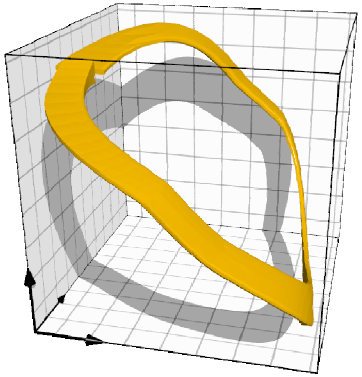
Verification of Hybrid Systems

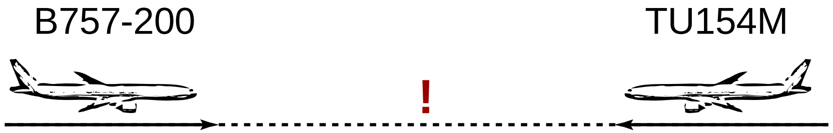


Goran Frehse
Universite Grenoble 1, Verimag

- with work from Thao Dang, Antoine Girard and Colas Le Guernic -

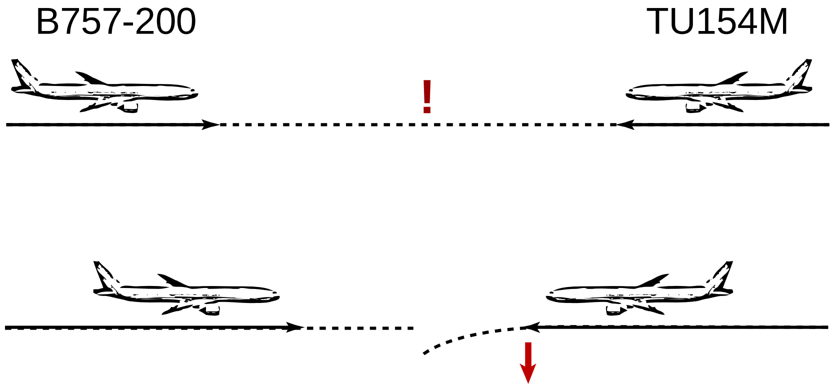
MOVEP'08, June 25, 2008

Boeing & Tupolew Collision



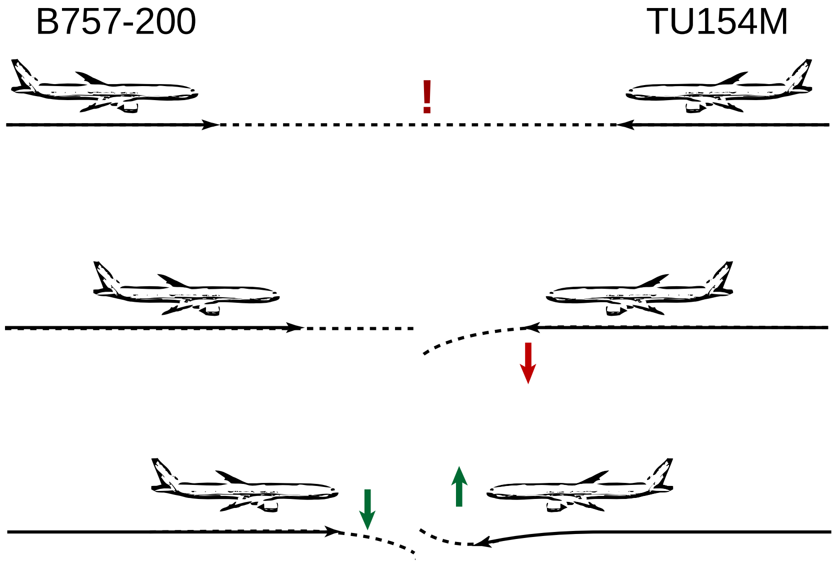
- **Überlingen, July 1, 2002**
- **21:33:03**
  - – Alarm from Traffic Collision Avoidance System (TCAS)

Boeing & Tupolew Collision



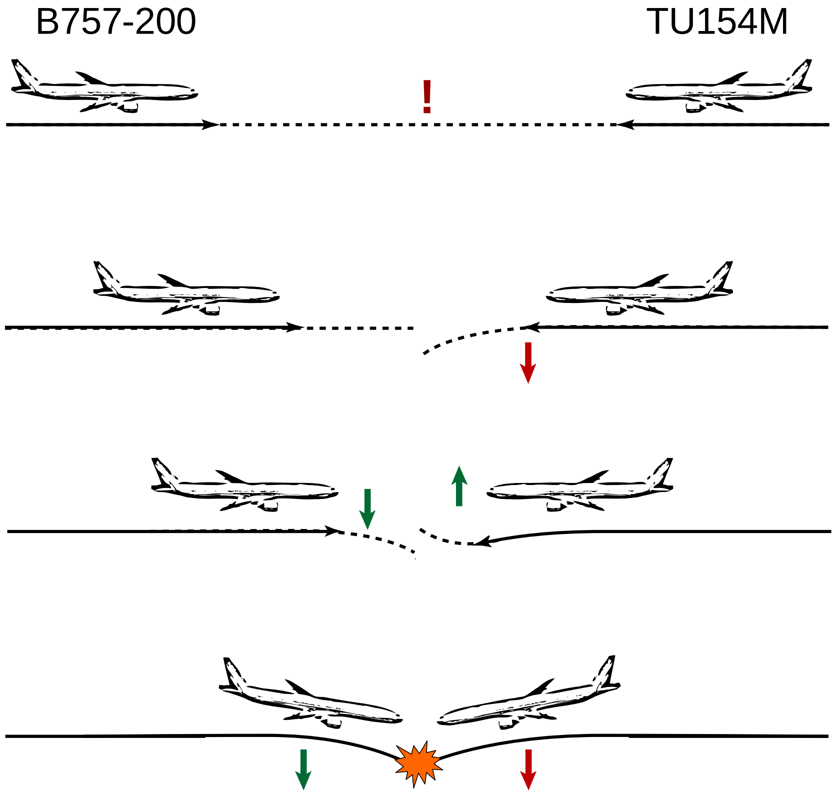
- **Überlingen, July 1, 2002**
- **21:33:03**
  - Alarm from Traffic Collision Avoidance System (TCAS)
- **21:34:49**
  - Human air traffic controller command

Boeing & Tupolew Collision



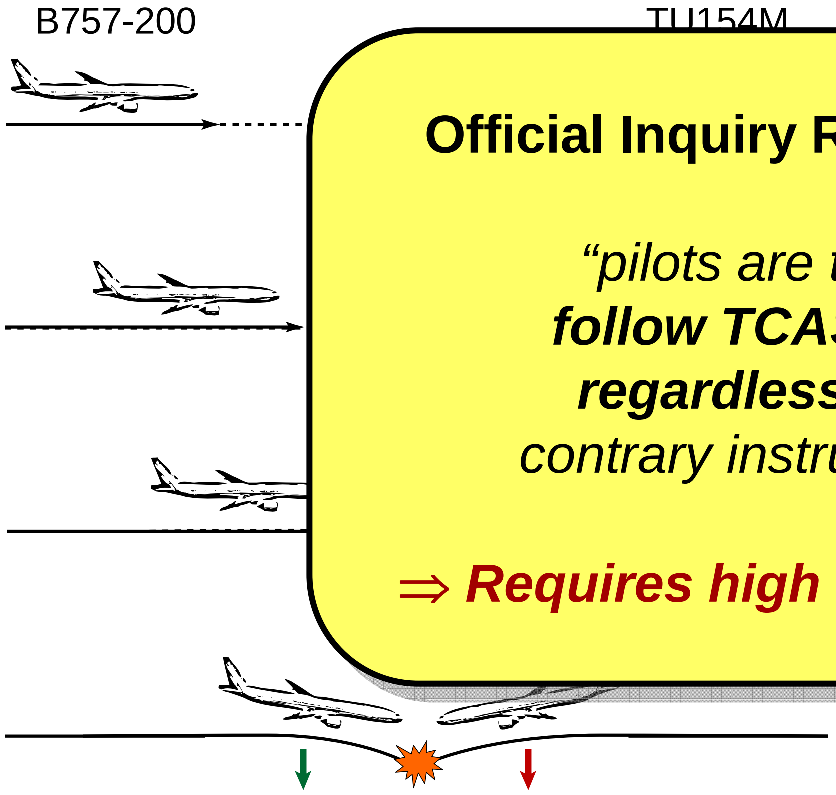
- **Überlingen, July 1, 2002**
- **21:33:03**
  - Alarm from Traffic Collision Avoidance System (TCAS)
- **21:34:49**
  - Human air traffic controller command
- **21:34:56**
  - TCAS recommendation

Boeing & Tupolew Collision



- **Überlingen, July 1, 2002**
- **21:33:03**
  - Alarm from Traffic Collision Avoidance System (TCAS)
- **21:34:49**
  - Human air traffic controller command
- **21:34:56**
  - TCAS recommendation
- **21:35:32**
  - Collision

Boeing & Tupolew Collision



- **Überlingen, July 1, 2002**

ic Collision
em (TCAS)

controller

ndation

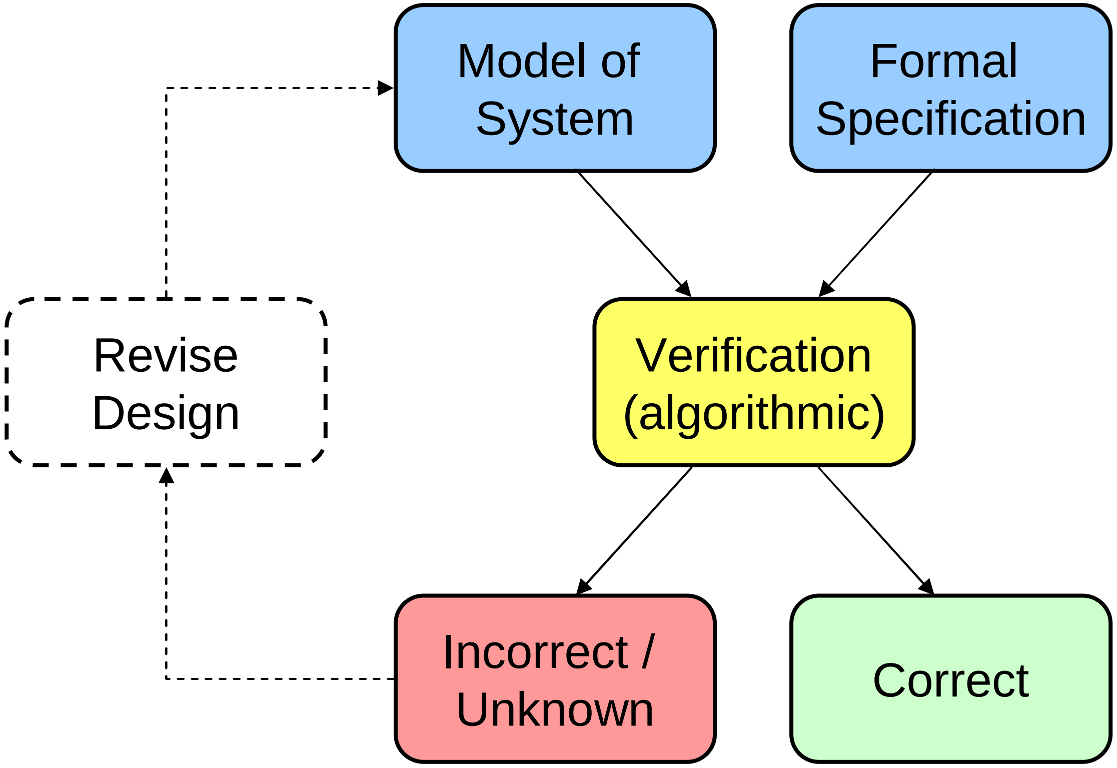
Official Inquiry Recommendation:

“pilots are to obey and ***follow TCAS*** *advisories,* ***regardless*** *of whether contrary instruction is given”*

⇒ ***Requires high confidence design***

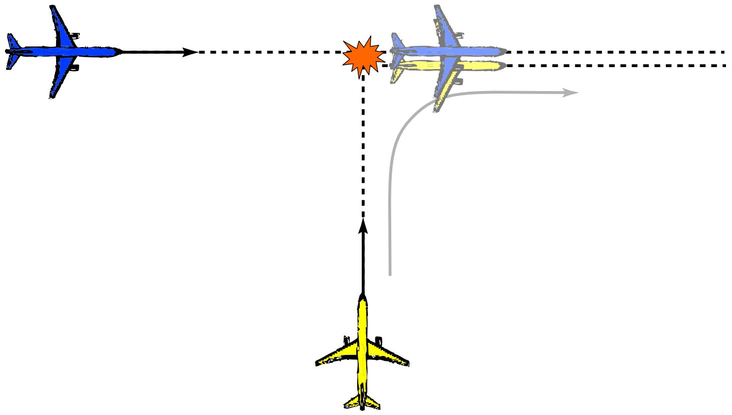
- **21:35:32**
  - Collision

Formal Verification



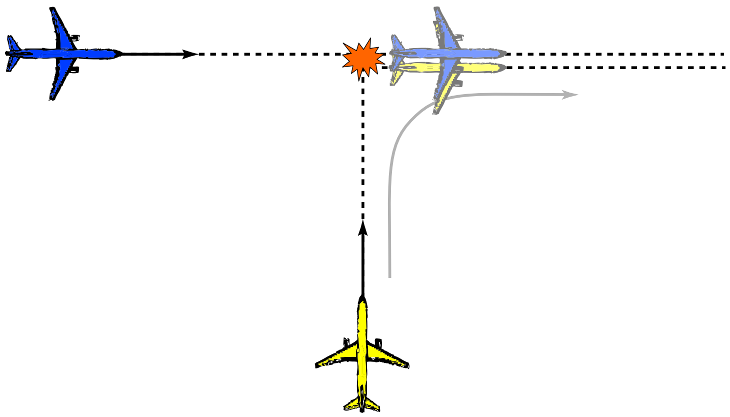
TCAS verified in part
[Livadas, Lygeros, Lynch, ’00]

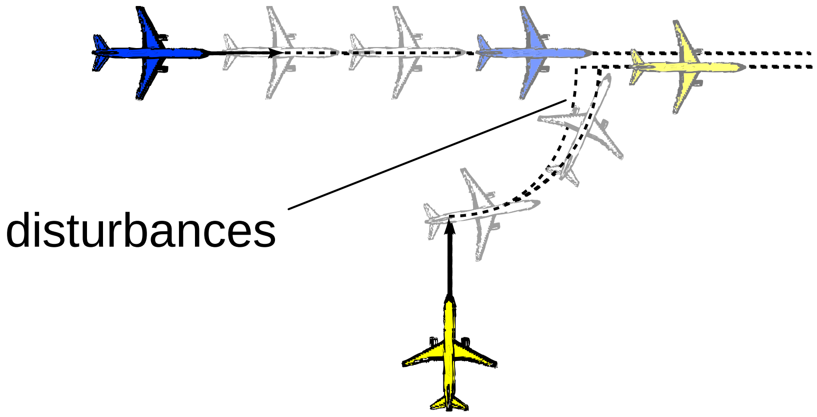
Join Maneuver [Tomlin et al.]



- **Traffic Coordination Problem**
  - join paths at different speed
- **Goals**
  - avoid collision
  - join with sufficient separation

Join Maneuver [Tomlin et al.]



