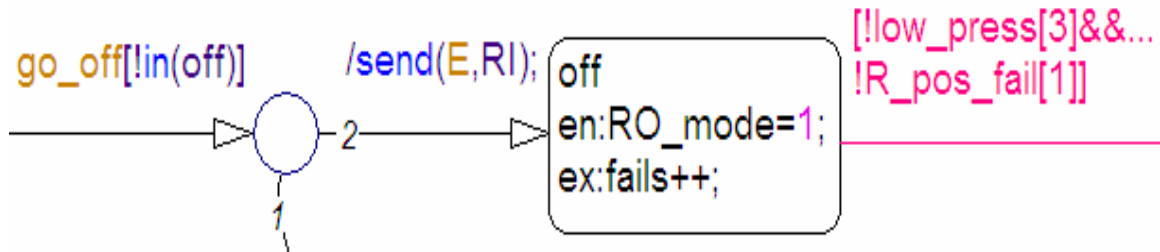
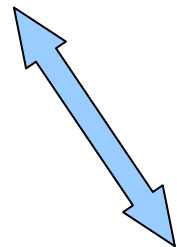
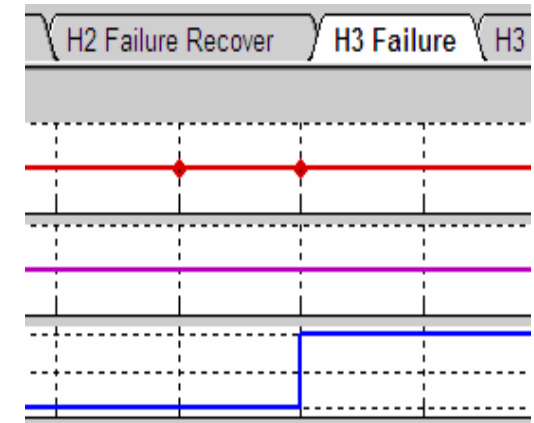
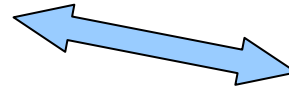


DEVS
Automata
Numerical Values
Polynomials

Model analysis

2.1.5 Hydraulic pressure 3 failure

If a failure is detected in the hydraulic pressure 3 system, while there are no other failures, isolate the fault by switching the Right Outer actuator to the off mode.

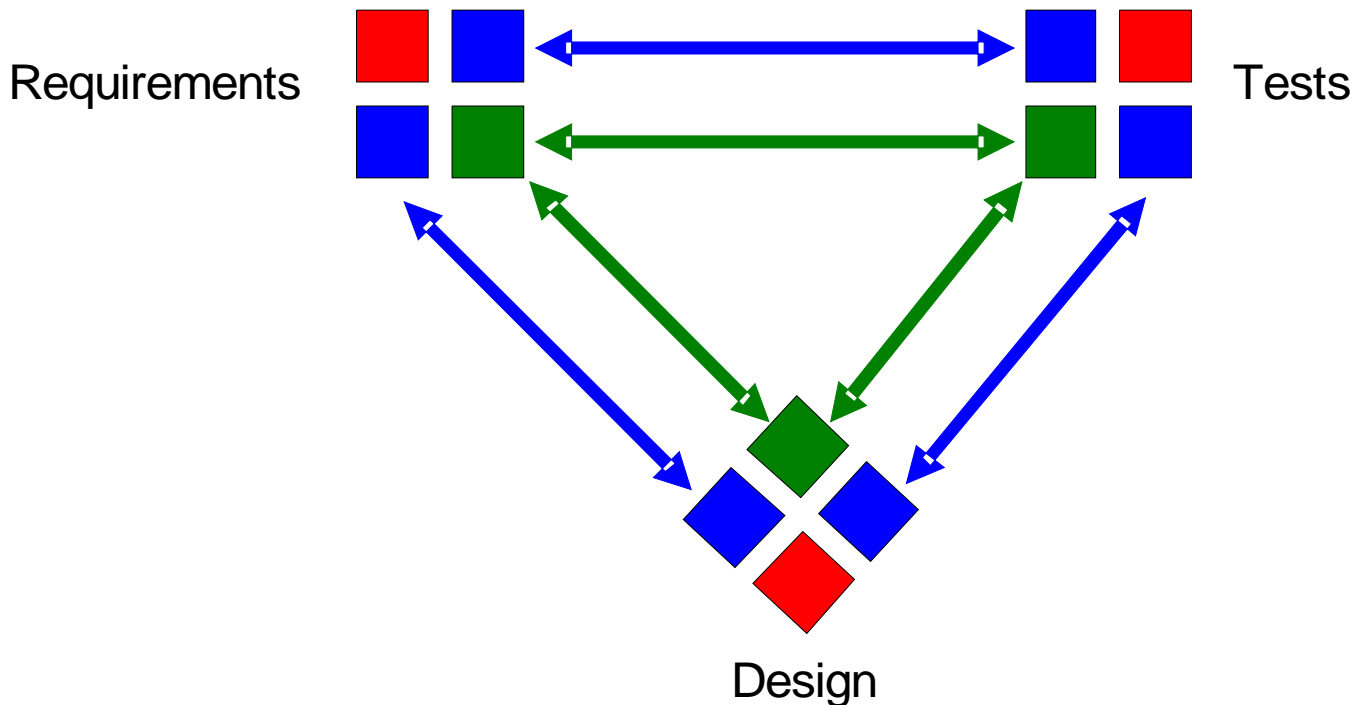


Control flow analysis

- Condition coverage
- Decision coverage
- Range (min/max) coverage
- Modified condition/decision coverage (MC/DC)
 - DO-178B
- Ensure there is a:
 - Requirement for each model element
 - Test for each model element
 - Test for each requirement

