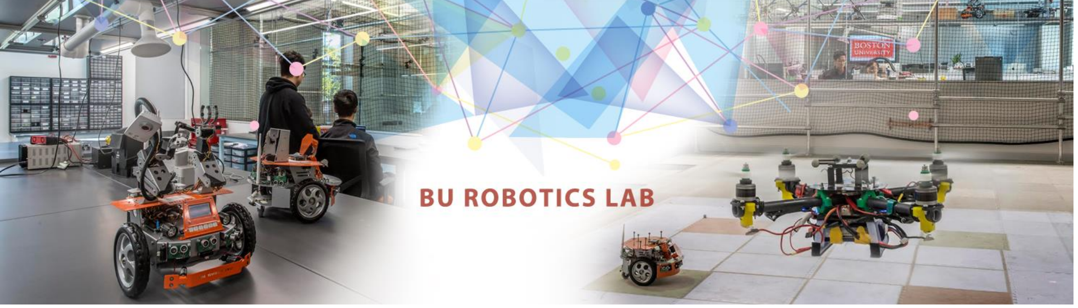
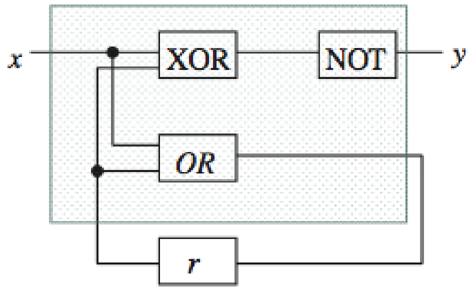


Formal Methods for Dynamical Systems

Calin Belta

Tegan Family Distinguished Professor
Mechanical Engineering, Systems Engineering,
Electrical and Computer Engineering
Boston University

A classical problem in formal methods



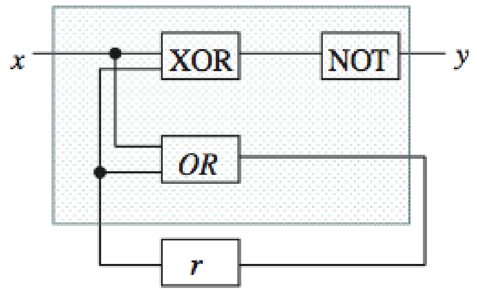
Process

C. Bayer and J-P Katoen, Principles of Model Checking, MIT Press, 2008

A classical problem in formal methods

Specification: “If x is set infinitely often, then y is set infinitely often.”

Process

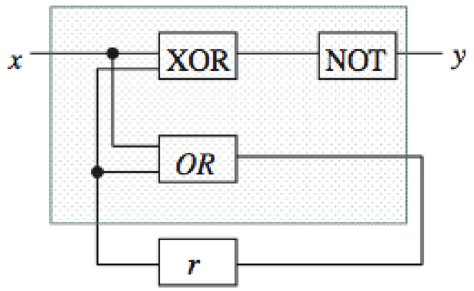


A classical problem in formal methods

Specification: "If x is set infinitely often, then y is set infinitely often."

Check if all the possible behaviors of the circuit satisfy the specification

Process



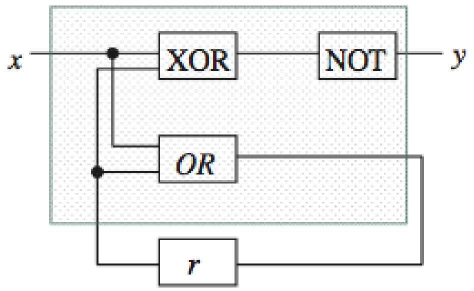
A classical problem in formal methods

Specification: "If x is set infinitely often, then y is set infinitely often."

Formalization

Temporal Logic Formula $\Box\Diamond x \rightarrow \Box\Diamond y$

Process



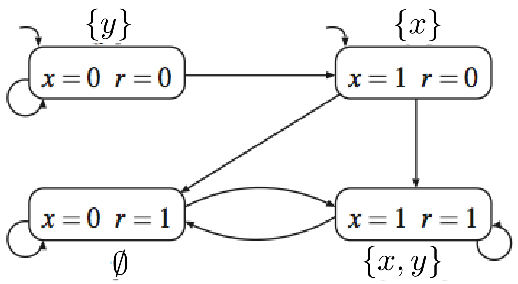
A classical problem in formal methods

Specification: “If x is set infinitely often, then y is set infinitely often.”

Formalization

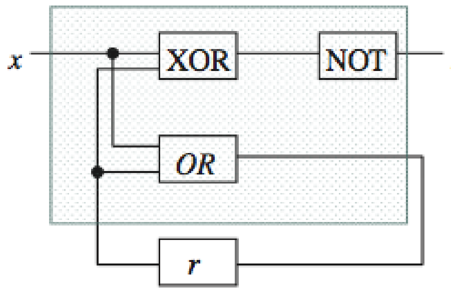
Temporal Logic Formula $\Box\Diamond x \rightarrow \Box\Diamond y$

Model



Mathematical modeling

Process



C. Bayer and J-P Katoen, Principles of Model Checking, MIT Press, 2008

A classical problem in formal methods

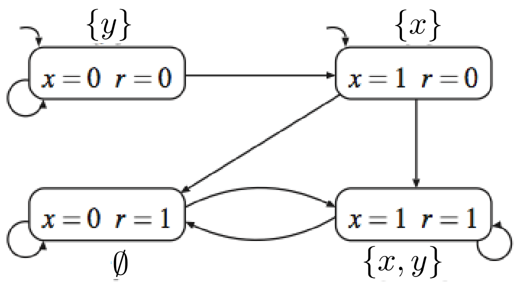
Specification: “If x is set infinitely often, then y is set infinitely often.”

Formalization

Temporal Logic Formula $\Box\Diamond x \rightarrow \Box\Diamond y$

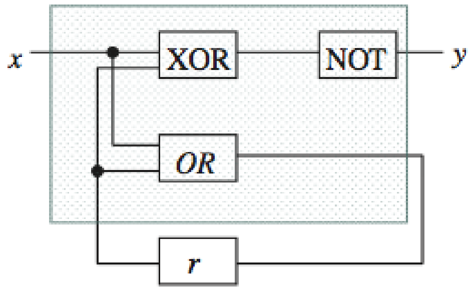
Model checking (verification)

Model



Mathematical modeling

Process



A classical problem in dynamical systems

Process

S. Sastry – Nonlinear Systems: analysis, stability, and control, Springer, 1999

A classical problem in dynamical systems

Specification: "drive from A to B."

Process

B

A

A classical problem in dynamical systems

Specification: "drive from A to B."

Generate a robot control strategy

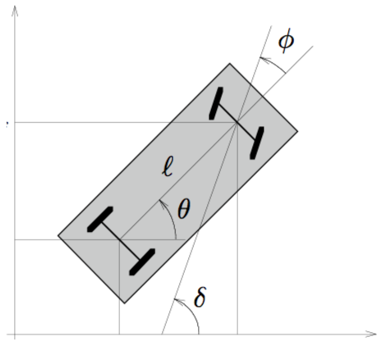
Process

A classical problem in dynamical systems

Specification: “drive from A to B.”

Model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2$$



Mathematical modeling

Process

S. Sastry – Nonlinear Systems: analysis, stability, and control, Springer, 1999

A classical problem in dynamical systems

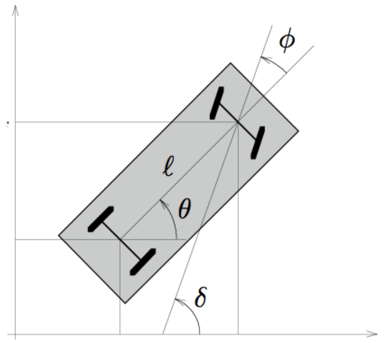
Specification: "drive from A to B."

Formalization

Stabilization Problem: "make B an asymptotically stable equilibrium"

Model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2$$



Mathematical modeling

Process

A classical problem in dynamical systems

Specification: "drive from A to B."

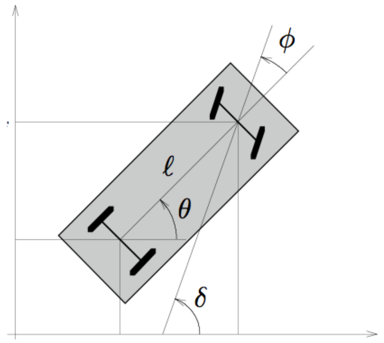
Formalization

Stabilization Problem: "make B an asymptotically stable equilibrium"

Control

Model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2$$



Mathematical modeling

Process

S. Sastry – Nonlinear Systems: analysis, stability, and control, Springer, 1999

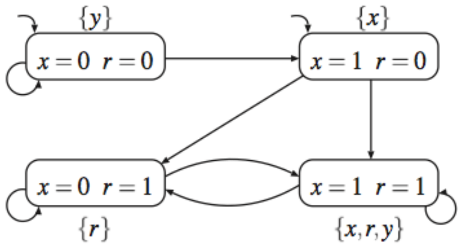
Formal methods vs. dynamical systems

Specification

"If x is set infinitely often, then y is set infinitely often."

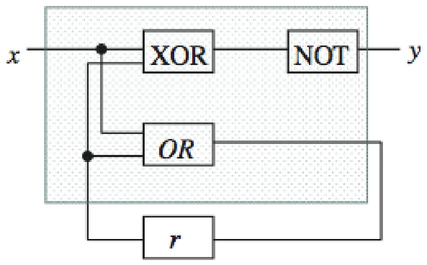
"Drive from A to B."

Model

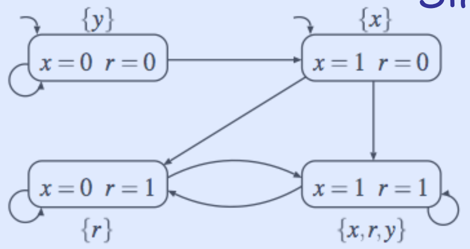
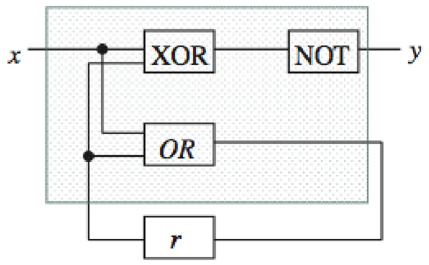


$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2$$

Process

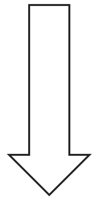


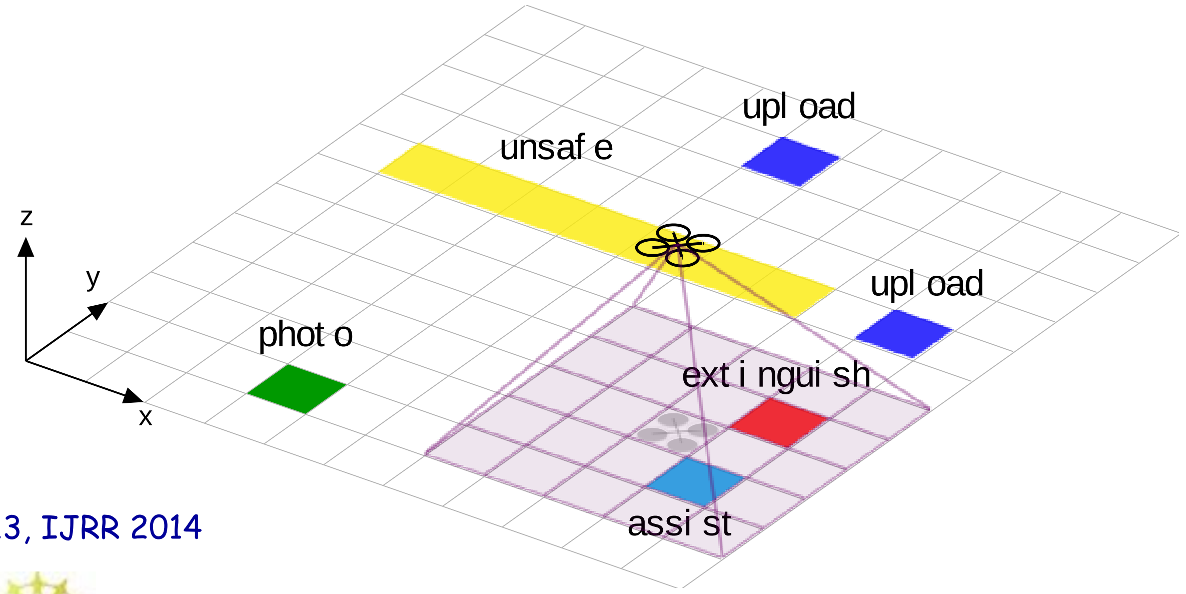
Formal methods vs. dynamical systems

Specification	"If x is set infinitely often, then y is set infinitely often." Complex	"Drive from A to B." Simple
Model	Simple  $\{y\}$ $\{x\}$ $x=0\ \ r=0$ $x=1\ \ r=0$ $x=0\ \ r=1$ $x=1\ \ r=1$ $\{r\}$ $\{x,r,y\}$	Complex $\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\\ \tan\phi/\ell\\ 0\end{bmatrix} v_1 + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix} v_2$
Process		

Need for formal methods in dynamical systems

Spec: **Off-line:** "Keep taking photos and upload current photo before taking another photo. **On-line:** Unsafe regions should always be avoided. If fires are detected, then they should be extinguished. If survivors are detected, then they should be provided medical assistance. If both fires and survivors are detected locally, priority should be given to the survivors."



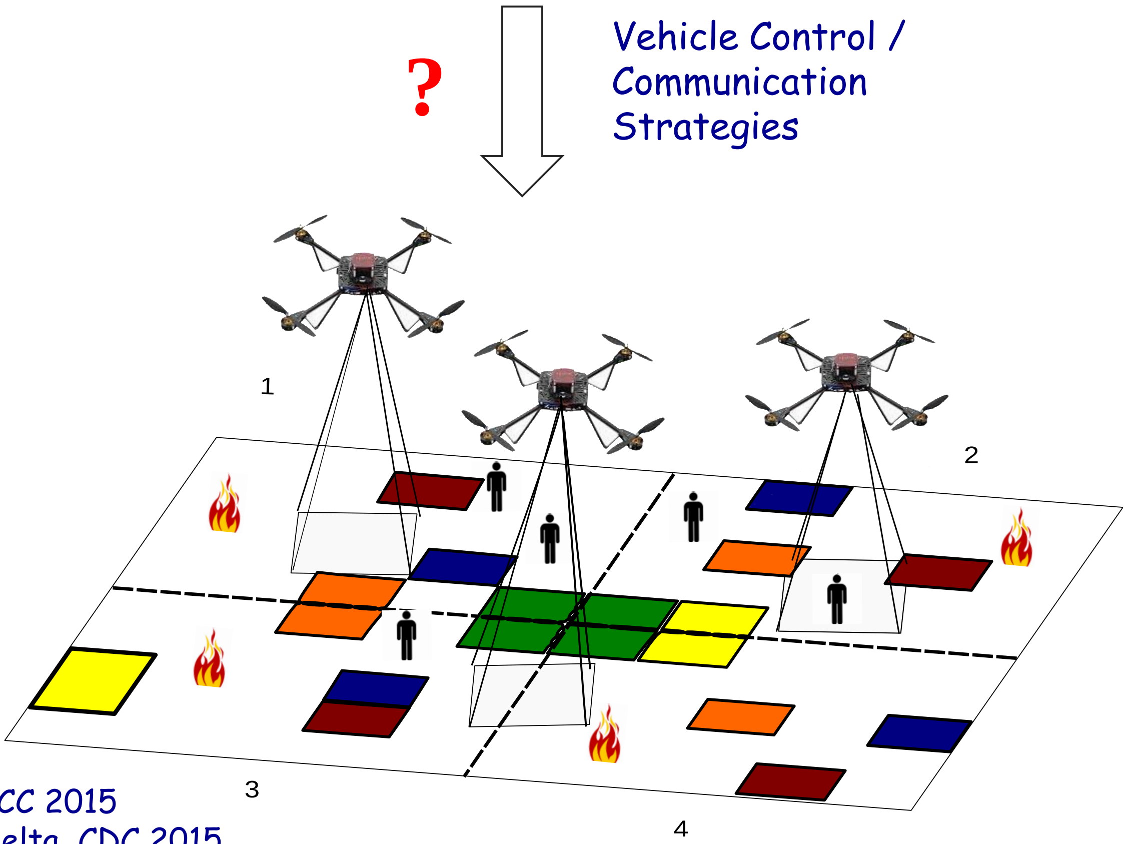


Ulusoy, Belta, RSS 2013, IJRR 2014

Solution later in this talk

Need for formal methods in dynamical systems

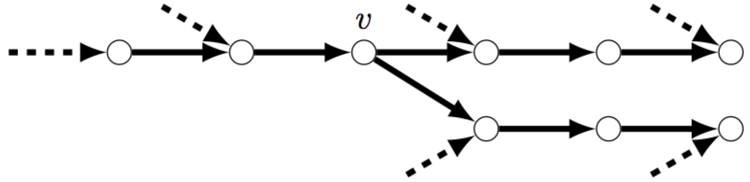
Mission Specification: " **If** a fire **or** survivor are located with **enough certainty**, **then** take photos **and next** upload them at upload regions (blue). **Always** avoid obstacles (red regions). Type 1 (orange) **or** Type 2 (yellow) radiations area allowed, but **not both**. **After** all fires have been localized with **enough certainty and** the data has been uploaded, return to recharging stations (green) and wait for redeployment. Minimize overall distance travelled."

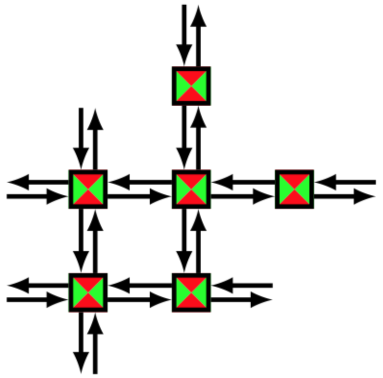


Jones, Schwager, Belta, ACC 2015
Leahy, Jones, Schwager, Belta, CDC 2015

Not in this talk

Need for formal methods in dynamical systems



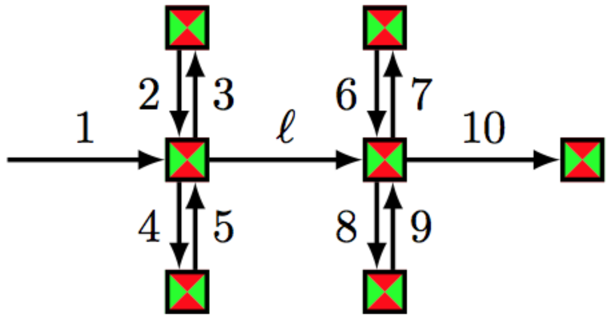


- eventually each link will have ≤30 vehicles
- upstream link will have low demand until downstream link is no longer congested
- each queue at a junction will be actuated at least once every two minutes
- whenever the number of vehicles on link l exceeds C1, it is eventually the case that the number of vehicles on link l decreases below C2."

?
Traffic light
control strategy

Coogan, et. al., ACC 2015, IEEE TCNS 2016
Sadradini, Belta, ACC 2016
Coogan, Arcak, Belta, ACC 2016

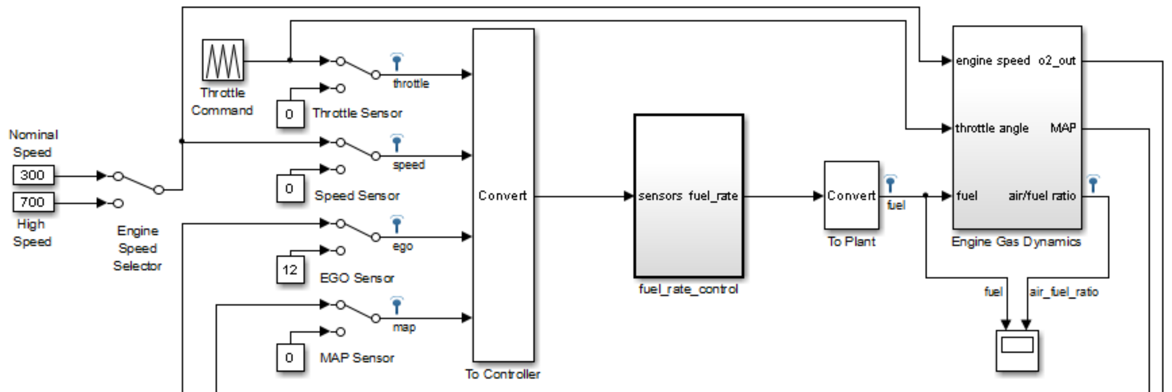
Not in this talk

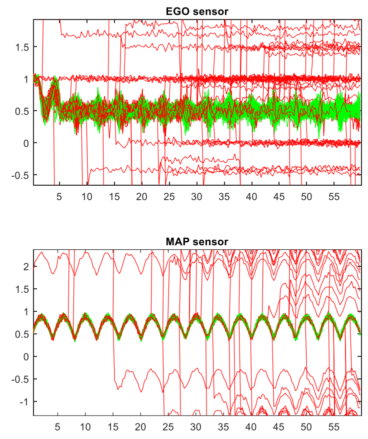


Need for formal methods in dynamical systems

1. Off-line / on-line data collection

Fuel Control System





2. Supervised / unsupervised learning (good behavior)

$$F_{[0,60)}((G_{[9.7,59.7)}x_3 < 0.875) \wedge (G_{[0.1,59.7)}x_4 < 0.98) \wedge (G_{[0.5,59.7)}x_4 > 0.29))$$

i.e., "EGO is less than 0.875 for all times in between 9.7s and 59.7s and MAP is less than 0.98 for all times in between 0.1s and 59.7s and MAP is greater than 0.29 for all times in between 0.5s and 59.7s.

