Model-Based Verification and Testing

Bruce H. Krogh

***PhD students*:**
Ajinkya Bhave
Stacey Ivol
Hitashyam Maka

***Post doc*:**
Dilsun Kaynar

***Visiting professor*:**
Goran Frehse

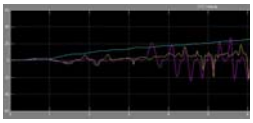
MURI Review Meeting
Frameworks and Tools for High-Confidence Design of Adaptive, Distributed Embedded Control Systems
Berkeley, CA
September 6, 2007

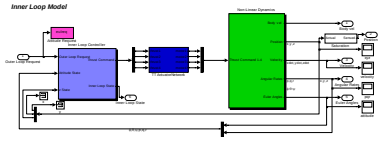
Model-Based Verification

Timing in Networked Control Systems

(Bhave, Krogh)

- impact on performance and stability of feedback loops
- analytical and simulation tools



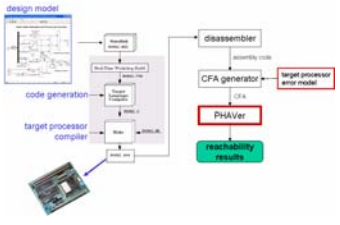


$$\left| \frac{P_{\text{alias}}(\omega)C(e^{j\omega})}{1 + P_{\text{ZOH}}(e^{j\omega})C(e^{j\omega})} \right| < \frac{1}{N|e^{j\omega} - 1|},$$

Verification of Numerical Code

(Maka, Freshe, Krogh)

- verification for target environment
- reachability with polyhedral domains
- widening for iterative computations

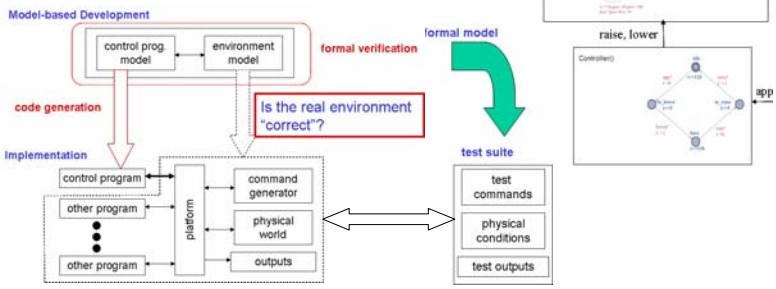


Model-Based Testing Generation

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

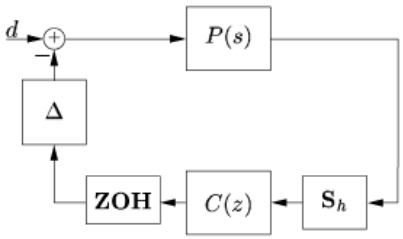
Frequency Domain Stability Analysis

- 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

Frequency Domain Stability Analysis

Formula for Stability:

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



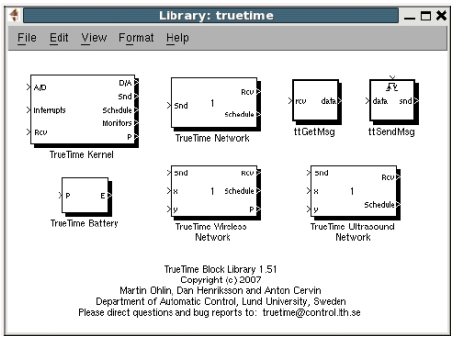
0 <= Delay Δ <= Nh where h is the sampling period

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.

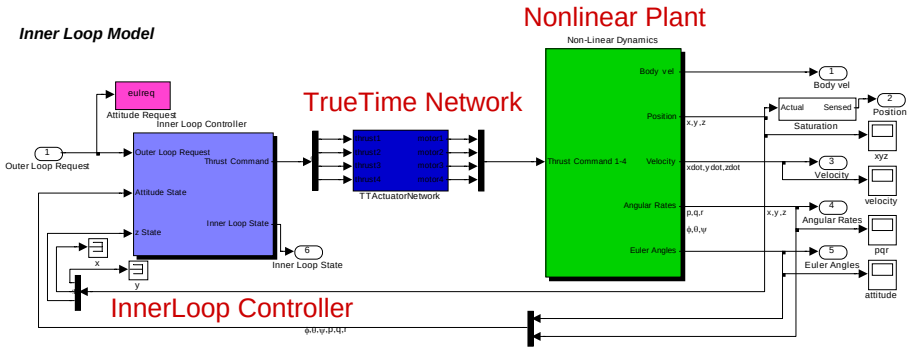
- **A Kernel block, 3 Network blocks, 1 Battery block**
  - Simulink S-functions written in C++
  - Event-based execution using zero-crossing functions
  - Portable to other simulation environments



