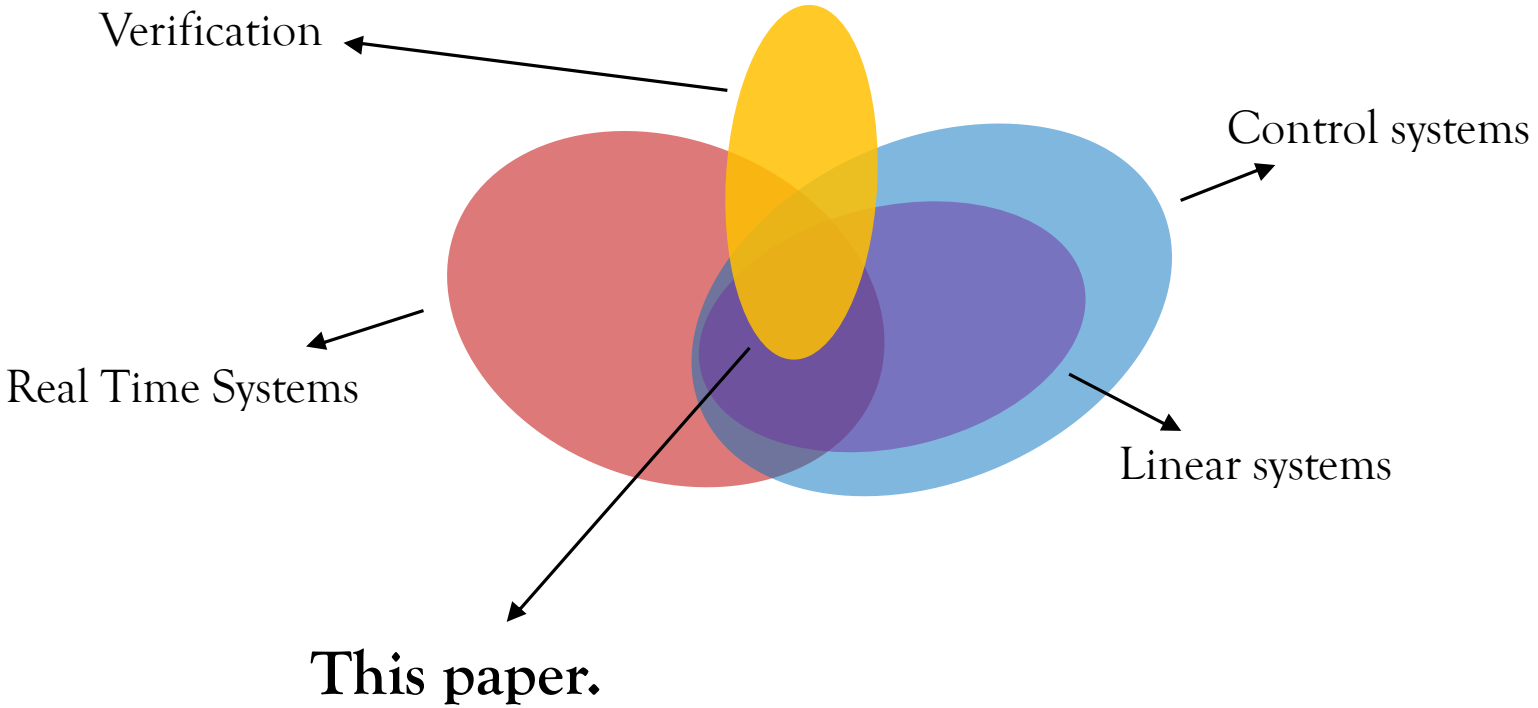
ANALYZING REAL TIME LINEAR CONTROL SYSTEMS USING SOFTWARE VERIFICATION

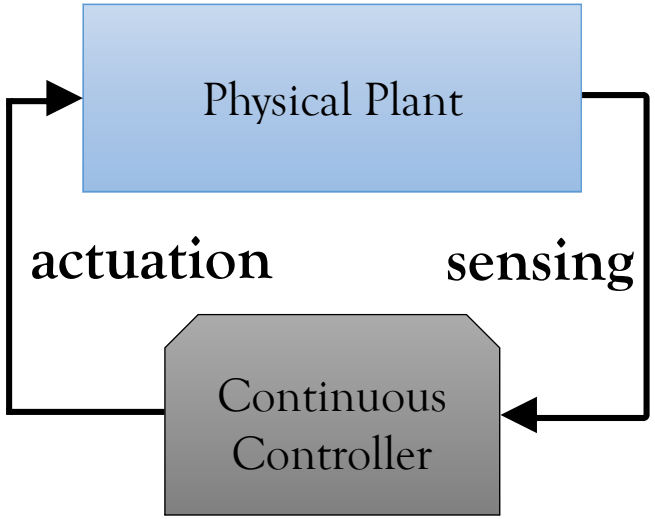
Parasara **Sridhar** Duggirala – UConn
Mahesh Viswanathan – UIUC

Real-Time Systems
+
Linear Control Systems
+
Verification



Isn't That Hybrid Systems Verification?