3. Monitoring and anomaly mitigation

DENSO Corporation, Japan DENSO

Jones, et.al., CDC 2014
Kong, et.al., HSCC 2014
Bombara, et.al., HSCC 2016

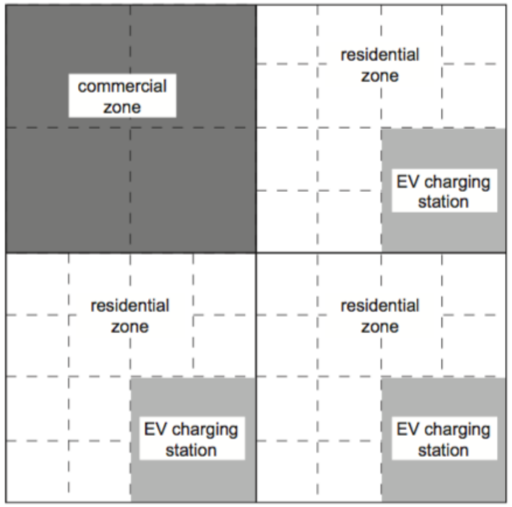
Not in this talk

Need for formal methods in dynamical systems

1. Off-line / on-line data collection

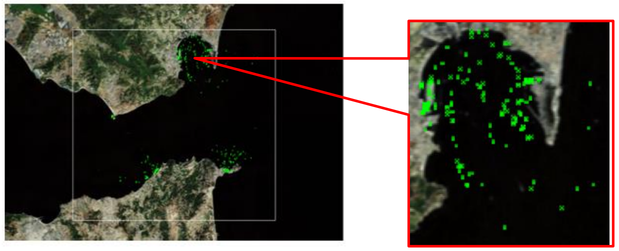
2. Supervised / unsupervised learning (good behavior)

"Always, for each of the four 'neighborhoods', the power consumption level m is below 300 and the power consumption is below 200 in each of the neighborhoods' quadrants at least once per hour. After 6 hours, the power consumption in all residential areas is above level 3."



$$\begin{aligned}\Phi_3 := \quad & G_{[0,18)}F_{[0,1)}(\forall_{(NW,NE,SW,SE)} \bigcirc (m \leq 300 \wedge \\ & \forall_{(NW,NE,SW,SE)} \bigcirc m \leq 200)) \wedge \\ & G_{[6,18)}(\forall_{(NE,SE,SW)} \bigcirc \forall_{(NW,NE,SW)} \bigcirc m \geq 3).\end{aligned}$$

3. Monitoring and anomaly mitigation



Low Earth Orbit (LEO) satellites can gather temporal-spatial data (the figure shows the intense-traffic Strait of Gibraltar)

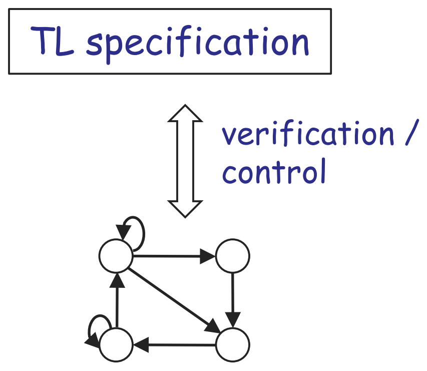
Haghighi, et.al., HSCC 2015

Not in this talk

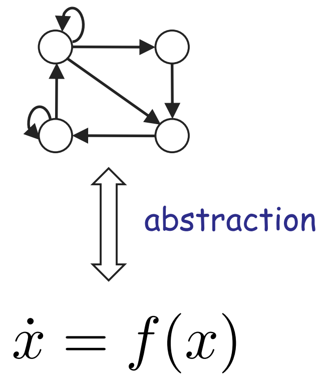
Outline

Verification and control for finite systems

Conservative control for dynamical systems

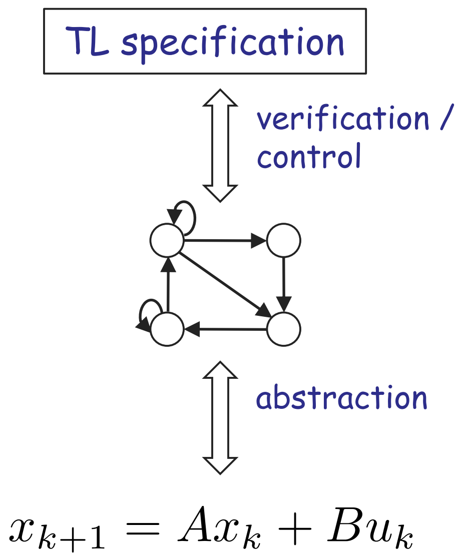


Finite quotients of continuous-space systems: main ideas



Verification for discrete-time linear systems

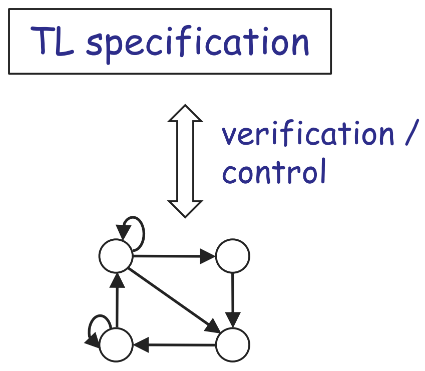
Control for discrete-time linear systems



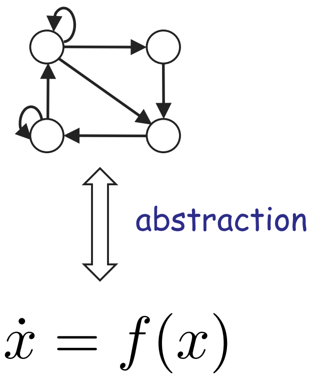
Outline

Verification and control for finite systems

Conservative control for dynamical systems

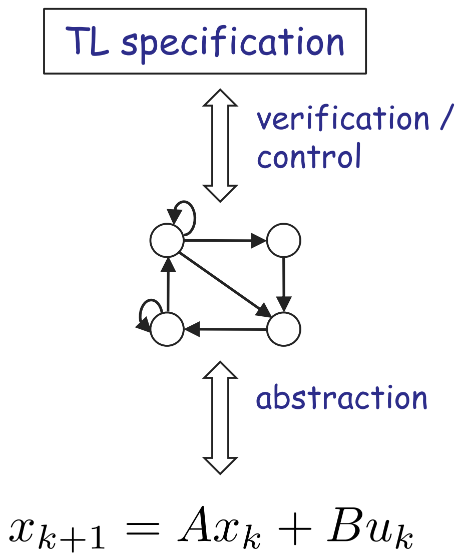


Finite quotients of continuous-space systems: main ideas



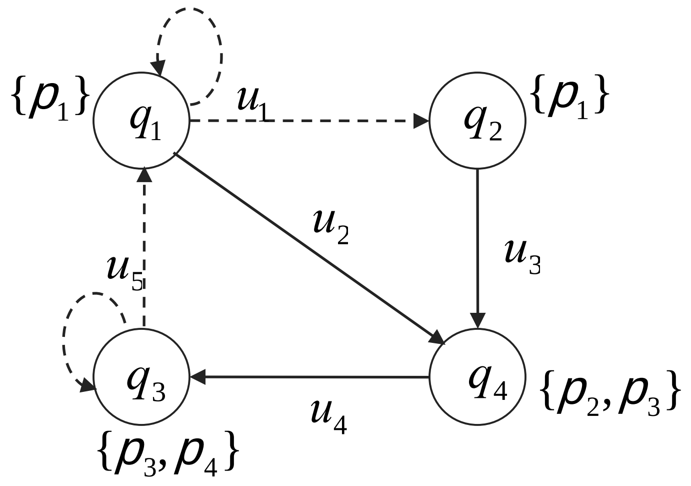
Verification for discrete-time linear systems

Control for discrete-time linear systems



Verification and control for finite systems

(Fully-observable) nondeterministic (non-probabilistic) labeled transition systems with finitely many states and actions and fully observable state



Verification and control for finite systems

Linear Temporal Logic (LTL)

Syntax

$$\underset{\text{eventually}}{\Diamond}\pi_2 \qquad \Diamond\underset{\text{always}}{\Box}(\pi_3 \wedge \pi_4) \qquad (\pi_1 \vee \pi_2)\underset{\text{until}}{U}\pi_4$$

Verification and control for finite systems

Linear Temporal Logic (LTL)

Syntax

$$\Diamond \pi_2 \qquad \Diamond \Box (\pi_3 \wedge \pi_4) \qquad (\pi_1 \vee \pi_2) U \pi_4$$

eventually always until

Semantics

Word: $\{\pi_1\}\{\pi_2, \pi_3\}\{\pi_3, \pi_4\}\{\pi_3, \pi_4\} \cdots$

Verification and control for finite systems

Linear Temporal Logic (LTL)

Syntax

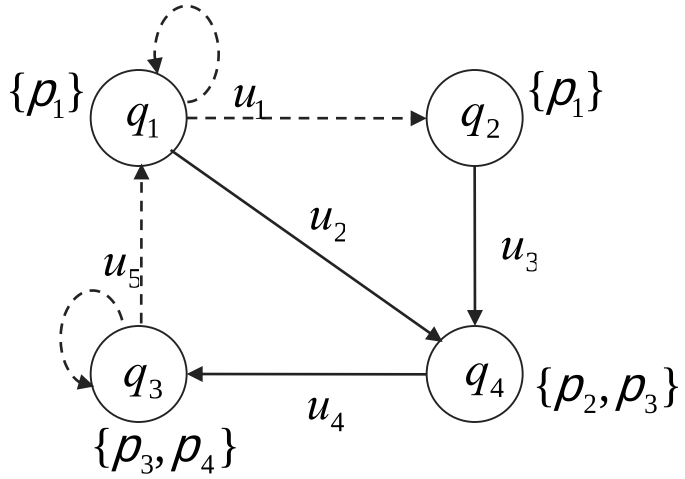
$$\Diamond \pi_2 \qquad \Diamond \Box (\pi_3 \wedge \pi_4) \qquad (\pi_1 \vee \pi_2) U \pi_4$$

eventually

always

until

Semantics



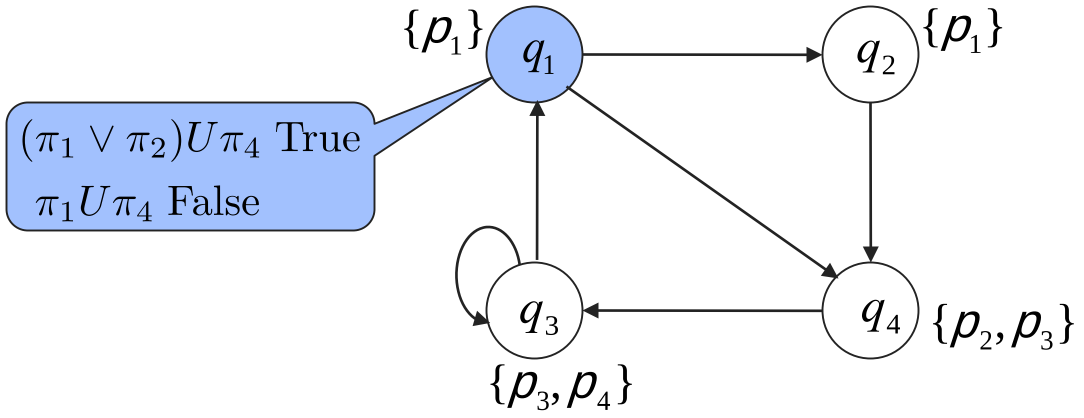
Run (trajectory): $q_1, q_4, q_3, q_3, \ldots$

Word: $\{\pi_1\}\{\pi_2, \pi_3\}\{\pi_3, \pi_4\}\{\pi_3, \pi_4\} \cdots$

Verification and control for finite systems

LTL model checking

Given a transition system and an LTL formula over its set of propositions, check if the language (i.e., all possible words) of the transition system starting from all initial states satisfies the formula.

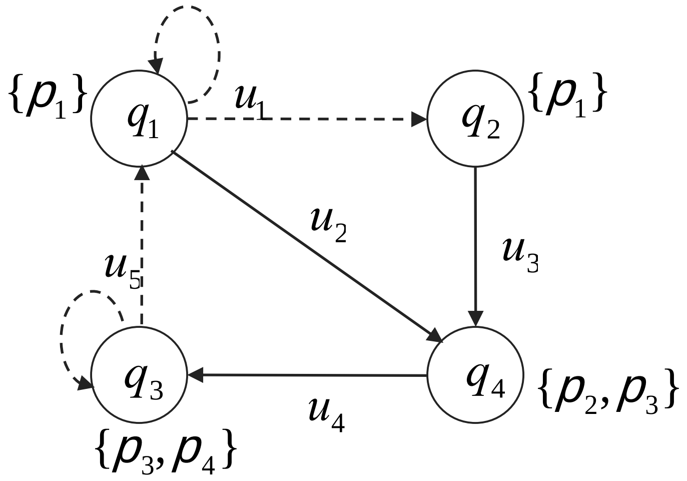


SPIN, NuSMV, PRISM, …

Verification and control for finite systems

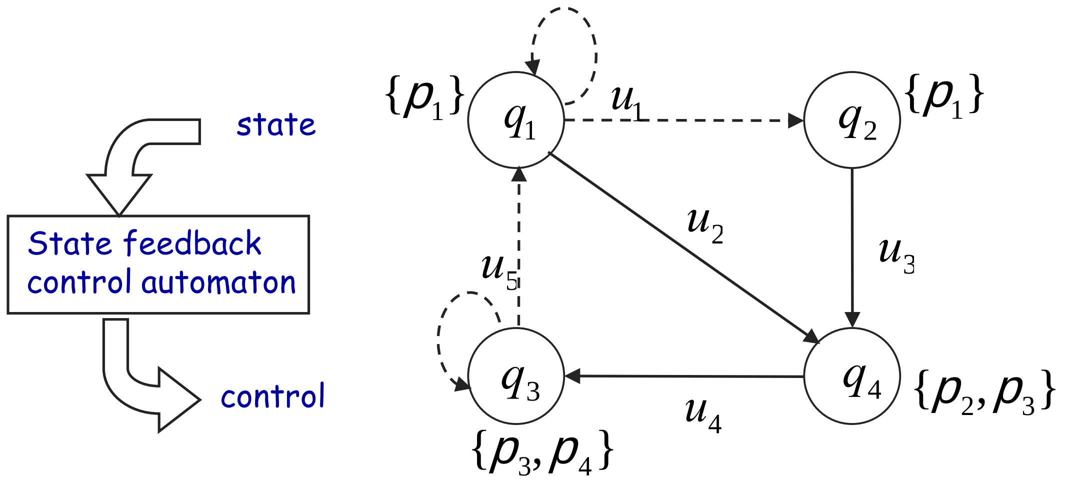
LTL control

Given a transition system and an LTL formula over its set of propositions, find a set of initial states and a control strategy for all initial states such that the produced language of the transition system satisfies the formula.



Verification and control for finite systems

LTL control



Büchi / Rabin games

Particular cases (no need to play a game)

- Deterministic systems: adapted off-the-shelf model checking
- “Finite time” LTL specs (syntactically co-safe LTL):
    - Djistra’s algorithm for deterministic systems
    - Fixed-point algorithms for non-deterministic systems

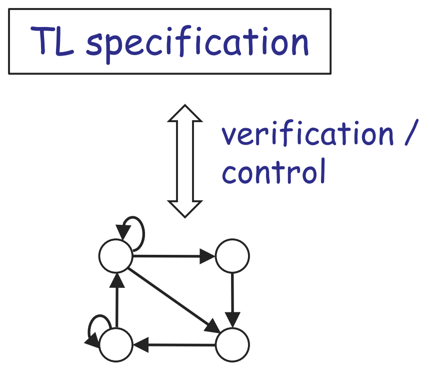
Extensions

Optimal Temporal Logic Control for Finite Deterministic Systems
Optimal Temporal Logic Control for Finite MDPs
Temporal Logic Control for POMDPs
Temporal Logic Control and Learning

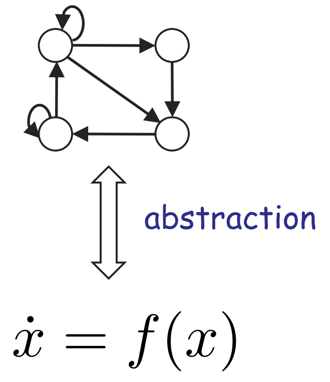
Outline

Verification and control for finite systems

Conservative control for dynamical systems

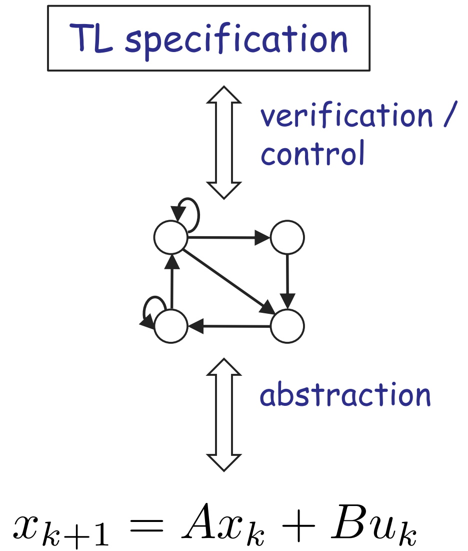


Finite quotients of continuous-space systems: main ideas



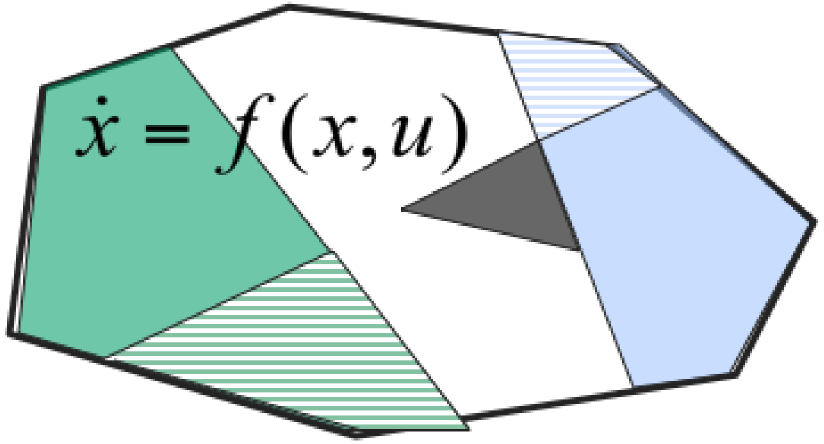
Verification for discrete-time linear systems

Control for discrete-time linear systems



Conservative Control for Dynamical Systems

1. **Conservative abstractions for simple dynamics**

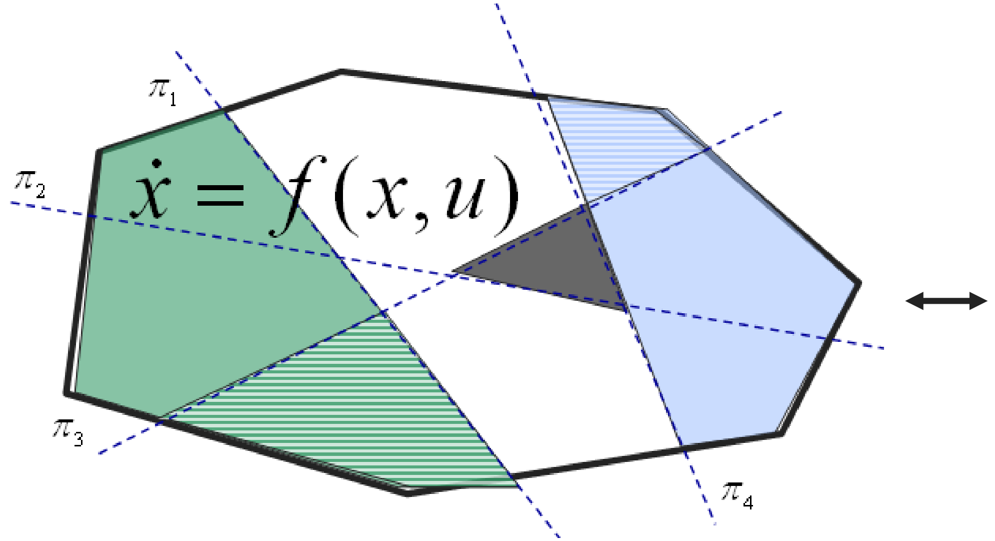


$\longleftrightarrow$ "Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order."

Conservative Control for Dynamical Systems

1. Conservative abstractions for simple dynamics

“(pi2 = TRUE and pi4 = FALSE and pi3 = FALSE) should never happen. Then pi4 = TRUE and then pi1 = TRUE should happen. After that, (pi3 = TRUE and pi4 = TRUE) and then (pi1 = TRUE and pi3 = FALSE) should occur infinitely often.”



“Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order.”

Conservative Control for Dynamical Systems

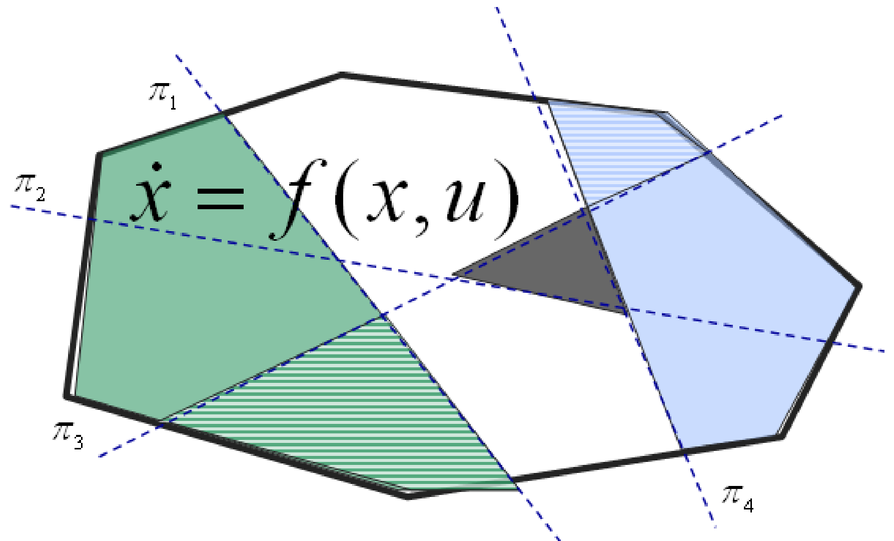
1. Conservative abstractions for simple dynamics

$$\Box\neg(\pi_2 \wedge \neg\pi_4 \wedge \neg\pi_3)) \wedge$$
$$\Diamond(\pi_4 \wedge \Diamond(\pi_1 \wedge \Diamond$$
$$(\Box\Diamond((\pi_3 \wedge \pi_4) \wedge \Diamond(\pi_1 \wedge \neg\pi_3)))))$$

$\updownarrow$

"(pi2 = TRUE and pi4 = FALSE and pi3 = FALSE) should never happen. Then pi4 = TRUE and then pi1 = TRUE should happen. After that, (pi3 = TRUE and pi4 = TRUE) and then (pi1 = TRUE and pi3 = FALSE) should occur infinitely often."

