Lecture 3.
Reachability Analysis

Talk Outline

- ➲ Symbolic Reachability Analysis
- ❑ Timed Automata (Kronos, Uppaal)
- ❑ Linear Hybrid Automata (HyTech)
- ❑ Polyhedral Flow-pipe Approximations (CheckMate)
- ❑ Orthogonal polyhedra (d/dt)

Advantages

Automated formal verification, Effective debugging tool

Moderate industrial success

In-house groups: Intel, Microsoft, Lucent, Motorola...

Commercial model checkers: FormalCheck by Cadence

Obstacles

Scalability is still a problem (about 100 state vars)

Effective use requires great expertise

Components of a Model Checker

- Modeling language
  - Concurrency, non-determinism, simple data types
- Requirements language
  - Invariants, deadlocks, temporal logics
- Search algorithms
  - Enumerative vs symbolic + many optimizations
- Debugging feedback

We focus on checking invariants of a single state machine

Reachability Problem

- Model variables X ={x1, ... xn}
  - Each var is of finite type, say, boolean
- Initialization: I(X) condition over X
- Update: T(X,X')
  - How new vars X' are related to old vars X as a result of executing one step of the program
- Target set: F(X)
- Computational problem:
  - Can F be satisfied starting with I by repeatedly applying T ?
- Graph Search problem

Symbolic Solution

Data type: region to represent state-sets

R:=I(X)

Repeat

  If R intersects T report “yes”

  Else if R contains Post(R) report “no”

  Else R := R union Post(R)

Post(R): Set of successors of states in R

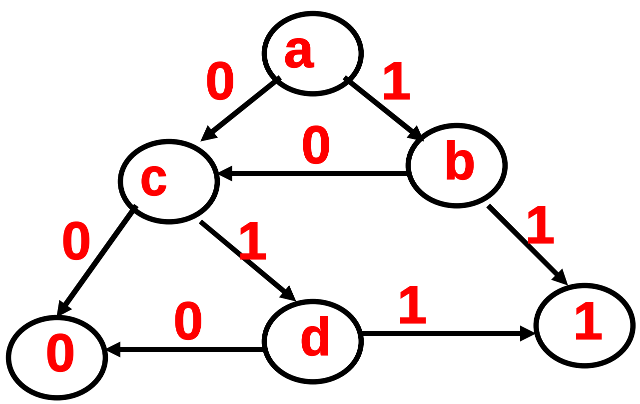
Termination may or may not be guaranteed

Symbolic Representations

- **Necessary operations on Regions**
  - Union
  - Intersection
  - Negation
  - Projection
  - Renaming
  - Equality/containment test
  - Emptiness test
- **Different choices for different classes**
  - BDDs for boolean variables in hardware verification
  - Size of representation as opposed to number of states

Ordered Binary Decision Diagrams

Popular representations for Boolean functions



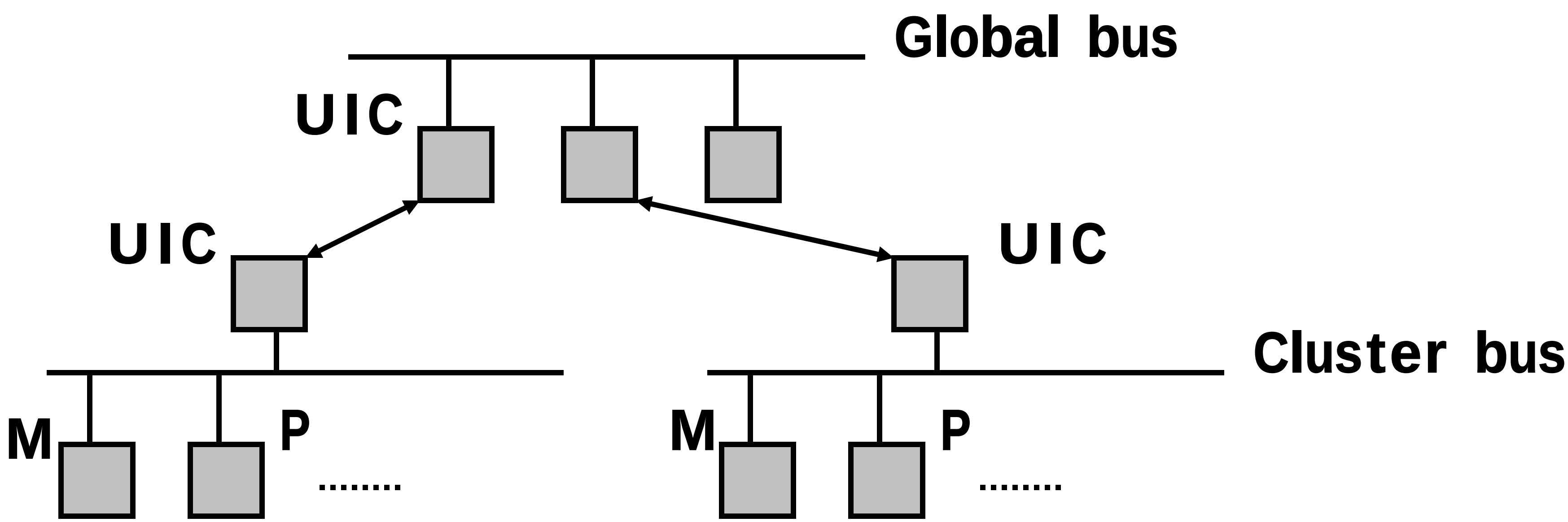
Like a decision graph
No redundant nodes
No isomorphic subgraphs
Variables tested in fixed order

Function: (a and b) or (c and d)

Key properties:
- Canonical!
- Size depends on choice of ordering of variables
- Operations such as union/intersection are efficient

Example: Cache consistency: Gigamax

Real design of a distributed multiprocessor



Read-shared/read-owned/write-invalid/write-shared/...

Deadlock found using SMV

Similar successes: IEEE Futurebus+ standard, network RFCs

Reachability for Hybrid Systems

- Same algorithm works in principle
- What's a suitable representation of regions?
  - Region: subset of R^k
  - Main problem: handling continuous dynamics
- Precise solutions available for restricted continuous dynamics
  - Timed automata
  - Linear hybrid automata
- Even for linear systems, over-approximations of reachable set needed

Talk Outline

- ✓ Symbolic Reachability Analysis
- ➲ Timed Automata (Kronos, Uppaal)
- ❑ Linear Hybrid Automata (HyTech)
- ❑ Polyhedral Flow-pipe Approximations (CheckMate)
- ❑ Orthogonal polyhedra (d/dt)

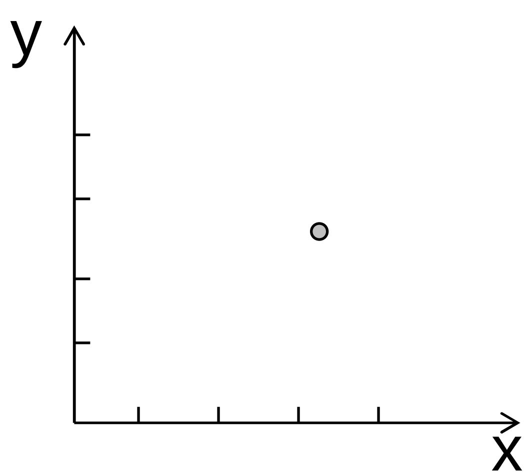
Timed Automata

- Only continuous variables are timers
- Invariants and Guards: x<const, x>=const
- Actions: x:=0
- Reachability is decidable
- Clustering of regions into zones desirable in practice
- Tools: Uppaal, Kronos, RED …
- Symbolic representation: matrices
- Techniques to construct timed abstractions of general hybrid systems

Zones
Symbolic computation

State

(n, $x=3.2$, $y=2.5$)



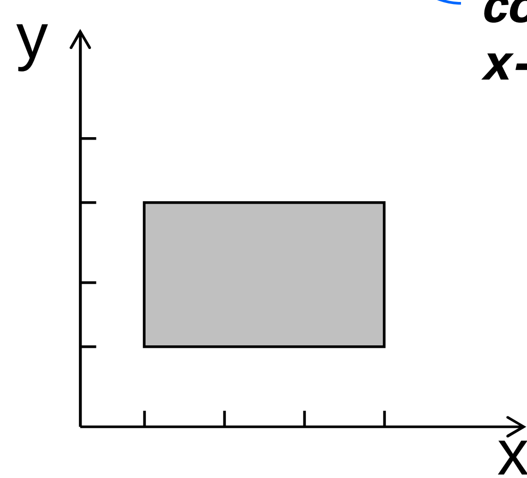
Symbolic state (set)

(n, $1 \le x \le 4, 1 \le y \le 3$)

