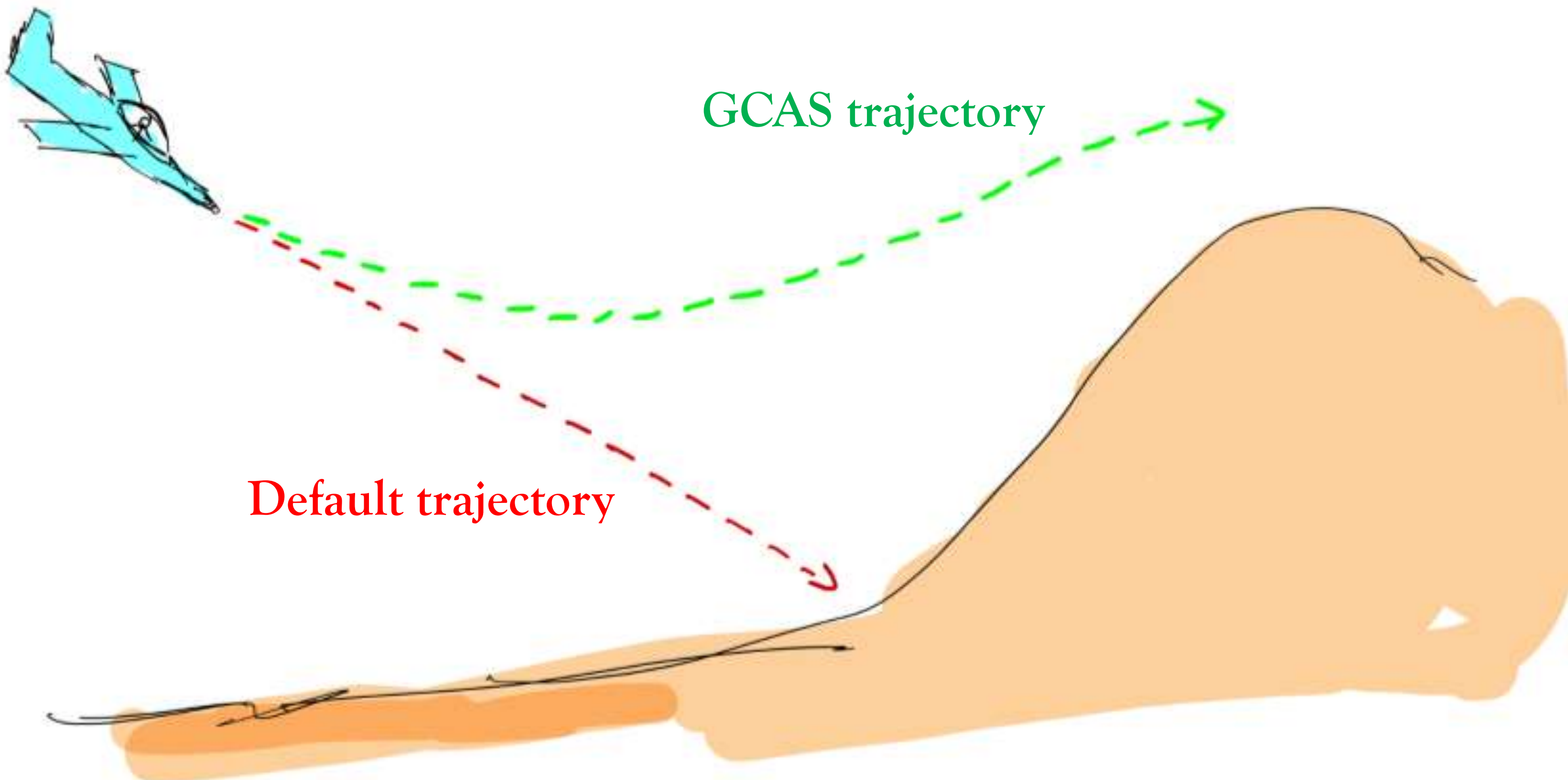
Scalable Formal Verification of Cyber-Physical Systems

Parasara Sridhar Duggirala

UCONN
UNIVERSITY OF CONNECTICUT

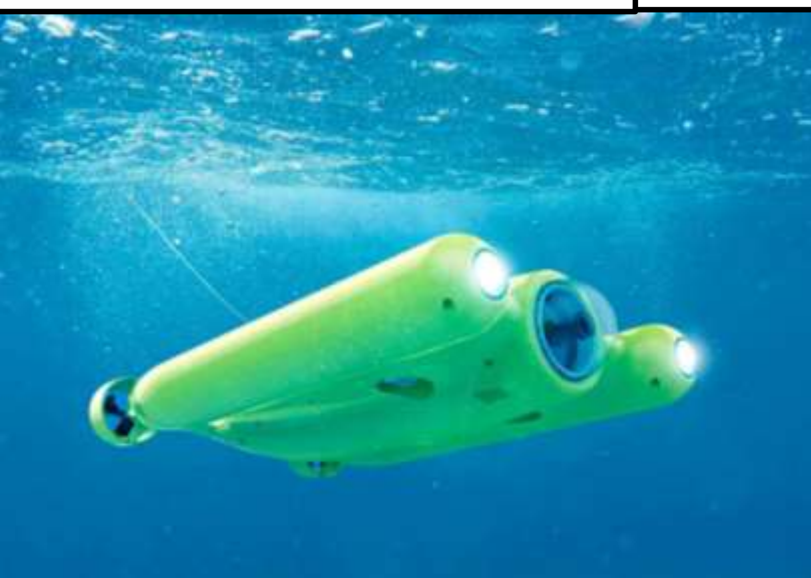
Ground Collision Avoidance System

What Happened?



Life saved because of software!

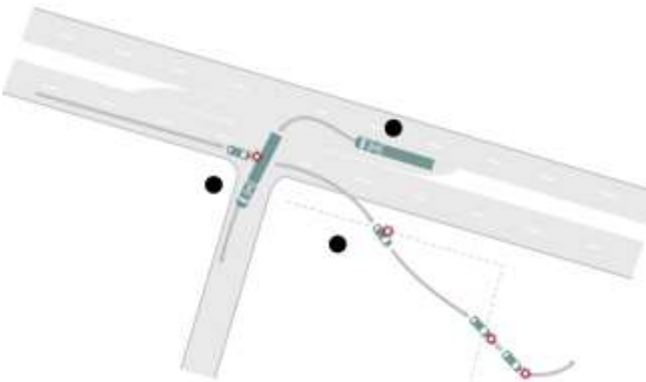
Cyber-Physical Systems
Are Everywhere!



Sometimes, CPS have bugs

BUSINESS DAY

Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says



BUSINESS DAY

Tesla Says Crashed Vehicle Had Been on Autopilot Before Fatal Accident

THE VERGE TECH SCIENCE CULTURE CARS REVIEWS LONGFORM VIDEO MORE

POLICY & LAW \ US & WORLD \ TRANSPORTATION

Uber suspended from autonomous vehicle testing in Arizona following fatal crash

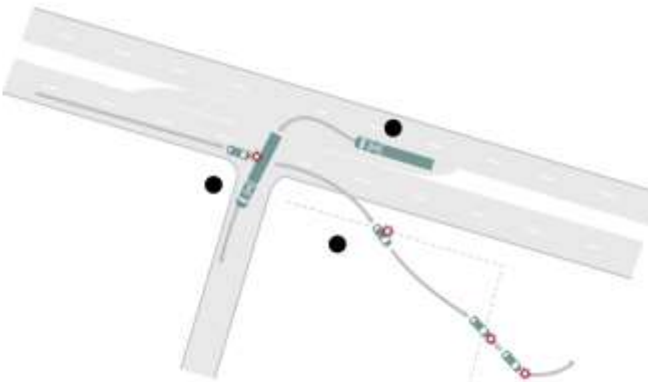
Arizona governor calls Uber crash an 'unquestionable failure'

By Nick Statt | @nickstatt | Mar 26, 2018, 9:12pm EDT

Sometimes, CPS have bugs

BUSINESS DAY

Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says



THE VERGE TECH SCIENCE CULTURE CARS REVIEWS LONGFORM VIDEO MORE

POLICY & LAW \ US & WORLD \ TRANSPORTATION

Uber suspended from autonomous vehicle testing in Arizona following fatal crash

Arizona governor calls Uber crash an 'unquestionable failure'

California's Autonomous Car Reports Are The Best In The Country—But Nowhere Near Good Enough

Ryan Felton

2/01/18 10:29am · Filed to: GENERAL MOTORS

Disengagement rates

0.16 – 0.78 for 1000 miles

BUSINESS DAY

Tesla Says Crashed Vehicle Had Been on Autopilot Before Fatal Accident

Uber self driving car running red light.
https://www.youtube.com/watch?v=_CdJ4oae8f4

- ❑ Toyota recalls of Prius vehicles (> 20M).
- ❑ Software failures in medical devices (approx. 25%)
- ❑ Northeast power grid blackouts.

EVENING
Courier
THE VOICE OF CALERDALE

TUESDAY 1ST APRIL 2025

Check out our jobs section

Inside: pages of homes for sale

AMAZON DRONE CAUSES TROUBLE

AMAZON DRONE DROPS PACKAGE ON PEDESTRIANS HEAD, INJURES HIM SERIOUSLY.

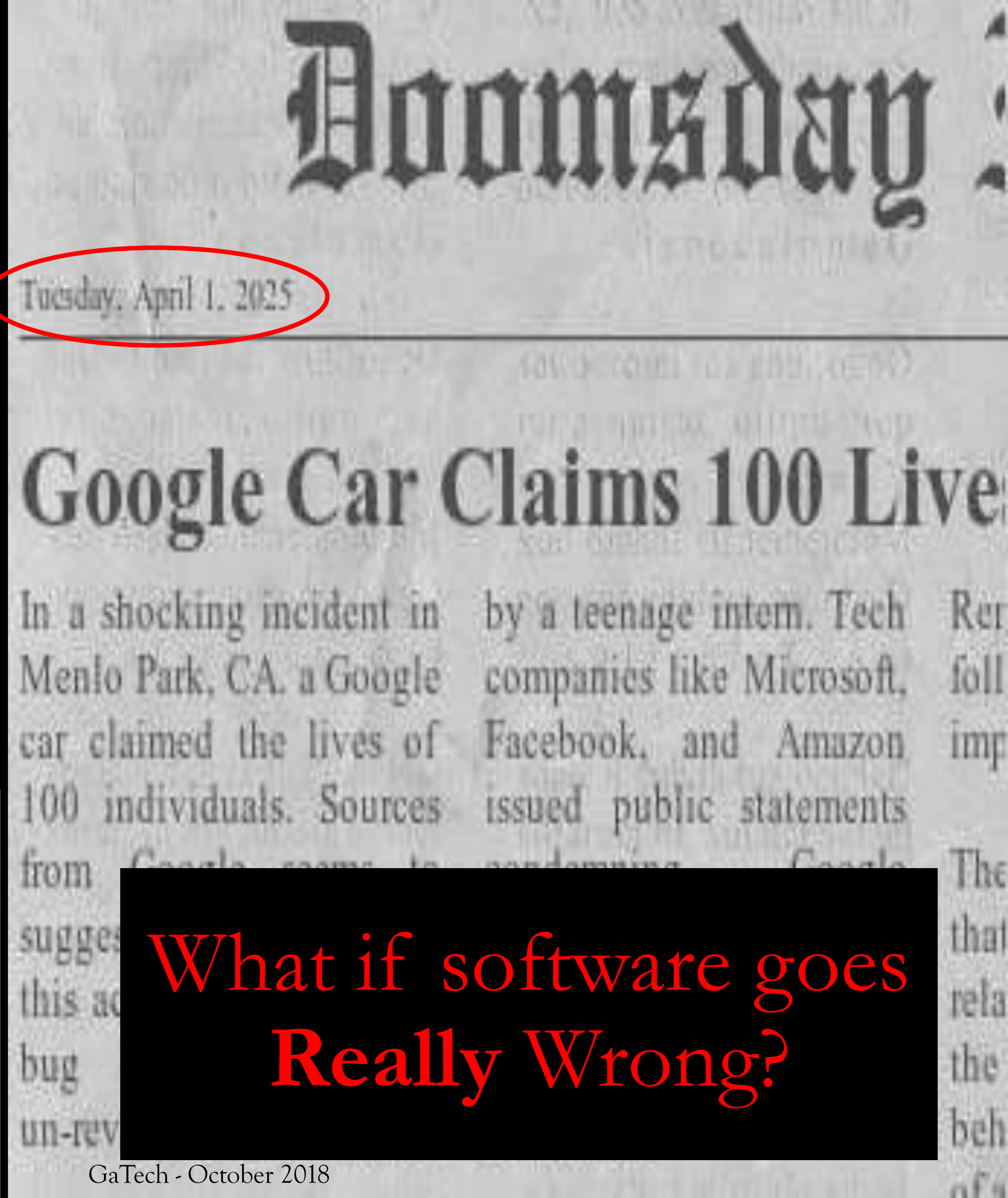
In a disastrous consequence of events in Chicago, an Amazon drone carrying delivery package to a prime account holder accidentally on a pedestrian. The pedestrian accused Amazon saying that the company deliberately did this because he cancelled his Prime account recently. Jeff Bezos, CEO of Amazon said that it was purely an accident caused by a bug in the drone software.

Drones have been subjected to a lot of criticism recently. The software seems to be written by underpaid and overworked interns without serious code reviews. Also, no one apart from the poor graduate students who developed the initial drones, no one seems to understand what is going on with these drones. Parasara Sridhar Duggirala, a graduate student in 2015 mentioned that this situation could have been prevented had he been offered enough funding to pursue research on safety mechanisms in civic drones.

Alibaba, another online retailer company spokesperson remarked "hahahahahaha". He added "see what happens when you allow reckless companies to do whatever they want. It was a stupid idea to let companies operate drones in civic spaces in first place, let alone without less regulations and certification mechanisms for these drones in place." He remarked "First lets stop Amazon from hurting more people, and then we can discuss these issues".

Smart-Grid plays dumb, causes 48 hours national blackout

By TECHNOLOGY CYNIC



Doomsday

Tuesday, April 1, 2025

Google Car Claims 100 Live

In a shocking incident in Menlo Park, CA, a Google car claimed the lives of 100 individuals. Sources from

by a teenage intern. Tech companies like Microsoft, Facebook, and Amazon issued public statements

What if software goes **Really** Wrong?

Cyber-Physical Systems Are Everywhere!

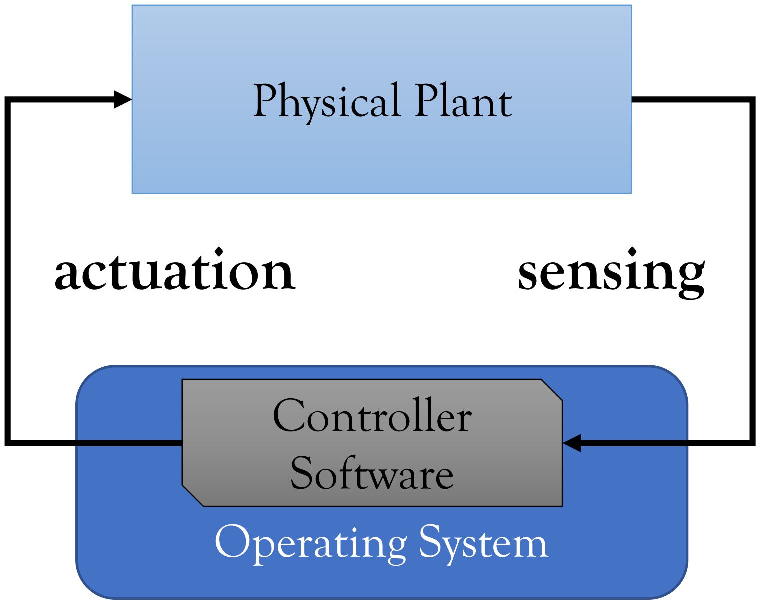




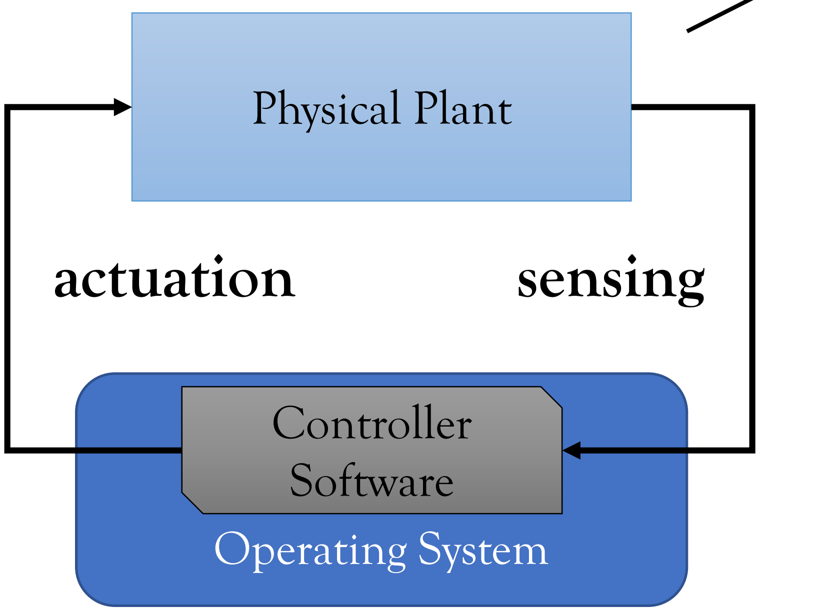
My Research Goal

Develop Principles, Algorithms, and Tools for Design, Analysis, and Verification of CPS

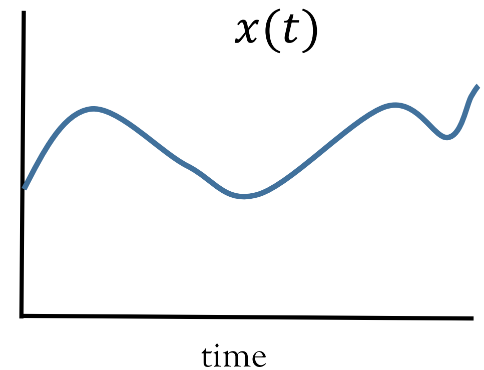

Why is CPS Verification Hard?



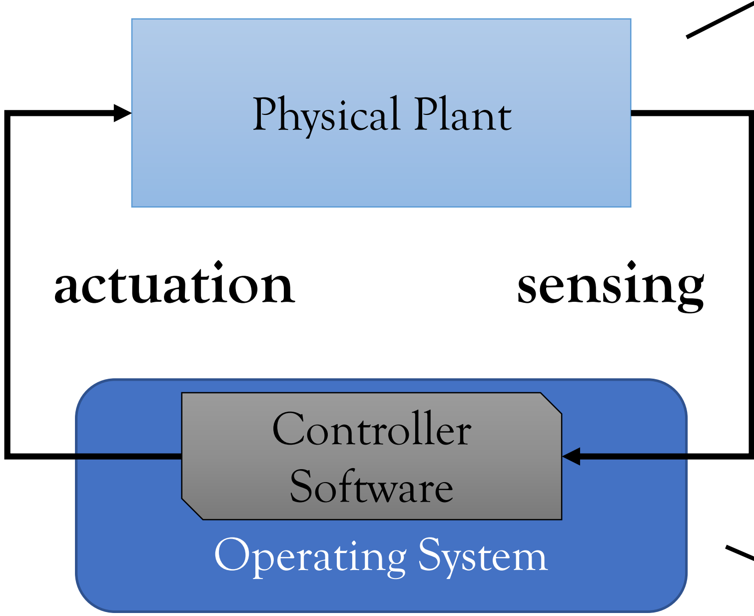
Why is CPS Verification Hard?



State of plant x
evolves as
$\dot{x} = f(x, u)$

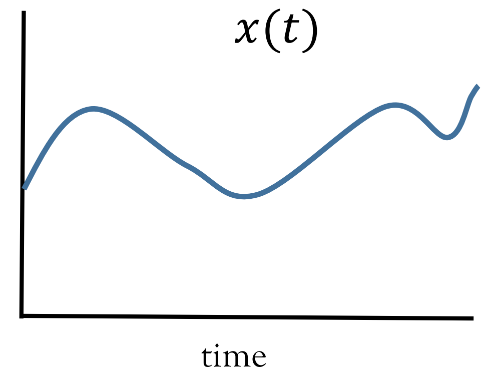


Why is CPS Verification Hard?



State of plant x evolves as

$$\dot{x} = f(x, u)$$



Code

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

Controls &(vs?) Computer Science

Old School

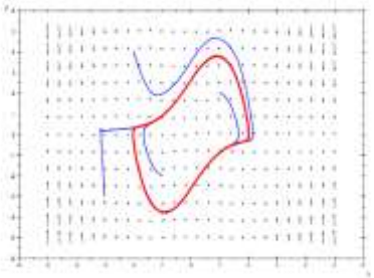
New School

Controls &(vs?) Computer Science

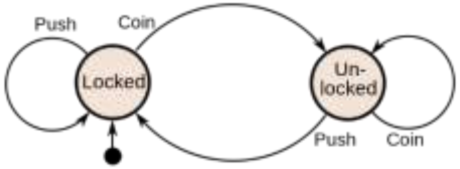
Old School

New School

Continuous domain



Discrete domain



Based on calculus

$$\dot{x} = f(x, u)$$

Based on Logic

$$((a \wedge \neg b) \Rightarrow c) \vee (d \wedge e)$$

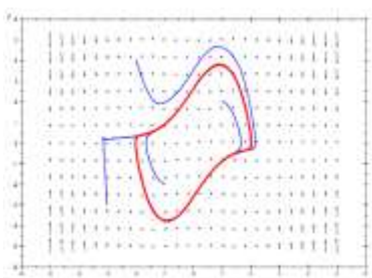
Controls &(vs?) Computer Science

هل يمكنك إثبات ذلك؟

Old School

New School

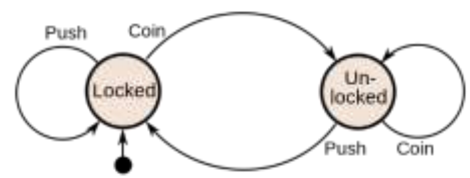
Continuous domain



Based on calculus

$$\dot{x} = f(x, u)$$

Discrete domain



Based on Logic

$$((a \wedge \neg b) \Rightarrow c) \vee (d \wedge e)$$

Controls &(vs?) Computer Science

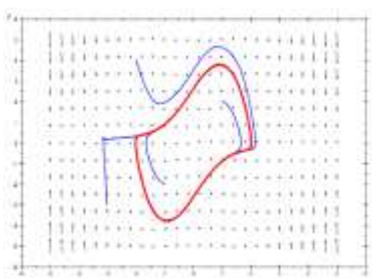
Old School

هل يمكنك إثبات ذلك؟

New School

是的，我可以證明它！

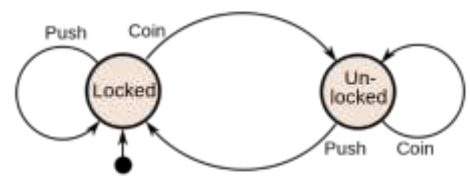
Continuous domain



Based on calculus

$$\dot{x} = f(x, u)$$

Discrete domain



Based on Logic

$$((a \wedge \neg b) \Rightarrow c) \vee (d \wedge e)$$

Controls &(vs?) Computer Science

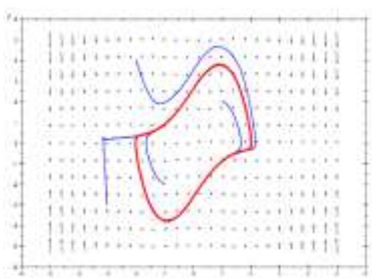
Old School

هل يمكنك إثبات ذلك؟

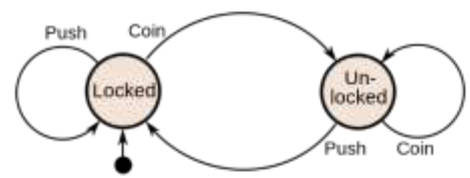
New School

是的，我可以證明它！

Continuous domain



Discrete domain



Based on calculus

$$\dot{x} = f(x, u)$$

Based on Logic

$$((a \wedge \neg b) \Rightarrow c) \vee (d \wedge e)$$

Not an ideal marriage!
But a necessary one.

Challenges in practice

- CPS that keep track of time: verification problem is **PSPACE Complete**
- CPS that have simple discontinuity: verification problem is **Undecidable**

Challenges in practice

- CPS that keep track of time: verification problem is **PSPACE Complete**
- CPS that have simple discontinuity: verification problem is **Undecidable**

- If the dynamics is given as “nice” differential equation $\dot{x} = Ax$ the solution for ODE is given as e^{At} where $e^{At} = I + At + \frac{1}{2!}(At)^2 + \cdots$.
- Scalability – 50 dimensions (before my work).

Challenges in practice

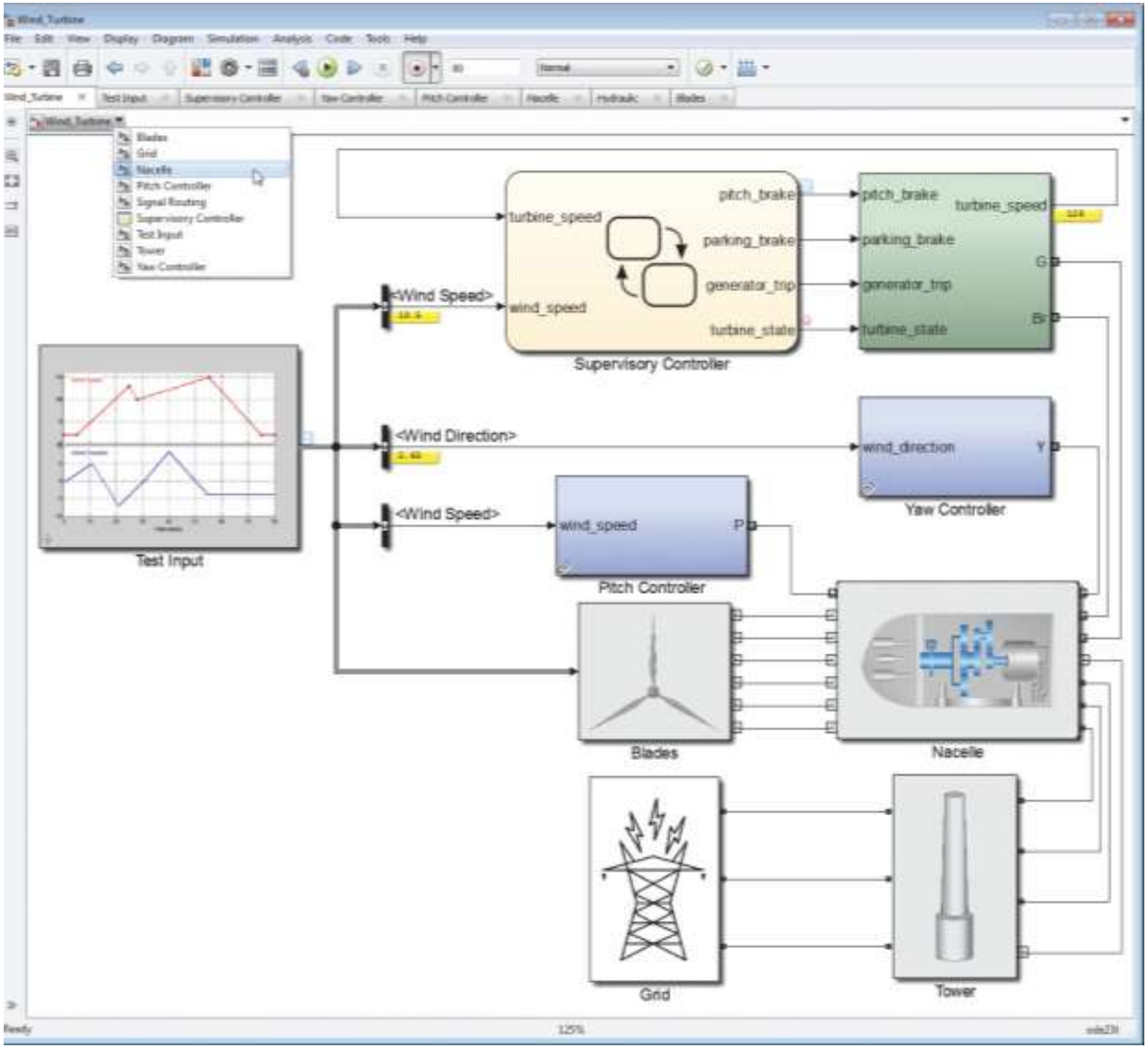
- CPS that keep track of time: verification problem is **PSPACE Complete**
- CPS that have simple discontinuity: verification problem is **Undecidable**

- If the dynamics is given as "nice" differential equation $\dot{x} = Ax$ the solution for ODE is given as e^{At} where $e^{At} = I + At + \frac{1}{2!}(At)^2 + \cdots$.
- Scalability – 50 dimensions (before my work).

- For nonlinear systems? Phew! The closed form solution does not exist!
- Scalability is just 7-8 dims (for general cases).

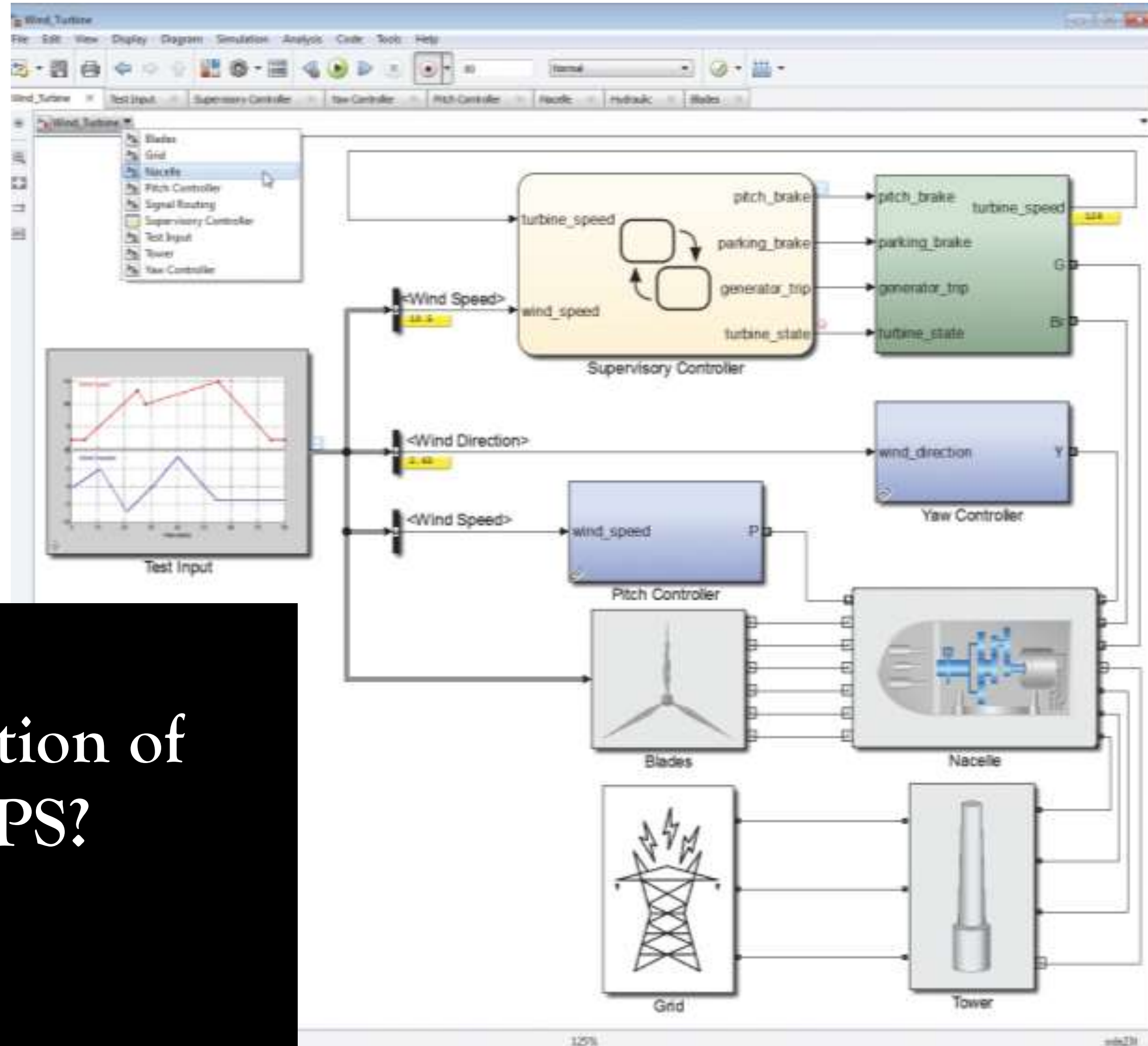
What Real Systems Look Like?

- Nonlinear
- Complex software
- Distributed
- Heterogenuous time scales
- Uncertainities
- Failures



What Real Systems Look Like?