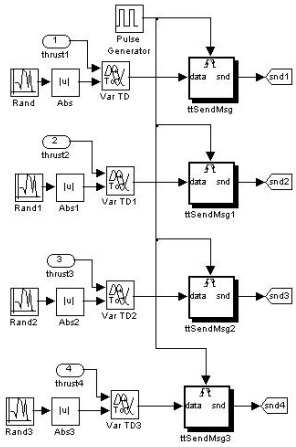
STARMAC Example Scenario

- 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

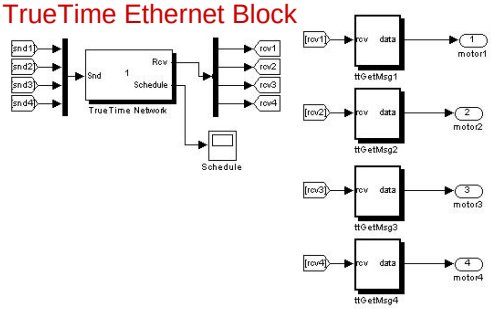
STARMAC Vehicle Model



Actuator Network



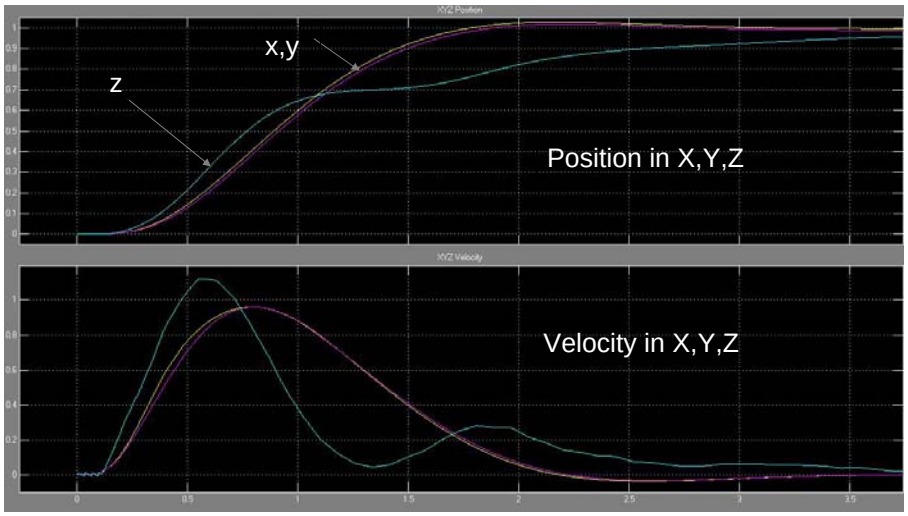
TrueTime Ethernet Block



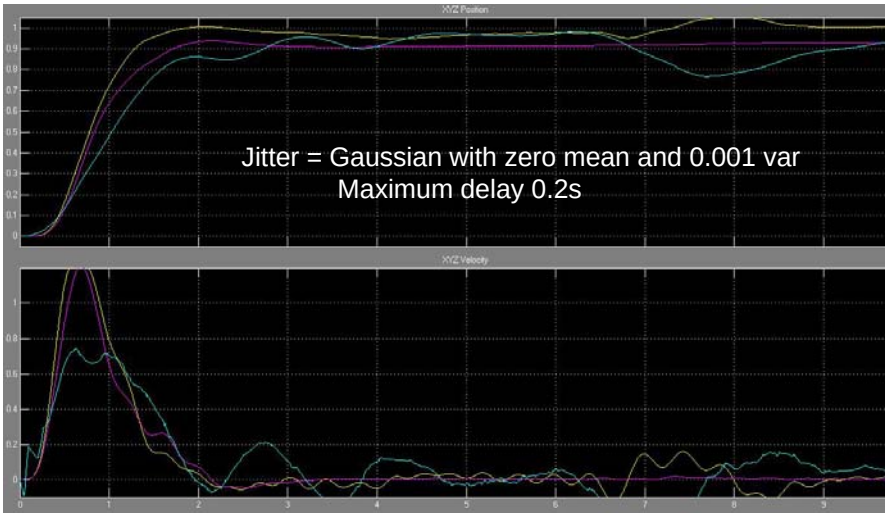
Receiving Nodes

Sending Nodes with Jitter

STARMAC Step Response



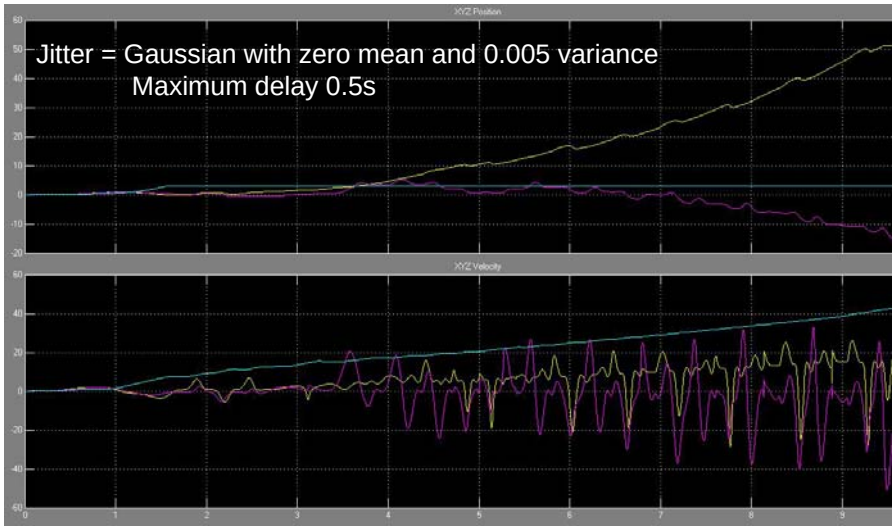
STARMAC Response with Jitter



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

15

STARMAC Response with Instability



Jitter = Gaussian with zero mean and 0.005 variance
Maximum delay 0.5s

16

References – Timing Analysis & Simulation

- Lincoln, B. (2002), “A simple stability criterion for control systems with varying delays”, In Proceedings of the 15th IFAC World Congress, Barcelona.
- Kao, C.-Y., & Lincoln, B. (2004), “Simple stability criteria for systems with time-varying delays”, In Automatica 40 (2004), 1429-1434.
- Kao, C.-Y., & Rantzer, A. (2003). Stability criteria for systems with bounded uncertain time-varying delays. In Proceedings of the 2003 European Control Conference. Cambridge, UK.
- Gu, K., Niculescu S.I., “Survey on recent results in the stability and control of time-delay systems”, Journal of Dynamic Systems, Measurement, and Control, vol. 125, pp. 158- 165, June 2003.
- Cervin, A., “How does control timing affect performance”, IEEE Contr. Syst Mag., vol. 2, no. 2, pp. 16–30, June 2003.
- K. Gu, V. L. Kharitonov, and J. Chen, “Stability of Time-Delay Systems”, 1st ed. Boston, MA: Birkhauser, 2003

Verification of Numerical Code

- **Numerical code verification**
- **Verification for target processors using polyhedra**
- **Widening for iterative computations**
- **Future work**