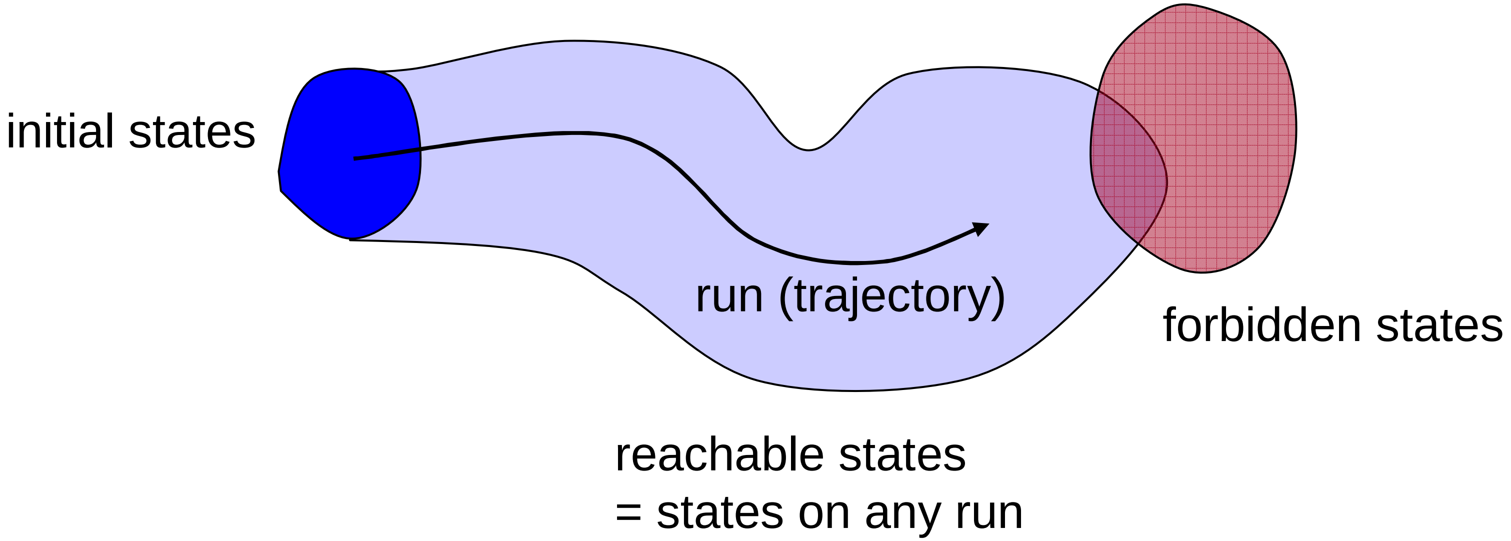


- **Traffic Coordination Problem**
  - join paths at different speed
- **Goals**
  - avoid collision
  - join with sufficient separation

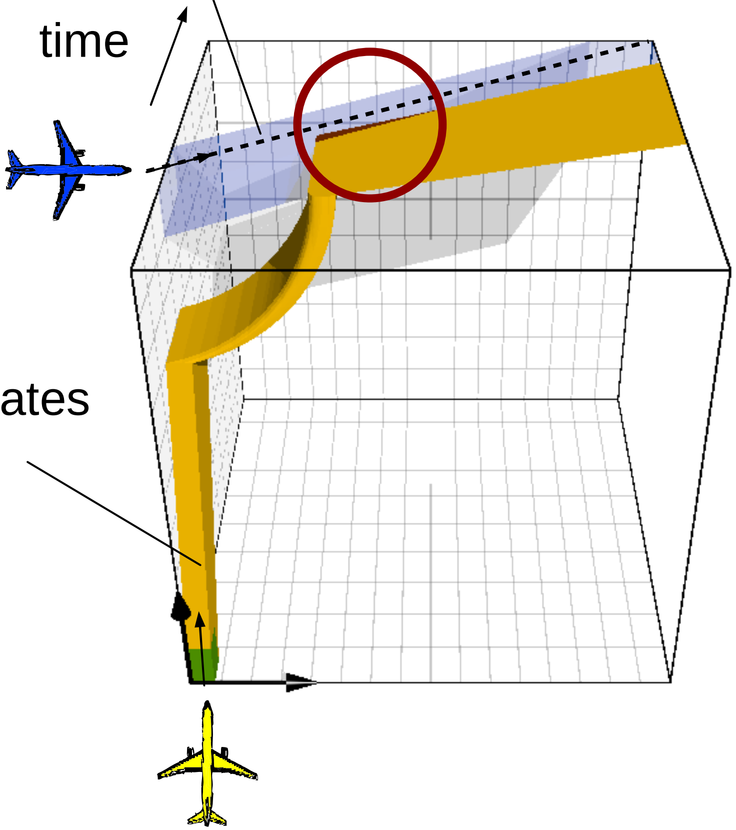
- **Models**
  - Environment: Planes
  - Software: Controller
    - switches fast/slow
- **Specification**
  - keep min. distance

Formal Verification

- **Characteristics**
  - mathematical rigor (sound proofs & algorithms)
  - exhaustive
- **In this talk: Reachability Analysis**



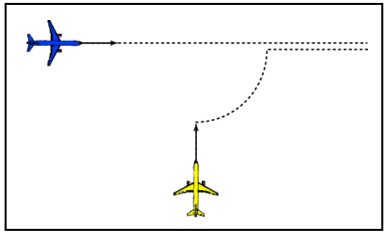
Join Maneuver [Tomlin et al.]



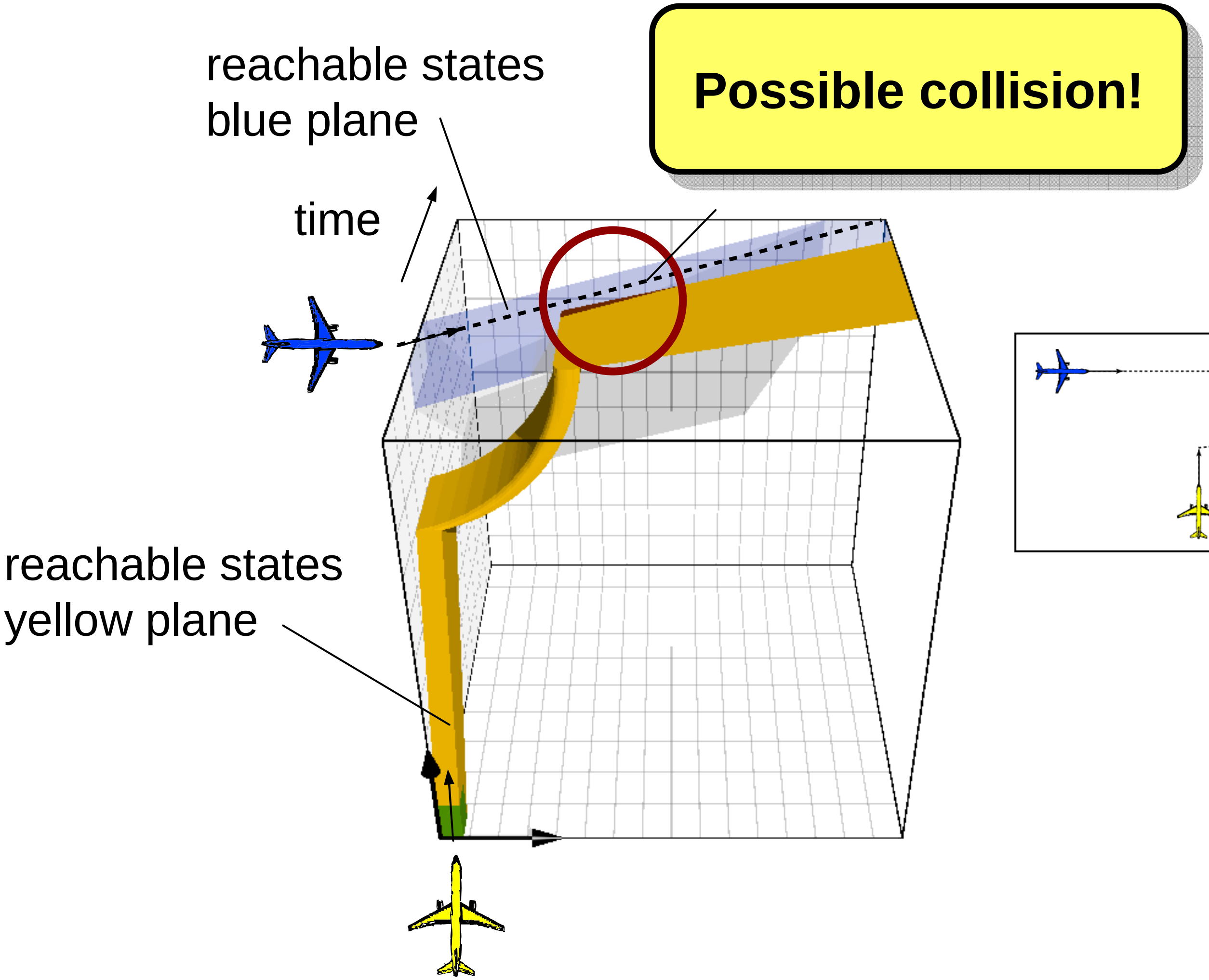
reachable states
blue plane

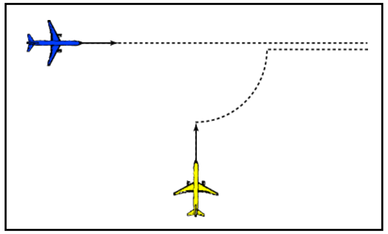
time

reachable states
yellow plane



Join Maneuver [Tomlin et al.]





Formal Verification

- **Key Problems**
  - computable (decidable) only for simple dynamics
  - computationally expensive
  - representation of / computation with continuous sets

Formal Verification

- **Fighting complexity with overapproximations**
  - simplify dynamics
  - set representations
  - set computations
- **Overapproximations should be**
  - conservative
  - easy to derive and compute with
  - accurate (not too many false positives)

Outline

I. **Hybrid Automata and Reachability**

II. **Reachability for Simple Dynamics**

   a) Linear Hybrid Automata

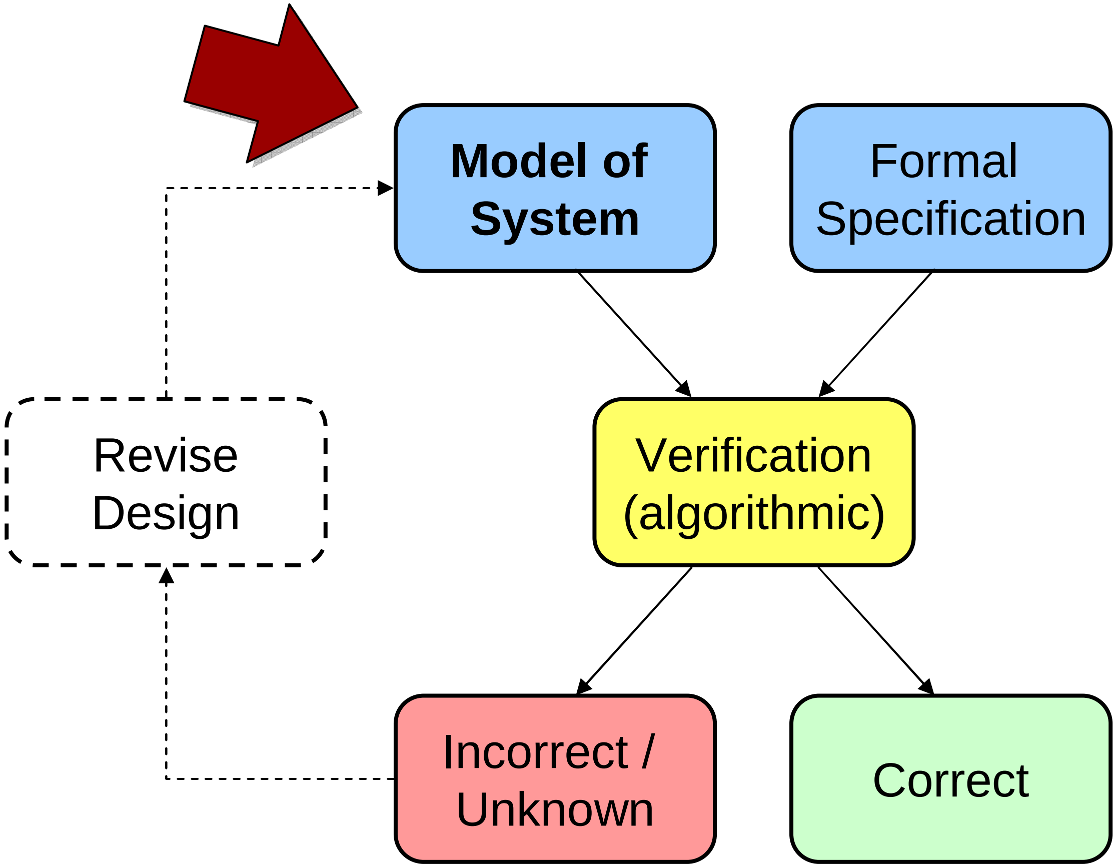
   b) Piecewise Affine Hybrid Systems

III. **Application to Complex Dynamics**

   a) Hybridization Techniques

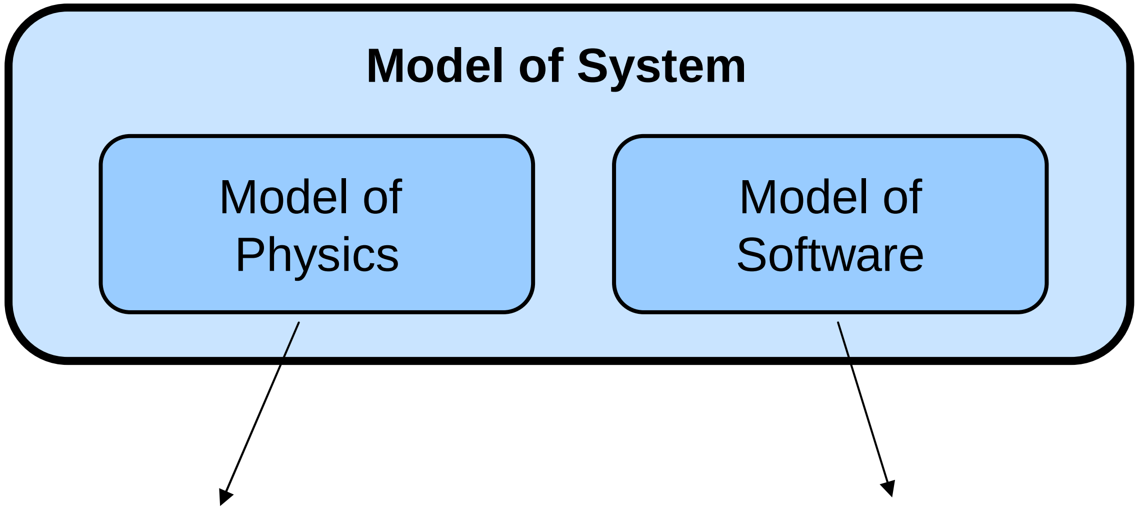
   b) Abstraction Refinement

Formal Verification



TCAS verified in part
[Livadas, Lygeros, Lynch, ’00]

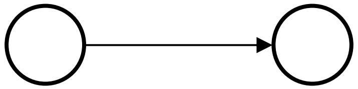
Formal Verification



continuous dynamics

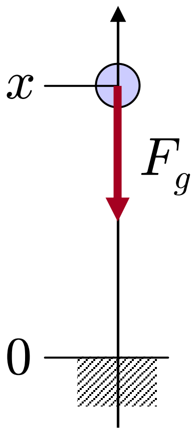
$$\dot{x} \quad = \quad f(x)$$

discrete dynamics



Modeling Hybrid Systems

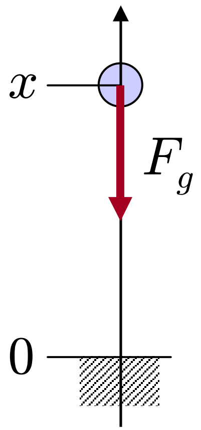
- **Example: Bouncing Ball**
  - ball with mass m and position x in free fall
  - bounces when it hits the ground at $x = 0$
  - initially at position x_o and at rest



)

Part I – Free Fall

- **Condition for Free Fall**
  - ball above ground: $x \geq 0$



- **First Principles (physical laws)**
  - gravitational force :

$$F_g = -mg$$

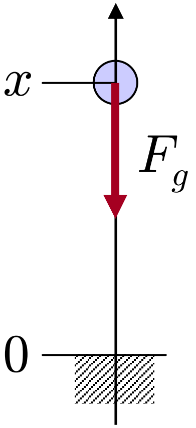
$$g = 9.81\mathrm{m/s}^2$$

  - Newton's law of motion :

$$m\ddot{x} = F_g$$

Part I – Free Fall

$$\begin{aligned} F_g &= -mg \\ m\ddot{x} &= F_g \end{aligned}$$



- **Obtaining 1st Order ODE System**
  - ordinary differential equation $\dot{x} = f(x)$
  - transform to 1st order by introducing variables for higher derivatives
  - here: $v = \dot{x}$:

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -g \end{aligned}$$

Part II – Bouncing

- **Conditions for “Bouncing”**
  - ball at ground position: $x = 0$
  - downward motion: $v < 0$
- **Action for “Bouncing”**
  - velocity changes direction
  - loss of velocity (deformation, friction)
  - $v := -cv$, $0 \leq c \leq 1$

Combining Part I and II

- **Free Fall**
  - while $x \geq 0$,

$$\left.\begin{array}{rcl} \dot{x} & = & v \\ \dot{v} & = & -g \end{array}\right\}$$