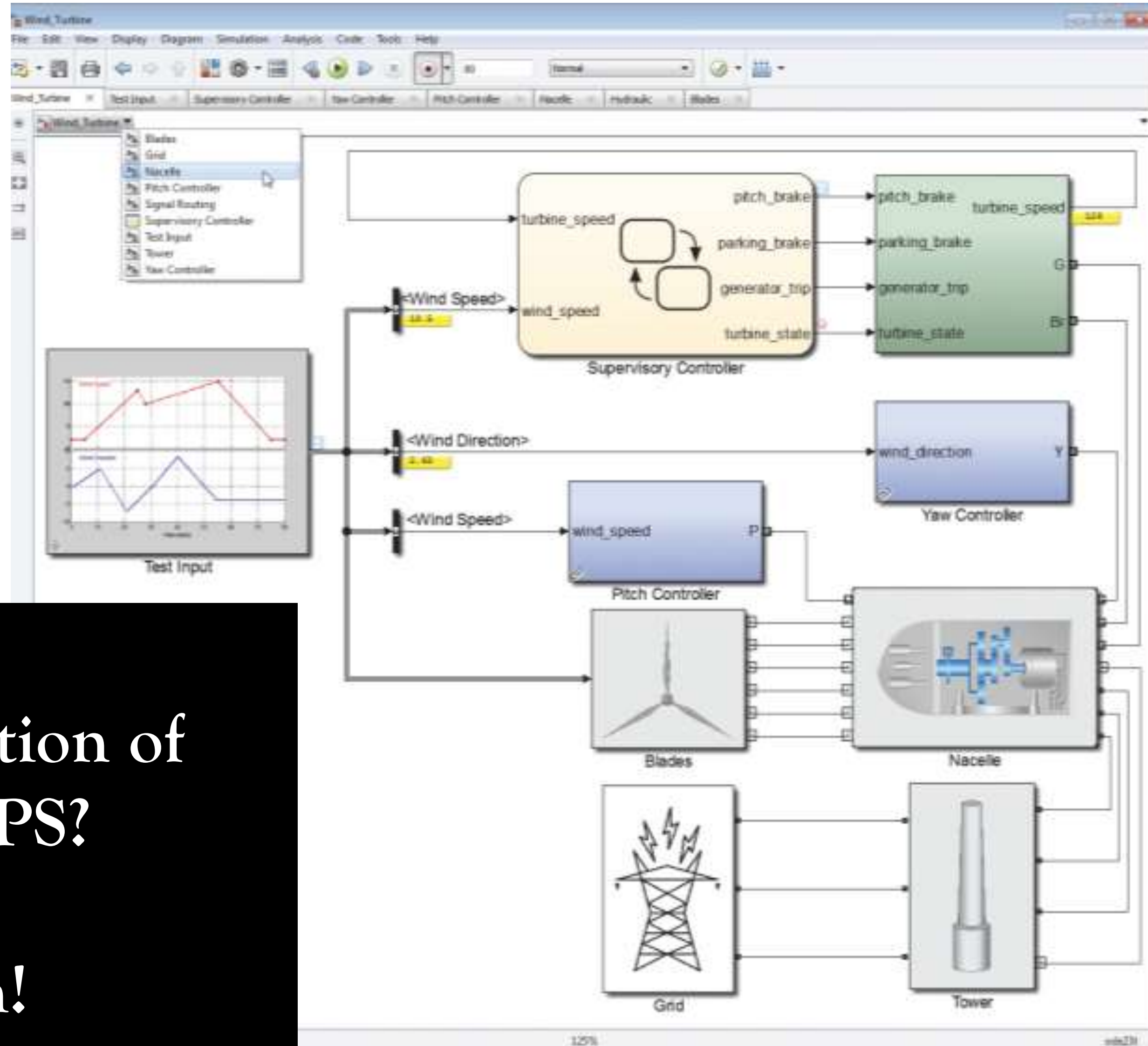
- Nonlinear
- Complex software
- Distributed
- Heterogenuous time scales
- Uncertainities
- Failures



Formal Verification of industrial CPS?

What Real Systems Look Like?

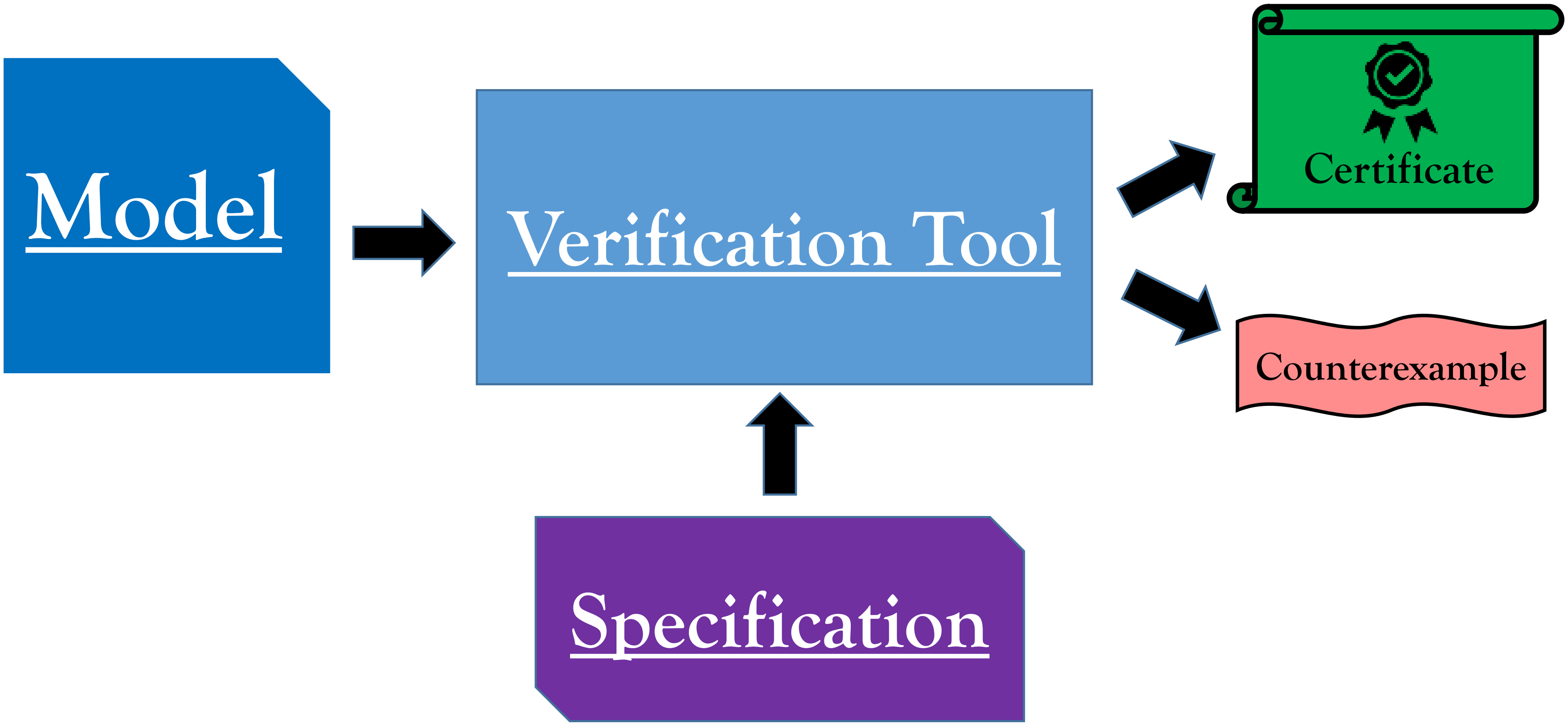
- Nonlinear
- Complex software
- Distributed
- Heterogenuous time scales
- Uncertainities
- Failures



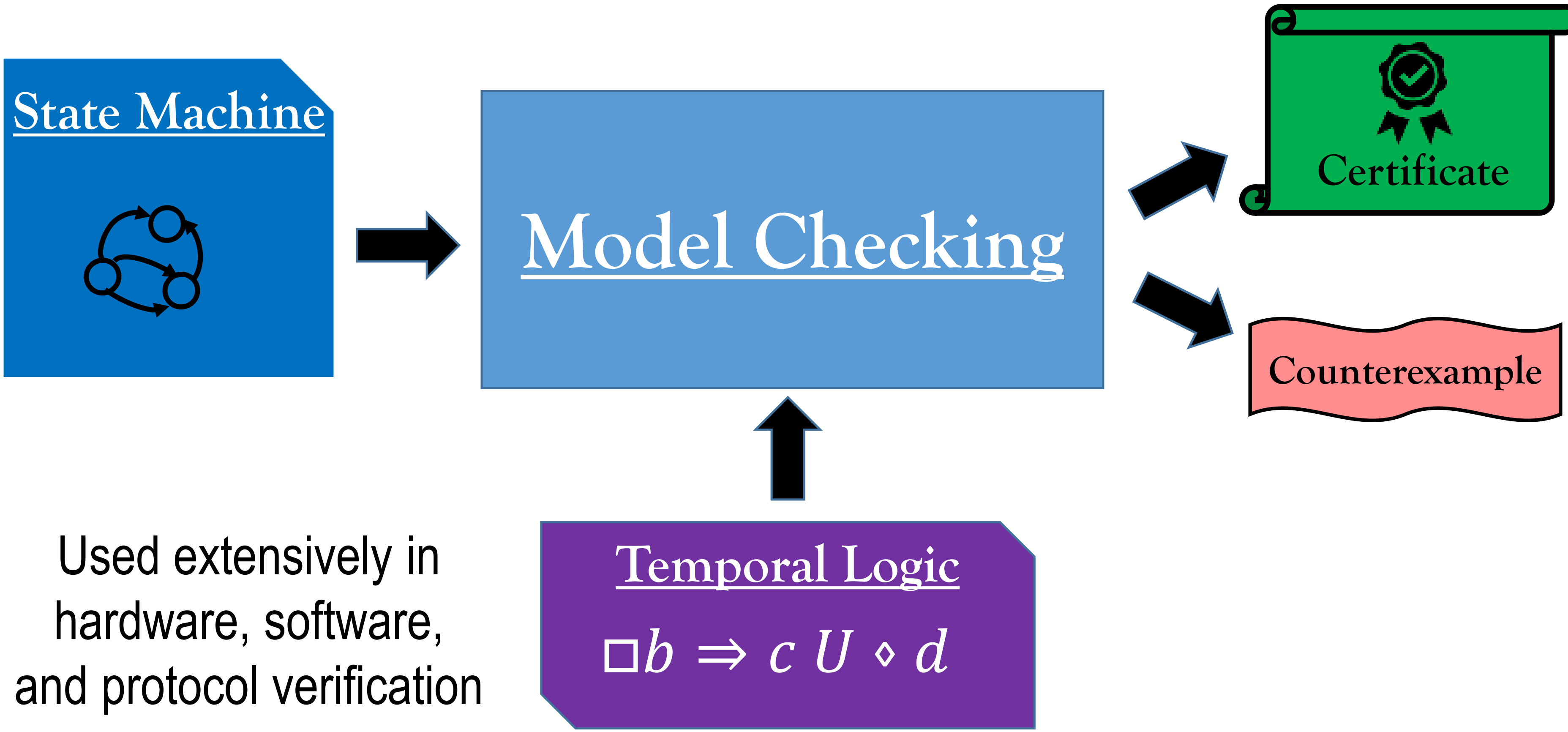
Formal Verification of industrial CPS?

Hallelujah!

Formal Verification 101



Formal Verification 101



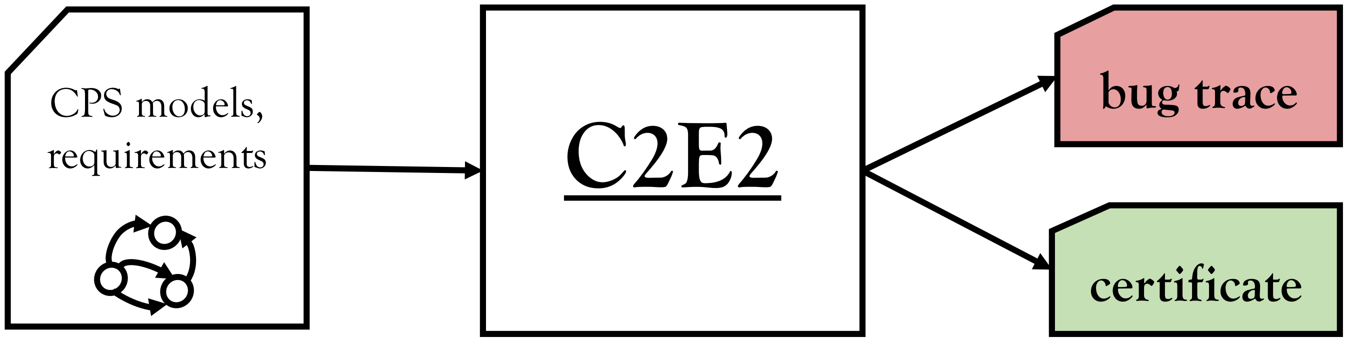
Used extensively in hardware, software, and protocol verification

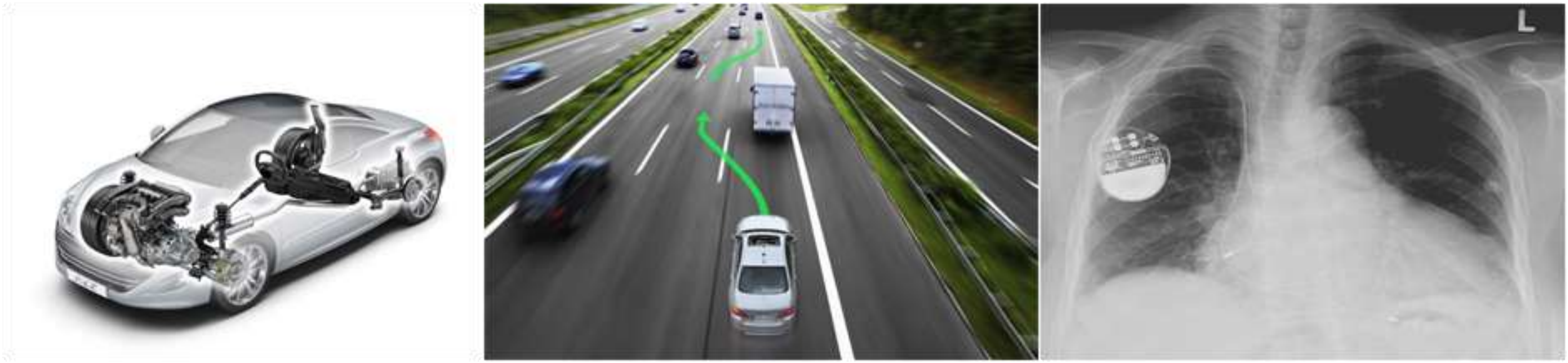
Outline

- ✓ Motivation
- Research Overview
- Scalable Verification of Linear Control Systems
- Future Work

Brief Summary of Past & Ongoing Projects

C2E2: A Tool For Verifying CPS Models with Nonlinear Dynamics





D, Mitra, Viswanathan EMSOFT'13
D, Mitra, Viswanathan, Potok, TACAS'15.
Fan, Potok, Mitra, Viswanathan , D CAV'16.

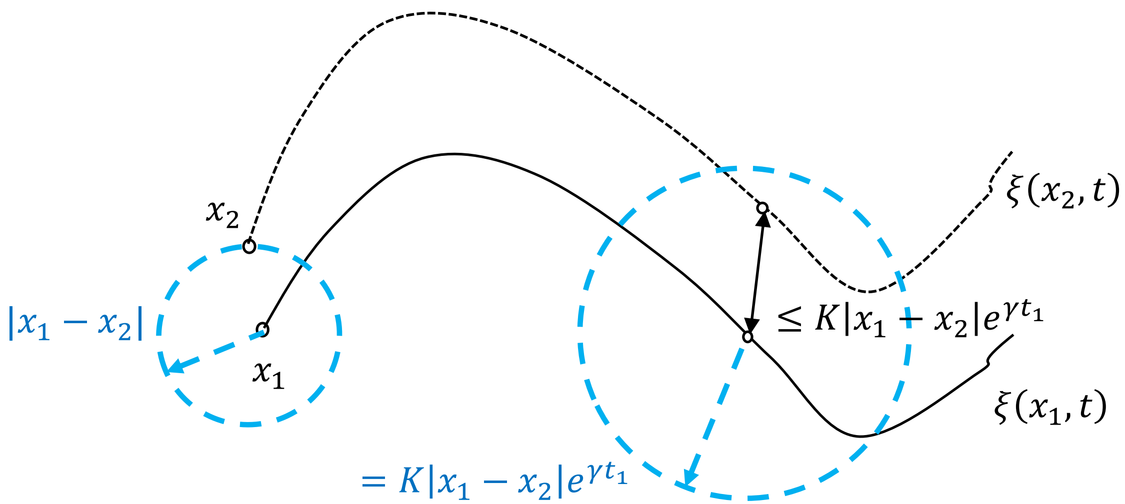
Zhenqi et.al. [CAV'14]
Fan et.al. [EMSOFT'16]

Safety Verification Problem

- **Problem statement:** Given dynamics $\dot{x} = \mathrm{f}(x)$, initial set Θ, unsafe set U, and time bound T, are all trajectories $\xi(x,t)$ starting from Θ, safe?

- Tool that is useful: Discrepancy function.

$\langle K,\gamma\rangle$ is called an exponential discrepancy function of the system if for any two states x_1 and $x_2 \in X$, for any t $|\xi(x_1,t) - \xi(x_2,t)| \le K|x_1 - x_2|e^{\gamma t}$



Soundness and Relative Completeness Results

- Always performing a sound analysis :

$$|x_1(t) - x_2(t)| \leq \beta(|x_1 - x_2|, t)$$

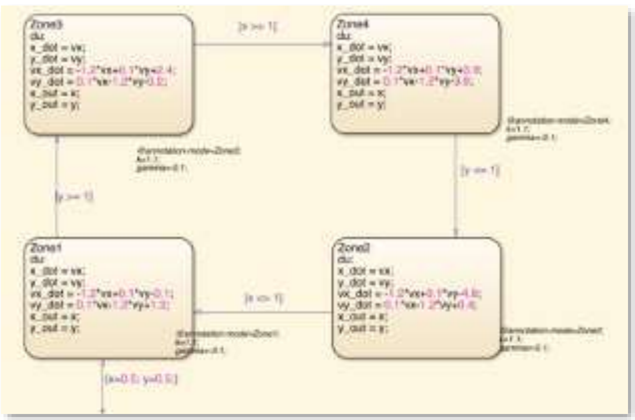
- Improving the partitioning improves the approximation

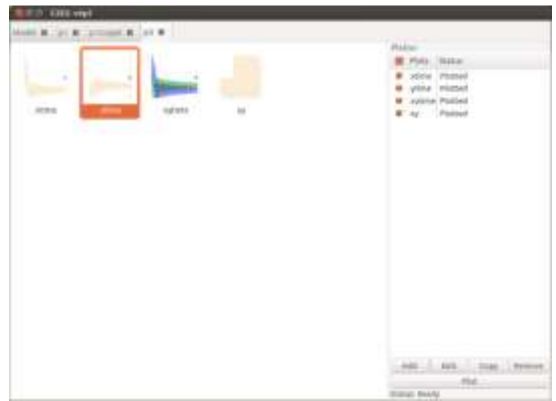
$$\beta(|x_1 - x_2|, t) \to 0 \text{ as } |x_1 - x_2| \to 0$$

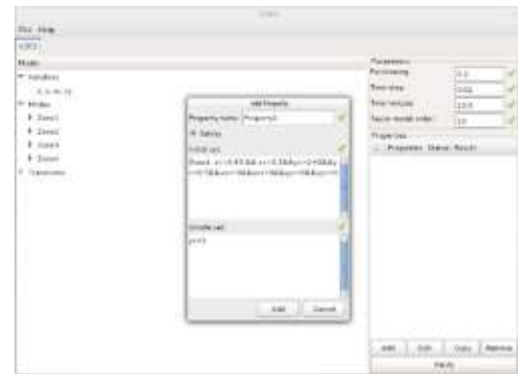
Theorem[Soundness]: Given any HA A, with an initial set Θ, and unsafe set U, if the algorithm terminates and returns **safe (unsafe)** then the system is indeed **safe (unsafe)**

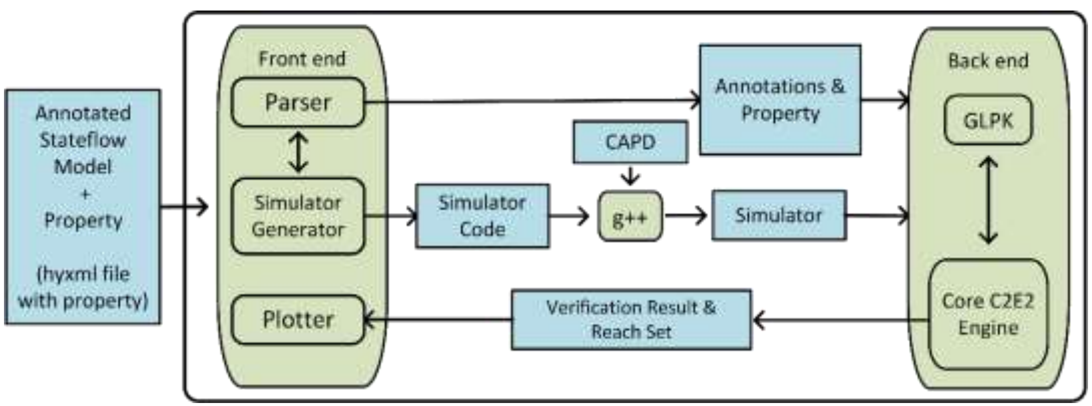
Theorem[Relative Completeness]: Given any HA A, with an initial set Θ, and unsafe set U, if the system is robustly **safe (unsafe)** then the algorithm will terminates and return the correct answer

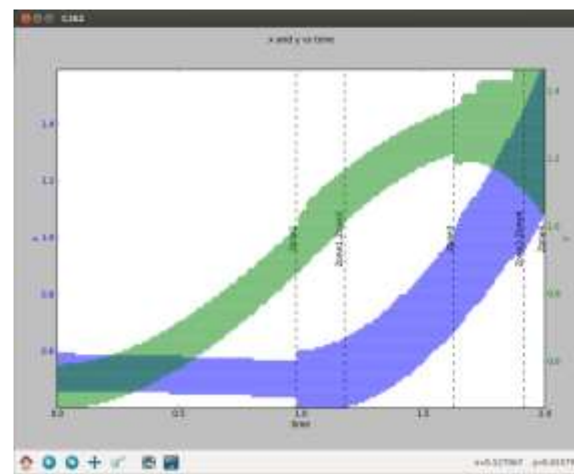
C2E2: A Tool For Verifying CPS Models











Powertrain Control Systems

- Fuel control and transmission subsystem
  - Software control: increasing complexity (100M LOC)
  - Constraints: Emissions, Efficiency, etc.
  - Strict performance requirements
  - Early bug detection using formal methods

Powertrain Control Systems

- Fuel control and transmission subsystem
  - Software control: increasing complexity (100M LOC)
  - Constraints: Emissions, Efficiency, etc.
  - Strict performance requirements
  - Early bug detection using formal methods

- Powertrain control benchmarks from Toyota Jin et.al. [HSCC'14]
- Complexity "*similar*" to industrial systems
- Benchmark tool/challenge problems for academic research

D, Fan, Mitra, Viswanathan CAV 2015 Fan, **D**, Mitra, Viswanathan ARCH 2015

Challenge Problem: Verifying one of the models in the powertrain control benchmark

Verifying Powertrain Control System
(Challenges)

Hybrid Systems Model
Polynomial ODE Plant
+
Modes of operation

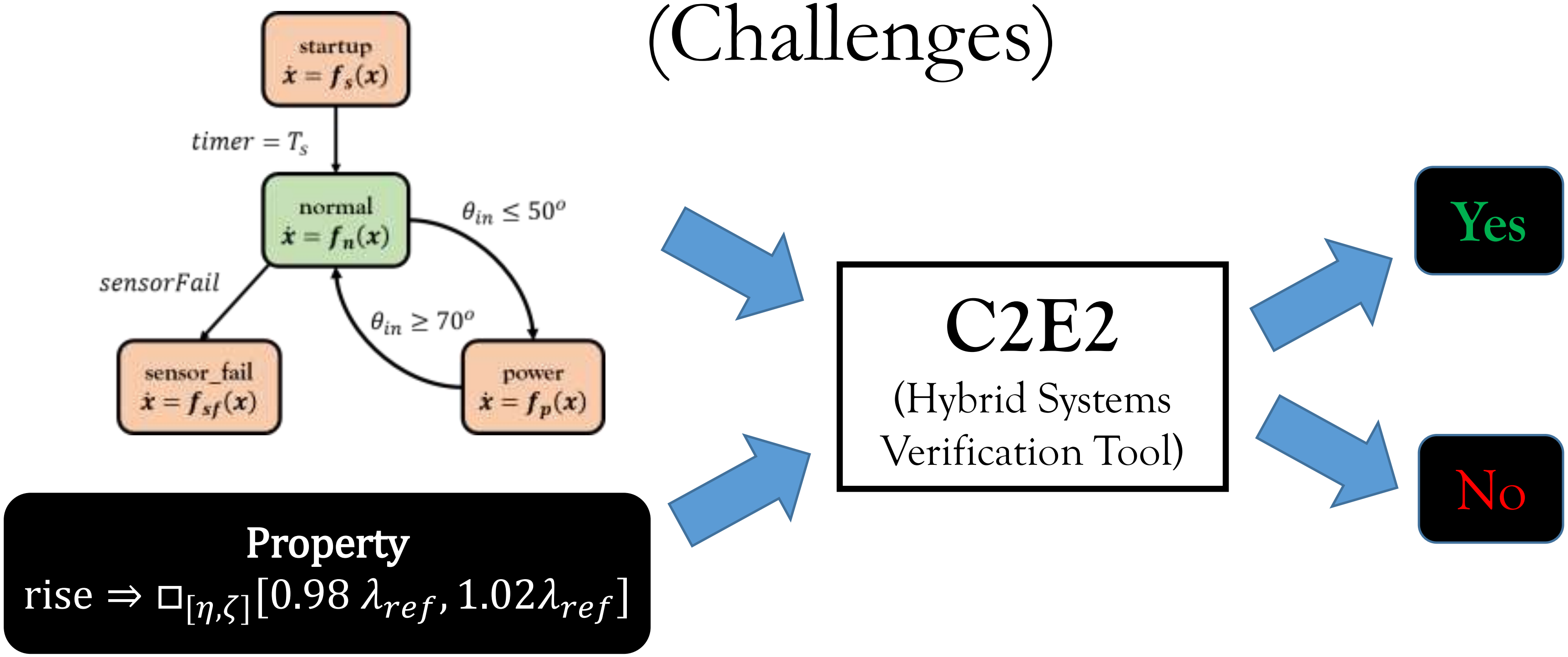
Property
rise $\Rightarrow \square_{[\eta,\zeta]}[0.98\,\lambda_{ref}, 1.02\lambda_{ref}]$

C2E2
(Hybrid Systems Verification Tool)

Yes

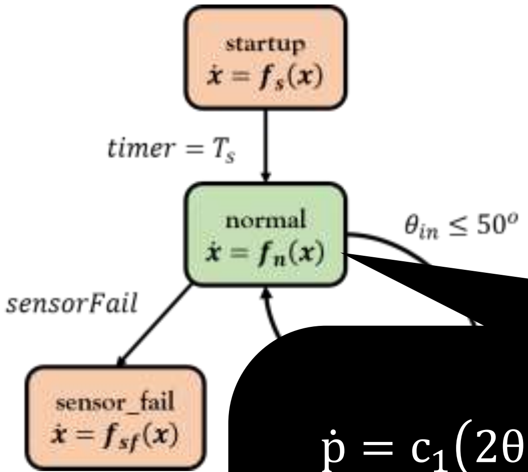
No

Verifying Powertrain Control System (Challenges)



- Mix of discrete and continuous behaviors.

Verifying Powertrain Control System
(Challenges)



Pro
rise $\Rightarrow \square_{[\eta,\zeta]}[0.$

$$\dot{p} = c_1\big(2\theta(c_{20}p^2 + c_{21}p + c_{22}) - c_{12}(c_2 + c_3\omega p + c_4\omega p^2 + c_5\omega p^2)\big)$$
$$\dot{\lambda} = c_{26}(c_{15} + c_{16}c_{25}F_c + c_{17}c_{25}^2F_c^2 + c_{18}\dot{m}_c + c_{19}\dot{m}_c c_{25}F_c - \lambda)$$
$$\dot{p_e} = c_1\big(2c_{23}\theta(c_{20}p^2 + c_{21}p + c_{22}) - (c_2 + c_3\omega p + c_4\omega p^2 + c_5\omega p^2)\big)$$
$$i = c_{14}(c_{24}\lambda - c_{11})$$

where

$$F_c = \frac{1}{c_{11}}(1 + i + c_{13}(c_{24}\lambda - c_{11}))(c_2 + c_3\omega p + c_4\omega p^2 + c_5\omega p^2)$$
$$\dot{m}_c = c_{12}(c_2 + c_3\omega p + c_4\omega p^2 + c_5\omega p^2)$$

- Mix of discrete and continuous behaviors.
- Nonlinear Ordinary Diff. Eqns. – scalability problems

Powertrain Verification Results

Verified many key specification for a given set of driver behaviors

Property	Mode	Sat	Sim.	Time
$\Box\,\lambda \in [0.8\lambda_{ref}, 1.2\lambda_{ref}]$	all modes	Yes	53	11m58s
$\Box\,\lambda \in [0.8\lambda_{ref}, 1.2\lambda_{ref}]$	startup	Yes	50	10m21s
$\Box\,\lambda \in [0.8\lambda_{ref}, 1.2\lambda_{ref}]$	normal	Yes	50	10m21s
$\Box\,\lambda \in [0.8\lambda_{ref}^{pwr}, 1.2\lambda_{ref}^{pwr}]$	power	Yes	53	11m12s
$\Box\,\lambda \in [0.8\lambda'_{ref}, 1.2\lambda'_{ref}]$	power	No	4	0m43s
$rise \Rightarrow \Box_{(\eta,\xi)}\lambda \in [0.98\,\lambda_{ref}, 1.02\lambda_{ref}]$	normal	Yes	50	10m15s
$(l = pwr) \Rightarrow \Box_{(\eta,\xi)}\lambda \in [0.95\,\lambda_{ref}, 1.05\lambda_{ref}]$	power	Yes	53	11m35s
$(l = pwr) \Rightarrow \Box_{(\eta/2,\xi)}\lambda \in [0.95\,\lambda_{ref}, 1.05\lambda_{ref}]$	power	No	4	0m45s

Safety properties

Performance properties

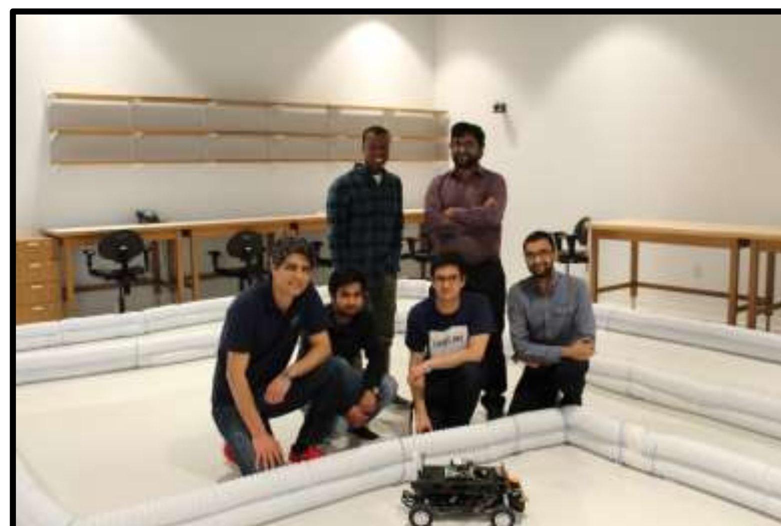
Won the 'Best Paper Award' at ARCH@CPSWeek 2015

Autonomous Vehicle Racing

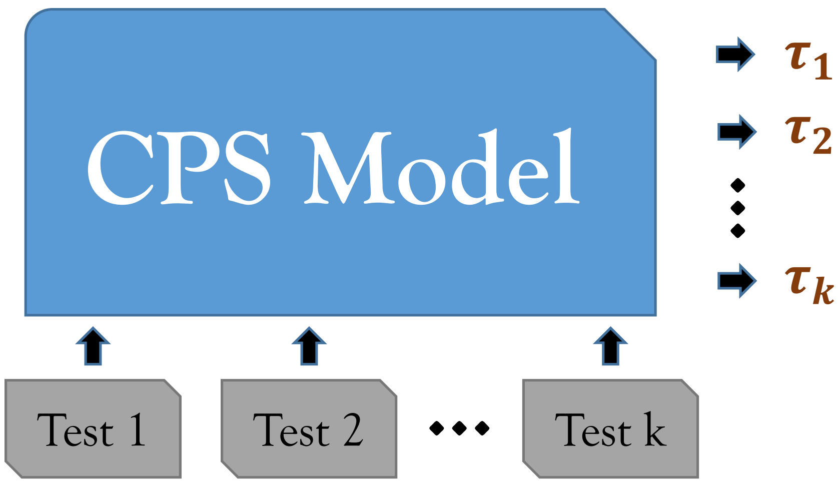
F1/10

Autonomous Vehicles
Racing Competition

CPSWeek (April 2018) – Placed 2^{nd}.
ESWeek (October 2018)– Placed 5^{th}.



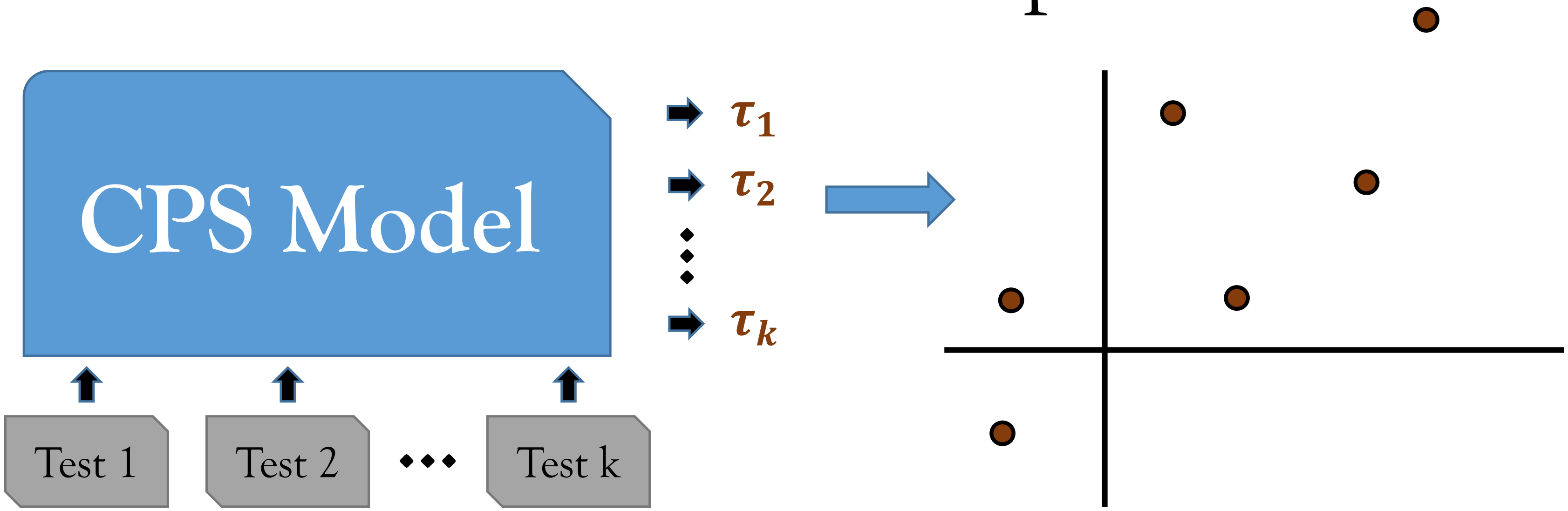
Embedding Trajectories into Lower Dimensional Spaces



- Trajectories of CPS are difficult to analyze because of 2 reasons.
    1. The state space itself is high-dimensional.
    2. Trajectories (functions of time) are infinite–dimensional artifacts.