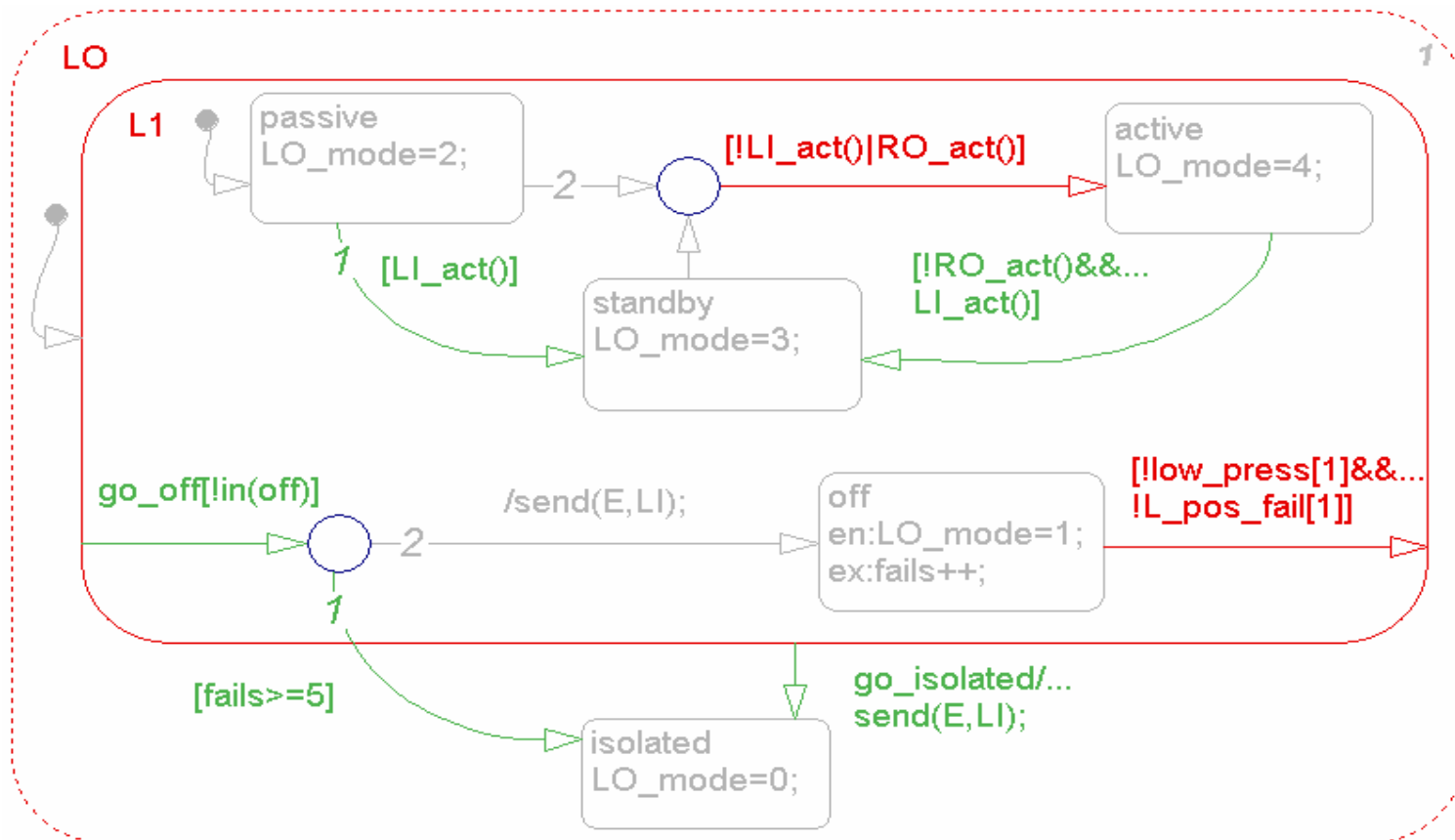
Yet another design triangle

- Complete ■
- Lacking ■
- Superfluous ■



Lacking test

- Never fully tested going to active

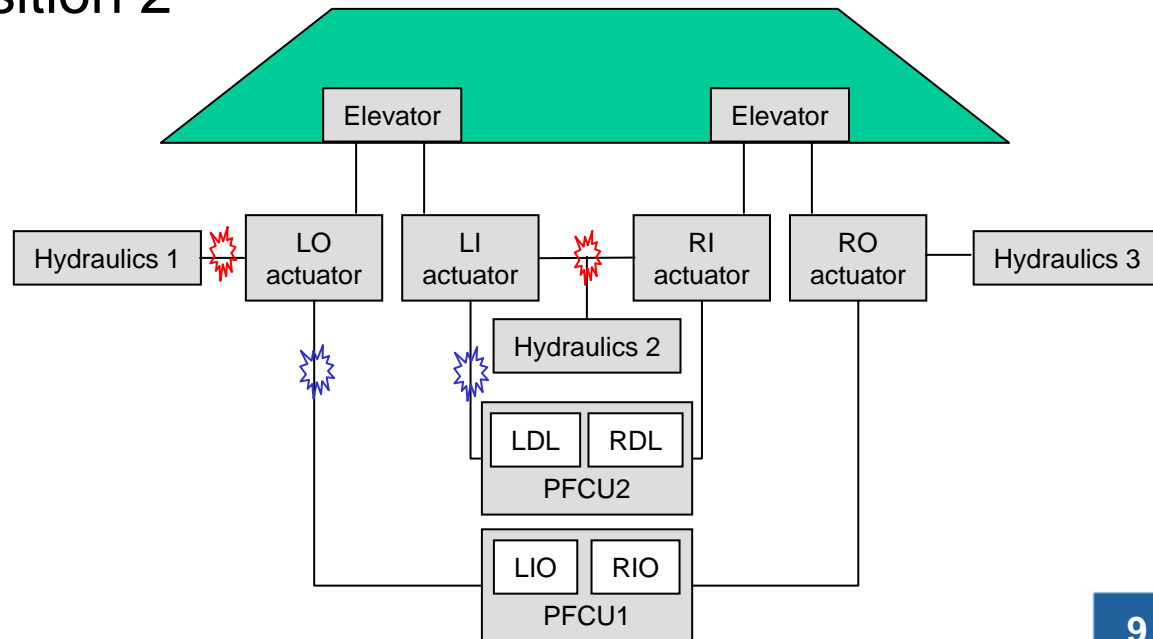


Misinterpreted requirement

- Four possible left multiple fault permutations
 - Hydraulic 1 && hydraulic 2
 - Hydraulic 1 && position 2
 - Position 1 && hydraulic 2
 - Position 1 && position 2

2.1.15 Multiple failures on Left hydraulics and actuators

If multiple failures are detected in hydraulic pressure 1 **or** 2 systems **and** either the Left Outer actuator position sensor **or** Left Inner actuator position sensor, isolate the fault by switching both the Left Outer actuator **and** Left Inner Actuator to the isolated mode.



Model transformations

- New synthesis (model elaboration) methods
 - Adapt synthesis implementation
- Domain specific modeling formalisms
 - Evolve constantly