$\updownarrow$

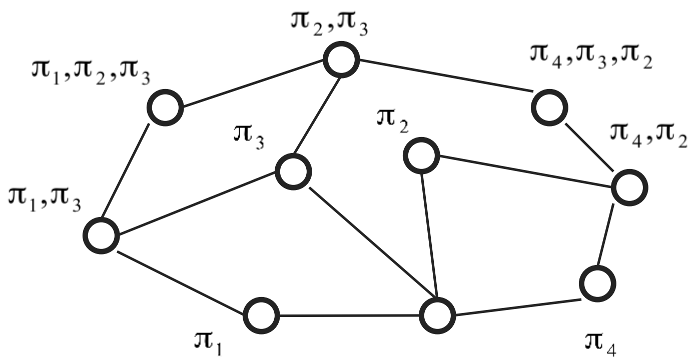


$\leftrightarrow$

"Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order."

Conservative Control for Dynamical Systems

1. Conservative abstractions for simple dynamics

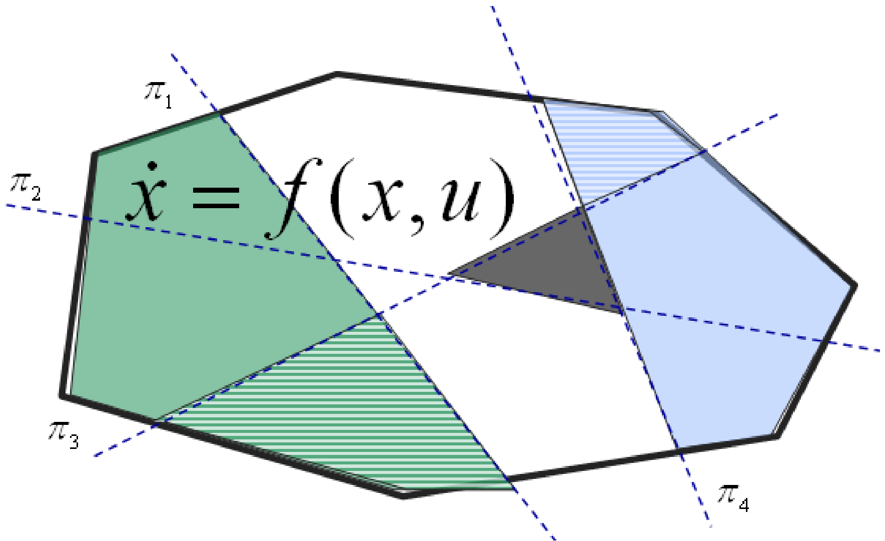


$$\Box\neg(\pi_2 \wedge \neg\pi_4 \wedge \neg\pi_3)) \wedge \Diamond(\pi_4 \wedge \Diamond(\pi_1 \wedge \Diamond (\Box\Diamond((\pi_3 \wedge \pi_4) \wedge \Diamond(\pi_1 \wedge \neg\pi_3)))))$$

$\updownarrow$

"(pi2 = TRUE and pi4 = FALSE and pi3 = FALSE) should never happen. Then pi4 = TRUE and then pi1 = TRUE should happen. After that, (pi3 = TRUE and pi4 = TRUE) and then (pi1 = TRUE and pi3 = FALSE) should occur infinitely often."

$\updownarrow$

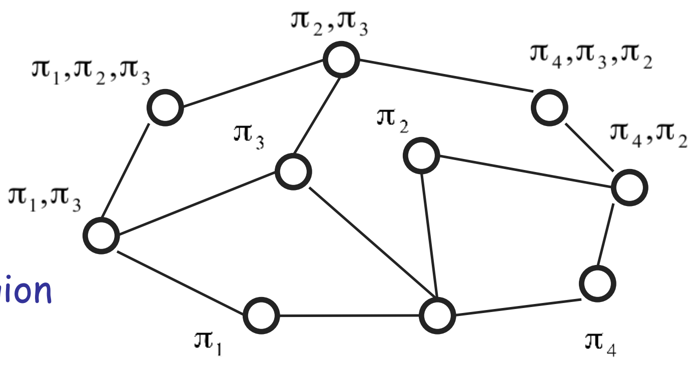


$\leftrightarrow$

"Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order."

Conservative Control for Dynamical Systems

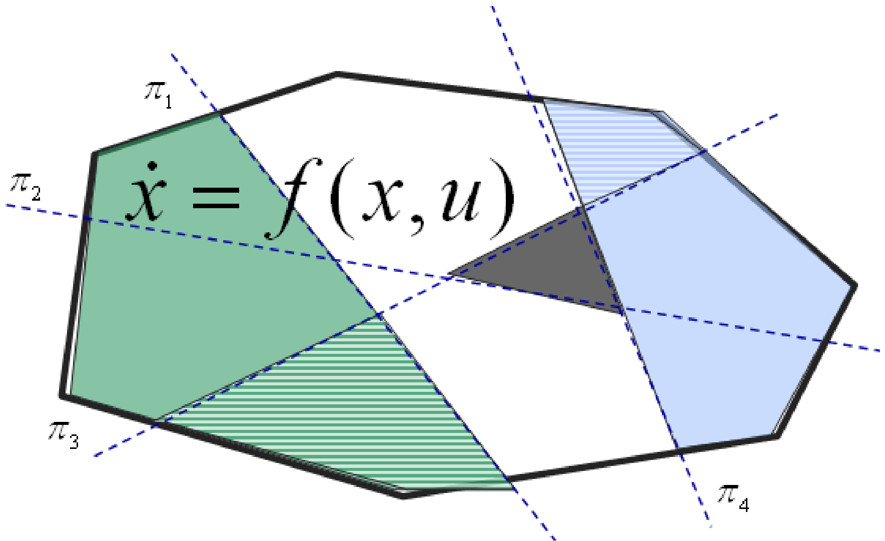
1. Conservative abstractions for simple dynamics



$$\Box\neg(\pi_2 \wedge \neg\pi_4 \wedge \neg\pi_3)) \wedge \Diamond(\pi_4 \wedge \Diamond(\pi_1 \wedge \Diamond (\Box\Diamond((\pi_3 \wedge \pi_4) \wedge \Diamond(\pi_1 \wedge \neg\pi_3)))))$$

Assume that in each region we can check for the existence of / construct feedback controllers driving all states in finite time to a subset of facets (including the empty set – controller making the region an invariant)

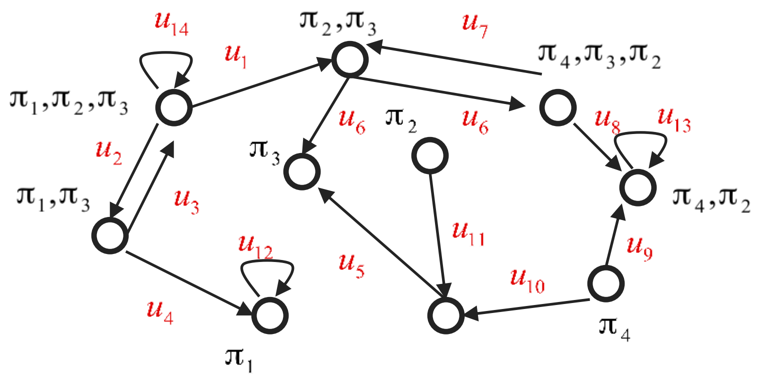
"(pi2 = TRUE and pi4 = FALSE and pi3 = FALSE) should never happen. Then pi4 = TRUE and then pi1 = TRUE should happen. After that, (pi3 = TRUE and pi4 = TRUE) and then (pi1 = TRUE and pi3 = FALSE) should occur infinitely often."



"Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order."

Conservative Control for Dynamical Systems

1. Conservative abstractions for simple dynamics

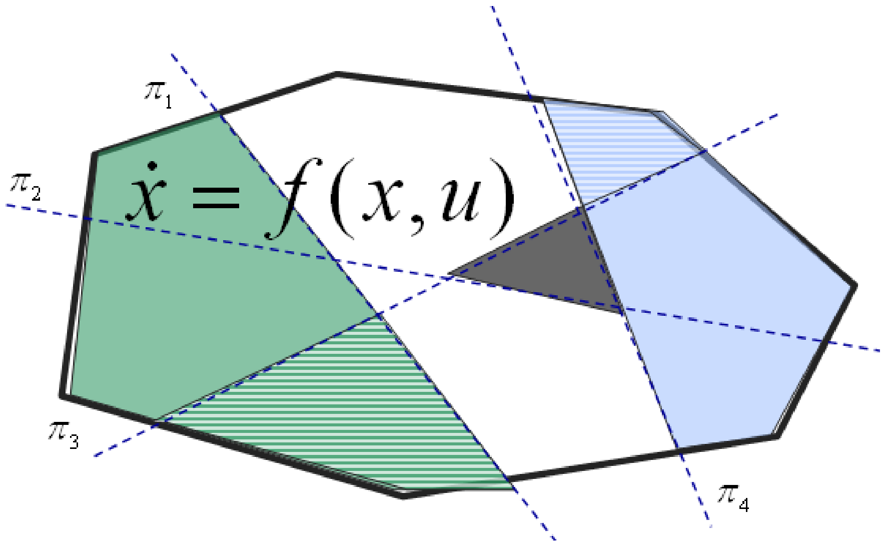


$$\Box\neg(\pi_2 \wedge \neg\pi_4 \wedge \neg\pi_3)) \wedge \Diamond(\pi_4 \wedge \Diamond(\pi_1 \wedge \Diamond (\Box\Diamond((\pi_3 \wedge \pi_4) \wedge \Diamond(\pi_1 \wedge \neg\pi_3)))))$$

$\updownarrow$

"(pi2 = TRUE and pi4 = FALSE and pi3 = FALSE) should never happen. Then pi4 = TRUE and then pi1 = TRUE should happen. After that, (pi3 = TRUE and pi4 = TRUE) and then (pi1 = TRUE and pi3 = FALSE) should occur infinitely often."

$\updownarrow$

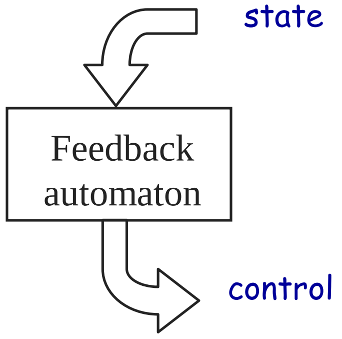


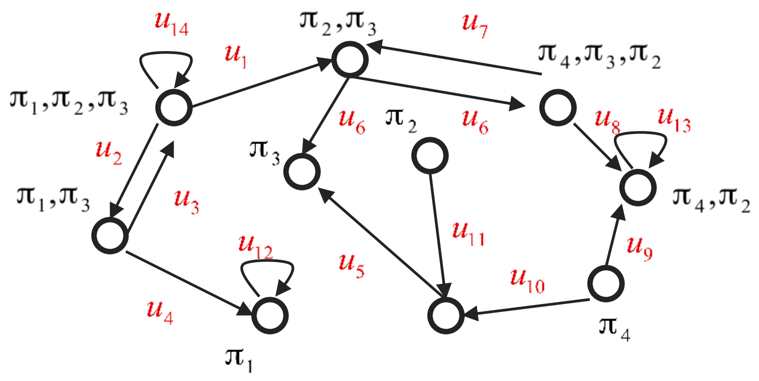
$\leftrightarrow$

"Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order."

Conservative Control for Dynamical Systems

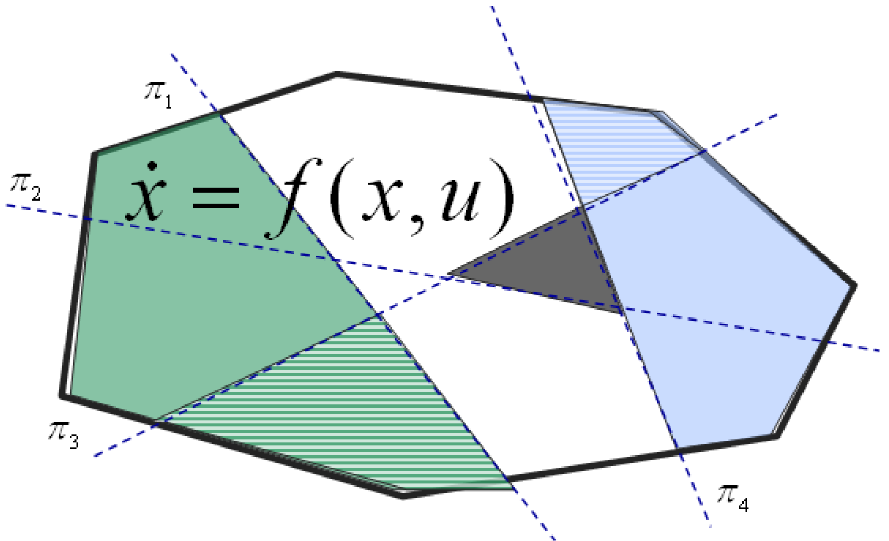
1. Conservative abstractions for simple dynamics





$$\Box\neg(\pi_2 \wedge \neg\pi_4 \wedge \neg\pi_3)) \wedge \Diamond(\pi_4 \wedge \Diamond(\pi_1 \wedge \Diamond (\Box\Diamond((\pi_3 \wedge \pi_4) \wedge \Diamond(\pi_1 \wedge \neg\pi_3)))))$$

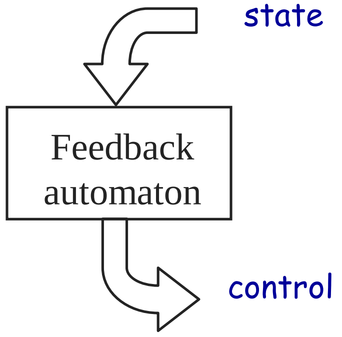
"(pi2 = TRUE and pi4 = FALSE and pi3 = FALSE) should never happen. Then pi4 = TRUE and then pi1 = TRUE should happen. After that, (pi3 = TRUE and pi4 = TRUE) and then (pi1 = TRUE and pi3 = FALSE) should occur infinitely often."

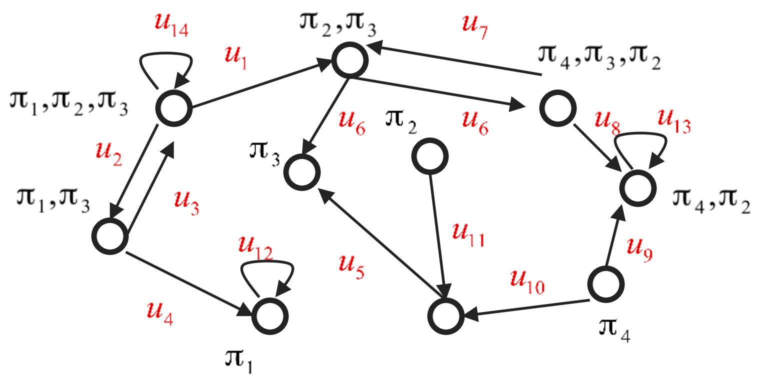


"Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order."

Conservative Control for Dynamical Systems

1. Conservative abstractions for simple dynamics

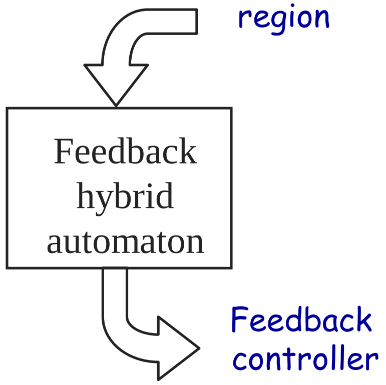


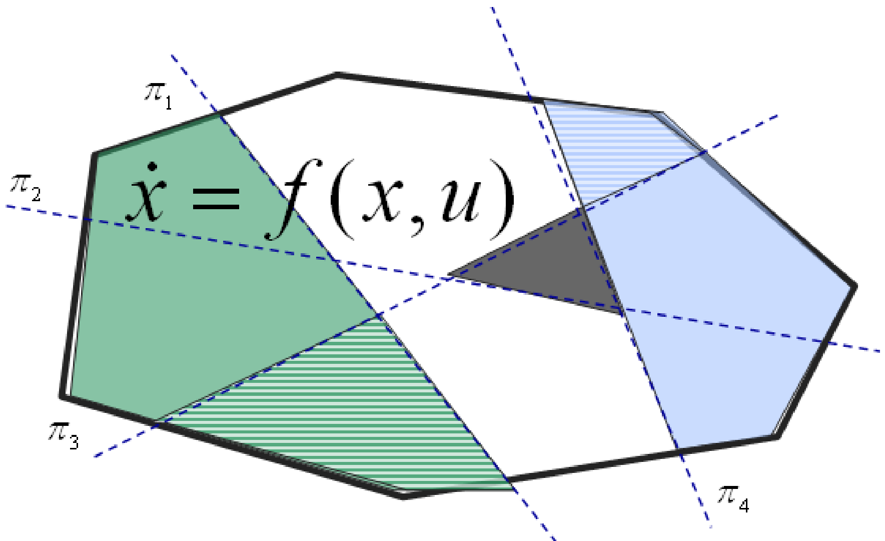


$$\Box\neg(\pi_2\wedge\neg\pi_4\wedge\neg\pi_3)\wedge \Diamond(\pi_4\wedge\Diamond(\pi_1\wedge\Diamond(\Box\Diamond((\pi_3\wedge\pi_4)\wedge\Diamond(\pi_1\wedge\neg\pi_3)))))$$

"(pi2 = TRUE and pi4 = FALSE and pi3 = FALSE) should never happen. Then pi4 = TRUE and then pi1 = TRUE should happen. After that, (pi3 = TRUE and pi4 = TRUE) and then (pi1 = TRUE and pi3 = FALSE) should occur infinitely often."

Refinement





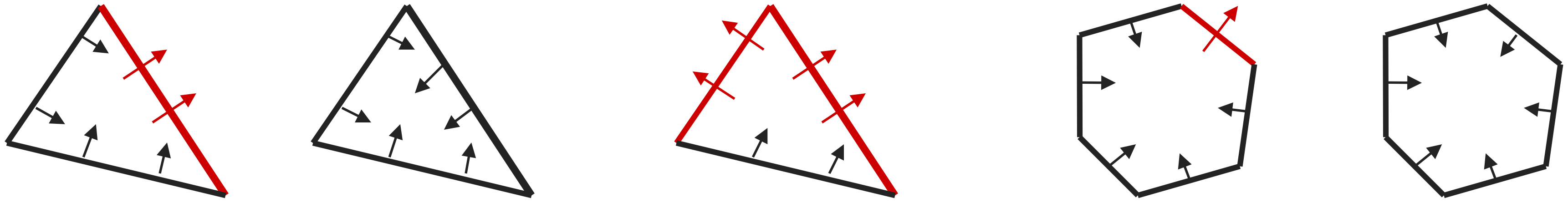
"Avoid the grey region for all times. Visit the blue region, then the green region, and then keep surveying the striped blue and green regions, in this order."