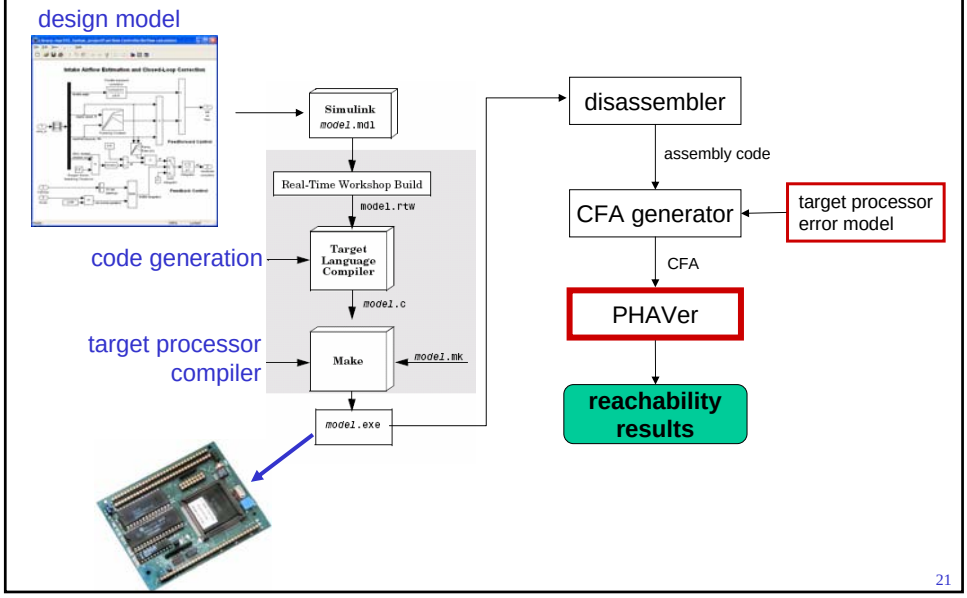
Issues in Numerical Code

- **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)

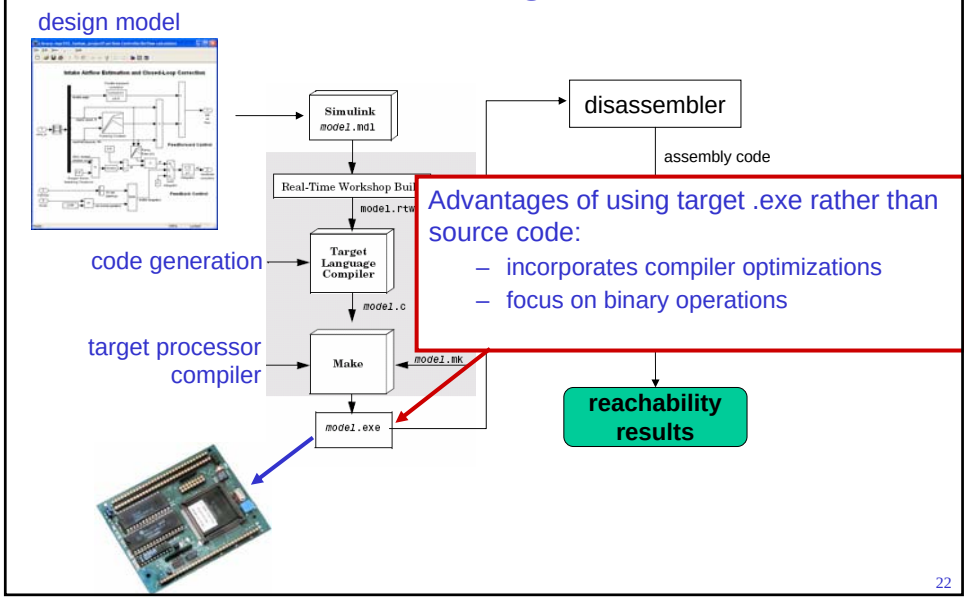
Verifying Numerical Code

- **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

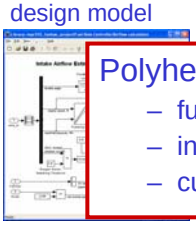


Verification for Target Processors



Verification for Target Processors

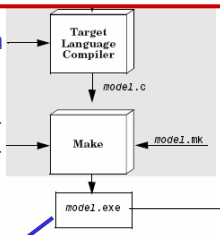
design model



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

code generation

target processor compiler



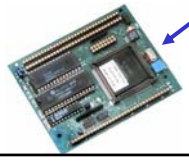
code

target processor error model

CFA

PHAVer

reachability results



* G. Frehse, PHAVer: Algorithmic Verification of Hybrid Systems past HyTech, *Software Tools for Technology Transfer*, to appear

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 stricly increasing. (termination)

Accelerates convergence to a fixed point.

P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pp. 238-252, New York, 1977. ACM Press.

A New Widening Operator

For polyhedra P,Q represented as linear inequalities with integer coeffcients

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

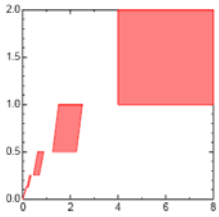
where for a polyhedron R, $coeflimit(R, k)$ is a polyhedron that contains R with all coefficients with less than k bits.