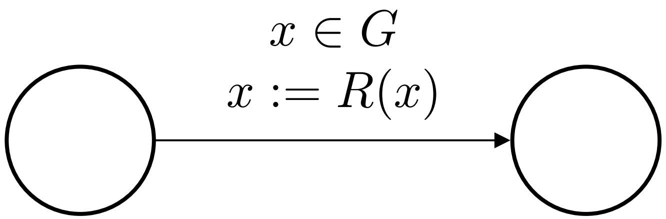
continuous dynamics

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

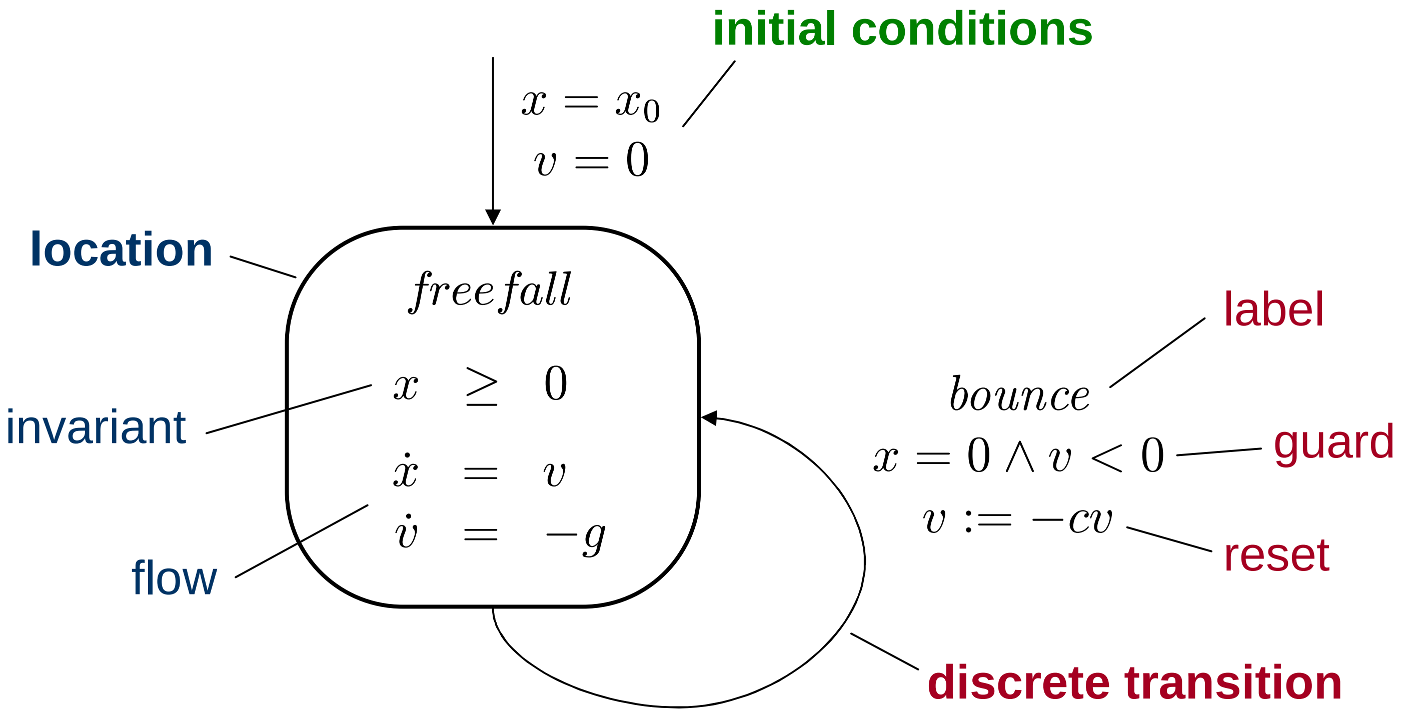
- **Bouncing**
  - if $x = 0$ and $v < 0$

$$\left.\begin{array}{rcl} v & := & -cv \end{array}\right\}$$

discrete dynamics



Hybrid Automaton Model



initial conditions

$x = x_0$

$v = 0$

location

$freefall$

invariant

$x \geq 0$

flow

$\dot{x} = v$

$\dot{v} = -g$

label

$bounce$

guard

$x = 0 \wedge v < 0$

reset

$v := -cv$

discrete transition

Hybrid Automata

$$H = (\mathit{Loc}, \mathit{Var}, \mathit{Ini}, \mathit{Inv}, \mathit{Trans}, \mathit{Lab}, \mathit{Flow})$$

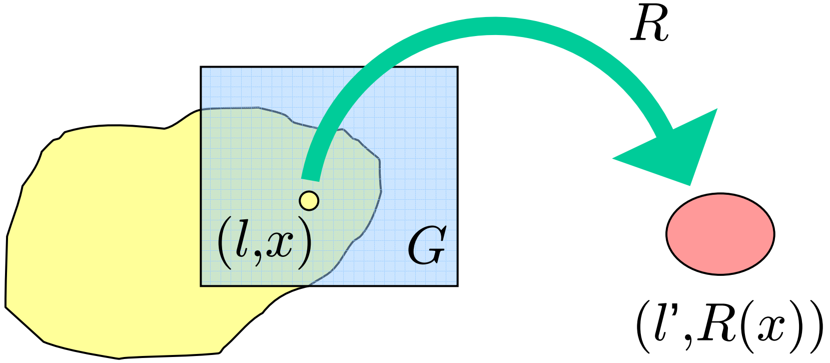
- **Defining Inhabited State Space:**
  - Locations Loc $\quad \{\mathit{freefall}\}$
  - Variables Var $\quad \{x, v\}$
    - Valuation: $x \in \mathbb{R}^{\mathit{Vars}}$ attributes a real value to each variable
    - State: $s = (l, x)$, with $l \in \mathit{Loc}$, $x \in \mathbb{R}^{\mathit{Vars}}$
  - Initial states $\mathit{Ini} \subseteq \mathit{Loc} \times \mathbb{R}^{\mathit{Vars}}$ $\quad \{(\mathit{freefall}, (x = x_0, v = 0))\}$
  - Invariant $\mathit{Inv} \subseteq \mathit{Loc} \times \mathbb{R}^{\mathit{Vars}}$ $\quad \{(\mathit{freefall}, (x \geq 0, v \in \mathbb{R}))\}$

Hybrid Automata – Discrete Dynamics

- **Defining Discrete Dynamics:** Trans

  $(l, \alpha, G, R, l') \in \mathit{Trans}$, with

  - label $\alpha \in \mathit{Lab}$,
  - guard $G \subseteq \mathbb{R}^{\mathit{Vars}}$,
  - reset $R : \mathbb{R}^{\mathit{Vars}} \rightarrow 2^{\mathbb{R}^{\mathit{Vars}}}$



- **Semantics: Discrete Transition**
  – can jump from (l,x) to (l', x') if $x \in G$ and $x' \in R(x)$

Hybrid Automata – Cont. Dynamics

- **Defining Continuous Dynamics:** Flow

$$\mathit{Flow} : \mathit{Loc} \times \mathbb{R}^{\mathrm{Vars}} \rightarrow 2^{\mathbb{R}^{\mathrm{Vars}}}$$

  - for each location l differential inclusion

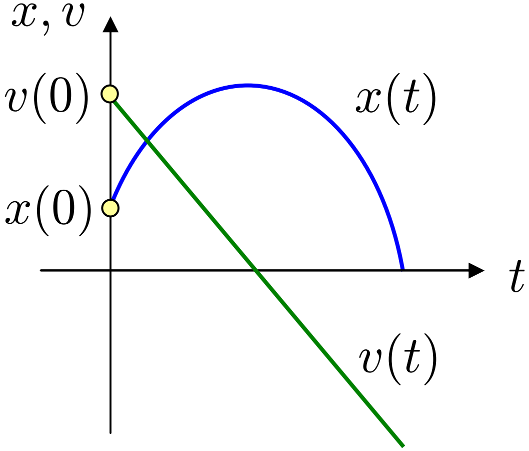
$$\dot{x} \in \mathit{Flow}(l, x)$$

- **Semantics: Time Elapse**
  - change state along $x(t)$ as time elapses
  - $x(t)$ must be in invariant Inv
  - $\dot{x}(t) \in \mathit{Flow}(l, x)$

Hybrid Automata – Cont. Dynamics

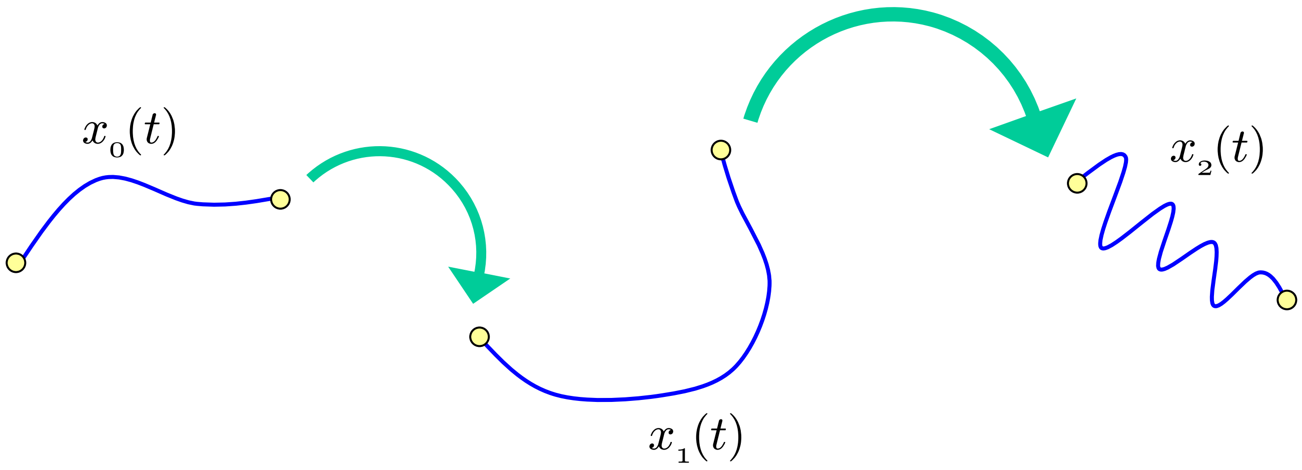
- **Bouncing Ball:**
  - Flow:

$$\begin{aligned}\dot{x} &= v\\ \dot{v} &= -g\end{aligned}$$

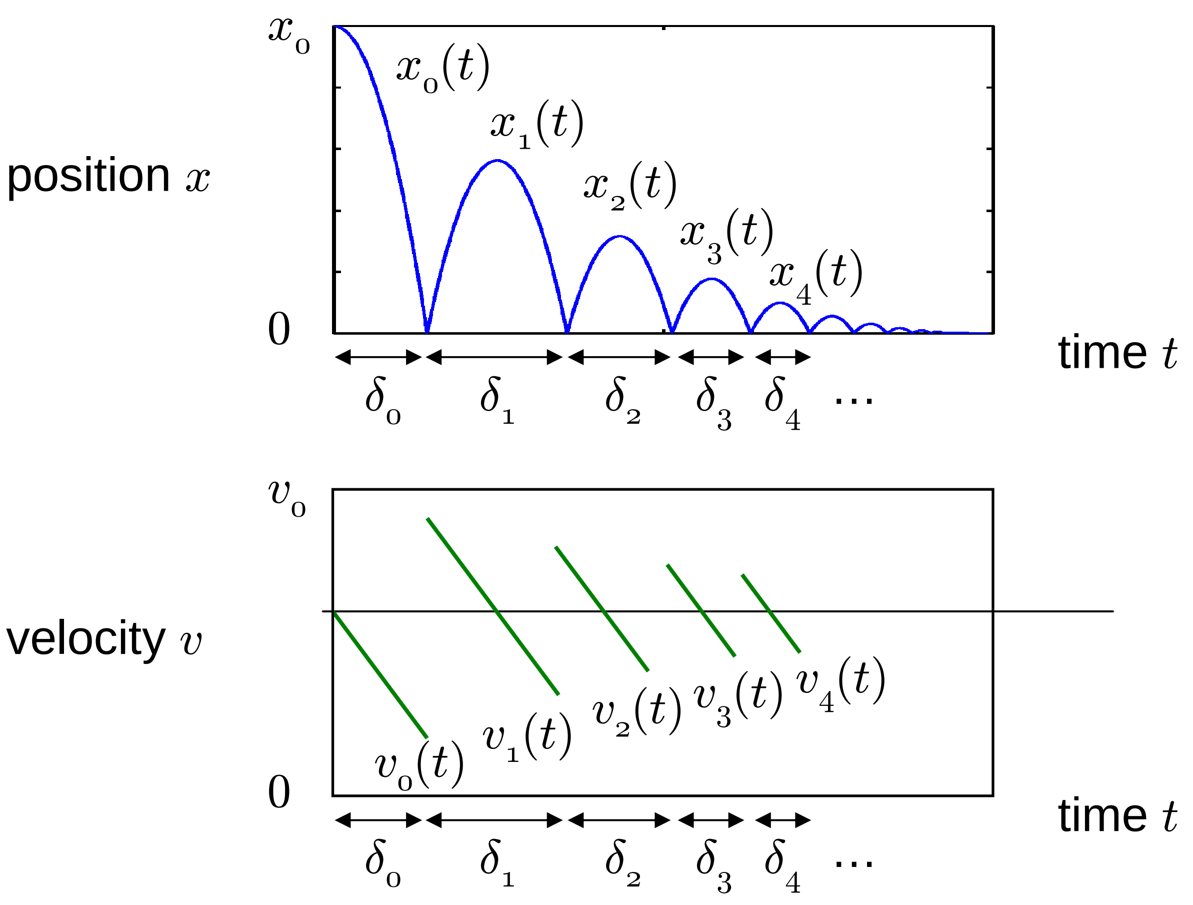


Hybrid Automata - Semantics

- **Run**
  - sequence of discrete transitions and time elapse
- **Execution**
  - run that starts in the initial states

