A Compositional Semantics for Modelica-style Variable-structure Systems

Peter Pepper
Alexandra Mehlhase
Christoph Höger
Lena Scholz

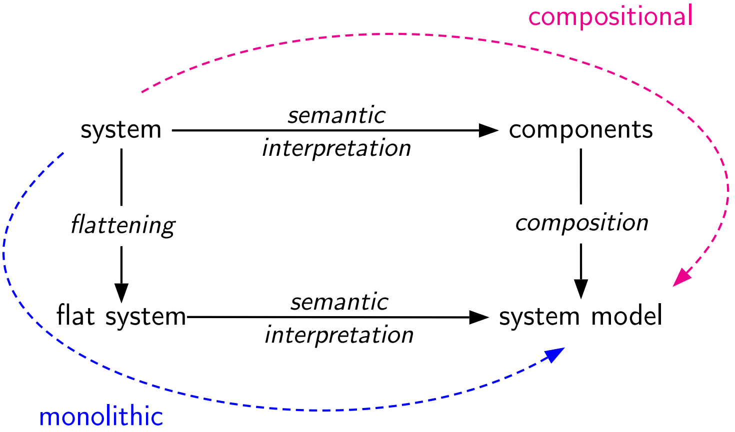
Technische Universität Berlin

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*



Ideal Semantics

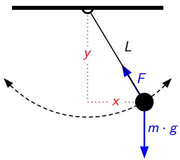
Ideal semantics . . .

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

Two levels of modeling concepts:

- fixed-structure systems (classical Modelica)
  $\rightsquigarrow$ simplify semantic presentation
- variable-structure systems (extended Modelica)
  $\rightsquigarrow$ prepare extended concepts (dynamic systems, separate compilation)

Base Case: Atomic Fixed-Structure Component



Class *Pendulum*	
PARAM *Length L*	parameters
PARAM *Mass m*	
CONST *Acceleration g*	
Length x	variables V
Length y	
Force f	
$m \cdot \ddot{x} = -\frac{x}{L} \cdot f$	equations E
$m \cdot \ddot{y} = -m \cdot g - \frac{y}{L} \cdot f$	
$x^2 + y^2 = L^2$	

Syntax:

Class $C = (V_e, V_l, E)$

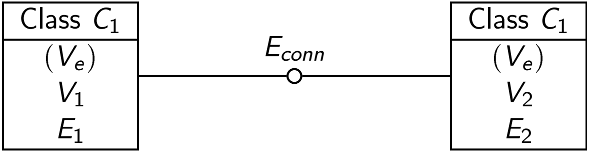
- V_e: external variables
- V_l: local variables
- E: hybrid DAEs

Semantics:

Model $M = (F_e, F_l)$

- F_e set of functions $(\widehat{=} V_e)$
- F_i set of functions $(\widehat{=} V_i)$
- $M \models E$

Composition

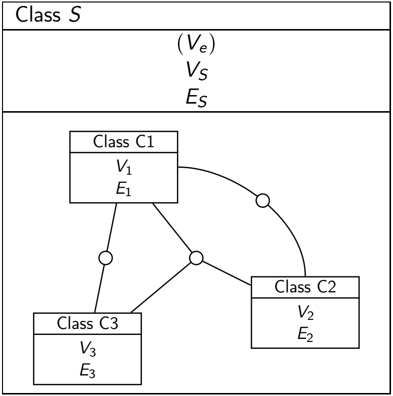


Composition is essentially a "pushout":

$$
\begin{aligned}
Mod(S) &= (Mod(C_1) \otimes Mod(C_2)) \mid E_{conn} \\
&\stackrel{def}{=} \{\, M_1 \cup M_2 \mid \; M_1 \in Mod(C_1), \\
&\qquad\qquad\qquad\quad M_2 \in Mod(C_2), \\
&\qquad\qquad\qquad\quad M_1|_{V_e} = M_2|_{V_e}, \\
&\qquad\qquad\qquad\quad M_1 \cup M_2 \models E_{conn} \,\}
\end{aligned}
$$

Note: Name clashes avoided by scoping rules (compiler)

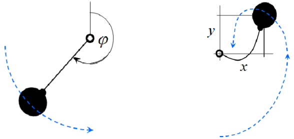
Subsystems



$$Mod(S) = \big(Mod(C_1) \otimes \cdots \otimes Mod(C_n)\big) \mid (E_S \cup E_{conn_1} \cup \cdots \cup E_{conn_k})$$

Variable-structure systems

Base case: components with modes



class StringPendulum

Mass m
Acceleration g
Length L

$\varphi = \ldots$

Swing

Angle φ
Force F
$\ddot{\varphi} = -\frac{g}{L} \cdot sin(\varphi)$
$F = m \cdot g \cdot cos(\varphi) + m \cdot L \cdot \dot{\varphi}^2$

$F < 0 \Rrightarrow x = \ldots \; y = \ldots \; vx = \ldots \; vy = \ldots$

$r = L \Rrightarrow \ldots$

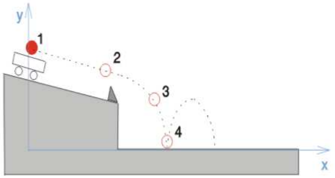
Fall

Length x, y, r
Velocity vx, vy
$m \cdot \dot{vx} = 0$
$m \cdot \dot{vy} = -g \cdot m$
$r = sqrt(x^2 + y^2)$

Variable-Structure Systems

Changing Topology

Modes can have different components



Mode *Roll*

slope : *Angle*, *dist* : *Length*, . . .