 - Require modifications of transformations
 - Model transformations
 - Risk of making a change in software is proportional to the size of the software, not the change...

Agenda

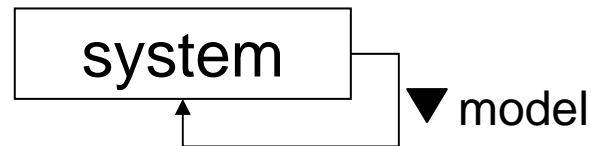
- Model-Based Design
- **Computer Automated Multiparadigm Modeling**
 - Application
- Hybrid Dynamic Systems
- Summary

Computer Automated Multiparadigm Modeling

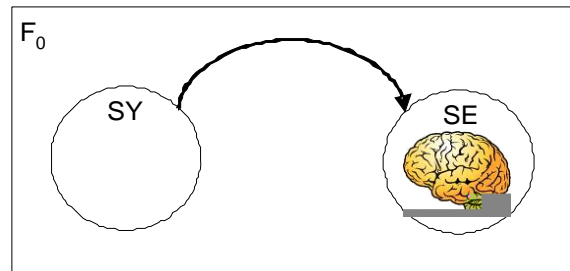
- Computer Automated Multiparadigm Modeling (CAMPaM)
 - Annual McGill Bellairs workshop
- Three elements of CAMPaM
 - Multi-formalism modeling
 - Meta-modeling
 - Multiple levels of abstraction
- Model Model transformation
 - Graph grammars
 - ...

What is a model anyway?

- A system that represents a system

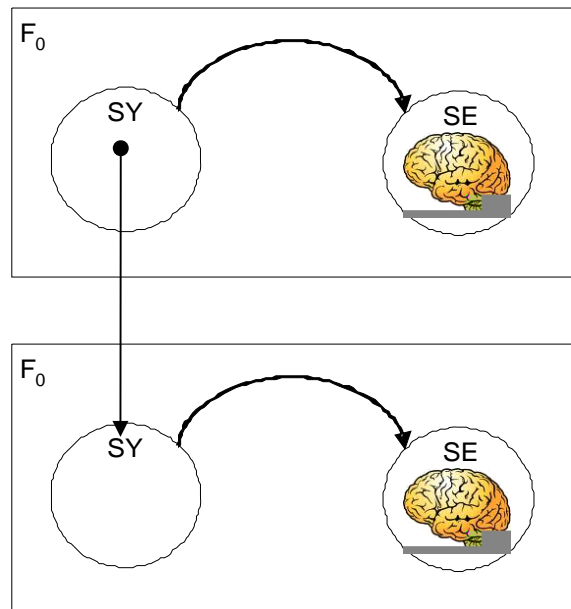


- Representation in a formalism
- Formally define a formalism (Ogden-Richards triangle of semiotics)
 - Syntactic domain
 - Semantic domain