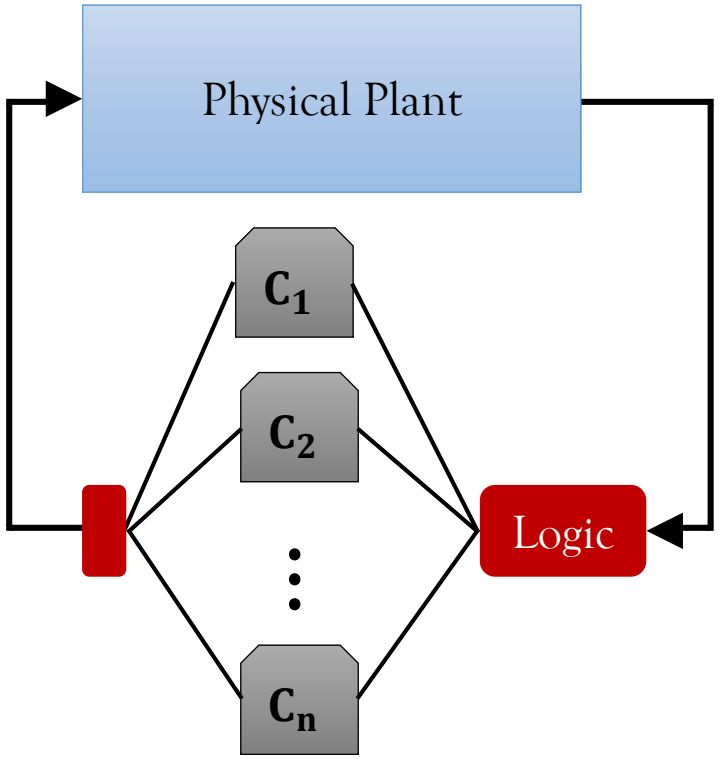
- Yes and No.



Typical control system

Isn't That Hybrid Systems Verification?

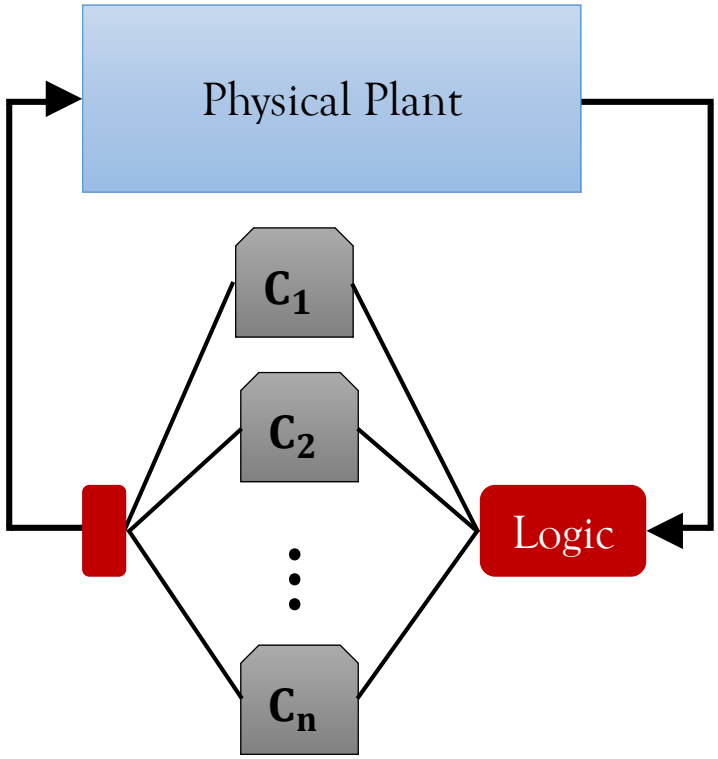
- Yes and No.



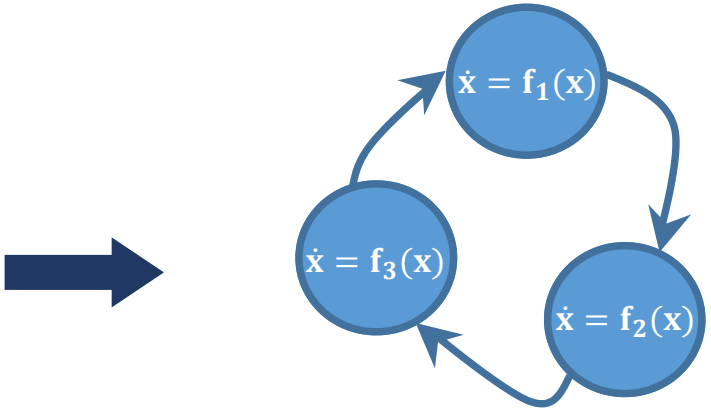
Typical hybrid system

Isn't That Hybrid Systems Verification?

- Yes and No.



Typical hybrid system



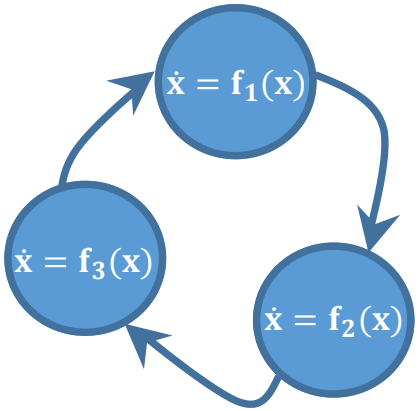
Hybrid Automata

Assumptions:

1. Continuous feedback
2. Exact computations

Isn't That Hybrid Systems Verification?

- Technically Yes, practically No.