D, Sheehy CCCG'18

Embedding Trajectories into Lower Dimensional Spaces



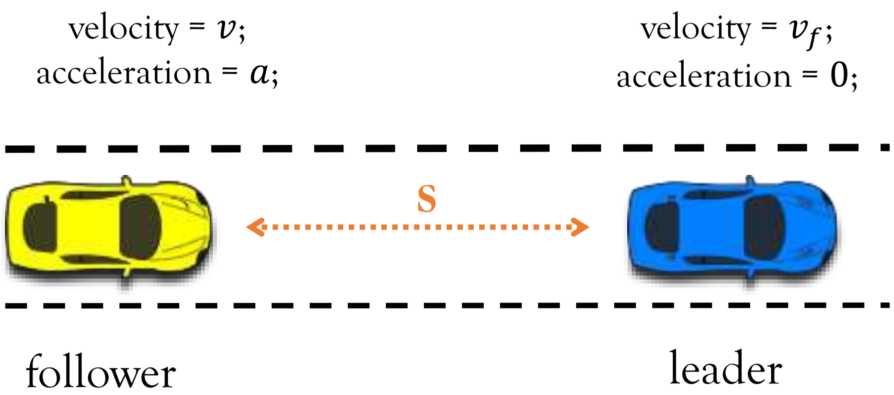
- Trajectories of CPS are difficult to analyze because of 2 reasons.
  1. The state space itself is high-dimensional.
  2. Trajectories (functions of time) are infinite–dimensional artifacts.
- How to reduce dimensionality and think of trajectories as points.
- Properties of embeddings (Lipschitz).
- Efficiency?

D, Sheehy CCCG'18

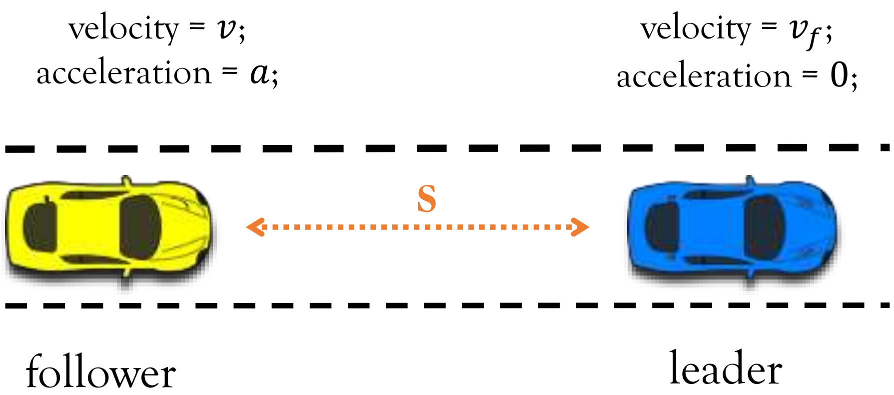
Scalable Verification of Linear Control Systems

D, Viswanathan CAV'16
Bak, **D** TACAS'17
Bak, **D** CAV'17
Bak, **D** ARCH@CPSWeek'17

Leader-Follower System



Leader-Follower System



Dynamics of the system

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

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

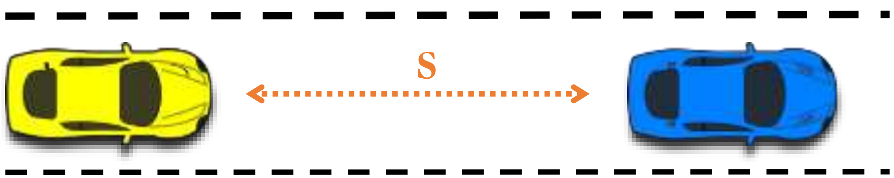
$\dot{a} = u;$

k_{aero} is the air–drag

Leader-Follower System

velocity = v;
acceleration = a;

velocity = v_f;
acceleration = 0;



follower

leader

Dynamics of the system

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

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

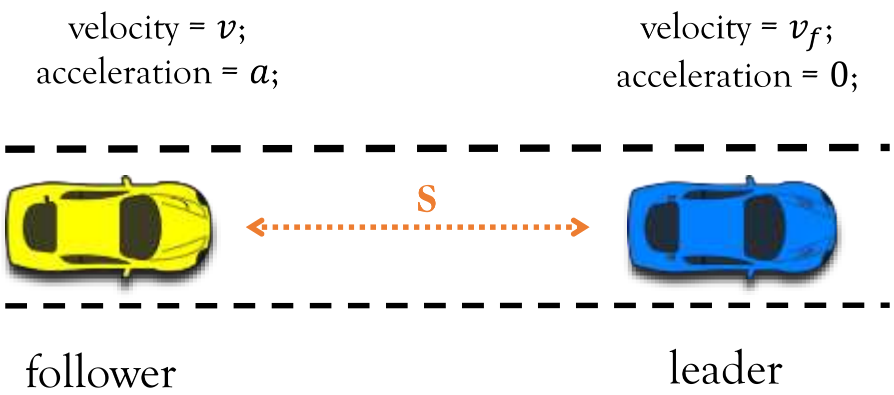
$\dot{a} = u;$

k_{aero} is the air–drag

Control Law

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

Leader-Follower System



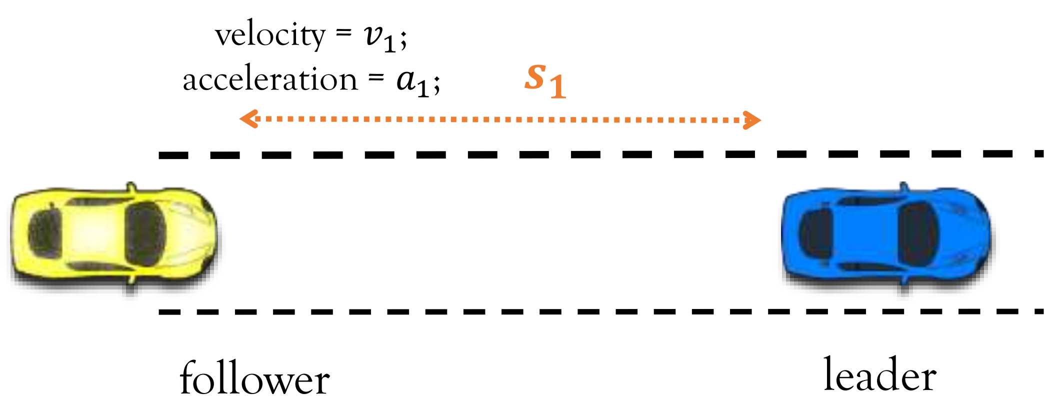
Dynamics of the system

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

k_{aero} is the air–drag

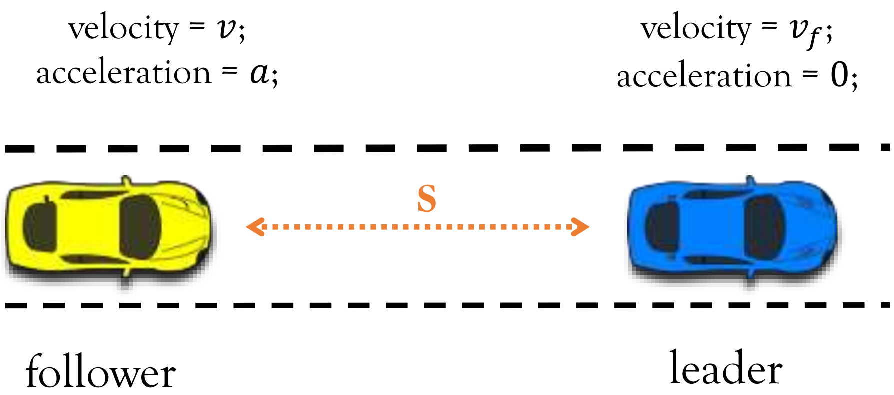
Control Law

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



Test scenario

Leader-Follower System



Dynamics of the system

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

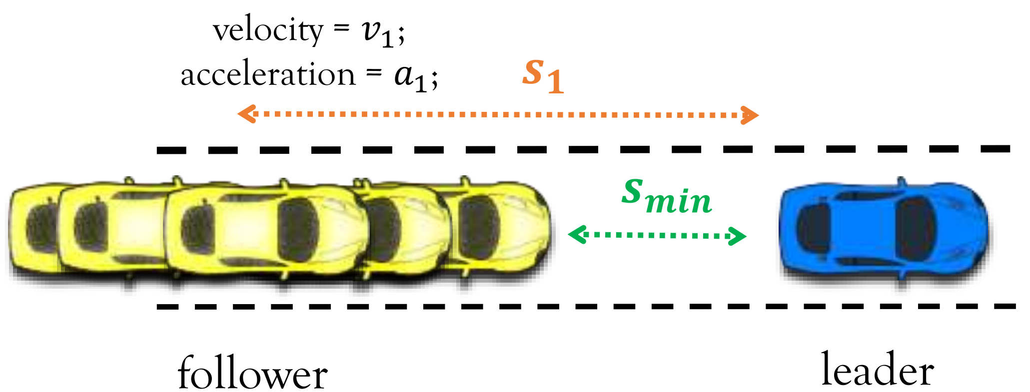
$\dot{v} = a - k_{aero} v;$

$\dot{a} = u;$

k_{aero} is the air-drag

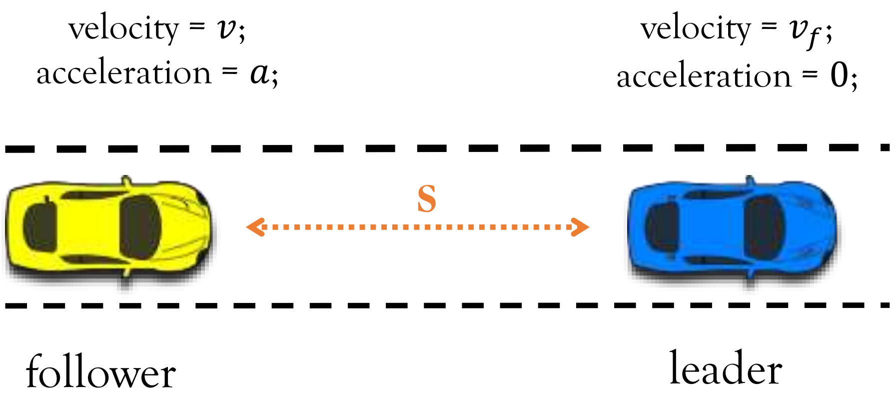
Control Law

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



Test scenario

Leader-Follower System



Dynamics of the system

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

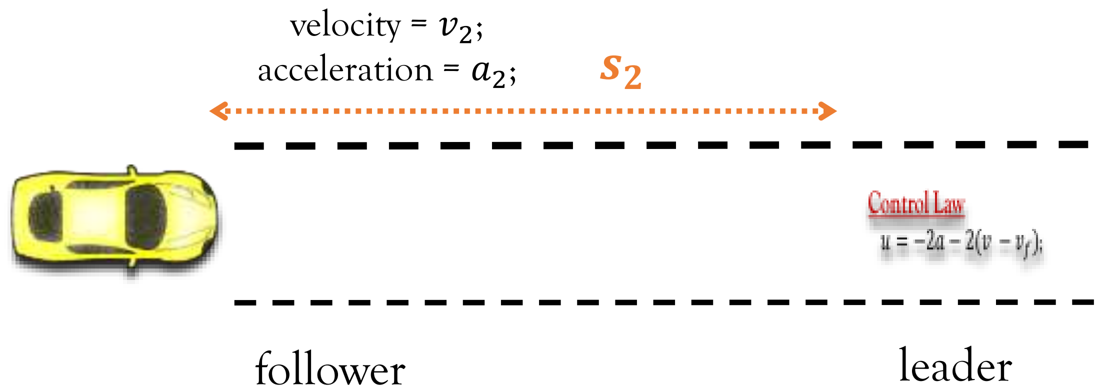
$\dot{v} = a - k_{aero} v;$

$\dot{a} = u;$

k_{aero} is the air-drag

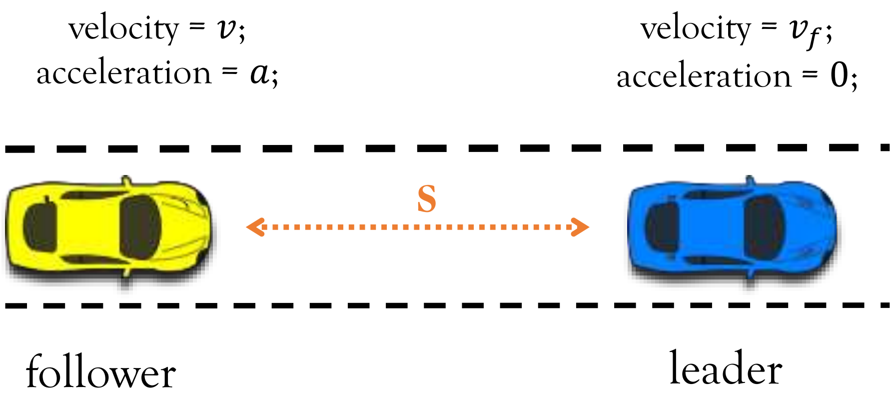
Control Law

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



Bad scenario?

Leader-Follower System



Dynamics of the system

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

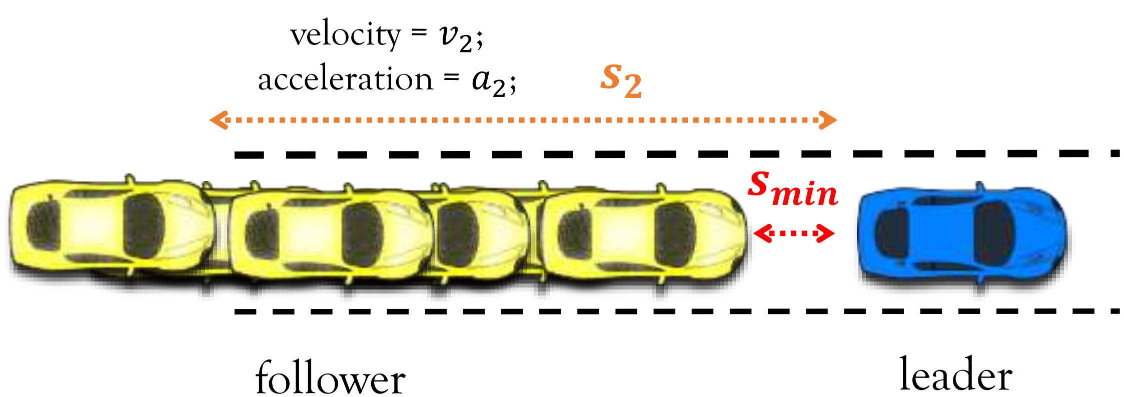
$\dot{v} = a - k_{aero} v;$

$\dot{a} = u;$

k_{aero} is the air–drag

Control Law

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



Bad scenario?

Safety Verification Problem

Given a Linear System $\dot{x} = Ax$, with initial set Θ and unsafe set U, are all the behaviors starting from Θ for bounded time T_b are safe?

Dynamics of the system

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

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

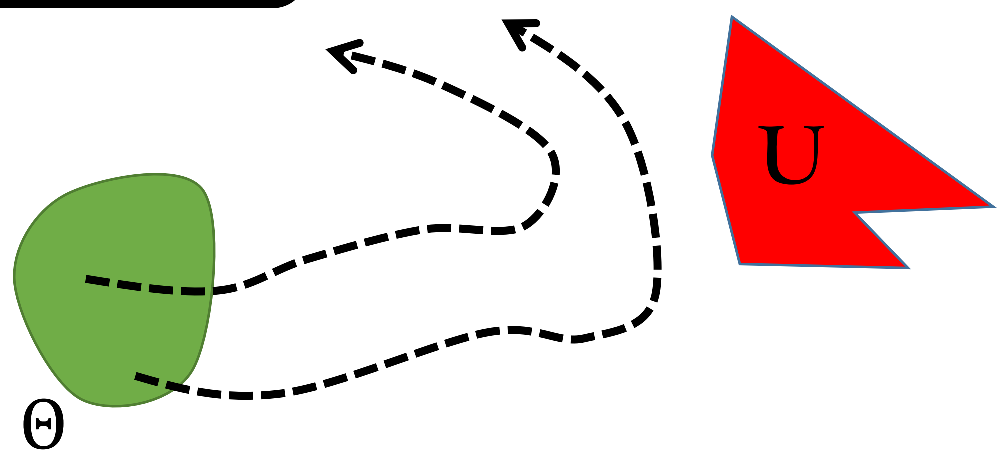
$\dot{a} = u;$

k_{aero} is the air-drag

Control Law

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

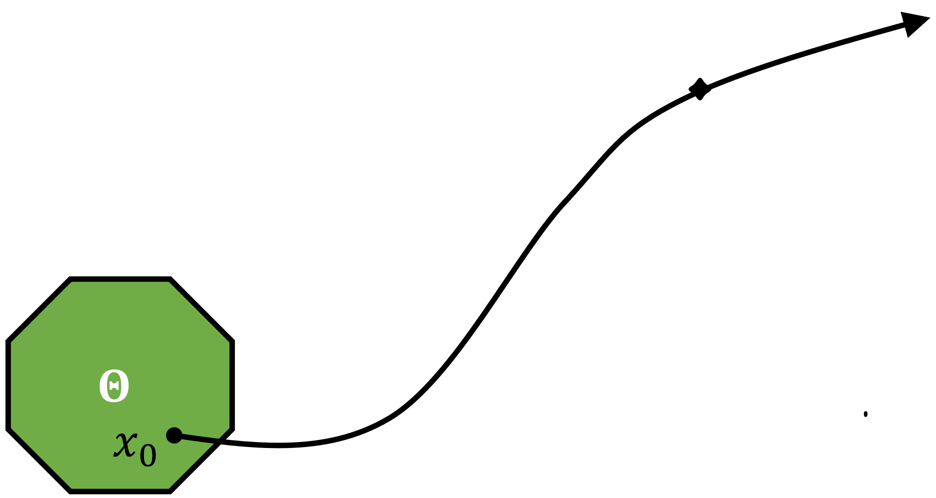
$$\triangleq \begin{bmatrix} \dot{s} \\ \dot{v} \\ \dot{a} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & -k_{aero} & 1 \\ 0 & -2 & -2 \end{bmatrix} \begin{bmatrix} s \\ v \\ a \end{bmatrix} + \begin{bmatrix} v_f \\ 0 \\ 2v_f \end{bmatrix}$$



Solution: Reachable Set

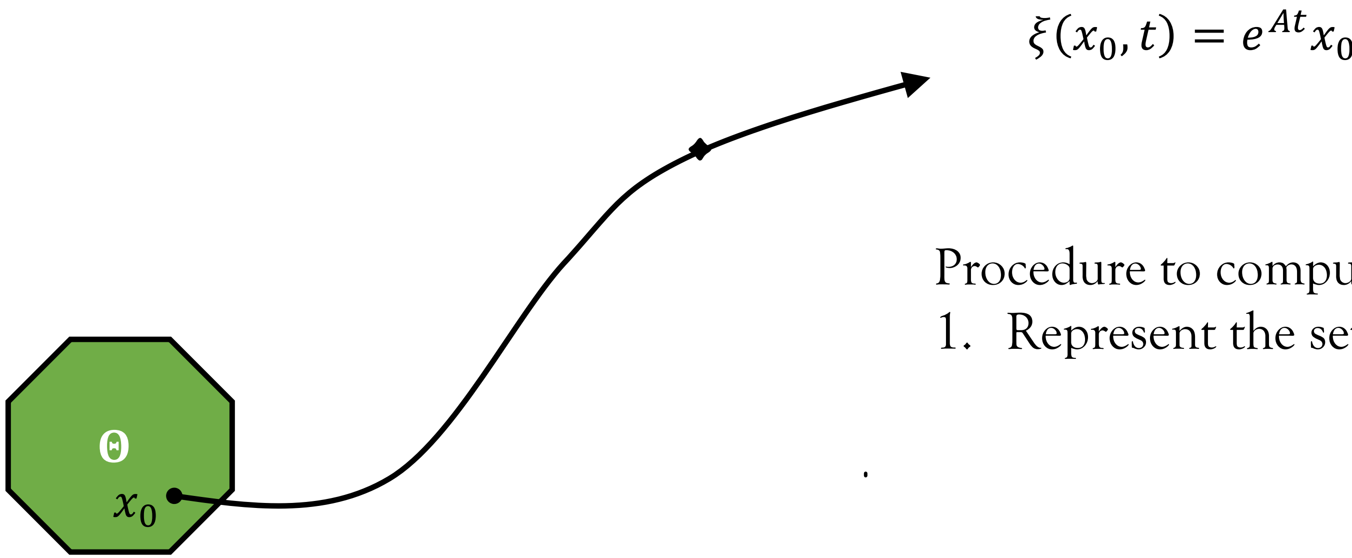
System: $\dot{x} = Ax$, initial set Θ (polyhedra), unsafe set U.

$\xi(x_0, t) = e^{At}x_0$



Solution: Reachable Set

System: $\dot{x} = Ax$, initial set Θ (polyhedra), unsafe set U.



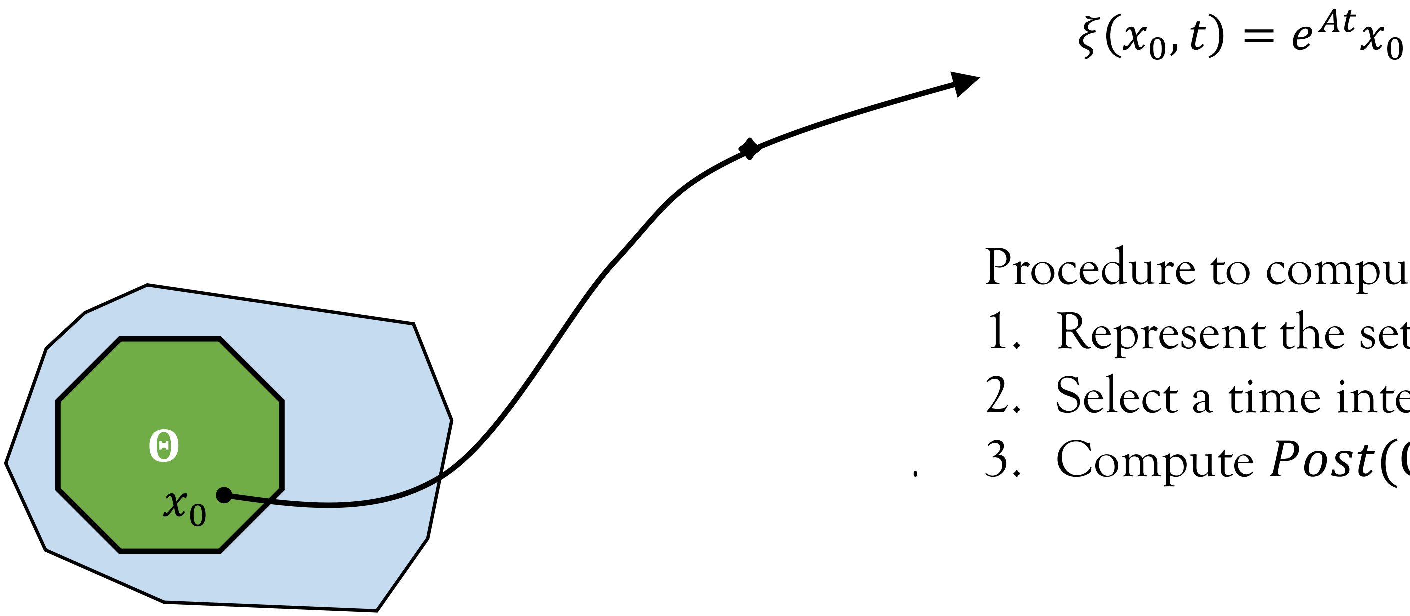
Procedure to compute reachable set

1. Represent the set Θ using data structure

Data structure
SpaceEx – Support Functions
CORA – Zonotopes
Flow* – Taylor Models

Solution: Reachable Set

System: $\dot{x} = Ax$, initial set Θ (polyhedra), unsafe set U.



Procedure to compute reachable set

1. Represent the set Θ using data structure
2. Select a time interval h.
3. Compute $Post(\Theta, h)$ for $[0, h]$

Data structure

SpaceEx – Support Functions

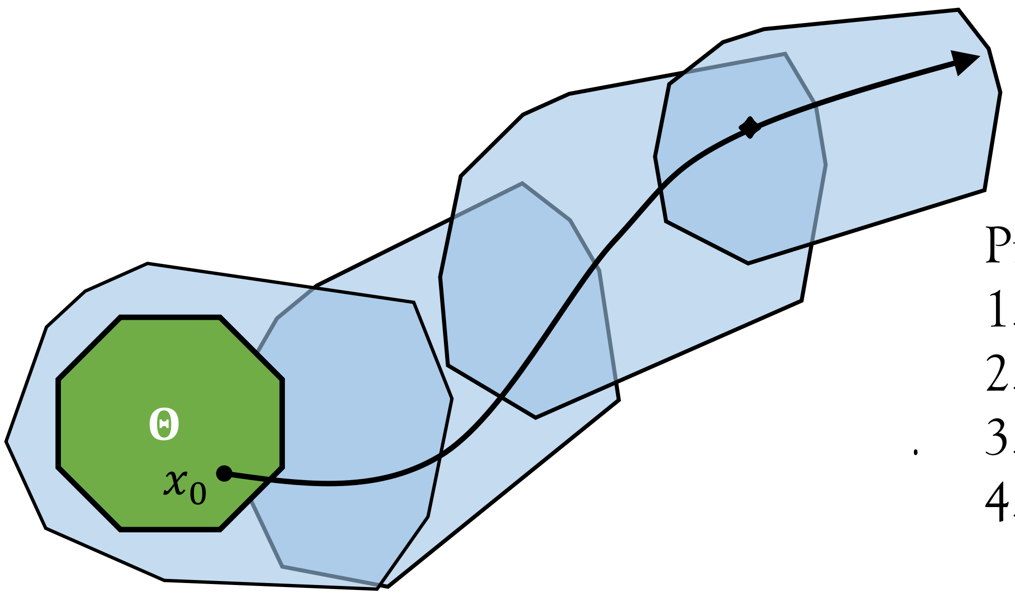
CORA – Zonotopes

Flow* – Taylor Models

Solution: Reachable Set

System: $\dot{x} = Ax$, initial set Θ (polyhedra), unsafe set U.

$\xi(x_0, t) = e^{At} x_0$



Procedure to compute reachable set

1. Represent the set Θ using data structure
2. Select a time interval h.
3. Compute $Post(\Theta, h)$ for $[0, h]$
4. Iterate for future intervals.

Data structure

SpaceEx – Support Functions

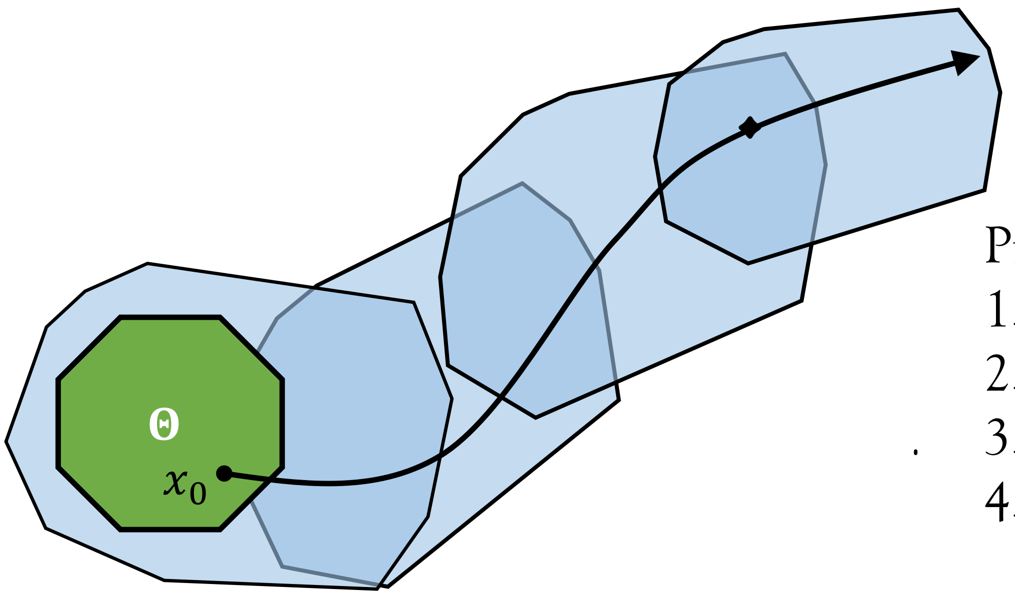
CORA – Zonotopes

Flow* – Taylor Models

Solution: Reachable Set

System: $\dot{x} = Ax$, initial set Θ (polyhedra), unsafe set U.

$$\xi(x_0, t) = e^{At} x_0$$



Procedure to compute reachable set

1. Represent the set Θ using data structure
2. Select a time interval h.
3. Compute $Post(\Theta, h)$ for $[0, h]$
4. Iterate for future intervals.

Data structure

SpaceEx – Support Functions

CORA – Zonotopes

Flow* – Taylor Models

Drawbacks

1. Representation cost grows with n
2. Cannot be directly applied for time varying linear systems
3. When set changes, entire computation needs to be done

Background

Setup:

- Initial set: $Hx \leq g$ (bounded polyhedron).
- Unsafe set: $Qx \leq r$ (conjunction of half-spaces).

- Initial attempts (2002): Uses vertices of polyhedral - $O(2^n)$.
- Second attempt (2008): Uses support functions - $O(k \times n^2)$

- [D., Viswanathan] (2015): Uses sparse matrix multiplication.

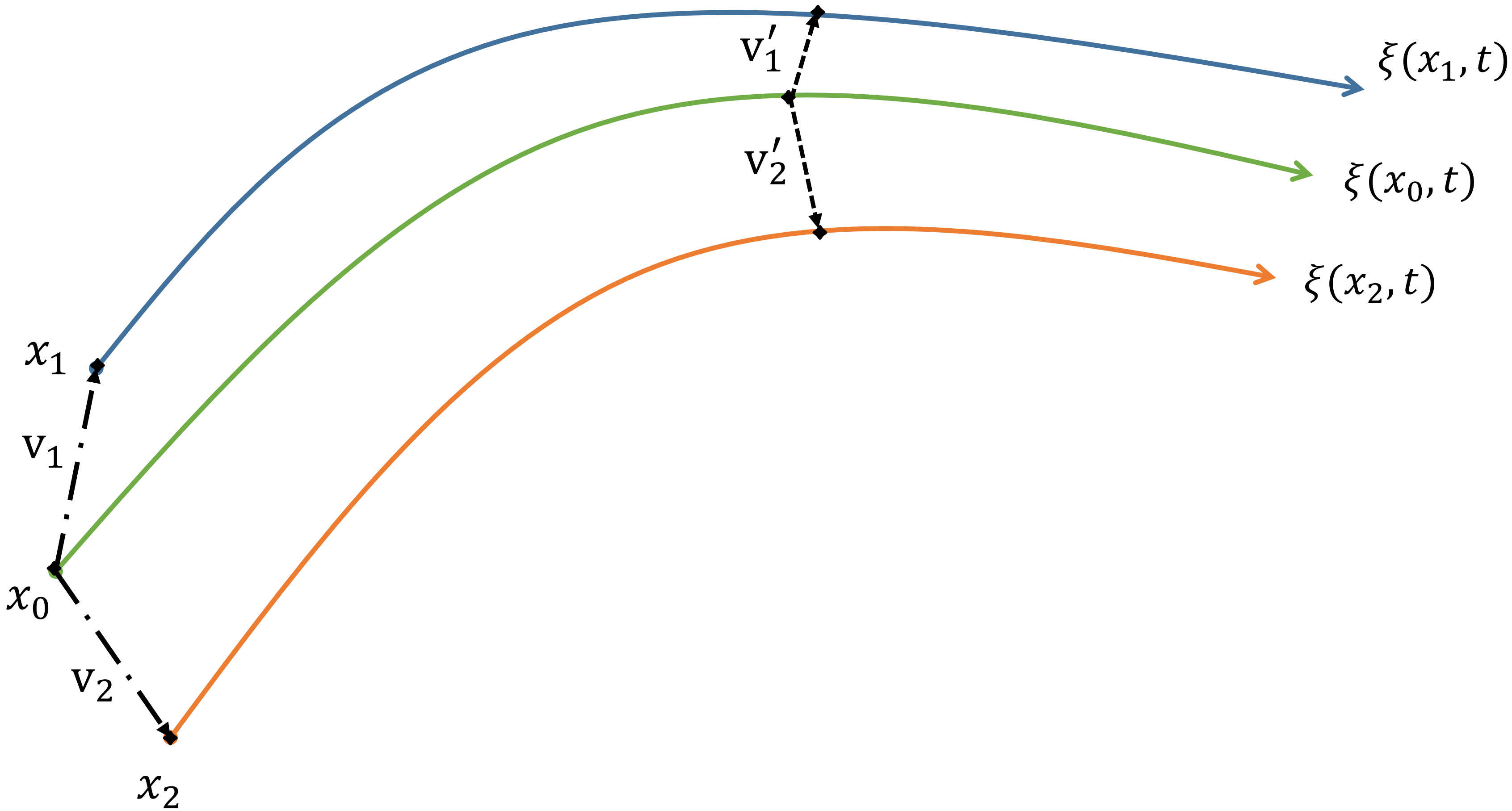
Main Insight

Dynamics $\dot{x} = Ax$ has nice properties.

Why not develop representations that leverage these properties!