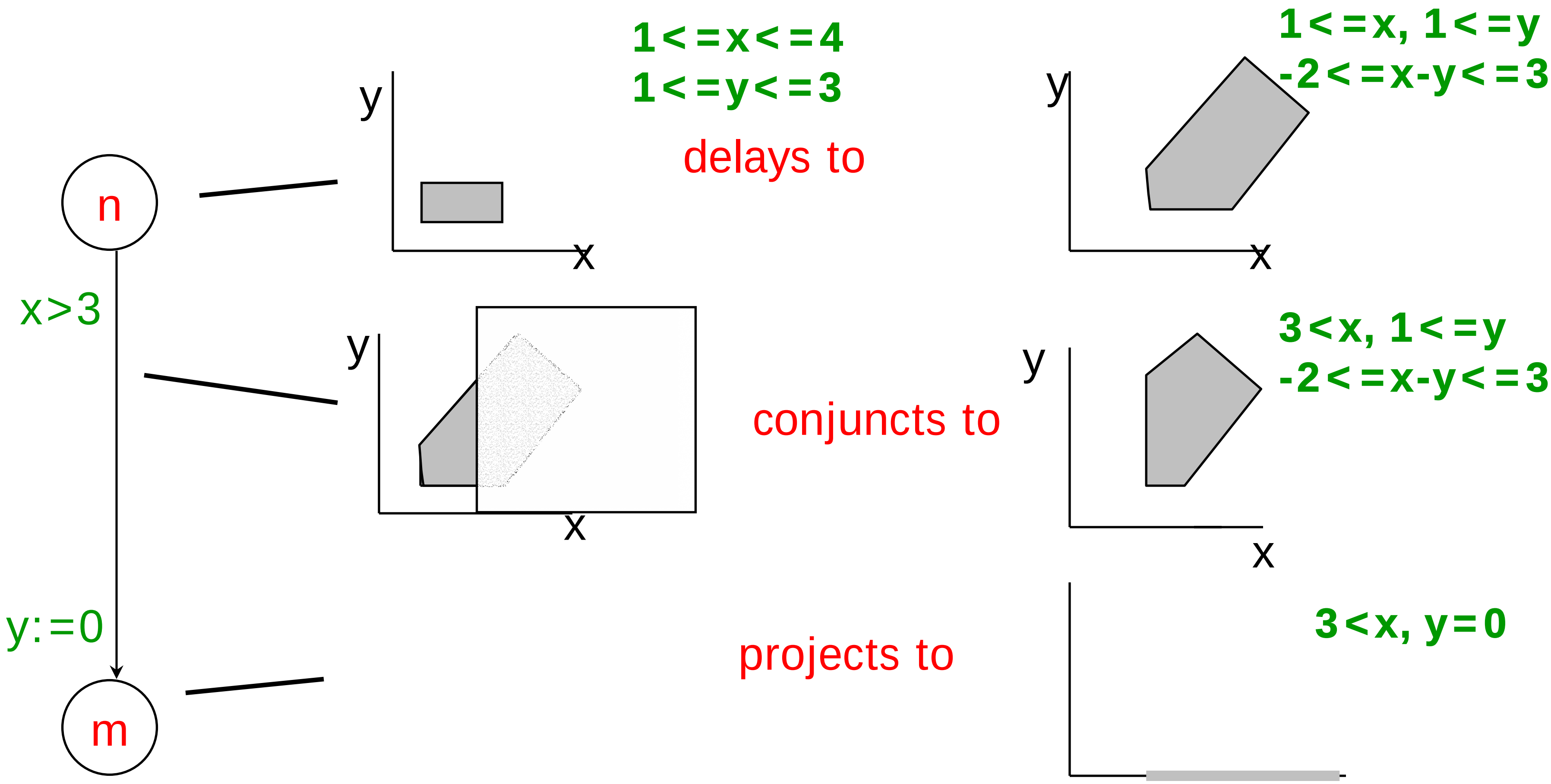
Zone:

conjunction of

x-y<=n, x<=>n



Symbolic Transitions



Thus (n,1<=x<=4,1<=y<=3) ==> (m,3<x, y=0)

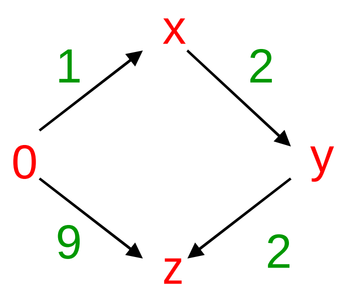
Canonical Data-structures for Zones
Difference Bounded Matrices

When are two sets of constraints equivalent?

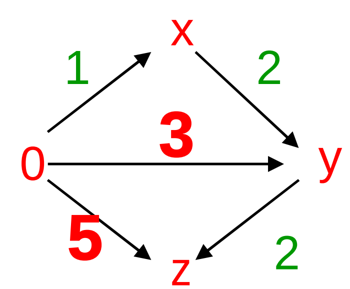
D1

x<=1
y-x<=2
z-y<=2
z<=9

Graph



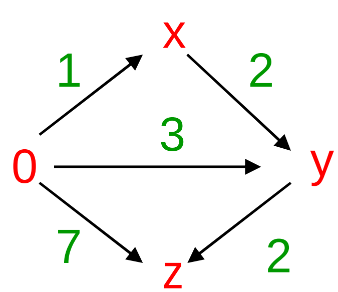
Shortest Path Closure



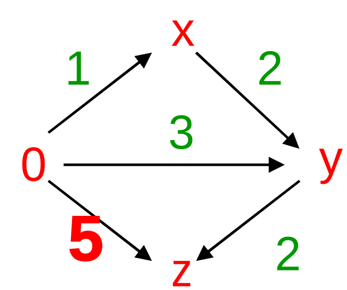
D2

x<=1
y-x<=2
y<=3
z-y<=2
z<=7

Graph



Shortest Path Closure



Difference Bounds Matrices

- Matrix representation of constraints (bounds on a single clock or difference betn 2 clocks)
- Reduced form obtained by running all-pairs shortest path algorithm
- Reduced DBM is canonical
- Operations such as reset, time-successor, inclusion, intersection are efficient
- Popular choice in timed-automata-based tools

Talk Outline

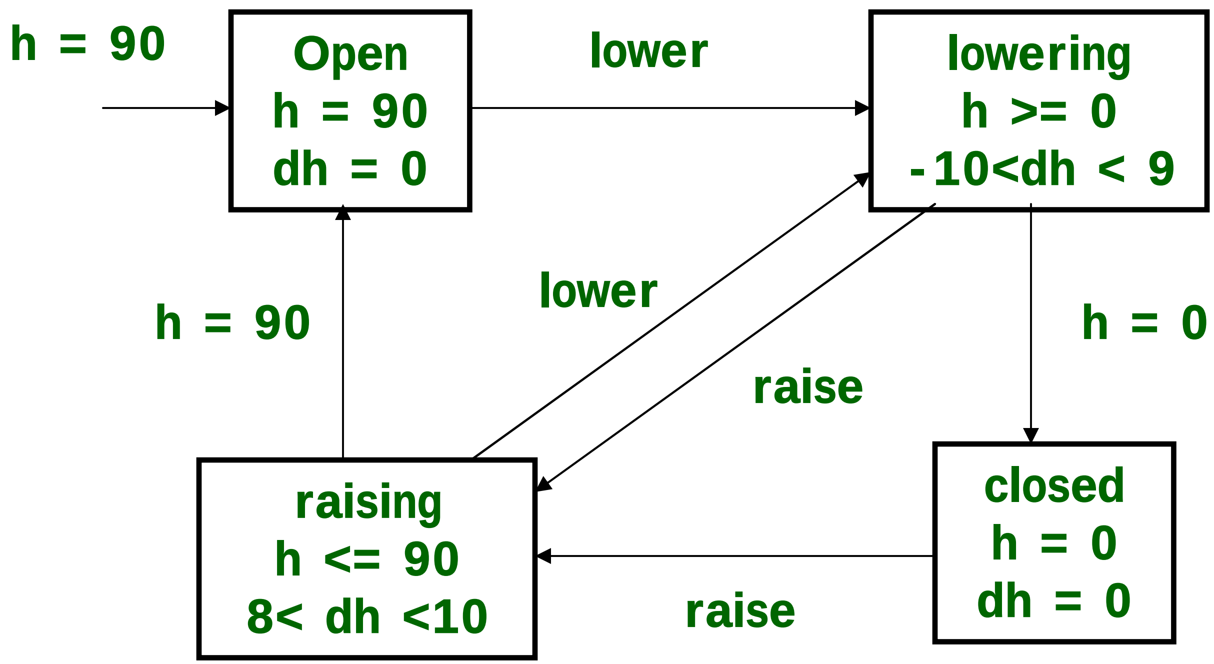
- ✓ Symbolic Reachability Analysis
- ✓ Timed Automata (Kronos, Uppaal)
- ➲ Linear Hybrid Automata (HyTech)
- ❑ Polyhedral Flow-pipe Approximations (CheckMate)
- ❑ Orthogonal polyhedra (d/dt)

Linear Hybrid Automata

- Invariants and guards: linear ($Ax \le b$)
- Actions: linear transforms ($x := Ax$)
- Dynamics: time-invarint, state-independent specified by a convex polytope constraining rates
  E.g. $2 < \dot{x} \le 3,\ \dot{x} = \dot{y}$

- Tools: HyTech
- Symbolic representation: Polyhedra
- Methodology: abstract dynamics by differential inclusions bounding rates

Example LHA
Gate for a railroad controller



Reachability Computation

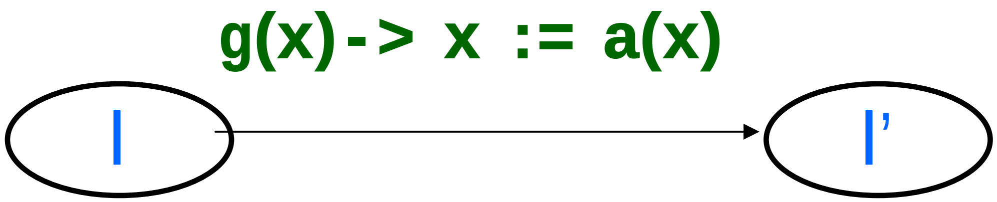
Basic element: (location l, polyhedron p)

Set of visited states: a list of (l,p) pairs

Key steps:

- Compute "discrete" successors of (l,p)
- Compute "continuous" successor of (l,p)
- Check if p intersects with "bad" region
- Check if newly found p is covered by already visited polyhedra p1,..., pk (expensive!)