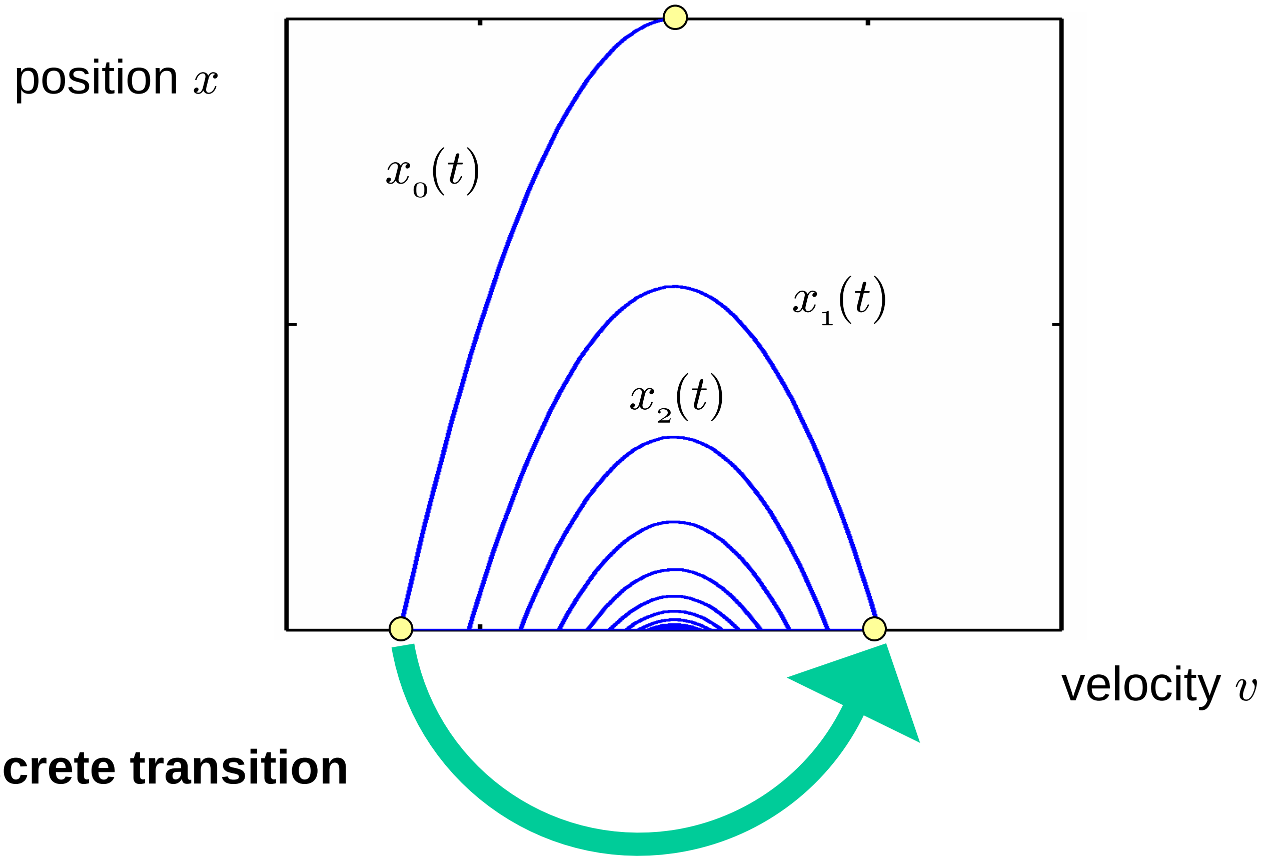


Execution of Bouncing Ball

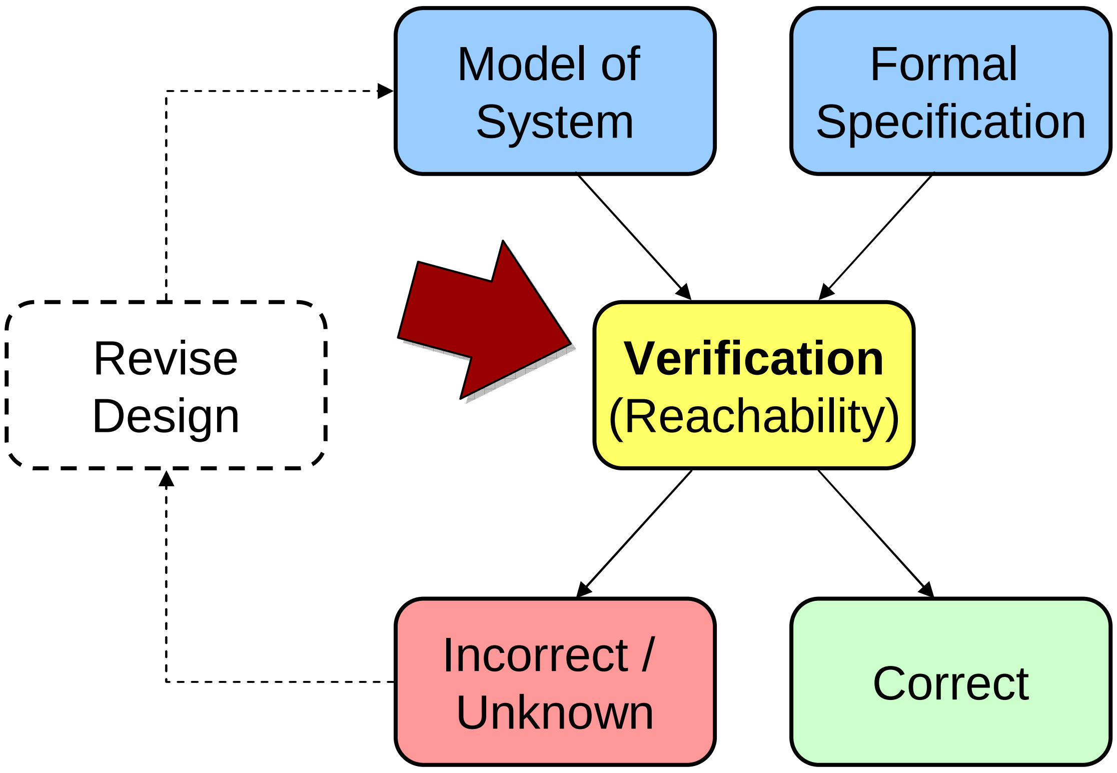


Execution of Bouncing Ball

- **State-Space View (infinite time range)**



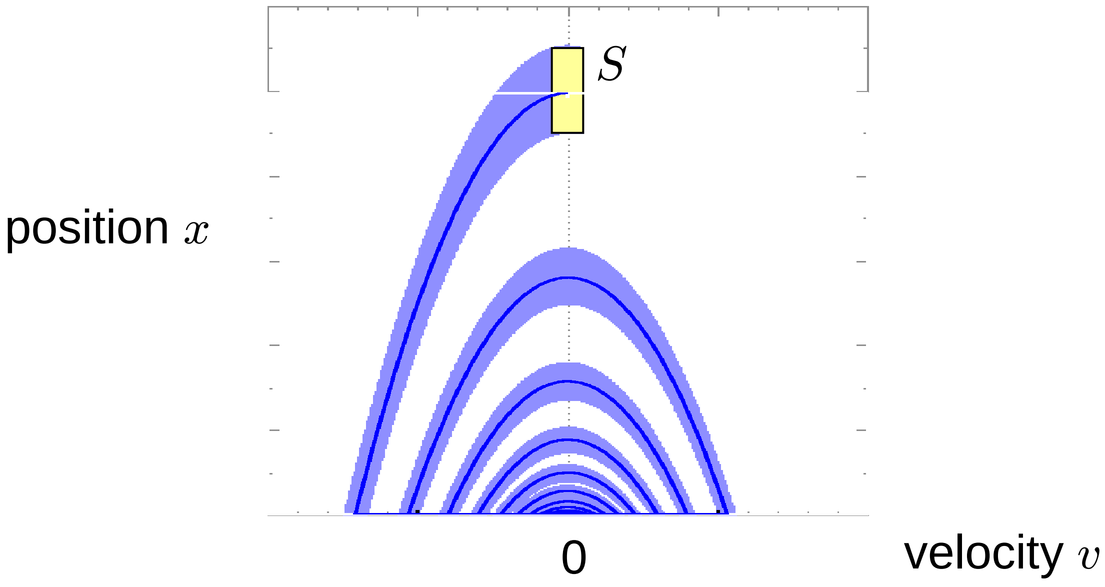
Formal Verification



TCAS verified in part
[Livadas, Lygeros, Lynch, ’00]

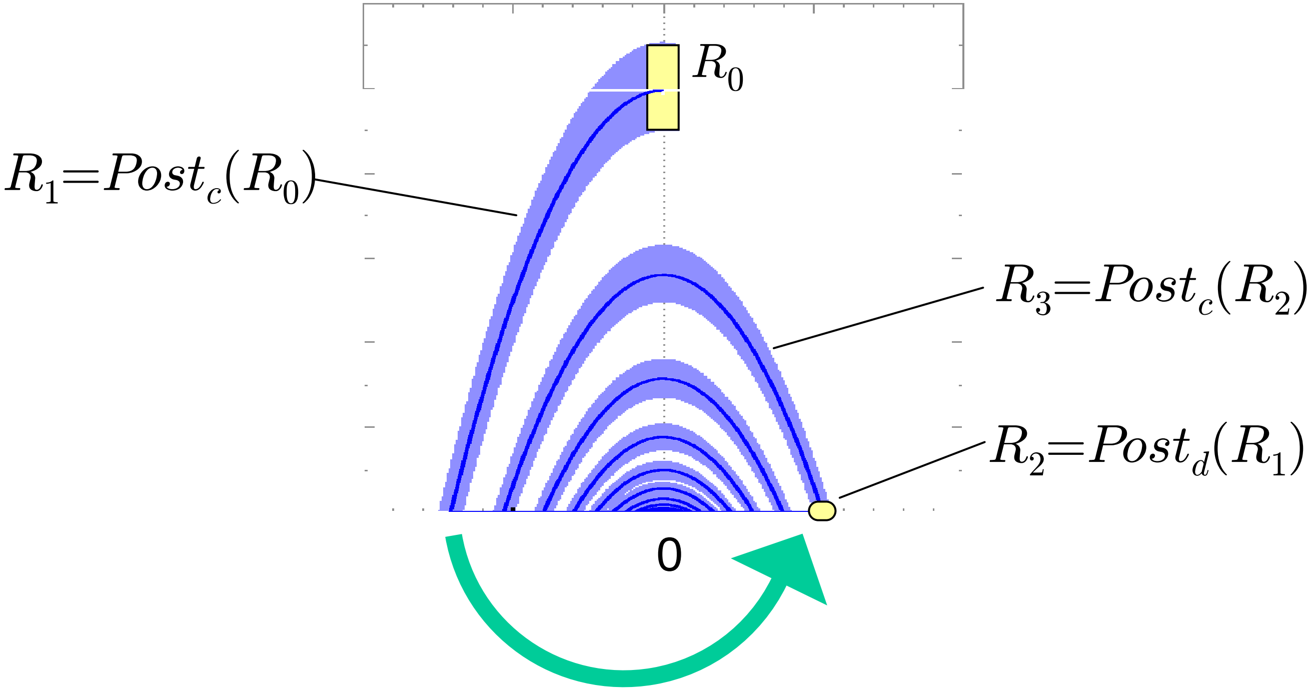
Computing Reachable States

- **Reachable states:** $Reach(S)$
  - any state encountered in a run starting in S



Computing Reachable States

- **Compute successor states**
  - discrete transitions : $Post_d(R)$
  - time elapse : $Post_c(R)$



Computing Reachable States

- **Fixpoint computation**
  - Initialization: $R_0 = Ini$
  - Recurrence: $R_{k+1} = R_k \cup Post_d(R_k) \cup Post_c(R_k)$
  - Termination: $R_{k+1} = R_k \Rightarrow Reach = R_k$.
- **Problems**
  - in general termination not guaranteed
  - time-elapse very hard to compute with sets

Chapter Summary

- **Why should we care?**
  - Reachability Analysis is a set-based computation that can answer many interesting questions about a system (safety, bounded liveness,…)
- **What's the problem?**
  - The hardest part is computing time elapse.
  - Explicit solutions only for very simple dynamics.
- **What's the solution?**
  - First study simple dynamics.
  - Then apply these techniques to complex dynamics.

Outline

I. Hybrid Automata and Reachability

II. Reachability for Simple Dynamics

   a) Linear Hybrid Automata

   b) Piecewise Affine Hybrid Systems

III. Application to Complex Dynamics

   a) Hybridization Techniques

   b) Abstraction Refinement

In this Chapter…

- **A very simple class of hybrid systems**
- **Exact computation of discrete transitions and time elapse**
  - Note: Reachability (and pretty much everything else) is nonetheless **undecidable**.
- **A case study**

Linear Hybrid Automata

- **Continuous Dynamics**
  - piecewise constant: $\dot{x} = 1$
  - intervals: $\dot{x} \in [1, 2]$
  - conservation laws: $\dot{x}_1 + \dot{x}_2 = 0$
  - general form: conjunctions of linear constraints

$$a \cdot \dot{x} \bowtie b, \qquad a \in \mathbb{Z}^n, b \in \mathbb{Z}, \bowtie \in \{<, \leq\}.$$

= convex polyhedron over derivatives

Linear Hybrid Automata

- **Discrete Dynamics**
  - affine transform: $x := ax + b$
  - with intervals: $x_2 := x_1 \pm 0.5$
  - general form: conjunctions of linear constraints (new value x')

$$a \cdot x + a' \cdot x' \bowtie b, \qquad a, a' \in \mathbb{Z}^n, b \in \mathbb{Z}, \bowtie \in \{<, \leq\}$$

= convex polyhedron over x and x’

Linear Hybrid Automata

- **Invariants, Initial States**
  - general form: conjunctions of linear constraints

$$a \cdot x \bowtie b, \qquad a \in \mathbb{Z}^n, b \in \mathbb{Z}, \bowtie \in \{<, \leq\},$$

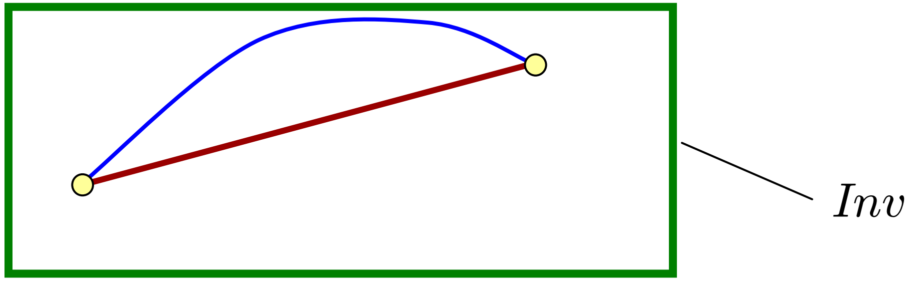
= convex polyhedron over x

Reachability with LHA

- **Compute discrete successor states** $Post_d(S)$
  - all x’ for which exists $x \in S$ s.t.
    - $x \in G$
    - x’ $\in R(x) \cap Inv$
- **Operations:**
  - existential quantification
  - intersection
  - standard operations on convex polyhedra

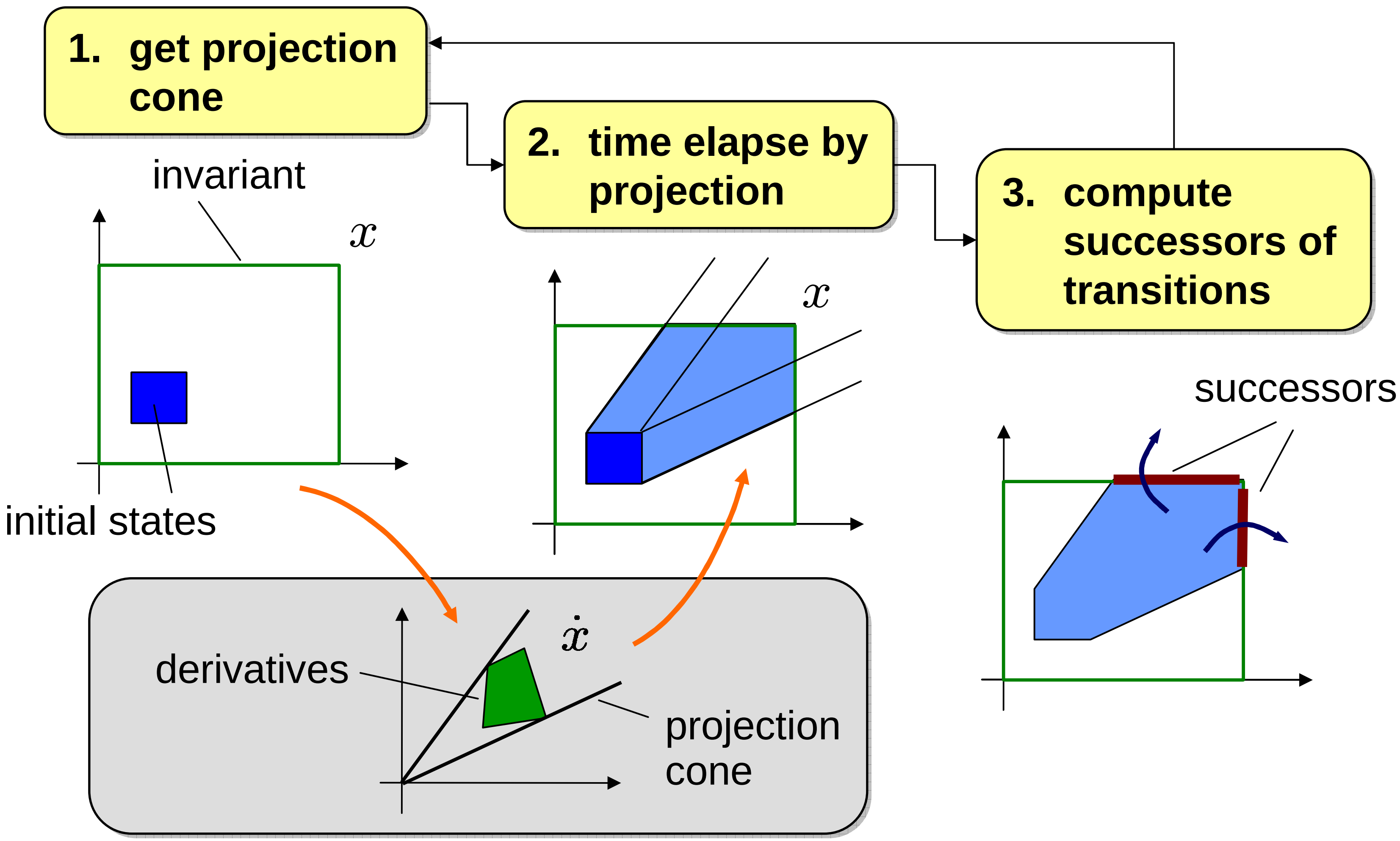
Reachability with LHA

- **Compute time elapse states** $Post_c(S)$
- **Theorem** [Alur et al.]
  - Time elapse along arbitrary trajectory iff time elapse along straight line (convex invariant).



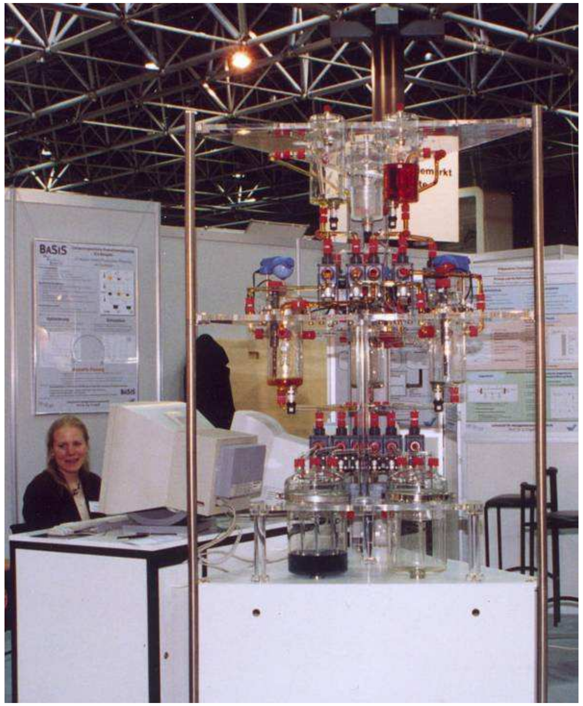
  - time elapse along straight line can be computed as projection along cone [Halbwachs et al.]

