Software and Model Synthesis in Complex CPS Design Flow

Janos Sztipanovits

Institute for Software Integrated Systems
Vanderbilt University
Email: janos.sztipanovits@vanderbilt.edu

WSS Workshop/EMSOFT
4 October 2013
Acknowledgement

- Ted Bapty, PI
- Ethan Jackson*
- Larry Howard, PI
- Xenofon Koutsoukos
- Zsolt Lattmann
- Tihamer Levendovszky
- Adam Nagel
- Sandeep Neema, PI
- Joseph Porter
- Gabor Simko
- and many others at ISIS – Vanderbilt

* Microsoft Research
Overview

- Context: OpenMETA-AVM
- Integration Challenge
- Design Flow Evolution and Consequences
- OpenMETA Integration Layers
- Semantic Integration
- Research Challenges
- Summary
End-to-end model- and component-based design and integrated manufacturing of a new generation of amphibious infantry vehicle – a complex, real-life cyber-physical system. From infrastructure to manufactured vehicle prototype in five years (2010-2014).

Engineering/economic goals:

- **Decrease development time by 80%** in defense systems (brings productivity consistent with other industries)
- Enable the adoption of **fabless design** and **foundry** concept in CPS: link design and manufacturing
- “**Democratize**” design by open source tool chain, crowed-sourced model library and prize-based design challenges
Achieving AVM goals require pushing the limits of “correct-by-construction” design using

- **Model-based Technologies**
  Computational models that predict properties of cyber-physical systems “as designed” and “as built”.
  
  **Challenge:** Develop domain-specific abstraction layers for complex CPS that are evolvable, heterogeneous, yet semantically sound and supported by tools.

- **Component-based Technologies**
  Reusable units of knowledge (models) and manufactured components.
  
  **Challenge:** Go beyond interoperability – find opportunities for composition where system-level properties can be computed from the properties of components
Major Components of AVM

- Model Library; Curation
- Vehicle Forge
- OpenMETA Tools
- FANG Competition Coordination
- Foundry

ISIS

- Curated Components
- Requirements, Test Benches
- Seed Designs, Scores
- Produces Design Data
- MFG Feedback

- Analysis Components, Designs, Design Spaces
- Use Tools
- Collaborate Using VF
- FANG Competitors
VehicleForge: A Gateway to the AVM Design Flow

**Components**
- Component discovery interface based on taxonomical- and faceted search
- Component view/visualization

**Design Projects**
- Self-provisioned collaboration tools
  - Wiki,
  - Discussion Forum,
  - Issue tracking for managing team work.
- Git/SVN repositories for design artifacts
- Project and tool-based permission control
- Notification and Messaging system (in e-mail or as Dashboard messages)
- Set of available tools is extensible

**Designers**
- Public profile to show recent activities and involvement in design projects
- Designer portfolio publishing résumé and for self-promotion
- Find designers based on expertise and résumé
- Private profile for customizing account and notification settings
- User dashboard showing feeds of activities from projects, public/private messages from other users, announcements from forge-message channels
VF Service Integration Platform

Component Vendors

Analysis & Simulation Service Providers

Manufacturers & Foundries

In-cloud Compute & Test bench Services

Exchange

Design & Manufacturing Components

Ontologies

Licensing

Ordering

Teams’ Design Storage

• MongoDB
• Git, SVN, Swift
• Apache SOLR
• TurboGears (Web Framework)
• REST Service APIs

• Sharing and Collaboration
• Cloud-based Analysis
• Access to Remote Resources

Browser-based

✓ Coordination and Monitoring Tools
✓ Design-space Evaluation and Visualizers
✓ Team-collaboration Tools
✓ Component Discovery and Subscription
✓ Service and Resource Allocation

CyPhy Desktop Tool Environment

http://vehicleforge.org/
Overview

- Context: OpenMETA-AVM
- Integration Challenge
- Design Flow Evolution and Consequences
- OpenMETA Integration Layers
- Semantic Integration
- Research Challenges
- Summary
## Notional OpenMETA Design Flow