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

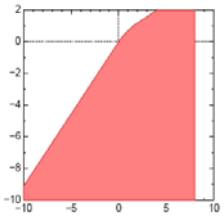
Example

Program 1

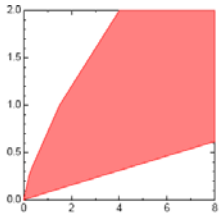
```
1: x ∈ [4, 8]
2: y ∈ [1, 2]
3: while true do
4:     x = 0.25x + 0.25y
5:     y = 0.5y
6: end while
```



Reachable Region of Prog.1



Standard widening.



CL widening.

Rate-Limiter Application*

```
Program 4
1: Y ← 0
2: while true do
3:    X ← [−128, 128]
4:    D ← [0, 16]
5:    S ← Y
6:    R ← X − S
7:    Y ← X
8:    if R ≤ −D then
9:       Y ← S − D
10:   else if D ≤ R then
11:      Y ← S + D
12:   end if
13: 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

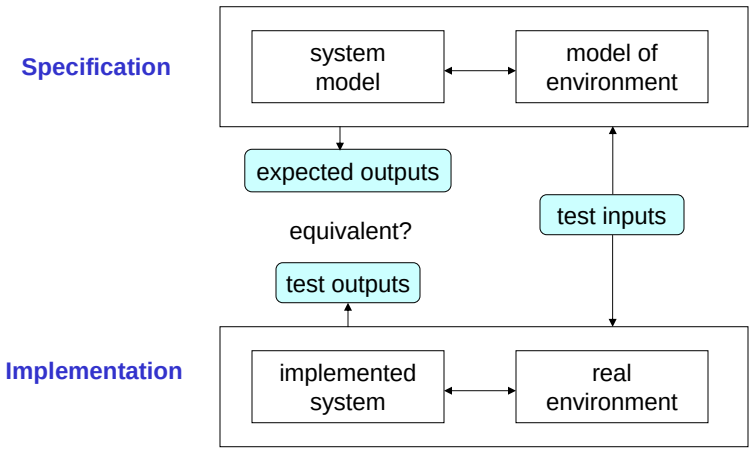
* Antoine Miné. Relational abstract domains for the detection of floating-point run-time errors. In David A. Schmidt, editor, ESOP, volume 2986 of Lecture Notes in Computer Science, pages 3–17. Springer, 2004.

