ANALYZING REAL TIME LINEAR CONTROL SYSTEMS USING SOFTWARE VERIFICATION

Parasara Sridhar Duggirala – UConn
Mahesh Viswanathan – UIUC
Real-Time Systems
+ Linear Control Systems
+ Verification

This paper.
Isn’t That Hybrid Systems Verification?

- Yes and No.

Typical control system
Isn’t That Hybrid Systems Verification?

- Yes and No.

Typical hybrid system
Isn’t That Hybrid Systems Verification?

- Yes and No.

Typical hybrid system

Physical Plant

\[
\begin{align*}
C_1 \\
C_2 \\
\vdots \\
C_n
\end{align*}
\]

Logic

Hybrid Automata

\[
\begin{align*}
\dot{x} &= f_1(x) \\
\dot{x} &= f_2(x) \\
\dot{x} &= f_3(x)
\end{align*}
\]

Assumptions:
1. Continuous feedback
2. Exact computations
Isn’t That Hybrid Systems Verification?

- Technically Yes, practically No.

Hybrid Automata

VS

Plant
+ Noisy environment

Floating points,
Data structures, ...

Scheduling, ...

Hardware, ...

Approx. model, ...

main(){
if (...) then
else ...
}
Closely Related Works

1. Fluctuat, Martinez et.al. [Floating Points]
2. Sahvy, HybridFluctuat – periodic actuation.
3. Frehse et.al. [Scheduling]

Floating points, Data structures, ...

Scheduling, ...

Hardware, ...

Plant

+ Noisy environment

Approx. model, ...
Closely Related Works

1. Fluctuat, Martinez et.al. [Floating Points]
2. Sahvy, HybridFluctuat – periodic actuation.
3. Frehse et.al. [Scheduling]

This paper:
Verification (at discrete instances) while taking into account the computation time of software and scheduling of RTOS.
This Paper; Briefly

State of plant $x$ evolves as $\dot{x} = Ax + Bu$

Verification that takes all the three aspects into account

```c
main()
{
    .......
    if (...) then
    ...
    else ...
}
```
Outline

- Introduction
- Computational model
- Drawbacks of existing techniques (or advantages?)
- Software verification inspired technique
  - Analyzing linear control systems
  - Accounting for timing analysis
- Software verification techniques used
- Results
- Discussion and Future work
Computational Model

1. Control program is a task on RTOS (periodically scheduled).
2. Delay between sensing and actuation (computation time).
3. Control program may or may not make the deadline.
1. Control program is a task on RTOS (periodically scheduled).
2. Delay between sensing and actuation (computation time).
3. Control program may or may not make the deadline.

1. Control program is run every T time units.
2. It may/may not make the deadline (TWCRT).
3. If it makes the deadline, results of computation are given as actuation parameters.
4. If it does not make the deadline, computation results are thrown away.
Motivating Example
Leader-Follower System

velocity = $v$; acceleration = $a$;

follower

velocity = $v_f$; acceleration = 0;

leader

Dynamics of the system
\[ \dot{s} = v_f - v; \]
\[ \dot{v} = a - k_{aero} v; \]
\[ \dot{a} = u; \]

$k_{aero}$ is the air-drag Control Law
\[ u = -2a - 2(v - v_f) \]
Motivating Example
Leader-Follower System

Controller operates at 100Hz frequency.
(computation time = 0).

Hybrid systems model:

1. Add continuous variables $u, t$
2. Update $u$ every 0.01 sec.
3. Reset $t$ every 0.01 sec.

Dynamics of the system
\[
\begin{align*}
\dot{s} &= v_f - v; \\
\dot{v} &= a - k_{aero} v; \\
\dot{a} &= u;
\end{align*}
\]

$k_{aero}$ is the air-drag

Control Law
\[
u = -2a - 2(v - v_f)\]
Motivating Example
Leader-Follower System

- Controller operates at 100Hz frequency. (computation time = 0).
- Hybrid systems model:
  1. Add continuous variables $u, t$
  2. Update $u$ every 0.01 sec.
  3. Reset $t$ every 0.01 sec.