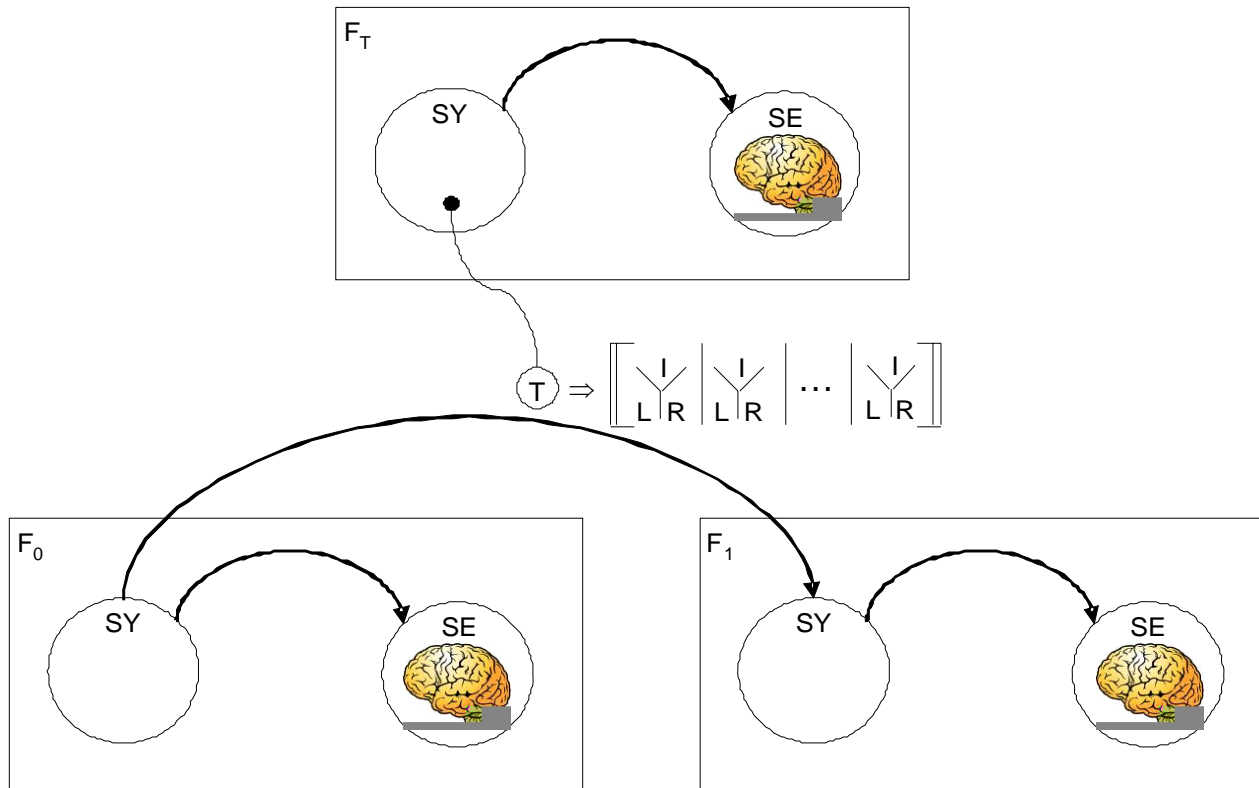
Defining a formalism

- The syntax can be defined by
 - Enumeration
 - A meta model
 - For example, a grammar



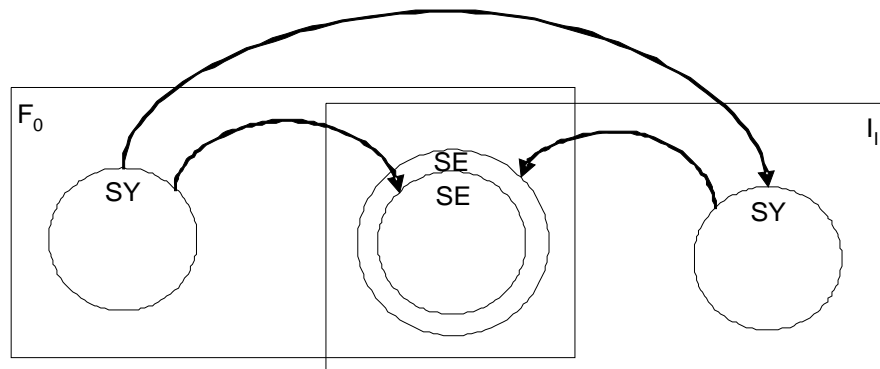
How do you define the semantics?

- Model transformation



Mapping the semantic domains

- The semantic domain to be defined needs to be subsumed by the semantic domain of the defining formalism



What is a 'formal method'?

- IEC 61508-7

B.2.2 Formal methods

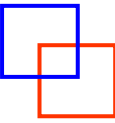

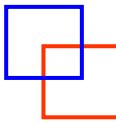


Aim: To express a specification unambiguously and consistently, so that mistakes and omissions can be detected.