Reachability with LHA [Halbwachs, Henzinger, 93-97]

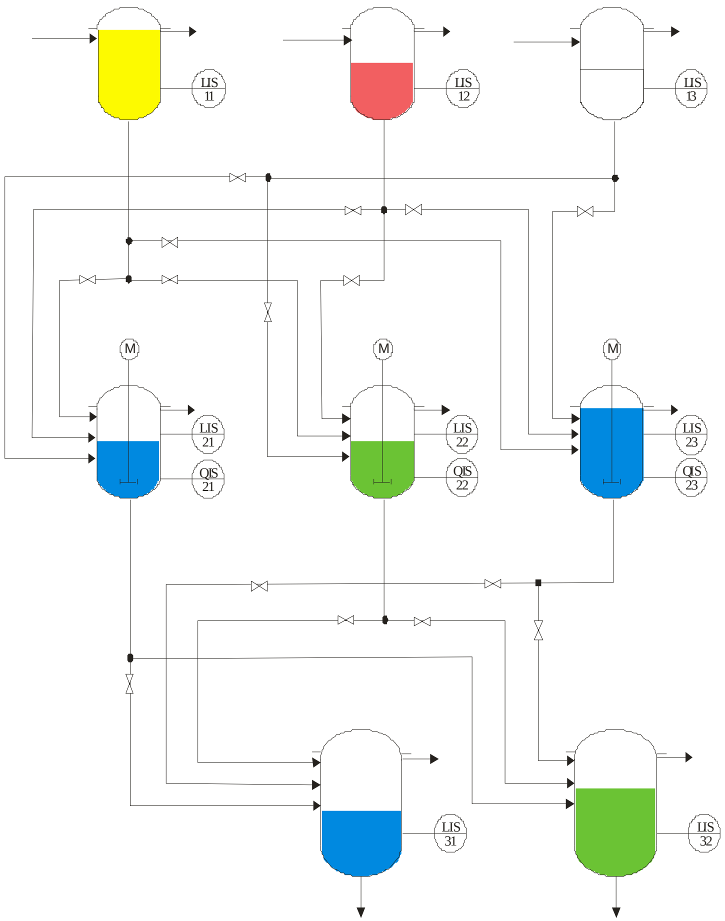


1. get projection cone
2. time elapse by projection
3. compute successors of transitions

Multi-Product Batch Plant



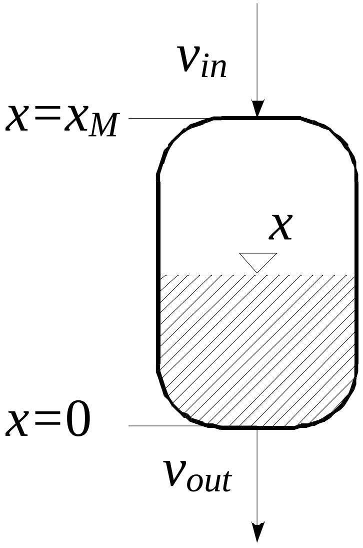
Multi-Product Batch Plant



- **Cascade mixing process**
  - 3 educts via 3 reactors
    ⇒ 2 products
- **Verification Goals**
  - Invariants
    - overflow
    - product tanks never empty
  - Filling sequence
- **Design of verified controller**

Switched Buffer Network

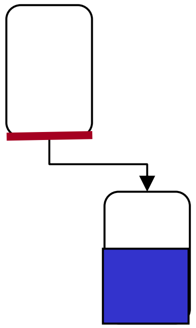
- **Buffers** $s_1, \ldots, s_n$
  - store material → continuous level $x_1, \ldots, x_n$
- **Channels**
  - transport material from buffer to buffer → continuous throughput $v(s, s')$, nondeterministic inside interval
- **Switching**
  - activate/deactivate channels discretely



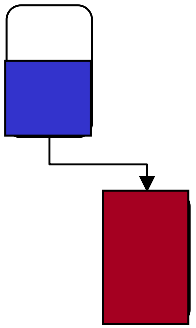
Buffer

Continuous Dynamics

- **Stationary throughput**
  - $v \in [a,b]$



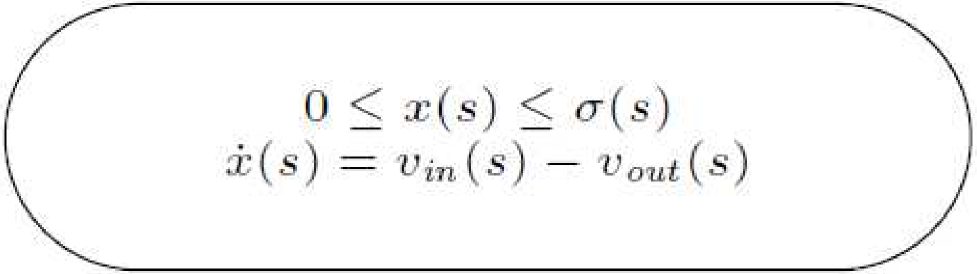
- **Source buffer empty**
  - throughput may seize, $v \in [0,b]$
  - **inflow of source = outflow of source**



- **Target buffer full**
  - throughput may seize, $v \in [0,b]$
  - **inflow of target = outflow of target**

Buffer Automaton Model

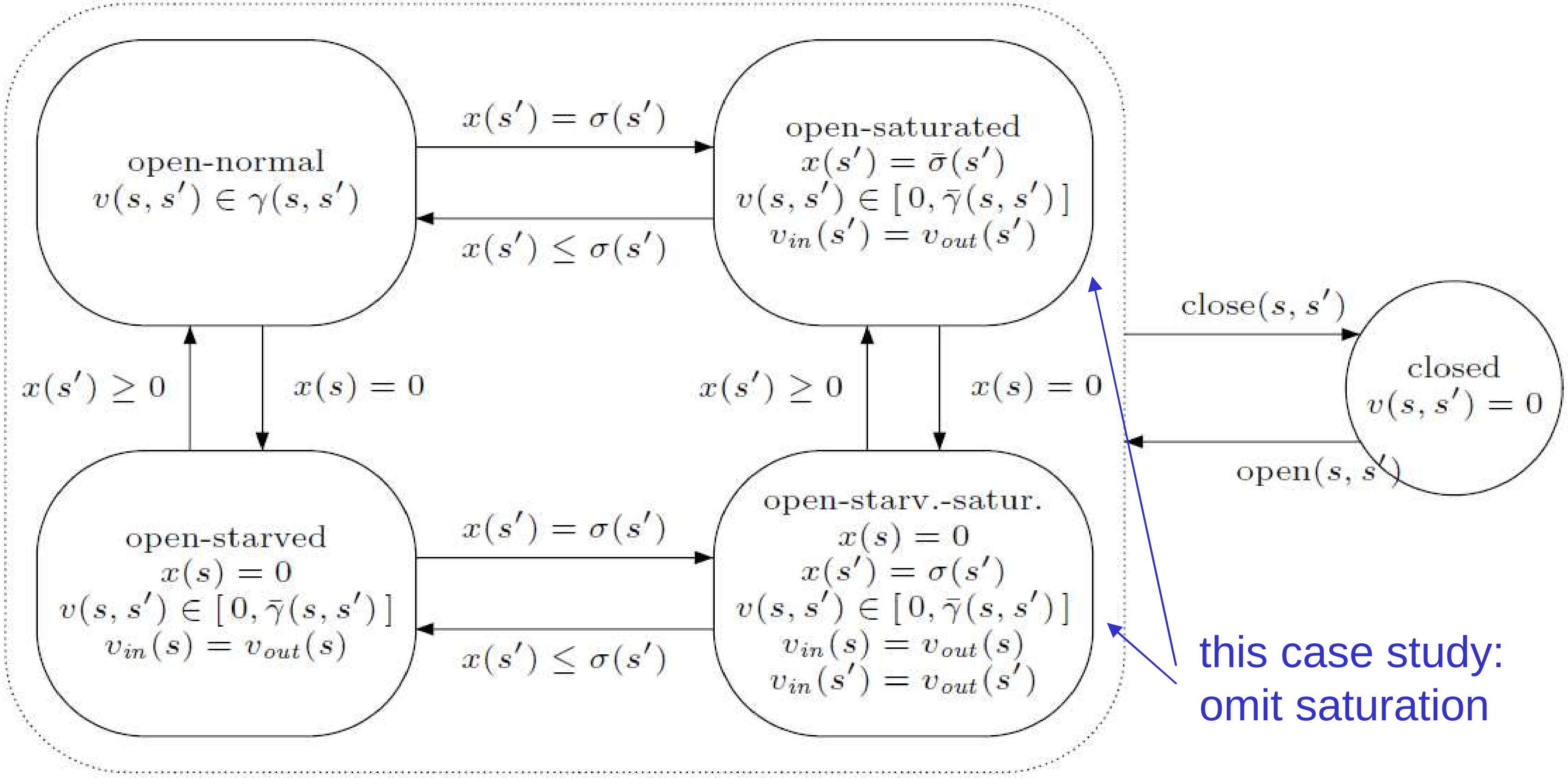
- tank levels = cont. variables x_i
- incoming flow $v_{in}(s)=\sum_{s'} v(s',s)$
- outgoing flow $v_{out}(s)=\sum_{s'} v(s,s')$



$$0 \leq x(s) \leq \sigma(s)$$
$$\dot{x}(s) = v_{in}(s) - v_{out}(s)$$

Channel Automaton Model

- throughput = algebraic variable (will be projected away)



this case study:
omit saturation

Production Schedule

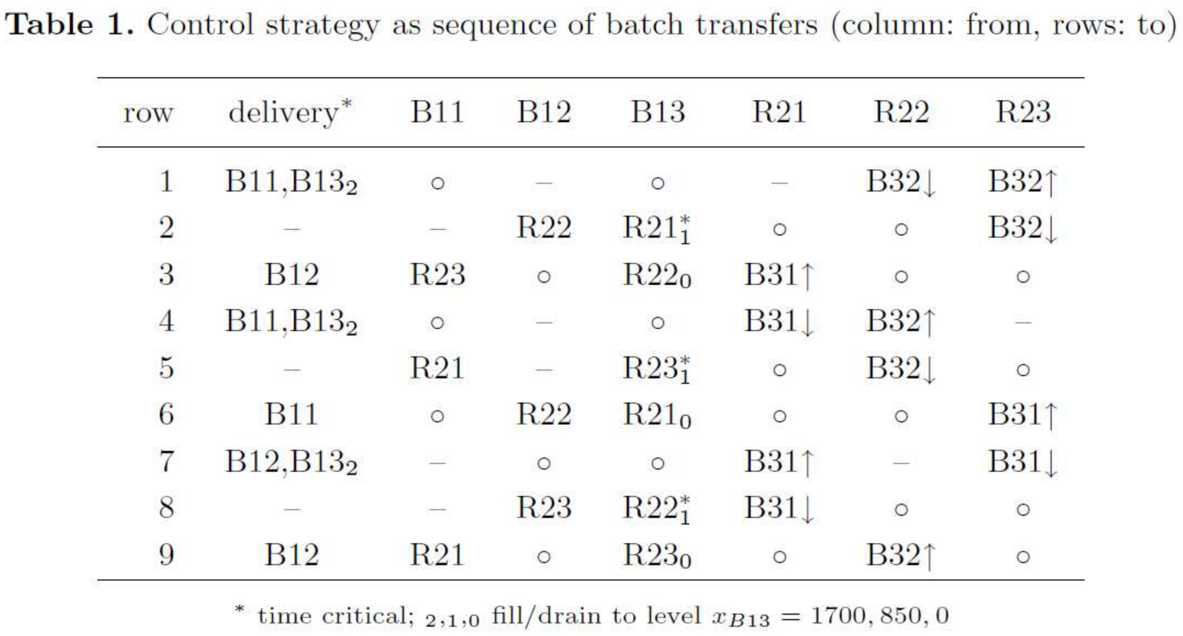


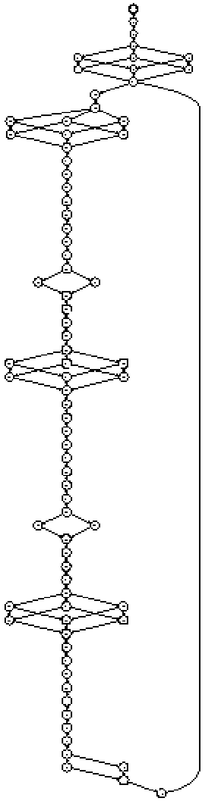
Table 1. Control strategy as sequence of batch transfers (column: from, rows: to)

row	delivery*	B11	B12	B13	R21	R22	R23
1	B11,B13$_2$	○	–	○	–	B32↓	B32↑
2	–	–	R22	R21$^*_1$	○	○	B32↓
3	B12	R23	○	R22$_0$	B31↑	○	○
4	B11,B13$_2$	○	–	○	B31↓	B32↑	–
5	–	R21	–	R23$^*_1$	○	B32↓	○
6	B11	○	R22	R21$_0$	○	○	B31↑
7	B12,B13$_2$	–	○	○	B31↑	–	B31↓
8	–	–	R23	R22$^*_1$	B31↓	○	○
9	B12	R21	○	R23$_0$	○	B32↑	○

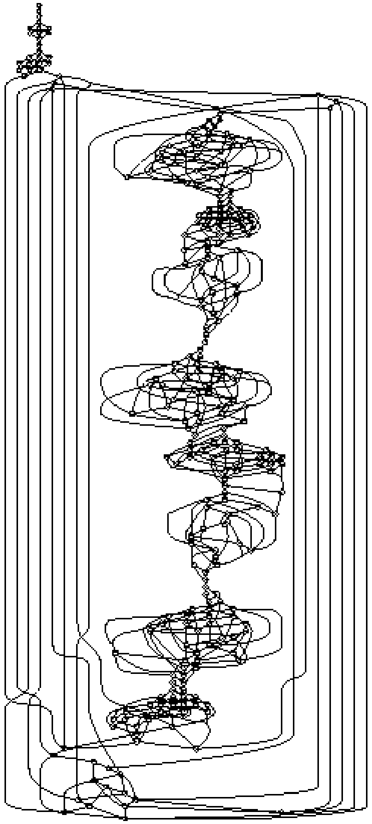
* time critical; $_{2,1,0}$ fill/drain to level $x_{B13} = 1700, 850, 0$

- uses 3 reactors in parallel
- transfers of batches from one tank to another
- formally a control strategy: locations × cont. variables → locations

Verification with PHAVer



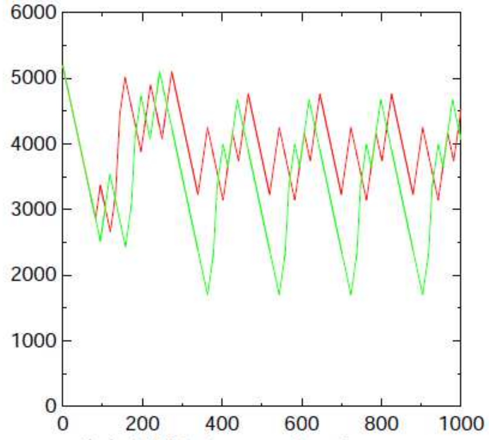
Controller



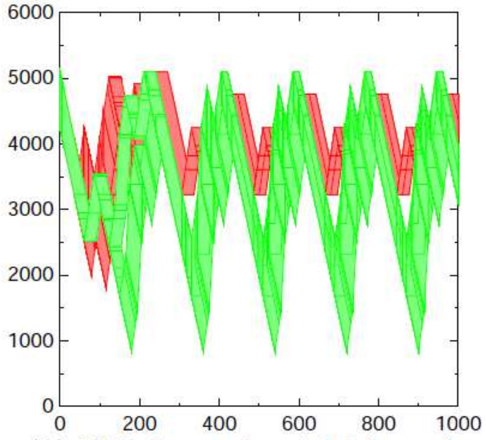
Controlled Plant

- **Controller automaton model**
  - 78 locations
  - ASAP transitions
- **Controller + Plant**
  - 266 locations, 823 transitions (~150 reachable)
- **Reachability over infinite time**
  - 120s—1243s, 260—600MB
  - computation cost increases with nondeterminism (intervals for throughputs, initial states)

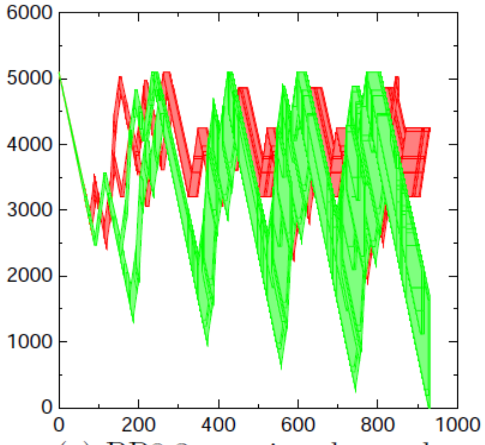
Verification with PHAVer



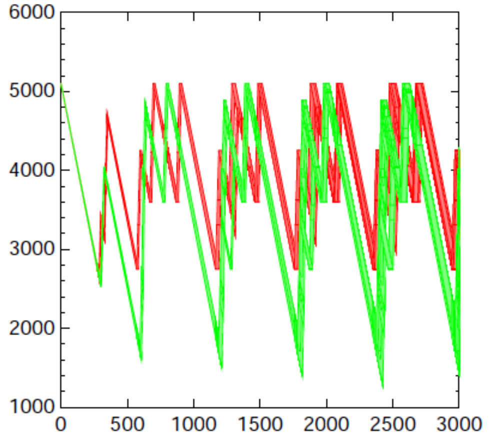
(a) BP8.1: nominal case



(b) BP8.2: varying initial cond.



(c) BP8.3: varying demand



(d) BP8.4: varying but slow demand

					Automaton		Reachable Set	
Instance	Time [s]	Mem. [MB]	Depth[a]	Checks[b]	Loc.	Trans.	Loc.	Poly.
BP8.1	120	267	173	279	266	823	130	279
BP8.2	139	267	173	422	266	823	131	450
BP8.3	845	622	302	2669	266	823	143	2737
BP8.4	1243	622	1071	4727	266	823	147	4772

* on Xeon 3.20 GHz, 4GB RAM running Linux; [a] lower bound on depth in breadth-first search; [b] number of applications of post-operator

Outline

I. Hybrid Automata and Reachability

II. Reachability for Simple Dynamics

  a) Linear Hybrid Automata

  b) Piecewise Affine Hybrid Systems

III. Application to Complex Dynamics

  a) Hybridization Techniques

  b) Abstraction Refinement

In this Chapter…

- **Another class of (not quite so) simple dynamics**
  - but things are getting serious (no explicit solution for sets)
- **Exact Computation time elapse only at discrete points in time**
  - used to overapproximate continuous time
- **Efficient data structures**

Piecewise Affine Hybrid Systems

- **Affine dynamics**
  - Flow:

    $\dot{x} = Ax + b$ (deterministic)

    $\dot{x} \in Ax + B$, with B a set (nondeterministic)

  - For time elapse it’s enough to look at a single location.