Conservative Control for Dynamical Systems

1. Conservative abstractions for simple dynamics

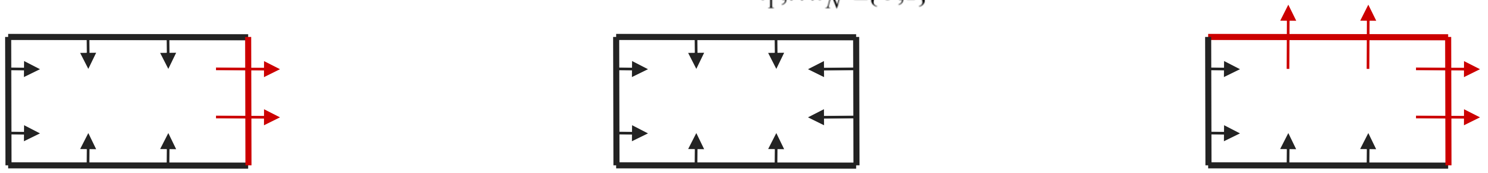
Library of controllers for polytopes

$\dot{x} = Ax + b + Bu \qquad x \in \Re^n \qquad u \in U \subset \Re^m \qquad U$ polyhedral



Control-to-facet Stay-inside Control-to-set-of-facets Control-to-face Stay-inside

$\dot{x} = g(x) + Bu \qquad x \in \Re^n \qquad g(x) = \sum_{i_1,\dots,i_N \in \{0,1\}} c_{i_1,\dots,i_N} x_1^{i_1} \dots x_n^{i_n} \qquad u \in U \subset \Re^m$



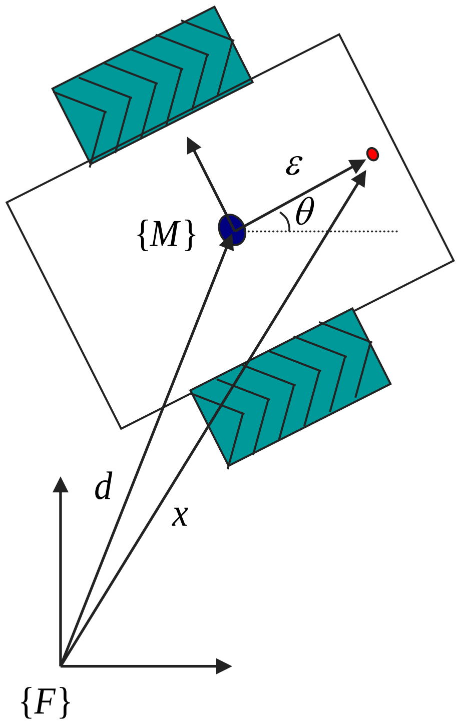
Control-to-facet Stay-inside Control-to-set-of-facets

- checking for existence of controllers amounts to checking the non-emptiness of polyhedral sets in U
- if controllers exist, they can be constructed everywhere in the polytopes by using simple formulas

L.C.G.J.M. Habets and J. van Schuppen, Automatica 2005
M. Kloetzer, L.C.G.J.M. Habets and C. Belta, CDC 2006
C. Belta and L.C.G.J.M. Habets, IEEE TAC, 2006

Conservative Control for Dynamical Systems

2. Mapping complex dynamics to simple dynamics



$$\dot{x} = u \qquad u \in U$$

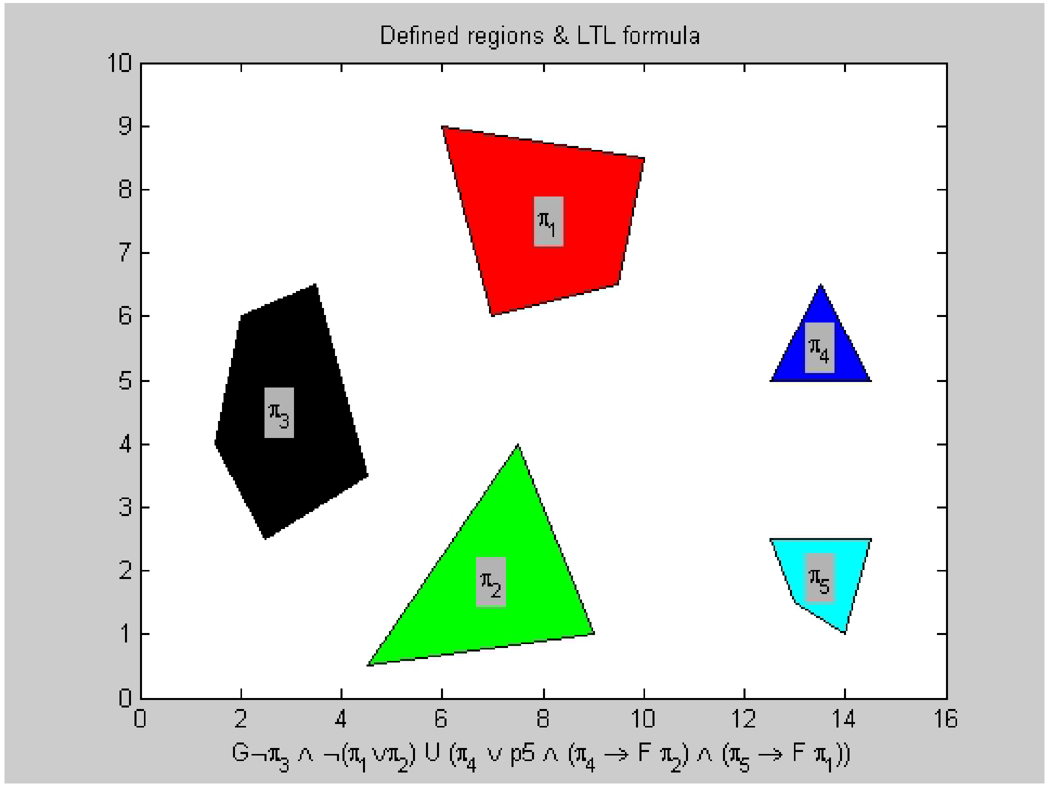
$$\dot{x} = REw \quad \Updownarrow \quad w = E^{-1}R^{T}u \qquad E = \begin{bmatrix} 1 & 0 \\ 0 & \varepsilon \end{bmatrix}$$

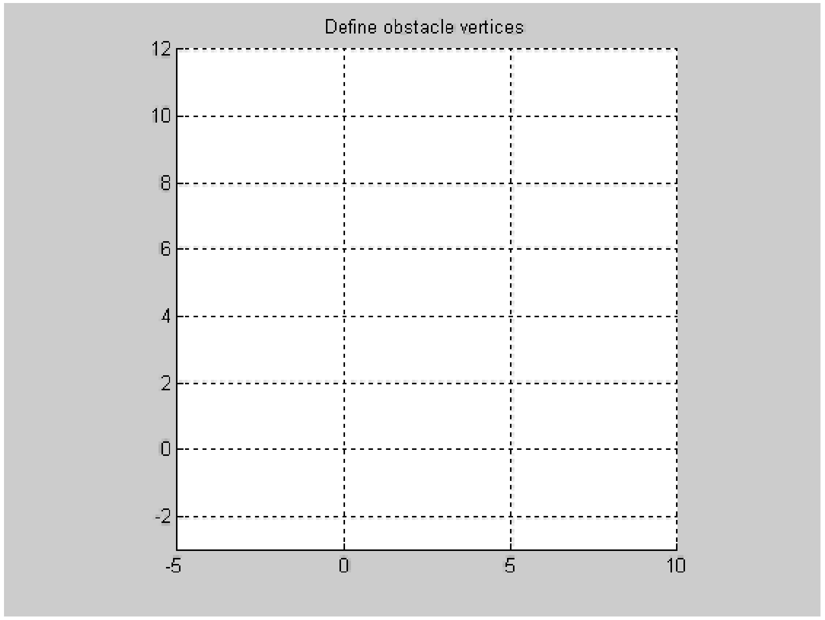
$$\begin{bmatrix} \dot{d}_x \\ \dot{d}_x \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} w_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w_2 \qquad w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in W$$

J. Desai, J.P. Ostrowski, and V. Kumar. ICRA, 1998.

Conservative Control for Dynamical Systems

"Always avoid black. Avoid red and green until blue or cyan are reached. If blue is reached then eventually visit green. If cyan is reached then eventually visit red."



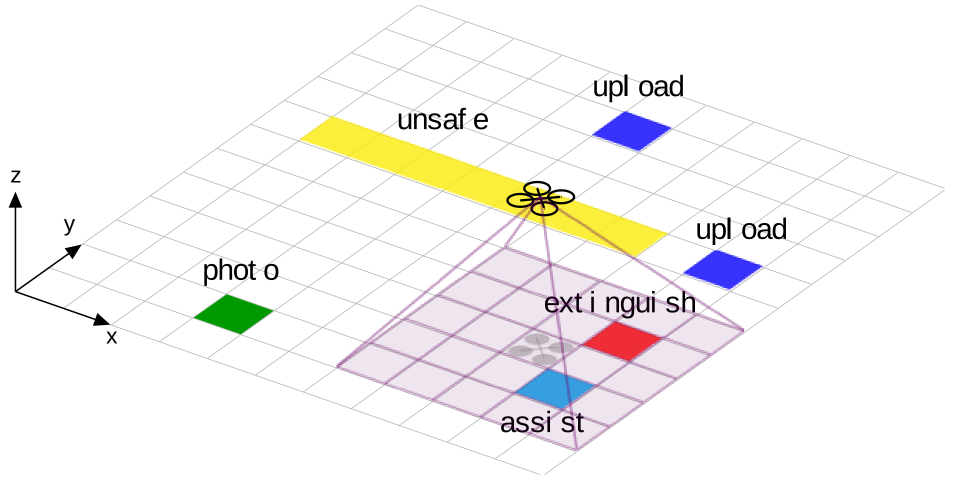


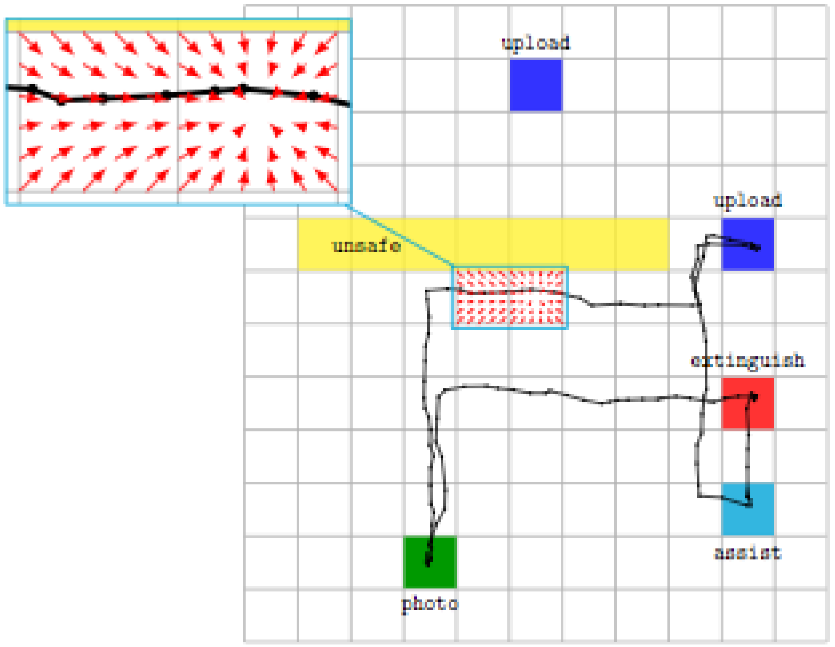
Conservative Control for Dynamical Systems

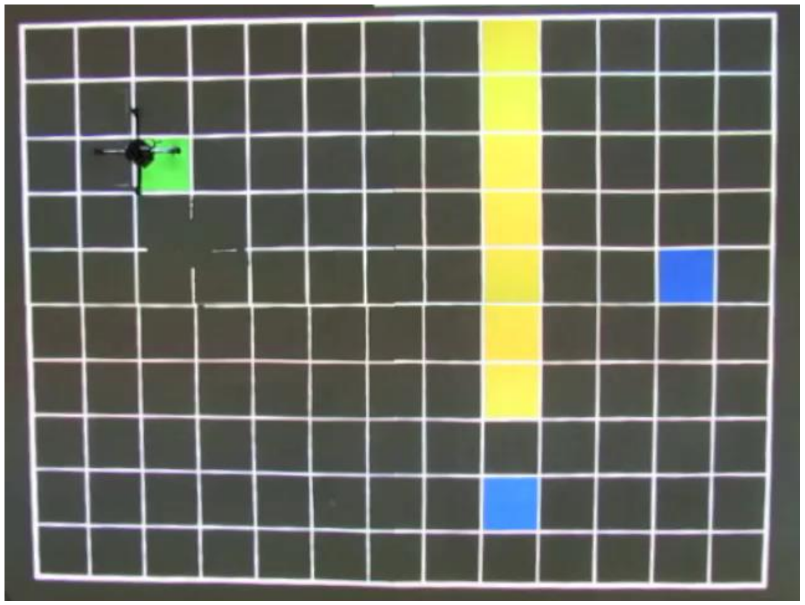
Quadrotor I/O Linearization Mellinger and Kumar, 2011.
Hoffmann, Waslander, and Tomlin, 2008.

Conservative Control for Dynamical Systems

Spec: "Keep taking photos and upload current photo before taking another photo. Unsafe regions should always be avoided. If fires are detected, then they should be extinguished. If survivors are detected, then they should be provided medical assistance. If both fires and survivors are detected locally, priority should be given to the survivors."





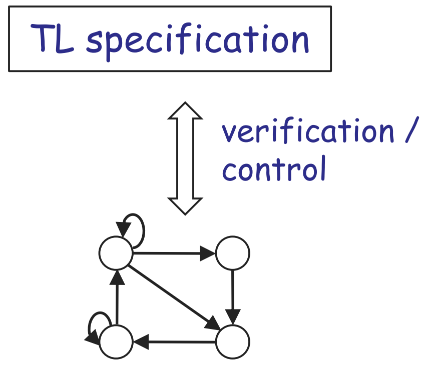


Ulusoy, Marrazzo, Belta, 2013

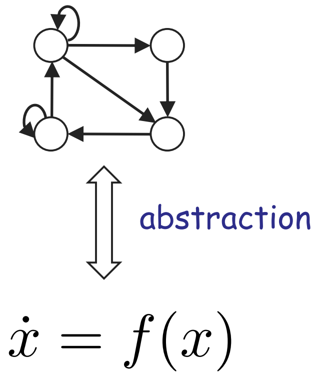
Outline

Verification and control for finite systems

Conservative control for dynamical systems

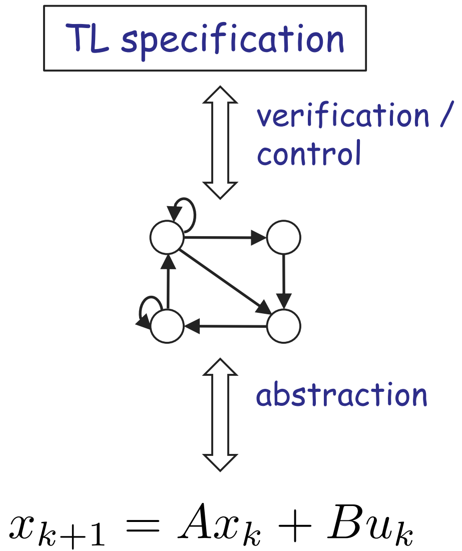


Finite quotients of continuous-space systems: main ideas

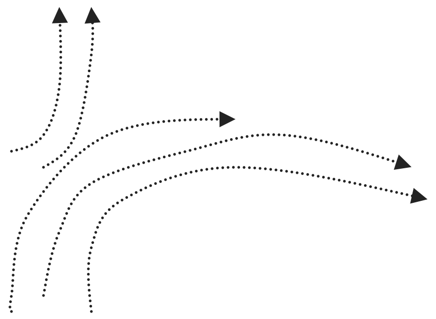


Verification for discrete-time linear systems

Control for discrete-time linear systems

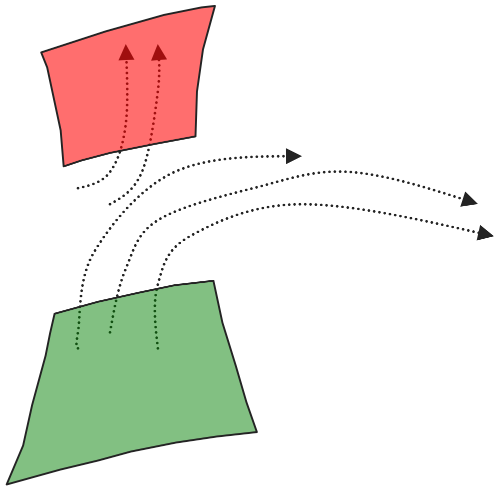


Finite quotients of continuous-space systems



$\dot{x} = f(x)$ (or $x(k+1) = f(x(k))$)

Finite quotients of continuous-space systems

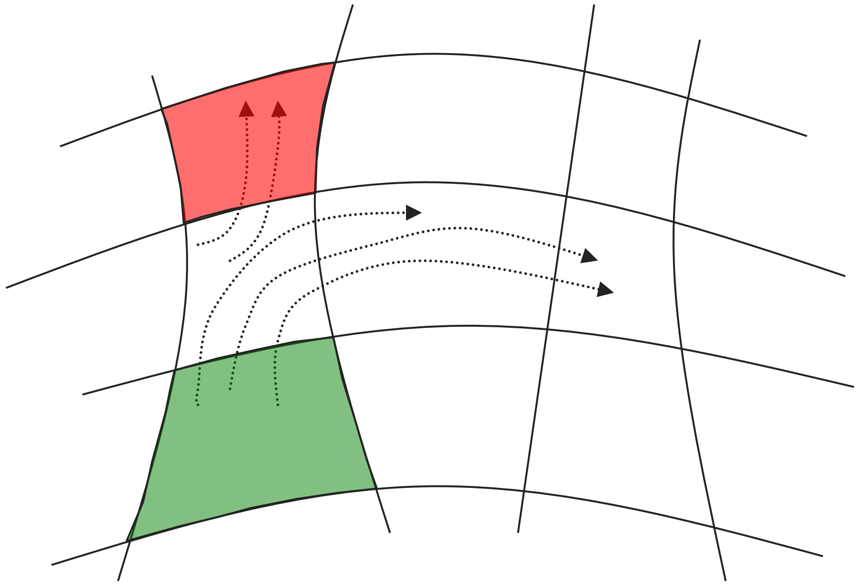


$$\dot{x} = f(x)$$

"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems

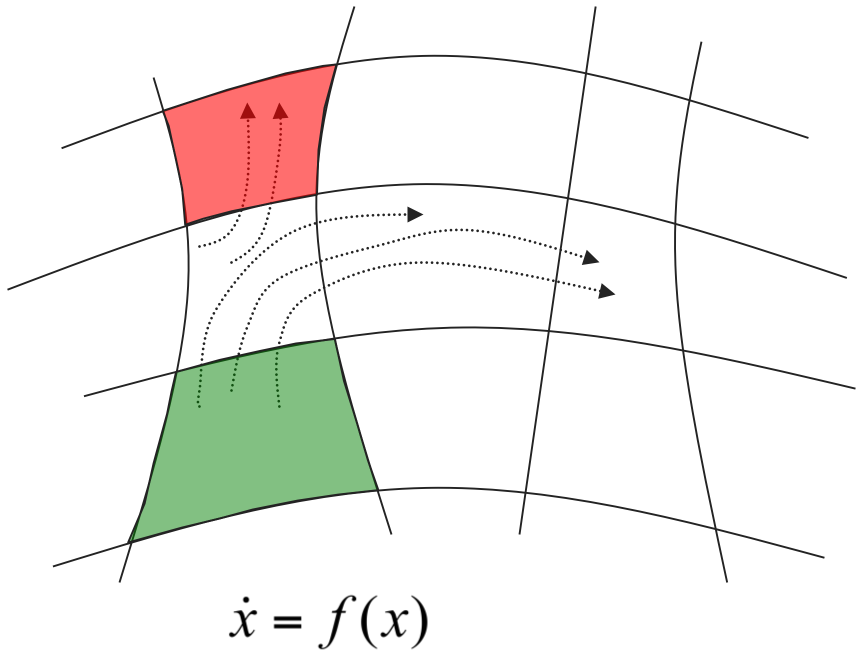


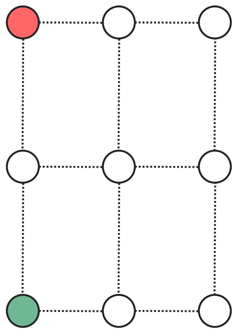
$\dot{x} = f(x)$

"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems

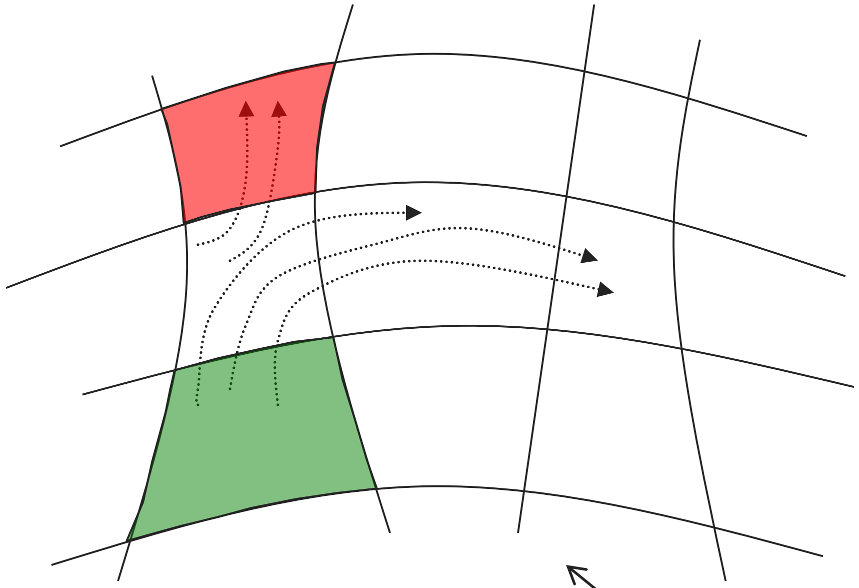




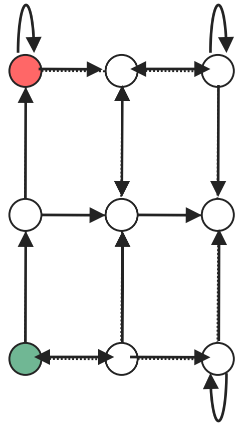
"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems



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



ideally

$\Leftrightarrow$

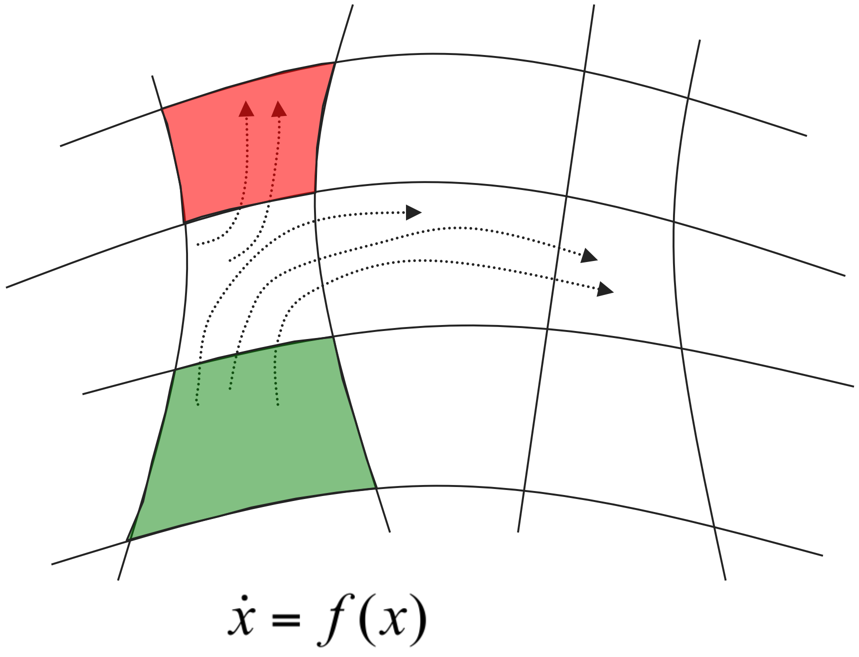
or, at least

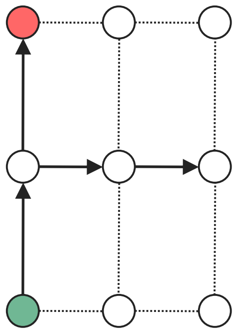
$\Leftarrow$

"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems



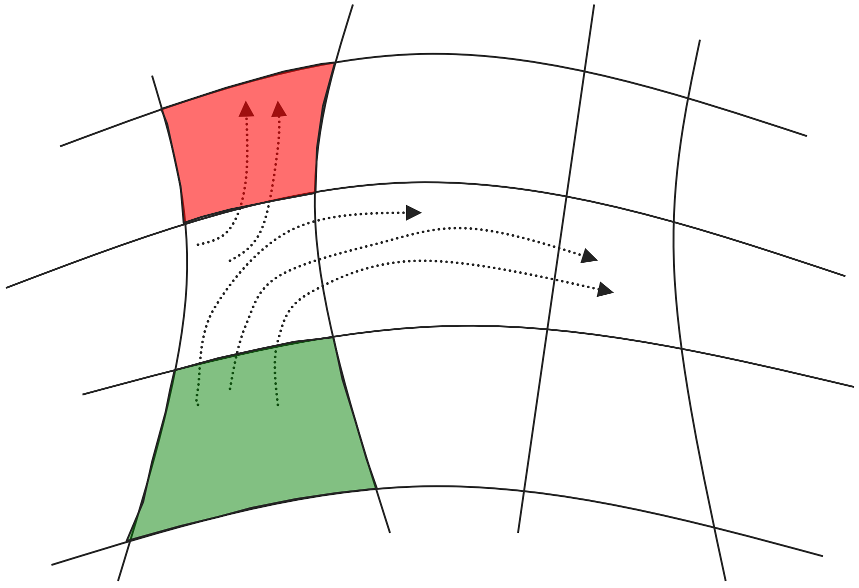


Assume we can decide whether there is a trajectory going from one region to an adjacent region

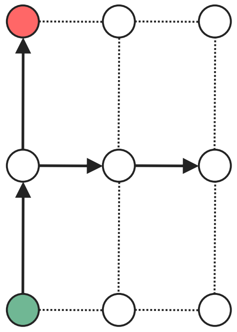
"There is no trajectory reaching from green to red" – True or False?