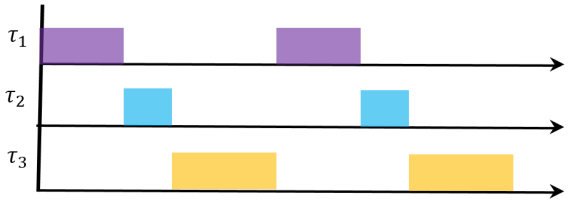


Hybrid Automata

VS

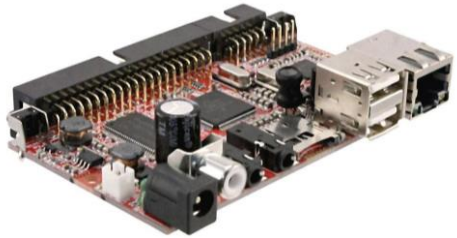
Floating points,
Data structures, …

+



Scheduling, …

+



Hardware, …

+

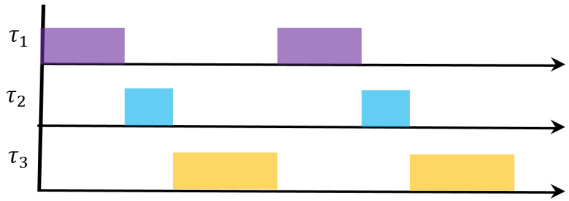
Plant
+
Noisy environment

Approx. model, …

Closely Related Works

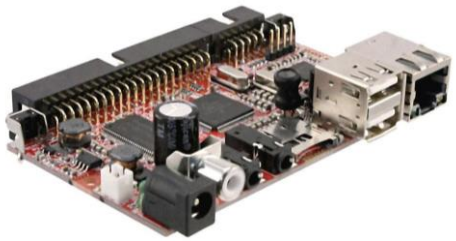
Floating points,
Data structures, …

\+



Scheduling, …

\+



Hardware, …

\+

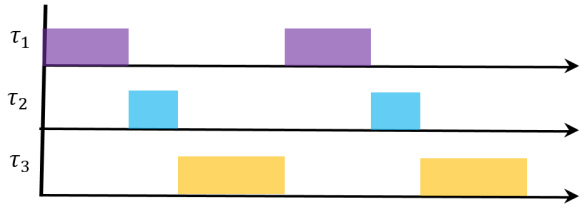
Plant
\+
Noisy environment

Approx. model, …

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

Closely Related Works

+



Scheduling

+

Physical Plant

Linear System

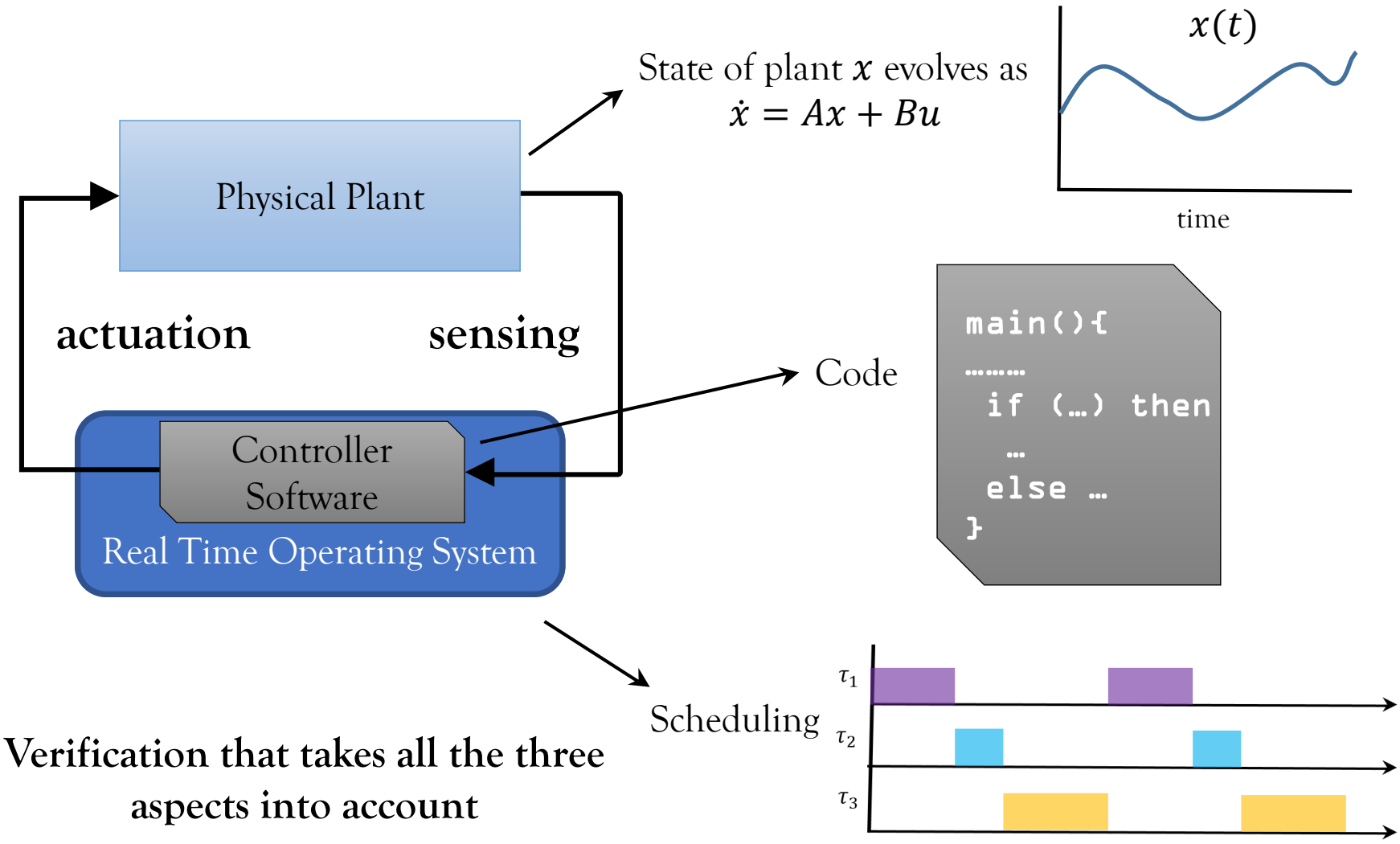
Computation delay