Description: Formal methods provide a means of developing a description of a system at some stage in its specification or design. The resulting description takes a **mathematical form and can be subjected to mathematical analysis** to detect various classes of inconsistency or incorrectness. Moreover, the description can **in some cases** be analysed by a machine with a rigour similar to the syntax checking of a source program by a compiler, or animated to display various aspects of the behaviour of the system described. **Animation can give extra confidence** that the system meets the real requirement as well as the formally specified requirement, because it improves human recognition of the specified behaviour.

A formal method will **generally offer** a notation (generally some form of discrete mathematics being used), a technique for deriving a description in that notation, and various forms of analysis for checking a description for different correctness properties. Starting from a mathematically formal specification, the design can be transformed by a series of step-wise refinements to a logic circuit design.

In terms of formalism elements

- Informal: One can create a sentence that is not provable part of the language
- Semi-formal: One can create a sentence with an unknown logical value
- Formal: All sentences have known logical values

	syntax	semantics
informal		
semi-formal		
formal		

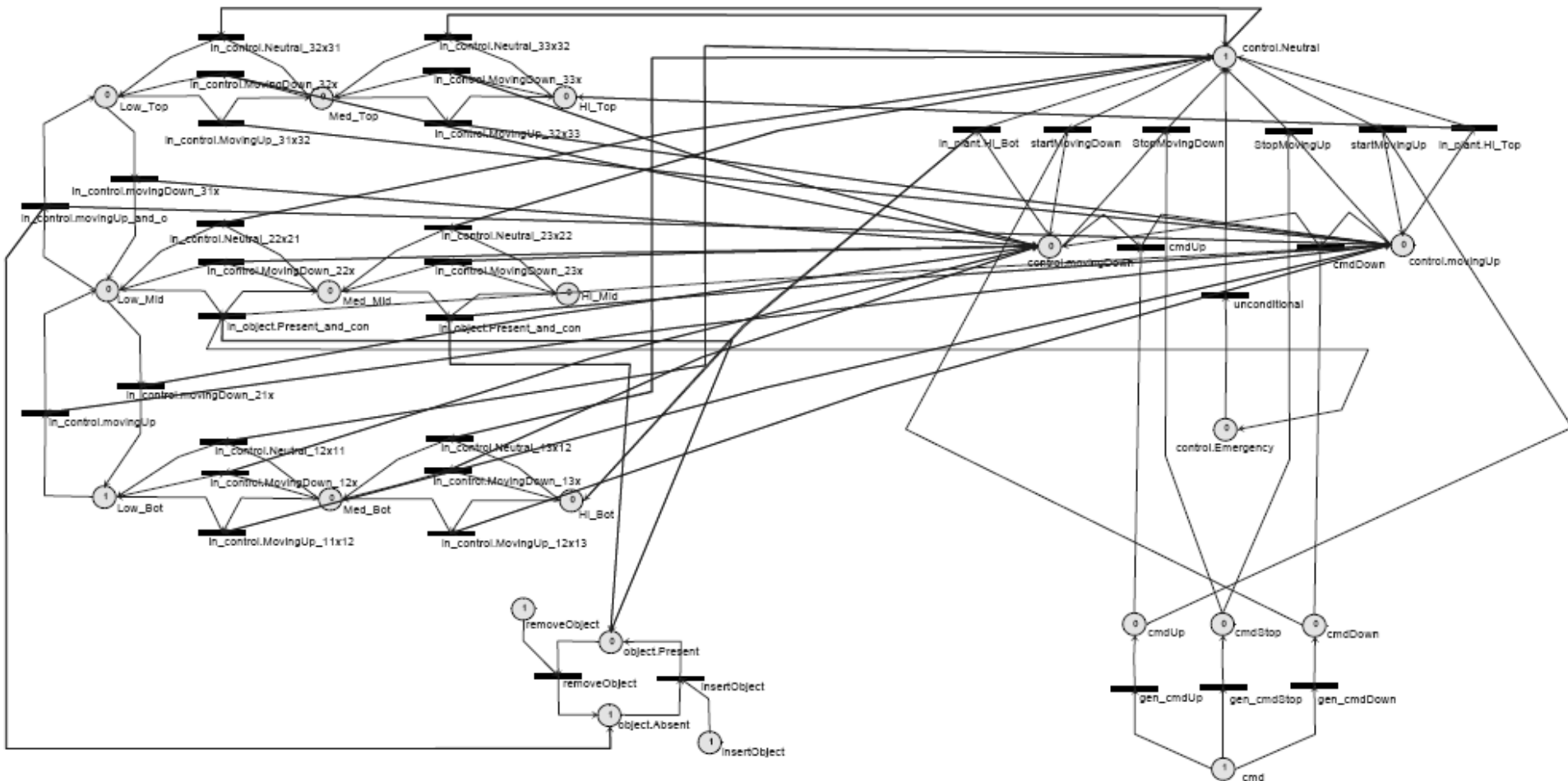
Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
 - [Application](#)
- Hybrid Dynamic Systems
- Summary

Model-Based Design application

- Analysis of power window system
- Model plant with a Petri net
- Model controller with a statechart
 - Automatic transformation into a Petri net
- Compose plant and controller Petri net
- Generate reachability graph
- Perform model checking
 - The state with high current while in between top and bottom of the frame cannot be reached?

