Abstract Simulation: a Static Analysis of Simulink Models

Alexandre Chapoutot and Matthieu Martel

LIP6 - Université Pierre et Marie Curie
Paris, France

ICESS’09
May 25th 2009
Purpose of the study

Study of control-command systems
Example: electronic throttle controller in cars

Four components:

- One software (the controller)
- One physical environment (the controlled element)
- Some actuators (to act on the environment)
- Some sensors (to get information on the environment)

Heterogeneity of components: HYBRID SYSTEMS
Purpose of the study

Study of control-command systems
Example: electronic throttle controller in cars

Four components:
- **One software** (the controller)
- One physical environment (the controlled element)
- Some actuators (to act on the environment)
- Some sensors (to get information on the environment)

Heterogeneity of components: HYBRID SYSTEMS
Purpose of the study

Study of control-command systems
Example: electronic throttle controller in cars

Four components:
- **One software** (the controller)
- **One physical environment** (the controlled element)
- Some actuators (to act on the environment)
- Some sensors (to get information on the environment)

Heterogeneity of components: HYBRID SYSTEMS
Purpose of the study

Study of control-command systems
Example: electronic throttle controller in cars

Four components:
- **One software** (the controller)
- **One physical environment** (the controlled element)
- **Some actuators** (to act on the environment)
- Some sensors (to get information on the environment)

Heterogeneity of components: **HYBRID SYSTEMS**
Purpose of the study

Study of control-command systems
Example: electronic throttle controller in cars

Four components:
- **One software** (the controller)
- **One physical environment** (the controlled element)
- **Some actuators** (to act on the environment)
- **Some sensors** (to get information on the environment)

Heterogeneity of components: **HYBRID SYSTEMS**
Purpose of the study

Study of control-command systems
Example: electronic throttle controller in cars

Four components:
- **One software** (the controller)
- **One physical environment** (the controlled element)
- **Some actuators** (to act on the environment)
- **Some sensors** (to get information on the environment)

Heterogeneity of components: **HYBRID SYSTEMS**
Design in Simulink

Simulink model: program describing the hybrid system
Numerical approximations

Approximations in the real system

Approximations in the Simulink model
Abstract Simulation: overview

Fact: Numerical approximations may have bad influence on computations

But: Simulations of Simulink models is often used as validation method!

Abstract Simulation: contributions

Static analysis by abstract interpretation of Simulink models

In order to study numerical precision

- Formal definition of the semantics of a Simulink subset
- Definition of numerical abstract domains
Outline

1. Simulink language and its semantics
2. Abstract numerical domains
3. Case study
Outline

1. Simulink language and its semantics
2. Abstract numerical domains
3. Case study
Overview of the Simulink language

**Generality**
- Matlab extension
- Graphical tool **for designing** embedded systems
- **Numerical simulation** engine
- Many specific libraries: signal processing, control theory, etc.
- **De facto standard in the industry** (in particular in automotive)

**Features**
- Many data formats (float and fixed-point)
- Numerical algorithms: Euler, Runge-Kutta, etc.
- Code generator, debugger, etc.

**Various kinds of models**
- Continuous-time
- Discrete-time
- **Hybrid**
Modelling with Simulink

Simulink model
Model of the electronic throttle control

Simulink model of the system

Mathematical model of the throttle

\[
\begin{align*}
T(t) &= \text{Direction} \times \text{Effort} \times C_s \\
\dot{\omega}(t) &= \frac{1}{J}(-K_s(\theta(t) - \theta_{eq}) - K_d\omega(t) + T(t)) \quad 0 < \theta < \pi/2 \\
\text{if } (\theta < 0 \land \text{sgn}(\dot{\omega}(t) = -1)) \lor ((\theta > \pi/2 \land \text{sgn}(\dot{\omega}(t) = 1))) \\
\text{then } \dot{\omega}(t) &= 0
\end{align*}
\]
Model of the electronic throttle control

Simulink model of the throttle

```
Simulink model of the throttle

2  In2
   L_23
   ×
   Product
   Cs
   1  Gain
   I_25
   I_26
   I_27
   teq
   Constant
   Add
   1
   Out1

1  In1
   L_24
   l_25
   Product
   l_26
   l_27
   l_38
   l_39

Gain
Ks
Kd
Gain1
Gain2
Gain3
Integrator1
Integrator
1/J
In2
2
In1

Relational Operator
Sign
Logical Operator
NOT
Product
Product1
1/J
Gain

Add2
Add

l_30
l_31
l_32
l_33
l_34
l_35
l_36
l_37
l_39
l_28
```
Model of the electronic throttle control

Simulink model of the throttle (continuous-time system)

Temporal operation: Integrator (continuous-time)
Model of the controller

Simulink model of the system

Mathematical model of the controller (PI regulator)

\[
\begin{align*}
    y(k) &= y_p(k) + y_i(k) \\
    y_p(k) &= K_p e(k) \\
    y_i(k + 1) &= y_i(k) + K_i T_s e(k) \quad 0 < y_i(k) < 1
\end{align*}
\]