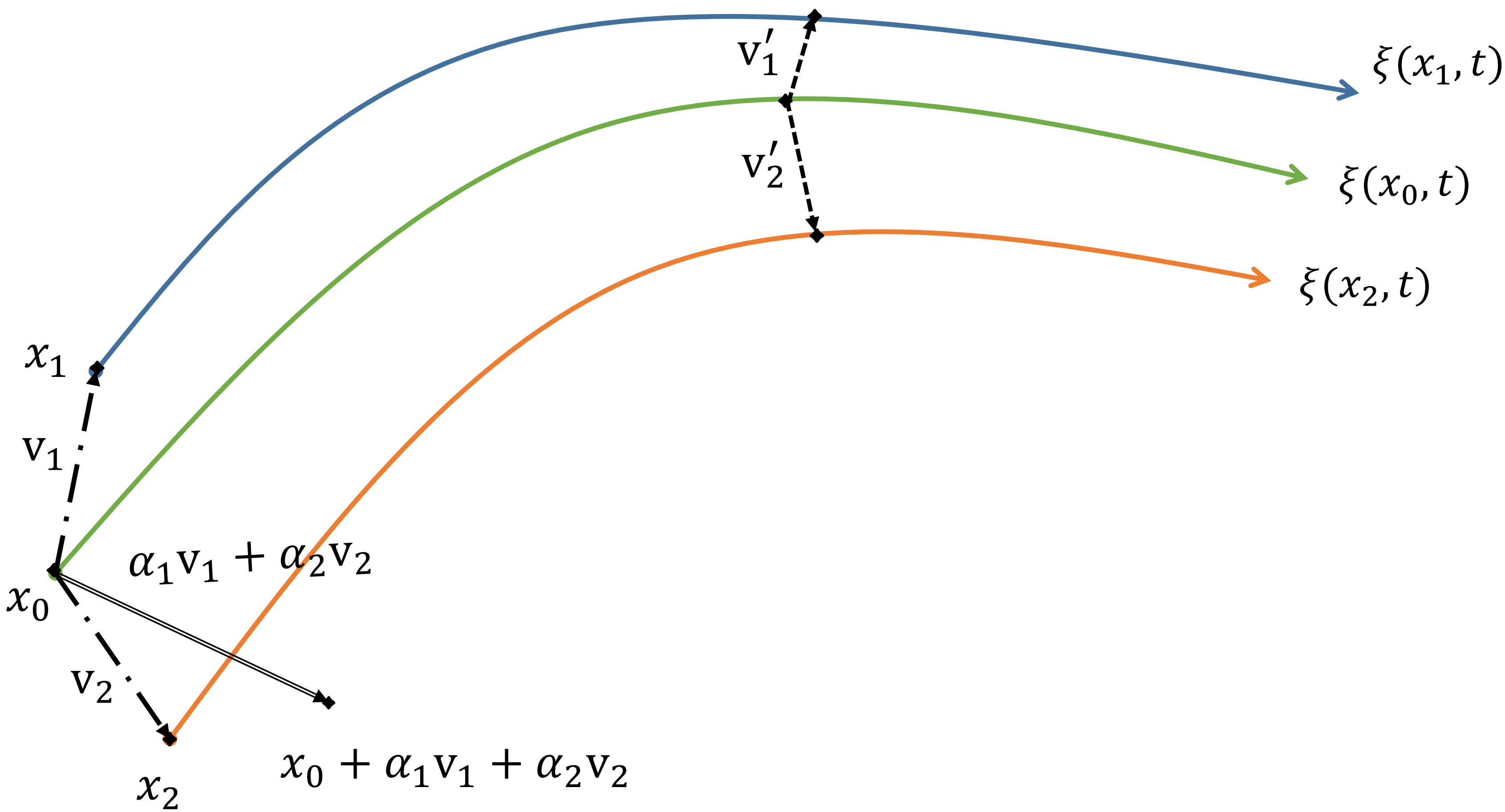
Property: Superposition

The trajectories form a vector space!



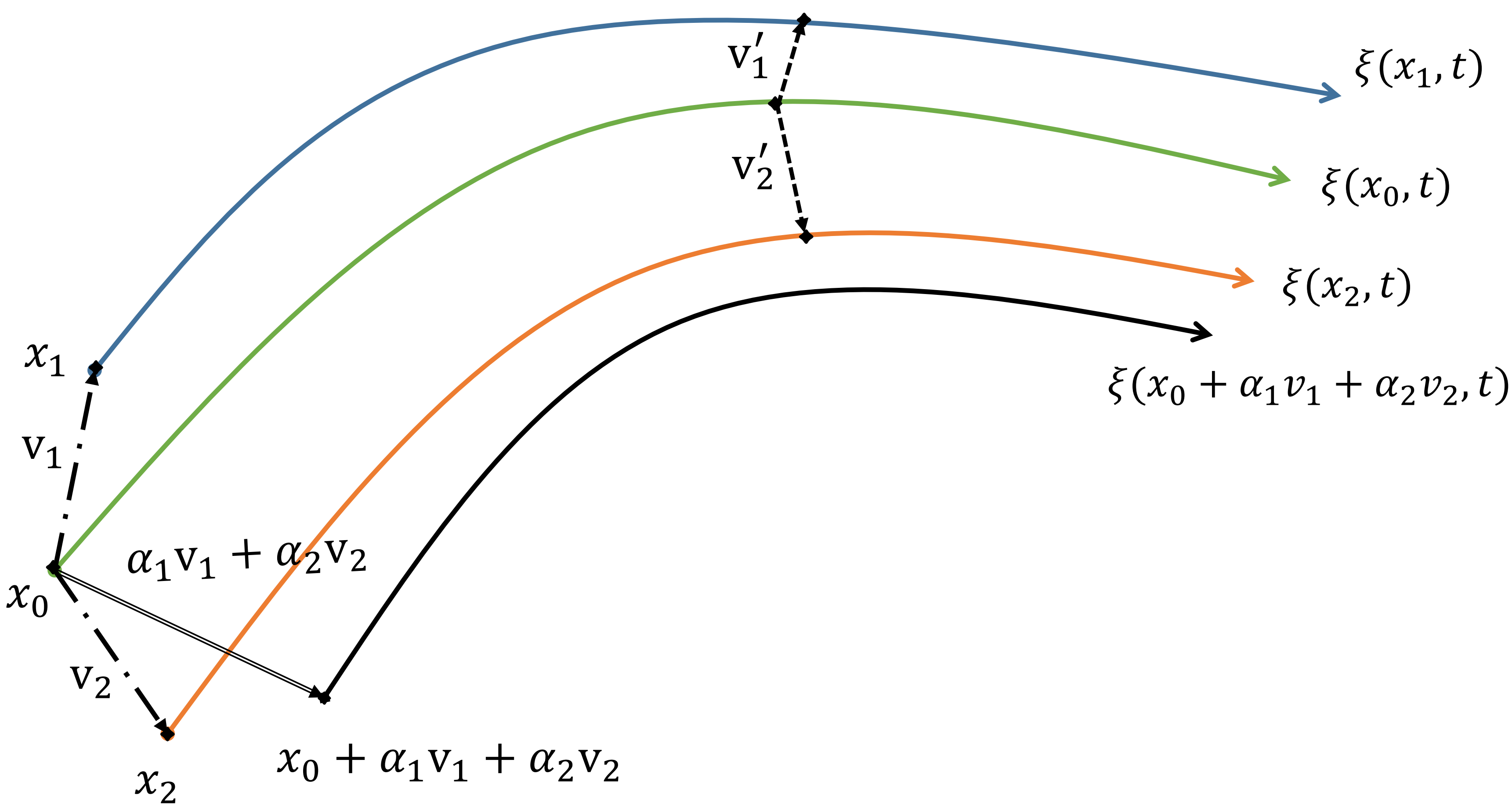
Property: Superposition

<u>The trajectories form a vector space!</u>



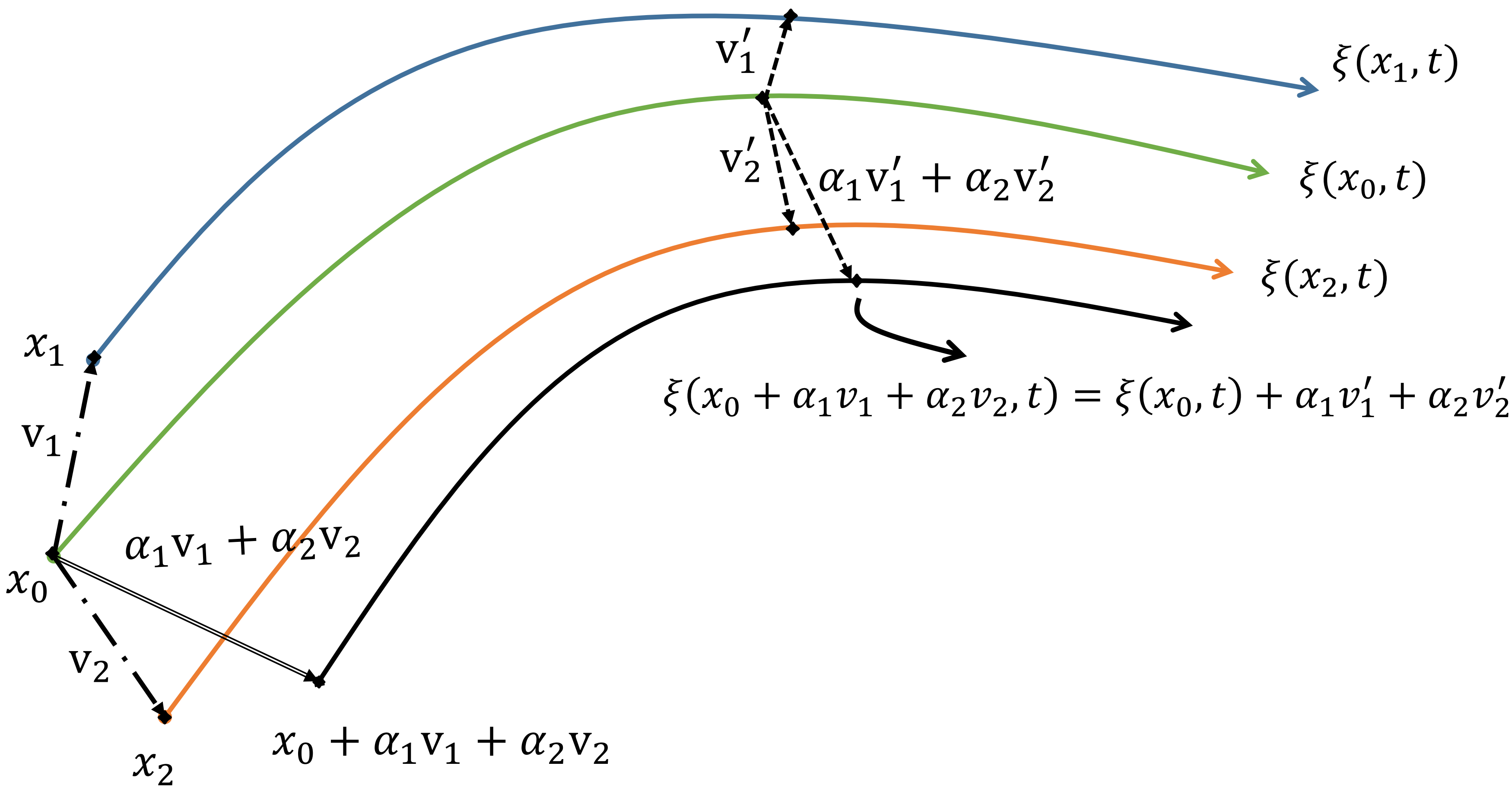
Property: Superposition

The trajectories form a vector space!



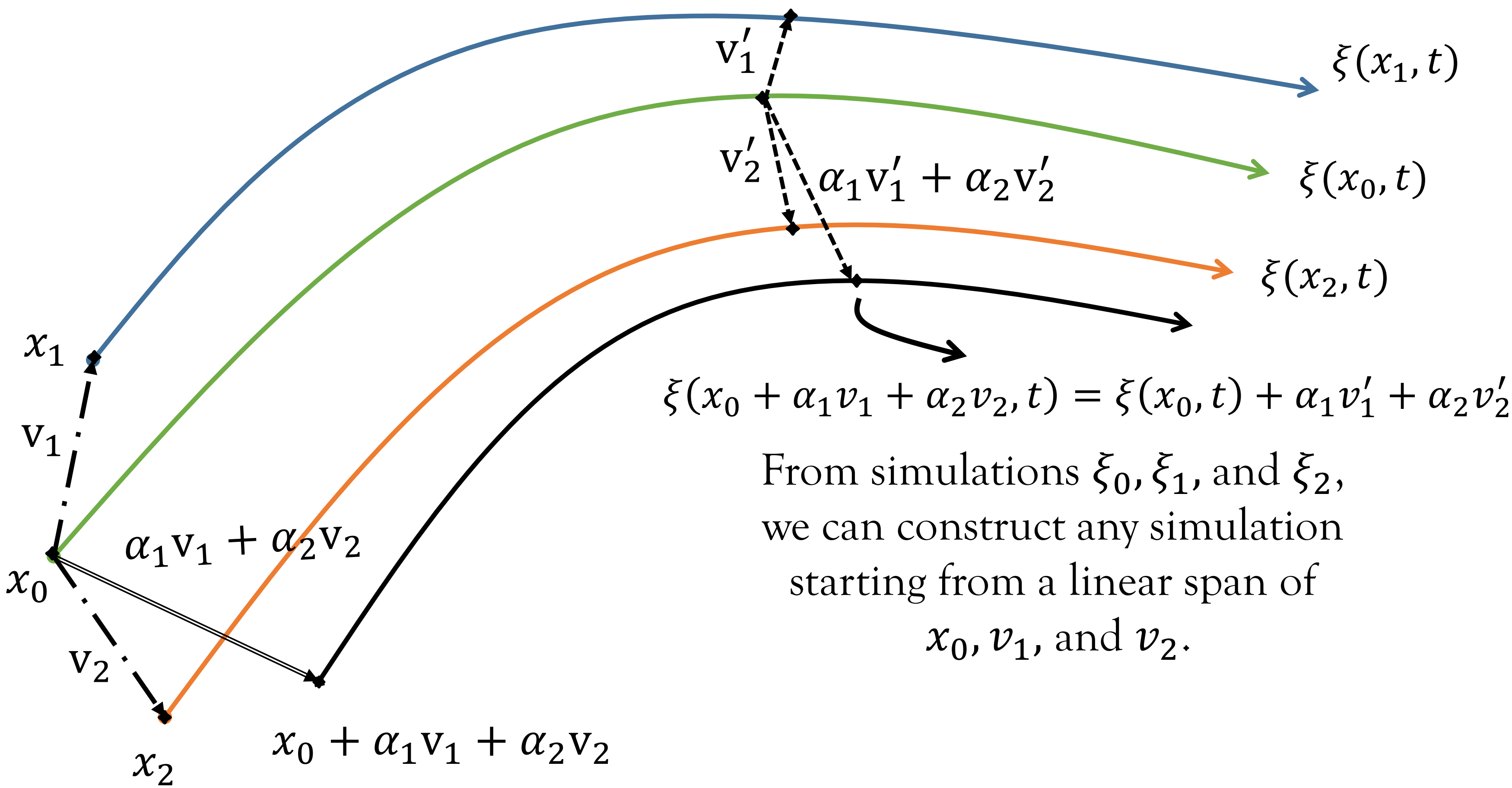
Property: Superposition

<u>The trajectories form a vector space!</u>



Property: Superposition

The trajectories form a vector space!

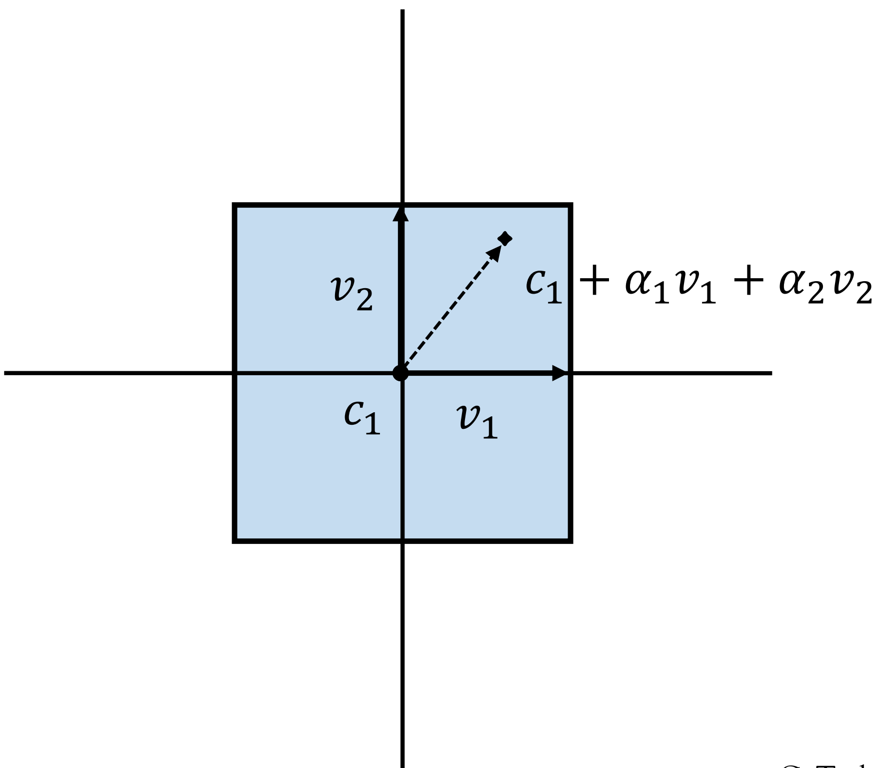


From simulations ξ_0, ξ_1, and ξ_2, we can construct any simulation starting from a linear span of x_0, v_1, and v_2.

Representation: Generalized Stars

- Generalized star is represented as $\langle c, V, P \rangle$
- c – center, V – set of vectors, P – predicate.

$$\langle c, V, P \rangle = \{\, x \mid \exists \bar{\alpha} = (\alpha_1, \dots, \alpha_n), \mathrm{c} + \Sigma_i \alpha_i v_i = x, P(\bar{\alpha}) = \top\}$$

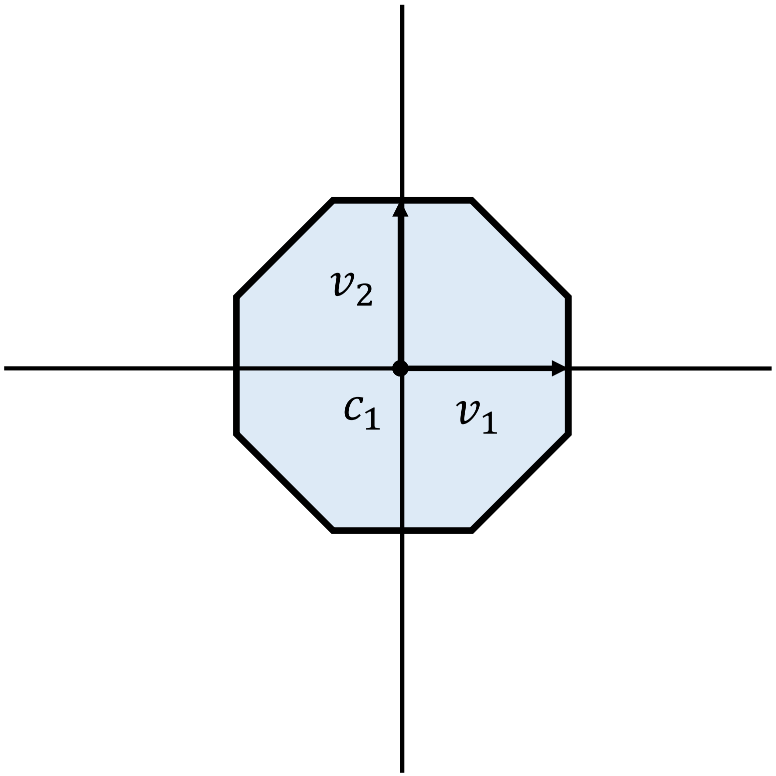


$$P(\langle \alpha_1, \alpha_2 \rangle) \triangleq |\alpha_1| \leq 1 \wedge |\alpha_2| \leq 1$$

Representation: Generalized Stars

- Generalized star is represented as $\langle c, V, P\rangle$
- c – center, V – set of vectors, P – predicate.

$$\langle c, V, P\rangle = \{\, x \mid \exists\bar{\alpha} = (\alpha_1, \dots, \alpha_n), \mathrm{c} + \Sigma_i \alpha_i v_i = x, P(\bar{\alpha}) = \top\}$$

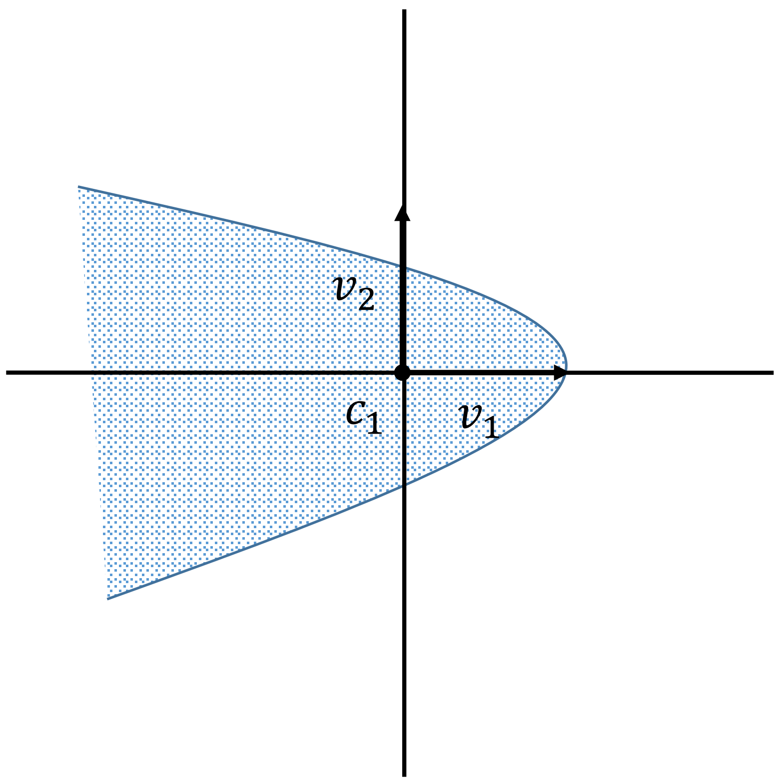


$$P(\langle\alpha_1, \alpha_2\rangle) \triangleq |\alpha_1| \le 1 \wedge |\alpha_2| \le 1 \wedge |\alpha_1 + \alpha_2| \le 1.5$$

Representation: Generalized Stars

- Generalized star is represented as $\langle c, V, P \rangle$
- c – center, V – set of vectors, P – predicate.

$$\langle c, V, P \rangle = \{\, x \mid \exists \bar{\alpha} = (\alpha_1, \dots, \alpha_n), \mathrm{c} + \Sigma_i \alpha_i v_i = x, P(\bar{\alpha}) = \top \}$$

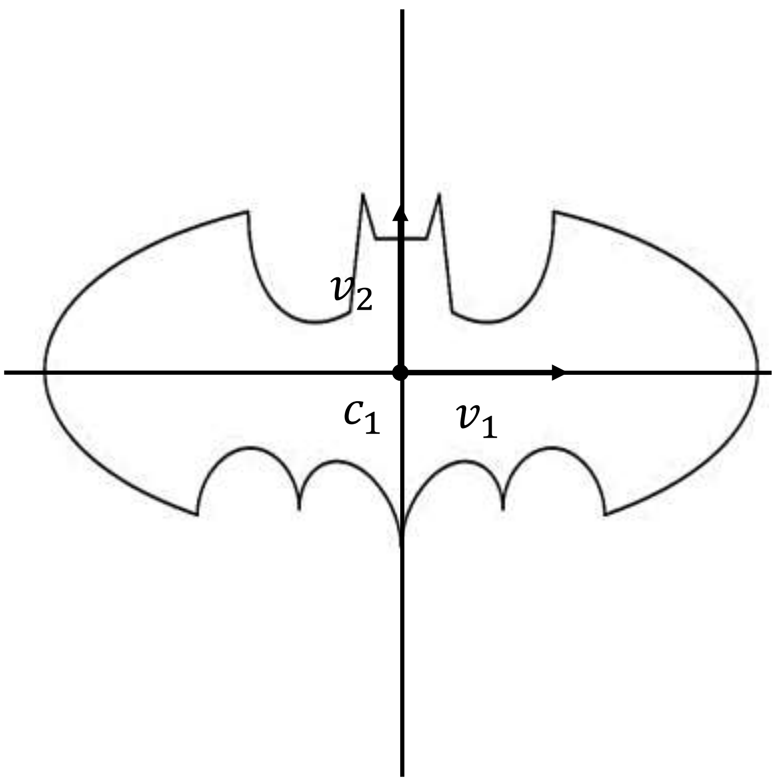


$$P(\langle \alpha_1, \alpha_2 \rangle) \triangleq \alpha_1 \leq 1 - \alpha_2^2$$

Representation: Generalized Stars

- Generalized star is represented as $\langle c, V, P \rangle$
- c – center, V – set of vectors, P – predicate.

$$\langle c, V, P \rangle = \{\, x \mid \exists \bar{\alpha} = (\alpha_1, \ldots, \alpha_n), \mathrm{c} + \Sigma_i \alpha_i v_i = x, P(\bar{\alpha}) = \top \}$$

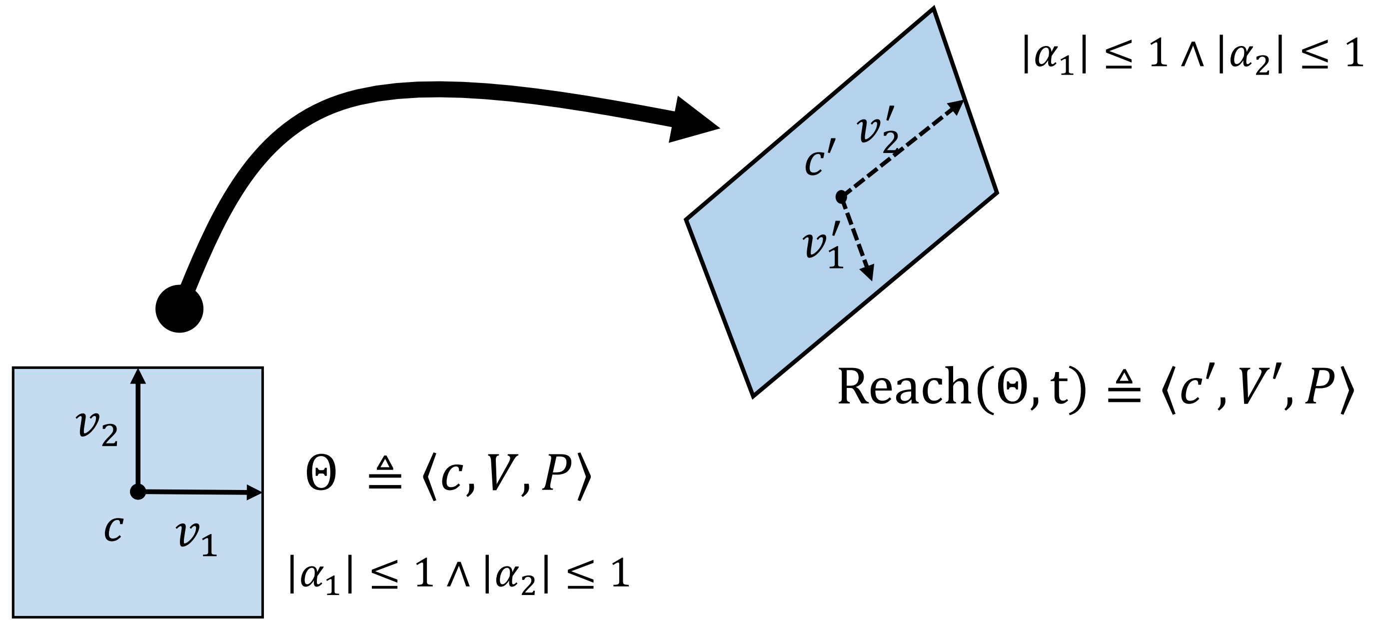


$$P(\langle \alpha_1, \alpha_2 \rangle) \triangleq$$

$$1.5 * sqrt\left((\text{-}abs(abs(x) - 1)) * \frac{abs(3 - abs(x))}{(abs(x) - 1) * (3 - abs(x))} \right) * \left(1 + \frac{abs(abs(x) - 3)}{abs(x) - 3} \right) * sqrt\left(1 - \left(\frac{x}{7}\right)^2 \right) +$$
$$\left(4.5 + 0.75 * (abs(x - 0.5) + abs(x + 0.5)) - 2.75 * (abs(x - 0.75) + abs(x + 0.75)) \right) * \left(1 + \frac{abs(1 - abs(x))}{1 - abs(x)}$$
$$(\text{-}3) * sqrt\left(1 - \left(\frac{x}{7}\right)^2 \right) * sqrt\left(\frac{abs(abs(x) - 4)}{abs(x) - 4} \right), abs\left(\frac{x}{2}\right) - 0.0913722 * x^2 - 3 + sqrt(1 - (abs(abs(x) - 2) - 1)^2)$$
$$(2.71052 + 1.5 - 0.5 * abs(x) - 1.35526 * sqrt(4 - (abs(x) - 1)\hat{}2)) * sqrt(abs(abs(x) - 1)/(abs(x) - 1))$$

Technique: Basic Idea

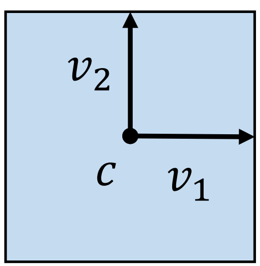
- Given initial set $\Theta = \langle c, V, P \rangle$, the **Reach** is computed not as new predicate, but is done by changing the *center* and the *basis* vectors.



D, Viswanathan CAV'16

Technique
Representation + Superposition

Given $\Theta \triangleq \langle c, V, P \rangle$ to compute reachable set



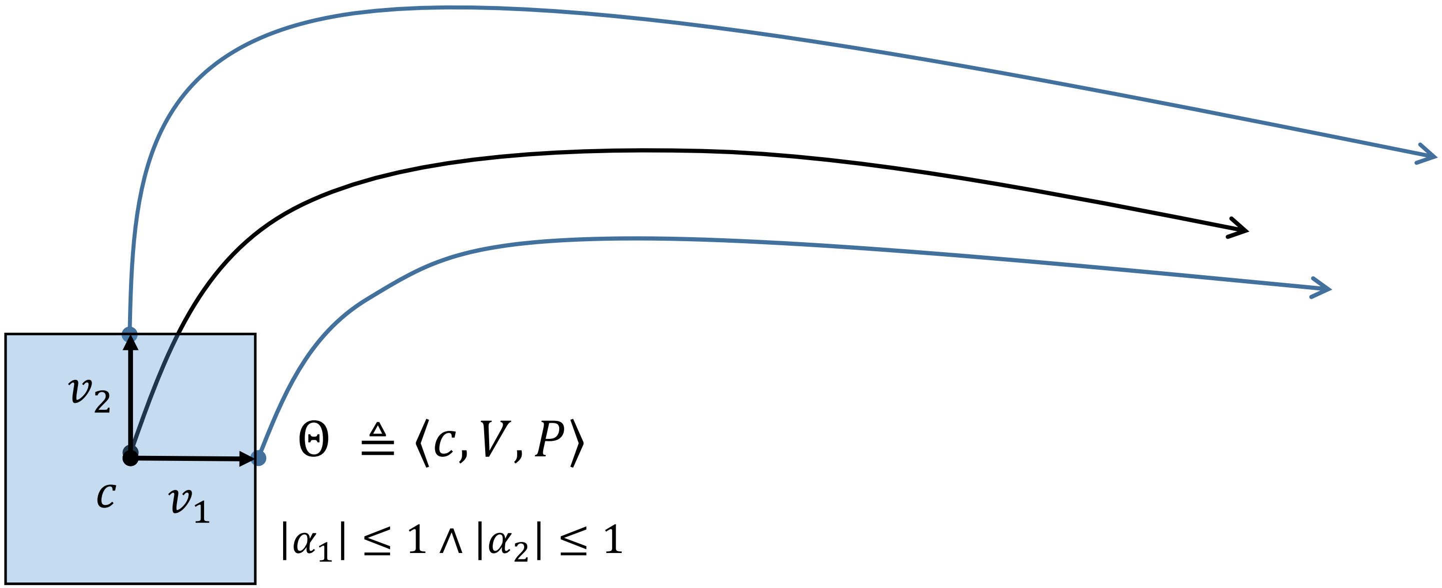
$\Theta \triangleq \langle c, V, P \rangle$

$|\alpha_1| \leq 1 \wedge |\alpha_2| \leq 1$

Technique
Representation + Superposition

Given $\Theta \triangleq \langle c, V, P \rangle$ to compute reachable set

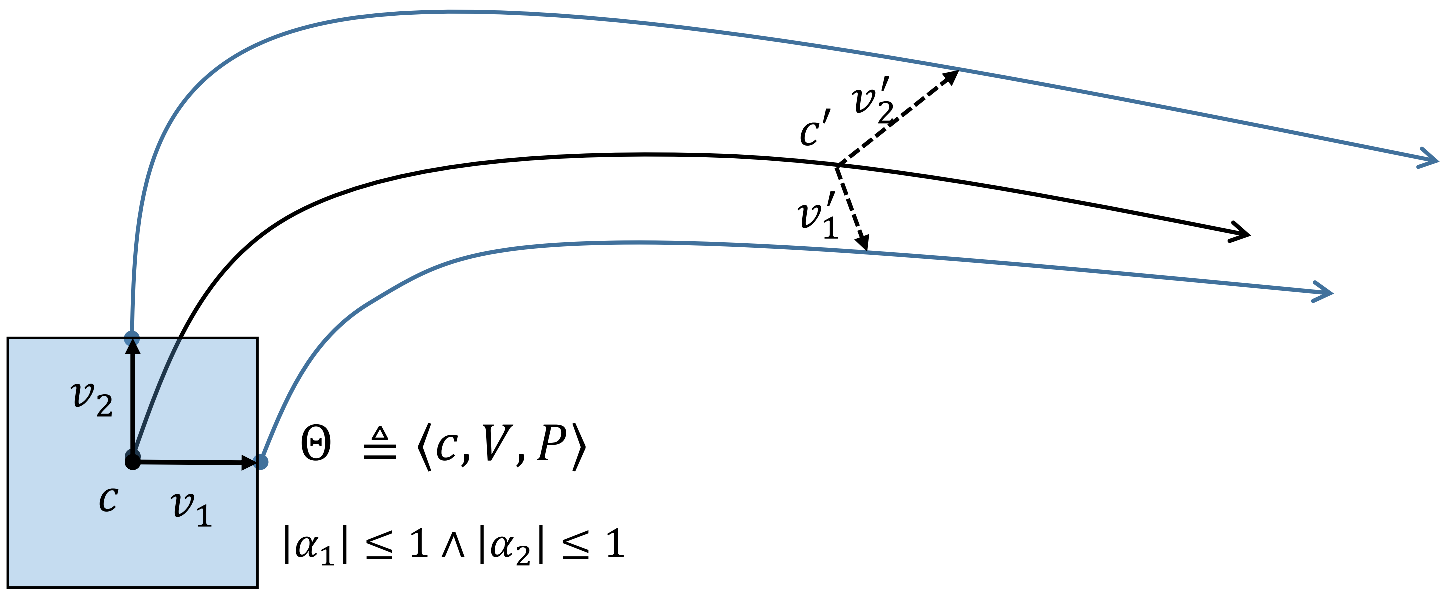
1. Simulate from c
2. Simulate from $c + v_i$ for each i



