SIMULATION OF HYBRID REO CONNECTORS

Ebrahim Ardeshir-Larijani,
Alireza Farhadi and Farhad Arbab
Ebrahim Ardeshir-Larijani

I am a senior postdoctoral researcher at the School of Computer Science of The Institute for Research in Fundamental Sciences (IPM). I received my PhD in Computer Science from University of Warwick. I am interested in applying formal methods to the software aspects of cyber physical systems.
PLAN OF THE TALK

- Introduction and Objectives
- Reo Coordination Language for Component Based Systems
- Specification of Hybrid Systems in Reo
- Transformation of Hybrid Reo specifications to UPPAAL SMC models
- Case Study: Strategy Based Control Systems
- Conclusion and Future works
**INTRODUCTION**

- CPSs consist of a large number of interactive/dynamic components (such as sensors, actuators, controllers, etc.). This leads to **Coordination** Problem.

- Other Challenges: **Design, Simulation, Verification** and **Security**.

- **Verification:** **Correctness by Construction** vs. **Model Checking/Theorem Proving/Abstract Interpretation** etc.
A SUITABLE FORMALISM
OBJECTIVES

- To model component-based and compositional hybrid connectors in a high level coordination language Reo.

- Defining composable new primitives for designing well-defined coordination layer in component based CPS software.

- Using Statistical Model Checking (SMC), to check properties of hybrid Reo connectors, through (numerical) simulation of their behaviour and the physical environment.
REO COORDINATION LANGUAGE

- Node Types (with respect to flow of data): Source, Sink and Mixed
Channel Types (relation/constraint on data flow): sync, lossy sync, drain, filter, ...
**REO Coordination Language**

- **Sync(a,b):** cyclically gets a data item through its end a and synchronously (atomically) outputs that data item through its end b.

- **Syncdrain(a,b):** cyclically accepts a data item on each of its ends synchronously and loses them. This kind of channel is used to synchronize independent flows of data as necessary.

- **FIFO(a,b):** synchronously accepts a data item, d, through its channel end a and stores it in its internal buffer, which can hold only a single data item. The channel then changes state and offers the data item in its buffer through its channel end b, which clears its buffer when b dispenses the data item.

- **Transformer<f>(a,b):** behaves like a sync channel, except that it applies the unary function f to every data item that it receives from a. The channel loses all data items taken from a that are not in the domain of the function f.
Hybridtimer\( \langle x' = f(x,y), q \rangle(a,b) \): is a specific variant of the transformer channel that expresses hybrid (continuous) behavior by a differential equation, represented by \( x' = f(x,y) \), which the next state variable \( x' \) is determined by the current state variable \( x \), the initial value \( y \) and the function \( f \). To take the initial value, we use a FIFO channel as a part of hybrid timer. The parameter \( q \) specifies a condition that when satisfied, the differentiation process will be reset.
HYBRID REO CONNECTORS

- Consider the model of a thermostat that coordinates a **user** and a **heater** component. The running of this simple model leads the heater "start" and "stop", depending on the temperature which varies according to ODEs.

- To express this model in hybrid Reo, we construct a connector out of ordinary Reo channels and two hybrid timer channels, one for heating up and another for cooling down the room temperature.
composedThermostat = (set, feedback, start, stop)
{sync(set, a); replicator(a, a1, a2);
 filter(x < down)(a1, b); filter(x ≥ down)(a2, e2);
 replicator(b, b1, b2); sync(b1, start);
 hybridtimer(x' == 5 - 0.1 * x, x ≤ up)(b2, e1); merger(e1, e2, e);
 sync(e, d); replicator(d, d1, d2);
 lossysync(d1, stop); hybridtimer(x' == -0.1 * x, x ≥ down)(d2, c);
 sync(c, feedback)}
Based on a hybrid variant of Constraint Automata [1], where transitions are labeled with data constraints, port names and reset conditions.

Example: expiring FIFO channel, where a data item is lost if it is not taken out from the buffer through the sink end within $\tau$ time units after it enters the source end.

UPPAAL SMC

- UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata.

- Modification of the classical model checking framework of UPPAAL, which the non-deterministic choices of time delays are subjected to probability distributions.

- The result of such an extension involves stochastic timed automata (STA) where, the interacting system’s components are modelled by a network of synchronizing STA’s.
Each type of hybrid Reo channels is represented as a template in UPPAAL SMC with ports as parameters.

The instantiation of templates involves defining UPPAAL broadcast channels corresponding to the connector ports.
UPPAAL TEMPLATES (I)