Composed controller/plant model



Model checking results

```

MODE LABELS:

0['Low_Bot', 'cmd', 'insertObject', 'object.Absent', 'removeObject', 'control.Neutral', 'True']
1['Low_Bot', 'control.Neutral', 'removeObject', 'object.Present', 'cmd', 'True']
2['Low_Bot', 'control.Neutral', 'cmd', 'object.Absent', 'True']
3['Low_Bot', 'control.Neutral', 'object.Absent', 'cmdUp', 'True']
4['Low_Bot', 'control.movingUp', 'object.Absent', 'True']
5['Low_Mid', 'object.Absent', 'control.movingUp', 'True']
6['Low_Top', 'control.movingUp', 'object.Absent', 'True']
7['Med_Top', 'object.Absent', 'control.movingUp', 'True']
8['Hi_Top', 'control.movingUp', 'object.Absent', 'True']
9['Hi_Top', 'control.Neutral', 'object.Absent', 'True']
10['Med_Top', 'control.Neutral', 'object.Absent', 'True']
11['Low_Top', 'control.Neutral', 'object.Absent', 'True', 'deadlock']
12['Low_Bot', 'control.Neutral', 'object.Absent', 'cmdStop', 'True', 'deadlock']
13['Low_Bot', 'control.Neutral', 'cmdDown', 'object.Absent', 'True']
14['Low_Bot', 'object.Absent', 'control.movingDown', 'True']
15['Med_Bot', 'object.Absent', 'control.movingDown', 'True']
16['Hi_Bot', 'object.Absent', 'control.movingDown', 'True']
17['control.Neutral', 'object.Absent', 'Hi_Bot', 'True']
18['Med_Bot', 'control.Neutral', 'object.Absent', 'True']
19['Low_Bot', 'control.Neutral', 'object.Absent', 'True', 'deadlock']
20['Low_Bot', 'control.Neutral', 'object.Present', 'cmdUp', 'removeObject', 'True']
21['Low_Bot', 'object.Present', 'control.movingUp', 'removeObject', 'True']
22['Low_Mid', 'object.Present', 'control.movingUp', 'removeObject', 'True']
23['object.Present', 'Med_Mid', 'control.Emergency', 'removeObject', 'True']
24['control.Emergency', 'Med_Mid', 'object.Absent', 'True']
25['control.Neutral', 'object.Absent', 'Med_Mid', 'True']
26['control.Neutral', 'Low_Mid', 'object.Absent', 'True', 'deadlock']
27['control.Neutral', 'object.Present', 'Med_Mid', 'removeObject', 'True']
28['control.Neutral', 'Low_Mid', 'object.Present', 'removeObject', 'True']
29['Low_Bot', 'control.Neutral', 'object.Present', 'cmdStop', 'removeObject', 'True']
30['Low_Bot', 'control.Neutral', 'cmdDown', 'object.Present', 'removeObject', 'True']
31['Low_Bot', 'object.Present', 'removeObject', 'control.movingDown', 'True']
32['Med_Bot', 'object.Present', 'removeObject', 'control.movingDown', 'True']
33['object.Present', 'Hi_Bot', 'removeObject', 'control.movingDown', 'True']
34['control.Neutral', 'object.Present', 'Hi_Bot', 'removeObject', 'True']
35['Med_Bot', 'control.Neutral', 'object.Present', 'removeObject', 'True']
36['Low_Bot', 'control.Neutral', 'object.Present', 'removeObject', 'True']
37['Low_Bot', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'cmdUp', 'True']
38['Low_Bot', 'insertObject', 'control.movingUp', 'object.Absent', 'removeObject', 'True']
39['insertObject', 'Low_Mid', 'object.Absent', 'control.movingUp', 'removeObject', 'True']
40['Low_Top', 'insertObject', 'control.movingUp', 'object.Absent', 'removeObject', 'True']
41['Med_Top', 'insertObject', 'object.Absent', 'control.movingUp', 'removeObject', 'True']
42['Hi_Top', 'insertObject', 'control.movingUp', 'object.Absent', 'removeObject', 'True']
43['Hi_Top', 'object.Present', 'control.movingUp', 'removeObject', 'True']
44['Hi_Top', 'control.Neutral', 'object.Present', 'removeObject', 'True']
45['Med_Top', 'control.Neutral', 'object.Present', 'removeObject', 'True']
46['Low_Top', 'control.Neutral', 'object.Present', 'removeObject', 'True']
47['Hi_Top', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'True']
48['Med_Top', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'True']
49['Low_Top', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'True']
50['Med_Top', 'object.Present', 'control.movingUp', 'removeObject', 'True']
51['Low_Top', 'object.Present', 'control.movingUp', 'removeObject', 'True']
52['Low_Bot', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'cmdStop', 'True']
53['Low_Bot', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'cmdDown', 'True']
54['Low_Bot', 'insertObject', 'object.Absent', 'removeObject', 'control.movingDown', 'True']
55['Med_Bot', 'insertObject', 'object.Absent', 'removeObject', 'control.movingDown', 'True']
56['insertObject', 'Hi_Bot', 'object.Absent', 'removeObject', 'control.movingDown', 'True']
57['insertObject', 'control.Neutral', 'object.Absent', 'Hi_Bot', 'removeObject', 'True']
58['Med_Bot', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'True']
59['Low_Bot', 'insertObject', 'control.Neutral', 'object.Absent', 'removeObject', 'True']

--> FORMULA = Hi_Mid
--> LIST OF MODES WHICH SATISFY THE FORMULA = []