Technique
Representation + Superposition

Given $\Theta \triangleq \langle c, V, P \rangle$ to compute reachable set

1. Simulate from c
2. Simulate from $c + v_i$ for each i



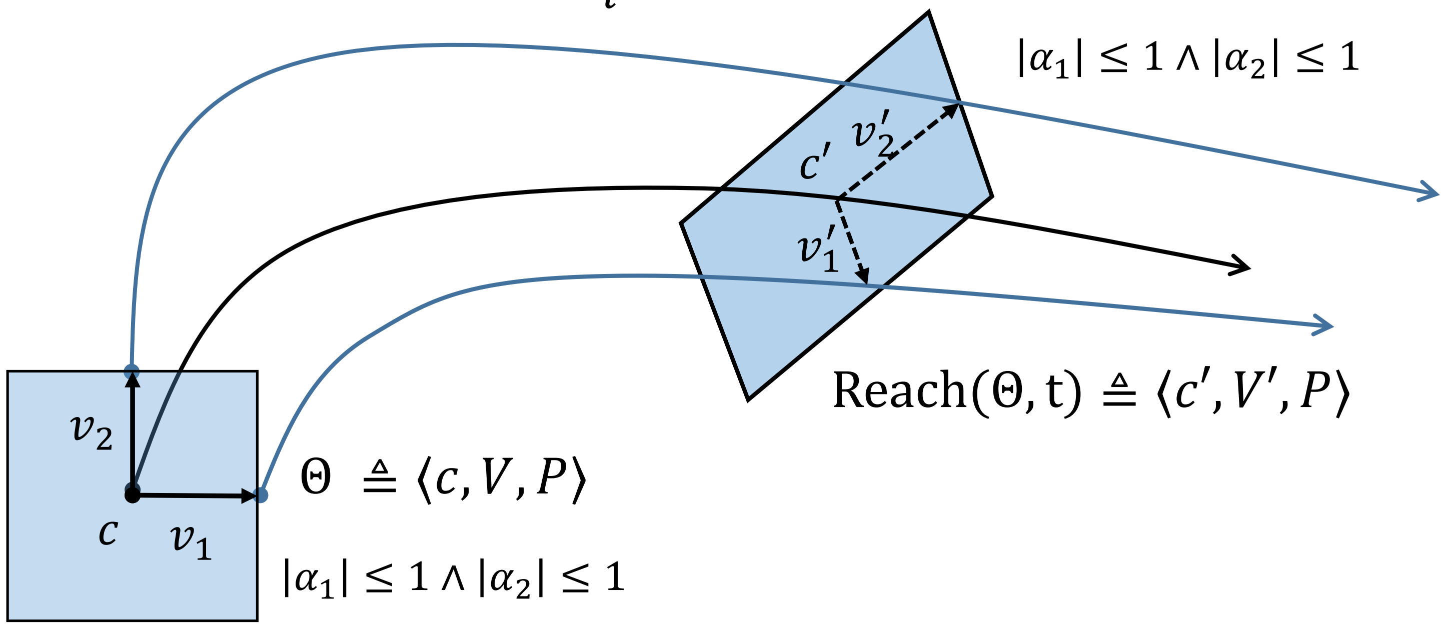
Reachable set at time t is given by $\langle c', V', P \rangle$ where

1. c' is the simulation corresponding to c
2. ${v_i}'$ is the difference of simulations from $c + v_i$ and from c

Technique
Representation + Superposition

Given $\Theta \triangleq \langle c, V, P \rangle$ to compute reachable set

1. Simulate from c
2. Simulate from $c + v_i$ for each i



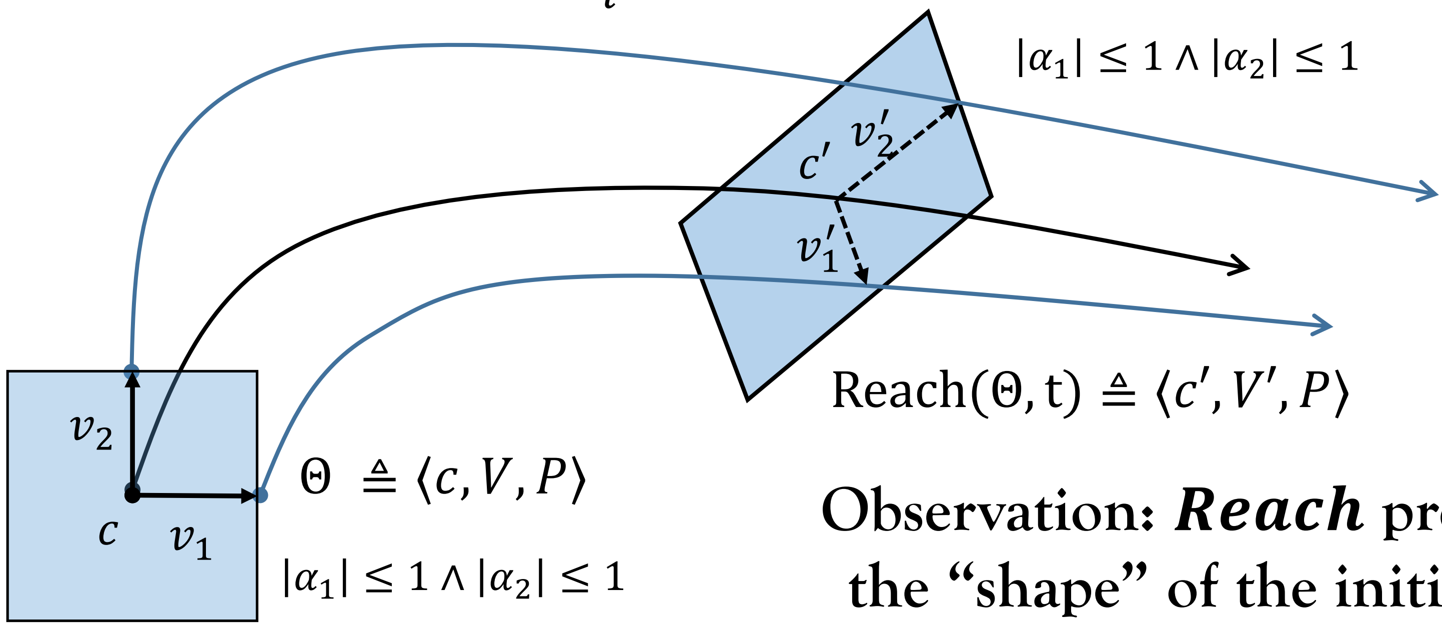
Reachable set at time t is given by $\langle c', V', P \rangle$ where

1. c' is the simulation corresponding to c
2. $v_i{}'$ is the difference of simulations from $c + v_i$ and from c

Technique
Representation + Superposition

Given $\Theta \triangleq \langle c, V, P \rangle$ to compute reachable set

1. Simulate from c
2. Simulate from $c + v_i$ for each i



Observation: *Reach* preserves the "shape" of the initial set.

Reachable set at time t is given by $\langle c', V', P \rangle$ where

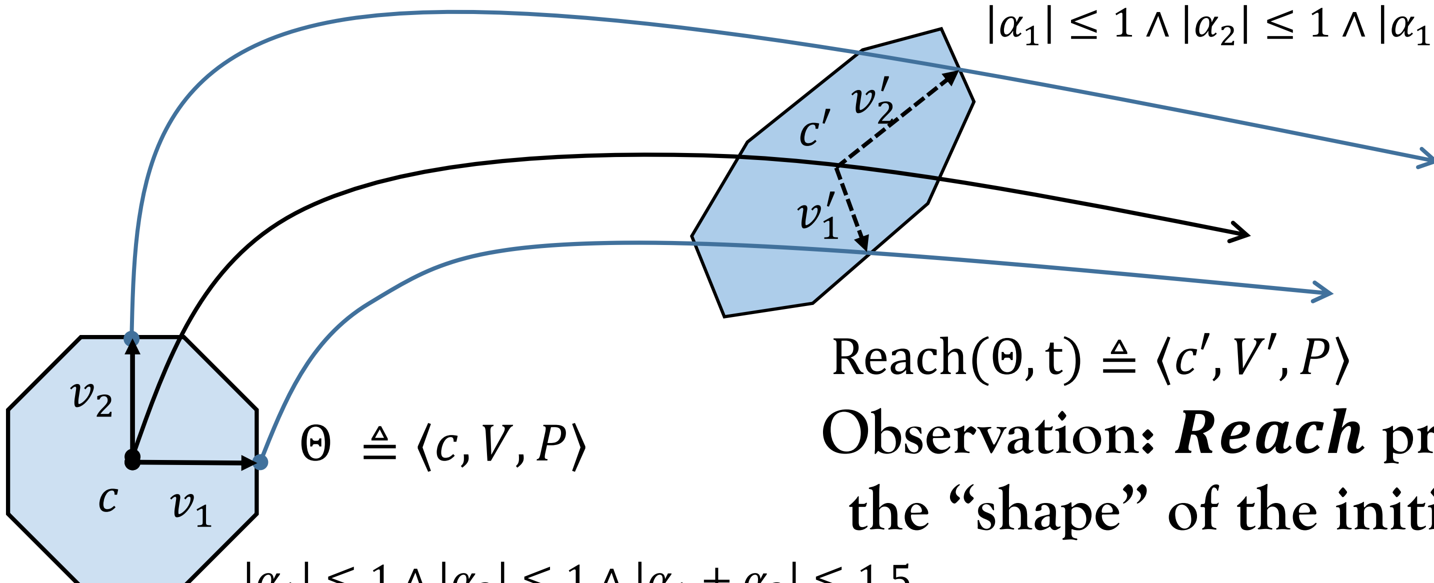
1. c' is the simulation corresponding to c

2. $v_i{}'$ is the difference of simulations from $c + v_i$ and from c

Technique
Representation + Superposition

Given $\Theta \triangleq \langle c, V, P \rangle$ to compute reachable set

1. Simulate from c
2. Simulate from $c + v_i$ for each i



Observation: *Reach* preserves the "shape" of the initial set.

Reachable set at time t is given by $\langle c', V', P \rangle$ where

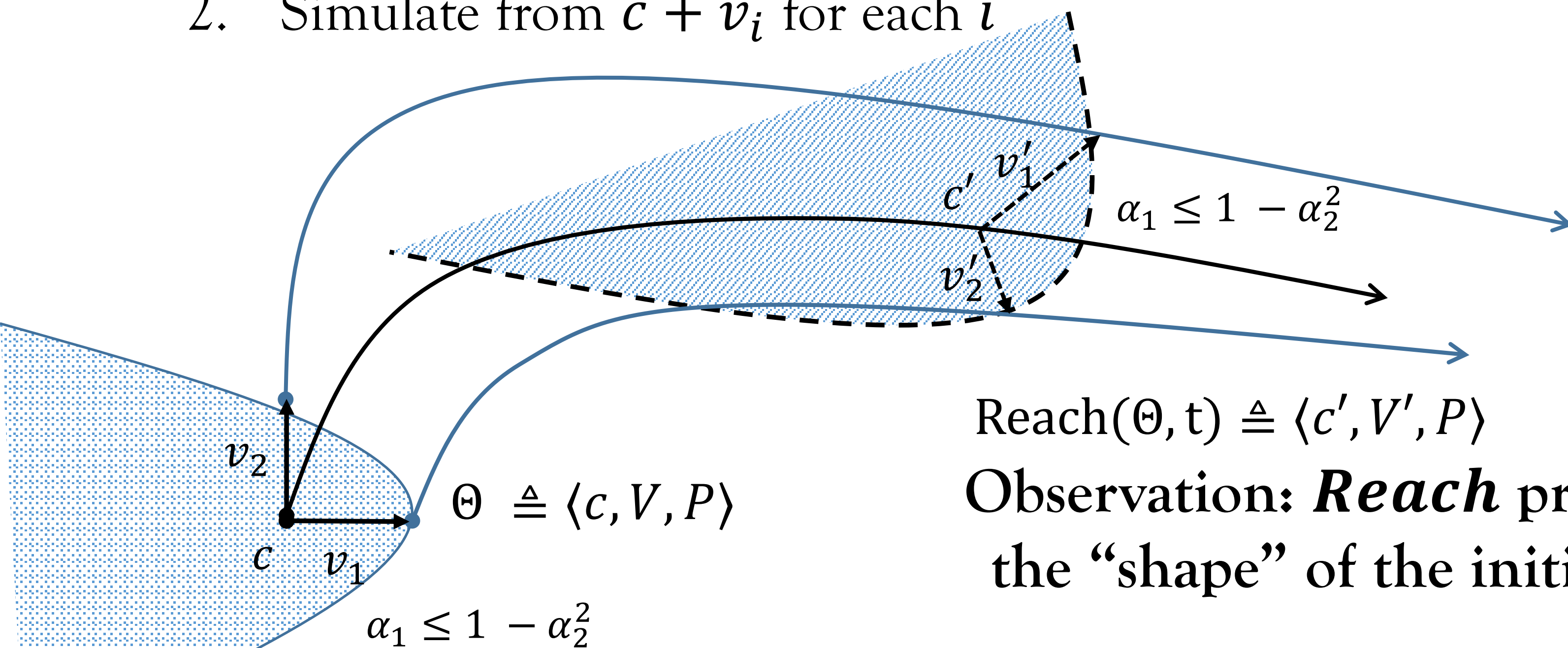
1. c' is the simulation corresponding to c

2. v_i' is the difference of simulations from $c + v_i$ and from c

Reachable Set Computation Using Simulations For Generalized Stars

Given $\Theta \triangleq \langle c, V, P \rangle$ to compute reachable set

1. Simulate from c
2. Simulate from $c + v_i$ for each i



$\text{Reach}(\Theta, \text{t}) \triangleq \langle c', V', P \rangle$

Observation: *Reach* preserves the "shape" of the initial set.

Reachable set at time t is given by $\langle c', V', P \rangle$ where

1. c' is the simulation corresponding to c
2. $v_i{}'$ is the difference of simulations from $c + v_i$ and from c

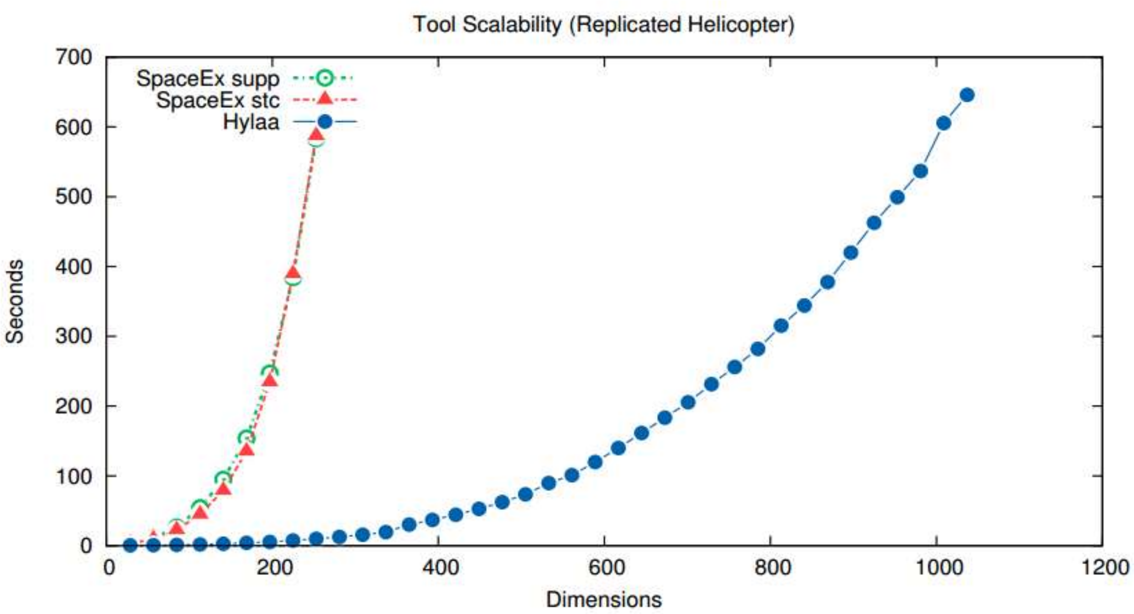
Demo

Extensions

- Accommodate mode switches.
- Developed new invariant constraint propagation technique.
- Dynamic aggregation and deaggregation methods.
- Handle Linear systems with inputs/disturbances.

Experimental Evaluation
HyLAA

Scalability with respect to number of dimensions.



# Dims	supp	stc	HyLAA
29	2.98	2.60	0.42
57	10.93	9.48	0.67
141	94.83	79.23	2.65
253	583.27	587.42	9.79
449	-	-	52.67
1009	-	-	605.38

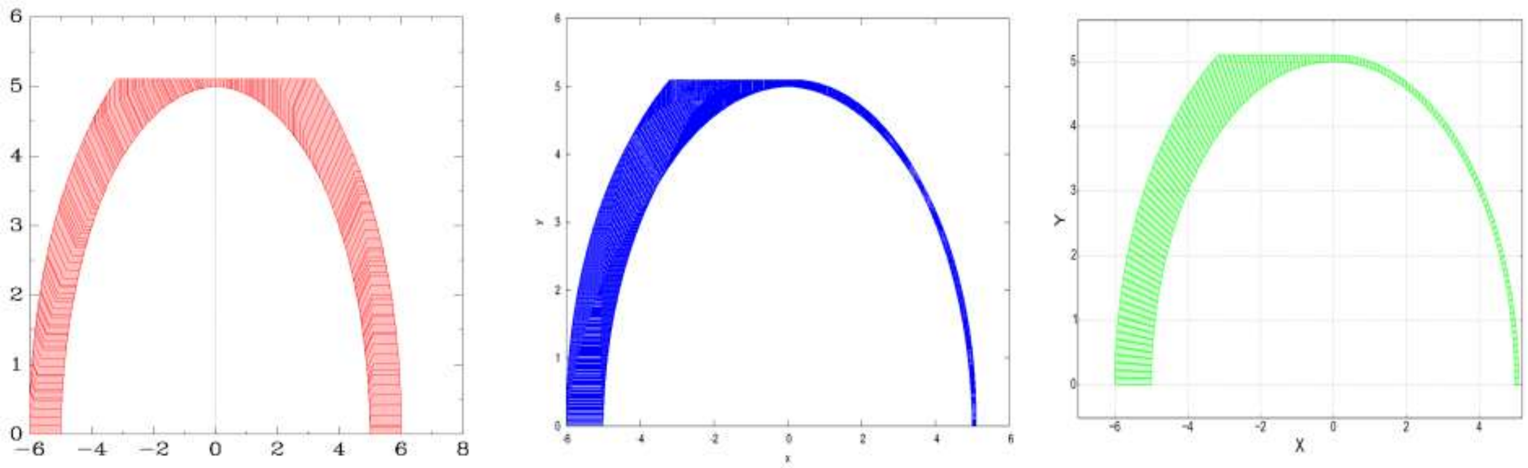
http://stanleybak.com/hylaa/

Running HyLAA on High Dimensional Benchmarks

- Motor (11 dims)
- Building (50 dims)
- Partial Differential Equation (86 dims)
- Heat (202 dims)
- International Space Station (274 dims)
- Clamped Beam (350 dims)
- MNA1 (588 dims)
- FOM (1008 dims)
- MNA5 (10923 dims)

Won the 'Best Paper Award' at ARCH@CPSWeek 2017

HyLAA
Constraint Propagation

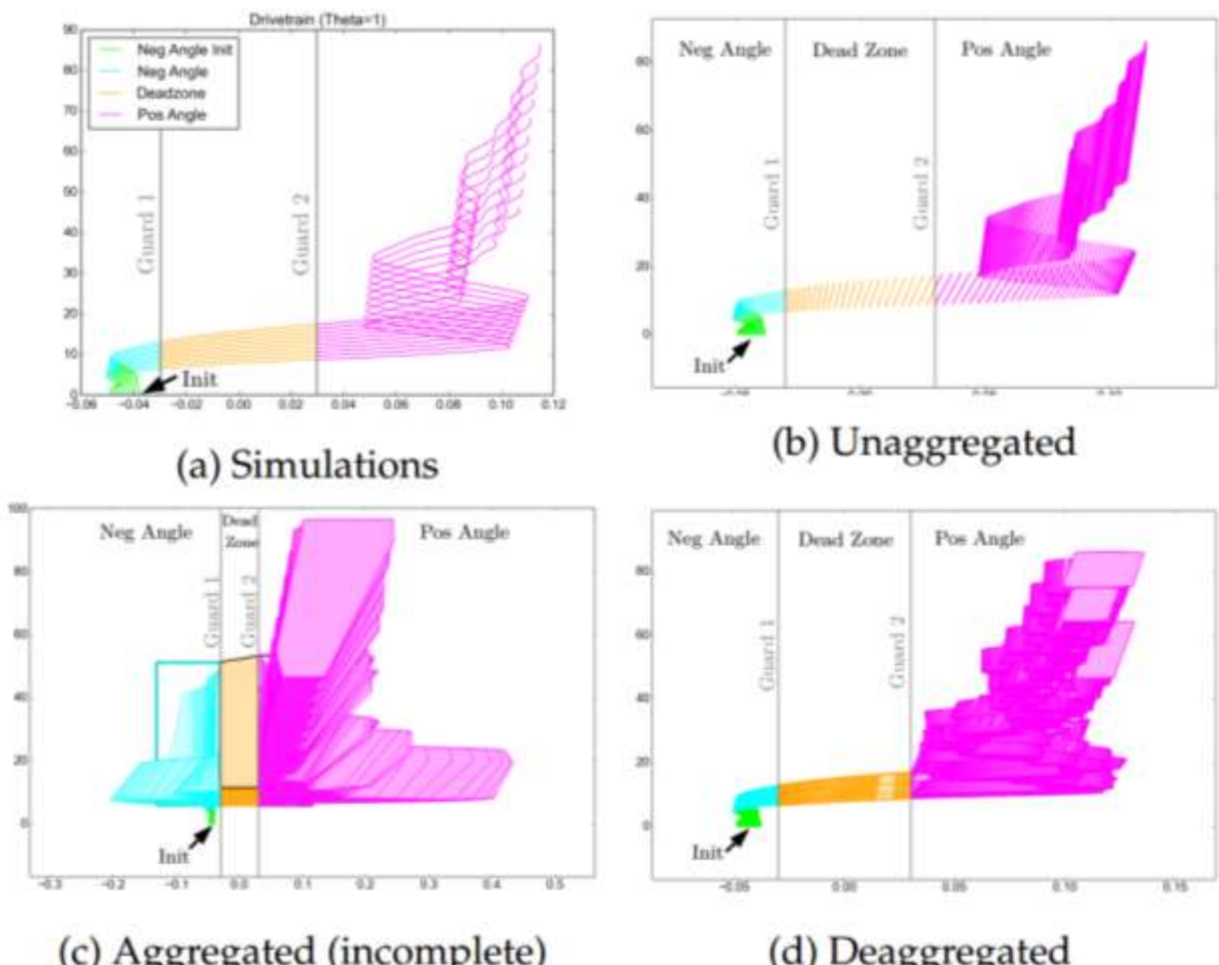


(a) SpaceEx `stc` (b) Flow* (c) HyLAA

Step	No Trim	Trim
0.05	16	5
0.005	119	9
0.001	576	25
0.0005	1148	45

http://stanleybak.com/hylaa/

HyLAA
Aggregation and Deaggregation



(a) Simulations (b) Unaggregated (c) Aggregated (incomplete) (d) Deaggregated

- Expensive to not have any aggregation.
- Completely aggregated introduces new transitions and doesn't terminate.
- Dynamic deaggregation has 1.2x – 5x speedup based on the system.

http://stanleybak.com/hylaa/

HyLAA
Aggregation and Deaggregation

# Dims	10	12	14	16	18	20	24	30	42
Deaggregated	25.70	44.94	24.71	131.82	47.72	267.71	450.42	331.57	516.21
Unaggregated	112.94	79.24	98.63	145.87	214.80	409.55	561.47	384.55	672.60

- Automotive drivetrain system with additional masses ($8 + 2\theta$).
- In lower dimensions, the synchronous behavior of masses gives a better performance for aggregation.
- In higher dimensions, the benefits of aggregation are low because deaggregation is performed more often.

http://stanleybak.com/hylaa/

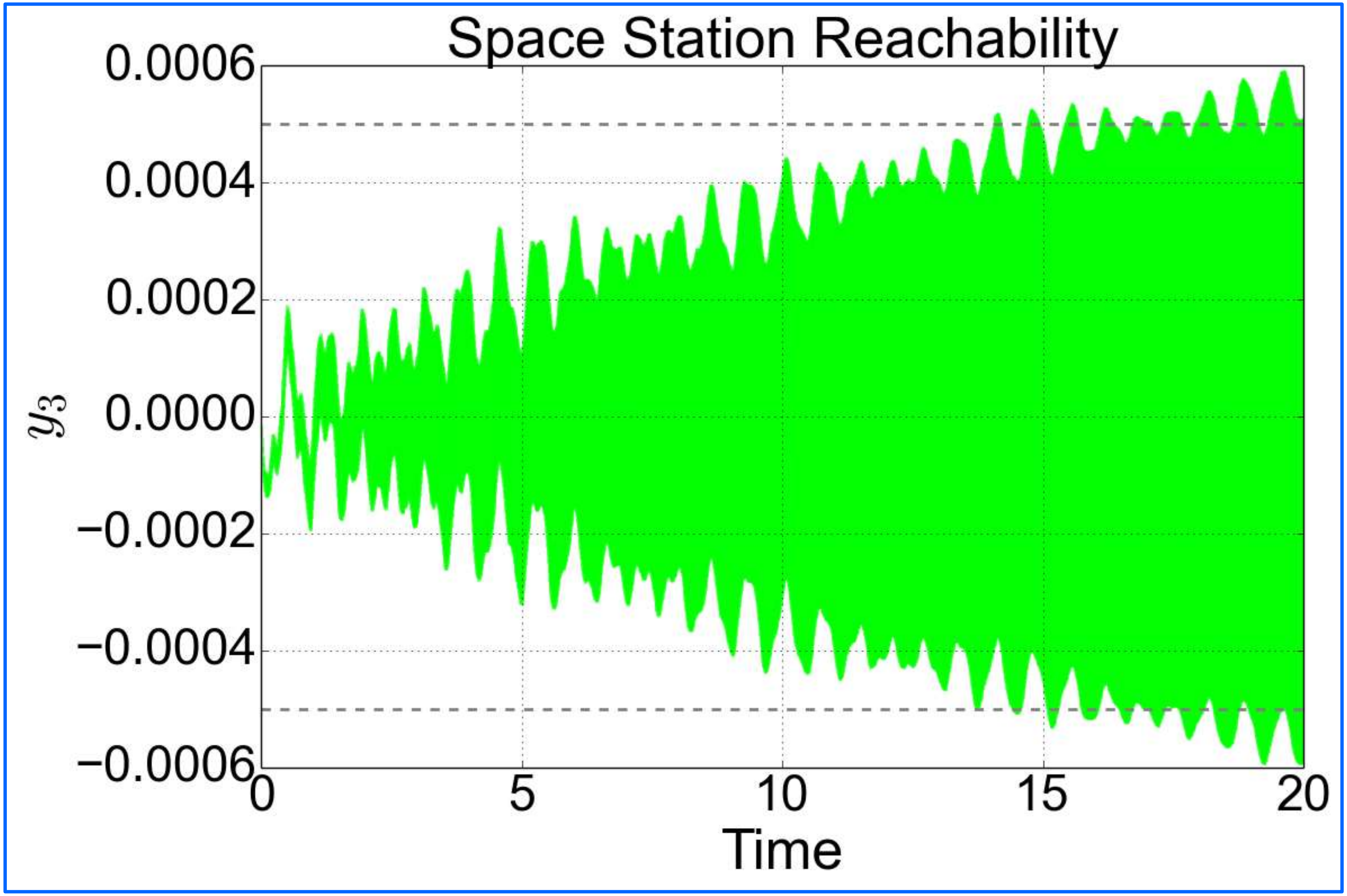
International Space Station Model (271 dimensions)

ISS	271	$y_3 \notin [-0.0007, 0.0007]$	Hylaa	1m28s	✓	-	-
ISS*	271	$y_3 \notin [-0.0005, 0.0005]$	Hylaa	1m23s		$8.5 \cdot 10^{-6} / 1.3 \cdot 10^{-5}$	13.71

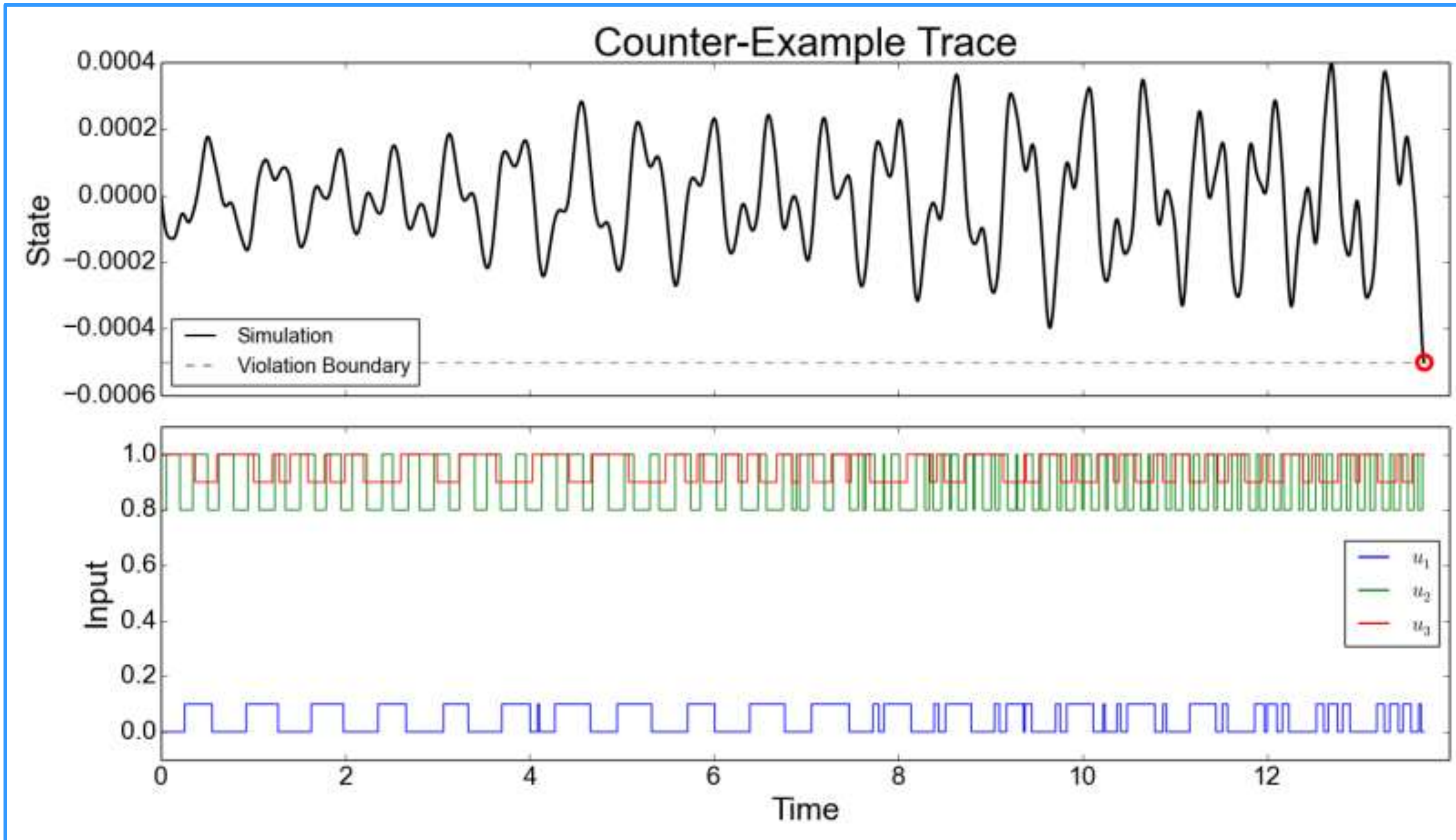
- The original safety specification was created using simulations. For most models it was safe.

- For the International Space Station model, however, **it was not!** This shows that simulation can miss errors. The error was not known before analysis with **Hylaa**.

Reachability Plot



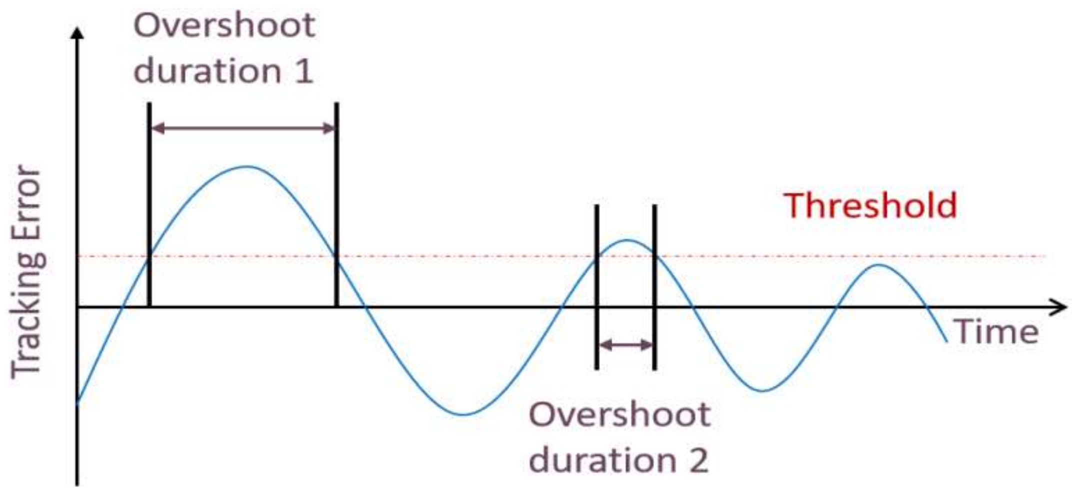
Space Station Specification Violation



- $2^{270} \times 8^{(13.71/0.005)} = 3 \times 10^{2557}$ cases!
- Falsification tool did not succeed after 4 hours.

Counterexample Generation

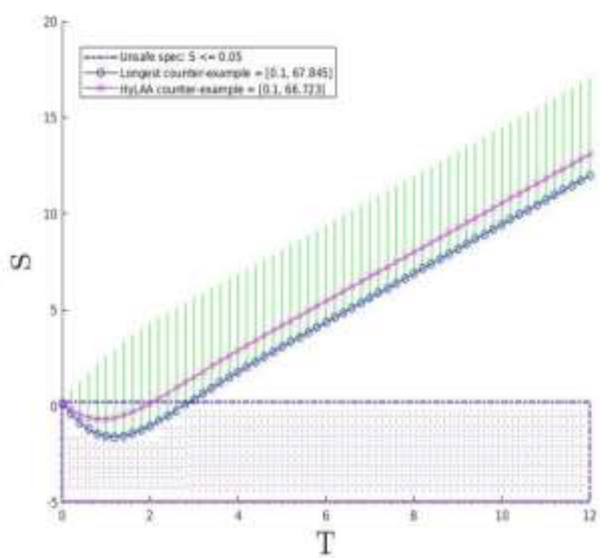
- Control parameter tuning for regulation.