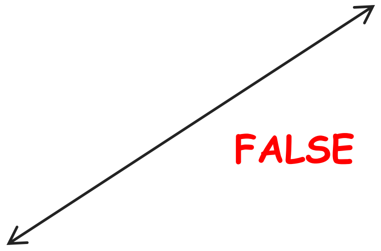
$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems



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



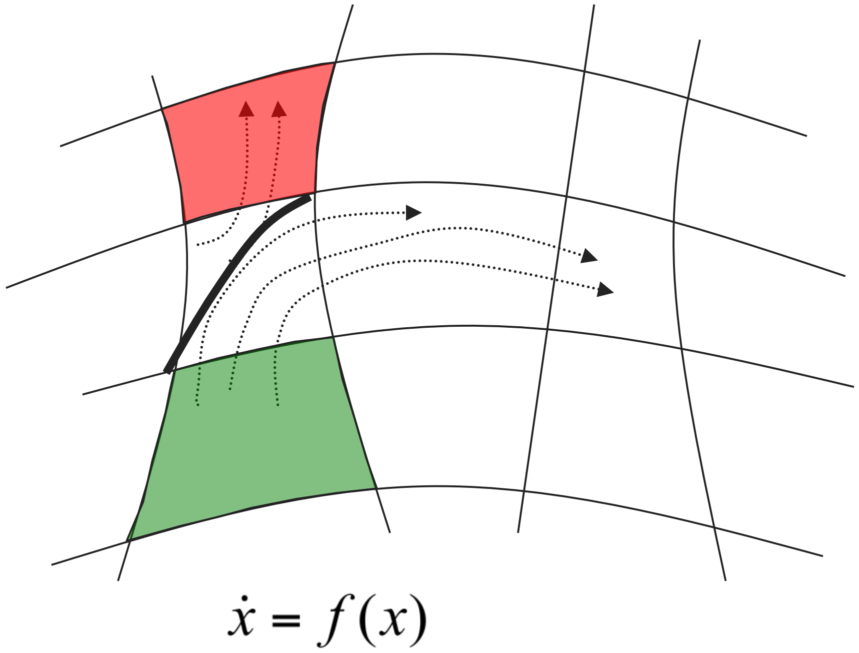


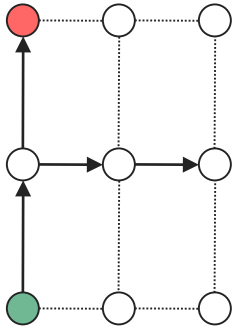
FALSE

"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems



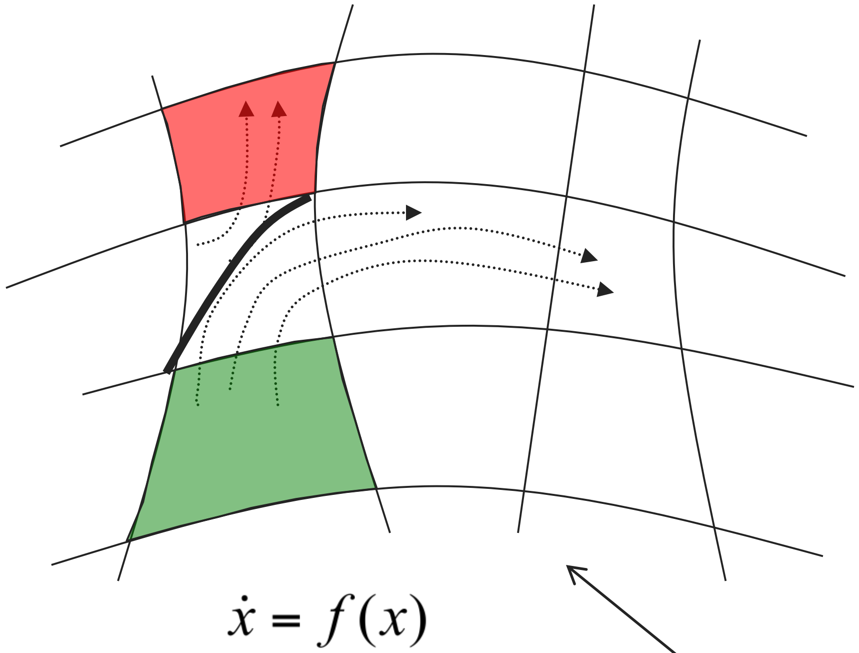


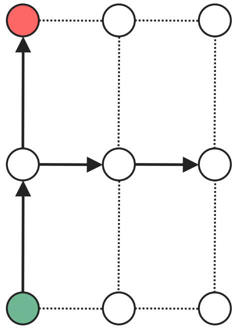
FALSE

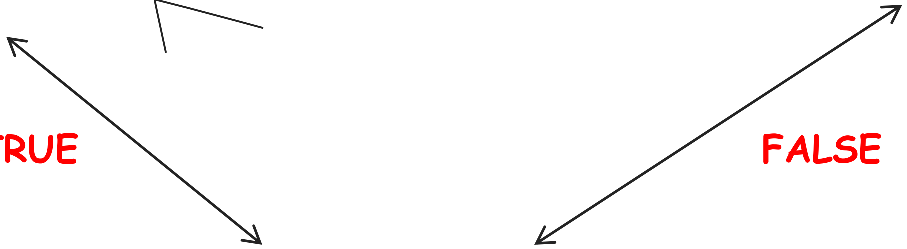
"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems





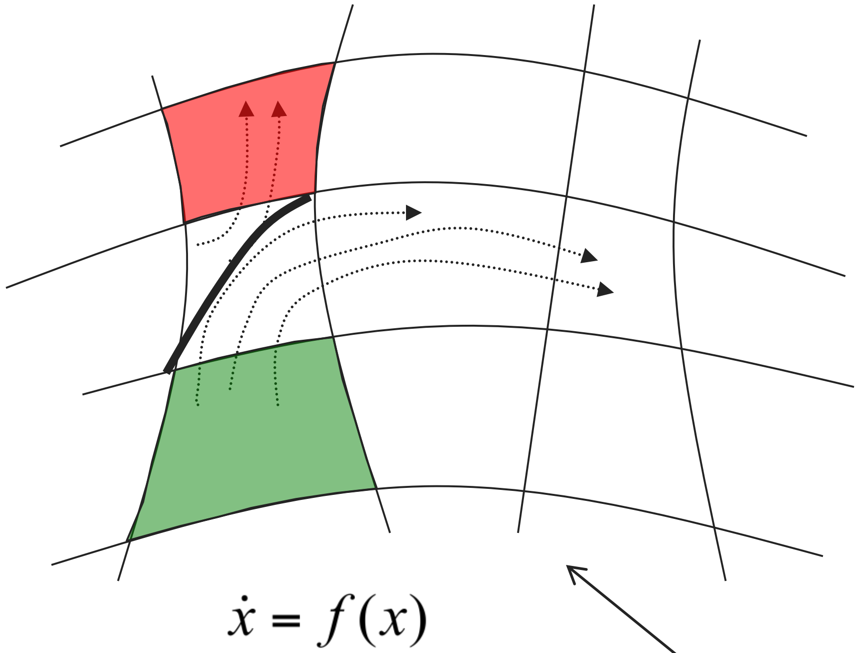


“There is no trajectory reaching from green to red” – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

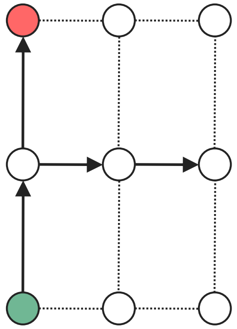
Finite quotients of continuous-space systems

Is there something wrong with the quotient?



simulation

$<$



TRUE

FALSE

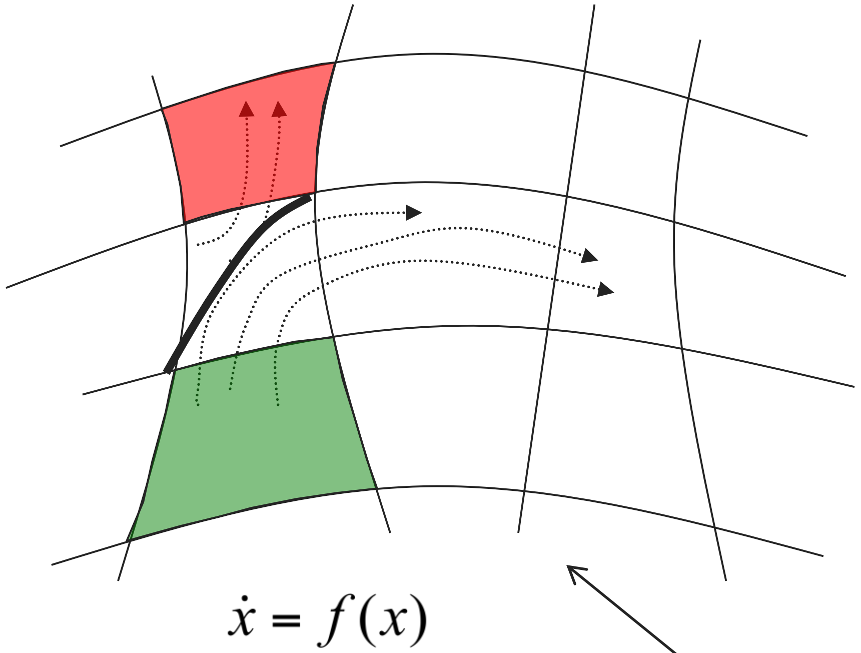
"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems

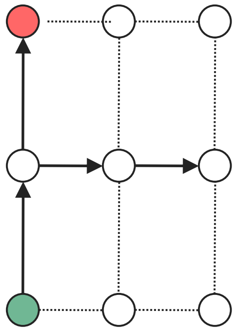
Is there something wrong with the quotient?

No, but it's too "rough" for proving this particular property.



simulation

$<$



TRUE

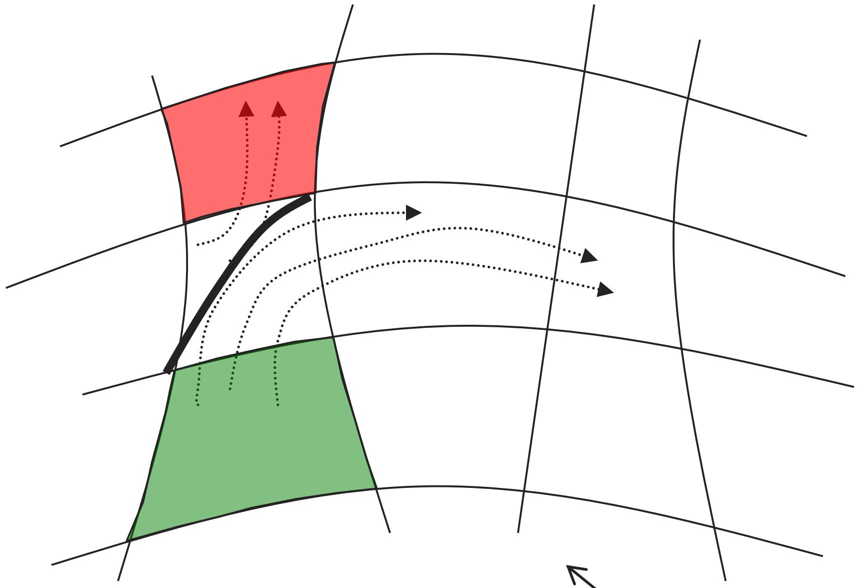
FALSE

"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

Finite quotients of continuous-space systems

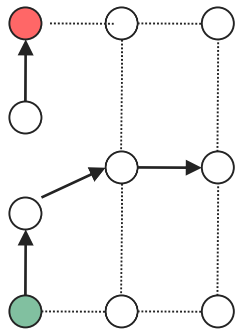
Refinement is necessary.



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

simulation

$<$



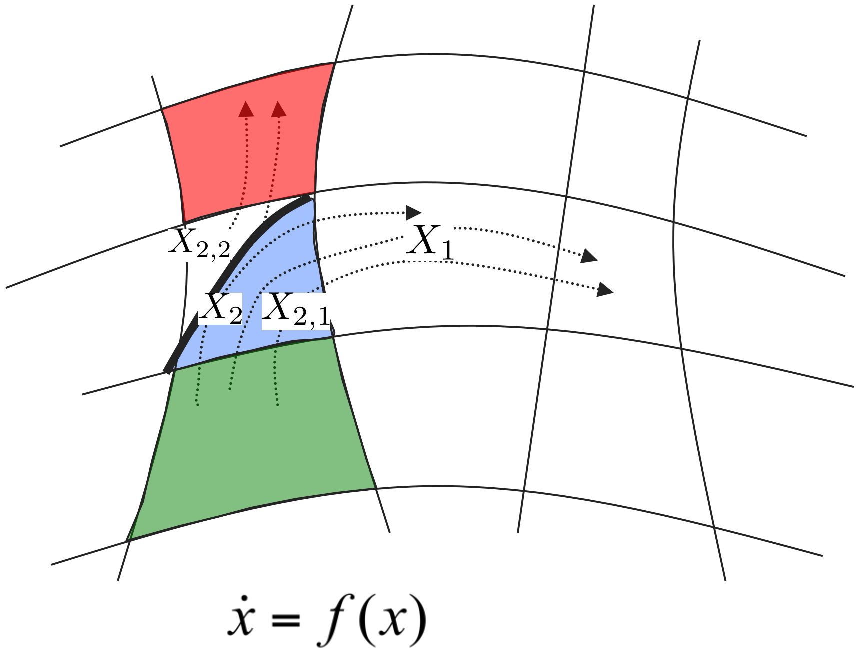
TRUE $\Leftarrow$ TRUE

"There is no trajectory reaching from green to red" – True or False?

$\neg(green \wedge \Diamond red)$ for all trajectories

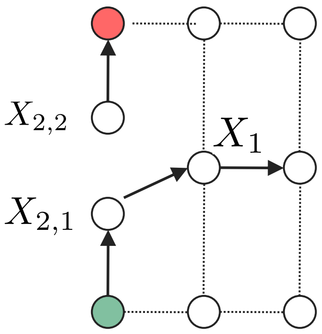
Finite quotients of continuous-space systems

Refinement is necessary.



simulation

$<$



$$Pre(X_1) = \{x \mid \exists t \geq 0 \, \exists x' \in X_1 \; s.t. \; x' = \phi(x,t)\}$$

$$X_{2,1} = Pre(X_1) \cap X_2$$

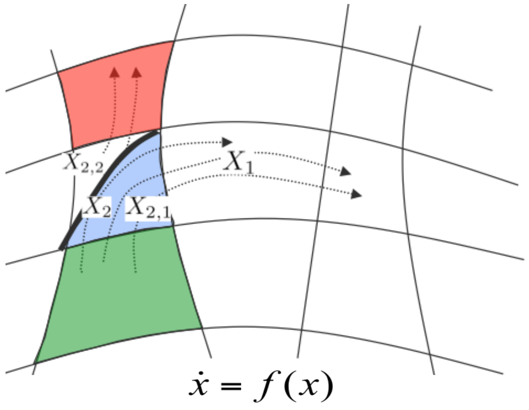
$$X_{2,2} = X_2 \setminus X_{2,1}$$

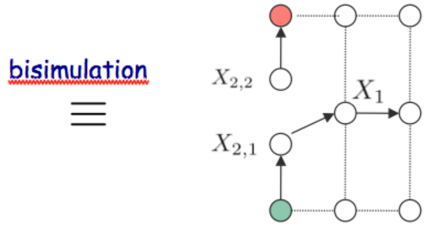
Finite quotients of continuous-space systems

Iterative refinement (bisimulation) algorithm

```
While there exist $X_i$ , $X_j$ such that $\emptyset \subset X_i \cap Pre(X_j) \subset X_i$
        $X_{i,1} = X_i \cap Pre(X_j)$
        $X_{i,2} = X_i \setminus X_{i,1}$
        remove $X_i$
        add $X_{i,1}$ , $X_{i,2}$
endwhile
```

A. Bouajjani, J.-C. Fernandez, and N. Halbwachs, 1991.





If the algorithm terminates, the finite quotient and the original system are called bisimilar, and the quotient can be used in lieu of the original system for verification from very general specs

Challenges:

Computability: set representation, computation of Pre, set intersection and difference, emptyness of sets

Termination: finite number of iterations

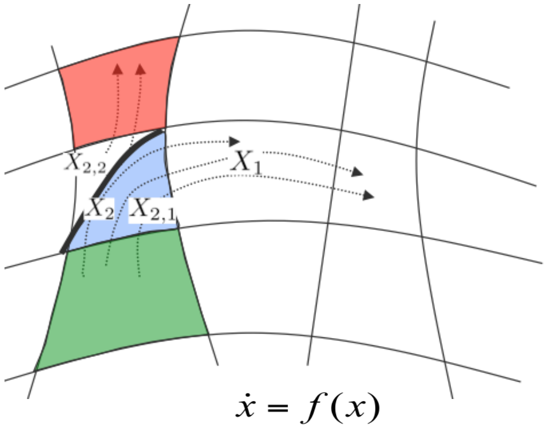
Decidability = Computability & Termination -> very restrictive classes of systems (e.g., timed automata, multi-rate automata, o-minimal systems)

R. Alur and D. L. Dill, 1994; R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho, 1993; G. Lafferriere, G. J. Pappas, and S. Sastry, 2000.

Finite quotients of continuous-space systems

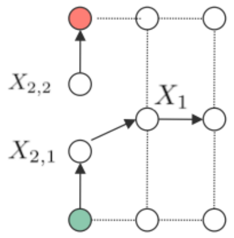
Give up termination

```
While there exist $X_i$ , $X_j$ such that $\emptyset \subset X_i \cap Pre(X_j) \subset X_i$
        $X_{i,1} = X_i \cap Pre(X_j)$
        $X_{i,2} = X_i \setminus X_{i,1}$
        remove $X_i$
        add $X_{i,1}$ , $X_{i,2}$
        construct the quotient
        model check the quotient
        if the spec is satisfied
                break
        endif
endwhile
```



simulation

<



A. Chutinan and B. H. Krogh, 2001.

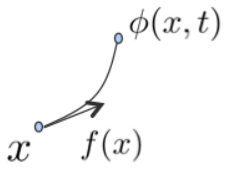
Verification only against universal properties, i.e., if all the trajectories of the quotient satisfy a spec, then all the trajectories of the original system satisfy the spec.

Computability:

- Still limited to very restrictive classes (should allow for quantifier elimination)

- Computation is very expensive

$$Pre(X_1) = \{x \mid \exists t \geq 0\, \exists x' \in X_1\, s.t.\, x' = \phi(x,t)\}$$

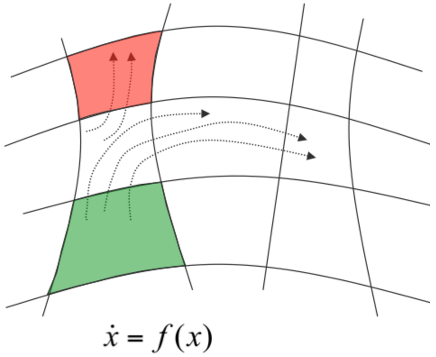


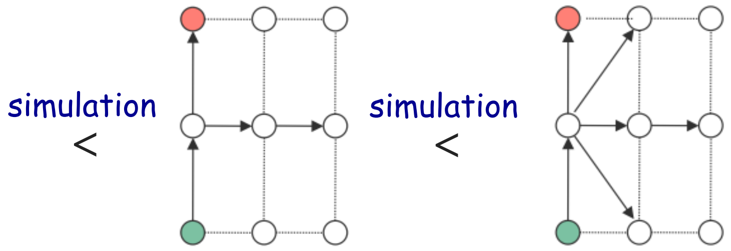
G. Lafferriere, G. J. Pappas, and S. Yovine, 2001.