Dynamics of the system

\begin{align*}
\dot{s} &= v_f - v; \\
\dot{v} &= a - k_{aero} v; \\
\dot{a} &= u;
\end{align*}

$k_{aero}$ is the air-drag

Control Law

\[ u = -2a - 2(v - v_f) \]
Naïve Hybrid Systems Verification With SpaceEx

Property: If $v_f = 60, v_0 \in [59,61], s_0 = 100$

then always $v \leq 62 \land s \geq 50$
Naïve Hybrid Systems Verification
With SpaceEx

Property: If $v_f = 60$, $v_0 \in [59, 61]$, $s_0 = 100$ then always $v \leq 62 \land s \geq 50$

Property cannot be inferred! Overapproximation is too high
Why It Does Not Work
(And Why It Should Not)

- Two sources of overapproximation
  1. Discrete transitions.
  2. Mismatch between the actuated values and sensed values.
     If $v \in [59, 61], u \in [-2, 2]$ but $u > 0$ if and only if $v < 60$.
     `SpaceEx` algorithm does conservative estimate.
Why It Does Not Work
(And Why It Should Not)

- Two sources of overapproximation
  1. Discrete transitions.
  2. Mismatch between the actuated values and sensed values.
     \[ v \in [59, 61], \quad u \in [-2, 2] \quad \text{but} \quad u > 0 \quad \text{if and only if} \quad v < 60. \]
     SpaceEx algorithm does conservative estimate.

- Why it should not? (#myPerspective)
  - Hybrid Systems verification tools are supposed to find the flaws at the design level.
  - Ensuring lower level details are “coherent” with higher level design should be the job of system developer (or a different verification tool?).
  - Problem: But many bugs happen during the implementation!
Outline

- Motivation
- Computational model
- Drawbacks of existing techniques (or advantages?)
  - Software verification inspired technique
    - Analyzing linear control systems
    - Accounting for timing analysis
- Software verification techniques used
- Results
- Discussion and Future work
Generated code simulates the closed loop system by tracking the software state and physical state of the plant.
Software Verification Inspired Technique: Outline

Code Piece 1

Code Piece 2

Physical Plant

Software Verification Tools
Part 1 – Analyzing Linear Control System

- Linear ODE for plant $\dot{x} = Ax + Bu$.
- Closed form expression for the behavior
  
  $$e^{At}x(0) + \int_{0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau.$$
Part 1 – Analyzing Linear Control System

- Linear ODE for plant $\dot{x} = Ax + Bu$.

- Closed form expression for the behavior
  
  $$e^{At}x(0) + \int_{0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau.$$ 

- Observation: $u(t)$ is constant for a given time period (T).
  
  $$x(T) = e^{AT}x(0) + G(A,T)Bu$$

- Since $T, A$ are known, $x(T)$ can be computed as a func. of $x(0)$. 
Part 1 – Analyzing Linear Control System

■ Linear ODE for plant \( \dot{x} = Ax + Bu. \)

■ Closed form expression for the behavior

\[
e^{At} x(0) + \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau.
\]

■ Observation: \( u(t) \) is constant for a given time period (T).

\[
x(T) = e^{AT} x(0) + G(A, T)Bu
\]

■ Since \( T, A \) are known, \( x(T) \) can be computed as a func. of \( x(0) \).