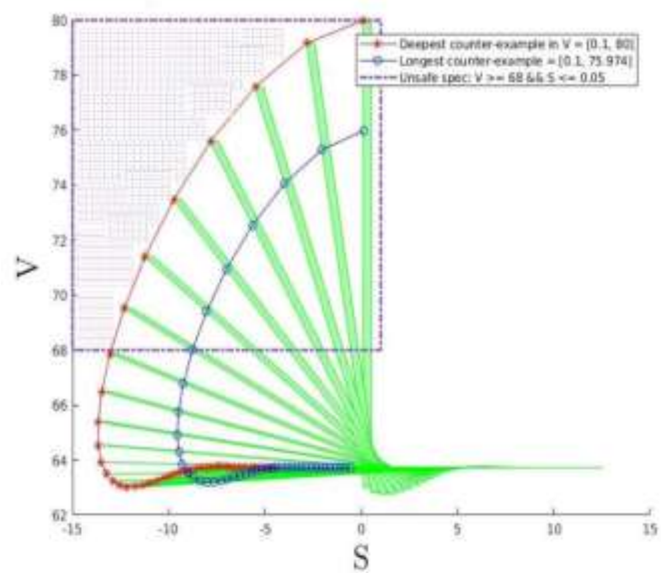


- Any execution that crosses the threshold is not useful.
- Executions that go "maximum" beyond the threshold are more important than others.
- Executions that stay longer above threshold are also important.

Longest and Deepest Counterexamples

- **Deepest counterexample**: execution that ventures into unsafe set resulting in a maximum “depth”.
- **Longest counterexample**: execution that stays for longest in unsafe set contiguously.
- Using constraint propagation, developed a new technique that generates these two counterexamples.

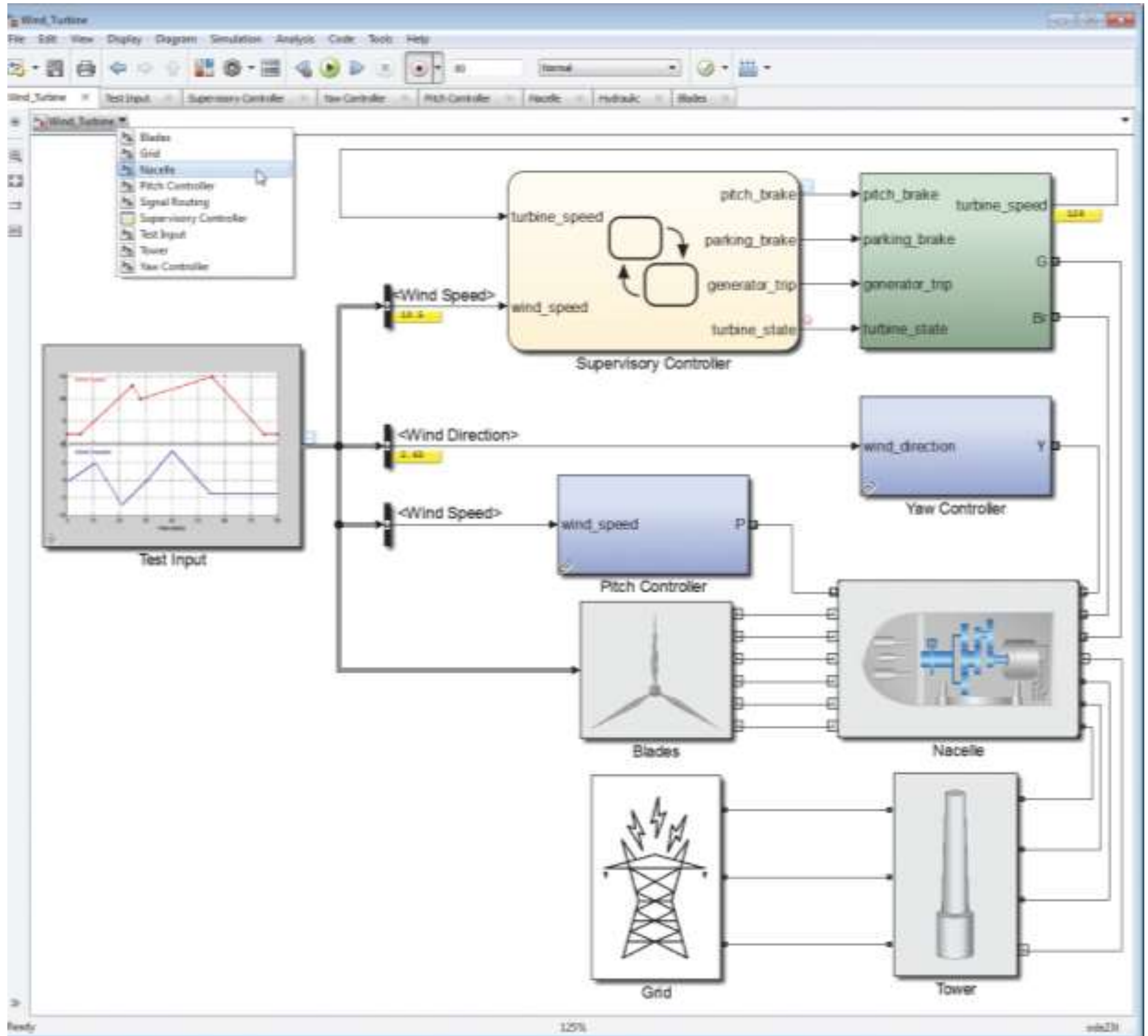




Future Work

What Real Systems Look Like?

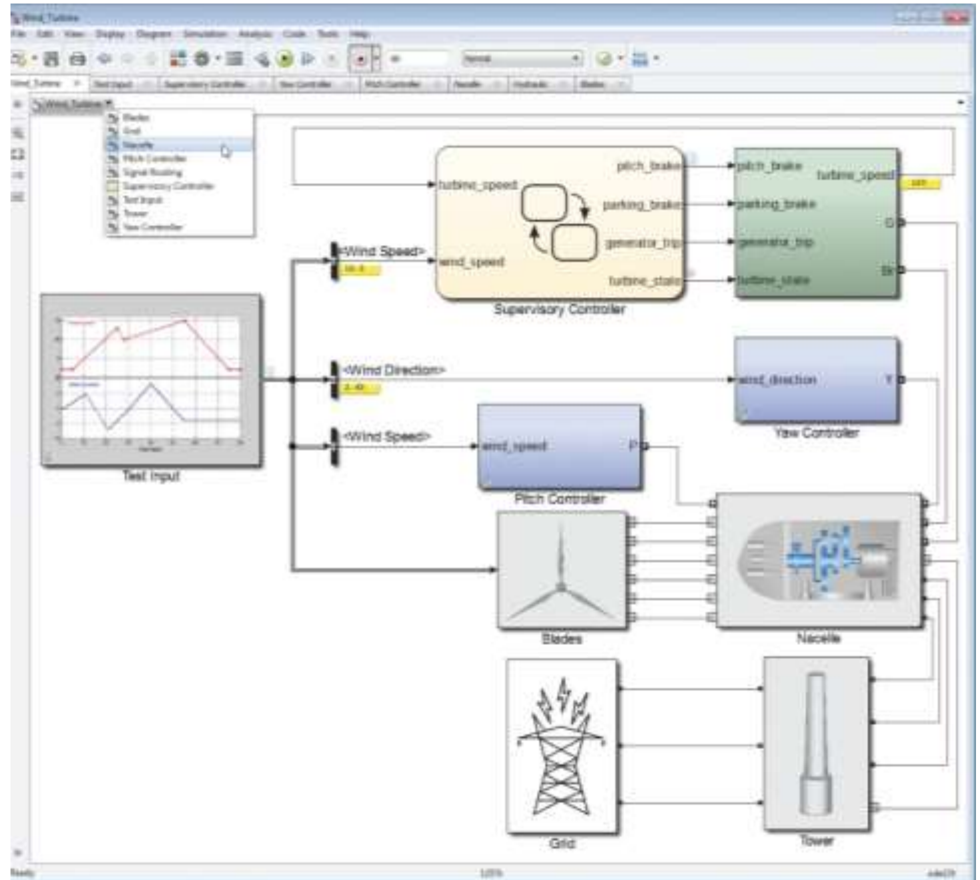
- Nonlinear
- Complex software
- Distributed
- Heterogenuous time scales
- Uncertainities
- Failures



Who Gives The Specification?

- For each component?
- In a temporal logic?

Absolutely unrealistic!

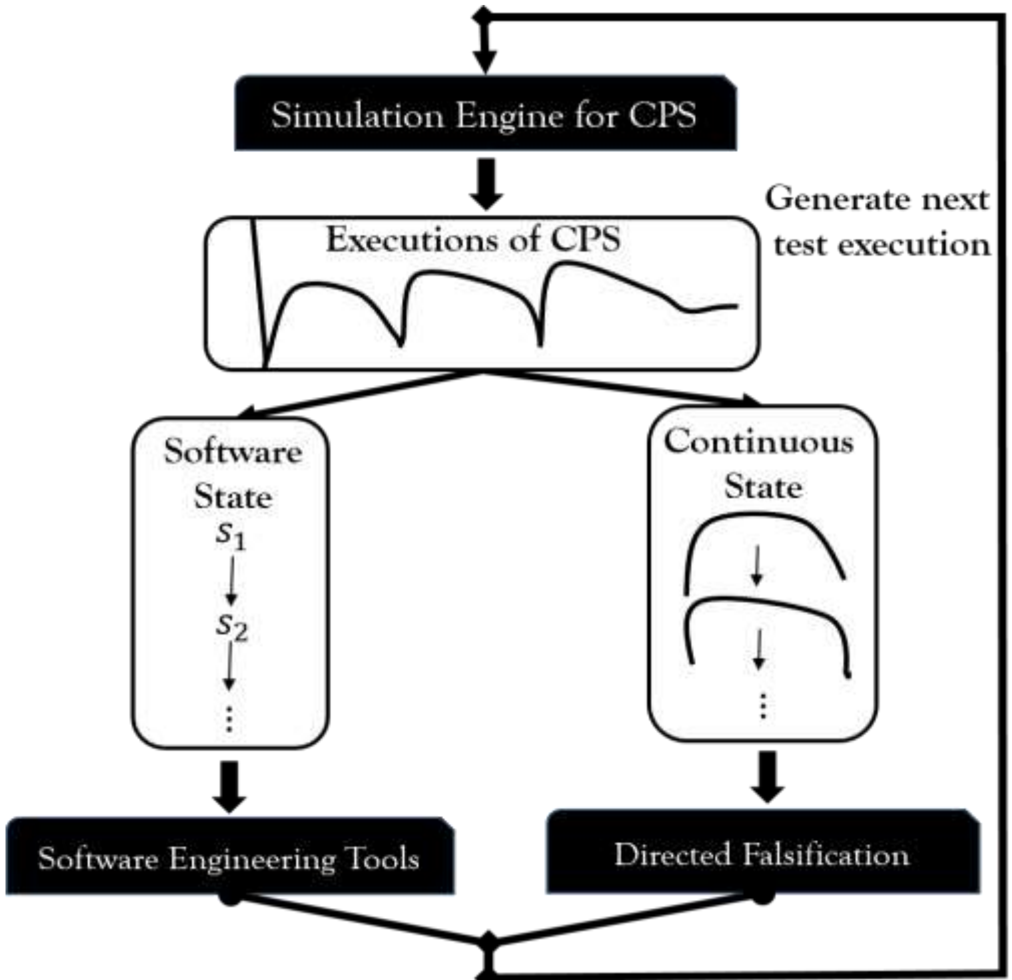


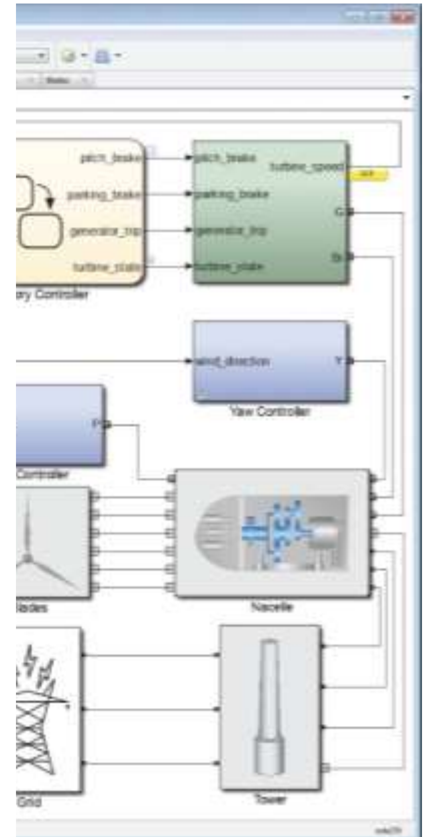
Who Gives The Specification?

- For e:
- In a t

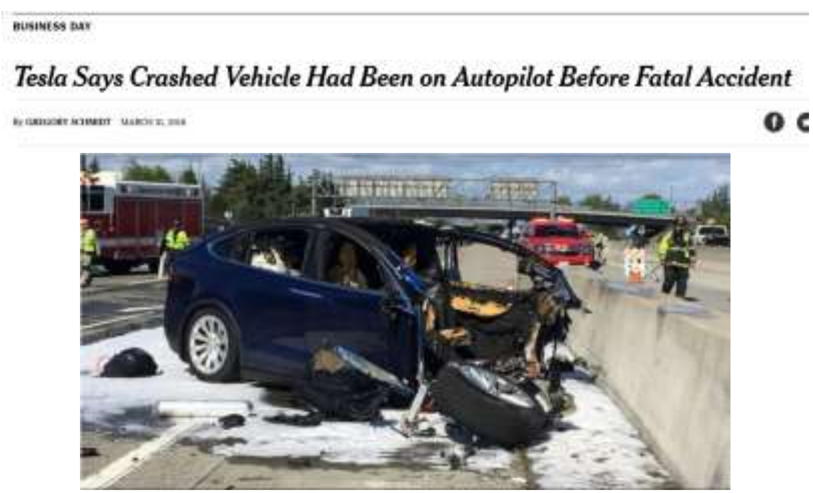
Absolu

Verification without specification!





Sometimes, CPS have bugs

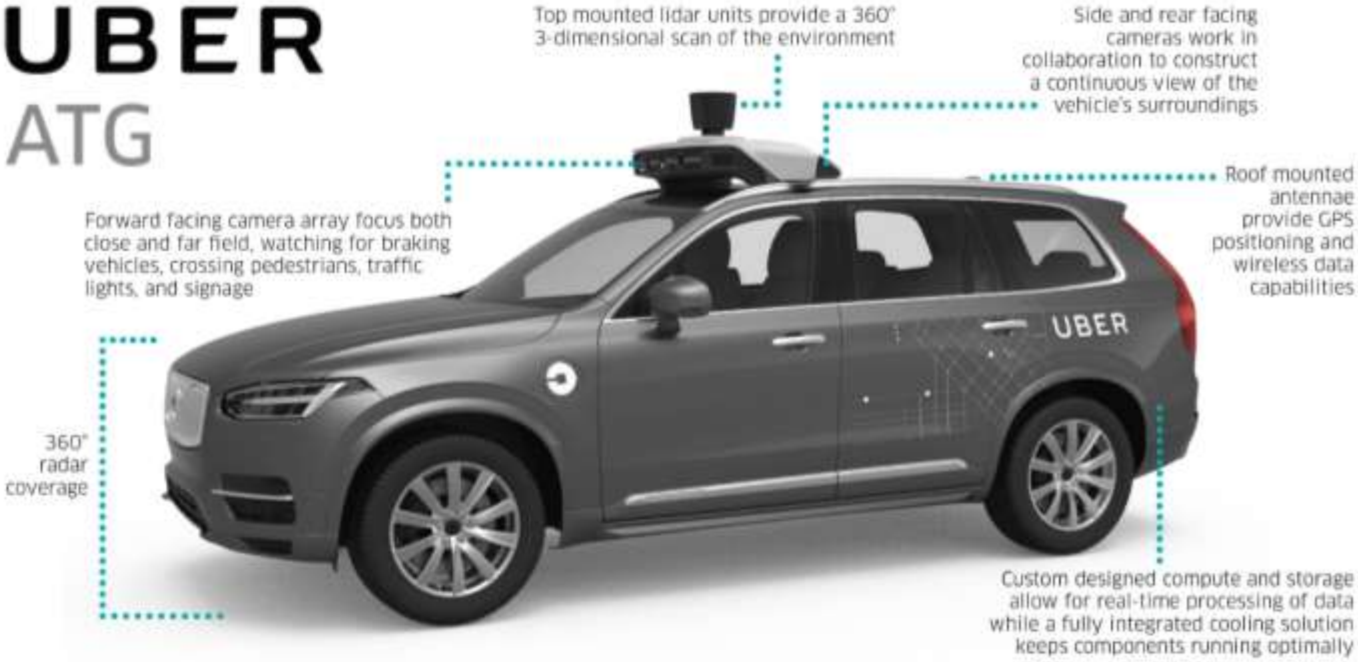


Disengagement rates
0.16 – 0.78 for 1000 miles

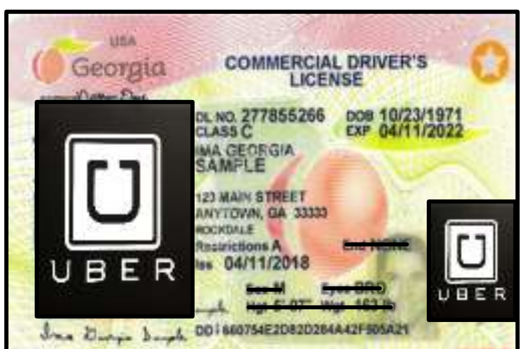
Uber self driving car running red light.
https://www.youtube.com/watch?v=_CdJ4oae8f4

- ❑ Toyota recalls of Prius vehicles (> 20M).
- ❑ Software failures in medical devices (approx. 25%)
- ❑ Northeast power grid blackouts.

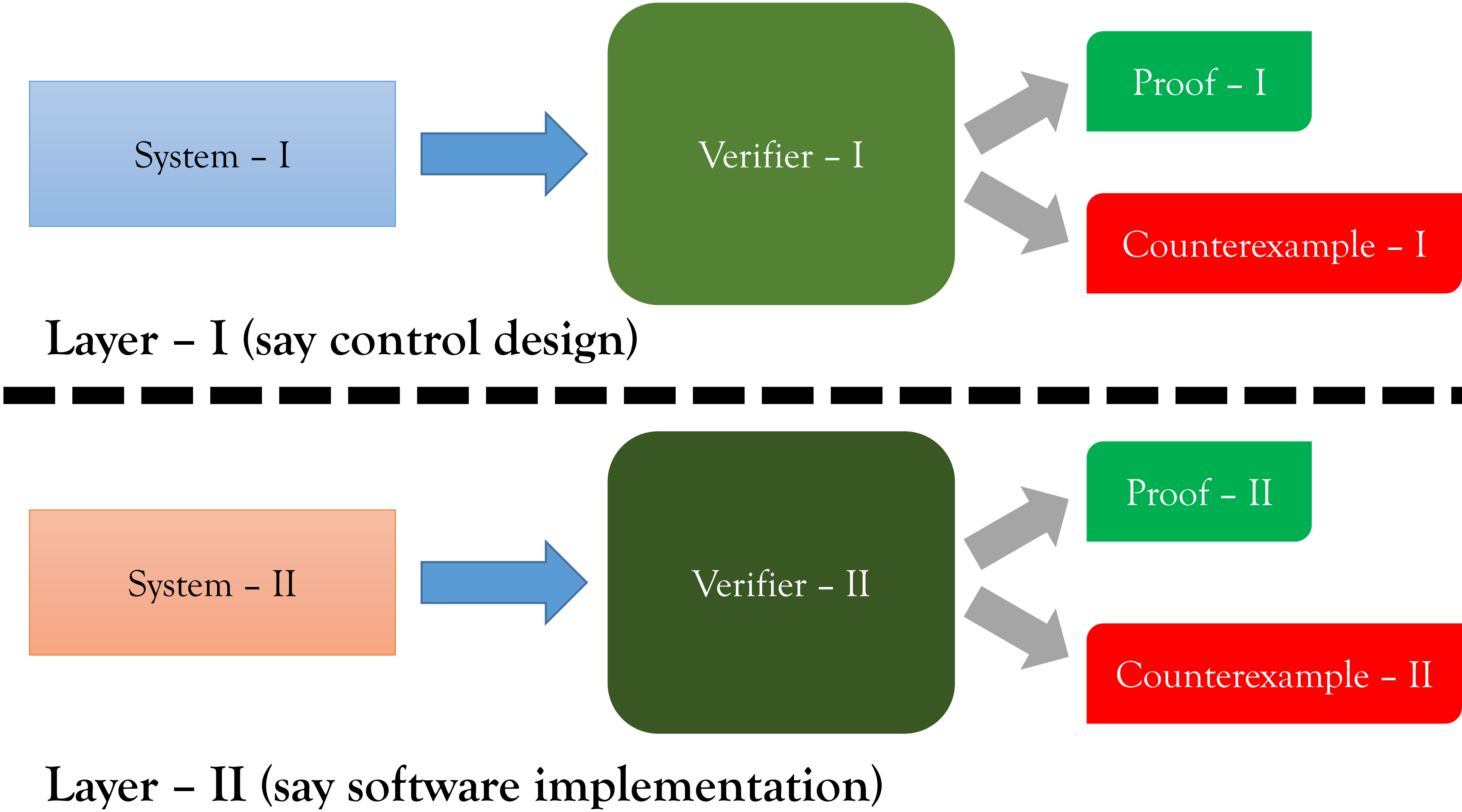
An Enabling Technology



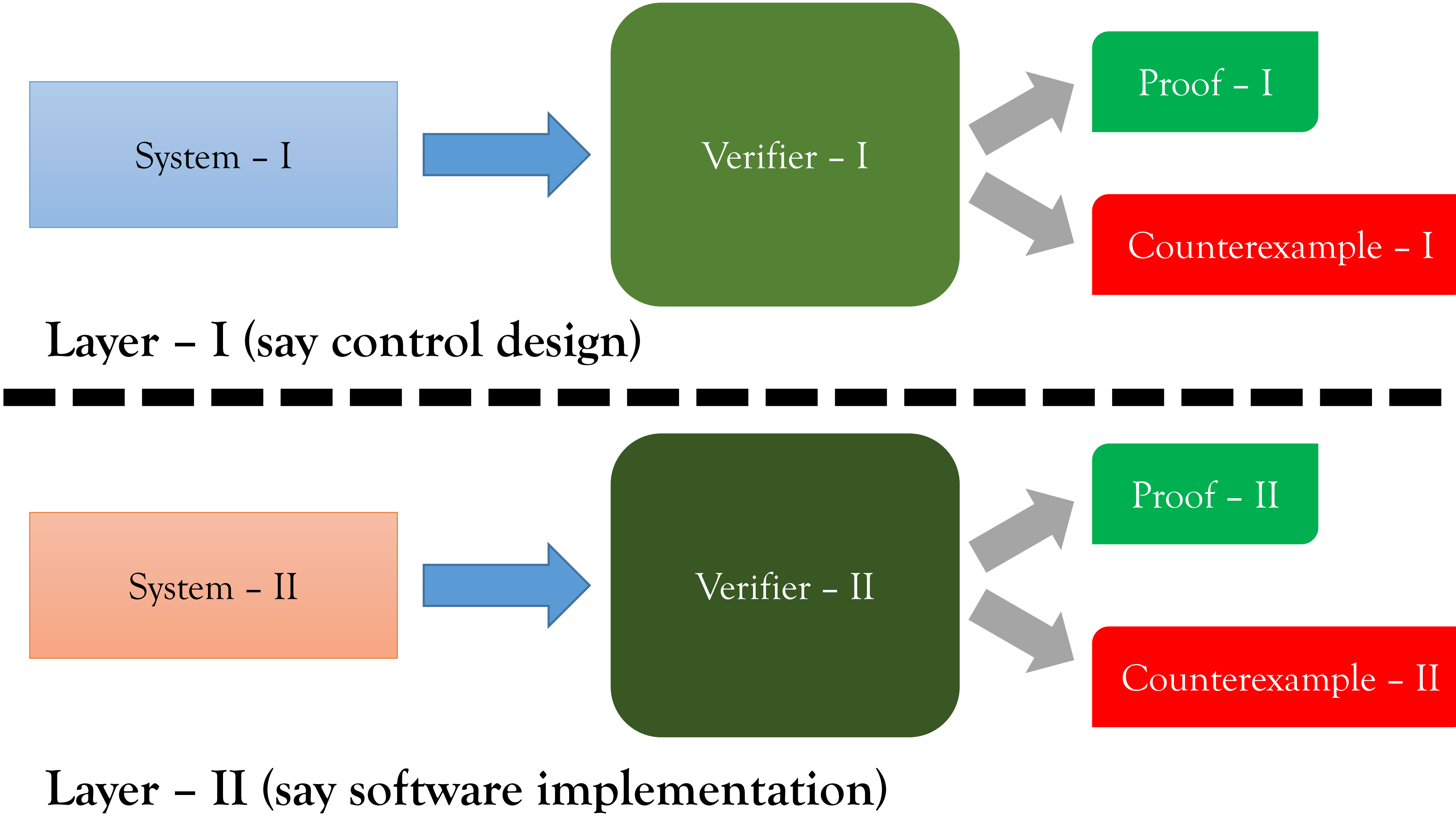
An Enabling Technology



How Is This Different From Design Verification?

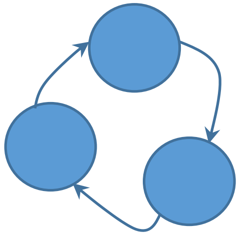


How Is This Different From Design Verification?



Do the proofs work together?

A Layered Approach For End-To-End Verification of Autonomous Vehicles



Model checking
hybrid systems

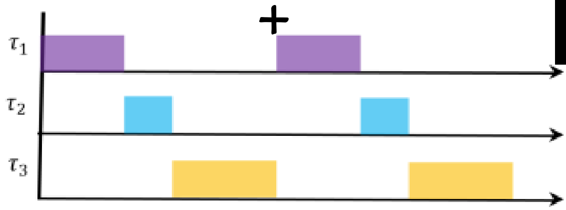
+

Robust w.r.t. perturbation proof.

Software verification of
embedded code

+

Conformance Checking



Scheduling analysis

+

Scheduler verification

Hardware correctness proofs

+

System Identification Analysis

Plant
+
Noisy environment

Sound approx. model

Let’s Hope For a Day Where Autonomous Vehicles and Humans Coexist Peacefully

Thank You

- Developed algorithms for verification of nonlinear systems.
- Scalable linear systems verification.

Future work

- Verification without specification
- Certification of autonomous vehicles





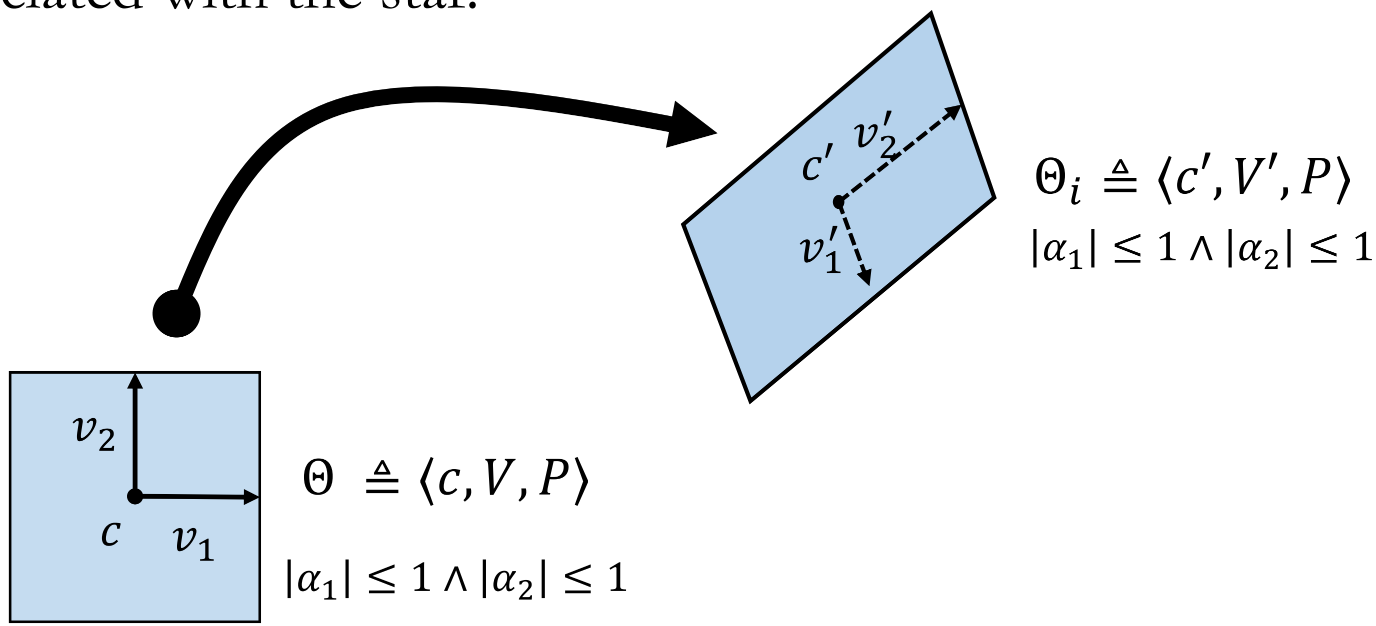


Questions?

Backup Slides

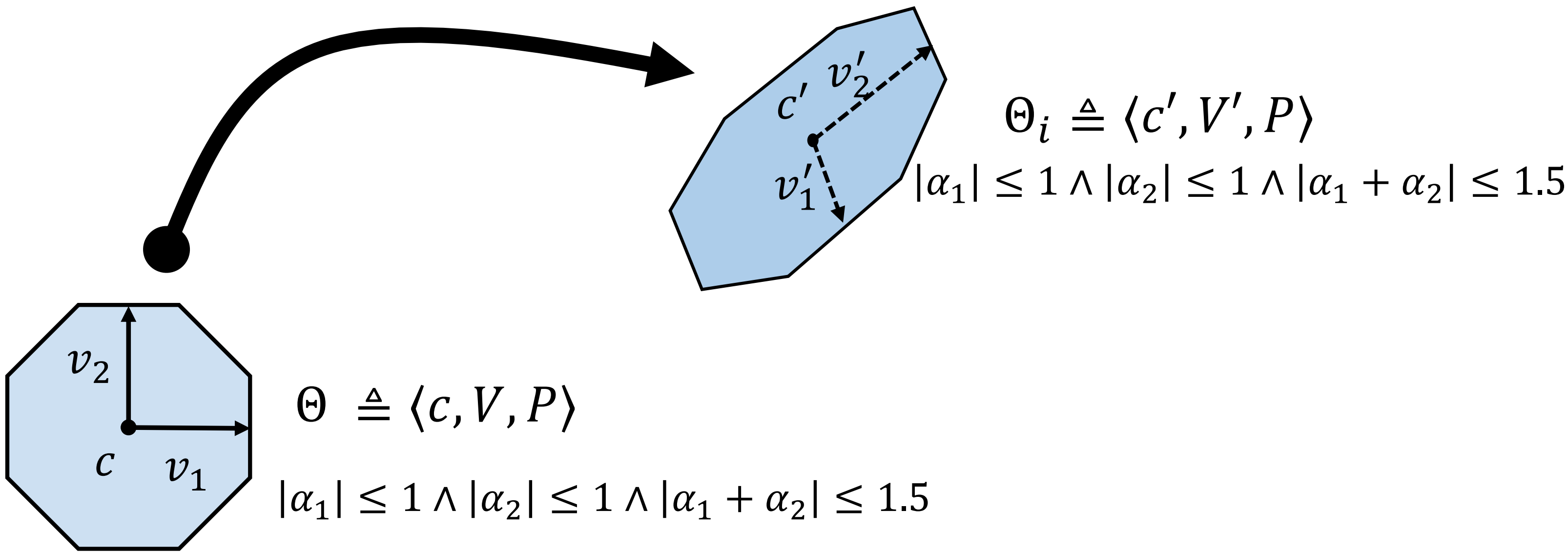
Observations

1. The discrete time reachable set doesn't change the predicate associated with the star.



Observations

1. The discrete time reachable set doesn't change the predicate associated with the star.

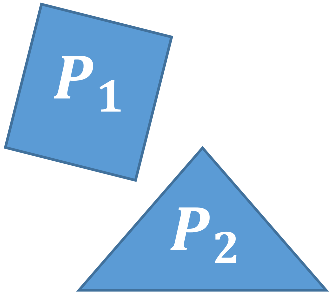


To compute reachable set of a new initial set, just changing the predicate suffices!