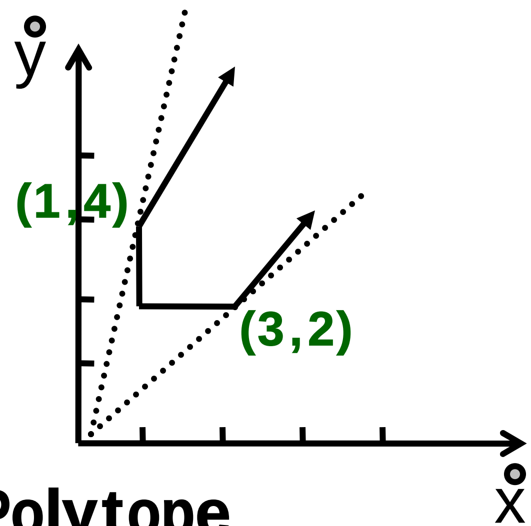
Computing Discrete Successors



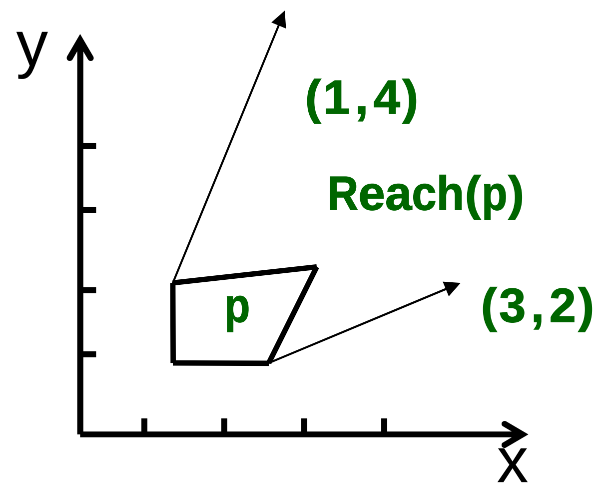
Discrete successor of (l,p)

- Intersect p with g (result r is a polyhedron)
- Apply linear transformation a to r (result r' is a polyhedron)
- Successor is (l',r')