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**

CarnegieMellon

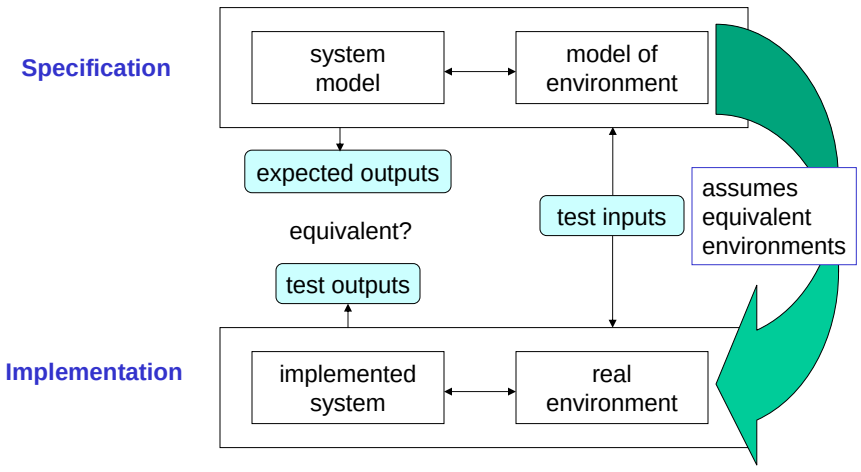
Standard Conformance Testing



29

CarnegieMellon

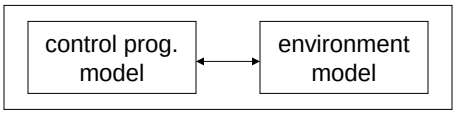
Conformance Testing



30

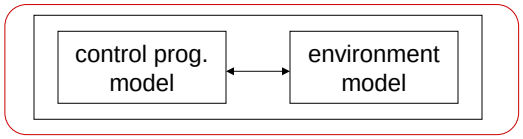
Our Scenario

Model-based Development



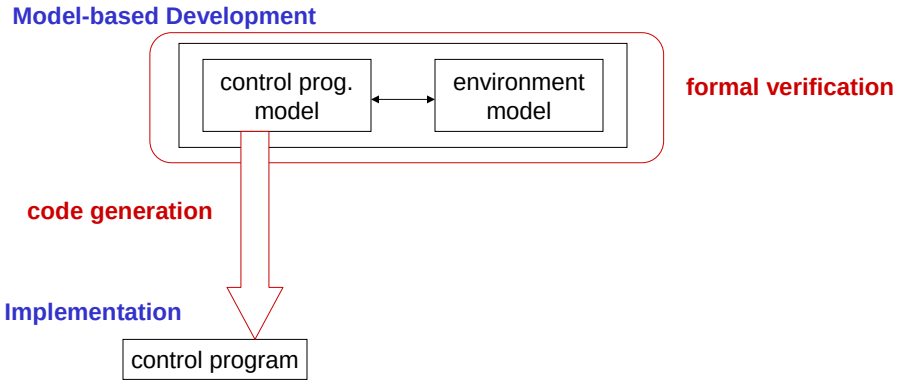
Our Scenario

Model-based Development

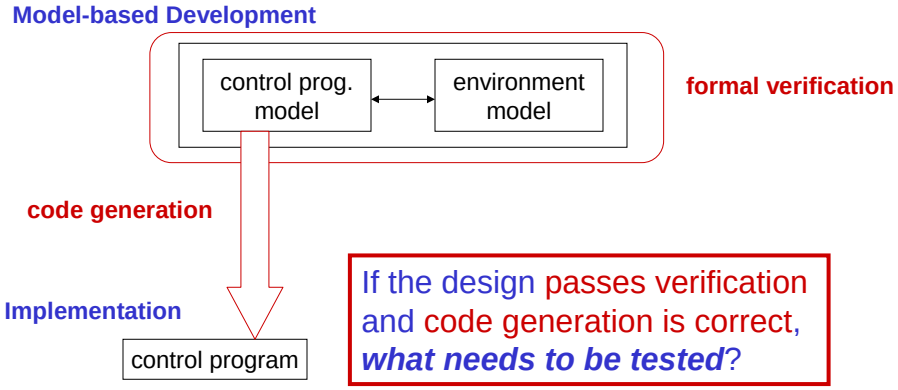


formal verification

Our Scenario



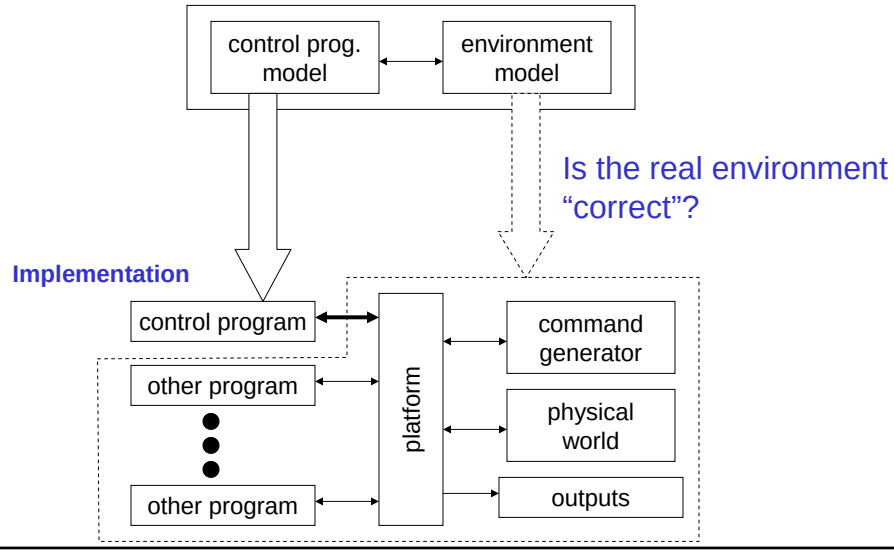
Our Scenario



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

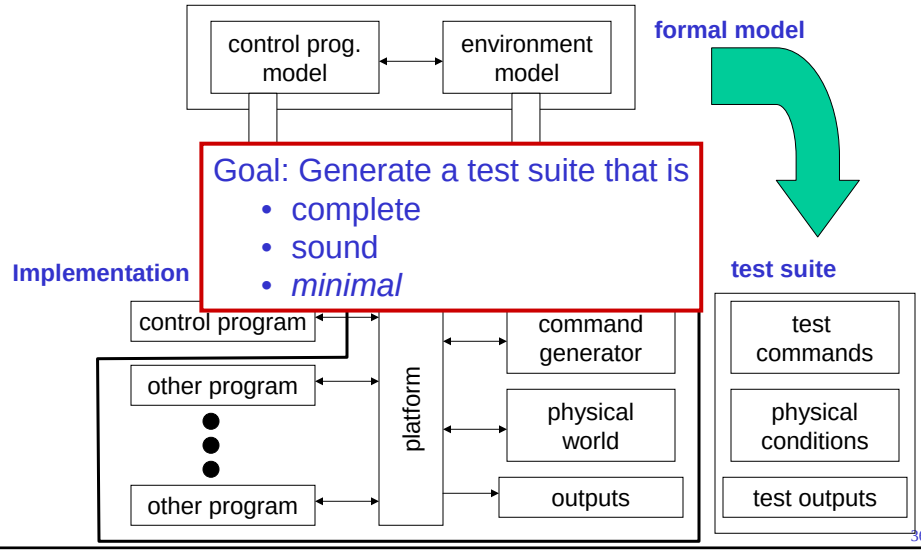
Our Scenario

Model-based Development

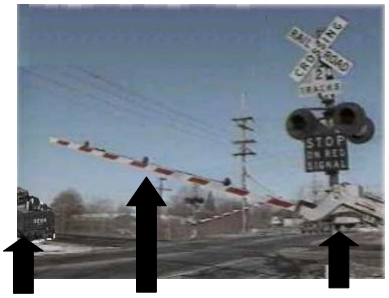