Observations

2. It is easy to aggregate and de-aggregate sets on-the-fly.

$\boldsymbol{\Theta_1} = \langle \boldsymbol{c}, \boldsymbol{V}, \boldsymbol{P_1} \rangle$



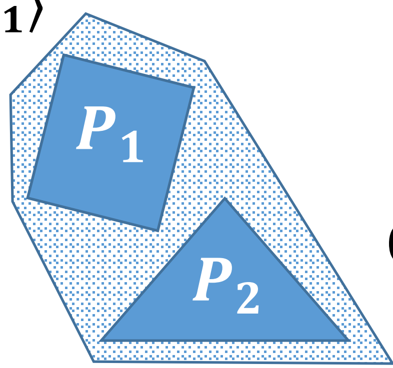
$\boldsymbol{\Theta_2} = \langle \boldsymbol{c}, \boldsymbol{V}, \boldsymbol{P_2} \rangle$

Notice: all have same center and basis in their representation

Observations

2. It is easy to aggregate and de-aggregate sets on-the-fly.

$\boldsymbol{\Theta_1 = \langle c, V, P_1 \rangle}$



$\boldsymbol{\Theta_{agg} = \langle c, V, P_{agg} \rangle}$

$\boldsymbol{(P_1 \vee P_2) \Rightarrow P_{agg}}$

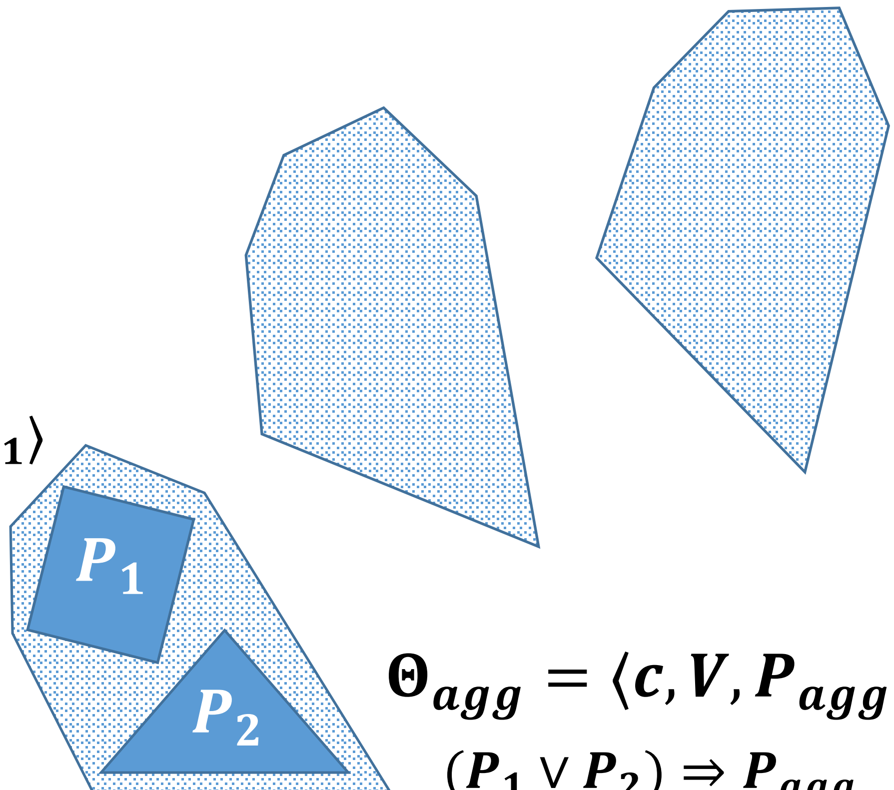
$\boldsymbol{\Theta_2 = \langle c, V, P_2 \rangle}$

Notice: all have same center and basis in their representation

Observations

2. It is easy to aggregate and de-aggregate sets on-the-fly.

$\Theta_1 = \langle c, V, P_1 \rangle$



$$\Theta_{agg} = \langle c, V, P_{agg} \rangle$$

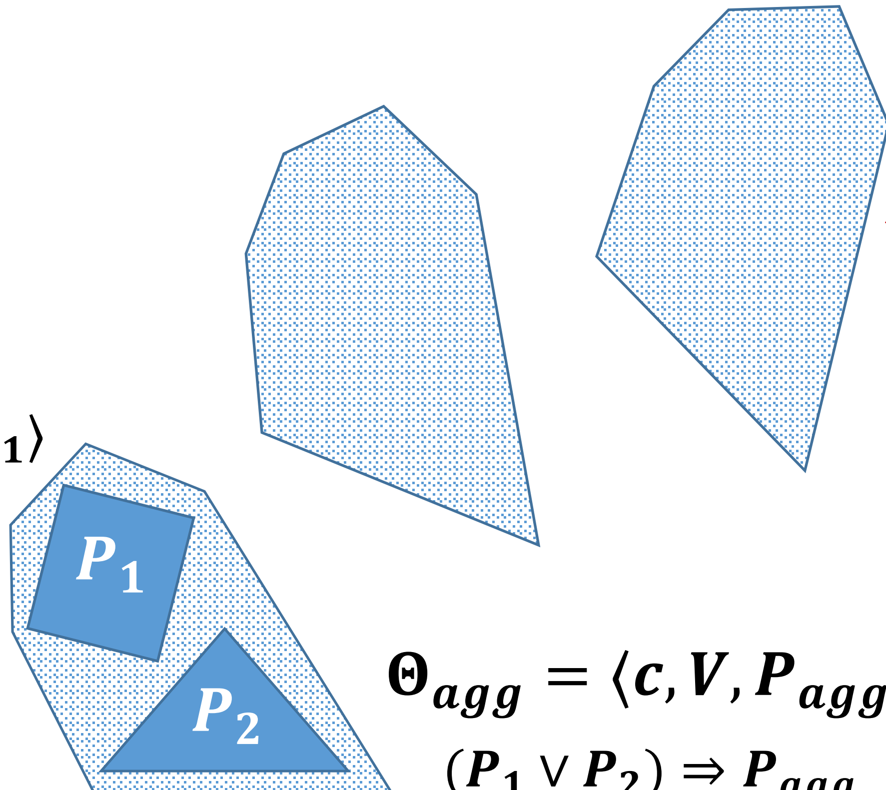
$$(P_1 \vee P_2) \Rightarrow P_{agg}$$

$\Theta_2 = \langle c, V, P_2 \rangle$

Notice: all have same center and basis in their representation

Observations

2. It is easy to aggregate and de-aggregate sets on-the-fly.



$\Theta'_{agg} = \langle c', V', P_{agg} \rangle$

Want to deaggregate?

$\Theta_1 = \langle c, V, P_1 \rangle$

$\Theta_{agg} = \langle c, V, P_{agg} \rangle$

$(P_1 \vee P_2) \Rightarrow P_{agg}$

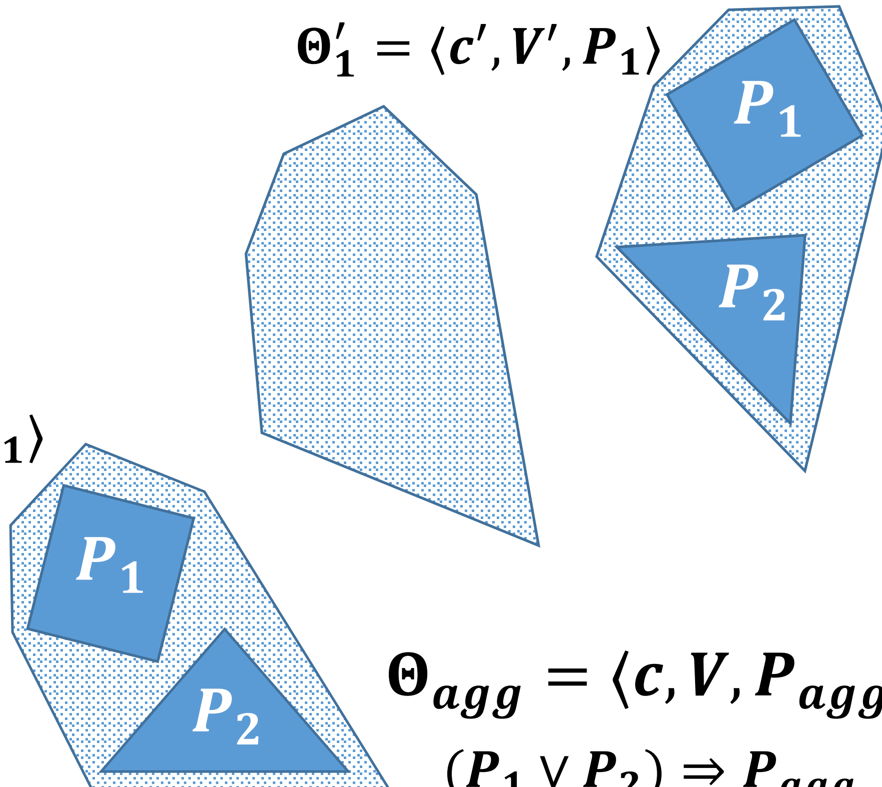
$\Theta_2 = \langle c, V, P_2 \rangle$

Notice: all have same center and basis in their representation

Observations

2. It is easy to aggregate and de-aggregate sets on-the-fly.

$\Theta_1' = \langle c', V', P_1 \rangle$



$\Theta_{agg}' = \langle c', V', P_{agg} \rangle$

Want to deaggregate?

Just change the predicates!

$\Theta_2' = \langle c', V', P_2 \rangle$

$\Theta_1 = \langle c, V, P_1 \rangle$

$\Theta_{agg} = \langle c, V, P_{agg} \rangle$

$(P_1 \vee P_2) \Rightarrow P_{agg}$

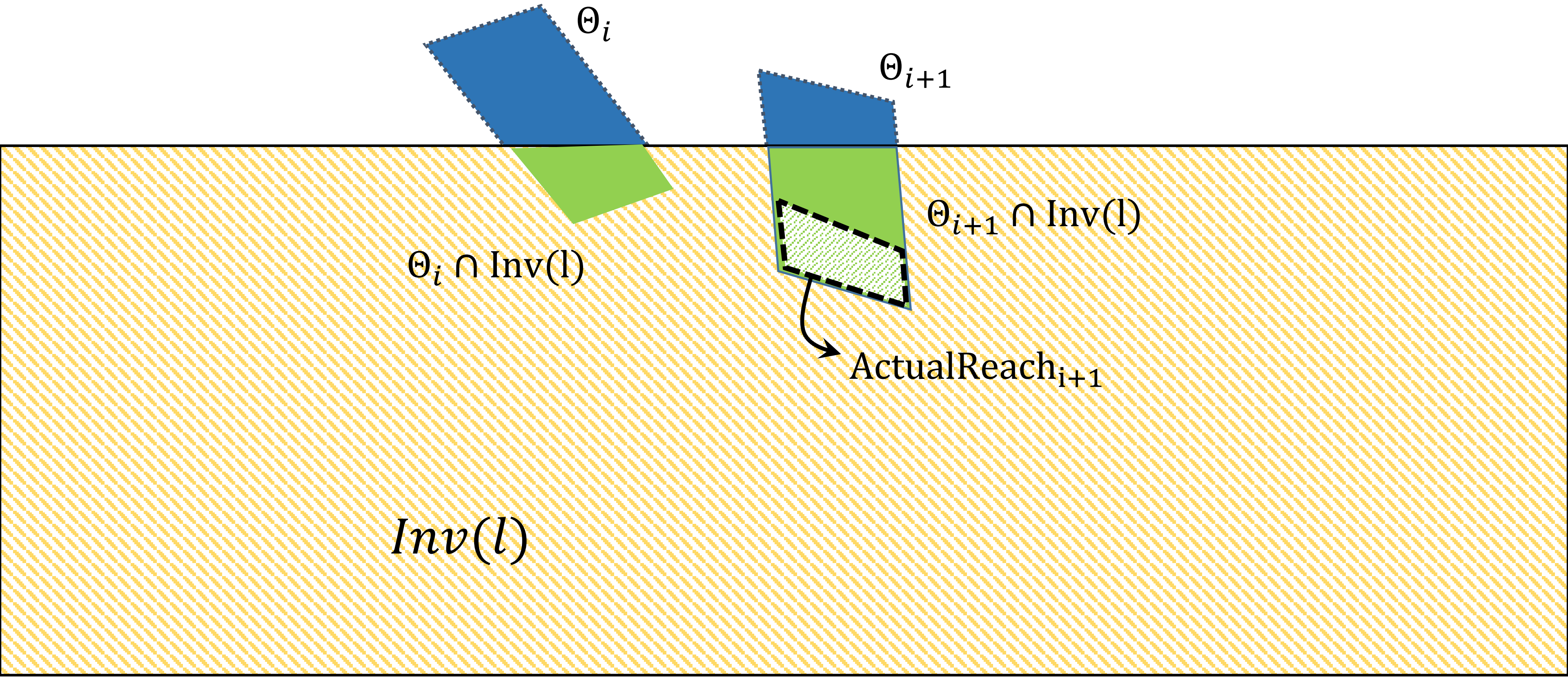
$\Theta_2 = \langle c, V, P_2 \rangle$

Notice: all have same center and basis in their representation

Handling Invariants and Discrete Transitions

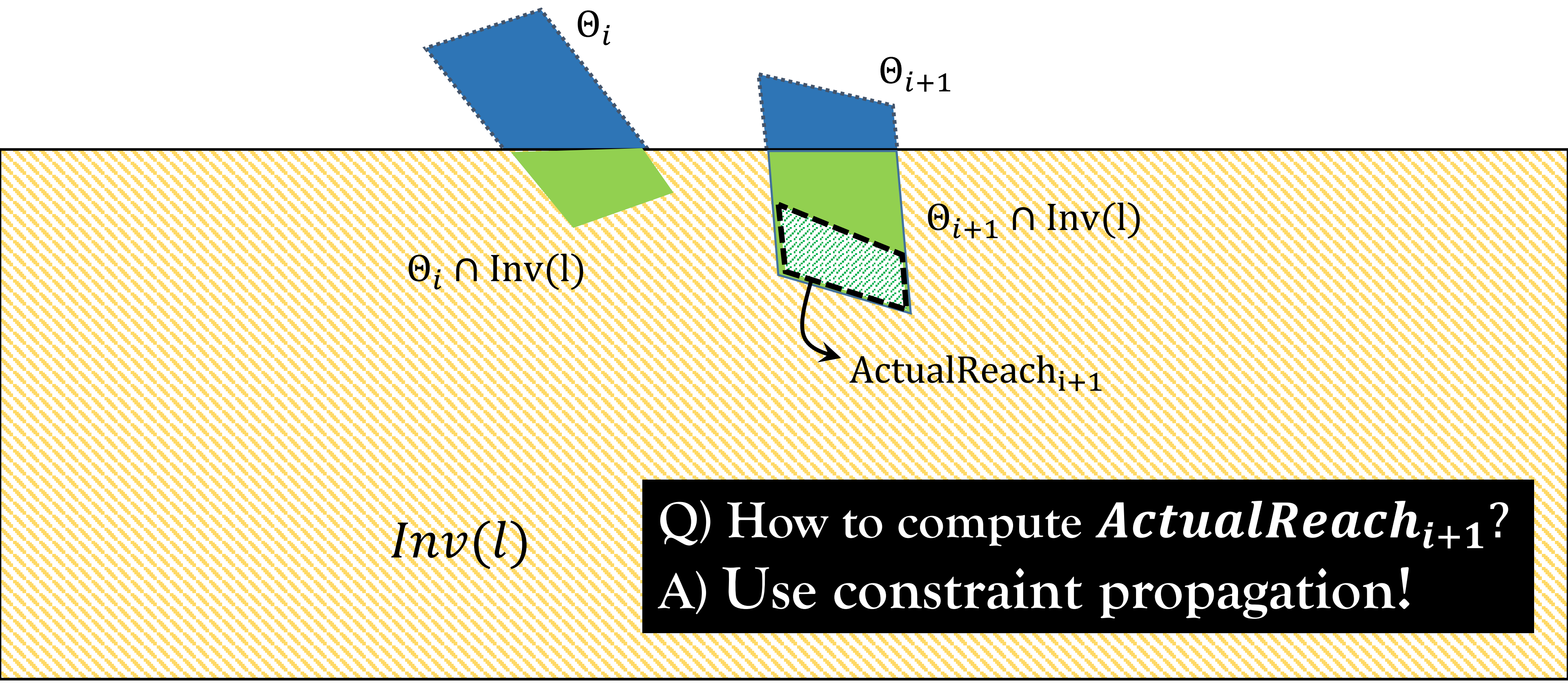
The Problems With Invariants

- Given $\Theta_1, \Theta_2, \ldots, \Theta_k$ as discrete time reachable sets for a given mode, performing just $\Theta_j \cap Inv$ only gives an overapproximation.



The Problems With Invariants

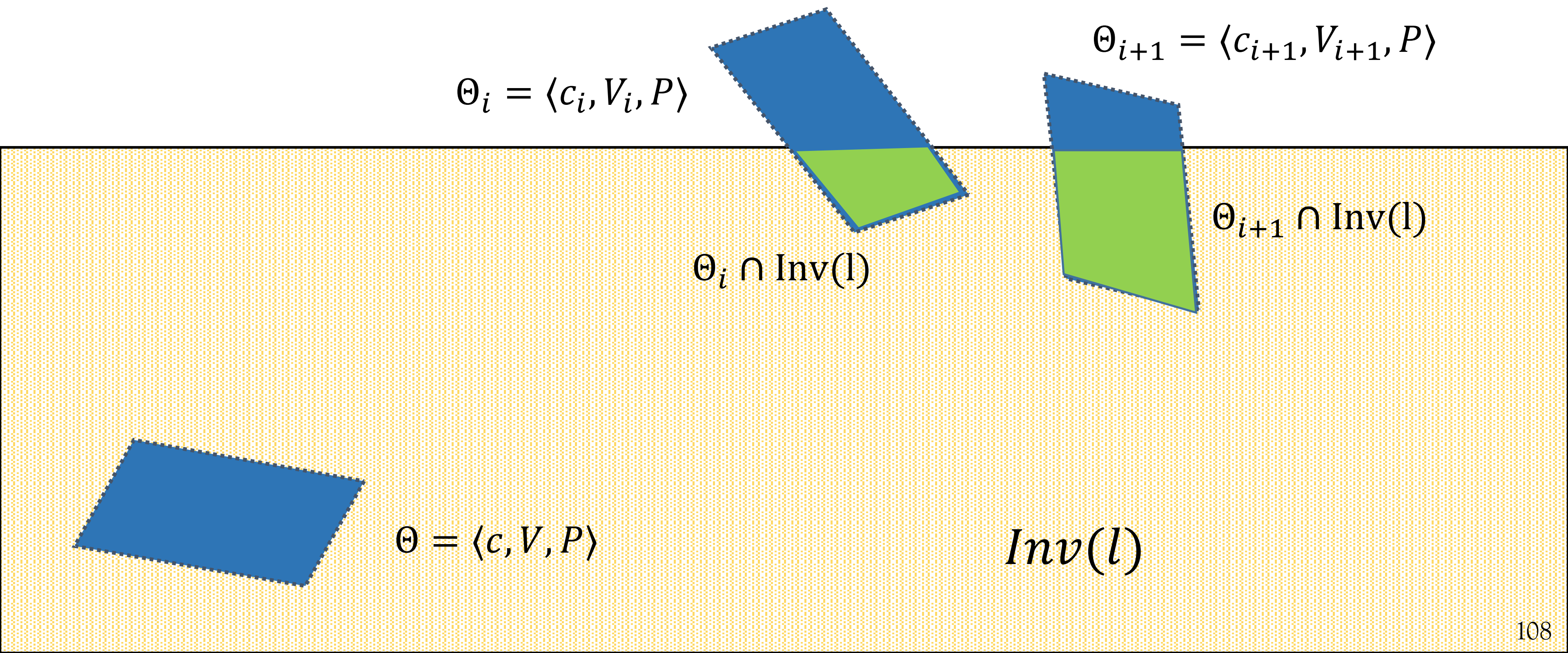
- Given $\Theta_1, \Theta_2, \ldots, \Theta_k$ as discrete time reachable sets for a given mode, performing just $\Theta_j \cap Inv$ only gives an overapproximation.



Q) How to compute $\boldsymbol{ActualReach_{i+1}}$?
A) Use constraint propagation!

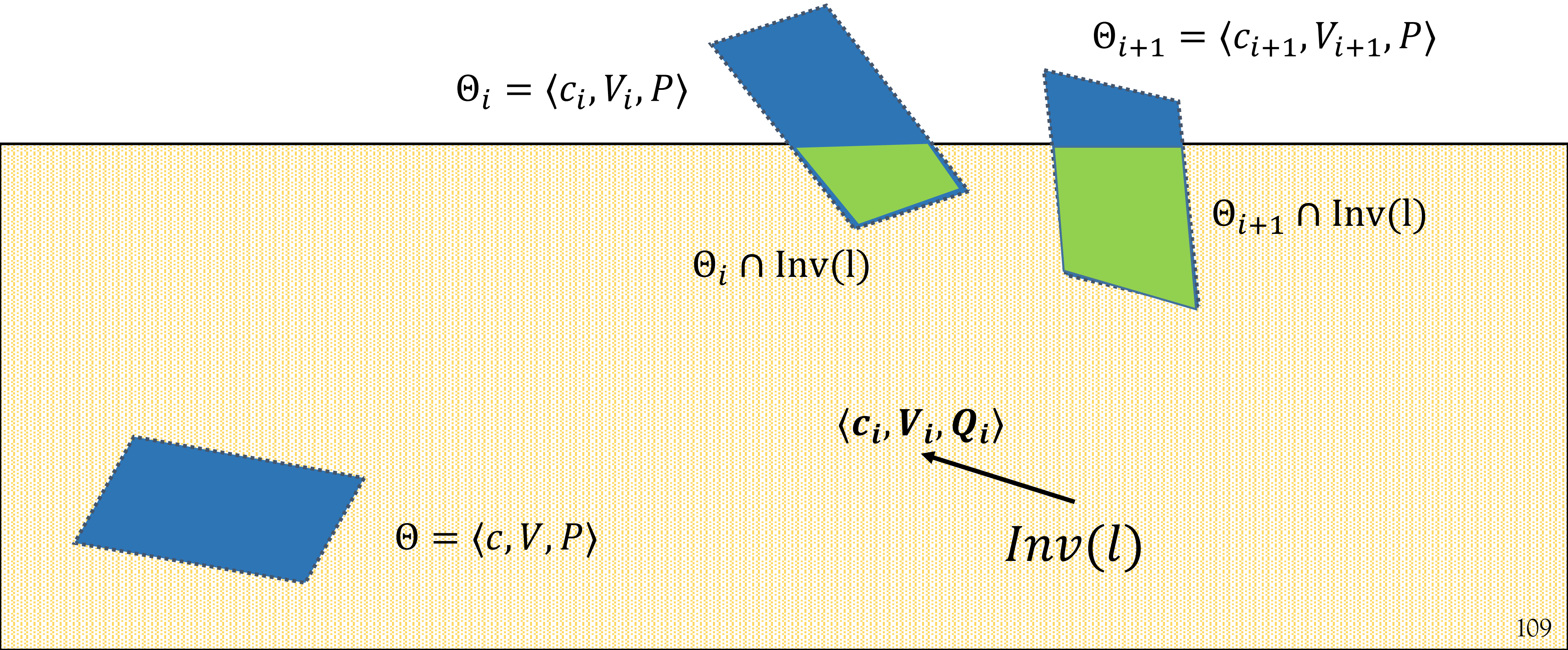
Forward Constraint Propagation

1. Convert Inv into the center and basis of i^{th} star as $\langle c_i, V_i, Q_i \rangle$.
2. $\Theta \cap Inv = \langle c_i, V_i, P \wedge Q_i \rangle$



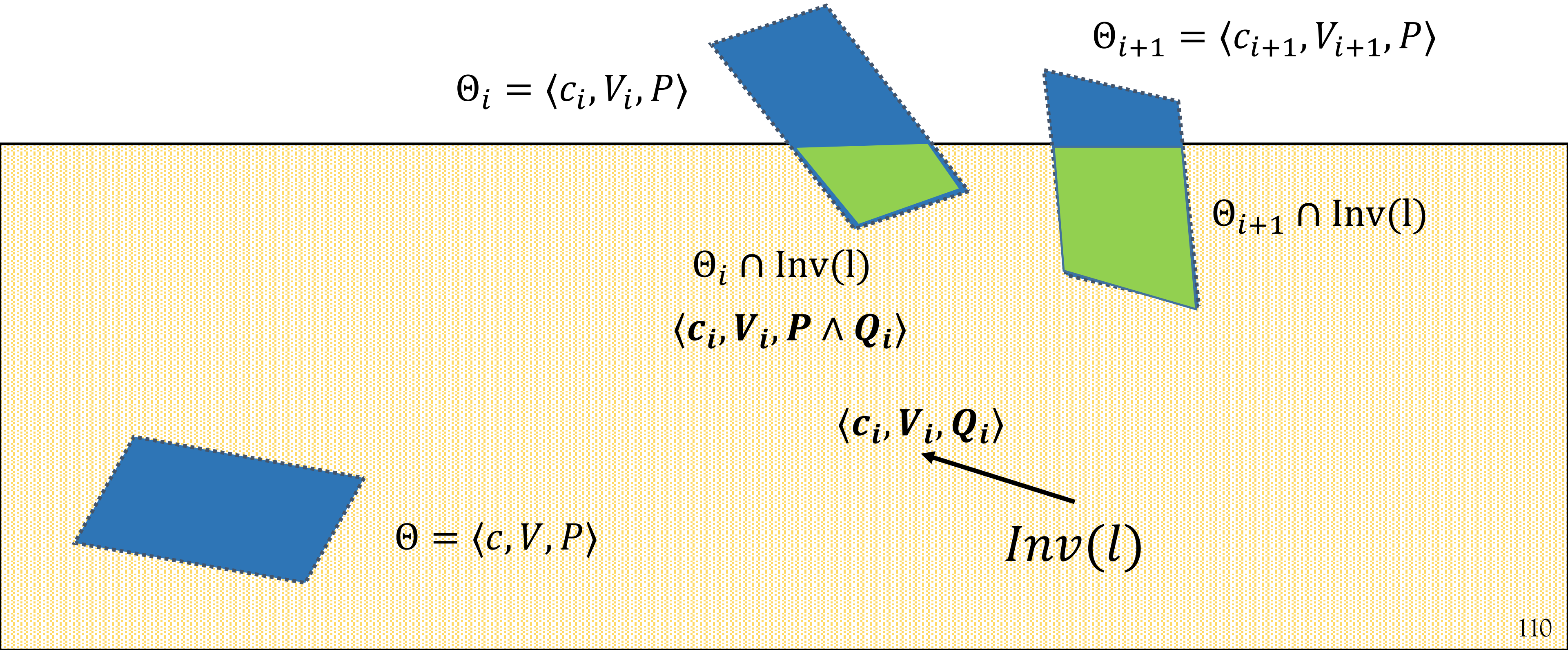
Forward Constraint Propagation

1. Convert Inv into the center and basis of i^{th} star as $\langle c_i, V_i, Q_i \rangle$.
2. $\Theta \cap Inv = \langle c_i, V_i, P \wedge Q_i \rangle$



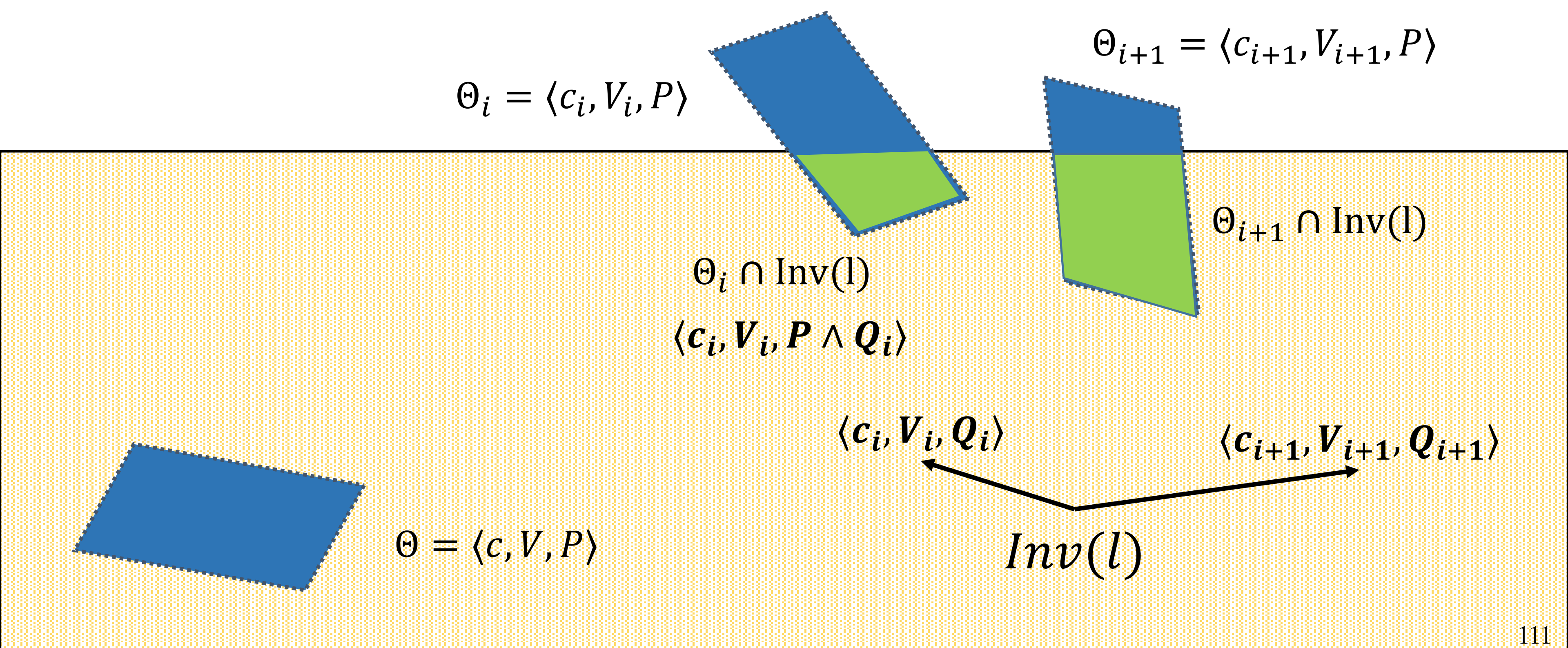
Forward Constraint Propagation

1. Convert Inv into the center and basis of i^{th} star as $\langle c_i, V_i, Q_i \rangle$.
2. $\Theta \cap Inv = \langle c_i, V_i, P \wedge Q_i \rangle$



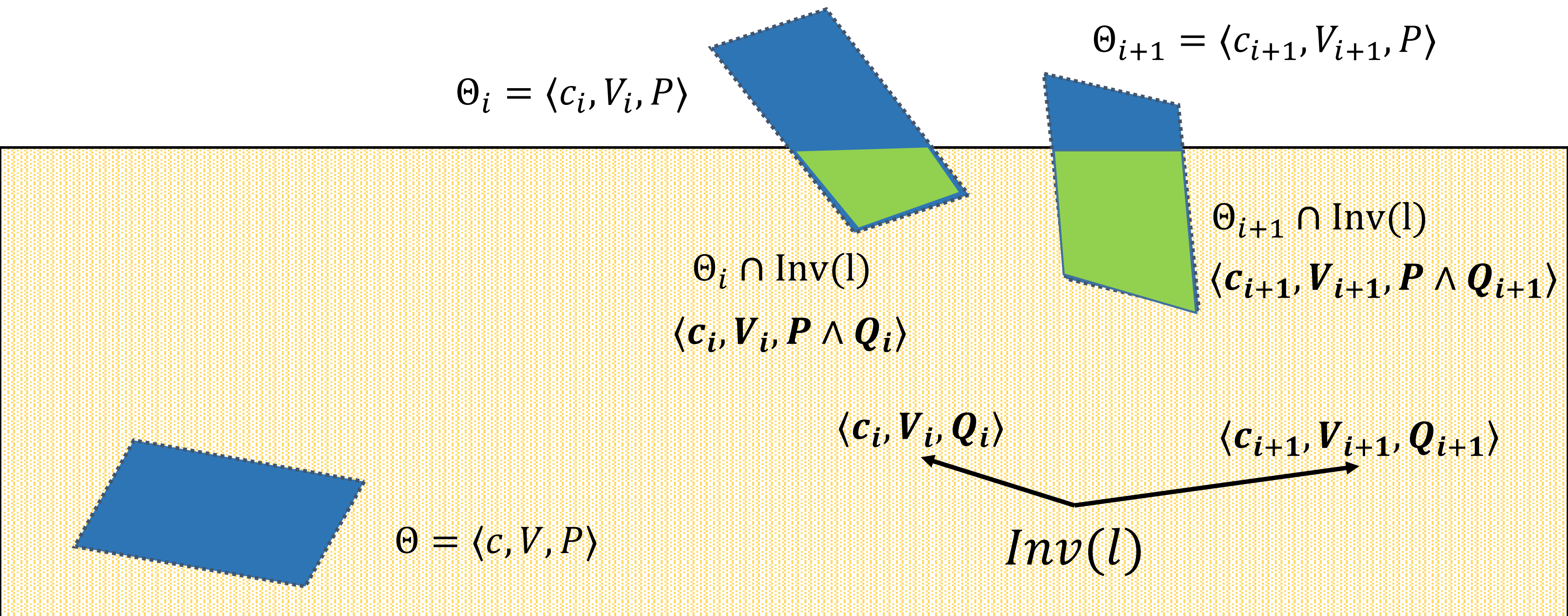
Forward Constraint Propagation

1. Convert Inv into the center and basis of i^{th} star as $\langle c_i, V_i, Q_i \rangle$.
2. $\Theta \cap Inv = \langle c_i, V_i, P \wedge Q_i \rangle$



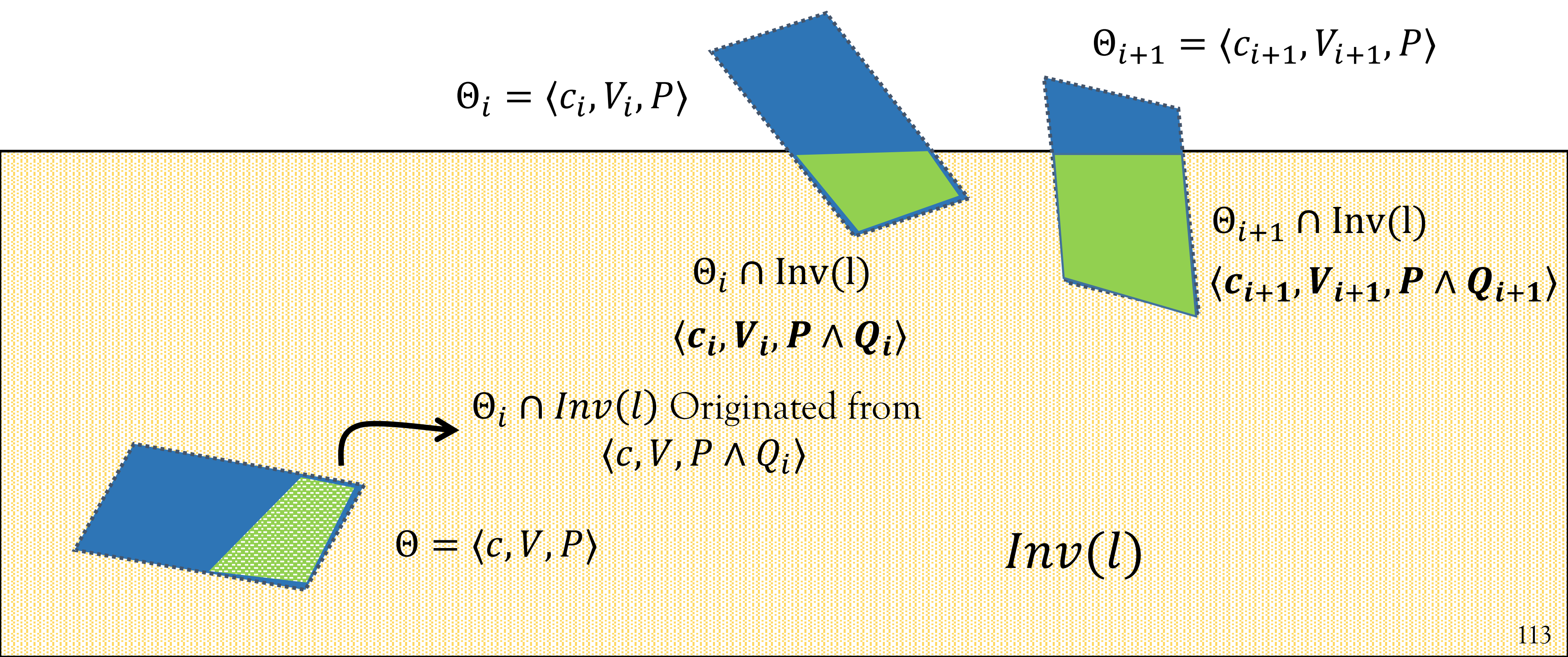
Forward Constraint Propagation

1. Convert Inv into the center and basis of i^{th} star as $\langle c_i, V_i, Q_i \rangle$.
2. $\Theta \cap Inv = \langle c_i, V_i, P \wedge Q_i \rangle$



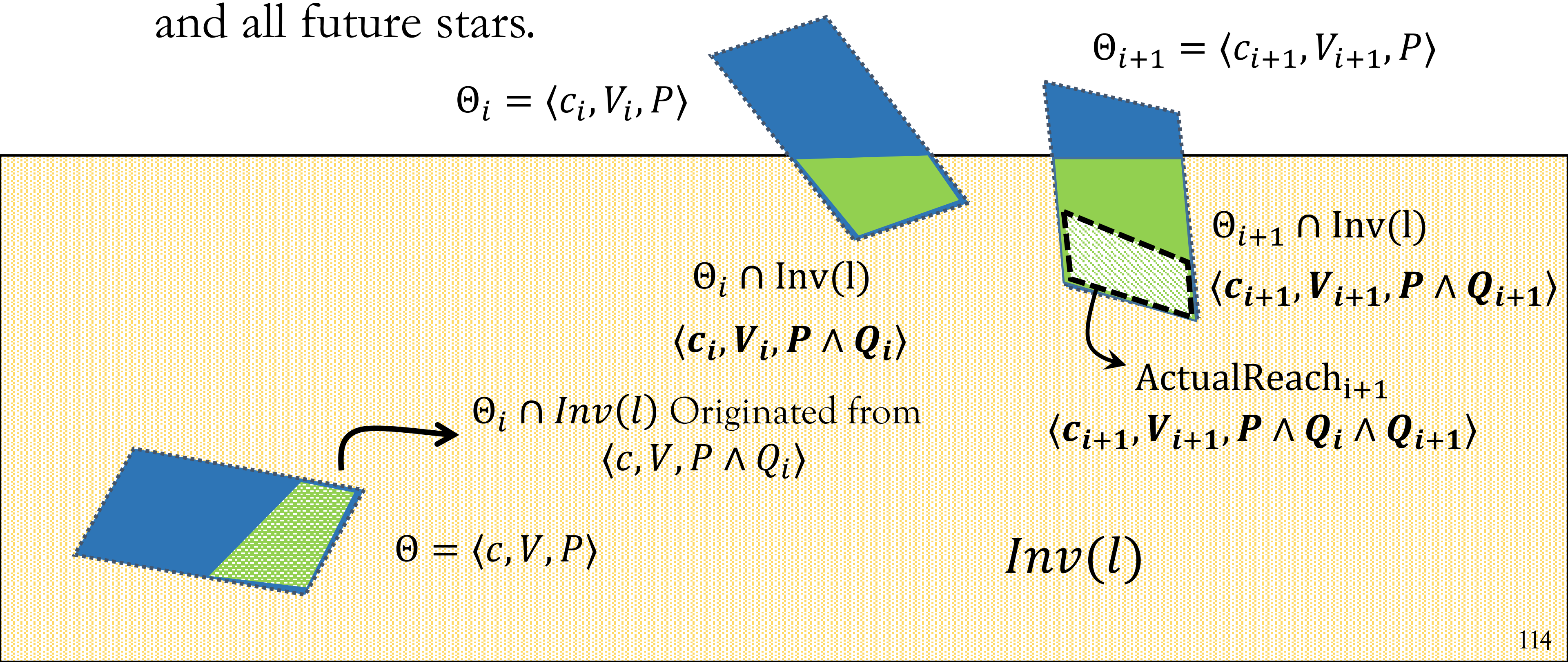
Forward Constraint Propagation

1. Convert Inv into the center and basis of i^{th} star as $\langle c_i, V_i, Q_i \rangle$.
2. $\Theta \cap Inv = \langle c_i, V_i, P \wedge Q_i \rangle$
3. These should originate from $\langle c, V, P \wedge Q_i \rangle$ in Θ



Forward Constraint Propagation

1. Convert Inv into the center and basis of i^{th} star as $\langle c_i, V_i, Q_i \rangle$.
2. $\Theta \cap Inv = \langle c_i, V_i, P \wedge Q_i \rangle$
3. These should originate from $\langle c, V, P \wedge Q_i \rangle$ in Θ
4. Propagate constraint Q_i forward --- add it to predicates of itself and all future stars.



