Faithful automatic implementation of synchronous models on several computing architectures

Paul Caspi
Verimag-CNRS (emeritus)

TORRENTS Workshop November 2010
Faithful automatic implementation of synchronous models on several computing architectures

Paul Caspi
Verimag-CNRS (emeritus)

TORRENTS Workshop November 2010

a key issue in model-based design for critical control systems
Critical control systems

Flight Control

Emergency Shutdown

Speed Control

Full Automation
State of the Art

- automatic code generation
- architecture choice
- formal verification
- automatic import
- debugging
- simulation
- modelling
Some comments

- Only industrial tools are considered:
  - stands for Simulink/Stateflow
  - stands for SCADE

- Simulink also allows for automatic code generation but lacks qualification/certification
State of the Art

Some issues:

- Flat versus structured heterogeneity
- LTTA
- Faithfulness
Practitioners seem to prefer flat heterogeneity:
e.g.,

- Simulink/Stateflow encompasses seamlessly CT (ODEs), DE, SR (inherited sample times)

- SCADE V6 encompasses seamlessly SR (Lustre-like) and FSM (Esterel and mode automata) at any hierarchical level

This raises semantical issues
Semantics of Heterogeneity

- Seminal work of Lee & San Giovanni-Vincentelli
  Provides a solution
  But very general and thus not very effective

- Tagged Kahn Process Network (e.g. Caspi et al. HSCC09):
  Streams of type \((\text{Tag} \times \text{Value})\)
  Works seamlessly for KPN, SR, FSM, DE, and partially for CT (doesn’t handle true continuous time nor zero-crossing)

- Non standard analysis (Bliudze & Krob, Benveniste & al.)
  Not very convincing

Some hint: Cauchy Process Networks (CPN)
Cauchy Process Networks

- Some hints (Lee & al.)
  - Cantor metric for streams ⇒ Complete Metric Space
    Banach fix-point theorem
  - Cauchy-Lipschitz theorem for CT
- Both theorems use Cauchy sequences

Cauchy sequences:

\[ \forall \epsilon, \exists n, \forall n_1, n_2 \geq n, d(s_{n_1}, s_{n_2}) \leq \epsilon \]

Cauchy Process Network: where Cauchy sequences replace Kahn chains

Metric is not bound to decrease uniformly:
- allows for merging CT with KPN, SR, DE
- allows for backtracking (e.g., zero-crossing)
LT TA Origins

From analog to computerised control (early eighties)

Further evolution: TTA (hardware clock synchronisation)
LTTA Properties

- Interest:
  - Allows for seamless replacement of previous designs
  - Robustness: each computing unit has its own clock, power, memory...

- Problems:
  - bounded clock drifts
  - bounded communication delays
  - messages can be lost or duplicated

Departs from the synchronous model

How to faithfully implement synchronous designs on such architecture?
Continuous Computations

- **Uniform continuity**

\[ \forall \epsilon \exists \eta \forall t, t', |t - t'| \leq \eta \Rightarrow |x(t) - x(t')| \leq \epsilon \]

bounded communication delays $\Rightarrow$ bounded errors

- **Stability**

$H_\infty$ methods in control design
Voting

- Continuous computations $\Rightarrow$ bounded error vote

- Discrete computations

  Airbus solution (Caspi & Salem): bounded delay vote

  \[ \begin{array}{c}
  x_1 \\
  x_2 \\
  \text{v} \\
  \text{alarm}
  \end{array} \]

- Hybrid computations:

  (Caspi & Kossentini): mixed bounded error and delay vote
Handshakes

Based on logical synchronisation

- Synchronous systems can be represented as Marked Graphs
- In a synchronous unit, no further synchronisation is needed
- Implement handshakes at the boundaries between synchronous units

Yet a non robust solution
A Time-based Solution

(Caspi & Benveniste CDC 2008)

- The handshake method doesn’t take advantage of the LTTA clock properties
- A solution based on each computing unit spying the bus and knowing the drift margins of other units

Requires some slow-down but is robust
Faithfulness

In any case, even single-threaded computations depart from the synchronous hypothesis.

Proposed solutions

▶ Verify that the implementation still satisfies some property:

  e.g., Bertrane & Cousot use abstract interpretation for LTTA systems

▶ Correctness by construction:

  Check properties on the model and infer their satisfaction for the implementation

Designers seem to prefer the later solution
<table>
<thead>
<tr>
<th></th>
<th>Faithfulness</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>model</strong></td>
<td><img src="image1" alt="Faithfulness model diagram" /></td>
<td><img src="image2" alt="Simulation model diagram" /></td>
</tr>
<tr>
<td><strong>implementation</strong></td>
<td><img src="image3" alt="Faithfulness implementation diagram" /></td>
<td><img src="image4" alt="Simulation implementation diagram" /></td>
</tr>
</tbody>
</table>
Faithfulness vs. Simulation

Solution: choose an approximation distance

<table>
<thead>
<tr>
<th>model</th>
<th>Faithfulness</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>implementation</td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Approximation Metrics

Some proposals:

1. Henzinger & al. : Hausdorff metric between
   \(((t, s(t)) | t \in R^+)\) and \(((t, s'(t)) | t \in R^+)\)

2. Caspi & Benveniste : Skorokhod metric:
   \[
   d_S(s, s') = \inf_{\delta} \{ \sup_t |s(t) - s'(t - \delta)| + |\delta| \}
   \]

3. Caspi & Kossentini : Local \(L_1\) metric
   \[
   d_T(s, s') = \sup_t \int_t^{t+T} \frac{|s(t) - s'(t)|}{T} dt
   \]
Some Hint

Combinational boolean functions should be “continuous”

A small input variation should produce a small output one

Only Local $\mathcal{L}_1$ does the job
Papas & al., Girard: Approximate (bi)simulations

- Combines bisimulation with metrics. Could be used with any of the metrics proposed above

Still a lot of job needed
Perspectives

Faithfulness
tests
automatic code generation
architecture choice
formal verification
automatic import
debugging
simulation
modelling
more modelling frameworks
more formal tools
more architectures
more test methods
P. Caspi and A. Benveniste. 
Toward an approximation theory for computerised control.  
EMSOFT02, LNCS 2491, 2002.

P. Caspi and R. Salem.  
Threshold and bounded-delay voting in critical control systems.  

Ch. Kossentini and P. Caspi.  
Approximation, sampling and voting in hybrid computing systems.  
HSCC06, LNCS 3927, 2006.

P. Caspi and A. Benveniste. 
Time-robust discrete control over networked Loosely Time-Triggered Architectures. 
IEEE Decision and Control Conference, Cancun, December, 2008.

P. Caspi, A. Benveniste, R. Lublinerman, and S. Tripakis. 
Actors without directors: A Kahnian view of heterogeneous systems.  
HSCC09, LNCS 5469, 2009.

S. Tripakis, C. Pinello, A. Benveniste, A. San Giovanni-Vincentelli, P. Caspi, and M. di Natale.  
Implementing synchronous models on Loosely Time-Triggered Architectures.  
IEEE trans. on Computers, 57(10), 2008.