<table>
<thead>
<tr>
<th>Architecture Design</th>
<th>Integrated Multi-physics/Cyber Design</th>
<th>Detailed Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Exploration</td>
<td>Modeling Simulation V&amp;V</td>
<td>Modeling Analysis</td>
</tr>
<tr>
<td>Rapid exploration</td>
<td>Exploration with integrated optimization and V&amp;V</td>
<td>Deep analysis</td>
</tr>
</tbody>
</table>

- **Architecture Design**
  - Design Space + Constraint Modeling
  - Architecture Modeling
  - Static Component Modeling (multiphysics)

- **Integrated Multi-physics/Cyber Design**
  - Design Space + Behavioral Constraint Modeling
  - Architecture Modeling
  - Dynamics Modeling (multiple abstractions and multiphysics)
  - CAD/assembly modeling
  - Coarse Manufacturing Constraint Modeling

- **Detailed Design**
  - Architecture Modeling
  - Detailed Domain Modeling
    - CAD
    - FEA; thermal, fluid…
    - Surrogate gen.
  - Detailed Mfg. modeling
  - RT SW modeling
## Analysis Tools

### Simulators

<table>
<thead>
<tr>
<th>Simulators</th>
<th>Lumped param. dynamics</th>
<th>OpenMod.</th>
<th>Open</th>
<th>Integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Modelica</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dymola</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jmodelica</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulink/Stateflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Formal specification, verification

<table>
<thead>
<tr>
<th>Formal specification, verification</th>
<th>Executable semantic specification</th>
<th>Microsoft</th>
<th>Open</th>
<th>Integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMULA</td>
<td>Verification of hybrid dynamics</td>
<td>SRI</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>HybridSAL</td>
<td>Verification using qualitative simulation</td>
<td>PARC</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>QRM</td>
<td>Probabilistic verification</td>
<td>SIFT/U. of Oxford</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>PRISMATIC</td>
<td>Probabilistic certificate of correctness</td>
<td>OSU</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>PCC</td>
<td>Schedule synthesis</td>
<td>KTH/Monash</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>Gecode</td>
<td>Digital modeling</td>
<td>Lund</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>TrueTime</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Design space exploration, optimization and visualization

<table>
<thead>
<tr>
<th>Design space exploration, optimization and visualization</th>
<th>Design space exploration</th>
<th>ISIS-VU</th>
<th>Open</th>
<th>Integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESERT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenMDAO</td>
<td>Parametric Optimization</td>
<td>NASA</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td>Project Analyzer</td>
<td>Dashboard and data visualizer</td>
<td>Georgia Tech.</td>
<td>Open</td>
<td></td>
</tr>
</tbody>
</table>
## Analysis Tools + Repositories

### CAD and FEA tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Provider</th>
<th>Type</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creo</td>
<td>CAD Models, product definition</td>
<td>PTC</td>
<td>Proprietary</td>
<td>Integrated</td>
</tr>
<tr>
<td>Abaqus</td>
<td>FEA, structural, thermal, fatigue, and modal analysis</td>
<td>Dassault Systemes</td>
<td>Proprietary</td>
<td>Integrated</td>
</tr>
<tr>
<td>Nastran</td>
<td>FEA, structural, thermal, fatigue, and modal analysis</td>
<td>MSC Software</td>
<td>Proprietary</td>
<td>Partially integrated</td>
</tr>
<tr>
<td>ANSYS</td>
<td>FEA, structural, thermal, fatigue, and modal analysis</td>
<td>ANSYS, Inc.</td>
<td>Proprietary</td>
<td>Planned</td>
</tr>
<tr>
<td>OpenFoam</td>
<td>CFD analysis</td>
<td>OpenFoam</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>METALink</td>
<td>Synchronization between GME</td>
<td>ISIS-VU</td>
<td>Open</td>
<td>Integrated</td>
</tr>
</tbody>
</table>