Invariant Constraint Propagation

1. Compute reachable sets $\Theta_1, \Theta_2, \ldots, \Theta_k$.

2. Convert Inv into star representation of Θ_i as $\langle c_1, V_1, Q_1 \rangle, \langle c_2, V_2, Q_2 \rangle, \ldots, \langle c_k, V_k, Q_k \rangle$

3. Add constraint Q_i to the predicate of $\Theta_i, \Theta_{i+1}, \ldots, \Theta_k$.

Invariant Constraint Propagation

1. Compute reachable sets $\Theta_1, \Theta_2, \ldots, \Theta_k$.
2. Convert Inv into star representation of Θ_i as $\langle c_1, V_1, Q_1 \rangle, \langle c_2, V_2, Q_2 \rangle, \ldots, \langle c_k, V_k, Q_k \rangle$
3. Add constraint Q_i to the predicate of $\Theta_i, \Theta_{i+1}, \ldots, \Theta_k$.

Isn't this expensive?

Invariant Constraint Propagation

1. Compute reachable sets $\Theta_1, \Theta_2, \ldots, \Theta_k$.
2. Convert Inv into star representation of Θ_i as $\langle c_1, V_1, Q_1 \rangle, \langle c_2, V_2, Q_2 \rangle, \ldots, \langle c_k, V_k, Q_k \rangle$
3. Add constraint Q_i to the predicate of $\Theta_i, \Theta_{i+1}, \ldots, \Theta_k$.

No. of constraints increase linearly with time?

Isn't this expensive?

Optimizations

1. If $\Theta_i \subseteq Inv$, then $P \wedge Q_i \equiv P$. Hence, no constraint is added.

2. If $\Theta_i \subseteq Inv^c$, then $P \wedge Q_i \equiv \bot$. Hence, no need to add Q_i.

Optimizations

1. If $\Theta_i \subseteq Inv$, then $P \land Q_i \equiv P$. Hence, no constraint is added.

2. If $\Theta_i \subseteq Inv^c$, then $P \land Q_i \equiv \bot$. Hence, no need to add Q_i.

3. Add a constraint Q_i to $P \land Q_1 \land \cdots \land Q_{i-1}$ if and only if

$$\neg(P \land Q_1 \land \cdots \land Q_{i-1} \Rightarrow Q_i)$$

Optimizations

1. If $\Theta_i \subseteq Inv$, then $P \wedge Q_i \equiv P$. Hence, no constraint is added.

2. If $\Theta_i \subseteq Inv^c$, then $P \wedge Q_i \equiv \bot$. Hence, no need to add Q_i.

3. Add a constraint Q_i to $P \wedge Q_1 \wedge \cdots \wedge Q_{i-1}$ if and only if
$$\neg(P \wedge Q_1 \wedge \cdots \wedge Q_{i-1} \Rightarrow Q_i)$$

4. **[Empirical heuristic]:** Compare successive constraints Q_i and Q_{i+1} and if Q_{i+1} is stronger than Q_i, replace Q_i with Q_{i+1}.

Discrete Transitions

- Discrete transitions are enabled when the reachable set overlaps with the guard condition.

- If reachable set from Θ overlaps with guard G_i at $\Theta_{i,1}, \Theta_{i,2}, \ldots, \Theta_{i,l}$. That is, Θ has l successor sets.

- After m discrete transitions, the number of sets to keep track will be l^m. (exponential blow-up).

Discrete Transitions

- Discrete transitions are enabled when the reachable set overlaps with the guard condition.
- If reachable set from Θ overlaps with guard G_i at $\Theta_{i,1}, \Theta_{i,2}, \ldots, \Theta_{i,l}$. That is, Θ has l successor sets.
- After m discrete transitions, the number of sets to keep track will be l^m. (exponential blow-up).

Solution: Aggregation

Aggregation – A Necessary Evil

- Necessary to reduce the number of sets to keep track of.

Aggregation – A Necessary Evil

- Necessary to reduce the number of sets to keep track of.
- Aggregation introduces overapproximation that we can never get rid of!
- Might cause spurious discrete transitions; cannot give concrete counterexamples.

Aggregation – A Necessary Evil

- Necessary to reduce the number of sets to keep track of.
- Aggregation introduces overapproximation that we can never get rid of!
- Might cause spurious discrete transitions; cannot give concrete counterexamples.

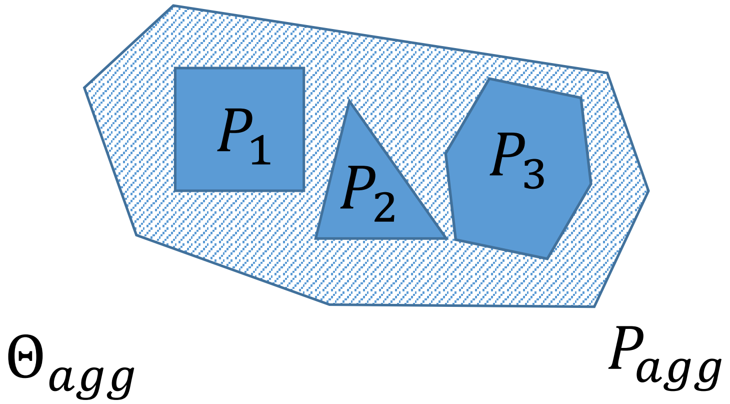
Aggregation – A Necessary Evil

- Necessary to reduce the number of sets to keep track of.
- Aggregation introduces overapproximation that we can never get rid of!
- Might cause spurious discrete transitions; cannot give concrete counterexamples.

Damned if you do!
Damned if you don’t!

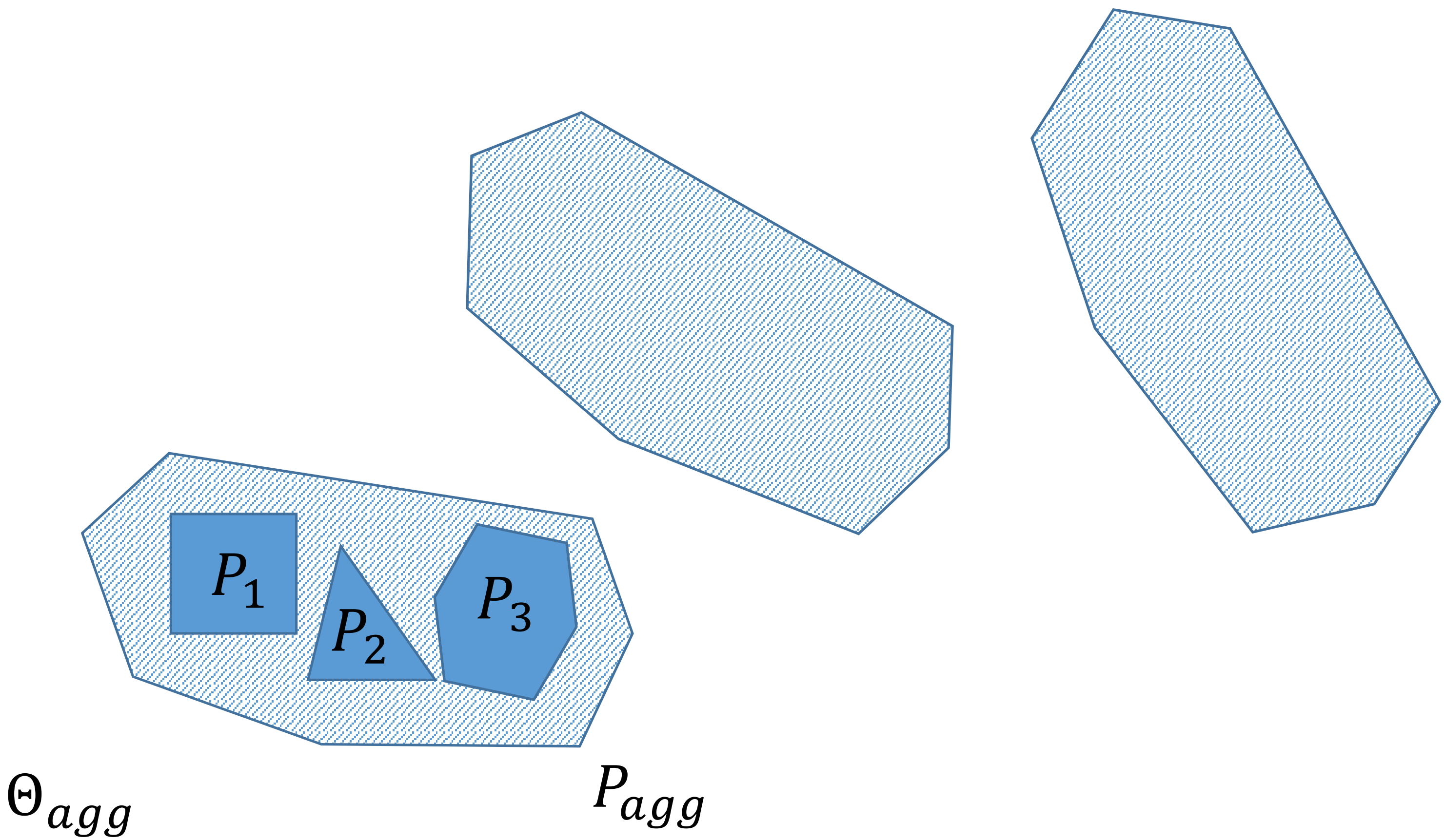
Dynamic Aggregation Illustration

1. Aggregate all the sets by default and compute reachable set.



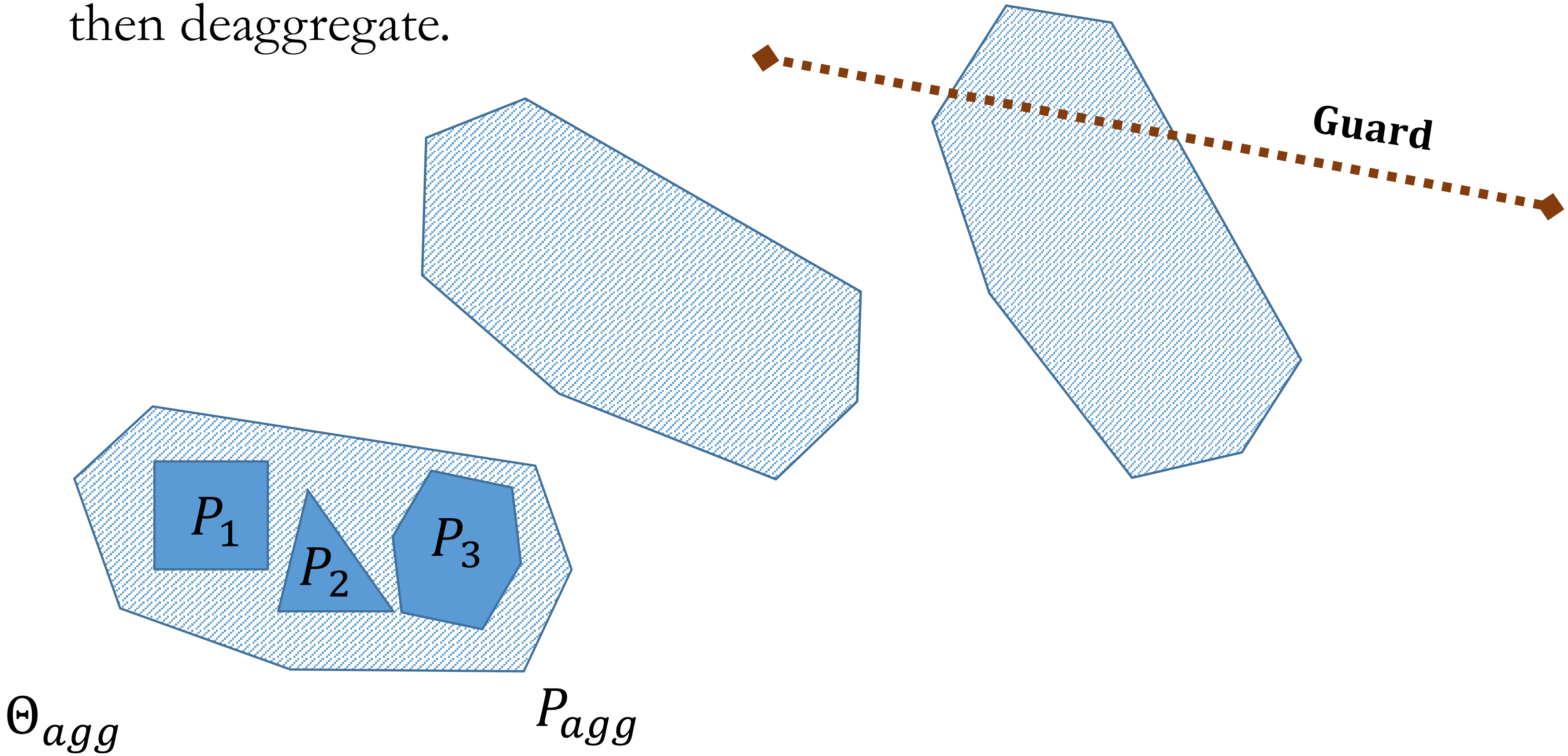
Dynamic Aggregation Illustration

1. Aggregate all the sets by default and compute reachable set.



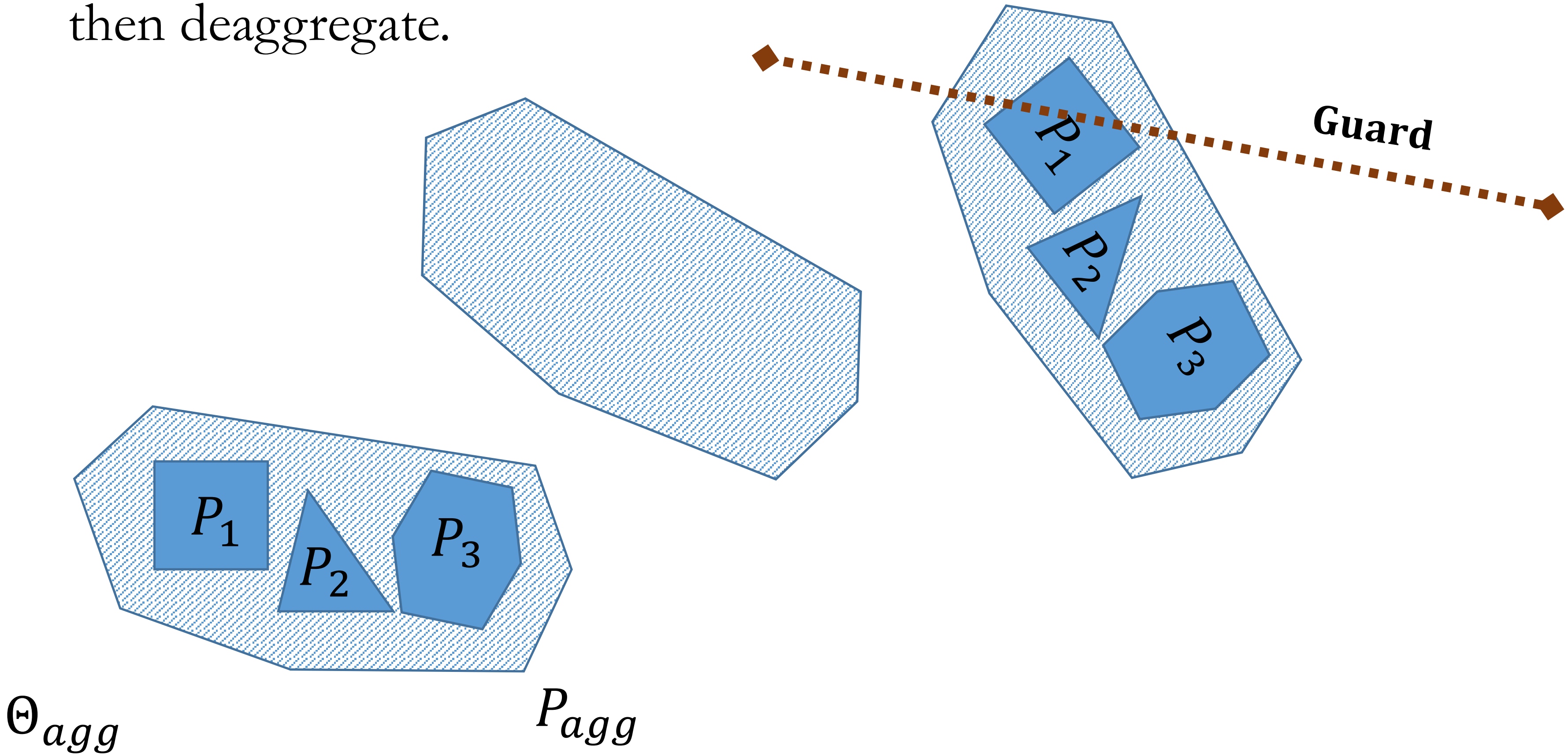
Dynamic Aggregation Illustration

1. Aggregate all the sets by default and compute reachable set.
2. When the aggregated set intersects with a guard or unsafe set, then deaggregate.



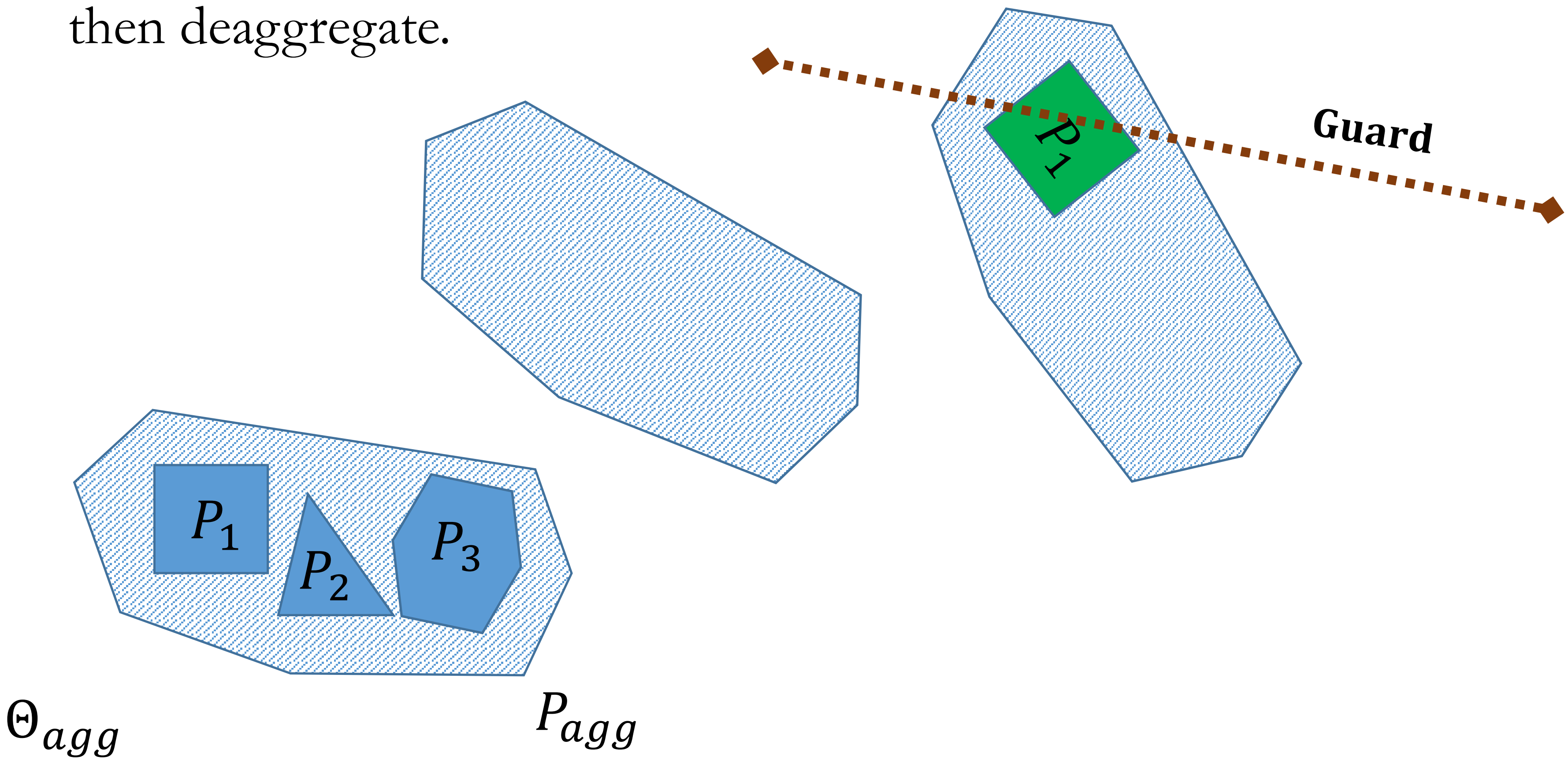
Dynamic Aggregation Illustration

1. Aggregate all the sets by default and compute reachable set.
2. When the aggregated set intersects with a guard or unsafe set, then deaggregate.



Dynamic Aggregation Illustration

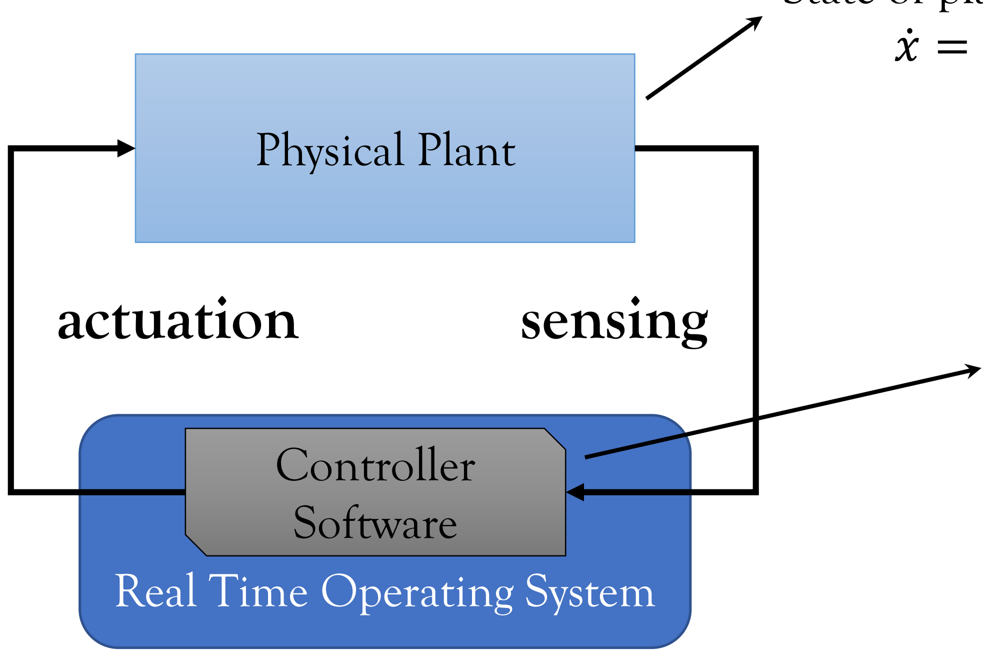
1. Aggregate all the sets by default and compute reachable set.
2. When the aggregated set intersects with a guard or unsafe set, then deaggregate.



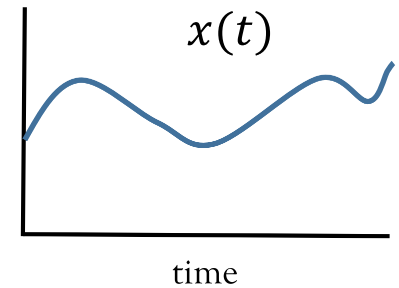
Model + Real-Time Operating Systems Behavior

Analyzing Real Time Linear Control Systems Using Software Verification

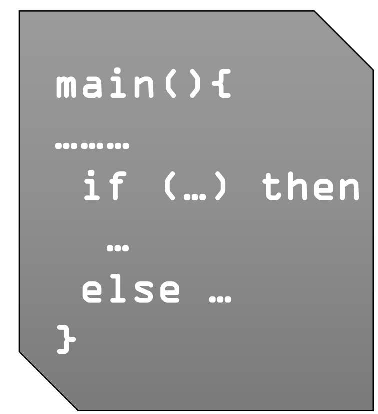
D, Viswanathan RTSS 2015



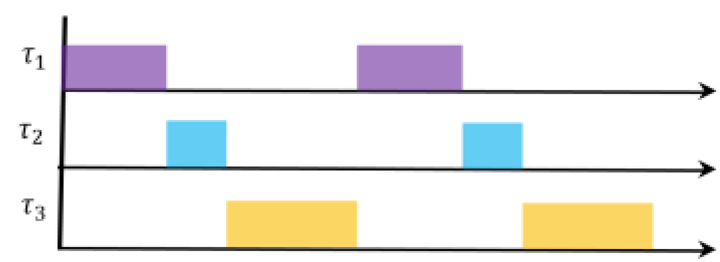
State of plant x evolves as
$\dot{x} = Ax + Bu$



Code



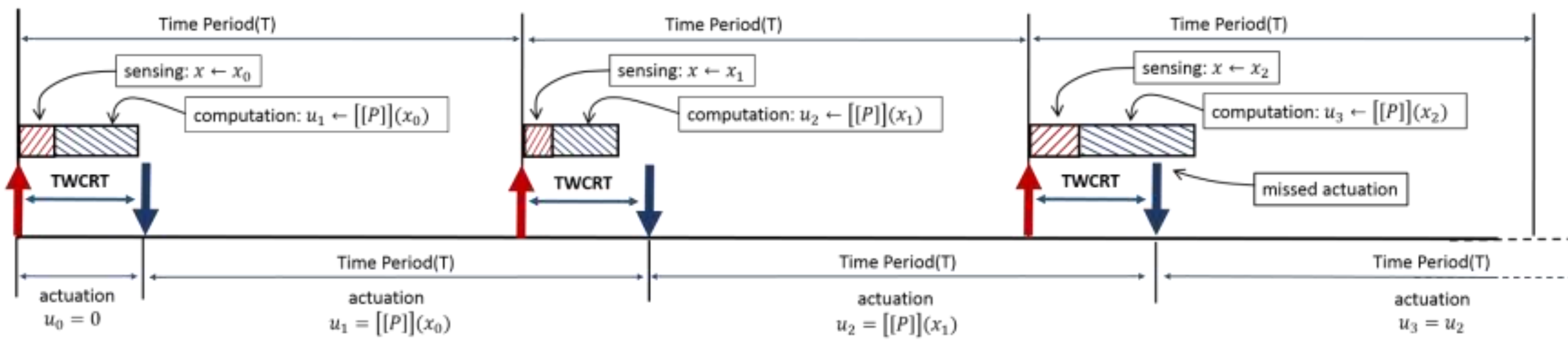
Scheduling



Verification that takes all the three aspects into account

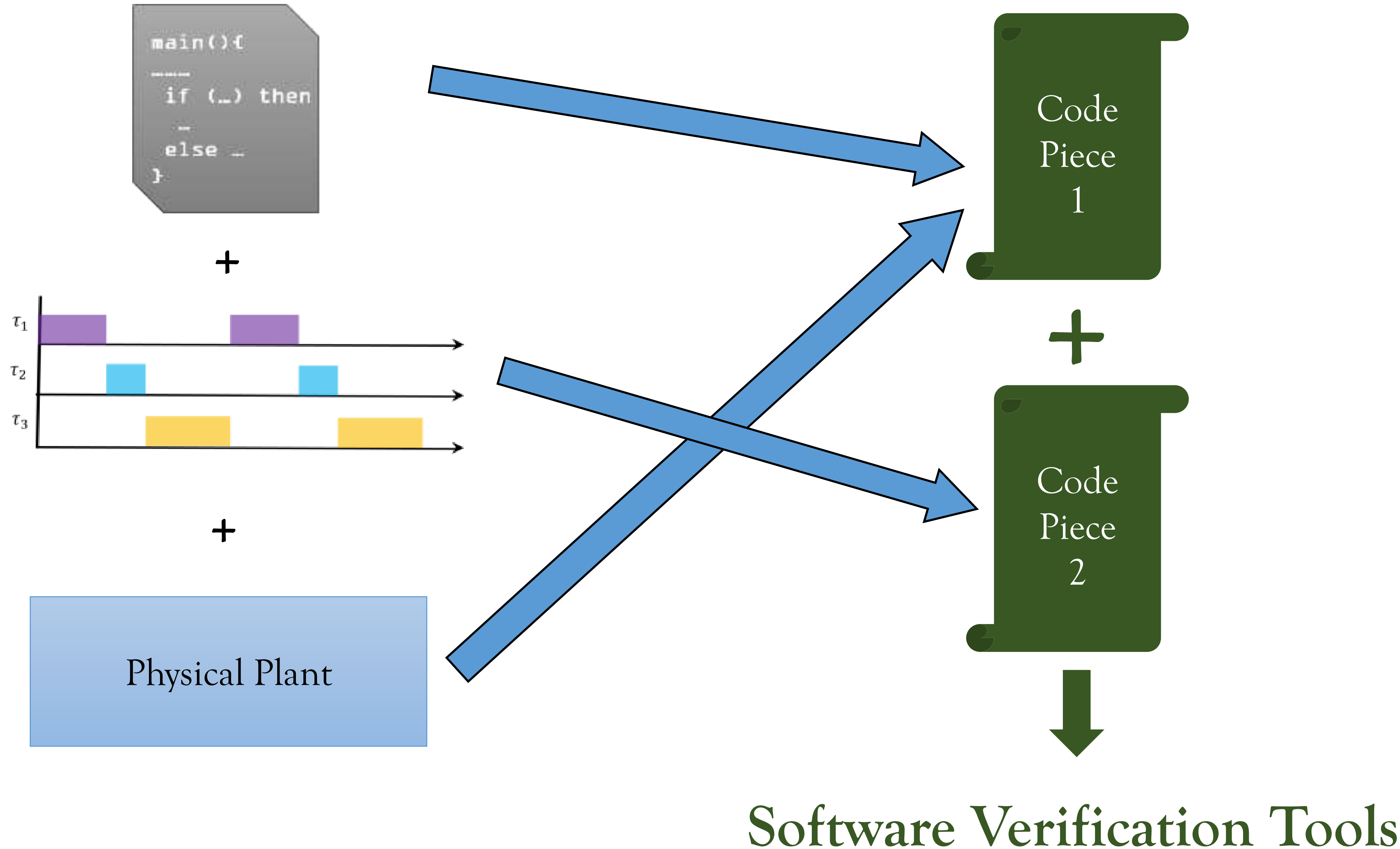
Computational Model

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**.

Software Verification Inspired Technique: Outline



Bringing These Two Together

Code Piece 1 + Code Piece 2 =

Controller code

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

Timing Behavior

```
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;
```

Updating actuation only when deadline is met

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

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;
```