Testing Environmental Assumptions

Model-based Development



Train Gate Example



Train-Gate Specification

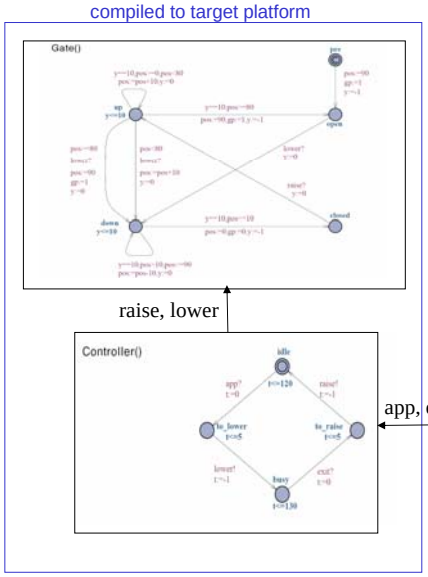
- train sends *approach* signal
- controller sends *lower* command to gate
- train sends *exit* signal
- controller sends *raise* command

Safety Requirement

Gate must be down when a train is passing.

Train Gate Timed I/O Automata Model

compiled to target platform



Verifies correctly

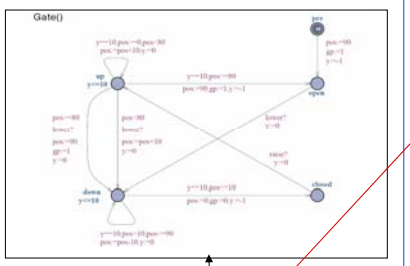
For standard conformance testing:

System: controller, gate

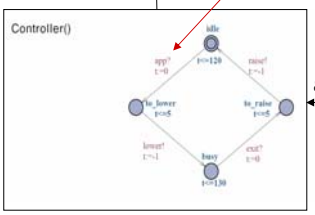
Environment: train

Train Gate Timed I/O Automata Model

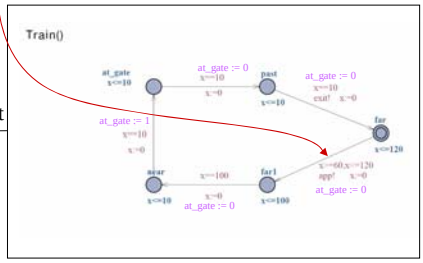
compiled to target platform



raise, lower



app, exit



Possible vulnerability:

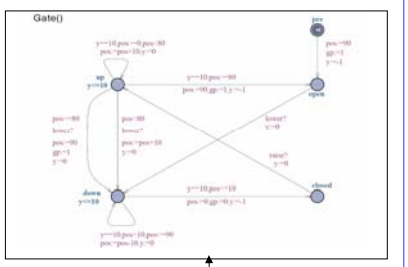