Computing Time Successor

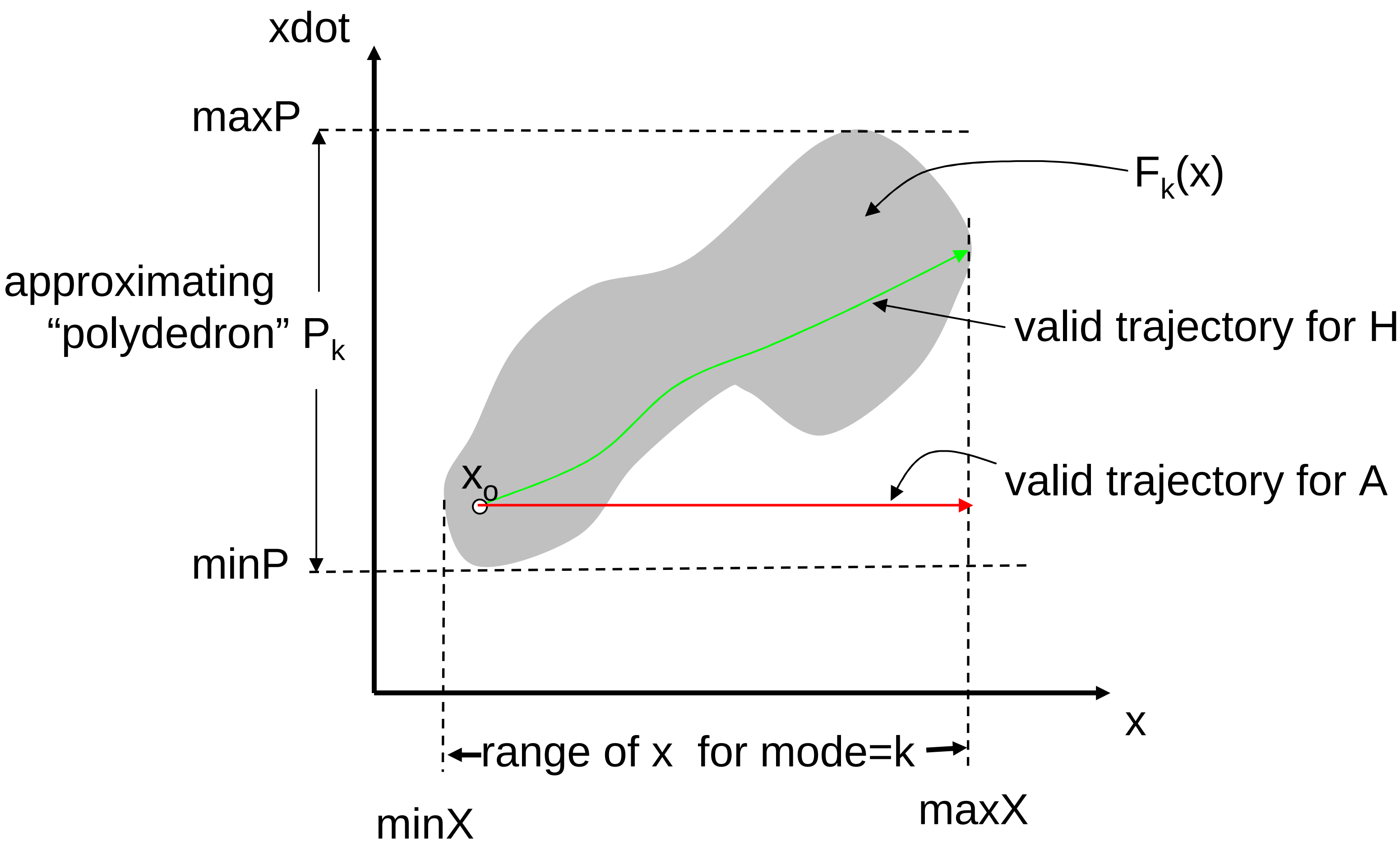


Rate Polytope

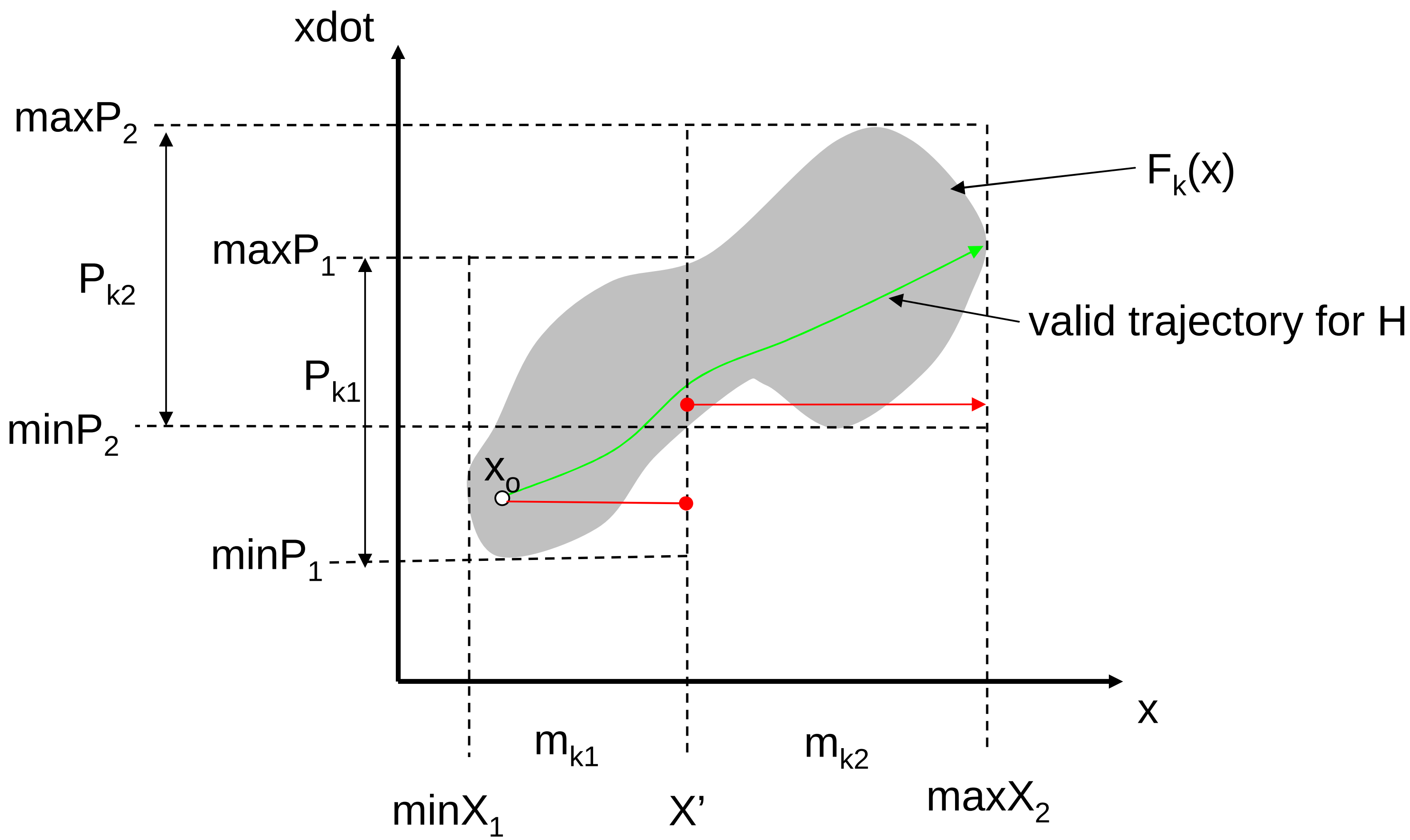


- Thm: If initial set p, invariant I, and rate constraint r, are polyhedra, then set of reachable states is a polyhedron (and computable)
- Basically, apply extremal rates to vertices of p

Linear Phase-portrait Approximation



Improving Linear Phase-Portrait Approximations: Mode Splitting



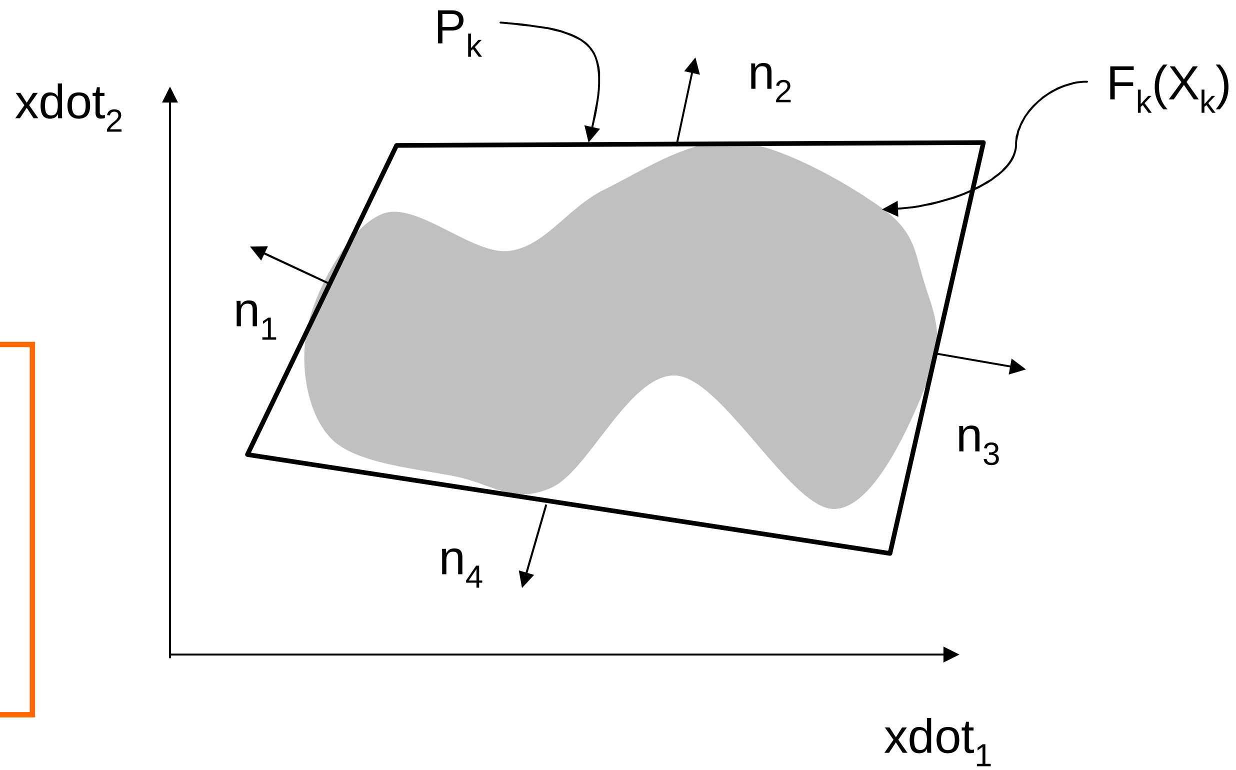
Computing Approximation

In general find P_k by solving the following optimization problem in a set of face-normal directions:

$$\max_{x,\, xdot} \quad n_i^T \, xdot$$

$$\text{s.t.} \quad xdot \in F_k(x)$$

$$x \in X_k$$



Problem: How to choose the n_i.

Linear Phase-Portrait Approximations

- guaranteed conservative approximations
- refinement introduces more discrete states
- for bounded hybrid automata, arbitrarily close approximation can be attained using mode splitting
- sufficient to use rectangular phase-portrait approximations ($n_i^T = [0...1...0]$)

Summary: Linear Hybrid Automata

- HyTech implements this strategy
- Core computation: manipulation of polyhedra
- Bottlenecks
  - proliferation of polyhedra (unions)
  - computing with higher dimensional polyhedra

- Many applications (active structure control, Philips audio control protocol, steam boiler…)

Talk Outline

- ✓ Symbolic Reachability Analysis
- ✓ Timed Automata (Kronos, Uppaal)
- ✓ Linear Hybrid Automata (HyTech)
- ➲ Polyhedral Flow-pipe Approximations (CheckMate)
- ❑ Orthogonal polyhedra (d/dt)

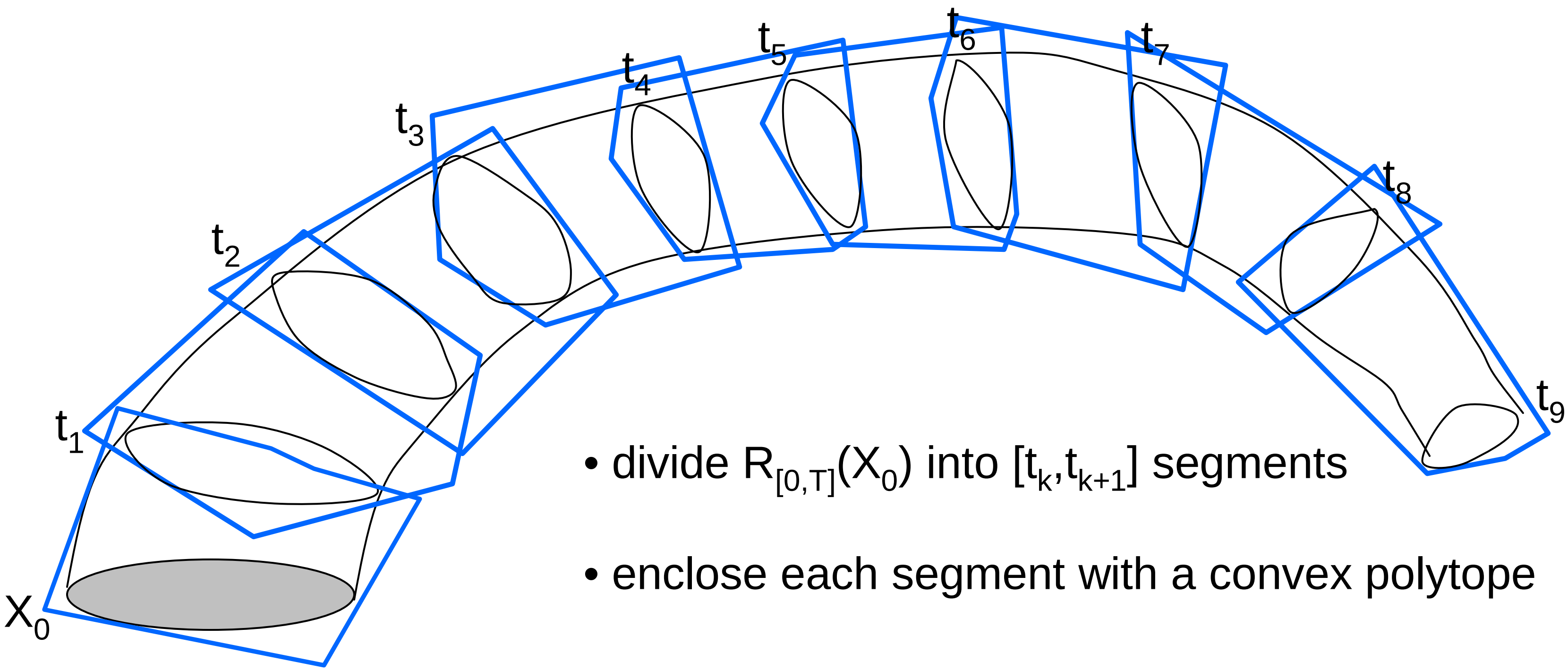
Approximating Reachability

Given a continuous dynamic system,

$$\dot{x} = F(x),$$

and a set of initial states, X_0 , conservatively approximate $Reach_{[0,t]}(X_o,F)$.