### Survivability Analysis

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Provider</th>
<th>Type</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCascade</td>
<td>Survivability Analysis, Ballistic</td>
<td>OpenCascade Org.</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>LS-DYNA</td>
<td>Survivability Analysis, Blast</td>
<td>LSTC</td>
<td>Proprietary</td>
<td>Partially Integrated</td>
</tr>
<tr>
<td>CTH Shock Physics</td>
<td>Survivability Analysis, Ballistic</td>
<td>Sandia</td>
<td>Proprietary</td>
<td>Partially Integrated</td>
</tr>
</tbody>
</table>

### VehicleForge (VF)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Provider</th>
<th>Type</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoDB</td>
<td>Document Oriented Database</td>
<td>10gen</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>Solr</td>
<td>Indexing</td>
<td>Apache</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>Redis</td>
<td>Distributed Key Store</td>
<td>Pivotal</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>Subversion</td>
<td>Optimistic VCS</td>
<td>Apache</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>Git</td>
<td>Distributed Optimistic VCS</td>
<td>Git</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>StepTools</td>
<td>WebGL CAD Visualization (STEP)</td>
<td>StepTools</td>
<td>Open</td>
<td>Integrated</td>
</tr>
</tbody>
</table>
# Frameworks

## Model, Tool and Simulation Integration

<table>
<thead>
<tr>
<th>Model, Tool and Simulation Integration</th>
<th>Description</th>
<th>VU/ISIS</th>
<th>Open Status</th>
<th>Integration Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCyPhy</td>
<td>Design tool integration framework</td>
<td>VU/ISIS</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>GME</td>
<td>Metaprogrammable modeling framework</td>
<td>VU/ISIS</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>Portico - HLA/RTI</td>
<td>Distributed Simulation Integration</td>
<td>Calytrix</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>FMU/FMI</td>
<td>Simulation Integration</td>
<td>Modelon</td>
<td>Open</td>
<td>Partially Integrated</td>
</tr>
</tbody>
</table>

## Collaboration and Repository

<table>
<thead>
<tr>
<th>Collaboration and Repository</th>
<th>Description</th>
<th>VU/ISIS</th>
<th>Open Status</th>
<th>Integration Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>VehicleForge</td>
<td>Model repository and collaborative design</td>
<td>VU/ISIS</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>TurboGears 2</td>
<td>Component Stack Web application framework</td>
<td>VU/ISIS</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>Allura</td>
<td>Forge application</td>
<td>SourceForge</td>
<td>Open</td>
<td>Integrated</td>
</tr>
<tr>
<td>Vulcan</td>
<td>Forge application framework</td>
<td>ISIS</td>
<td>Open</td>
<td>Integrated</td>
</tr>
</tbody>
</table>
Overview

- Context: OpenMETA-AVM
- Integration Challenge
- Design Flow Evolution and Consequences
- OpenMETA Integration Layers
- Semantic Integration
- Research Challenges
- Summary
Impact of Technology Evolution

CPS design methods and tools evolve...

- The tool chains must be open
- No future for standards that would freeze evolution of design methods
- Reusability of complex tools must be achieved by methods such as metamodeling and metaprogrammable tools
- Need for practical but rigorous semantic foundation that enables change without compromising soundness
**Model-Based Design**

**Key Idea:** Use models in domain-specific design flows and ensure that final design models are rich enough to enable production of artifacts with sufficiently predictable properties.

**Impact:** significant productivity increase in design technology

**Domain Specific Design Automation Environments:**
- Automotive
- Avionics
- Sensors...

**Tools:**
- Modeling
- Analysis
- Verification
- Synthesis

**Challenges:**
- Cost of tools
- Benefit only narrow domains
- Islands of Automation

**Mathematical and physical foundations**
Metaprogrammable Design Tools: “Freedom of Abstractions”

Key Idea: Ensure reuse of high-value tools in domain-specific design flows by introducing a metaprogrammable tool infrastructure.