Finite quotients of continuous-space systems

Give up computation of Pre

```
While TRUE
        construct (an over-approximation of) the quotient
        model check the quotient
        if the spec is satisfied
                break;
        endif
        refine (using some
                partitioning scheme)
endwhile
```





$$\overline{Post}(X) \supseteq Post(X) = \{x' \,|\, \exists x \in X\, \exists t > 0\; s.t.\; x' = \phi(x,t)\}$$

Continuous-time continuous-space polynomial dynamics and semi-algebraic regions (still requires quantifier elimination)

A. Tiwari and G. Khanna, 2002.

Continuous-time continuous-space affine and multi-affine dynamics and polytopic / rectangular / regions

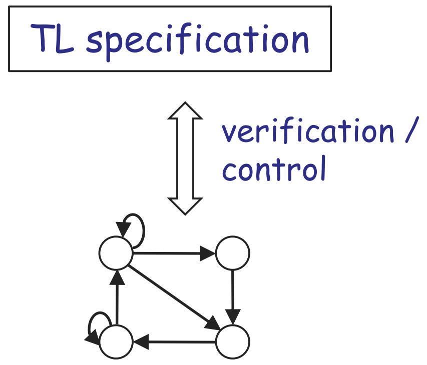
L.C.G.J.M. Habets and J.H. van Schuppen, 2004; C. Belta and L.C.G.J.M. Habets, 2006

M. Kloetzer and C. Belta, HSCC 2006, TIMC 2012

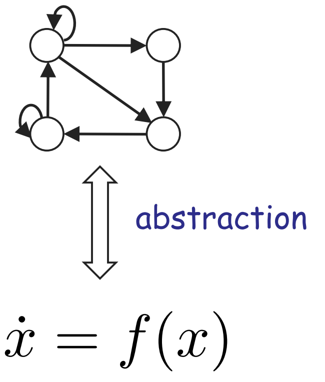
Outline

Verification and control for finite systems

Conservative control for dynamical systems

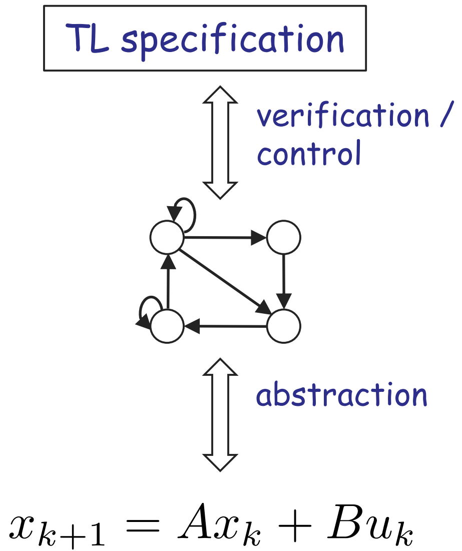


Finite quotients of continuous-space systems: main ideas



Verification for discrete-time linear systems

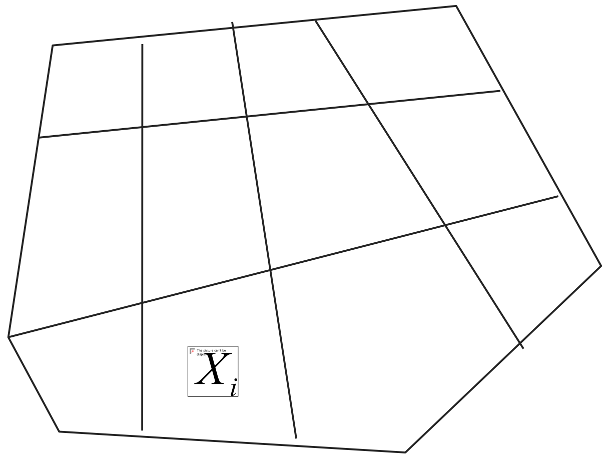
Control for discrete-time linear systems



Discrete-time PWA systems

$$x_{k+1} = A_i x_k + b_i,\ x_k \in X_i,\ i \in I$$

$X_i, i \in I$ polytopes



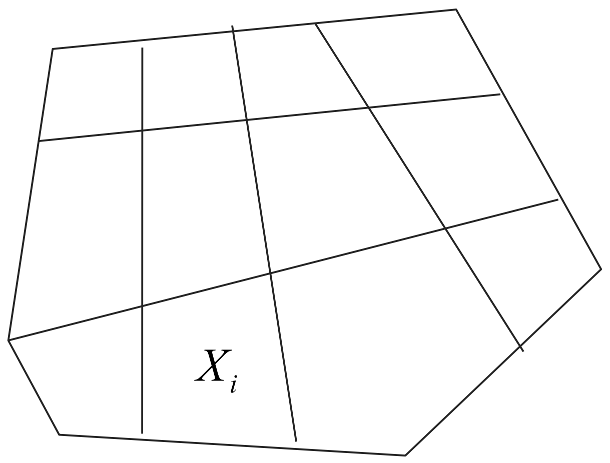
- Can approximate nonlinear systems with arbitrary accuracy [Lin and Unbehauen, 1992].
- Under mild assumptions, PWA systems are equivalent with several other classes of hybrid systems, including mixed logical dynamical (MLD), linear complementarity (LC), extended linear complementarity (ELC), and maxmin-plus-scaling (MMPS) systems [Heemels et al., 2001, Geyer et al., 2003]
- There exist tools for the identification of PWA systems from experimental data [Paoletti, Juloski, Ferrari-Trecate, Vidal, 2007]

Verification for discrete-time PWA systems

$x_{k+1} = A_i x_k + b_i,\ x_k \in X_i,\ i \in I$

$X_i, i \in I$ polytopes

$A_i, i \in I$ invertible

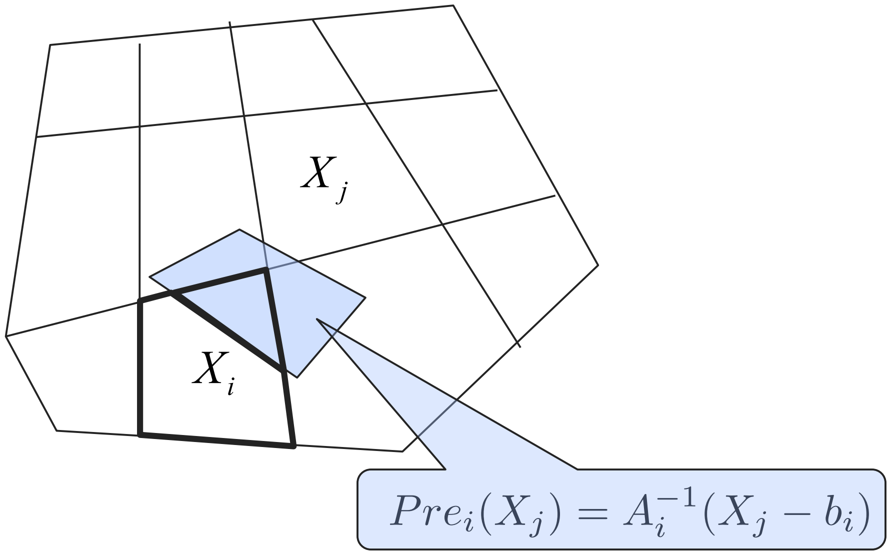


Verification for discrete-time PWA systems

$x_{k+1} = A_i x_k + b_i,\ x_k \in X_i,\ i \in I$

$X_i,\ i \in I$ polytopes

$A_i,\ i \in I$ invertible



```
While there exist $X_i$ , $X_j$ such that $\emptyset \subset X_i \cap Pre(X_j) \subset X_i$
        $X_{i,1} = X_i \cap Pre(X_j)$
        $X_{i,2} = X_i \setminus X_{i,1}$
        remove $X_i$
        add $X_{i,1}$ , $X_{i,2}$
        construct the quotient
        model check the quotient
        if the spec is satisfied
                break
        endif
endwhile
```

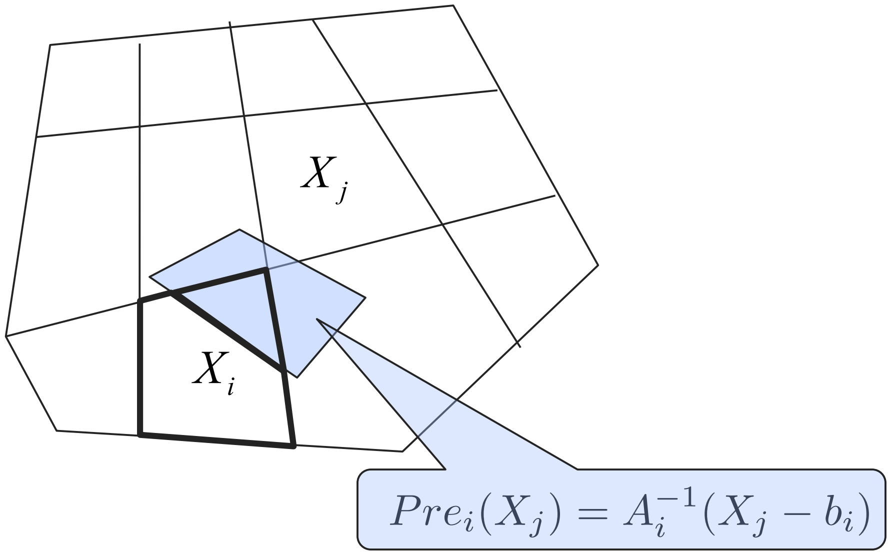
Everything is computable!

Verification for discrete-time PWA systems

$x_{k+1} = A_i x_k + b_i,\ x_k$ Î X_i, i Î I

X_i, i Î I polytopes

A_i, i Î I invertible



```
While there exist $X_i$ , $X_j$ such that $\emptyset \subset X_i \cap Pre(X_j) \subset X_i$
        $X_{i,1} = X_i \cap Pre(X_j)$
        $X_{i,2} = X_i \setminus X_{i,1}$
        remove $X_i$
        add $X_{i,1}$ , $X_{i,2}$
        construct the quotient
        model check the quotient
        if the spec is satisfied
                break
        endif
endwhile
```

Everything is computable!

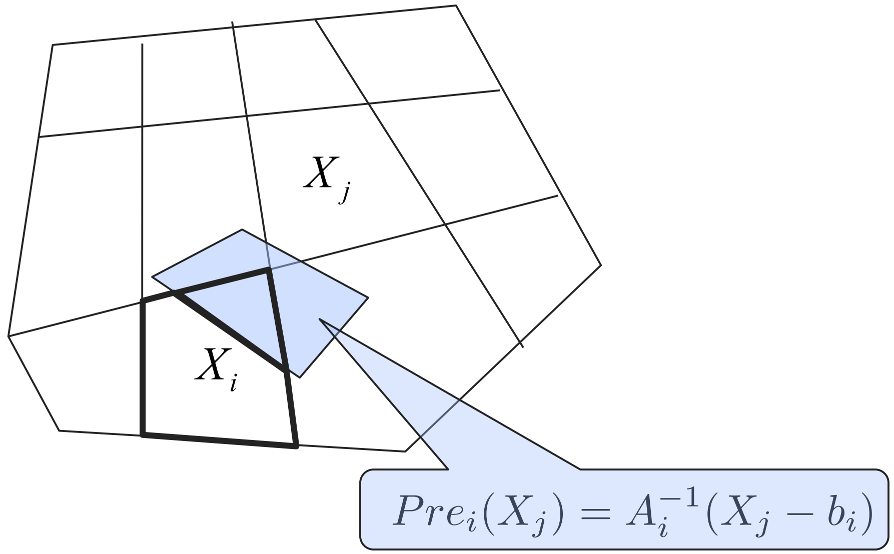
Problem Formulation: Find the largest subset of $\bigcup_{i \in I} X_i$ such that all the trajectories originating there satisfy an LTL formula f over I.

Verification for discrete-time PWA systems

$x_{k+1} = A_i x_k + b_i,\ x_k \in X_i,\ i \in I$

$X_i,\ i \in I$ polytopes

$A_i,\ i \in I$ invertible



```
While there exist $X_i$ , $X_j$ such that $\emptyset \subset X_i \cap Pre(X_j) \subset X_i$
        $X_{i,1} = X_i \cap Pre(X_j)$
        $X_{i,2} = X_i \setminus X_{i,1}$
        remove $X_i$
        add $X_{i,1}$ , $X_{i,2}$
        construct the quotient
        model check the quotient
        if the spec is satisfied
                break
        endif
endwhile
```

Everything is computable!

Can be optimized by checking with both f and $\neg f$ and partitioning only if necessary (no need to refine regions where the formula or its negation is satisfied at the corresponding state of the quotient).

Problem Formulation: Find the largest subset of $\bigcup_{i \in I} X_i$ such that all the trajectories originating there satisfy an LTL formula f over I.

B. Yordanov and C. Belta, IEEE TAC 2010

Verification for discrete-time PWA systems

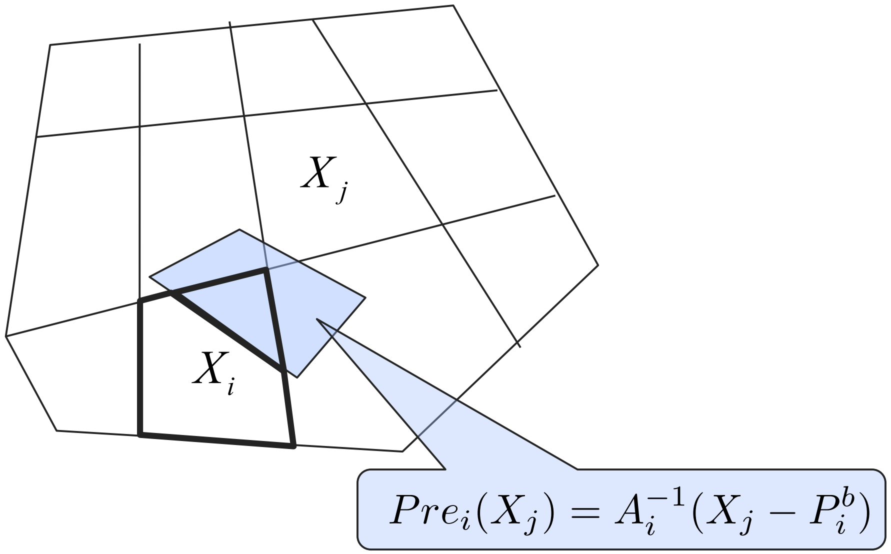
$x_{k+1} = A_i x_k + b_i,\ x_k \in X_i,\ i \in I$

$X_i, i \in I$ polytopes $\qquad P_i^b, i \in I$ polytopes

$A_i, i \in I$ invertible

What if $b_i \in P_i^b, i \in I$?

Everything still works with extra computational overhead.



```
While there exist $X_i$ , $X_j$ such that $\emptyset \subset X_i \cap Pre(X_j) \subset X_i$
        $X_{i,1} = X_i \cap Pre(X_j)$
        $X_{i,2} = X_i \setminus X_{i,1}$
        remove $X_i$
        add $X_{i,1}$ , $X_{i,2}$
        construct the quotient
        model check the quotient
        if the spec is satisfied
                break
        endif
endwhile
```

Everything is computable!

Can be optimized by checking with both ϕ and $\neg\phi$ and partitioning only if necessary (no need to refine regions where the formula or its negation is satisfied at the corresponding state of the quotient).

Problem Formulation: Find the largest subset of $\bigcup_{i \in I} X_i$ such that all the trajectories originating there satisfy an LTL formula ϕ over I.

B. Yordanov and C. Belta, IEEE TAC 2010

Verification for discrete-time PWA systems

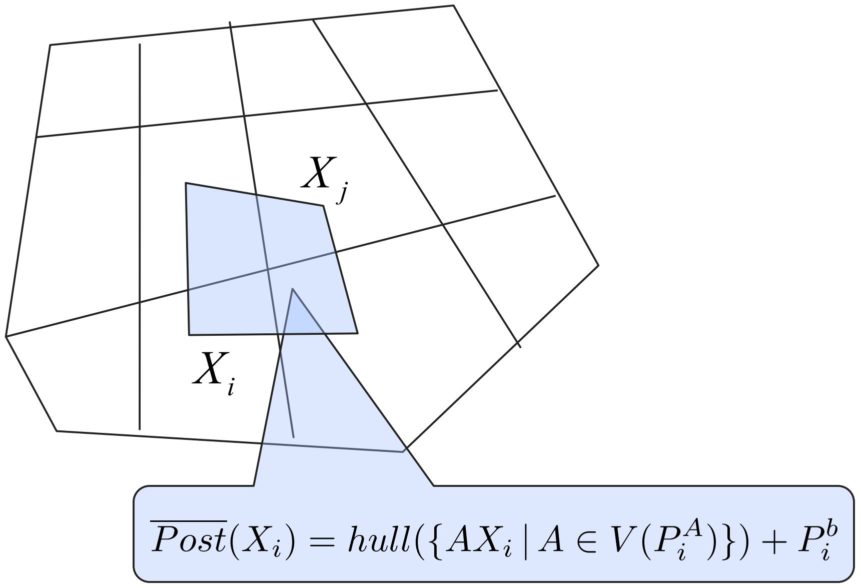
$x_{k+1} = A_i x_k + b_i,\ x_k \in X_i,\ i \in I$

$X_i, i \in I$ polytopes $\qquad P_i^b, i \in I$ polytopes

$A_i, i \in I$ invertible $\qquad P_i^A, i \in I$ polytopes

What if $b_i \in P_i^b, i \in I$ and $A_i \in P_i^A, i \in I$?

Pre is not computable anymore. A polyhedral over-approximation of Post is computable.



$$\overline{Post}(X_i) = hull(\{AX_i \mid A \in V(P_i^A)\}) + P_i^b$$

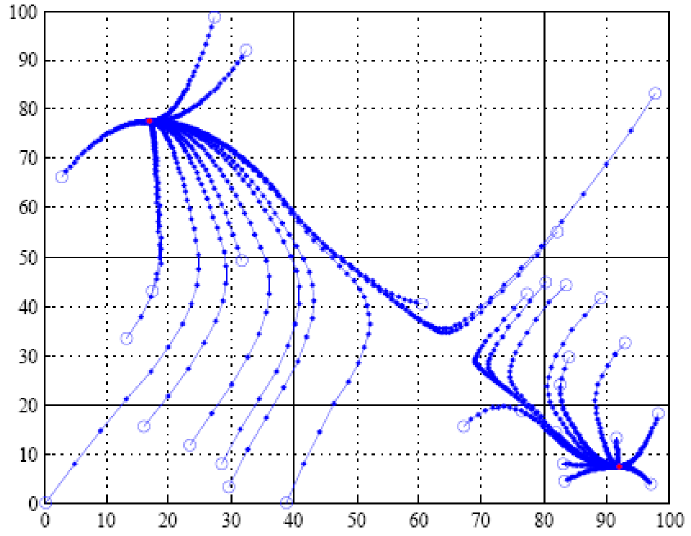
```
While TRUE
        construct (an over-approximation of) the quotient
        model check the quotient
        if the spec is satisfied
                break;
        endif
        refine (using arbitrary partitioning schemes)
endwhile
```

Problem Formulation: Find the largest subset of $\bigcup_{i \in I} X_i$ such that all the trajectories originating there satisfy an LTL formula f over I.