```

Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
 - Application
- Hybrid Dynamic Systems
- Summary

Formalisms with different computational semantics

- Two possible approaches
 - Transformation
 - Combination
- Often discrete event and continuous time behavior
 - Hybrid dynamic systems
- Hybrid dynamic systems
 - Discrete state perspective
 - For example, hybrid automata
 - Equation based perspective
 - For example, hybrid bond graphs

Generic execution structure

- DSblock
 - Modelica execution specification
- Simulink S-Function interface
 - Initialize
 - Update
 - Derivatives
 - Zero-crossing
 - Output
 - ...
- Discrete event, continuous time

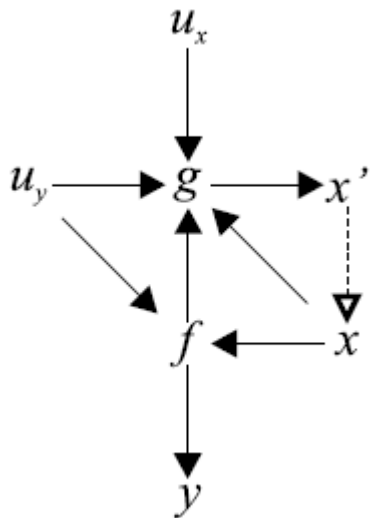
Generic execution structure

- DSblock
 - Modelica execution specification
- Simulink S-Function interface
 - Initialize
 - Update (g)
 - Derivatives
 - Zero-crossing
 - Output (f)
 - ...
- Discrete event, continuous time

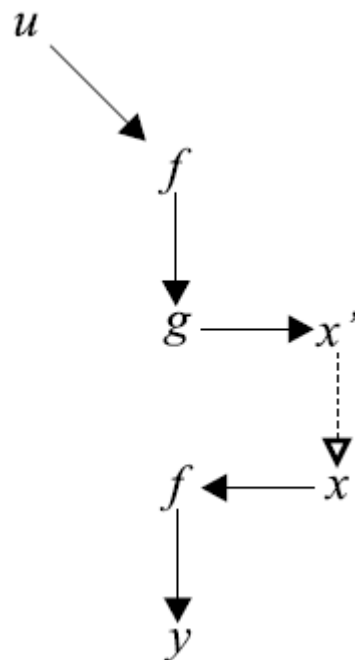
Block composition

- Compose functions across types

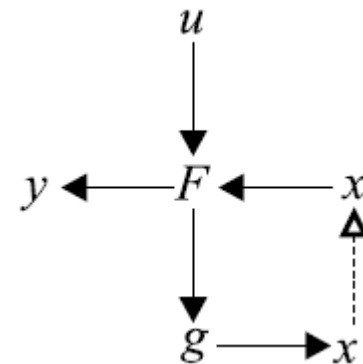
General block



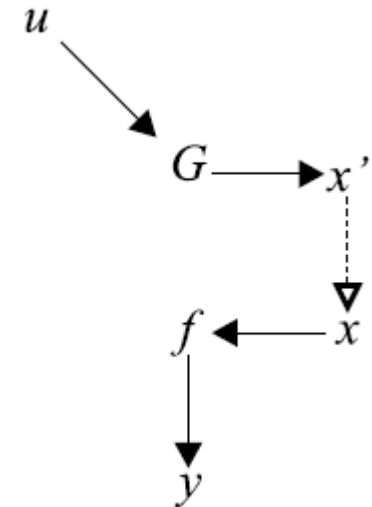
Gain/Delay combination



Output/Output aggregation

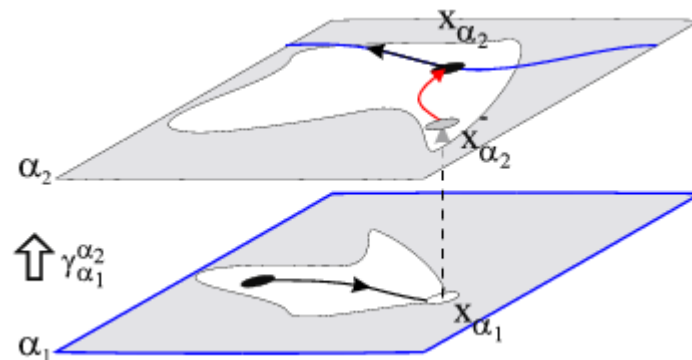
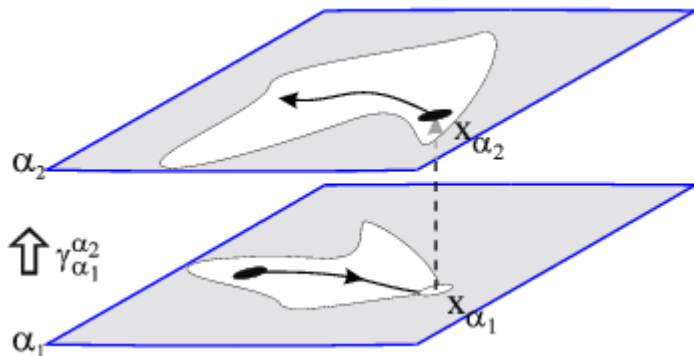