VU-ISIS implementation: Model Integrated Computing (MIC) tool suite (http://repo.isis.vanderbilt.edu/downloads/)

Domain Specific Design Automation Environments:
- Automotive
- Avionics
- Sensors...

Metaprogrammable Tool Infrastructure
- Model Building
- Model Transf.
- Model Mgmt.
- Tool Integration

Explicit Semantic Foundation
- Structural
- Behavioral

Semantic Foundation Component Libraries

Layer

Meta

Design Requirements

Domain-Specific Environments

Production Facilities

Metaprogrammable Tools, Environments

def transition (fsm as FSM, s as State, t as Transition) {
  require s.active
  step exitState (s)
  step if t.outputEvent <> null then emitEvent (fsm, t.outputEvent)
  step activateState (fsm, t.dst)
Overview

- Context: OpenMETA-AVM
- Integration Challenge
- Design Flow Evolution and Consequences
- OpenMETA Integration Layers
- Semantic Integration
- Research Challenges
- Summary
Components span:
- Multiple physics
- Multiple domains
- Multiple tools

Component-based:
- Physical
- Cyber
- Cyber-Physical

Model-based:
- Model-Integrated
- Design and
- Manufacturing Process
Example: AVM Component Model

**Param./Property Interfaces**
- characterize
- configure

**Signal Interfaces**
- causal/directional
- logical conn.
- no power transfer

**Power Interfaces**
- acausal
- physical phen. (torque/angle..)
- power flow

**Structural Interfaces**
- named datums
- surface/axis/point
- mapped to CAD
## Components, Designs, Design Spaces

<table>
<thead>
<tr>
<th>Components</th>
<th>Designs</th>
<th>Design Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-contained building block</td>
<td>Instantiate and connect Components</td>
<td>Sets of parameterized architectures</td>
</tr>
<tr>
<td>Properties and Parameters</td>
<td>Parameters, behaviors, geometry are composed</td>
<td>Extended around seed designs</td>
</tr>
<tr>
<td>Wrapper for detailed domain models</td>
<td>Can be wrapped as a component</td>
<td>Shaped by design and manufacturability constraints</td>
</tr>
<tr>
<td>Aggregates the domain interfaces into a single set of component interfaces</td>
<td>Aggregates the component interfaces into a single set of system interfaces</td>
<td>Accumulates, evolves design and manufacturing knowledge</td>
</tr>
</tbody>
</table>

**Table:**
- **Components:**
  - Pars
  - Stru.
  - C

**Designs:**
- C\_1(p)
- C\_2(q)
- C\_1\_2(p,q)

**Design Spaces:**
- C\_1(p)
- C\_2(q)
- D\_1\_2(p,q)
OpenMETA Information Architecture

Physical
- Hierarchical decomposition
  - Physical domains: Electromagnetic, Thermal, Mechanical, Hydraulic, Electrical
  - Behavior abstractions

Cyber
- Hierarchical decomposition
  - Continuous/discrete time
  - Logical time
  - Discrete event
  - Behavior Abstractions

Epistemic Semantics
(Vocabularies)

CyPhyML
HBG
SignalFlow

Modeling Language Semantics
(Metamodels)
Case for Model Integration Languages...

Summary of OpenMETA – Approach to Information Architecture

- Model-Integration Language: CyPhyML
- Use of Metaprogrammable tools (MIC Tool Suite of ISIS/Vanderbilt)
- Use of Semantic Integration (see later)
• Using each component’s mappings to detailed domain models, system-level analyses is automatically composed.
  • Static properties
  • Multi-physics dynamics
  • Geometry
  • FEA
• META Test Benches provide an analysis context, including stimulus, loading, and monitoring.
• Test Benches include algorithms to produce Metrics, which are used to evaluate the design against Requirements.
• META Design Models are mapped to these Test Benches.
• Design Spaces can also be mapped to Test Benches, enabling rapid evaluation of a family of point designs.
Overview