B. Yordanov and C. Belta, IEEE TAC 2010

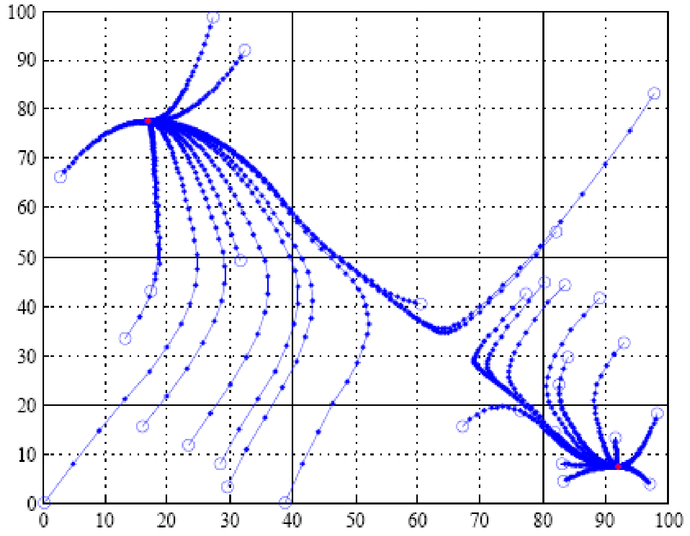
Verification for discrete-time PWA systems

Example: toggle switch

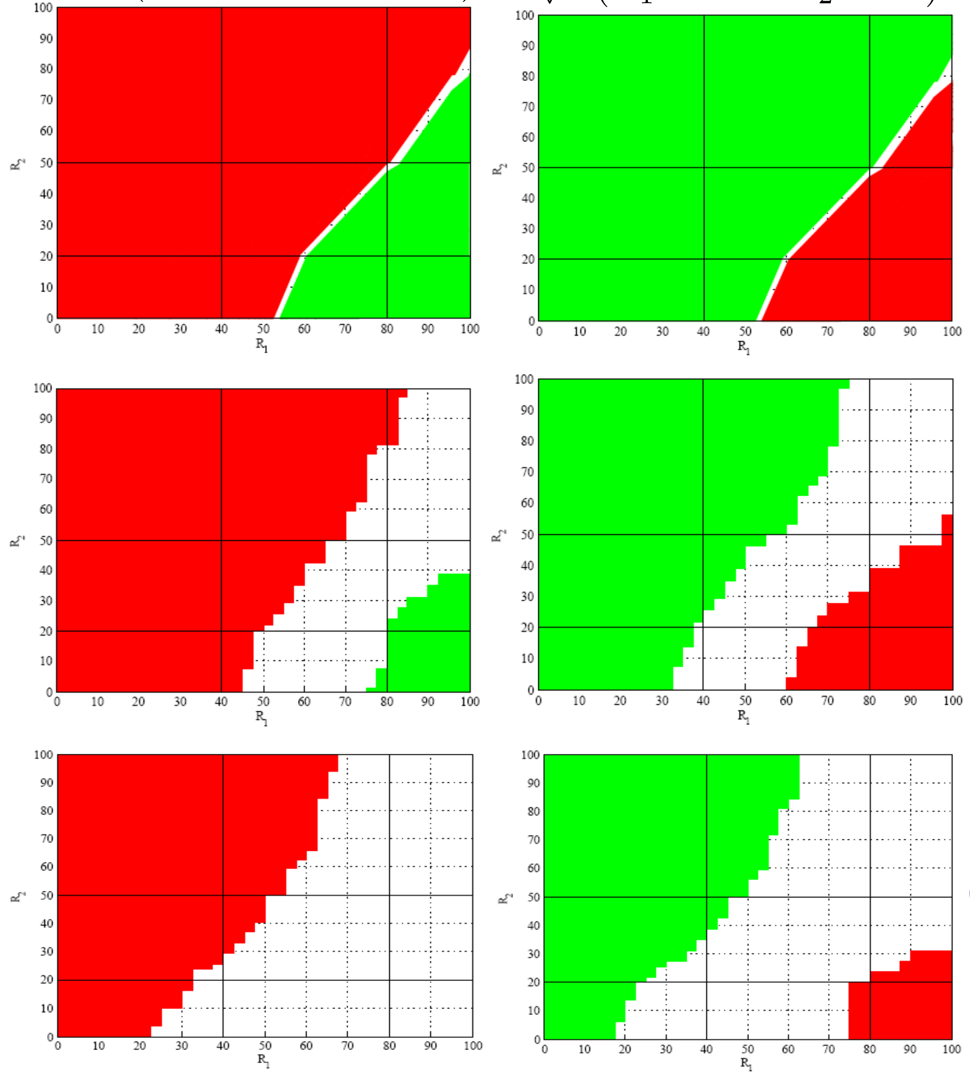


Verification for discrete-time PWA systems

Example: toggle switch



$\Diamond\Box(R_1 > 80 \wedge R_2 < 20)$ $\Diamond\Box(R_1 < 40 \wedge R_2 > 50)$



Fixed parameters

1% param uncertainty

10% param uncertainty

Initial satisfying states

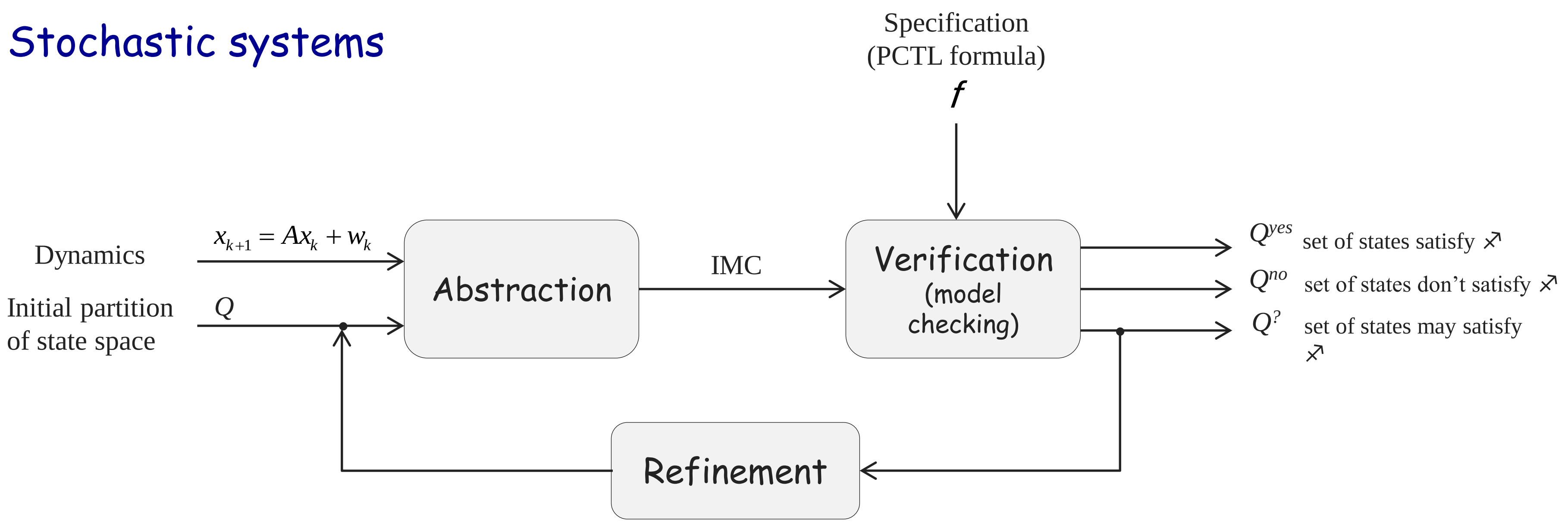
Initial violating states

B. Yordanov and C. Belta, IEEE TAC 2010

Matlab tool: “FaPAS” (hyness.bu.edu/software)

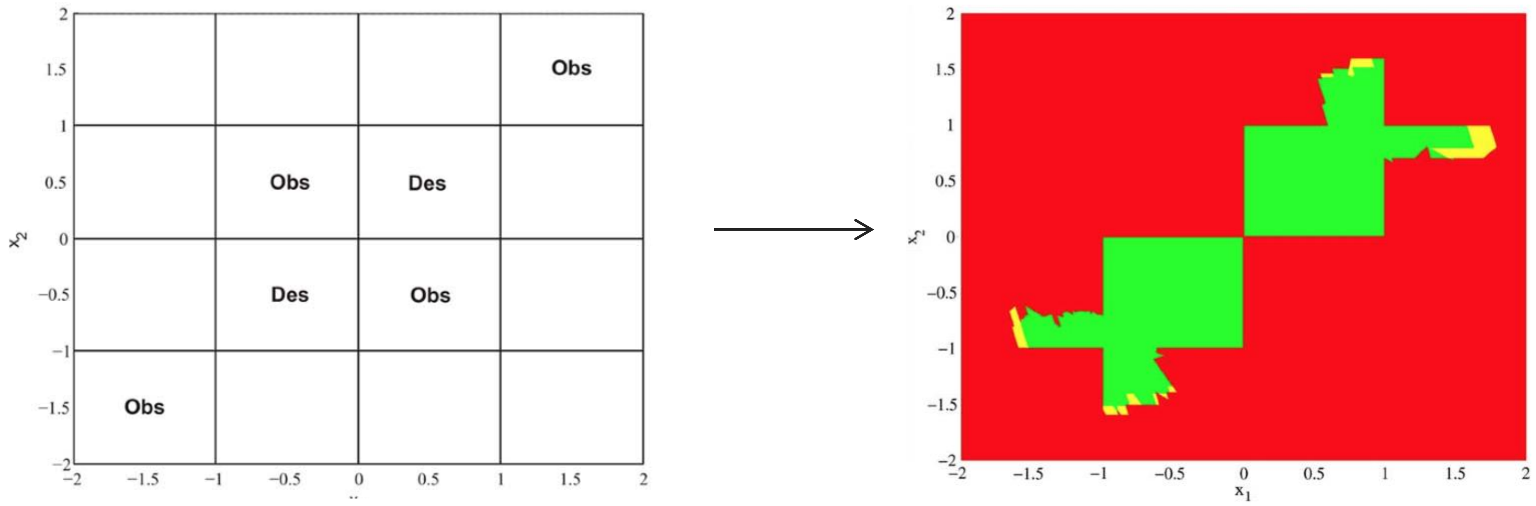
Verification for discrete-time linear systems

Stochastic systems


Specification
(PCTL formula)
f
Dynamics
$x_{k+1} = Ax_k + w_k$
Initial partition of state space
Q
Abstraction
IMC
Verification (model checking)
Q^{yes} set of states satisfy ↗
Q^{no} set of states don't satisfy ↗
$Q^{?}$ set of states may satisfy ↗
Refinement

$$\mathcal{P}_{\geq 0.90}\left[(\neg\mathbf{Obs} \wedge \mathcal{P}_{<0.05}[X\,\mathbf{Obs}])\,\mathcal{U}\,\mathbf{Des}\right]$$

"With probability 0.90 or greater reach *Destination* through the regions that are not *Obstacles* and that have a probability of less than 0.05 to converge to a region with an *Obstacle*."



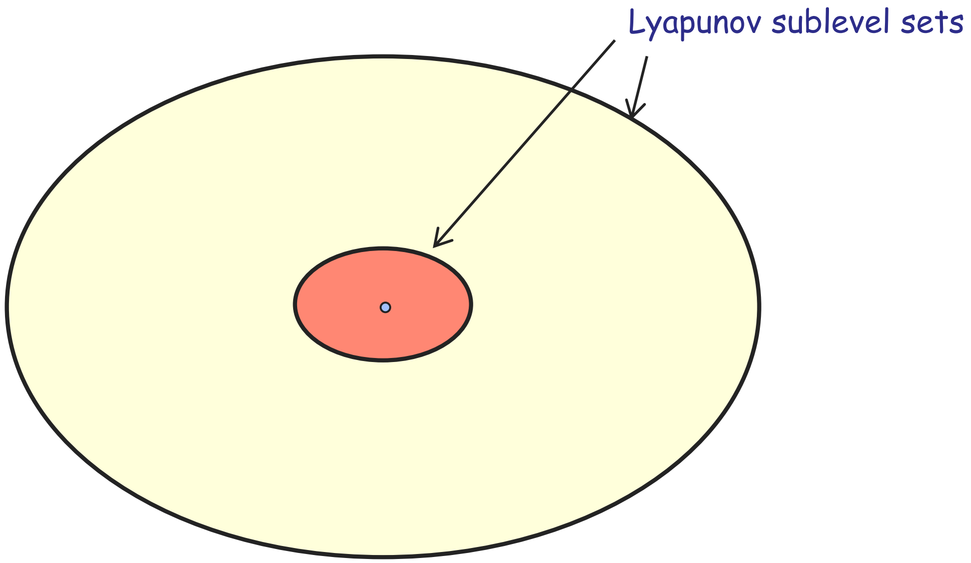
Initial states that definitely, possibly, and never satisfy are shown in green, yellow, and red, respectively.

Lahijanian, Andersson, Belta, IEEE TAC 2015

Verification for discrete-time systems

Using Lyapunov functions to construct finite bisimulations

$$x_{k+1} = \Phi(x_k)$$

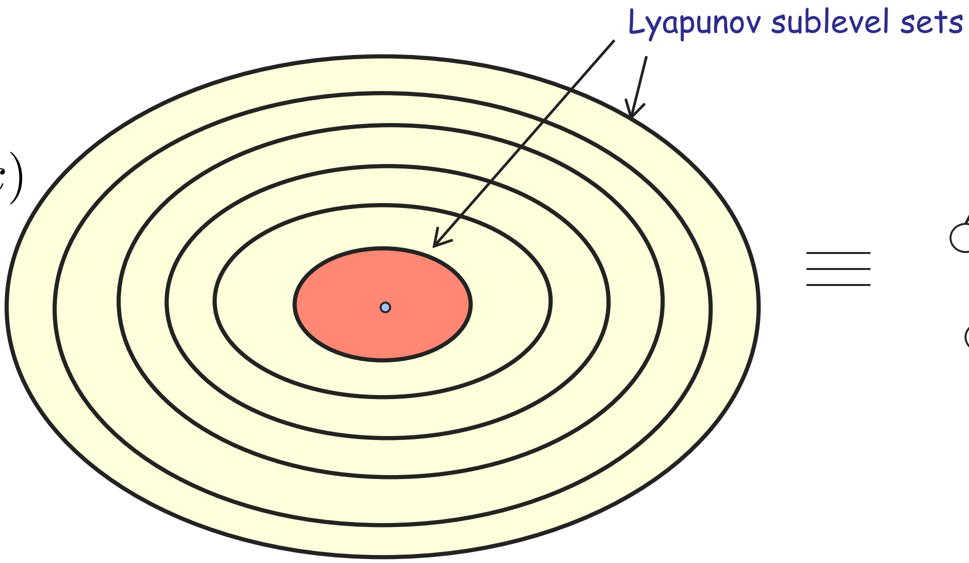


Verification for discrete-time systems

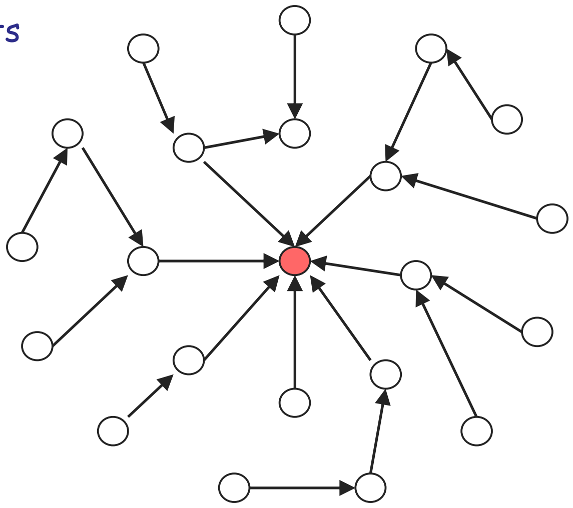
Using Lyapunov functions to construct finite bisimulations

$$x_{k+1} = \Phi(x_k)$$

$$V(\Phi(x)) \leq \rho V(x)$$



$\equiv$



Algorithm: Slice the space in between two sublevel sets into N slices (N determined by the contraction rate); Starting from the inner-most slice, compute the pre-image of the slice and intersect it with all the other slices.

Theorem: At the ith iteration, the partition of the inner region bounded by the ith slice is a bisimulation. As a result, a bisimulation for the whole region is obtained in N steps

Applicability:
- we can only reason about the behavior of the system in between two sublevel sets (we should not mind that all trajectories of the system eventually disappear in the region closest to the origin)
- need to be able to compute the pre-image of a slice through the dynamics of the system and the intersections with other slices

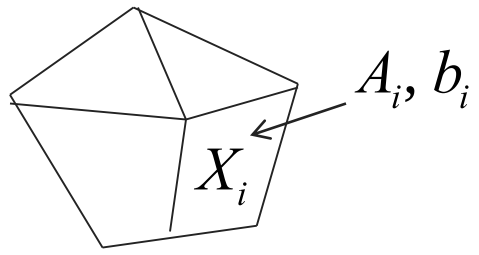
E. Aydin Gol, X.C. Ding,, M. Lazar, C. Belta ADHS 2012, CDC 2012, IEEE TAC 2014

Verification for discrete-time systems

Using Lyapunov functions to construct finite bisimulations

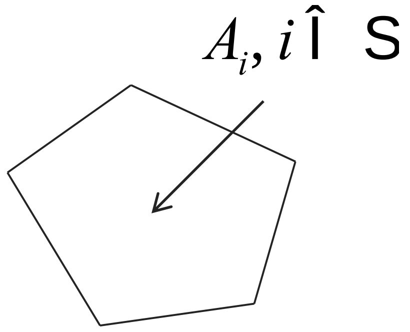
Computability

Discrete-time PWA systems



$$x_{k+1} = A_i x_k + b_i,\ x_k \in X_i,\ i \in I$$

Discrete-time switched linear systems



$$x_{k+1} = A_{\sigma(k)} x_k,\ \sigma(k) \in \Sigma$$

Lyapunov functions with polytopic sublevel sets can be constructed

$$V(x) = \|Lx\|_\infty$$

Blanchini 1994, Lazar 2010

Verification for discrete-time linear systems

Using Lyapunov functions to construct finite bisimulations

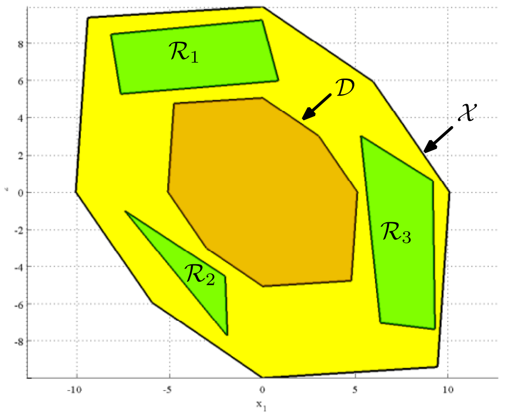
Example:

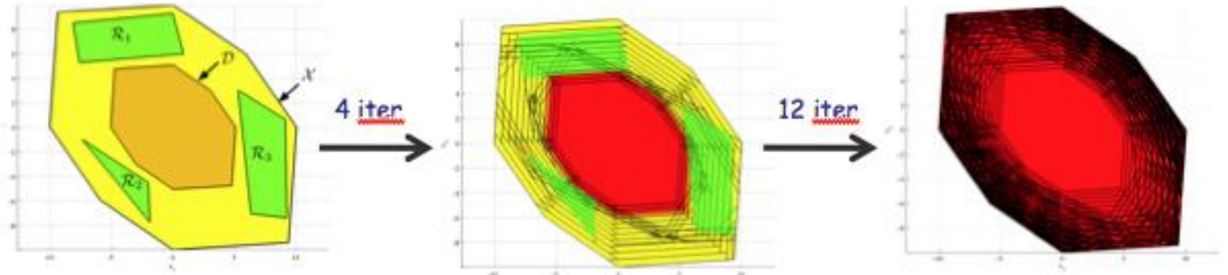
$$\Sigma = \{1,2\} \qquad A_1 = \begin{pmatrix} -0.65 & 0.32 \\ -0.42 & -0.92 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0.65 & 0.32 \\ -0.42 & -0.92 \end{pmatrix}$$

$$x_{k+1} = A_{\sigma(k)} x_k,\ \sigma(k) \in \Sigma$$

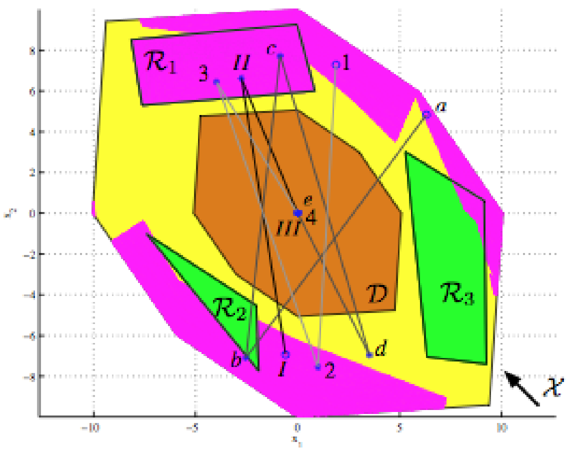
"A system trajectory never visits $\mathcal{R}_2$ and eventually visits $\mathcal{R}_1$. Moreover, if it visits $\mathcal{R}_3$ then it must not visit $\mathcal{R}_1$ at the next time step" can be translated to a scLTL formula:

$$\phi := (\neg \mathcal{R}_2 \,\mathsf{U}\, \Pi_{\mathcal{D}}) \wedge \mathsf{F}\, \mathcal{R}_1 \wedge ((\mathcal{R}_3 \Rightarrow \mathsf{X} \neg \mathcal{R}_1) \,\mathsf{U}\, \Pi_{\mathcal{D}})$$

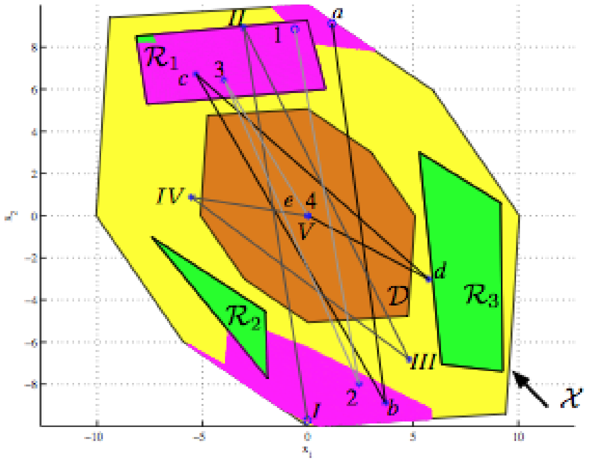




Purple: Sets of initial states for which there exists a switching strategy such that all trajectories satisfy the spec



Purple: Sets of initial states for which all trajectories satisfy the spec under all possible switches

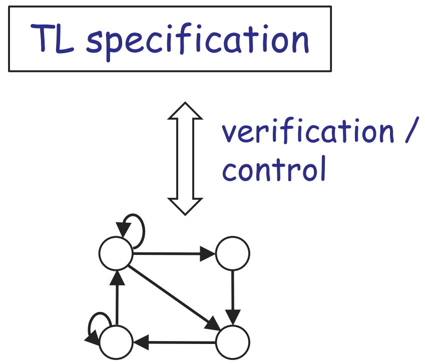


E. Aydin Gol, X.C. Ding,, M. Lazar, C. Belta ADHS 2012, CDC 2012, IEEE TAC 2014

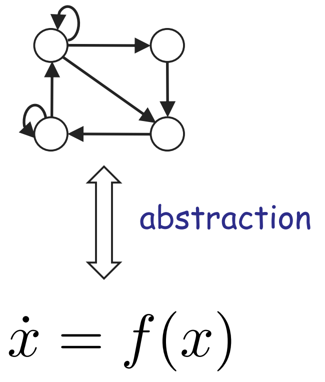
Outline

Verification and control for finite systems

Conservative control for dynamical systems

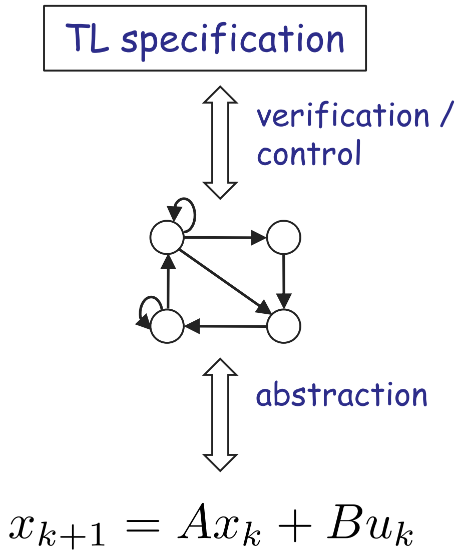


Finite quotients of continuous-space systems: main ideas



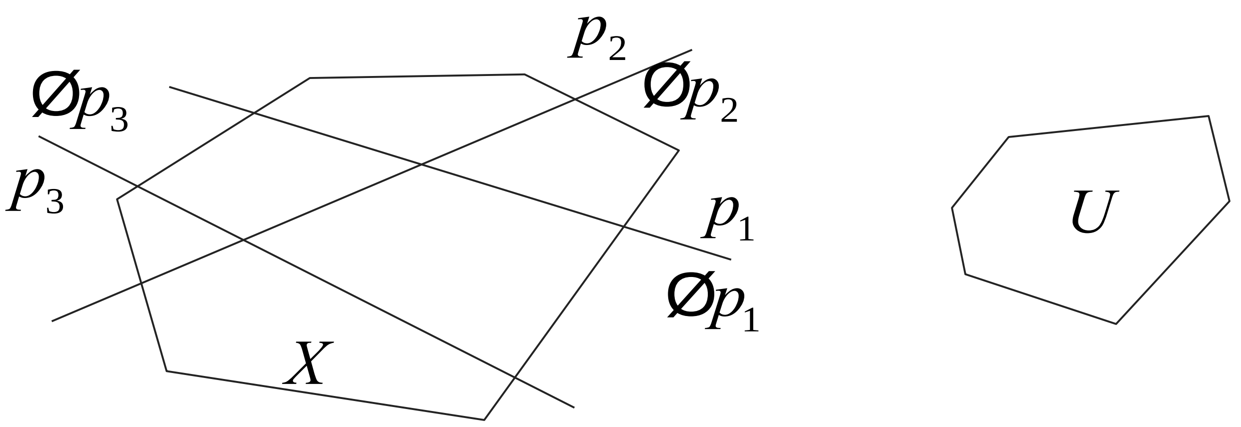
Verification for discrete-time linear systems