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



Verification that takes all the three aspects into account

Outline

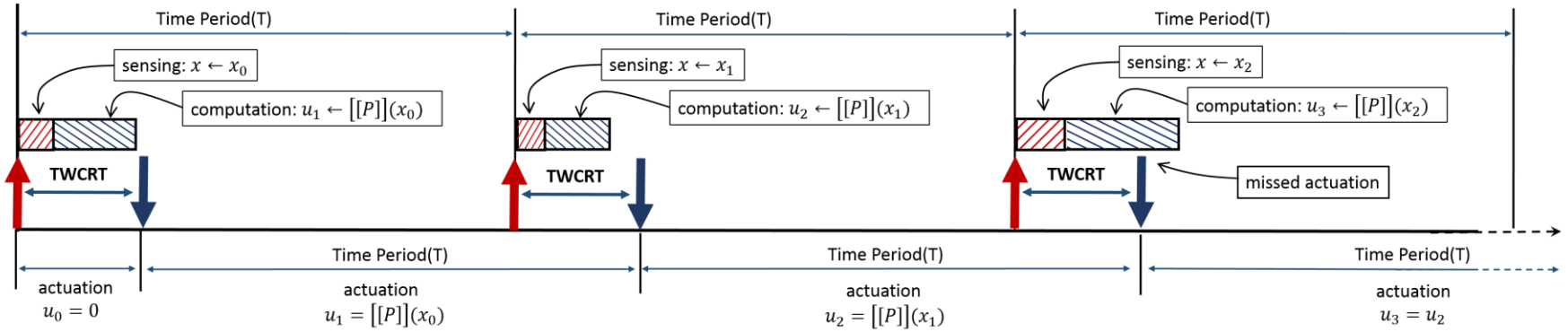
- Introduction
- Computational model
- Drawbacks of existing techniques (or advantages?)
- Software verification inspired technique
  - *Analyzing linear control systems*
  - *Accounting for timing analysis*
- Software verification techniques used
- Results
- Discussion and Future work

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.

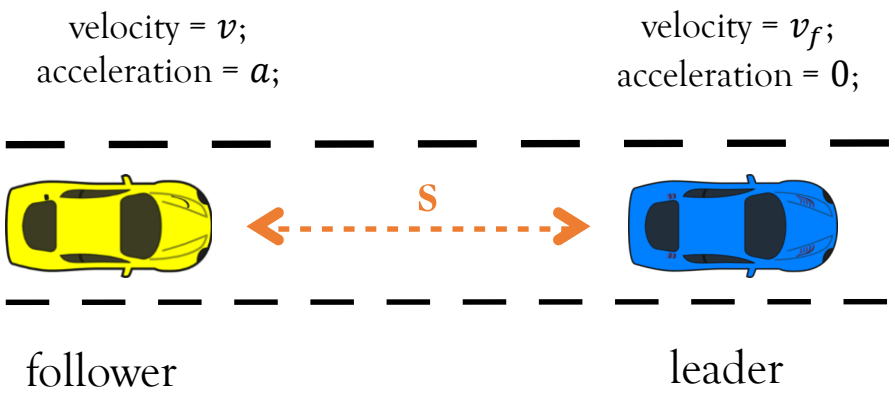
Computational Model

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 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