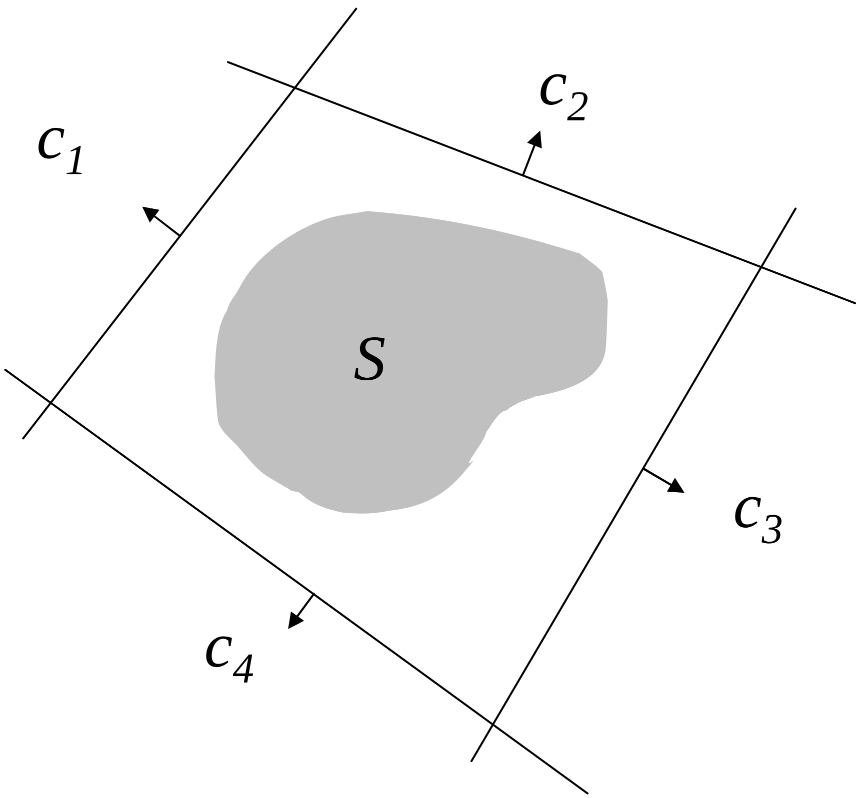
Polyhedral Flow Pipe Approximations



- divide $R_{[0,T]}(X_0)$ into $[t_k, t_{k+1}]$ segments
- enclose each segment with a convex polytope
- $R^M_{[0,T]}(X_0)$ = union of polytopes

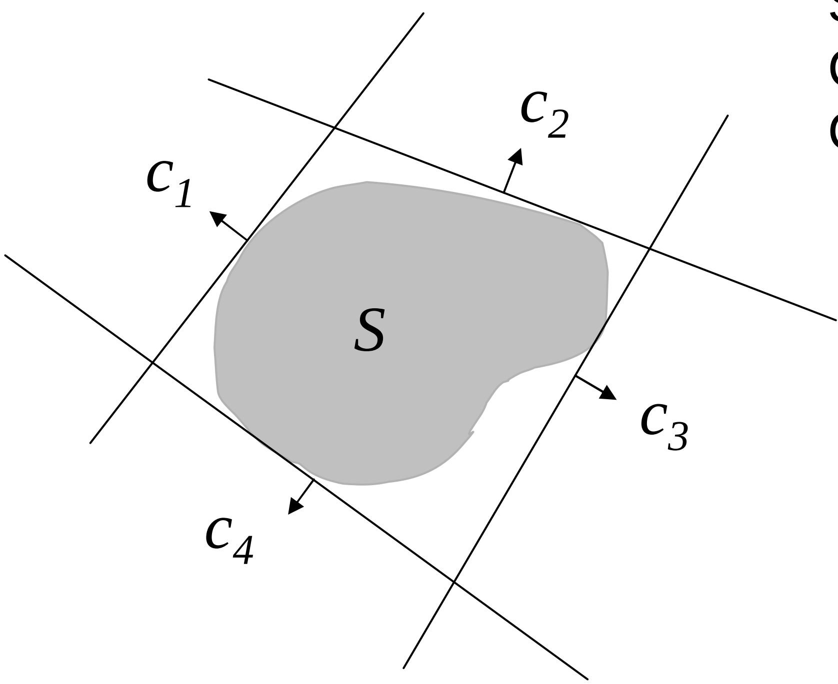
A. Chutinan and B. H. Krogh, Computing polyhedral approximations to dynamic flow pipes, IEEE CDC, 1998

Wrapping Hyperplanes Around a Set



Step 1:
Choose normal vectors, $c_{1,}...,c_m$

Wrapping Hyperplanes Around a Set



Step 2:
Compute optimal d in $Cx \leq d$,
$C^T = [c_1 \cdots c_m]$:

$$d_i = \max_{x \in S} \; c_i^T x$$

Wrapping a Flow Pipe Segment

Given normal vectors c_i, we wrap $R_{[tk,tk+1]}(X_0)$ in a polytope by solving for each i

$$d_i = \max_{x_o, t} \; c_i^T x(t, x_0)$$

$$\text{s.t.} \quad x_0 \in X_0$$

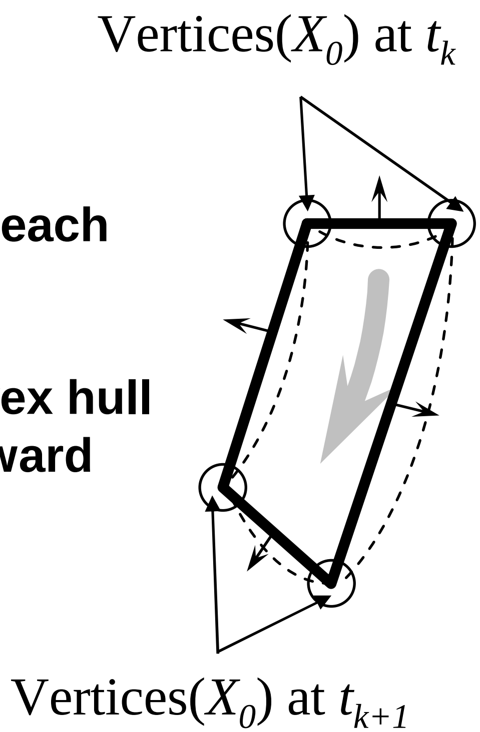
$$t \in [t_k, t_{k+1}]$$

Optimization problem is solved by embedding simulation into objective function computation

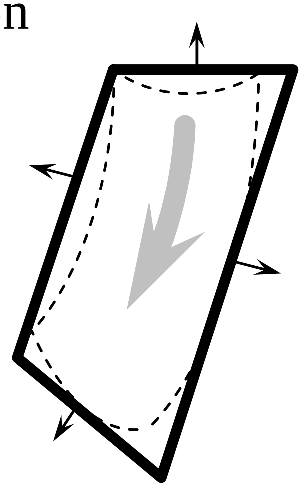
Flow Pipe Segment Approximation

Step 1.
a. Simulate trajectories from each vertex of X_0.

b. Take the convex hull and identify outward normal vectors.



Step 2.
Solve optimization for d_i



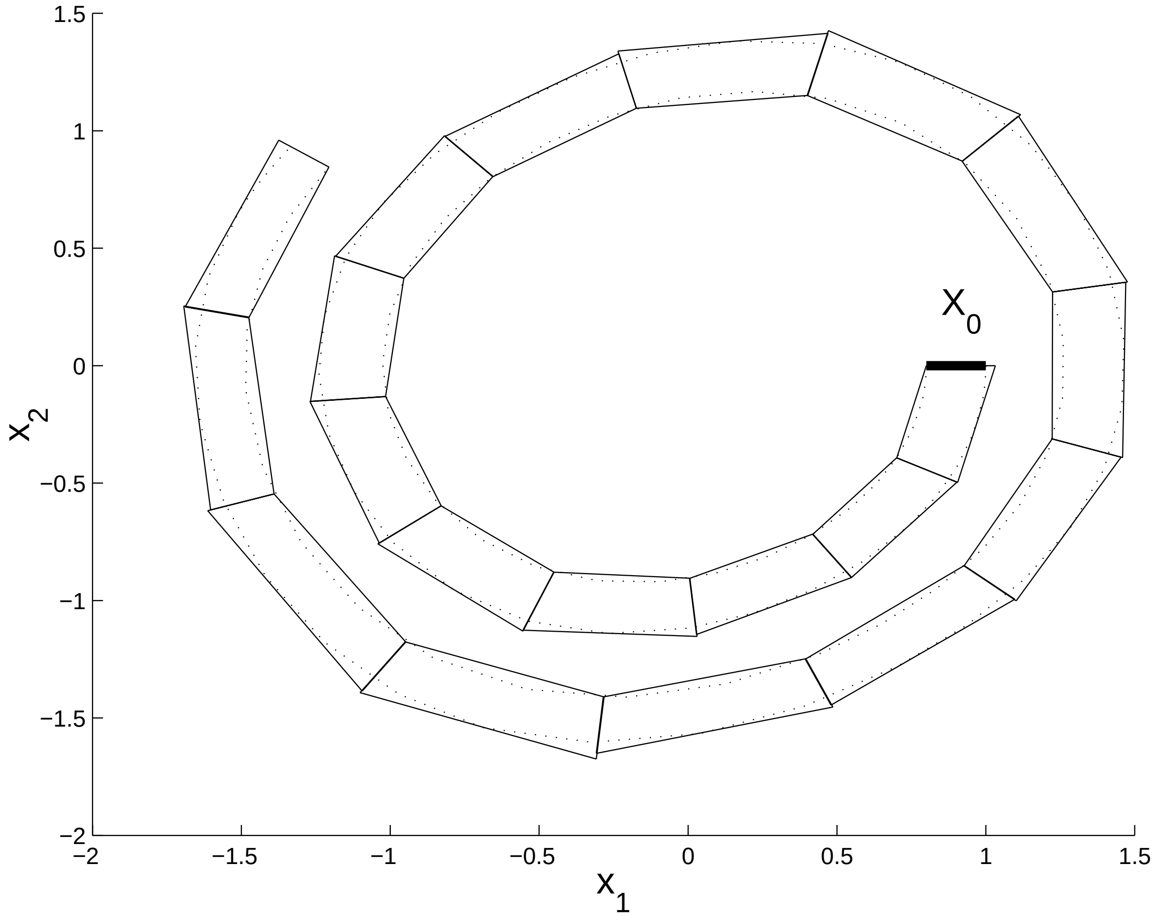
flow pipe segment approximated by
$\{ x \mid c_i^T x \leq d_i, \forall i \}$