Control for discrete-time linear systems



TL control for discrete-time linear systems

$x_{k+1} = Ax_k + Bu_k,\ x_k$ Î $X,\ u_k$ Î U $\qquad X, U$ polytopes

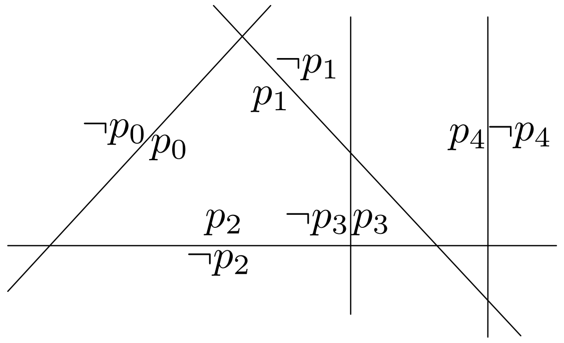


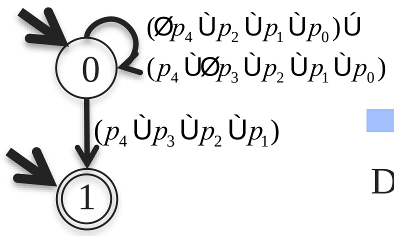
Problem Formulation: Find $X_0 \subseteq X$ and a state-feedback control strategy such that all trajectories of the closed loop system originating at X_0 satisfy an LTL formula f over the linear predicates p_i

TL control for discrete-time linear systems

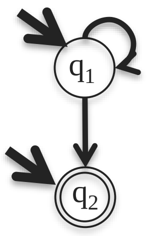
Approach: Language-guided controller synthesis and refinement

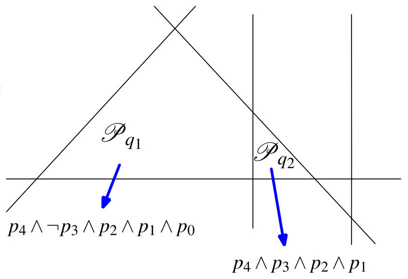
$$\Phi = (p_0 \wedge p_1 \wedge p_2) U (p_1 \wedge p_2 \wedge p_3 \wedge p_4)$$





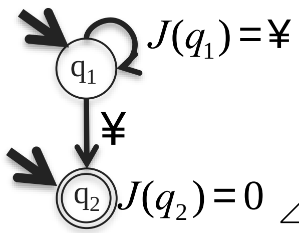
Dual

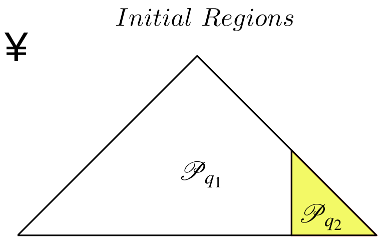




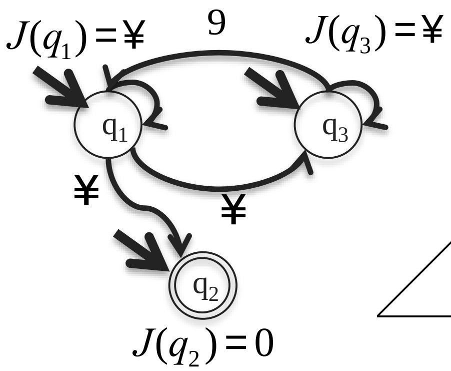
$\mathscr{P}_{q1}$
$\mathscr{P}_{q2}$
$p_4 \wedge \neg p_3 \wedge p_2 \wedge p_1 \wedge p_0$
$p_4 \wedge p_3 \wedge p_2 \wedge p_1$

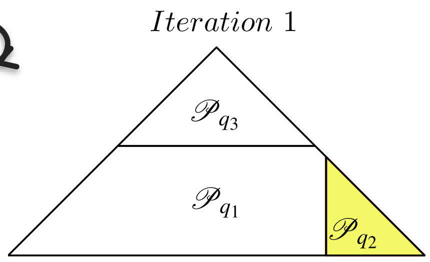
Controller Synthesis

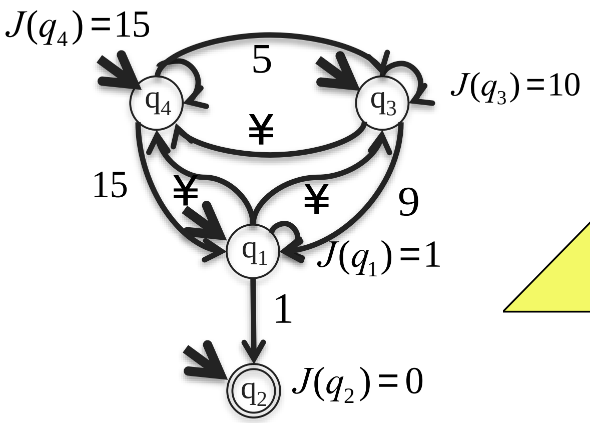


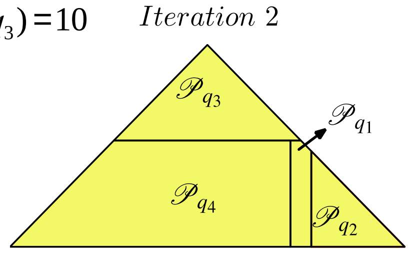


Refinement:









E. Aydin Gol, et.al., HSCC 2012, IEEE TAC 2014

TL control for discrete-time linear systems

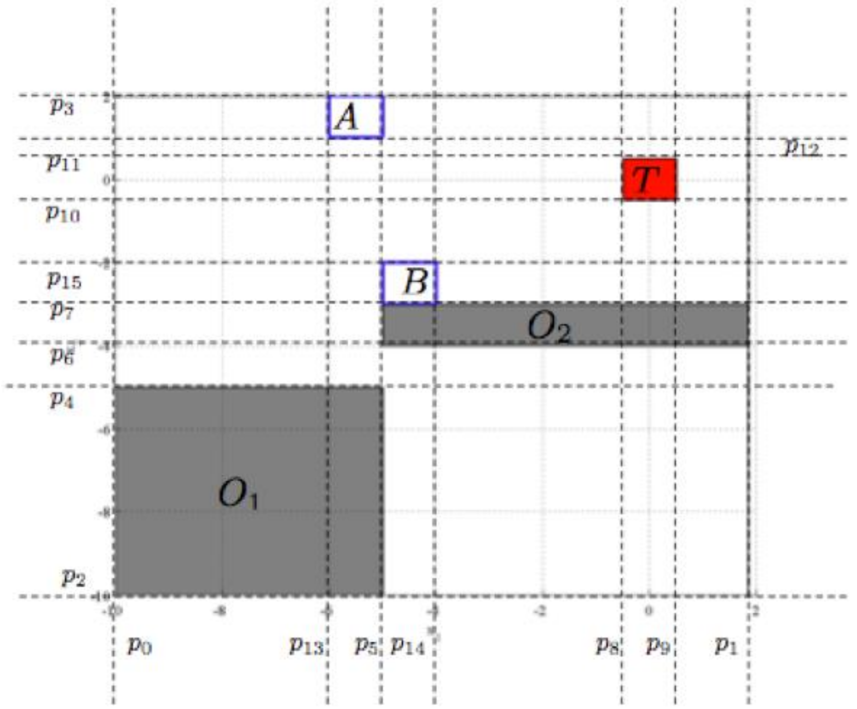
Example

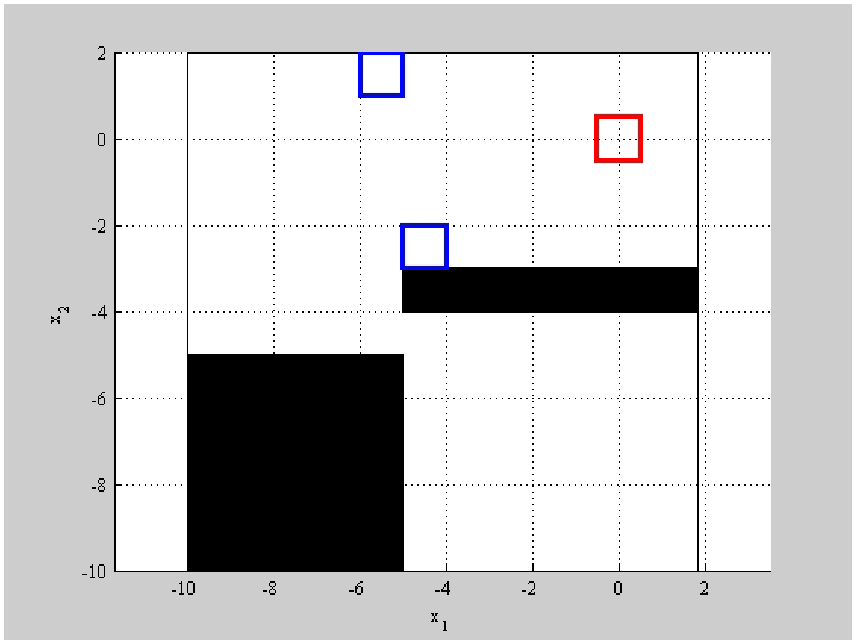
$$x_{k+1} = Ax_k + Bu_k, \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U},$$

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

"Visit region A or region B before reaching the target while always avoiding the obstacles"

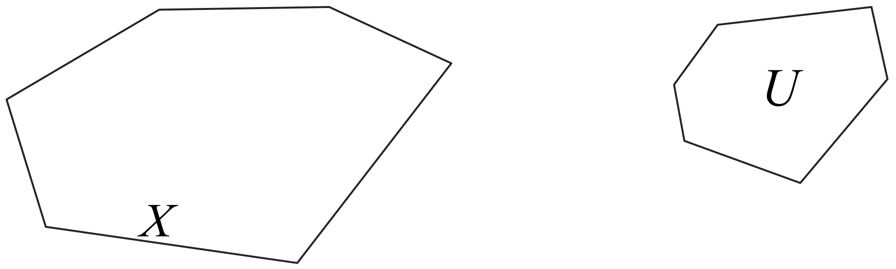
$$\Phi_2 = ((p_0 \wedge p_1 \wedge p_2 \wedge p_3 \wedge \neg(p_4 \wedge p_5) \wedge \neg(\neg p_5 \wedge \neg p_6 \wedge p_7)) \,\mathcal{U}\, (\neg p_8 \wedge p_9 \wedge \neg p_{10} \wedge p_{11})) \wedge (\neg(\neg p_8 \wedge p_9 \wedge \neg p_{10} \wedge p_{11}) \,\mathcal{U}\, ((p_5 \wedge \neg p_{12} \wedge \neg p_{13}) \vee (\neg p_5 \wedge \neg p_7 \wedge p_{14} \wedge p_{15})))$$





Optimal TL control for discrete-time linear systems

$$x_{k+1} = Ax_k + Bu_k, \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U}$$



Initial state: x_0

Reference trajectories:

$x_0^r, x_1^r \ldots$

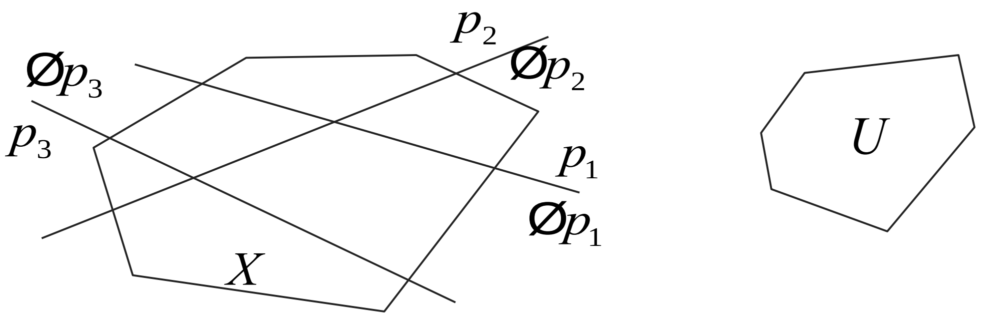
$u_0^r, u_1^r, \ldots$

Observation horizon : N

$$\begin{aligned} C(x_k, \mathbf{u}_k) = {} & (x_{k+N} - x_{k+N}^r)^\top L_N (x_{k+N} - x_{k+N}^r) \\ & + \sum_{i=0}^{N-1} \{(x_{k+i} - x_{k+i}^r)^\top L (x_{k+i} - x_{k+i}^r) \\ & + (u_{k+i} - u_{k+i}^r)^\top R (u_{k+i} - u_{k+i}^r)\}, \end{aligned}$$

Optimal TL control for discrete-time linear systems

$x_{k+1} = Ax_k + Bu_k, \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U}$



Initial state: x_0

Reference trajectories:

$x_0^r, x_1^r \dots$

$u_0^r, u_1^r, \dots$

Observation horizon : N

$$C(x_k, \mathbf{u}_k) = (x_{k+N} - x_{k+N}^r)^\top L_N (x_{k+N} - x_{k+N}^r) + \sum_{i=0}^{N-1} \left\{ (x_{k+i} - x_{k+i}^r)^\top L (x_{k+i} - x_{k+i}^r) + (u_{k+i} - u_{k+i}^r)^\top R (u_{k+i} - u_{k+i}^r) \right\},$$

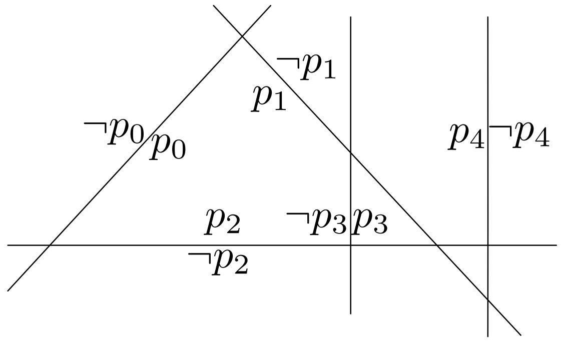
Syntactically co-safe LTL formula over linear predicates p_i

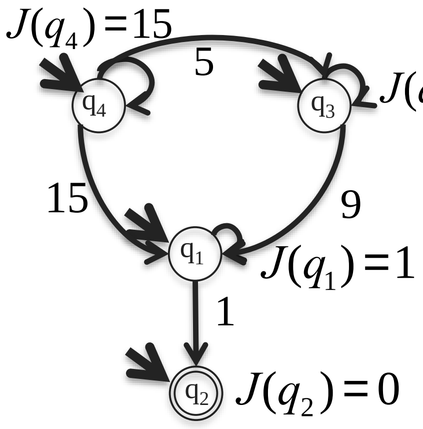
Problem Formulation: Find an optimal state-feedback control strategy such that the trajectory originating at x_0 satisfies the formula.

Optimal TL control for discrete-time linear systems

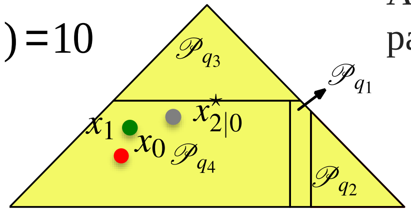
Approach

$$\Phi = (p_0 \wedge p_1 \wedge p_2) U (p_1 \wedge p_2 \wedge p_3 \wedge p_4)$$





Refined dual automaton



$N = 2, \quad x_0^r x_1^r x_2^r$

$u_0^r u_1^r$

Automaton paths: $q_4q_4q_4$, $q_4q_4q_3$, $q_4q_3q_3$, $q_4q_3q_1$, $q_4q_4q_1$, $q_4q_1q_1$, $q_4q_1q_2$

$\mathscr{P}_{q4}\mathscr{P}_{q4}\mathscr{P}_{q4}$

$\mathscr{P}_{q4}\mathscr{P}_{q4}\mathscr{P}_{q3}$

$\mathscr{P}_{q4}\mathscr{P}_{q3}\mathscr{P}_{q3}$

⋮

$$\min C(x_k, \mathbf{u}_k),$$
subject to
$$u_{i|k} \in \mathbb{U}, \quad i = 0, \ldots, N-1,$$
$$x_{i|k} \in \mathscr{P}_{q_{i|k}}, \quad i = 1, \ldots, N,$$
$$V(q_{N|k}, x_{N|k}) < V(q^*_{N|k-1}, x^*_{N|k-1}).$$

- Solve an optimization problem for each automaton path.(at each stage)
- Progress constraint: Distance to a satisfying automaton state eventually decreases.

E. Aydin Gol, M. Lazar, C. Belta, Automatica 2015

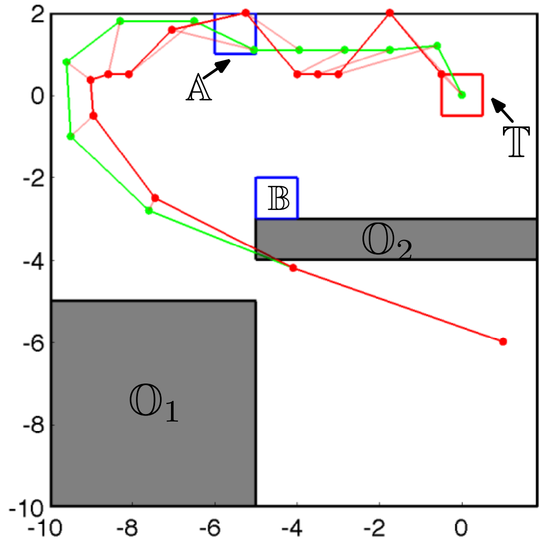
Optimal TL control for discrete-time linear systems

Example

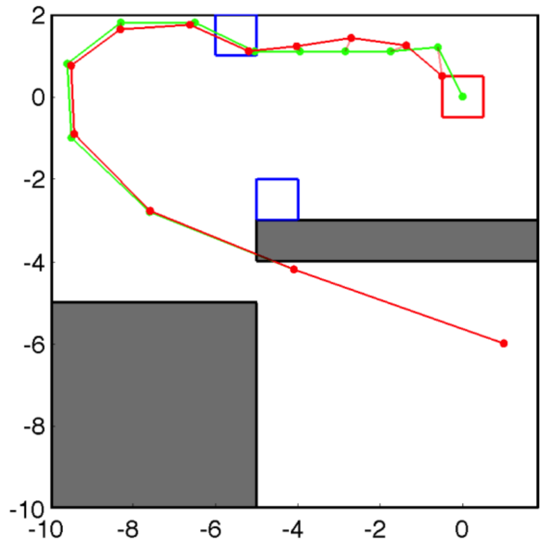
$$x_{k+1} = Ax_k + Bu_k, \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U}$$

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

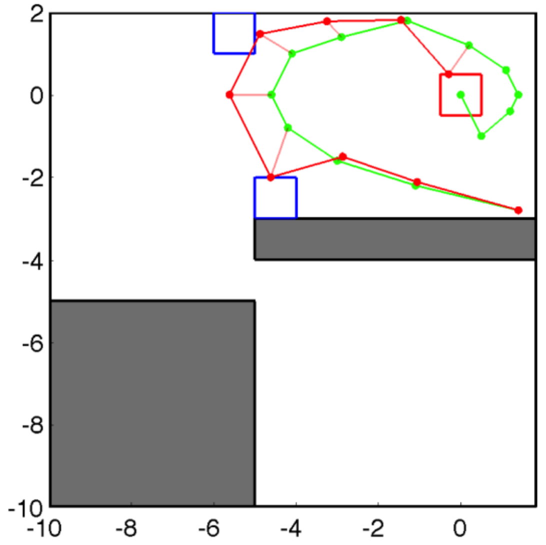
"Visit region A or region B before reaching the target while always avoiding the obstacles"



N = 2
total cost = 29.688



N = 4
total cost = 0.886



N = 6
total cost = 5.12

Reference trajectory violates the specification

Reference trajectory
Controlled trajectory

E. Aydin Gol, M. Lazar, C. Belta, Automatica 2015

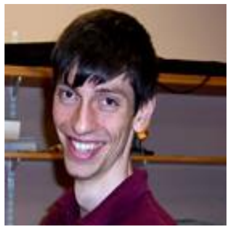
Summary

- Existing automata game algorithms can be adapted to produce control strategies for finite nondeterministic systems from LTL specifications
- Such strategies for finite systems can be directly used for to produce conservative control strategies
- Non-conservative bisimulation-type algorithms can be used for verification and control of discrete-time linear systems
- Lyapunov functions can help with the construction of finite abstractions

Ebru Aydin Gol
(now at Google)

Gregory Batt
(now at INRIA)

Dennis Ding
(now at UTRC)


Marius Kloetzer
(now at UT Iasi)

Jana Tumova
(now at KTH)

Alphan Ulusoy
(now at
Mathworks)

Boyan Yordanov
(now at Microsoft
Research)