Linear Dynamics

- **Let’s begin with “autonomous” part of the dynamics:**

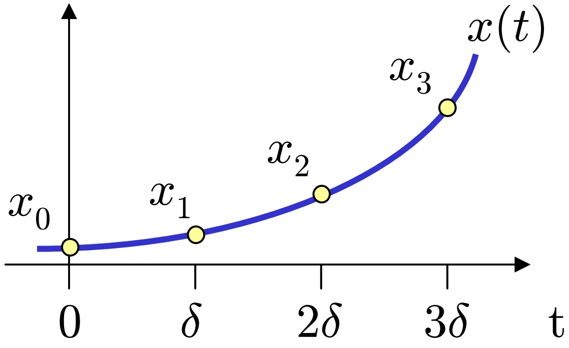
$$\dot{x} = Ax, \quad x \in \mathbb{R}^n$$

- **Known solutions:**
  - analytic solution in continuous time
  - explicit solution at discrete points in time (up to arbitrary accuracy)
- **Approach for Reachability:**
  - Compute reachable states over finite time: $Reach_{[0,\mathrm{T}]}(X_{Ini})$
  - Use time-discretization, but with care!

Time-Discretization for an Initial Point

- **Analytic solution:** $x(t) \quad = \quad e^{At} x_{Ini}$
  - with $t = \delta k$:

$$x(\delta(k+1)) \quad = \quad e^{A\delta} x(\delta k)$$



- **Explicit solution in discretized time (recursive):**

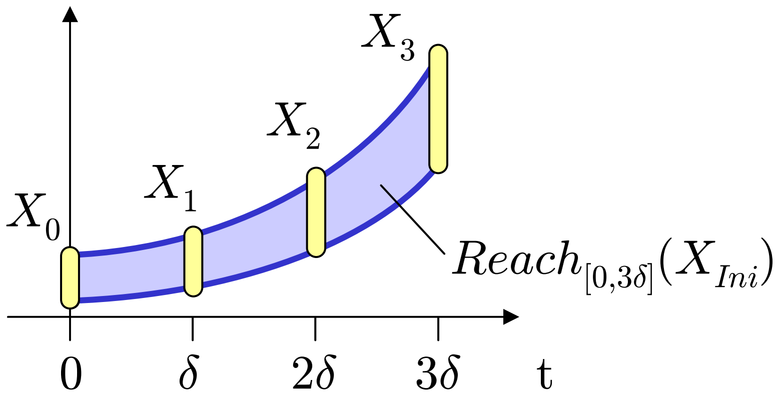
$$\begin{aligned} x_0 &= x_{Ini} \\ x_{k+1} &= e^{A\delta} x_k \end{aligned}$$

multiplication with const. matrix $e^{A\delta}$
= linear transform

Time-Discretization for an Initial Set

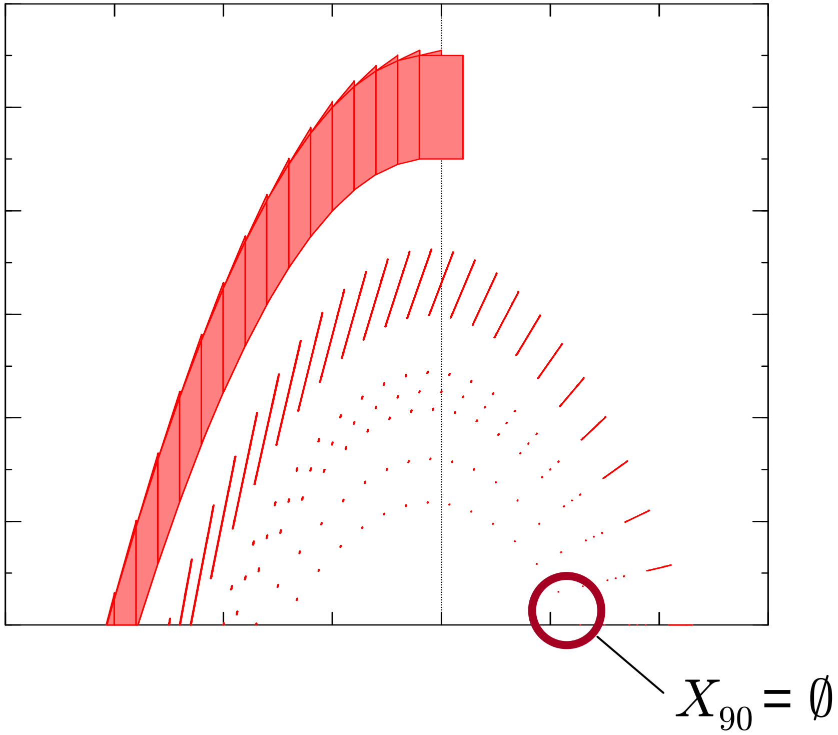
- **Explicit solution in discretized time**

$$\begin{aligned} X_0 &= X_{Ini} \\ X_{k+1} &= e^{A\delta} X_k \end{aligned}$$



- **Acceptable solution for purely continuous systems**
  - $x(t)$ is in $\epsilon(\delta)$-neighborhood of some X_k
- **Unacceptable for hybrid systems**
  - discrete transitions might “fire” between sampling times
  - if transitions are “missed,” $x(t)$ not in $\epsilon(\delta)$-neighborhood

Bouncing Ball



– In other examples this error might not be as obvious…

Reachability by Time-Discretization

- **Goal:**
  - Compute sequence Ω_k over bounded time $[0,N\delta]$ such that:

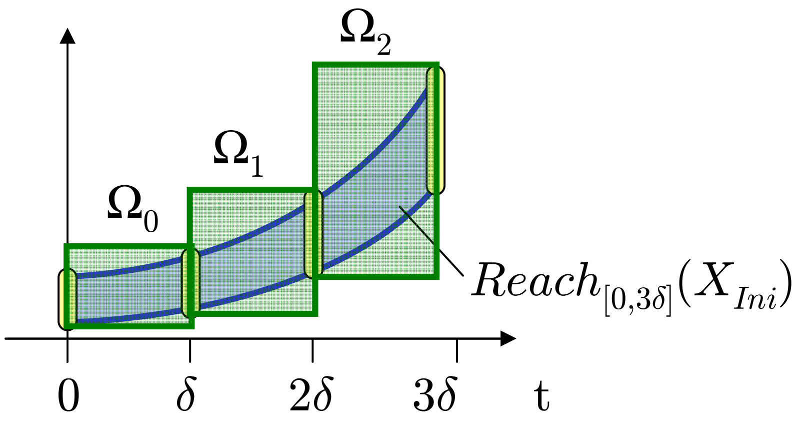
$$\mathrm{Reach}_{[0,N\delta]}(X_{Ini}) \subseteq \Omega_0 \cup \Omega_1 \cup \ldots \cup \Omega_N$$

- **Approach:**
  - Refine Ω_k by recurrence:

$$\Omega_{k+1} \quad = \quad e^{A\delta}\Omega_k$$

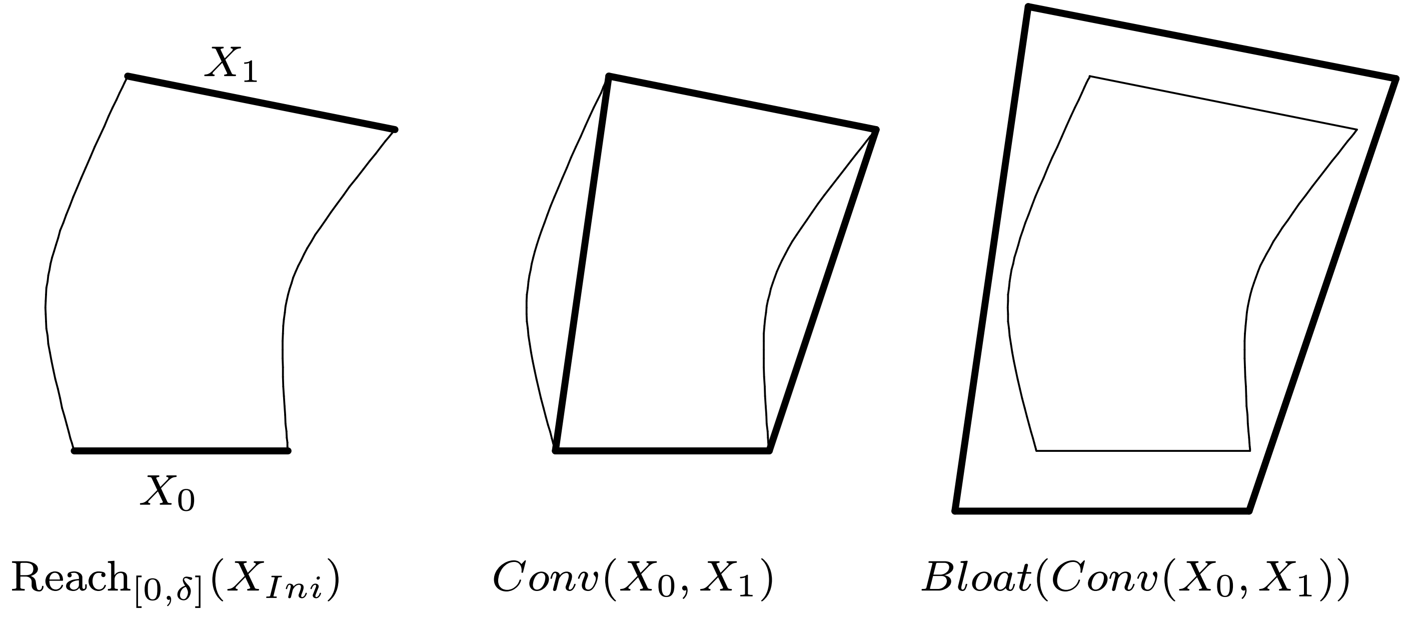
  - Condition for Ω_o:

$$\mathrm{Reach}_{[0,\delta]}(X_{Ini}) \subseteq \Omega_0$$



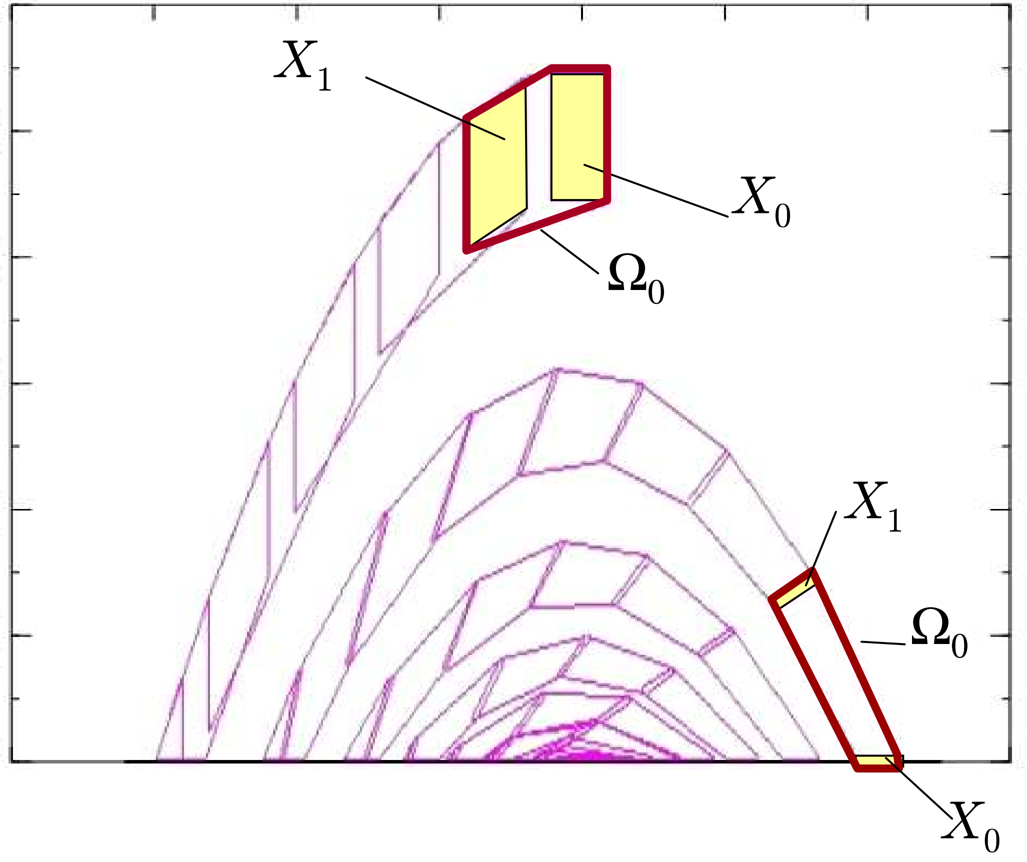
Time-Discretization with Convex Hull

- **Overapproximating** $Reach_{[0,\delta]}$**:**



Time-Discretization with Convex Hull

- **Bouncing Ball:**



Nondeterministic Affine Dynamics

- **Let's include the effect of inputs:**

$$\dot{x} = Ax + Bu, \quad x \in \mathbb{R}^n, u \in U \subseteq \mathbb{R}^p$$

  - variables $x_1,\ldots,x_n$, inputs $u_1,\ldots,u_p$

- **Input u models nondeterminism**

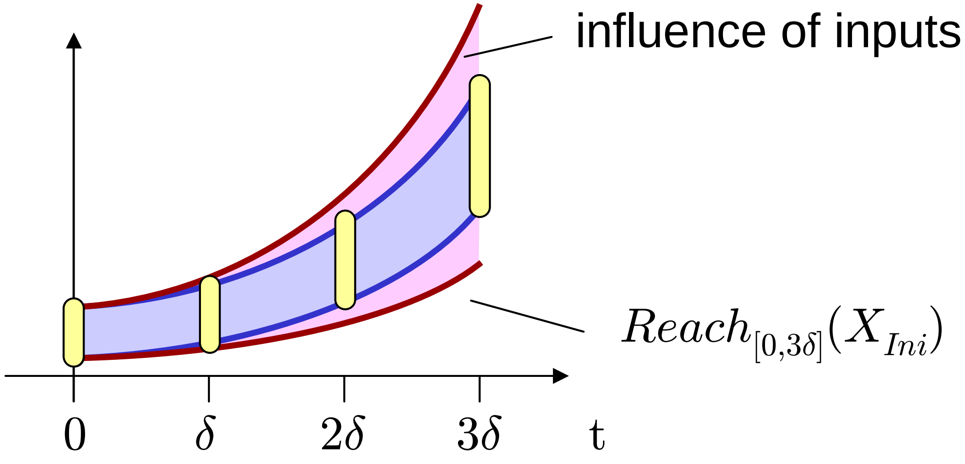
$$\dot{x} \in Ax + BU$$

  - used later for overapproximating nonlinear dynamics

Nondeterministic Affine Dynamics

- **Analytic Solution**

$$x(t) = \underbrace{e^{A\delta}x(0)}_{\text{autonomous dynamics}} + \underbrace{\int_0^{\tau} e^{A(\delta-\tau)}Bu(\tau)d\tau}_{\text{influence of inputs}}$$