Improvements for Linear Systems

- $\dot{x} = Ax \Rightarrow x(t, x_0) = e^{At} x_0$
- No longer need to embed simulation into optimization
- Flow pipe segment computation depends only on time step Δt
- A segment can be obtained by applying e^{At} to another segment of the same Δt

$$\boxed{\hat{R}_{[t,t+\Delta t]}(X_0) = e^{At} \hat{R}_{[0,\Delta t]}(X_0)}$$

Example 1: Van der Pol Equation



Van der Pol Equation

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -0.2(x_1^{\ 2} - 1)x_2 - x_1$$

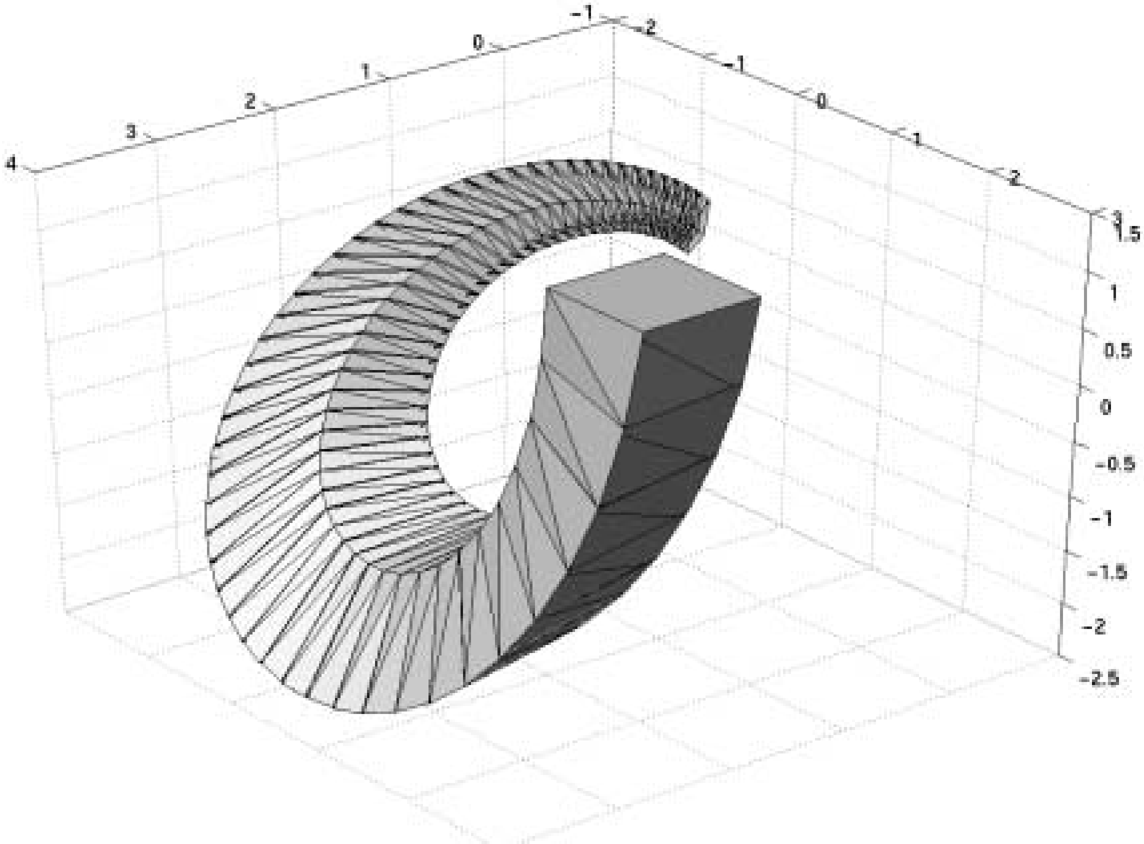
Initial Set

$$X_0 = \{0.8 \le x_1 \le 1, x_2 = 0\}$$

Uniform time step

$$\Delta t_k = 0.5$$

Example 2: Linear System



$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -2 \end{bmatrix}$$

Vertices for X_0

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}, \text{ and } \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Uniform time step

$\Delta t_k = 0.1$

Summary: Flow Pipe Approximation

- Applies in arbitrary dimensions
- Approximation error doesn't grow with time
- Estimation error (Hausdorff distance) can be made arbitrarily small with $\Delta t < \delta$ and size of $X_0 < \delta$
- Integrated into a complete verification tool (CheckMate)

Talk Outline

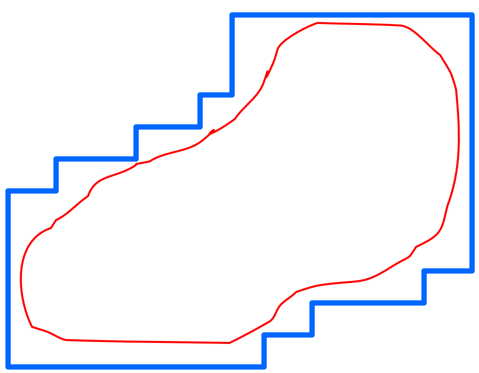
- ✓ Symbolic Reachability Analysis
- ✓ Timed Automata (Kronos, Uppaal)
- ✓ Linear Hybrid Automata (HyTech)
- ✓ Polyhedral Flow-pipe Approximations (CheckMate)
- ➲ Orthogonal Polyhedra (d/dt)

Approximations by Orthogonal Polyhedra

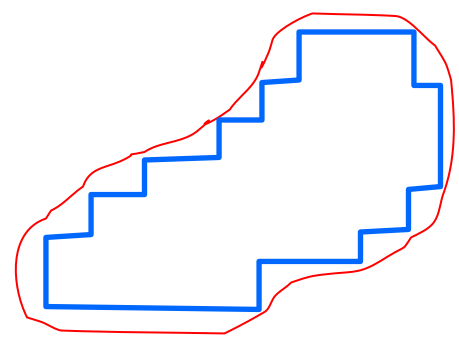
Non-convex orthogonal polyhedra (unions of hyperrectangles)

<u>Motivations</u>

- canonical representation, efficient manipulation in *any dimension* $\Rightarrow$ *easy extension to hybrid systems*
- *termination* can be guaranteed



Over-approximation



Under-approximation

Reachability Analysis of Continuous Systems

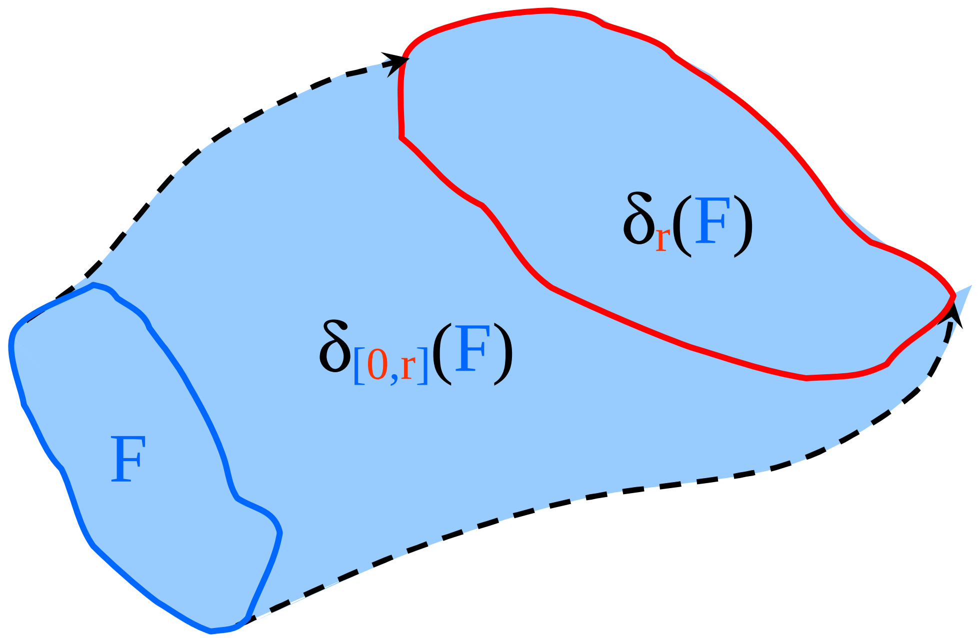
A continuous system $\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x})$; f is Lipschitz

