Rewriting-Logic-Based Formal Modeling and Analysis of Interacting Hybrid Systems

Muhammad Fadlisyah
Erika Ábrahám
Peter Csaba Ölveczky
Real-Time Maude

• Intuitive and expressive formalism for real-time systems
  – algebraic equational specification: data types
  – Instantaneous rewrite rules: instantaneous transitions
  – tick rewrite rules: time elapse

• Object oriented

• Analysis commands:
  – simulation
  – reachability analysis
  – LTL model-checking
Modeling in Real-Time Maude

Real-time rewrite theory (S,E,IR,TR)

S: signature
E: conditional equations

\begin{align*}
  eq & t=t' \\
  ceq & t=t' \text{ if } \text{cond}
\end{align*}

IR: instantaneous rules
TR: tick rules

\begin{align*}
  rl \ [l] & : \{t\} \Rightarrow \{t'\} \text{ in time } T \\
  crl \ [l] & : \{t\} \Rightarrow \{t'\} \text{ in time } T \text{ if } \text{cond}
\end{align*}

Classes and objects (also inheritance):

\begin{align*}
  class & C \mid \text{att}1: s1, \ldots, \text{att}n : sn . \\
  < & o : C \mid \text{att}1 : v1, \ldots, \text{att}n : vn >
\end{align*}
Simulation: timed rewrite

\[(\text{trew } t \text{ in } \text{time } <= T .)\]

Find earliest:

\[(\text{find earliest } t =>^* t').\]

Time-bounded LTL model checking:

\[(mc t |= t \text{ formula in time } <= T .)\]
Expressiveness vs Analytic Power

• Suited to model a wide range of complex systems
• Analysis: w.r.t. a set of time sampling strategies:
  – In general not all behaviour covered
  – Complete and sound analysis for interesting classes of systems
• Analysis for systems with
  – Advanced functions/data types and unbounded data structures
  – Different models of concurrency
• Example:
  – CASH state-of-the-art scheduling algorithm
  – Large communication protocols and sensor network algorithms
Motivation

• We extend Real-Time Maude to hybrid system modeling and analysis:
  – Object-oriented
  – Algebraic
  – Expressive
  – Simulation
  – Model checking

• Aim at complex hybrid systems
  – Need a *modular* way of defining the continuous dynamics of a complex hybrid system
Outline

Modeling Framework

Execution Framework

Case Studies
Modeling Physical Systems: *Effort* and *Flow* Approach

**Interaction:** energy

*Flow variable:* the energy flow

*Effort variable:* energy of the component

**Example:** Thermal System

*effort:* temperature

*flow:* heat flow rate
Effort/Flow In Different Physical Systems

Mechanical translation
- **effort**: force
- **flow**: velocity

Electrical
- **effort**: voltage
- **flow**: current

Fluidic
- **effort**: pressure
- **flow**: volume flow rate

Mechanical rotation
- **effort**: torque
- **flow**: angular velocity
Physical System Components

*Physical Entities and Physical Interactions*
Physical System Components

A *Physical Entity*
- *Effort* variable
- Interacts with one or more *Physical Entities*
A Physical Interaction
- Flow variable
- Connects two Physical Entities
Example: A Cup of Coffee in a Room

Thermal Entities:
- coffee, room

Thermal Interactions:
- conduction, convection

Behaviors:
- continuous + discrete
Example: A Cup of Coffee in A Room

Continuous behaviors

Discrete behaviors
Hybrid Behaviors Execution

Perform continuous behavior: as long as no discrete events possible.

- Time can advance
  - Execute Continuous Behaviors
  - Continuous Behaviors

- Time cannot advance
  - Execute Discrete Behaviors
  - Discrete Behaviors
Hybrid Behaviors Execution: Simulation

Euler method

Runge-Kutta method
Hybrid Behaviors Execution: 
*Fixed vs Adaptive Step Size*
Continuous Behaviors Execution

## Approximation

<table>
<thead>
<tr>
<th>Method</th>
<th>Fixed Step Size</th>
<th>Adaptive Step Size</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>✓</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Midpoint</td>
<td>✓</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Runge-Kutta 4th</td>
<td>✓</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Runge-Kutta-Fehlberg 4/5th</td>
<td>✓</td>
<td>✓</td>
<td>4 (or 5)</td>
</tr>
</tbody>
</table>

Based on approximation error.

Numerical
Execution Using the Euler Method

\[ y_{n+1} = hf(t_n, y_n) \]
Execution Using the Midpoint Method

\[ k_1 = f(t_n, y_n) \]
\[ k_2 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \]
\[ y_{n+1} = y_n + hk_2 \]
Discrete Behaviors Execution

Passive discrete events detection

Active discrete events detection
Outline

Modeling Framework

Execution Framework

Case Studies
Case 1: A Cup of Coffee in a Room

Using realistic physical parameters

Coffee and room are reaching the same temperature

Temperatures of Coffee & Room
Case 1: A Cup of Coffee in A Room

Continuous behaviors: exact vs approximations (Euler, RK2, RK4).

Temperatures of Coffee & Room

Approximation error
Case 1: A Cup of Coffee in a Room

Adaptive step size: RKF4/5

Step size in each time step is adjusted, based on error approximation.
Case 1: A Cup of Coffee in a Room

Comparing results: approximation error vs execution time (CPU time)
Case 2: Adding a Coffee Heater

The coffee has hybrid behaviors (phase change).

The heater gives a constant heat flow.

Use *find earliest*: how long to start melting, evaporating?

Temperatures of Coffee & Room
Case 2: Adding a Coffee Heater

Adaptive step size 1: RKF4/5, approx. error

Adaptive step size 2: discrete event detection

Temperatures of Coffee & Room

Adaptive Time Step Size 1

Adaptive Time Step Size 2
Case 2: Adding a Coffee Heater

Comparing results: the accuracy of discrete switch vs execution time
Case 3: Controlling the Warmth of the Coffee

A simple control (on/off): keep the coffee temperature between 70°C - 80°C.

Stability property: once the temperature has reached 70°C - 80°C, it will remain in that interval.
Case 3: Controlling the Warmth of the Coffee

Adaptive step size 2: discrete event detection.
Case 3: Controlling the Warmth of the Coffee

Use model LTL model checking:

\[
\square (\text{temp-ok} \rightarrow \square \text{temp-ok})
\]