Controller accepts approach signal only from the idle state.

Environmental assumption:

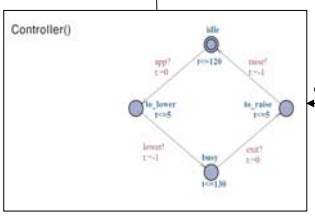
Single train with at least a 60 second delay before next approach

Train Gate Timed I/O Automata Model

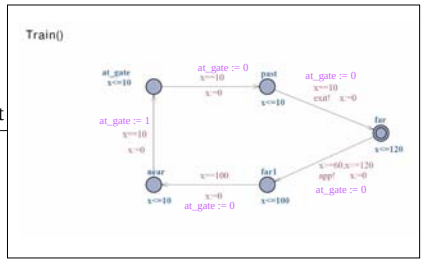
compiled to target platform



raise, lower



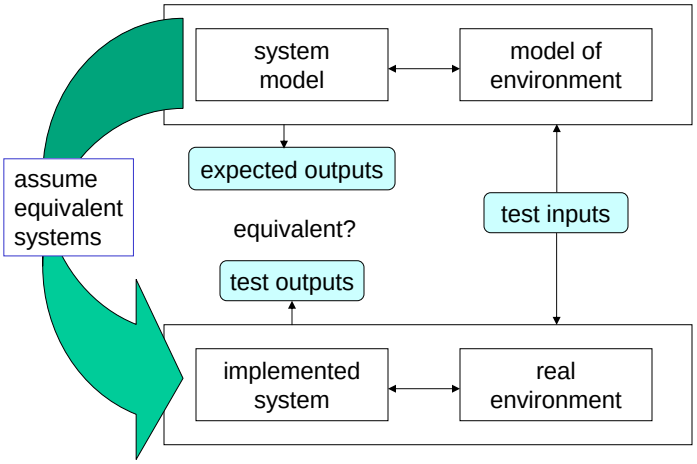
app, exit



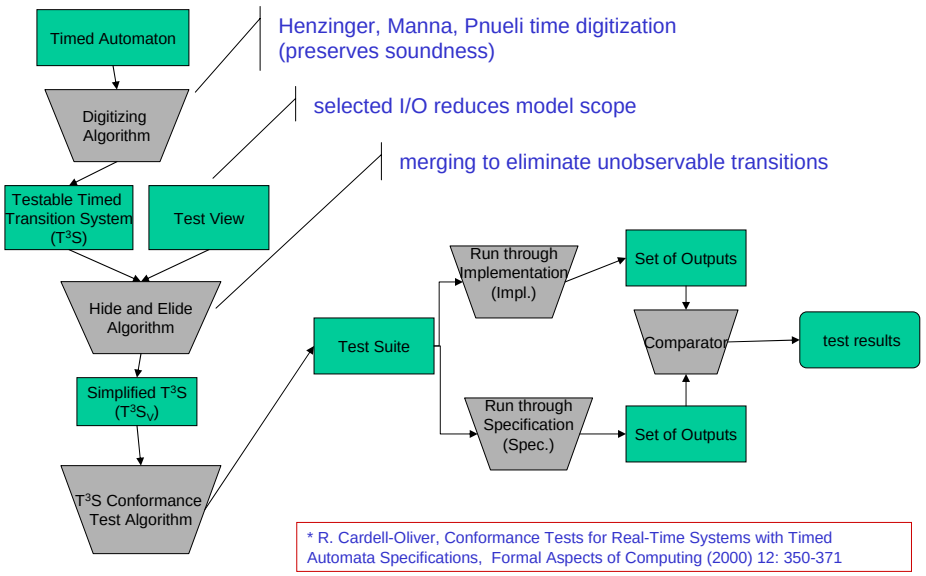
We aim to

- identify critical environmental assumptions
- generate test suite automatically

Approach: Apply conformance testing to environmental assumptions



Conformance Testing for TIOA*



* R. Cardell-Oliver, Conformance Tests for Real-Time Systems with Timed Automata Specifications, Formal Aspects of Computing (2000) 12: 350-371

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