A Compositional Semantics for Modelica-style Variable-structure Systems

Peter Pepper
Alexandra Mehlhase
Christoph Höger
Lena Scholz

Technische Universität Berlin
Introduction

Goals

- Compositional semantics
  - readability and simplicity
  - separate compilation
  - variable-structure systems (structure dynamics)

- Separation of concerns
  - "ideal semantics"
  - "solver semantics"

By contrast, Modelica has a monolithic, flattening-based semantics with a mixture of conceptual and numerics-oriented aspects.
① Compositional Semantics

- System
- Flattening
- Flat system
- Semantic interpretation
- Composition
- Components
- System model
- Monolithic
- Compositional
Ideal Semantics

Ideal Semantics

Ideal semantics . . .

- lives in the realm of pure mathematics
- focuses on modeling concepts
- ignores numerical problems

Two levels of modeling concepts:

- **fixed-structure systems** (classical Modelica)
  - simplify semantic presentation
- **variable-structure systems** (extended Modelica)
  - prepare extended concepts (dynamic systems, separate compilation)
**Ideal Semantics**

**Base Case: Atomic Fixed-Structure Component**

**Syntax:**

Class \( C = (V_e, V_l, E) \)
- \( V_e \): external variables
- \( V_l \): local variables
- \( E \): hybrid DAEs

**Semantics:**

Model \( M = (F_e, F_i) \)
- \( F_e \): set of functions \( (\cong V_e) \)
- \( F_i \): set of functions \( (\cong V_l) \)

\[
\begin{align*}
\text{Class Pendulum} & \\
\text{PARAM Length } L & \\
\text{PARAM Mass } m & \\
\text{CONST Acceleration } g & \\
\text{Length } x & \\
\text{Length } y & \\
\text{Force } f & \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m \cdot \ddot{x} = -\frac{x}{L} \cdot f )</td>
</tr>
<tr>
<td>( m \cdot \ddot{y} = -m \cdot g - \frac{y}{L} \cdot f )</td>
</tr>
<tr>
<td>( x^2 + y^2 = L^2 )</td>
</tr>
</tbody>
</table>

\( M \models E \)
Composition is essentially a “pushout”:

\[
\text{Mod}(S) = (\text{Mod}(C_1) \times \text{Mod}(C_2)) \upharpoonright E_{\text{conn}}
\]

\[
def \quad \{ M_1 \cup M_2 \mid \begin{align*}
    & M_1 \in \text{Mod}(C_1), \\
    & M_2 \in \text{Mod}(C_2), \\
    & M_1\mid_{V_e} = M_2\mid_{V_e}, \\
    & M_1 \cup M_2 \models E_{\text{conn}}
\end{align*} \}
\]

Note: Name clashes avoided by scoping rules (compiler)
**Ideal Semantics**

**Subsystems**

<table>
<thead>
<tr>
<th>Class $S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(V_e)$</td>
</tr>
<tr>
<td>$V_S$</td>
</tr>
<tr>
<td>$E_S$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class $C_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
</tr>
<tr>
<td>$E_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class $C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_2$</td>
</tr>
<tr>
<td>$E_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class $C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_3$</td>
</tr>
<tr>
<td>$E_3$</td>
</tr>
</tbody>
</table>

\[
\text{Mod}(S) = \left( \text{Mod}(C_1) \otimes \cdots \otimes \text{Mod}(C_n) \right) \mid \left( E_S \cup E_{\text{conn}_1} \cup \cdots \cup E_{\text{conn}_k} \right)
\]
Variable-Structure Systems

**Variable-structure systems**

Base case: components with modes

```
class StringPendulum

Mass m
Acceleration g
Length L

<table>
<thead>
<tr>
<th>Swing</th>
<th>Fall</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle ( \varphi )</td>
<td>Length ( x, y, r )</td>
</tr>
<tr>
<td>force ( F )</td>
<td>Velocity ( vx, vy )</td>
</tr>
<tr>
<td>( \ddot{\varphi} = -\frac{k}{l} \cdot \sin(\varphi) )</td>
<td>( m \cdot \dot{vx} = 0 )</td>
</tr>
<tr>
<td>( F = m \cdot g \cdot \cos(\varphi) + m \cdot L \cdot \dot{\varphi}^2 )</td>
<td>( m \cdot \dot{vy} = -g \cdot m )</td>
</tr>
<tr>
<td>( r = L \Rightarrow \ldots )</td>
<td>( r = \sqrt{x^2 + y^2} )</td>
</tr>
</tbody>
</table>
```