Dynamics of the system

$$\dot{s} = v_f - v;$$

$$\dot{v} = a - k_{aero} v;$$

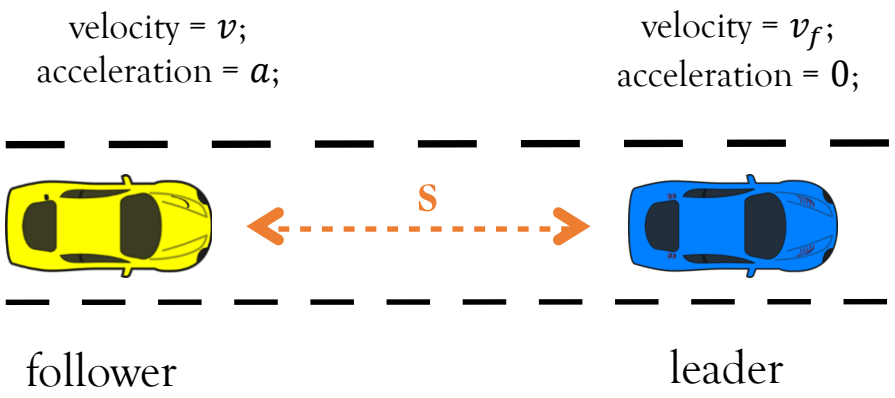
$$\dot{a} = u;$$

k_{aero} is the air-drag

Control Law

$$u = -2a - 2(v - v_f)$$

Motivating Example
Leader-Follower System



Dynamics of the system

$$\dot{s} = v_f - v;$$

$$\dot{v} = a - k_{aero} v;$$

$$\dot{a} = u;$$

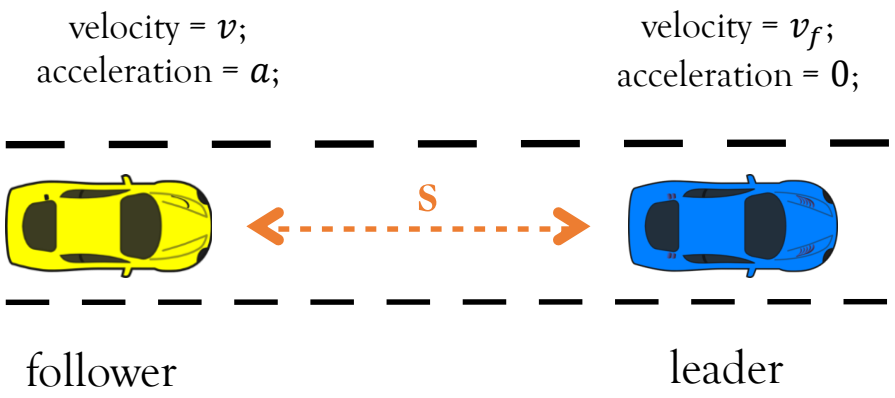
k_{aero} is the air-drag

Control Law

$$u = -2a - 2(v - v_f)$$

- 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.*

Motivating Example
Leader-Follower System



Dynamics of the system

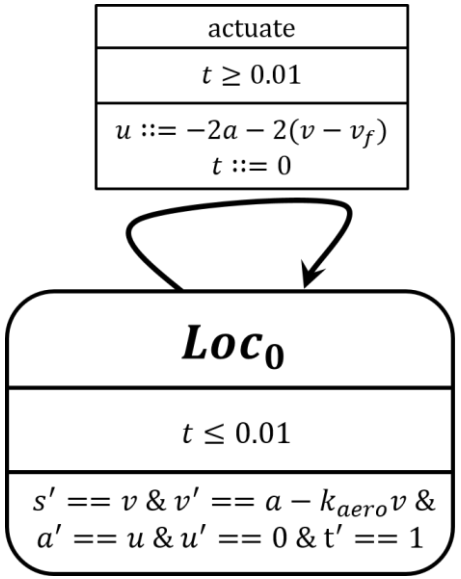
$$\dot{s} = v_f - v;$$
$$\dot{v} = a - k_{aero} v;$$
$$\dot{a} = u;$$

k_{aero} is the air-drag

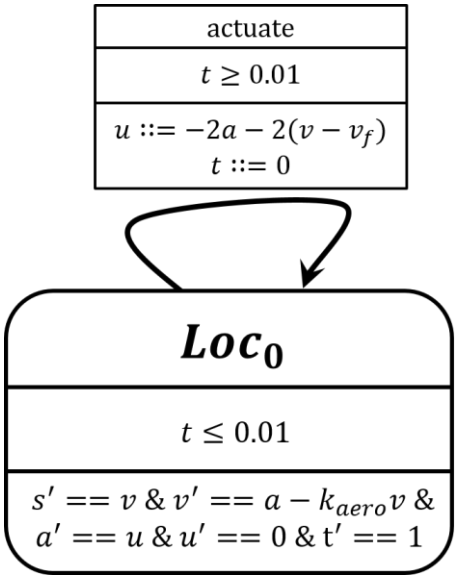
Control Law

$$u = -2a - 2(v - v_f)$$

- 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.*

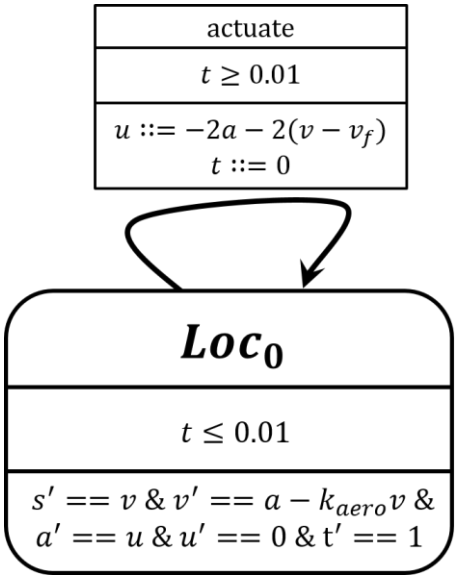


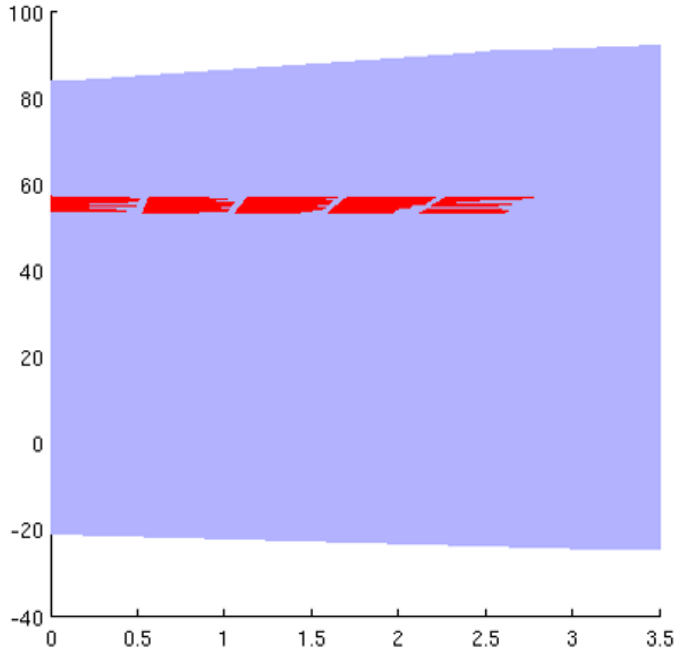
Naïve Hybrid Systems Verification With SpaceEx



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

Naïve Hybrid Systems Verification With SpaceEx





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

Property cannot be inferred!
Overapproximation is too high

Why It Does Not Work

(And Why It Should Not)

- Two source 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 source 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 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!*

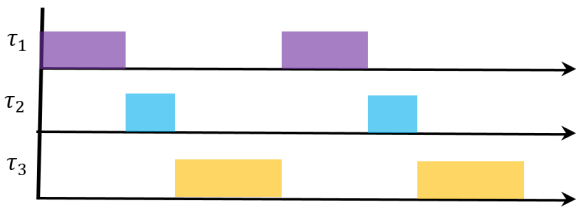
Outline

- 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

Software Verification Inspired Technique: Outline

```
main(){
……
 if (…) then
  …
 else …
}
```