$\mathbf{x}(0) \in \mathrm{F}$, set of *initial states*

Problem

Find an *orthogonal polyhedron over-approximating* the *reachable set* from F

Successor Operator



Reachable set from F: $\delta(F) = \delta_{[0,\infty)}(F)$

Algorithm for Calculating $\delta(F)$

r : *time step*

$P^0 := F$;
repeat $k = 0, 1, 2$ ..
$\quad P^{k+1} := P^k \cup \delta_{[0,r]}(P^k)$;
until $P^{k+1} = P^k$

Use orthogonal polyhedra to

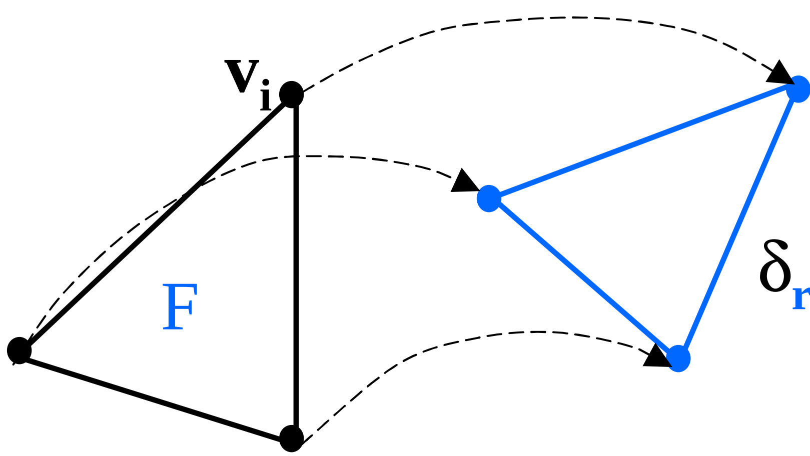
- represent P^k
- approximate $\delta_{[0,r]}$

Reachability of Linear Continuous Systems

A linear system $\dot{\mathbf{x}} = A\mathbf{x}$; F is the set of initial states

$\delta_{\mathbf{r}}(F) = e^{\mathbf{A}\mathbf{r}} F$

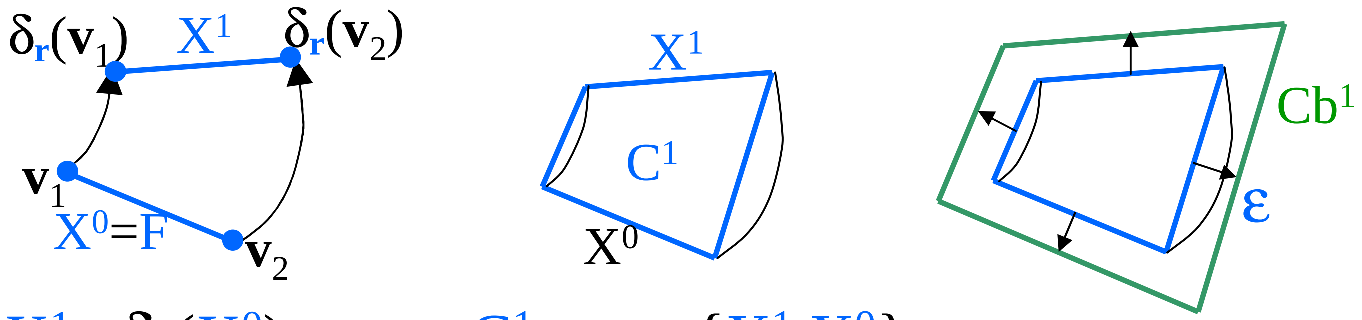
F is a convex polyhedron: $F = \mathrm{conv}\{\mathbf{v_1},..,\mathbf{v_m}\}$



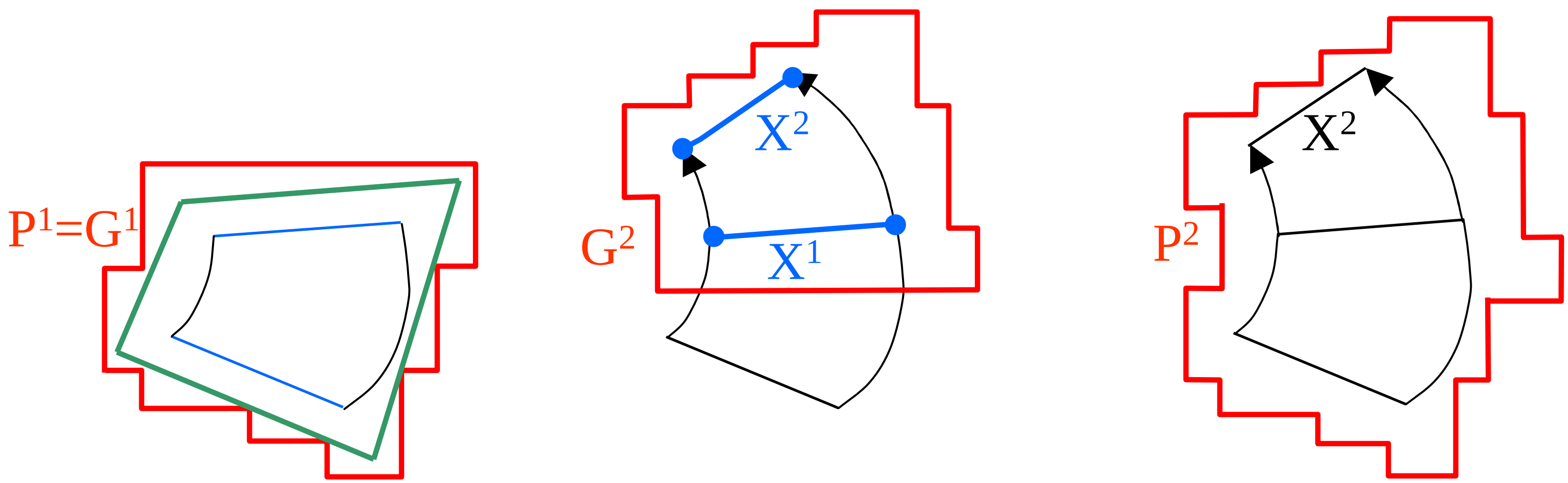
$\delta_{\mathbf{r}}(\mathbf{v_i}) = e^{\mathbf{A}\mathbf{r}}\,\mathbf{v_i}$

$\delta_{\mathbf{r}}(F) = \mathrm{conv}\{\delta_{\mathbf{r}}(\mathbf{v_1}),.., \delta_{\mathbf{r}}(\mathbf{v_m})\}$

Over-Approximating the Reachable Set



$X^1 = \delta_{\mathbf{r}}(X^0)$ $C^1 = \mathrm{conv}\{X^1, X^0\}$



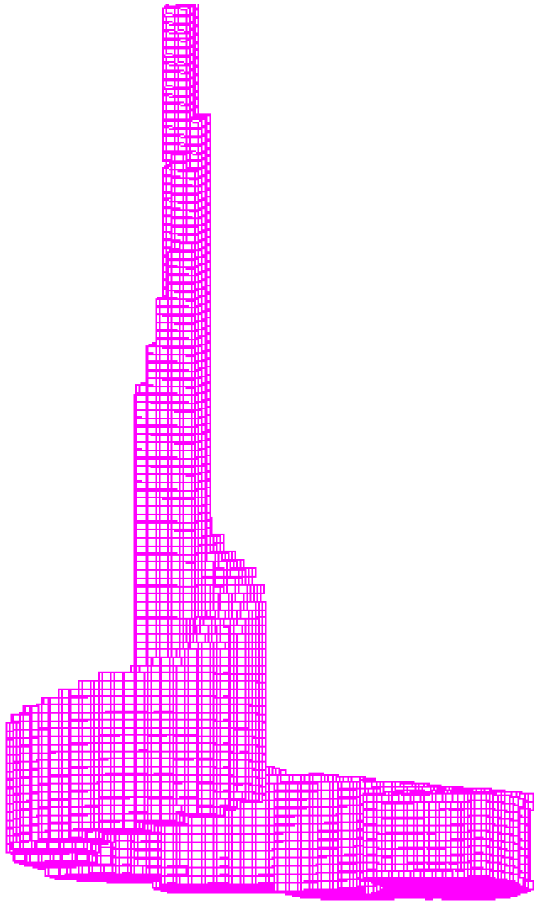
$\delta_{[0,r]}(F) \subseteq G^1$ $\delta_{[r,2r]}(F) \subseteq G^2$ $\delta_{[0,2r]}(F) \subseteq P^2 = G^1 \cup G^2$

