Model-Based Verification and Testing
Bruce H. Krogh

PhD students: Ajinkya Bhave
Stacey Ivol
Hitashyam Maka

Post doc: Dilsun Kaynar
Visiting professor: Goran Frehse

MURI Review Meeting
Frameworks and Tools for High-Confidence Design of Adaptive, Distributed Embedded Control Systems
Berkeley, CA
September 6, 2007

Model-Based Verification

Timing in Networked Control Systems
(Bhave, Krogh)
– impact on performance and stability of feedback loops
– analytical and simulation tools

Verification of Numerical Code
(Maka, Freshe, Krogh)
– verification for target environment
– reachability with polyhedral domains
– widening for iterative computations
Model-Based Testing Generation

Testing Environmental Assumptions
(Ivol, Kaynar, Krogh)
- new application of conformance testing
- initial implementation with timed i/o automata

Timing Analysis in Networked Control Systems

- **Timing Variations in Feedback Control Loops**
- **Analytical Tools**
- **Co-Simulation Tool**
- **Future Work**
Timing Variations in Feedback Control Loops

- **Sources**
  - single processor: interrupt routines, task scheduling, hardware (e.g., ADC)
  - multi-processor: buses, shared memory and DMA
  - physical interface: networked I/O*, sensors, actuators

- **Types**
  - jitter (sampling and input-output)*
  - skipped/lost packets (noise and data rate limits)
  - race conditions

- **Impact**
  - performance degradation*
  - loss of stability*
  - incorrect decisions

*Focus of first year work

Analytical Tools

- **Time delay robustness** is a large and extensively studied area with numerous theoretical results [Gu, K., Niculescu S.I]

- **Results categorized into**
  - delay dependent or delay independent stability criteria
  - frequency domain or time domain approaches

- **Majority of time domain results based on extensions of Lyapunov methods to infinite-dimensional systems**
  - Razumikhin method allows for bounded but arbitrarily time varying delays
  - Krasovskii method allows for delays bounded in both length and time variation

- **Most results are sufficient conditions**
  - may be excessively conservative or complex for practical application
**Frequency Domain Stability Analysis**

- Criteria for sampled-data, LTI control systems [Kao, Lincoln]
- Uses the Small Gain Theorem with delay modeled as uncertainty in the loop
- Current analysis done on SISO, stable, strictly proper continuous plant and discrete controller
- Extendable to MIMO via Integral Quadratic Constraint (IQC) approach with LMI [Kao, Rantzer]
- Delay is bounded with known worst-case bound but otherwise arbitrarily varying
- Stability is easily checked using closed-loop Bode plot

**Formula for Stability:**

\[
\frac{P_{alia}(\omega)C(\omega e^{j\omega})}{1 + P_{ZOH}(e^{j\omega})C(e^{j\omega})} < \frac{1}{N|\omega - 1|}, \quad \forall \omega \in [0, \infty]
\]

\[0 \leq \text{Delay } \Delta \leq Nh \text{ where } h \text{ is the sampling period}\]
Co-Simulation: TrueTime Simulink Blockset

- Co-simulation of
  - Control task execution
  - Network communication
  - Plant dynamics

- Investigate timing behavior of control loops implemented on digital computers

- System is subject to delays, jitter and lost samples

- Created at the Dept. of Automatic Control, Lund University

TrueTime – cont’d.

- A Kernel block, 3 Network blocks, 1 Battery block
  - Simulink S-functions written in C++
  - Event-based execution using zero-crossing functions
  - Portable to other simulation environments
STARMAC Example Scenario

- Attitude/Altitude controller connected to nonlinear plant through 10 Mbit Ethernet
- Motor thrust commands transmitted with jitter because of the random backoff network algorithm
- Pre/post processing of packets explicitly simulated with random delays at each node
- Each motor receives control signal at differing intervals which leads to deterioration in tracking
- Extreme jitter (order of plant dynamics) can cause instability of system

STARMAC Vehicle Model
Actuator Network

Sending Nodes with Jitter

Receiving Nodes

TrueTime Ethernet Block

STARMAC Step Response

Position in X,Y,Z

Velocity in X,Y,Z
STARMAC Response with Jitter

Jitter = Gaussian with zero mean and 0.001 var
Maximum delay 0.2s

STARMAC Response with Instability

Jitter = Gaussian with zero mean and 0.005 variance
Maximum delay 0.5s
References – Timing Analysis & Simulation


Verification of Numerical Code

- Numerical code verification
- Verification for target processors using polyhedra
- Widening for iterative computations
- Future work
Issues in Numerical Code

- numerical representations
  - integer, fixed point, floating point
- round-off error
- error accumulation
- divide-by-zero, overflow
- different results for different environments
  - compiler (optimizations)
  - OS (exception handling)
  - processor (word length, instruction set)