■ For leader trailer system – at discrete time units.

\[
s_n = s - 0.0995 \times (v - v_f) - 0.005 \times a - 0.002 \times u; \quad \text{Note: Relation between}
\]
\[
v_n = v_f + 0.99 \times (v - v_f) + 0.0995 \times a + 0.005 \times u; \quad u \text{ and } s_n, v_n, a_n \text{ is symbolic.}
\]
\[
a_n = a + 0.1 \times u;
\]

\[
\]
Part 1 – Analyzing Linear Control System

■ What about with the control law?

\[ u = -2a - 2(v - v_f); \]
\[ s_n = s - 0.0995 \times (v - v_f) - 0.005 \times a - 0.002 \times u; \]
\[ v_n = v_f + 0.99 \times (v - v_f) + 0.0995 \times a + 0.005 \times u; \]
\[ a_n = a + 0.1 \times u; \]

**Note:** \( u > 0 \) initially if and only if \( v < v_f \).
Part 1 – Analyzing Linear Control System

What about with the control law?

\[ u = -2a - 2(v - v_f); \]
\[ s_n = s - 0.0995(v - v_f) - 0.005a - 0.002u; \]
\[ v_n = v_f + 0.99(v - v_f) + 0.0995a + 0.005u; \]
\[ a_n = a + 0.1u; \]

**Note:** \( u > 0 \) initially if and only if \( v < v_f \).

Skipping details: Error analysis and soundness proof.
Part 2 – Handling the Timing Analysis and Scheduling

- Scheduling: fixed time period for control task.
- Timing behavior: Typical Worst Case Analysis.
  1. WCET might be too conservative.
  2. TWCA generalizes WCET.

What is Typical Worst Case Analysis?
Deadline is Typical Worst Case Response Time (TWCRT) – W.
  1. Task can miss a deadline “sometimes”.
  2. Number of deadline misses in the past “n” schedules is bounded.
Part 2 – Handling the Timing Analysis and Scheduling

Example:

<table>
<thead>
<tr>
<th># deadline misses</th>
<th>consecutive executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
Part 2 – Handling the Timing Analysis and Scheduling

Example:

<table>
<thead>
<tr>
<th># deadline misses</th>
<th>consecutive executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

\(d_i\) tracks whether the deadline is missed or met in the \(i^{th}\) last scheduling. Nondeterministic choice of deadline miss by \textit{Assume} statement.
**Part 2 – Handling the Timing Analysis and Scheduling**

- **Example:**

<table>
<thead>
<tr>
<th># deadline misses</th>
<th>consecutive executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

\( d_i \) tracks whether the deadline is missed or met in the \( i^{th} \) last scheduling. Nondeterministic choice of deadline miss by \textbf{Assume} statement.

```c
Code Piece 2
```