(a) Sync

(b) LossySync

(c) Merger

(d) Replicator
UPPAAL TEMPLATES (II)
<table>
<thead>
<tr>
<th>Rule</th>
<th>Textual Reo</th>
<th>Synchronization constraint</th>
<th>Data constraint</th>
<th>UPPAAL template</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>sync((in, out))</td>
<td>{\text{in, out}}</td>
<td>\text{inr \land outr}</td>
<td>Templates I(a)</td>
</tr>
<tr>
<td>(2)</td>
<td>hybridtimer([x' = f(x, y), q](in, out)]</td>
<td>{\text{in}} {\text{out}}</td>
<td>\begin{align*} x' &amp;= f(x, y) \land q \land \text{inr} \ -q \land \text{outr} \end{align*}</td>
<td>e.g., (x' = 5 - 0.1 \times x \land x &lt; \text{high}) Templates II(c)</td>
</tr>
<tr>
<td>(3)</td>
<td>filter([p](in, out)]</td>
<td>{\text{in, out}} {\text{in}}</td>
<td>\begin{align*} p \land \text{inr} \land \text{outr} \ -p \land \text{inr} \end{align*}</td>
<td>e.g., (x &lt; \text{down}) Templates II(a)</td>
</tr>
<tr>
<td>(4)</td>
<td>lossysync((in, out)]</td>
<td>{\text{in, out}} {\text{in}}</td>
<td>\begin{align*} \text{inr} \land \text{outr} \ \text{inr} \land -\text{outr} \end{align*}</td>
<td>Templates II(b)</td>
</tr>
<tr>
<td>(5)</td>
<td>transformer([y = f(x)](in, out)]</td>
<td>{\text{in, out}}</td>
<td>\begin{align*} y &amp;= f(x) \land \ \text{inr} \land \text{outr} \end{align*}</td>
<td>e.g., (\text{high} = \text{high+} \land \text{change[i]} \land \text{down} = \text{down+} \land \text{change[i]}) Templates II(e)</td>
</tr>
</tbody>
</table>
THERMOSTAT EXAMPLE

- In our example, the user could select from three ranges of temperature: [15 - 18], [18 - 20] and [21 - 24]. Here as soon as feedback is received by the user, it selects the next range and this process will be repeated indefinitely.

- To validate the behaviour of the thermostat, we visualize the trajectory of the clock $x$ (which represents room temperature) and the integer variables **high** and **down** (thresholds) in the red, blue and green, respectively.
Model composition in UPPAAL is done by a sequence of synchronizations (flattening), i.e., in a network of automata, multiple UPPAAL channels carry out single-channel actions. This is in contrast to Reo, where multi-actions are allowed.

We examine the relation of our transformation and the hybrid constraint automata (HCA) semantics of hybrid Reo connector, which also provides a method of proving the correctness of our transformation.
**Proposition:** Let $C_1 \ldots C_n$ be hybrid Reo components and $TS_{hca}$ be the transition graph of the multiplication of $C_i$’s HCAs. Let $TS_{upp}$ be the transition graph of $C_i$’s (here states defined by location vector, discrete and continuous evaluations $(l, v_d, v_c)$), that is obtained by our transformation, then $TS_{upp} \sim TS_{hca}$. 
The case study [2] consists of a building with several rooms and movable heaters, local thermostats, a central controller and a control strategy (non-functional properties).

Every room's temperature is monitored by a local thermostat, and the central controller can move a heater around the rooms and turn it on or off according to the control strategy.

Specifically, here we have two adjacent rooms and a heater that can be moved from one room to the another, according to the control strategy.

CASE STUDY: BEHAVIOUR

- Here we show only the synchronization constraints.

- State $s_1$ receives $\text{need}_2$ request from room2: if temperature difference between room2 and room1 is greater than $\text{diff}[2]$, then in state $s_2$ the connector takes the heater1 from room1 and moves it into room2 by synchronizing on requests $\text{move}_1$, $\text{release}_1$ and $\text{get}_2$ with heater1, room1 and room2 respectively.

- States $s_3$ and $s_4$ and their transitions are symmetric to $s_1$ and $s_2$ and their transitions.
CASE STUDY: REO CONNECTOR

- In the highlighted parts of the connector, there are three kinds of the sub-connector structures.
- Two of them surround the components rooms and heaters and,
- Third one is defined for adjacency relation between rooms.
CASE STUDY: REUSING SUB-CONNECTORS

- In our approach the control strategy, which is itself specified as a connector, is separated from the coordination sub-connector of the rooms and the heater.

- In case of changing in the building layout, the sub-connectors will be reused, while the other parts may be reconfigured as well.
IMPLEMENTATION

- A tool is developed that automatically compiles the specification of hybrid Reo connectors (in textual form) into UPPAAL SMC NHAs. To this end, the existing textual Reo compiler is extended to automatically generate an UPPAAL SMC network of HA for hybrid Reo connectors that coordinate a number of components whose behaviour are individually modelled by a UPPAAL HA.
Here $H[1]$ denotes the presence of heater in room 1 and $T[1]$ and $T[2]$ are temperatures of room 1 and room 2, respectively.

It can be seen that consistent with the control strategy the heater is moved from room 1 at a time close to 0.29 and moved back to the room around 0.58.
Suppose we want to check that the control strategy actually moves the heater around the rooms. We can do this by evaluating following probabilistic properties in UPPAAL SMC, which estimate the chance of the heater movement between the rooms:

1. $\Pr[<=100](<> \text{Room1.Empty and Room2.Full})$
2. $\Pr[<=100](<> \text{Room1.Full and Room2.Empty})$

The result is that both properties are satisfied with a high probability in $[0.902606; 1]$ with confidence $0.95$, in 36 simulation runs.
In this work we have presented:

A framework for **high level, reusable, composable** specification of coordination mechanisms in CPSs using Reo language.

A transformation that for a given hybrid Reo specification, produces an UPPAAL SMC network of hybrid automata.

Using **simulation, validation, and verification** capabilities of UPPAAL SMC tool support to validate and verify our case studies.
FUTURE WORKS

- Implementing a composition operator (e.g. HCA product) and apply various compilation optimization that are specific to Reo semantics. Currently composition is done by UPPAAL on a flattened model of NHA.

- Comparing with actor and event based modeling approaches for CPS, to understand where and why interaction based models such as Reo are effective in the design and analysis of CPSs.
REFERENCES