+

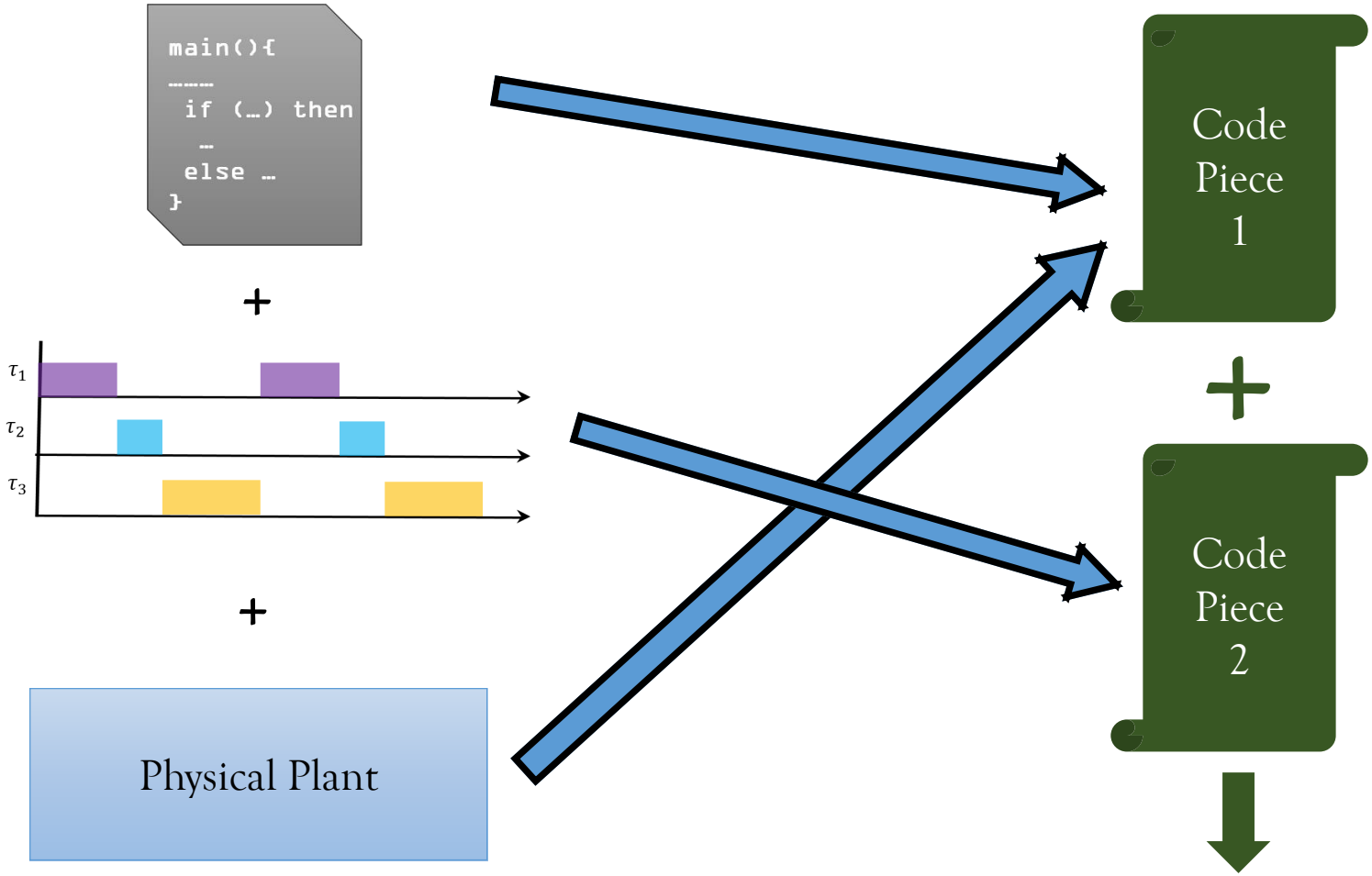


+

Physical Plant

Generated code simulates the closed loop system by tracking the software state and physical state of the plant.

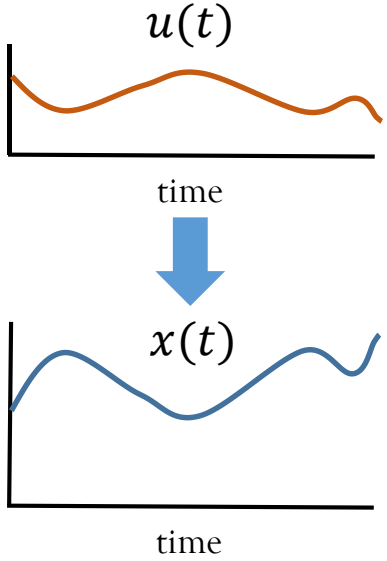
Software Verification Inspired Technique: Outline



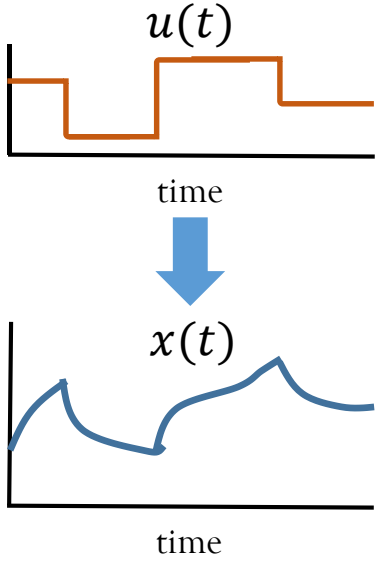
Part 1 – Analyzing Linear Control System

- 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

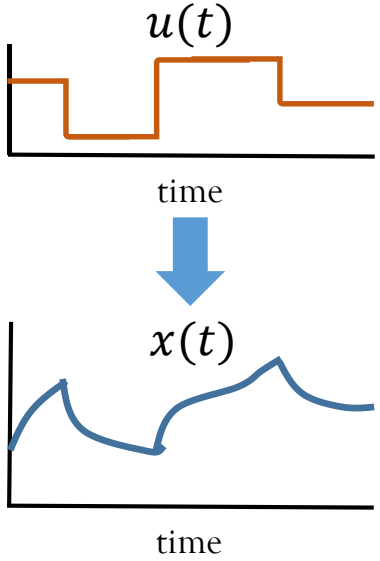
$$e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau\,.$$

- 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.

$$s_n = s - 0.0995 * (v - v_f) - 0.005 * a - 0.002 * u;$$
$$v_n = v_f + 0.99 * (v - vf) + 0.0995 * a + 0.005 * u;$$
$$a_n = a + 0.1 * u;$$

Note: Relation between u and s_n, v_n, a_n is **symbolic**.