Pepper et al., Modelica Semantics
Variable-Structure Systems

Changing Topology

Modes can have different components

Mode Roll

\[
\begin{align*}
\text{Class Car} & : \text{m} : \text{Mass}, \text{v}, \text{v}_x, \text{v}_y : \text{Speed}, \\
\text{Class Ball} & : \text{m} : \text{Mass}
\end{align*}
\]

\[
\text{blocked } \Rightarrow \\
\text{Ball}.v_x = \text{Car}.v_x \\
\text{Ball}.v_y = \text{Car}.v_y
\]

Mode Bounce

\[
\begin{align*}
\text{Class Ball} & : \text{m} : \text{Mass}, \\
\text{v}_x, \text{v}_y : \text{Speed}, \\
\end{align*}
\]
Variable-Structure Systems

The Full Picture of Dynamism

Class $C$

$V, E$

Class $C_1$

Class $C_2$

Class $C_3$

Mode $M_1$

$V_{M_1}, E_{M_1}$

$g_1 \Rightarrow a_1$

$g_2 \Rightarrow a_2$

Mode $M_2$

$V_{M_2}, E_{M_2}$

$a_0$
Variable-Structure Systems

**Modes**

Consider component $K$ of class $C = (V, E, S, D)$:

- Component lifetime $T_K = [t^\alpha, t^\omega)$ with $t^\alpha < t^\omega$
- Mode lifetime $T_{M_i} = [t_{M_i}^\alpha, t_{M_i}^\omega)$ (see next slide)
- Semantics during mode $M_i$: $C_{fix} \otimes S_{M_i} \mid E_{conn}$

Issue: global variables may exhibit different behaviors in different modes

Example: $M_1 : x_1 = 1$ \hspace{1cm} $M_2 : x_2 = 2$

$x = x_1$ \hspace{1cm} $x = x_2$
Variable-Structure Systems

**Transitions**

Transition point \( t^* \) is defined by

\[ t^* = \text{smallest } t, \quad t_1^\alpha < t \text{ such that} \]

- \( \text{guard}(t) = \text{true} \)
- \( \forall \tau, \quad t_1^\alpha < \tau < t. \quad \text{guard}(\tau) = \text{false} \)

Constraint: \( t_1^\alpha < t^* < t_2^\omega \) (modes shall not degenerate to zero length)

Problematic issues to be considered in language design:

- self loops
- conflicting guards (nondeterminism vs. error)
2 Simulation Semantics

Numerical solver issues:

- **Discretization**
  ideal functions over $\mathbb{R}$
  \[ f : \mathbb{R} \to \ldots \]
  are replaced by discrete sampling times
  \[ \hat{f} : \mathbb{T} \to \ldots \]
  where $\mathbb{T} = \{t_i\}$ with $t_{i+1} = t_i + h_i$ for step sizes $h_i$.

- **Precision**
  - **rounding errors** (limited number size)
  - **discretization errors** ($\tilde{x}_{\text{solver}} \approx x(t)$)
  - **approximation errors** (e.g. Newton algorithm)
  - **modeling errors** (parameters, input values)
  - **event detection** (e.g. zero crossing)
**Solver Semantics**

**Uncertainty** (work in progress)

- Simplest approach is interval-based: \[ x \leadsto \tilde{x} = x \pm \omega \]
- Semantics is defined relative to notion of "validity"; hence
  - new concept for validity: \( (\tilde{A} \models E) \)
  - new concept for models: \( \tilde{Mod}(S) = \{ \tilde{A} | \tilde{A} \models S \} \)
  - most other constructs (composition etc.) remain unchanged, since \( (\cdots \otimes \ldots | \ldots) \) is defined relative to validity
- Critical issue: guards
  - \( \tilde{t}^* \) is defined by \( \tilde{\text{guard}}(\tilde{t}^*) = \text{true} \)
  - \( \tilde{\text{guard}}(\tau) = \text{false} \) for all \( \tau \) with \( \tilde{t}_1^\alpha < \tau < \tilde{t}^* \)
  - interval \( \tilde{T}_1 = [\tilde{t}_1^\alpha, \tilde{t}^*] \) is blurred
  - computation traces can be changed (w.r.t. ideal semantics)
  - analysis techniques are field for intensive research in Numerics
Conclusion

Goal: Compositionality of semantics

- supports variable-structure systems
- supports separate compilation

Goal: Modelica targeted to engineers $\leadsto$ semantics as well

- semantics streamlined for Modelica
  $\leadsto$ no embedded DSL (such as Hydra/Haskell)
- as simple and understandable as possible
  $\leadsto$ no large mathematical framework (such as in CIF or Ptolemy)

Goal: Separation of concerns $\leadsto$ ideal vs. solver semantics

- clear description of modeling principles
- clear description of solver-based constraints
- adaptability to various solvers and solver technologies