Papillon des gaz

L43:

movl 16(%ebp), %eax
movl 12(%eax), %eax
movl %eax, -16(%ebp)
movl -12(%ebp), %eax
movl %eax, (%esp)
call L_free$stub

L34:

cmpl $0, -16(%ebp)
jne L37
movl $10, (%esp)
call L_putchar$stub

Capteurs
Actionneurs
Model of the controller

Simulink model of the regulator

![Simulink diagram](image-url)
Model of the controller

Simulink model of the regulator (discrete-time system)

Temporal operation: Unit Delay (discrete-time)
Numerical simulation: semantics of Simulink

Simulink model

Numerical simulation

1: Initial states
2: repeat
3: Read inputs
4: Compute outputs
5: Compute state
6: Compute next time-step
7: until end of time simulation

What is a state?
State: ”previous iteration values needed to compute the current output”

Informal definition of the semantics of Simulink
Formalization of the Simulink’s semantics

<table>
<thead>
<tr>
<th>Name</th>
<th>Bloc</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td><img src="image" alt="Constant" /></td>
<td>${\ell_1 = c, \emptyset}$</td>
</tr>
<tr>
<td>Add</td>
<td><img src="image" alt="Add" /></td>
<td>${\ell_3 = \ell_1 + \ell_2, \emptyset}$</td>
</tr>
<tr>
<td>Switch</td>
<td><img src="image" alt="Switch" /></td>
<td>${\ell_4 = \text{if}(p(\ell_1), \ell_2, \ell_3), \emptyset}$</td>
</tr>
<tr>
<td>Integrator</td>
<td><img src="image" alt="Integrator" /></td>
<td>${\ell_2(t) = \rho(t), \dot{\rho}(t) = \ell_1(t)}$</td>
</tr>
<tr>
<td>UnitDelay</td>
<td><img src="image" alt="UnitDelay" /></td>
<td>${\ell_2(k) = \rho(k), \rho(k + 1) = \ell_1(k)}$</td>
</tr>
</tbody>
</table>

Two kinds of equations:
- Output equations
- State equation: temporal operation in the semantics

Remark: Use the equivalence $y(t) = \int x(z)dz \equiv \dot{y}(t) = x(t)$
# Formalization of the Simulink’s semantics

<table>
<thead>
<tr>
<th>Name</th>
<th>Bloc</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td><img src="image" alt="Constant Block" /> $c \xrightarrow{} \ell_1$</td>
<td>${\ell_1 = c, \emptyset}$</td>
</tr>
<tr>
<td>Add</td>
<td><img src="image" alt="Add Block" /> $\ell_1 \rightarrow \ell_2 \rightarrow \ell_3$</td>
<td>${\ell_3 = \ell_1 + \ell_2, \emptyset}$</td>
</tr>
<tr>
<td>Switch</td>
<td><img src="image" alt="Switch Block" /> $\ell_1 \rightarrow \ell_2 \rightarrow \ell_4$</td>
<td>${\ell_4 = \text{if}(p(\ell_1), \ell_2, \ell_3), \emptyset}$</td>
</tr>
<tr>
<td>Integrator</td>
<td><img src="image" alt="Integrator Block" /> $\ell_1 \rightarrow 1/s \rightarrow \ell_2$</td>
<td>${\ell_2(t) = \rho(t), \dot{\rho}(t) = \ell_1(t)}$</td>
</tr>
<tr>
<td>UnitDelay</td>
<td><img src="image" alt="Unit Delay Block" /> $\ell_1 \rightarrow 1/z \rightarrow \ell_2$</td>
<td>${\ell_2(k) = \rho(k), \rho(k + 1) = \ell_1(k)}$</td>
</tr>
</tbody>
</table>

Two kinds of equations:
- **Output equations**
- **State equation**: temporal operation in the semantics

Remark: Use the equivalence $y(t) = \int x(z)dz \equiv \dot{y}(t) = x(t)$
Formalization of the Simulink’s semantics

<table>
<thead>
<tr>
<th>Name</th>
<th>Bloc</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td><img src="image" alt="Constant Bloc" /></td>
<td>${\ell_1 = c, \emptyset}$</td>
</tr>
<tr>
<td>Add</td>
<td><img src="image" alt="Add Bloc" /></td>
<td>${\ell_3 = \ell_1 + \ell_2, \emptyset}$</td>
</tr>
<tr>
<td>Switch</td>
<td><img src="image" alt="Switch Bloc" /></td>
<td>${\ell_4 = \text{if}(p(\ell_1), \ell_2, \ell_3, \emptyset)}$</td>
</tr>
<tr>
<td>Integrator</td>
<td><img src="image" alt="Integrator Bloc" /></td>
<td>${\ell_2(t) = \rho(t), \dot{\rho}(t) = \ell_1(t)}$</td>
</tr>
<tr>
<td>UnitDelay</td>
<td><img src="image" alt="Unit Delay Bloc" /></td>
<td>${\ell_2(k) = \rho(k), \rho(k + 1) = \ell_1(k)}$</td>
</tr>
</tbody>
</table>