Part 1 – Analyzing Linear Control System

- What about with the control law?

$u = -2a - 2(v - v_f)$;
$s_n = s - 0.0995 * (v - v_f) - 0.005 * a - 0.002 * u$;
$v_n = v_f + 0.99 * (v - vf) + 0.0995 * a + 0.005 * u$;
$a_n = a + 0.1 * u$;

Note: $u > 0$ initially if and only if $v < v_f$.

Part 1 – Analyzing Linear Control System

- What about with the control law?

$u = -2a - 2(v - v_f);$
$s_n = s - 0.0995 * (v - v_f) - 0.005 * a - 0.002 * u;$
$v_n = v_f + 0.99 * (v - vf) + 0.0995 * a + 0.005 * u;$
$a_n = a + 0.1 * u;$

Note: $u > 0$ initially if and only if $v < v_f$.

Code Piece 1 =

```
u = -2*a_s -2*(v_s - vf_s);
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;
```

Skipping details: Error analysis and soudness 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:

# deadline misses	consecutive executions
1	3
2	5

Part 2 – Handling the Timing Analysis and Scheduling

- Example:

# deadline misses	consecutive executions
1	3
2	5

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

Part 2 – Handling the Timing Analysis and Scheduling

- Example:

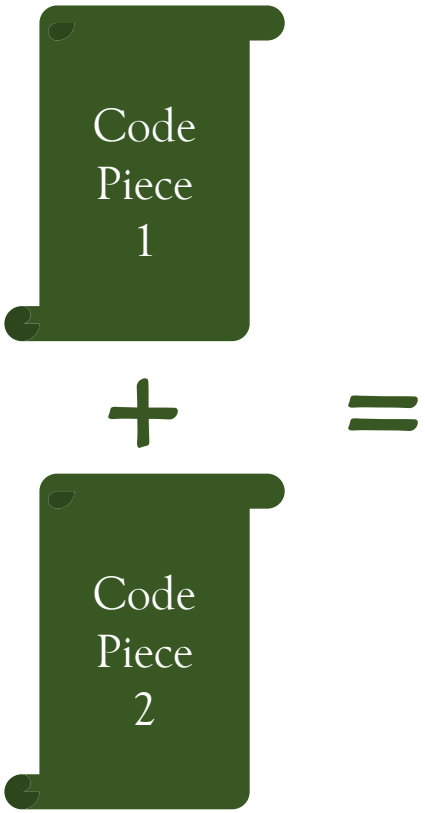
# deadline misses	consecutive executions
1	3
2	5

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

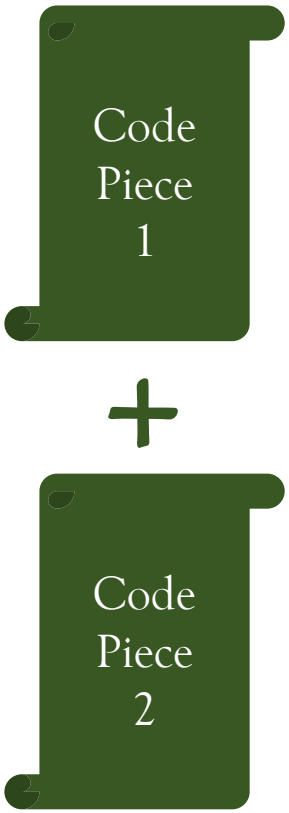
Code Piece 2 =

```
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;
```

Bringing These Two Together



Bringing These Two Together



=

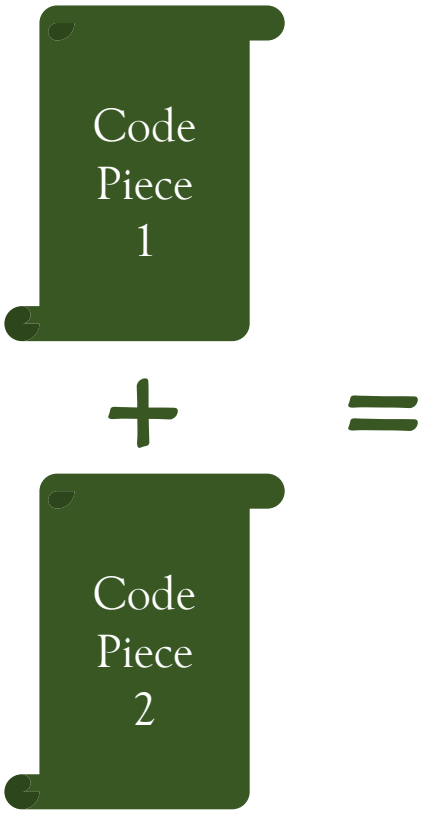
```
u = -2*a_s -2*(v_s - vf_s);
```

```
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;
```

```
// Update actuation parameters if deadline is met
if(deadline_met == 1)
  u_a = u;
endif;
```

```
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;
```

Bringing These Two Together



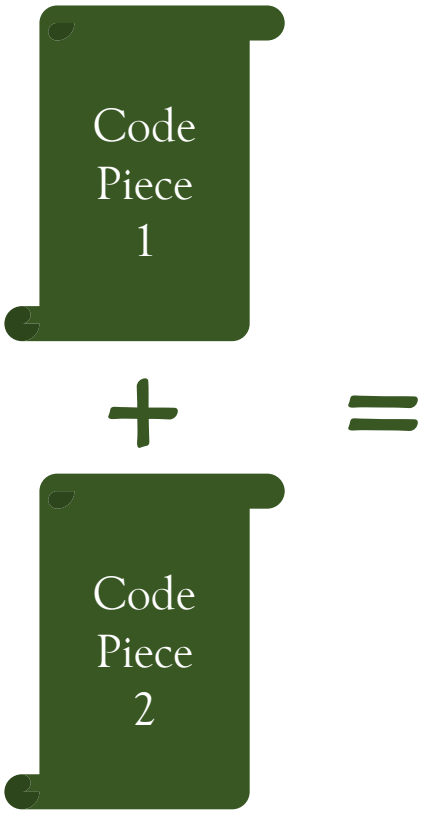

```
u = -2*a_s -2*(v_s - vf_s);
```

```
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;
```

```
// Update actuation parameters if deadline is met
if(deadline_met == 1)
  u_a = u;
endif;
```

```
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;
```