Class *Car*

m : *Mass*
v, v_x, v_y : *Speed*,
. . .

Class *Ball*

m : *Mass*

$$blocked \quad \Rrightarrow$$
$$Ball.v_x = Car.v_x$$
$$Ball.v_y = Car.v_y$$

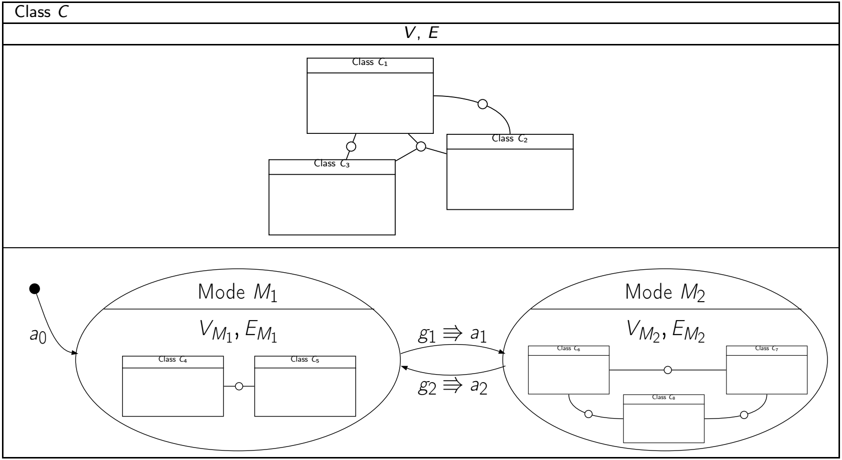
Mode *Bounce*

height : *Length*, . . .

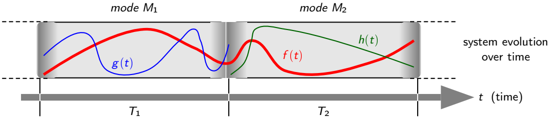
Class *Ball*

m : *Mass*,
v_x, v_y : *Speed*
. . .

The Full Picture of Dynamism



Modes



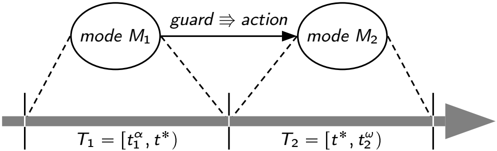
Consider component K of class $C = (V, E, S, D)$:

- Component lifetime $T_K = [t^\alpha, t^\omega)$ with $t^\alpha < t^\omega$
- Mode lifetime $T_{M_i} = [t^\alpha_{M_i}, t^\omega_{M_i})$ (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

$$\text{Example:} \quad M_1: \begin{array}{l} x_1 = 1 \\ x = x_1 \end{array} \qquad M_2: \begin{array}{l} x_2 = 2 \\ x = x_2 \end{array}$$

Transitions



Transition point t^* is defined by

$$t^* = \text{smallest } t,\ t_1^\alpha < t \text{ such that } \begin{array}{l} guard(t) = true \\ \forall \tau, t_1^\alpha < \tau < t.\ guard(\tau) = false \end{array}$$

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* ($\tilde{x}_{solver} \approx x(t)$)
  - *approximation errors* (e.g. Newton algorithm)
  - *modeling errors* (parameters, input values)
  - *event detection* (e.g. zero crossing)

Uncertainty (work in progress)

- Simplest approach is interval-based: $x \quad \rightsquigarrow \quad \tilde{x} = x \pm \omega$
- Semantics is defined relative to notion of "validity"; hence
  - new concept for validity: $(\widetilde{A} \Vdash E)$
  - new concept for models: $\widetilde{Mod}(S) = \{ \ \widetilde{A} \mid \widetilde{A} \Vdash S \ \}$
  - most other constructs (composition etc.) remain unchanged, since $(\cdots \otimes \ldots \mid \ldots)$ is defined relative to validity
- Critical issue: guards
  - $\tilde{t}^*$ is defined by $\quad \widetilde{guard}(\tilde{t}^*) = true$
    $\widetilde{guard}(\tau) = false$ for all τ with $\tilde{t}_1^\alpha < \tau < \tilde{t}^*$

$\rightsquigarrow$ interval $\tilde{T}_1 = [\tilde{t}_1^\alpha, \tilde{t}^*)$ is blurred

$\rightsquigarrow$ computation traces can be changed (w.r.t. ideal semantics)

$\rightsquigarrow$ 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 $\rightsquigarrow$ semantics as well

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

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

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