Verifying Numerical Code

- error models
  - interval arithmetic
  - ellipsoids
  - affine arithmetic
  - octagonal abstract domain (Cousot et al.)
- control-flow automaton (CFA)
  - states: precede program instructions (control locations)
  - edges: program guards (conditions)/actions (operations)
  - error model introduced in the actions
- verification (static analysis)
  - given initial ranges of input variables
  - propagate sets of variable values at each control location
  - check safety conditions
Verification for Target Processors

design model

code generation

target processor
compiler

disassembler

assembly code

CFA generator

target processor
error model

PHAVer

reachability
results

Advantages of using target .exe rather than source code:
- incorporates compiler optimizations
- focus on binary operations
Verification for Target Processors

Polyhedral reachability computations*
- full variable-space representation
- infinite precision implementation (PPL)
- currently for linear operations

Widening for Iterative Computations

For a lattice $\mathbb{L}$ with preorder relation $\sqsubseteq$ and join $\sqcup$, $\nabla : \mathbb{L} \times \mathbb{L} \to \mathbb{L}$ is a widening operator if:

i. For any $P, Q \in \mathbb{L}$, $P \sqcup Q \sqsubseteq P \nabla Q$. (overapproximation)

ii. For any increasing sequence $Q_1 \sqsubseteq Q_2 \sqsubseteq Q_3 \ldots$ the increasing sequence defined by $P_0 = Q_0$ and $P_{i+1} = P_I \nabla Q_{i+1}$ is not strictly increasing. (termination)

Accelerates convergence to a fixed point.

A New Widening Operator

For polyhedra $P, Q$ represented as linear inequalities with integer coefficients

$$P \nabla_{CL} Q = \text{coeflimit}(P \sqcup Q, k),$$

where for a polyhedron $R$, $\text{coeflimit}(R, k)$ is a polyhedron that contains $R$ with all coefficients with less than $k$ bits.

- performs better than standard widening when reachable sets are contracting
- this is often the case in iterative numerical computations

Example

Program 1
1: $x \in [4, 8]$
2: $y \in [1, 2]$
3: while true do
4:   $x = 0.25x + 0.25y$
5:   $y = 0.5y$
6: end while

Reachable Region of Prog. 1

Standard widening.

CL widening.
Rate-Limiter Application*

Program 4
1: \( Y := 0 \)
2: while true do
3: \( X := \text{floor}(X + D) \)
4: \( S := Y \)
5: \( Y := Y + S \)
6: if \( R \leq -D \) then
7: \( Y := Y - D \)
8: else if \( D \leq R \) then
9: \( Y := Y + D \)
10: end if
11: end if
12: end while

- \( Y \) tries to follow the value of \( X \) with the step size of \( D \).
- \( X \) and \( D \) change on each iteration
- Verify bounds on \( Y \)

Results:
- Using standard widening (Miné & Cousot) : \(|Y| < 144\)
- Using CL widening (Maka & Frehse) : \(|Y| < 128.046\)


Model-Based Test Generation

- Standard conformance testing (SCT)
- Testing environmental assumptions
- Mapping SCT to our problem
- Test generation for timed-automata
- Current implementation
- Next steps
Standard Conformance Testing

Specification

- System model
- Model of environment
- Expected outputs
- Equivalent?
- Test inputs
- Test outputs

Implementation

- Implemented system
- Real environment

Conformance Testing

Specification

- System model
- Model of environment
- Expected outputs
- Equivalent?
- Test inputs
- Test outputs

Implementation

- Implemented system
- Real environment

Assumes equivalent environments
Our Scenario

Model-based Development

control prog. model  →  environment model

formal verification
Our Scenario

If the design passes verification and code generation is correct, what needs to be tested?
Our Scenario

Model-based Development

- control prog. model
- environment model

Implementation

- control program
- other program
- platform
- command generator
- physical world
- outputs

Is the real environment “correct”?

Testing Environmental Assumptions

Model-based Development

Goal: Generate a test suite that is
- complete
- sound
- minimal

Implementation

- control prog. model
- environment model
- command generator
- physical world
- outputs

formal model
test suite
test commands
physical conditions
test outputs
Train Gate Example

Train-Gate Specification
- train sends approach signal
- controller sends lower command to gate
- train sends exit signal
- controller sends raise command

Safety Requirement
Gate must be down when a train is passing.

Train Gate Timed I/O Automata Model

compiled to target platform

Verifies correctly
For standard conformance testing:
System: controller, gate
Environment: train
Train Gate Timed I/O Automata Model

Possible vulnerability:
Controller accepts approach signal only from the idle state.

Environmental assumption:
Single train with at least a 60 second delay before next approach

We aim to
- identify critical environmental assumptions
- generate test suite automatically
Approach: Apply conformance testing to environmental assumptions

Conformance Testing for TIOA*

Henzinger, Manna, Pnueli time digitization (preserves soundness)

selected I/O reduces model scope

merging to eliminate unobservable transitions

Tool Development

- Specifications - UPPAAL TIOA
- XML interface to test generation program (C)
- Completed: digitization, test view hide/elide
- Currently debugging test suite generation

Model-Based Verification and Testing

- Next Steps -

Timing in Networked Control Systems
- extensions to distributed systems
- develop analytical/simulation toolbox

Verification of Numerical Code
- extensions to nonlinear computations
- verification of Simulink/Stateflow designs

Testing Environmental Assumptions
- test suite reduction
- extensions to richer dynamics