Bringing These Two Together



Controller code

```
u = -2*a_s -2*(v_s - vf_s);
```

```
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;
```

Timing Behavior

```
// Update actuation parameters if deadline is met
if(deadline_met == 1)
  u_a = u;
endif;
```

Updating actuation only when deadline is met

```
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;
```

Plant behavior

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

Problem	Steps	Z3	Interproc			SpaceEx		
			Box	Oct	Poly	Box	Oct	Poly
ACC1	25	P, 25.8 s	F, 0.2 s	F, 12.2 s	P, 18m 50 s	F, 0.3 s	F, 10.3 s	F, 32.8 s
ACC2	25	P, 25.9 s	P, 0.2 s	F, 12.1 s	P, 18m 22 s	F, 0.3 s	F, 10.3 s	F, 32.6 s
Kin1	25	P, 5.8 s	F, 0.05 s	F, 1.8 s	P, 4m 18 s	F, 0.2 s	F, 2.5 s	F, 25.9 s
Kin2	25	P, 5.8 s	P, 0.05 s	F, 1.8 s	P, 4m 20 s	F, 0.2 s	P, 2.4 s	F, 25.8 s

Results – Part 1
Z3 vs AI vs SpaceEx

Problem	Steps	Z3	Interproc			SpaceEx		
			Box	Oct	Poly	Box	Oct	Poly
ACC1	25	P, 25.8 s	F, 0.2 s	F, 12.2 s	P, 18m 50 s	F, 0.3 s	F, 10.3 s	F, 32.8 s
ACC2	25	P, 25.9 s	P, 0.2 s	F, 12.1 s	P, 18m 22 s	F, 0.3 s	F, 10.3 s	F, 32.6 s
Kin1	25	P, 5.8 s	F, 0.05 s	F, 1.8 s	P, 4m 18 s	F, 0.2 s	F, 2.5 s	F, 25.9 s
Kin2	25	P, 5.8 s	P, 0.05 s	F, 1.8 s	P, 4m 20 s	F, 0.2 s	P, 2.4 s	F, 25.8 s

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

Benchmark	Dimn.	Steps	Time
MTSC	4	15	12.6 s
MTSC	4	20	1m 14 s
MTSC	4	25	5m 55 s
Locomotive	3	30	42.4 s
Thermostat	5	35	6.9 s
Thermostat	5	40	15.1 s
Thermostat	5	45	33.4 s
Non.Lin.Kin.	3	20	2m 25 s

Inferences:

1. Verification time grows nonlinearly with time.
2. Nonlinear constraint solving takes much more time than linear.

Discussion And Future Work

- 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.*

Discussion And Future Work

- 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.*

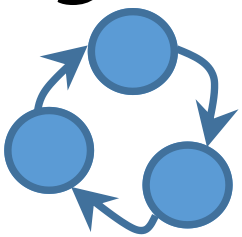
- Eventual goal of the work:

  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

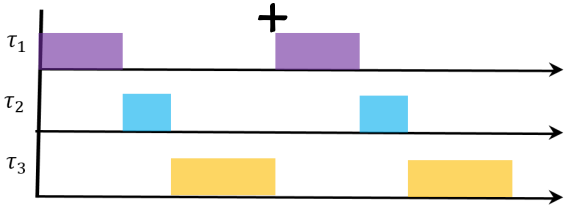


Model checking
hybrid systems

+

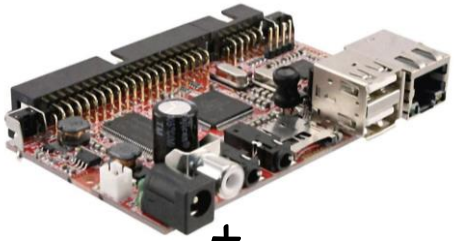
Software verification of
embedded code

+



Scheduling analysis

+



Hardware correctness proofs

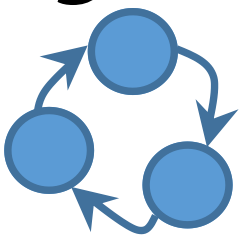
+

Plant
+
Noisy environment

Sound approx. model

Future Work

Exposing Proof Certificates At Each Layer

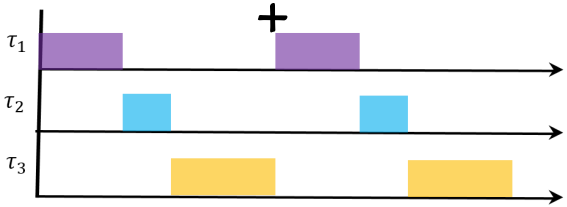


Model checking
hybrid systems

+

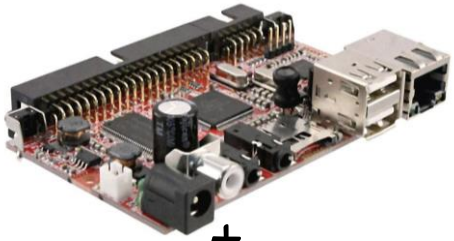
Software verification of
embedded code

+



Scheduling analysis

+



Hardware correctness proofs

+

Plant
+
Noisy environment

Sound approx. model

Thank You.

Questions?