- Context: OpenMETA-AVM
- Integration Challenge
- Design Flow Evolution and Consequences
- OpenMETA Integration Layers
- Semantic Integration
- Research Challenges
- Summary
## Integration Layers

<table>
<thead>
<tr>
<th>Physical Component Integration</th>
<th>Component Integration</th>
<th>Model Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bond Graph</td>
<td>Modelica Structure</td>
<td>BG Model</td>
</tr>
<tr>
<td>Modelica Equation</td>
<td>Composed Modelica</td>
<td>BG2 Model</td>
</tr>
<tr>
<td>behavior model</td>
<td>behavior model</td>
<td>CyPhy Structure</td>
</tr>
<tr>
<td>CyPhy Structure Model</td>
<td>CyPhy Composer</td>
<td>Modelica Equation</td>
</tr>
<tr>
<td>Composed Modelica Equation</td>
<td>SIM</td>
<td>VER</td>
</tr>
</tbody>
</table>
Need: Understanding Mapping to a Common Semantic Domain

// Model S generated on 10/02/2013 15:11:24
model S of Differential Equations at "BondSemantics.4ml" {
Eqaus(Effort(0), Effort(1))
Eqaus(Effort(0), Mul(Value("R_ta"), Flow(0)))
Eqaus(Effort(1), Effort(5))
Eqaus(Effort(2), Mul(Value("R_mt"), Flow(3)))
Eqaus(Effort(3), Effort(4))
Eqaus(Effort(4), Effort(6))
Eqaus(Flow(0), Flow(2))
Eqaus(Flow(2), Flow(3))
Eqaus(Flow(4), Mul(Value("C_m"), Derivative(Effort(4))))
Eqaus(Add(Add(Effort(1), Neg(Effort(2))), Neg(Effort(3))), Value("0"))
Eqaus(Add(Add(Flow(3), Neg(Flow(4))), Neg(Flow(5))), Value("0"))
equation
R_mti, mt = v.m;
R_ta ti, ta = v.ta;
C_m*der(v.m) = i.m;
v.m = R mt.p.v-R mt.n.v;
i.m = R mt.p.i=R mt.n.i;
v.ta = R ta.p.v-R ta.n.v;
i.ta = R ta.p.i=R ta.n.i;
v.m = R mp.p.v-R mn.v;
i.m = R mp.i=R mn.i;
sum(pin_p.i, i.m, i.ta) = 0;
pin_p.u = R mp.p;
sum(-i.m, pin_n1.i, -pin_n.i, -i.ta) = 0;
C_m.n.u = pin_n1.v;
sum(-pin_n.v, g.i, -i.m, pin_n1.i) = 0;
end component;

model component
Resistor R_ta(R=R);
equation
connect(pin_p, R_ta.p);
end component;

Resistor R_mt(R=R);
equation
connect(R_ta.p, R_mt.p);
end component;

Capacitor C_m(C=C);
equation
connect(R_ta.p, C_m.p);
end component;

model Resistor
package SUnits = Modelica.SIunits;
package Interfaces = Modelica.Electrical.Analog.Interfaces;
parameter SUnits.R ("Resistance"); // R = 1 ...
SUnits.Voltage v;
Interfaces.PositivePin p;
Interfaces.NegativePin n;
equation
0 = p.i + n.i;
v = p.v - n.v;
v = R*p.i;
end Resistor;
### Concept of “Semantic Integration”