Two kinds of equations:
- Output equations
- State equation: temporal operation in the semantics

Remark: Use the equivalence $y(t) = \int x(z)dz \equiv \dot{y}(t) = x(t)$
Formalization: semantics of Simulink

Simulink model

System of equations

\[
\begin{align*}
\ell_0 &= \text{In}1 \\
\ell_1 &= \ell_0 - \ell_6 \\
\ell_2 &= S_1(\ell_1) \\
(\ell_3, \ell_4) &= S_2(\ell_2) \\
\ell_5 &= S_3(\ell_3, \ell_4) \\
\ell_6 &= S_4(\ell_5)
\end{align*}
\]

Numerical simulation

1: Initial states
2: repeat
3: Read inputs
4: Compute outputs
5: Compute state evolution
6: Compute next time-step
7: until time of end

Approximation: Discretization of continuous-time functions

\[
\dot{y}(t) = x(t)\quad \text{transformed into} \quad \eta(k+1) = \text{solver}(\eta(k), x(k))
\]

\[
y(k) = \eta(k)
\]

Remark: several numerical integration methods then several semantics of Simulink
Abstract simulation vs Numerical simulation

### Numerical simulation

1. Initial states
2. \textbf{repeat}
3. Read inputs
4. Compute outputs
5. Compute state evolution
6. Compute next time-step
7. \textbf{until} time of end
Abstract simulation vs Numerical simulation

Abstract simulation
1: Initial states
2: while not fixed-point do
3: Read inputs
4: Compute outputs
5: Compute state
6: Compute next time-step
7: end while

Semantic approach of Simulink models.

Result: Evaluation of the robustness of control-command software to numerical approximations for a set of inputs

- Represent signals by interval sequences with finite lengths (domain of sequences)
## Outline

1. Simulink language and its semantics
2. Abstract numerical domains
3. Case study
Goal: Compute the distance between the results of the mathematical model and the results of the numerical simulation.

But: Different kinds of numerical approximations in the two types of systems.
Numerical approximation in continuous-time systems

Measure the distance between:

- the numerical simulation (using numerical integration methods)
- the mathematical results through perfect sensor
Numerical approximation in continuous-time systems

For a system of differential equations:
- Simulink: interval sequence approximate the solution
- Guarantee solution: interval sequence over-approximate the real solutions [Bouissou’08, Nedialkov’05]

Correction criteria:
- distance between the guarantee solution and the interval solutions following Simulink’s method
Set of values: intervals

Intervals [Moore’66] [Cousot&Cousot’77]

\[ [a, b] = \{ x \in \mathbb{R} \mid a \leq x \leq b \} \]

Dependency problem: \( X - X \neq 0 \) in general

Solution: centered form

Middle value theorem

\( \textbf{f}([a, b]) \subseteq \textbf{f}(m) + [\textbf{f}']([a, b])([a, b] - m) \)

\( m \) centered of the interval \([a, b]\)
Numerical approximation in discrete-time systems

Evaluate the distance between:

- the result of a floating-point computation
- and the result of the same computation did with real arithmetic
Computing the rounding errors

Domain of floating-point with errors numbers [Goubault’01, Martel’02]

Idea: breaking up a real value into

- Floating-point value \( f \)
- Rounding error \( e \): the distance between the real value and the floating-point value (\( \downarrow \))

Computation rules: e.g. addition

\[
a = (f_a, \ e_a) \quad \text{and} \quad b = (f_b, \ e_b)
\]

\[
a + b = (f_a + F f_b, \ e_a + e_b + \downarrow (f_a + f_b))
\]

Correction criteria:

- Smaller the errors are, more precise the floating-point result is (i.e. closer to the real result)
Outline

1. Simulink language and its semantics

2. Abstract numerical domains

3. Case study
Case study: electronic throttle controller

Simulink model

```
L43:
    movl  -16(%ebp), %eax
    movl   12(%eax), %eax
    movl    %eax, -16(%ebp)
    movl   -12(%ebp), %eax
    movl    %eax, (%esp)
    call    L_free$stub

L34:
    cmpl  $0, -16(%ebp)
    jne     L37
    movl    $10, (%esp)
    call    L_putchar$stub
```
Case study: electronic throttle controller

Model input: set of periodic inputs

Behavior of the throttle

```
L43:
    movl    -16(%ebp), %eax
    movl    12(%eax), %eax
    movl    %eax, -16(%ebp)
    movl    -12(%ebp), %eax
    movl    %eax, (%esp)
    call    L_free$stub

L34:
    cmpl    $0, -16(%ebp)
    jne     L37
    movl    $10, (%esp)
    call    L_putchar$stub
```
Case study: electronic throttle controller

Conclusion: the numerical results are close to the mathematical ones (errors are very small)
Conclusion

In summary

- Definition of a static analysis by abstract interpretation of Simulink models
- Formal verification of the numerical precision of the design of control-command systems
- Method taking into account all the elements of control-command systems

Future works among several

- Increase the subset of Simulink language (e.g. Stateflow)
- Numerical precision of fixed-point arithmetic
- A step further in validation: temporal properties