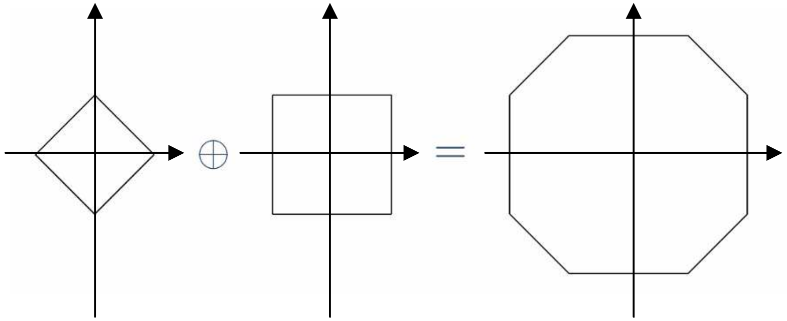


Nondeterministic Affine Dynamics

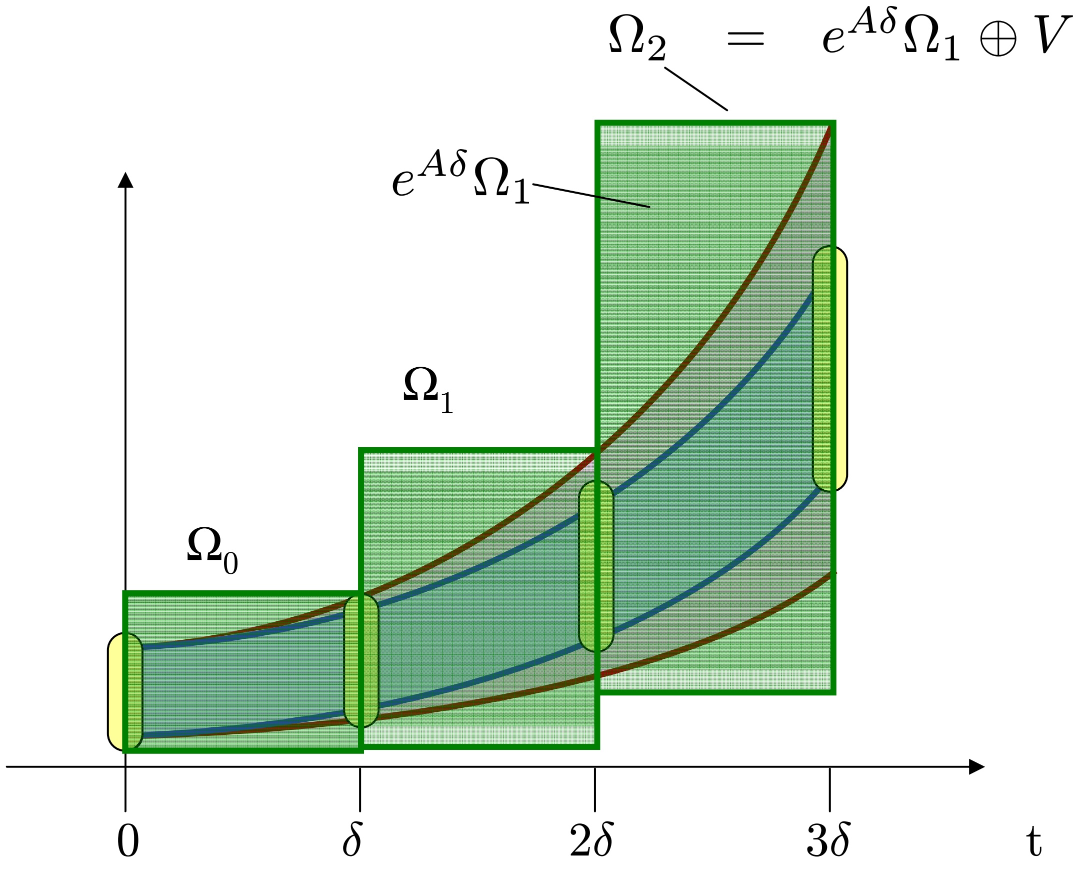
- **How far can the input "push" the system in δ time?**
  - $V =$ box with radius $\frac{e^{||A||\delta}-1}{||A||}\sup_{u\in U}||Bu||$

$$\begin{array}{lcl} \Omega_0 & = & Bloat(Conv(X_{Ini}, e^{A\delta}X_{Ini})) \oplus V \\ \Omega_{k+1} & = & e^{A\delta}\Omega_k \oplus V \end{array}$$

- **Minkowski Sum:** $A \oplus B = \{a+b \mid a \in A,\ b \in B\}$



Nondeterministic Affine Dynamics



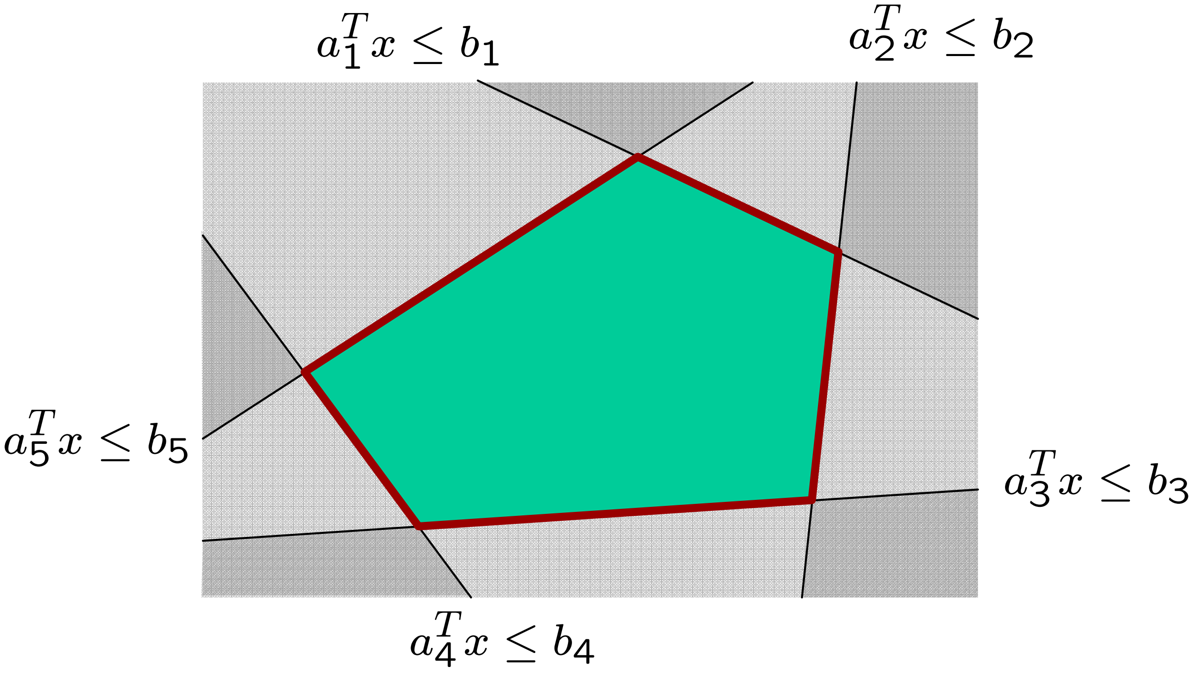
Implementing Reachability

- **Find representation for continuous sets with**
  - linear transformation ($\Omega_{\kappa+1} = \Phi\ \Omega_{\kappa}$)
  - Minkowski Sum
  - intersection (with guards)

Polyhedra

- **Finite conjunction of linear constraints**

$$P = \{x \mid Ax \leq b\}\,.$$



Operations on Polyhedra

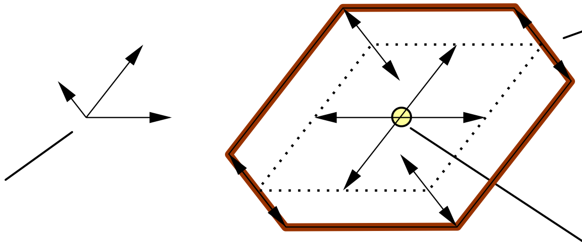
- **Linear Transformation**
  - transform matrix
  - $O(n^3)$
- **Minkowski Sum**
  - need to compute vertices
  - **O(exp(n))**
- **Intersection**
  - join lists of constraints
  - O(1)

Zonotopes

- **Central symmetric polyhedron**

$$Z = (c, \langle v_1, \ldots, v_m \rangle) = \left\{ c + \sum_{i=1}^{m} \alpha_i v_i \mid \alpha_i \in [-1, 1] \right\}.$$

center generators



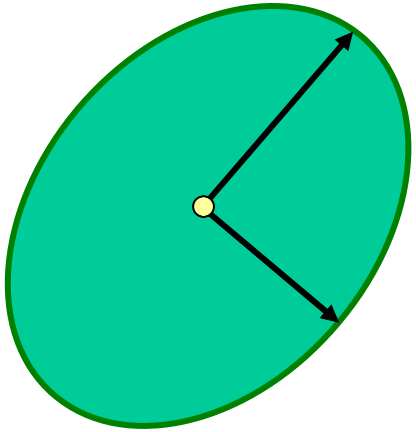
Operations on Zonotopes

- **Linear Transformation**
  - transform generators $\Phi Z = (\Phi c, \langle \Phi v_1, \ldots, \Phi v_m \rangle)$
  - **O(n^2)**
- **Minkowski Sum**
  - join lists of generators $Z \oplus Z' = (c + c', \langle v_1, \ldots, v_m, v'_1, \ldots, v'_{m'} \rangle)$
  - O(1)
- **Intersection**
  - Problem: intersection of zonotopes is not a zonotope
  - overapproximate

Ellipsoids

- **Quadratic form**
  - matrix or generator representation

$$E = \left\{ x \mid x^T Q x + A x \leq b \right\}.$$



Operations on Ellipsoids

- **Linear Transformation**
  - transform generators
  - **$O(n^2)$**
- **Minkowski Sum**
  - Problem: result is not an ellipsoid
  - overapproximate
- **Intersection**
  - Problem: intersection of ellipsoids is not an ellipsoid
  - overapproximate

Implementing Reachability

- **Complexity of 1 Step of Time Elapse:**
  - Polyhedra: O(exp(n))
  - Zonotopes: O(n^2) ✓

- **Problem: With each iteration, Ω_i get more complex**

$$\Omega_{k+1} \quad = \quad e^{A\delta}\Omega_k \oplus V$$

  - Minkowski sum increases number of
    - Polyhedra: constraints
    - Zonotopes: generators

Wrapping Effect

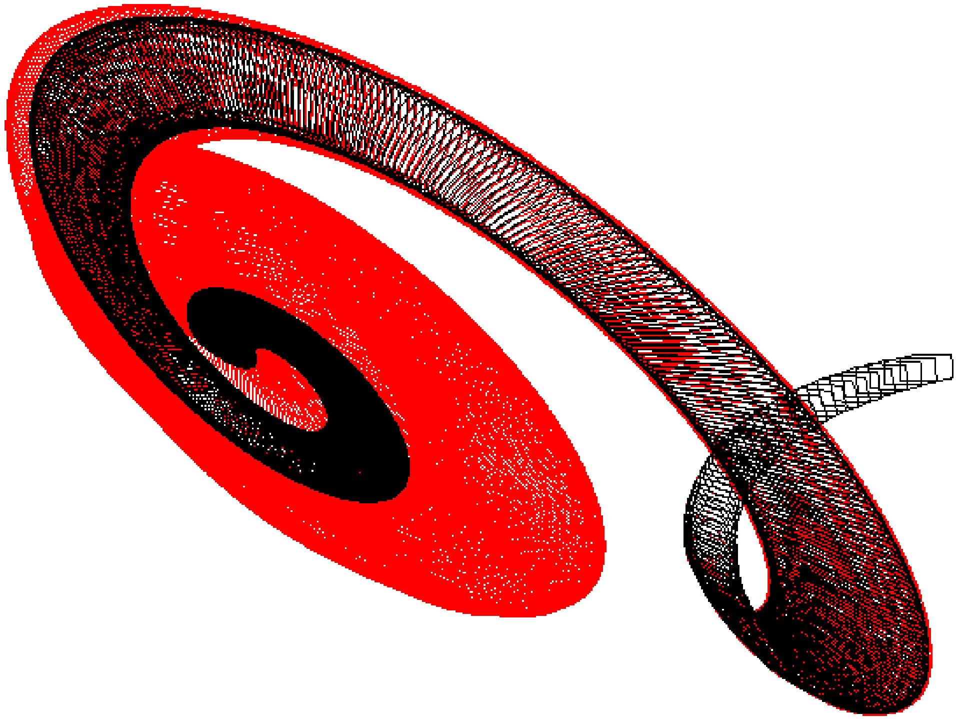
- **Fight complexity by overapproximation**
- **Overapproximated Sequence**

$$\hat{\Omega}_{k+1} \quad = \quad Approx(e^{A\delta}\hat{\Omega}_k \oplus V)$$

  - accumulation of approximations $\rightarrow$ Wrapping Effect
  - exponential increase in approximation error!

Wrapping Effect

- **Exact vs. overapproximation**
  - dimension 5 for 600 time steps
  - overapproximation with 100 generators



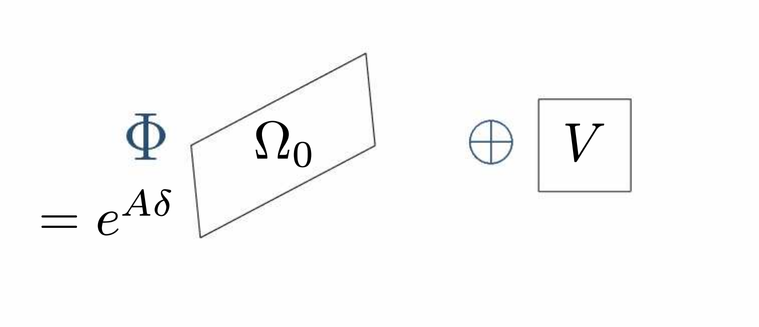
Wrapping Effect

$$\hat{\Omega}_{k+1} \quad = \quad Approx(e^{A\delta}\hat{\Omega}_k \oplus V)$$

- **How does error accumulate?**
  - linear transformation (scaling error up $\rightarrow$ exp)
  - adding V is added (adding some more error)

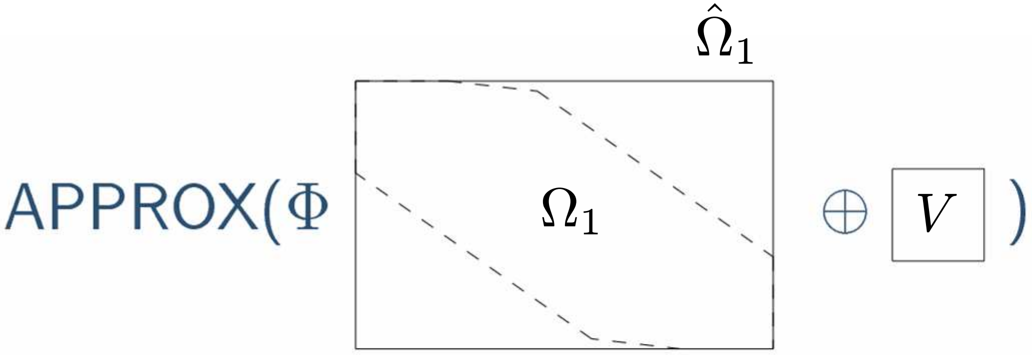
Wrapping Effect

$$\hat{\Omega}_{k+1} \quad = \quad Approx(e^{A\delta}\hat{\Omega}_k \oplus V)$$



Wrapping Effect

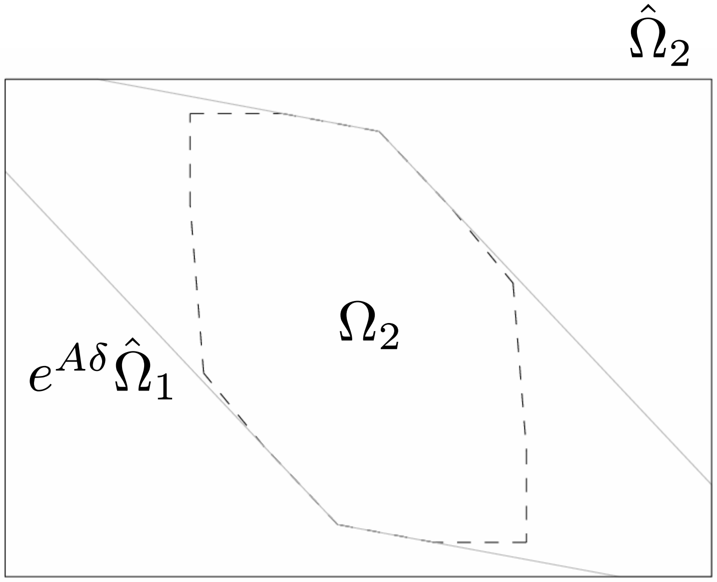
$$\hat{\Omega}_{k+1} \quad = \quad Approx(e^{A\delta}\hat{\Omega}_k \oplus V)$$



Wrapping Effect

$$\hat{\Omega}_{k+1} \quad = \quad Approx(e^{A\delta}\hat{\Omega}_k \oplus V)$$

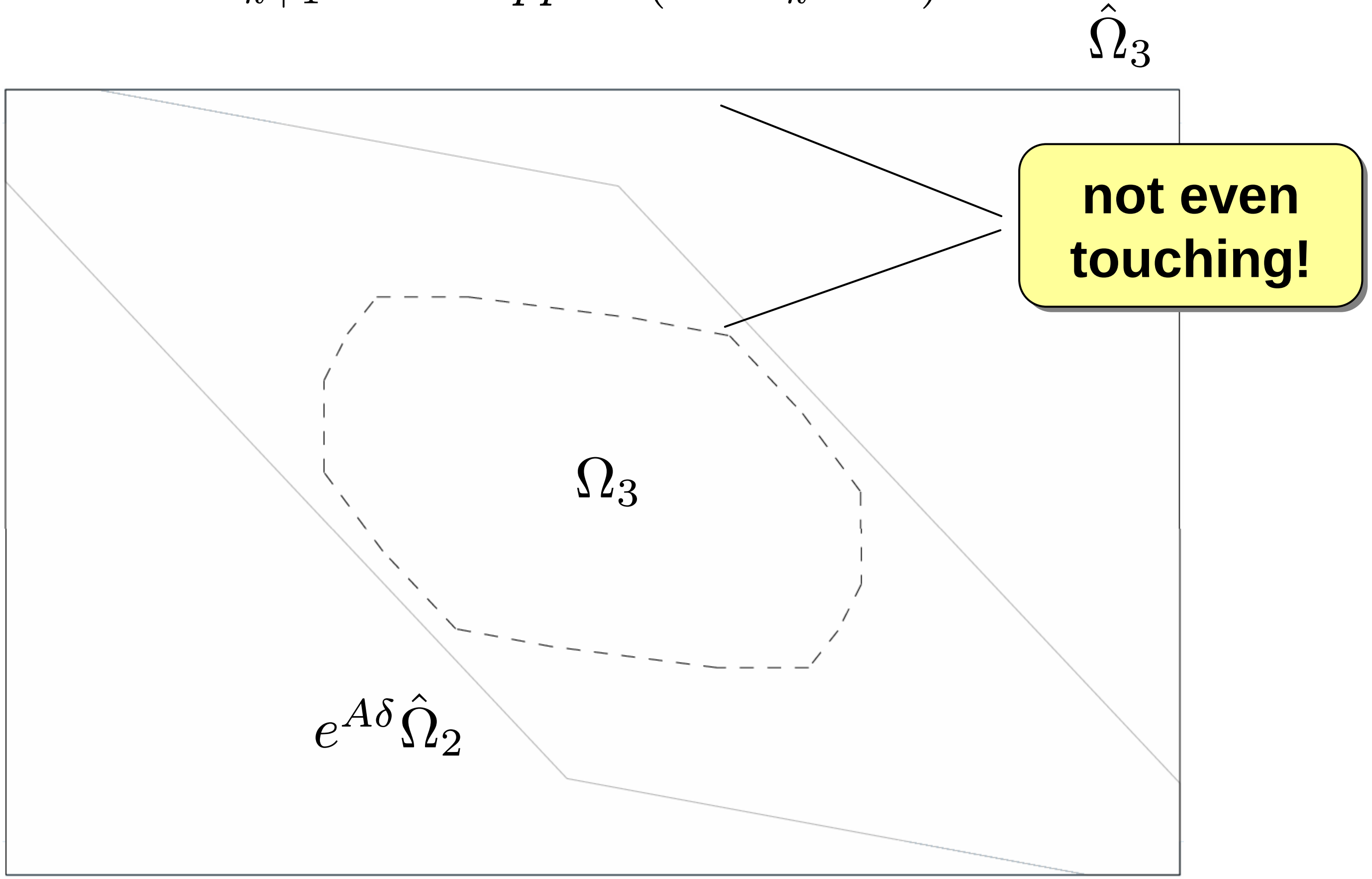
APPROX(Φ



$\oplus \boxed{V}$)