\[
\begin{align*}
    d_5 &= d_4; \\
    d_4 &= d_3; \\
    d_3 &= d_2; \\
    d_2 &= d_1; \\
    \text{deadline\_met} &= 0; \quad \text{// assume deadline miss} \\
    \text{Assume}(d_1 == 0 \text{ || } d_1 == 1); \\
    \text{if}((d_1 == 1) \text{ && } ((d_1 + d_2 + d_3 + d_4 + d_5 > 2) \\
    \text{ || } (d_1 + d_2 + d_3 > 1))) \text{ then} \\
    \text{d_1} &= 0; \quad \text{// according to TWCA} \\
    \text{endif}; \\
    \text{if}(d_1 == 0) \text{ then} \text{deadline\_met} &= 1; \quad \text{// deadline met} \\
    \text{endif};
\end{align*}
```
Bringing These Two Together

Code Piece 1

+  =

Code Piece 2
Bringing These Two Together

Code Piece 1

Code Piece 2

\[ u = -2a_s -2(v_s - vf_s); \]

\[ d_5 = d_4; d_4 = d_3; d_3 = d_2; d_2 = d_1; \]

\text{headline\_met} = 0; // assume headline miss

\text{Assume}(d_1 == 0 \| d_1 == 1);

\text{if}((d_1 == 1) \&\& ((d_1 + d_2 + d_3 + d_4 + d_5 > 2) \| (d_1 + d_2 + d_3 > 1))) \text{then}

\hspace{1cm} d_1 = 0; // according to TWCA

\text{endif};

\text{if}(d_1 == 0) \text{then} \text{headline\_met} = 1; // deadline met

\text{endif};

\text{// Update actuation parameters if deadline is met}

\text{if}(\text{headline\_met} == 1)

\hspace{1cm} u_a = u;

\text{endif};

\[ s_n = s - 0.0995(v-vf) -0.005a - 0.0002u_a; \]

\[ v_n = vf + 0.99(v-vf) + 0.0995a + 0.005u_a; \]

\[ a_n = a + 0.1u_a; \]
Bringing These Two Together

\[
\begin{align*}
u &= -2a_s -2(v_s - vf_s); \\
d_5 &= d_4; d_4 = d_3; d_3 = d_2; d_2 = d_1; \\
\text{deadline}_\text{met} &= 0; \quad // \text{assume deadline miss} \\
\text{Assume}(d_1 = 0 \ || \ d_1 = 1); \\
\text{if}((d_1 = 1) \ && ((d_1 + d_2 + d_3 + d_4 + d_5 > 2) \\
\quad \ || \ (d_1 + d_2 + d_3 > 1))) \quad \text{then} \\
\quad d_1 &= 0; \quad // \text{according to TWCA} \\
\text{endif;}
\end{align*}
\]

\[
\begin{align*}
\text{if}(d_1 = 0) \quad \text{then} \quad \text{deadline}_\text{met} &= 1; \quad // \text{deadline met} \\
\text{endif;}
\end{align*}
\]

// Update actuation parameters if deadline is met
\[
\begin{align*}
\text{if}(\text{deadline}_\text{met} = 1) \\
\quad u_a &= u; \\
\text{endif;}
\end{align*}
\]

\[
\begin{align*}
s_n &= s - 0.0995(v-vf) - 0.005a - 0.0002u_a; \\
v_n &= vf + 0.99(v-vf) + 0.0995a + 0.005u_a; \\
a_n &= a + 0.1u_a;
\end{align*}
\]
Bringing These Two Together

Controller code

\[ u = -2a_s -2(v_s - vf_s); \]

\begin{verbatim}
Controller code
Code Piece 1
\end{verbatim}

\begin{verbatim}
Code Piece 2
\end{verbatim}

\begin{verbatim}
// Update actuation parameters if deadline is met
if (deadline_met == 1)
u_a = u;
endif;
\end{verbatim}

Timing Behavior

\begin{verbatim}
d_5 = d_4; d_4 = d_3; d_3 = d_2; d_2 = d_1;
deadline_met = 0; // assume deadline miss
Assume(d_1 == 0 || d_1 == 1);
if((d_1 == 1) && ((d_1 + d_2 + d_3 + d_4 + d_5 > 2)
 || (d_1 + d_2 + d_3 > 1))) then
d_1 = 0; // according to TWCA
endif;
if(d_1 == 0) then deadline_met = 1; // deadline met
endif;
\end{verbatim}

Updating actuation only when deadline is met

\begin{verbatim}
Update actuation parameters if deadline is met
if (deadline_met == 1)
u_a = u;
endif;
\end{verbatim}

Plant behavior

\begin{verbatim}
Plant behavior
s_n = s - 0.0995(v-vf) - 0.005*a - 0.0002*u_a;
v_n = vf + 0.99*(v-vf) + 0.0995*a + 0.005*u_a;
a_n = a + 0.1*u_a;
\end{verbatim}
Verifying Safety Of Software For Bounded/Unbounded Time

1. Abstract Interpretation
   - Widely used in checking properties of embedded software.
   - Various abstract domains/analysis techniques.
   - Interproc abstract interpretation tool.

2. Bounded Model Checking using SMT solvers
   - Popular approach because of recent advancements.