<table>
<thead>
<tr>
<th>Physical Component Integration</th>
<th>Component Integration</th>
<th>Model Integration</th>
<th>Semantic Integration /FORMULA/</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Bond Graph" /></td>
<td><img src="image2" alt="Modelica" /></td>
<td><img src="image3" alt="Behavior Model" /></td>
<td><img src="image4" alt="BG Semantics" /></td>
</tr>
<tr>
<td><img src="image5" alt="Component Structure" /></td>
<td></td>
<td><img src="image6" alt="CyPhy Structure Model" /></td>
<td><img src="image7" alt="BG2Mod Semantics" /></td>
</tr>
<tr>
<td><img src="image8" alt="Equation Model" /></td>
<td></td>
<td><img src="image9" alt="Modelica Composer" /></td>
<td><img src="image10" alt="Mod. EQ Semantics" /></td>
</tr>
<tr>
<td><img src="image11" alt="Composition" /></td>
<td></td>
<td><img src="image12" alt="Composed Modelica Equation Model" /></td>
<td><img src="image13" alt="CyPhy Semantics" /></td>
</tr>
</tbody>
</table>
Convergence in Formal Framework: FORMULA

- History: Foundations for Embedded Systems ITR; Ethan Jackson at VU 2005-2008
- Microsoft Research (Bellevue & Aachen);
- Includes Satisfiability Modulo Theory Solver (Z3)
- http://research.microsoft.com/formula

- Foundation: Algebraic Data Types (ADT) and First-order logic with fixpoints (FPL)
- Parameterized with background theories (bit vectors, term algebras, etc.)
- Semantics is defined by constraint logic programming (CLP)
- Evolving structures; temporal logic
Metamodel and Formal Metamodel - ADTs

Metamodul of a simplified Acausal Bond Graph DSML

Meta-GME: Configures the metaprogrammable modeling Tool, GME

FORMULA
### Part of Structural Semantics for Acausal Bond Graphs

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td><code>domain AcausalBG extends AcausalBG_elements</code></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>`{</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>invalidBondDef := a is Bond, no Src(a,__)</td>
<td>Specifies invalid bond definition where a bond has no source.</td>
</tr>
<tr>
<td>26</td>
<td>invalidBondDef := a is Bond, no Dst(a,__)</td>
<td>Specifies invalid bond definition where a bond has no destination.</td>
</tr>
<tr>
<td>27</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>bondConn(a,x) :- Src(a,x); Dst(a,x).</td>
<td>Defines the connection between nodes a and x.</td>
</tr>
<tr>
<td>29</td>
<td>atLeastOneConnection(x) :- bondConn(_,x).</td>
<td>Checks if there is at least one connection for node x.</td>
</tr>
<tr>
<td>30</td>
<td>atLeastTwoConnections(x) :-</td>
<td>Checks if there are at least two connections for node x.</td>
</tr>
<tr>
<td>31</td>
<td>bondConn(a,x), bondConn(b,x), a != b.</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>exactlyOneConnection(x) :-</td>
<td>Checks if there is exactly one connection for node x.</td>
</tr>
<tr>
<td>33</td>
<td>atLeastOneConnection(x),</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>no atLeastTwoConnections(x).</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>invalidBlock := x is OnePort,</td>
<td>Specifies invalid block for one-port elements.</td>
</tr>
<tr>
<td>37</td>
<td>no exactlyOneConnection(x).</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>invalidBlock := x is TwoPort,</td>
<td>Specifies invalid block for two-port elements.</td>
</tr>
<tr>
<td>39</td>
<td>no exactlyTwoConnections(x).</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>invalidBlock := x is R, Src(_,x);</td>
<td>Specifies invalid block for resistor elements.</td>
</tr>
<tr>
<td>41</td>
<td>x is C, Src(_,x);</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>x is I, Src(_,x).</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>invalidBlock := x is TwoPort, no Src(_,x);</td>
<td>Specifies invalid block for two-port elements without a source.</td>
</tr>
<tr>
<td>44</td>
<td>x is TwoPort, no Dst(_,x).</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>conforms := !invalidBlock &amp;</td>
<td>Checks if the model conforms to the structural semantics.</td>
</tr>
<tr>
<td>47</td>
<td>!invalidBondDef &amp;</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>!invalidSrcDef &amp;</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>!invalidDstDef.</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