Output/Update aggregation



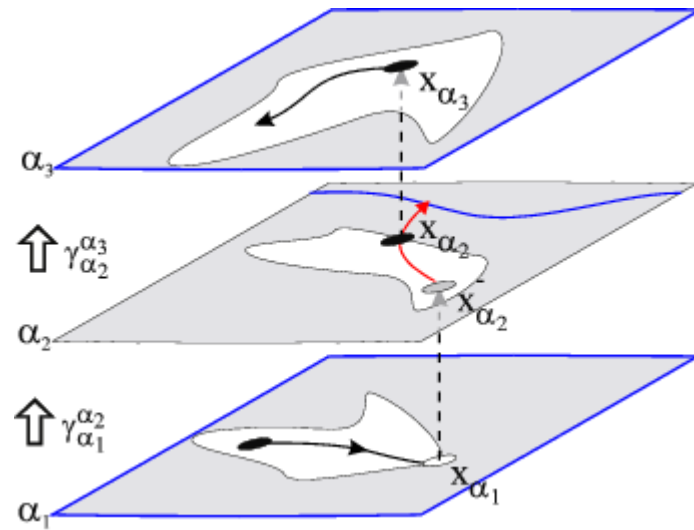
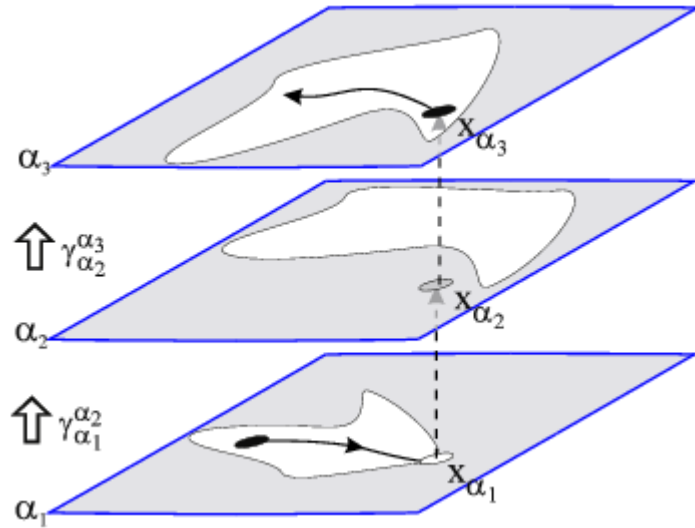
Hybrid dynamic system behavior

- Jumps **between** generalized state spaces
- State space constraints
 - Admissible state space
 - Jump **within** a state space



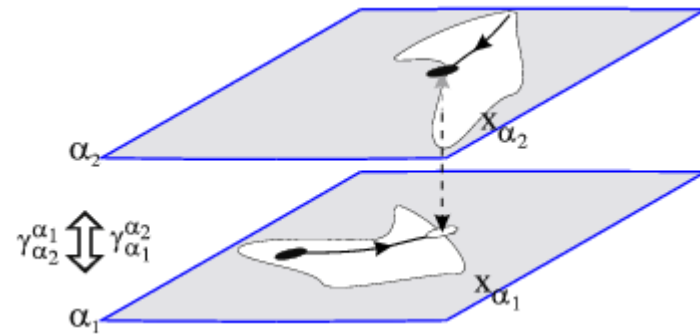
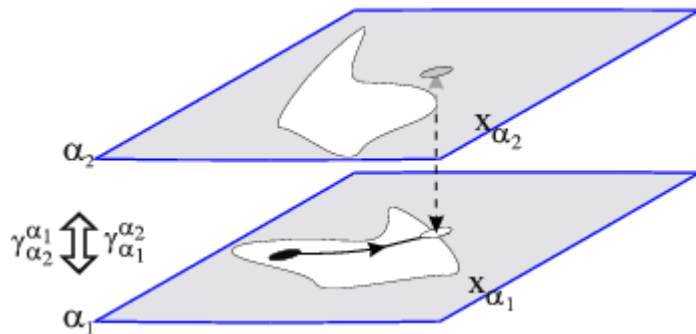
Sequences of mode changes

- Mythical mode
- Pinnacle



Pathological behaviors

- Divergence of time
- Chattering
- ... Zeno ...



Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
 - Application
- Hybrid Dynamic Systems
- **Summary**

Conclusions

- Model-Based Design
 - Enterprise-wide usage
- Model analysis ties together
 - Requirements
 - Design
 - Functional
 - Implementation
 - Tests
- Important research
 - Computer Automated Multiparadigm Modeling
 - Hybrid Dynamic Systems
 - ...

Acknowledgments

- Jason Ghidella
- Mirko Conrad
- Hans Vangheluwe
- Ben Denckla
- All participants of the annual Computer Automated Multiparadigm Modeling (CAMPaM) workshop at the McGill University Bellairs Campus, Barbados
- ...