   - Very efficient solvers for linear arithmetic (Simplex + SAT).
   - Z3 SMT solver.
## Results – Part 1

### Z3 vs AI vs SpaceEx

<table>
<thead>
<tr>
<th>Problem</th>
<th>Steps</th>
<th>Z3</th>
<th>Interproc</th>
<th>SpaceEx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Box</td>
<td>Oct</td>
</tr>
<tr>
<td>ACC1</td>
<td>25</td>
<td>P</td>
<td>25.8 s</td>
<td>F, 0.2 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACC2</td>
<td>25</td>
<td>P</td>
<td>25.9 s</td>
<td>P, 0.2 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kin1</td>
<td>25</td>
<td>P</td>
<td>5.8 s</td>
<td>F, 0.05 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kin2</td>
<td>25</td>
<td>P</td>
<td>5.8 s</td>
<td>P, 0.05 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Results – Part 1
### Z3 vs AI vs SpaceEx

<table>
<thead>
<tr>
<th>Problem</th>
<th>Steps</th>
<th>Z3</th>
<th>Interproc</th>
<th>SpaceEx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Box</td>
<td>Oct</td>
</tr>
<tr>
<td>ACC1</td>
<td>25</td>
<td>P</td>
<td>25.8 s</td>
<td>F</td>
</tr>
<tr>
<td>ACC2</td>
<td>25</td>
<td>P</td>
<td>25.9 s</td>
<td>P</td>
</tr>
<tr>
<td>Kin1</td>
<td>25</td>
<td>P</td>
<td>5.8 s</td>
<td>F</td>
</tr>
<tr>
<td>Kin2</td>
<td>25</td>
<td>P</td>
<td>5.8 s</td>
<td>P</td>
</tr>
</tbody>
</table>

Inferences:
1. Proving a property using Interproc and SpaceEx requires choosing appropriate domain.
2. Trivial – verification time depends on the domain chosen.
3. Bounded model checking seems to be fast and give precise verification results.
## Results – Part 2
### Evaluation with Z3

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Dimn.</th>
<th>Steps</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTSC</td>
<td>4</td>
<td>15</td>
<td>12.6 s</td>
</tr>
<tr>
<td>MTSC</td>
<td>4</td>
<td>20</td>
<td>1m 14 s</td>
</tr>
<tr>
<td>MTSC</td>
<td>4</td>
<td>25</td>
<td>5m 55 s</td>
</tr>
<tr>
<td>Locomotive</td>
<td>3</td>
<td>30</td>
<td>42.4 s</td>
</tr>
<tr>
<td>Thermostat</td>
<td>5</td>
<td>35</td>
<td>6.9 s</td>
</tr>
<tr>
<td>Thermostat</td>
<td>5</td>
<td>40</td>
<td>15.1 s</td>
</tr>
<tr>
<td>Thermostat</td>
<td>5</td>
<td>45</td>
<td>33.4 s</td>
</tr>
<tr>
<td>Non.Lin.Kin.</td>
<td>3</td>
<td>20</td>
<td>2m 25 s</td>
</tr>
</tbody>
</table>

Inferences:
1. Verification time grows nonlinearly with time.
2. Nonlinear constraint solving takes much more time than linear.
Contributions of this work:

1. Demonstrates that Off-the-shelf tools do not work when real time scheduling is taken into account.
2. Conceptually simple solution for verification.
3. Solution performs better than existing approaches.
Contributions of this work:

1. Demonstrates that Off-the-shelf tools do not work when real time scheduling is taken into account.
2. Conceptually simple solution for verification.
3. Solution performs better than existing approaches.

Eventual goal of the work:

End-to-end verification of real time CPS.

Is this one of the final solutions? - No.

Key new idea: Expose lower level implementation details to higher level for better verification.
Future Work

Exposing Proof Certificates At Each Layer

- Model checking hybrid systems
- Software verification of embedded code
- Scheduling analysis
- Hardware correctness proofs
- Sound approx. model
- Plant
- Noisy environment
Future Work

Exposing Proof Certificates At Each Layer

Model checking hybrid systems

Software verification of embedded code

Scheduling analysis

Hardware correctness proofs

Sound approx. model

Plant

Noisy environment

Thank You.

Questions?