- Structural semantics is composed of constraints on model structure.
- Modeling tools need to check constraints during modeling.
- A well-formed model can be mapped into some behavior.
Specifying Behavioral Semantics

\[ D(Y, C) = \{ r \in R_Y \mid r = C \} \]

\[ \left[ \right]: R_Y \mapsto R_{Y'} \]

\[ D(Y', C') = \{ r \in R_{Y'} \mid r = C' \} \]

\[ \left[ \right]: R_{Y'} \mapsto R_{Y''} \]

1. domain AcausalBG_elements
   2. {
   3.   primitive SF := (id: String).
   4.   primitive Se := (id: String).
   5.   primitive R := (id: String).
   6.   //...
   7.   primitive TF := (id: String).
   8.   primitive GY := (id: String).
  10.   primitive OneJunction := (id: String).
   11.   Source := SF + Se.
   12.   //..
   13. }

1. domain DAEquations
   2. {
   3.   primitive Variable ::= (name: String, id: String).
   4.   primitive Param ::= (id: String).
   5.   primitive Neg ::= (Term).
   6.   primitive Inv ::= (Term).
   7.   primitive Eq ::= (Variable, Term).
   8.   primitive DiffEq ::= (Variable, Term).
   9.   primitive SumZero ::= (Sum).
  10.   Equation ::= Eq + DiffEq + SumZero.
  11. }
domain DFA {
  primitive Event ::= (lbl: Integer).
  primitive State ::= (lbl: Integer).
  primitive Transition ::= (src: State, trg: Event, dst: State).
  primitive Current ::= (st: State).
  nonDeterTrans := Transition(s, e, sp), Transition(s, e, tp), sp != tp.
  conforms := !nonDeterTrans.
}

transform Step<fire: in1.Event> from in1::DFA to out1::DFA {
  out1.Event(x) :- in1.Event(x).
  out1.Transition(s, e, sp) :- in1.Transition(s, e, sp).
  out1.Current(sp) :- in1.Current(s), in1.Transition(s, fire, sp).
  out1.Current(s) :- in1.Current(s),
    fail in1.Transition(s, fire, _).
}
CyPhyML - Modelica Power Connections

The behavioral semantics of Modelica power ports is the same as that of CyPhyML. For example, in electrical domain effort is voltage and flow is current in both CyPhyML and Modelica.

\[ \forall (x, y) \in P, (e_x - e_y + f_x - f_y) \]

where \( P \) is the set of Modelica - CyPhyML power port mappings.

CyPhyML - Modelica Signal Connections

The behavioral semantics of Modelica signal ports is the same as that of CyPhyML.

\[ \forall (x, y) \in P, (s_x = s_y) \]

where \( P \) is the set of Modelica - CyPhyML signal port mappings.

CyPhyML - SignalFlow Signal Connections

While signal ports in signal-flow are discrete-time ports, signal ports in CyPhyML are continuous-time. Thus, signal flow output signals are integrated into CyPhyML by means of a hold function.

\[ \forall (x, y) \in P, (s_y = \text{hold}(s_x)) \]

where \( P \) is the set of SignalFlow output - CyPhyML signal port mappings.
Overview

- Context: OpenMETA-AVM
- Integration Challenge
- Design Flow Evolution and Consequences
- OpenMETA Integration Layers
- Semantic Integration
- Research Challenges
- Summary
Research Challenges

- **Composition**
  - Managing heterogeneity
  - Decoupling in design spaces
  - Goal-driven model composition

- **Model transformation and model synthesis**
  - Backbone of complex tool chains
  - Modeling their semantics is essential
  - Exploring formal frameworks is an interesting research challenge

- **New composable frameworks/architectures**
  - Guaranteed properties (stability, time-compositionality, security properties..)
Model and software synthesis is central for model-based design flows

CPS creates heterogeneity -> new challenges

Semantic foundations and related tools are essential