Extension to under-approximations

Example

$$\dot{\mathbf{x}} = A\mathbf{x},\quad F = [0.025, 0.05] \times [0.1, 0.15] \times [0.05, 0.1]$$

$$A = \begin{pmatrix} 1.0 & 4.0 & 0.0 \\ 4.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.5 \end{pmatrix}$$

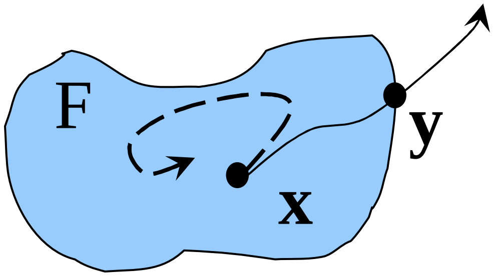


Nonlinear Systems

A continuous system $\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x})$; f is Lipschitz

$\mathbf{x}(0) \in$ F, set of initial states

➔ 'Face lifting' technique, inspired by [Greenstreet 96]



- *Continuity* of trajectories ⇒ compute from the boundary of F

- The initial set F is a *convex polyhedron*
  The boundary of F: union of its *faces*

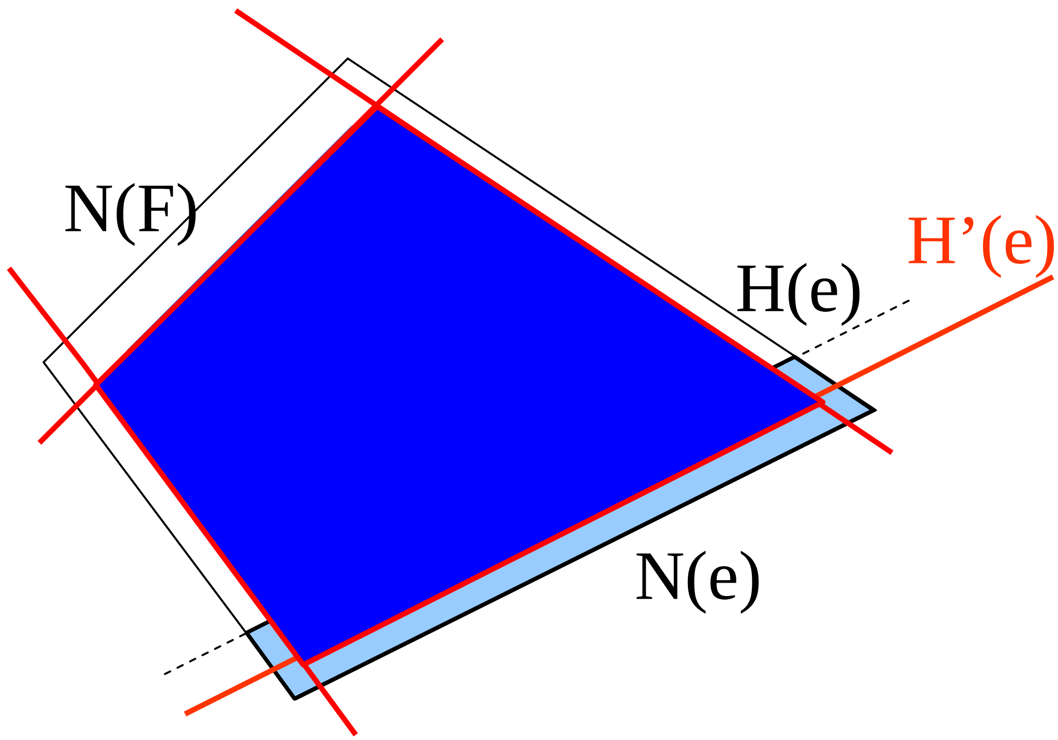
Over-Approximating $\delta_{[0,r]}(F)$

Step 1: *rough approximation* N(F)

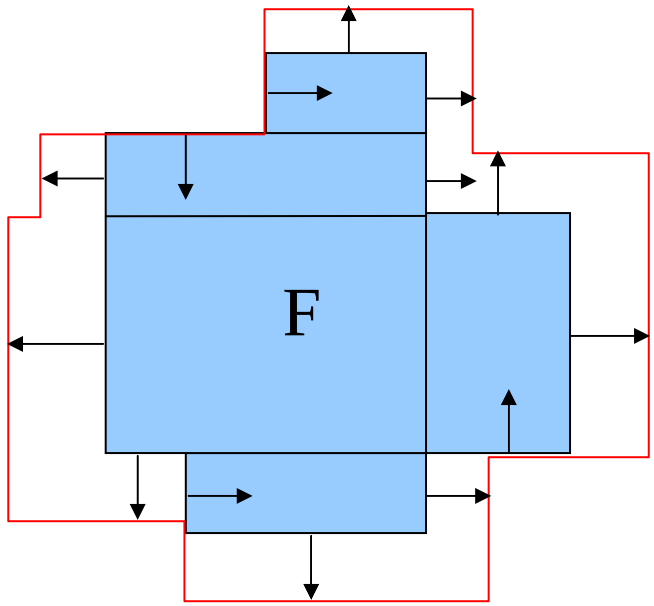
Step 2: *more accurate approximation*

f_e : projection of f on the outward normal to face e

$\hat{f}_e$: maximum of f_e over the neighborhood N(e) of e



Computation Procedure



- Decompose F into non-overlapping hyper-rectangles
- Apply the lifting operation to each hyper-rectangle (faces on the boundary of F)
- Make the union of the new hyper-rectangles

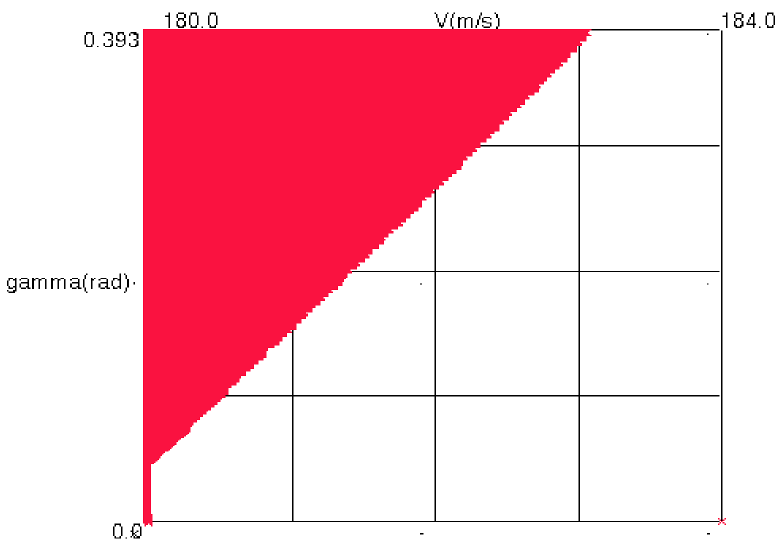
Example: Collision Avoidance

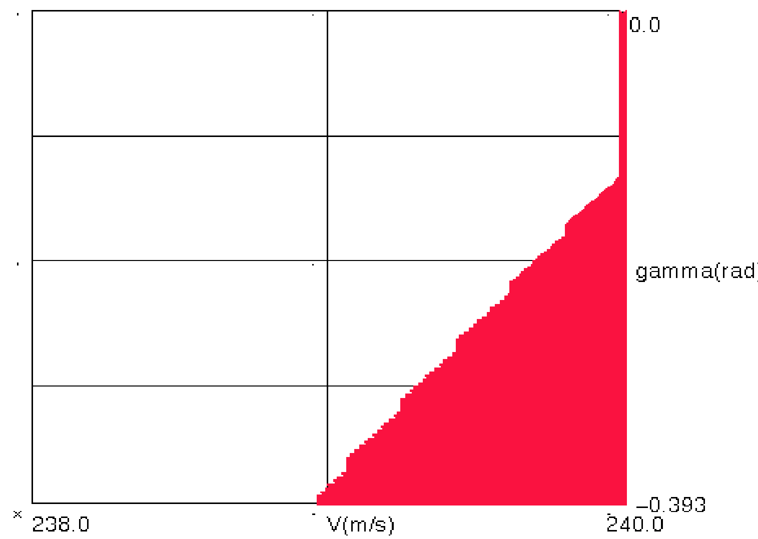
x_1 : velocity; x_2 : flight path angle

$$\dot{x}_1 = \frac{a_D x_1^2}{m} - g\sin x_2 + \frac{u_1}{m} \qquad \dot{x}_2 = \frac{a_L x_1(1-cx_2)}{m} - \frac{g\cos x_2}{x_1} + \frac{a_L c x_1}{m} u_2$$

$$u_1 = [T_{min}, T_{max}] \text{ (thrust); } u_2 = [\Theta_{min}, \Theta_{max}] \text{ (pitch angle)}$$

$$\mathbf{P} = [V_{min}, V_{max}] \times [\gamma_{min}, \gamma_{max}]$$





d/dt Summary

Techniques generalize to

Hybrid Systems

Dynamics with uncertain inputs

Controller synthesis problems

Tool available from Verimag

Applications

- collision avoidance (4 continuous variables, 1 discrete state)
- double pendulum (3 continuous variables, 7 discrete states)
- freezing system (6 continuous variables, 9 discrete states)