Wrapping Effect

$$\hat{\Omega}_{k+1} \quad = \quad Approx(e^{A\delta}\hat{\Omega}_k \oplus V)$$



Fighting the Wrapping Effect

- **Separate transformations and Minkowski sums:**

$$\Omega_{k+1} \quad = \quad \underbrace{e^{(k+1)\delta A}\Omega_0}_{R_{i+1}} \oplus \underbrace{\underbrace{e^{k\delta A}V}_{V_i} \oplus \underbrace{\left(e^{(k-1)\delta A}V \oplus \cdots \oplus V\right)}_{S_i}}_{S_{i+1}}.$$

- **4 Sequences:**

$$\begin{aligned} R_{i+1} &= e^{\delta A}R_i, \\ V_{i+1} &= e^{\delta A}V_i, \\ S_{i+1} &= S_i \oplus V_i, \\ \Omega_{i+1} &= R_{i+1} \oplus S_{i+1} \end{aligned} \qquad R_0 = \Omega_0,\ V_0 = V,\ S_0 = \{0\}$$

4-Sequence Algorithm

$$\begin{array}{rcl} R_{k+1} & = & e^{\delta A} R_k, \\ V_{k+1} & = & e^{\delta A} V_k, \\ S_{k+1} & = & S_k \oplus V_k, \\ \Omega_{k+1} & = & R_{k+1} \oplus S_{k+1} \end{array}$$

- **Only transformations in R_k and V_k**
  - complexity independent of k
  - no overapproximation necessary
- **Only Minkowski sum in S_k and Ω_k**
  - growing number of generators, but no longer transformed
  - $O(Nn^3)$ instead of $O(N^2n^3)$

4-Sequence Algorithm

$$\begin{array}{rcl} R_{k+1} & = & e^{\delta A} R_k, \\ V_{k+1} & = & e^{\delta A} V_k, \\ \hat{S}_{k+1} & = & \hat{S}_k \oplus Approx(V_k), \\ \hat{\Omega}_{k+1} & = & R_{k+1} \oplus \hat{S}_{k+1} \end{array}$$

- **Use overapproximation with**

$$Approx(X) \oplus Approx(Y) = Approx(X \oplus Y)$$

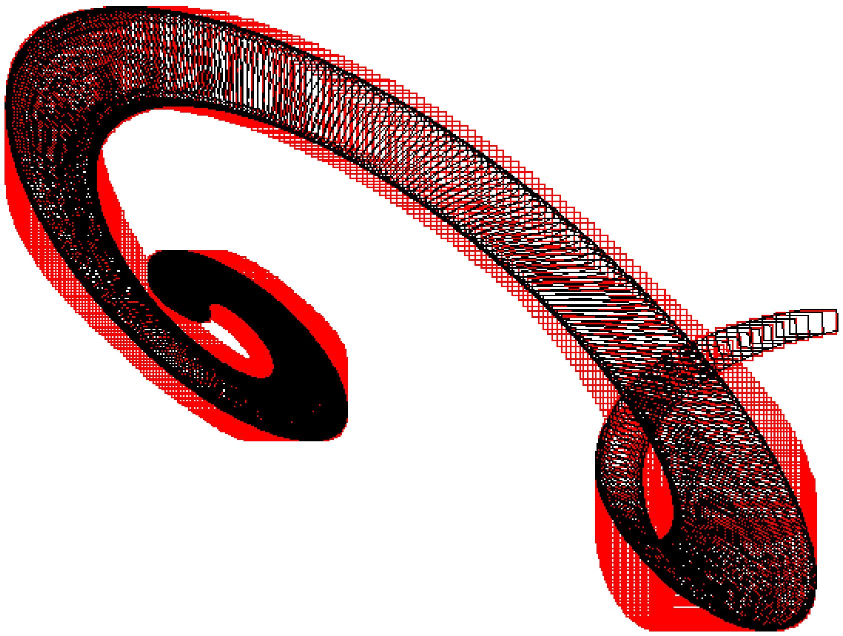
  - bounding box, octagonal, etc.

- **No accumulation of error:**

$$\begin{array}{rcl} \hat{S}_k & = & Approx(S_k) \\ \hat{\Omega}_k & \subseteq & Approx(\Omega_k) \end{array}$$

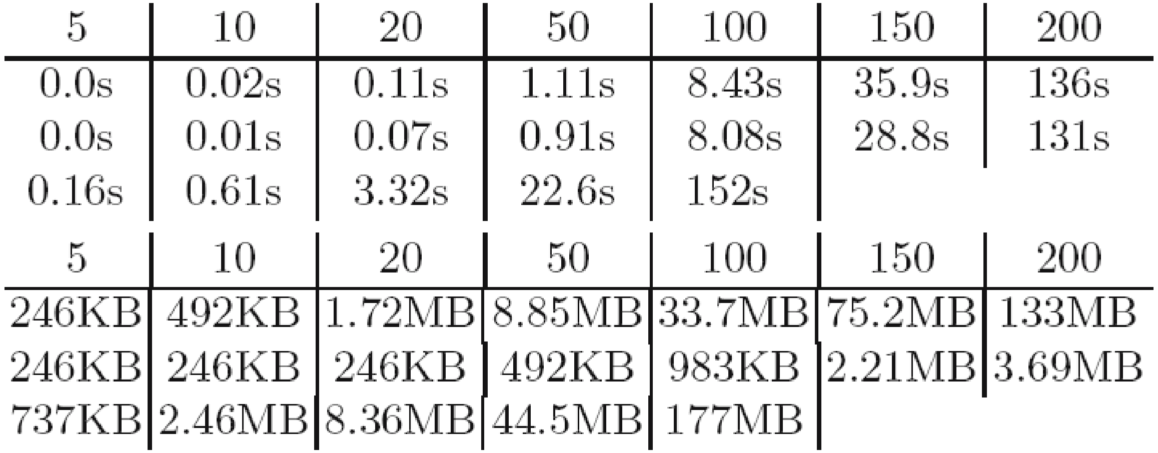
Fighting the Wrapping Effect

- **Exact vs. overapproximation**
  - dimension 5 for 600 time steps
  - overapproximation with bounding box



Experimental Results

- **Time and memory for 100 steps**



	5	10	20	50	100	150	200
4-Sequence Zonotopes	0.0s	0.02s	0.11s	1.11s	8.43s	35.9s	136s
4-Sequence Box	0.0s	0.01s	0.07s	0.91s	8.08s	28.8s	131s
Zonotope, 20 Gen.	0.16s	0.61s	3.32s	22.6s	152s		

	5	10	20	50	100	150	200
4-Sequence Zonotopes	246KB	492KB	1.72MB	8.85MB	33.7MB	75.2MB	133MB
4-Sequence Box	246KB	246KB	246KB	492KB	983KB	2.21MB	3.69MB
Zonotope, 20 Gen.	737KB	2.46MB	8.36MB	44.5MB	177MB		

Outline

I. Hybrid Automata and Reachability

II. Reachability for Simple Dynamics

  a) Linear Hybrid Automata

  b) Piecewise Affine Hybrid Systems

III. Application to Complex Dynamics

  a) Hybridization Techniques

  b) Abstraction Refinement

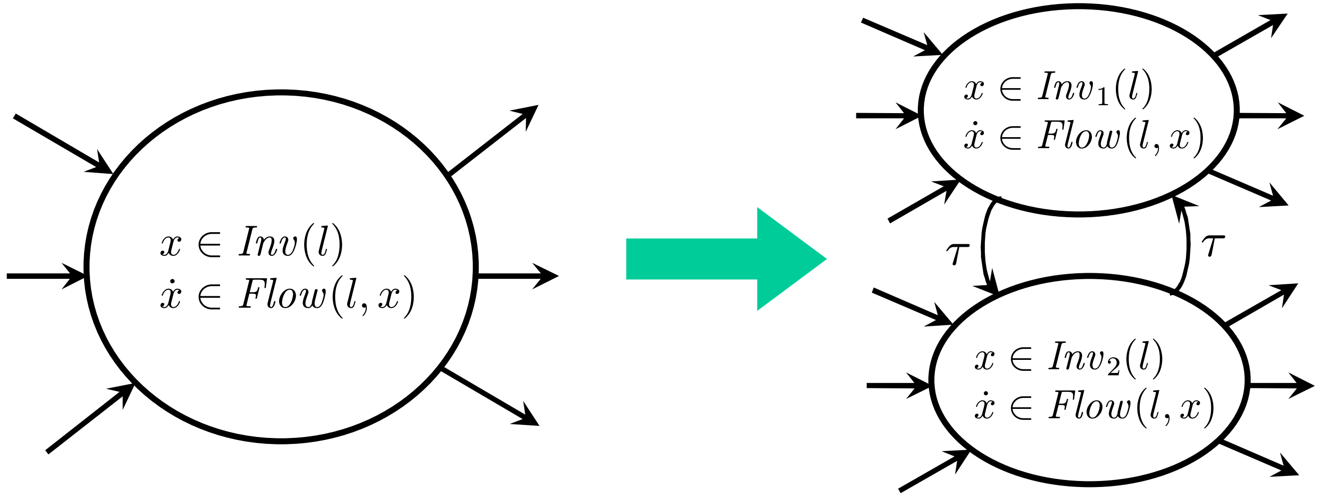
In this Chapter…

- **Complex nonlinear dynamics**
  - and how to overapproximate them with simpler dynamics
- **How to keep approximation error small**
- **Strategic heuristics to improve performance**

Hybridization

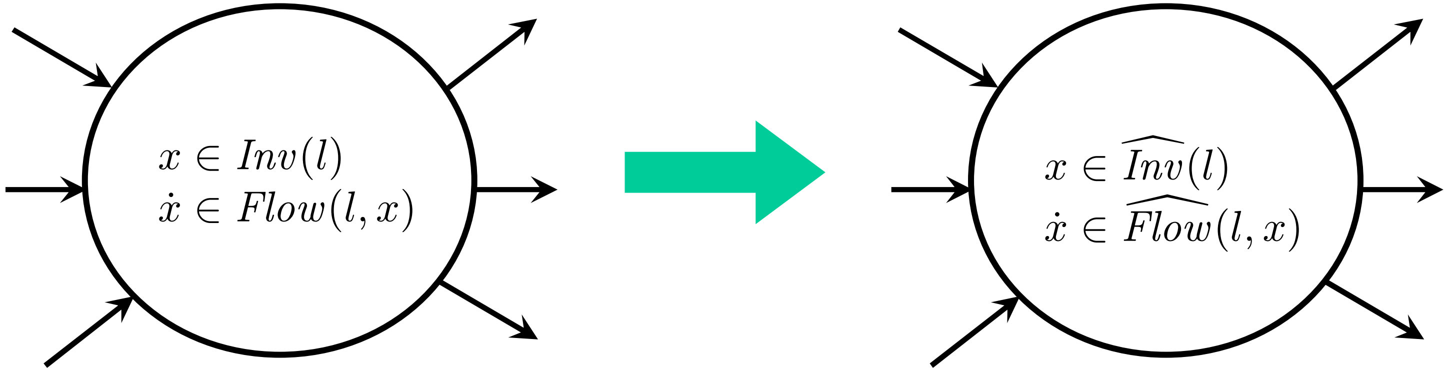
- **Goal: Overapproximation of H with**
  - simpler dynamics
  - approximation error $\leq \epsilon$
- **Observation:**
  - approximation error depends on size of invariant in each location
- **Approach:**
  - split locations until all invariants small enough
  - overapproximate dynamics in each location

Splitting Locations



- **same behavior as before if**
  - τ-transitions don’t change variables and are unobservable
  - $Inv_1 \cup Inv_2 = Inv$ (and some details)

Overapproximating Dynamics



$x \in Inv(l)$
$\dot{x} \in Flow(l, x)$

$x \in \widehat{Inv}(l)$
$\dot{x} \in \widehat{Flow}(l, x)$

- **same or more behavior as before if**

$$\begin{aligned} Inv(l) &\subseteq \widehat{Inv}(l) \\ Flow(l,x) &\subseteq \widehat{Flow}(l,x) \end{aligned}$$

From Affine to LHA-Dynamics

$\dot{x} \in Ax + B, \quad B \subseteq \mathbb{R}^n$ → $\dot{x} \in C, \quad C \subseteq \mathbb{R}^n$

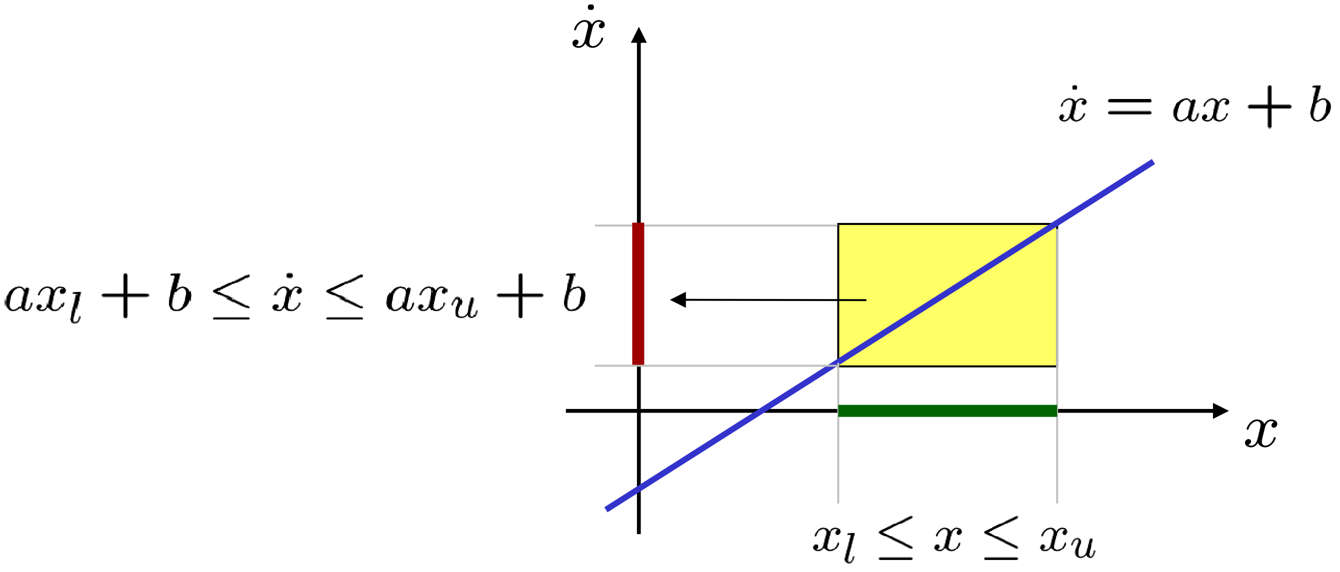
- **By definition $x \in Inv(l)$:**
  - overapproximation

$$C = \{x' \mid \exists x \in Inv(l) : x' \in Ax + B\}$$

- **If B,Inv polyhedra**
  - C polyhedron
  - $O(exp(n))$

From Affine to LHA-Dynamics

$$\dot{x} \in Ax + B, \quad B \subseteq \mathbb{R}^n \quad \Rightarrow \quad \dot{x} \in C, \quad C \subseteq \mathbb{R}^n$$



Hybridization with LHA

- **Bouncing Ball Dynamics**

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -g \end{aligned}$$

  - dynamics of x are affine (depend on v).

- **Invariant:** $x \geq 0$
  - no restriction on $v \Rightarrow \dot{x} \in \mathbb{R}$
  - entire invariant reachable

Hybridization with LHA

- **Bouncing Ball Dynamics**

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -g \end{aligned}$$

- **Split v–axis in K parts**
  - on bounded subset $v \in$ [-2,2]

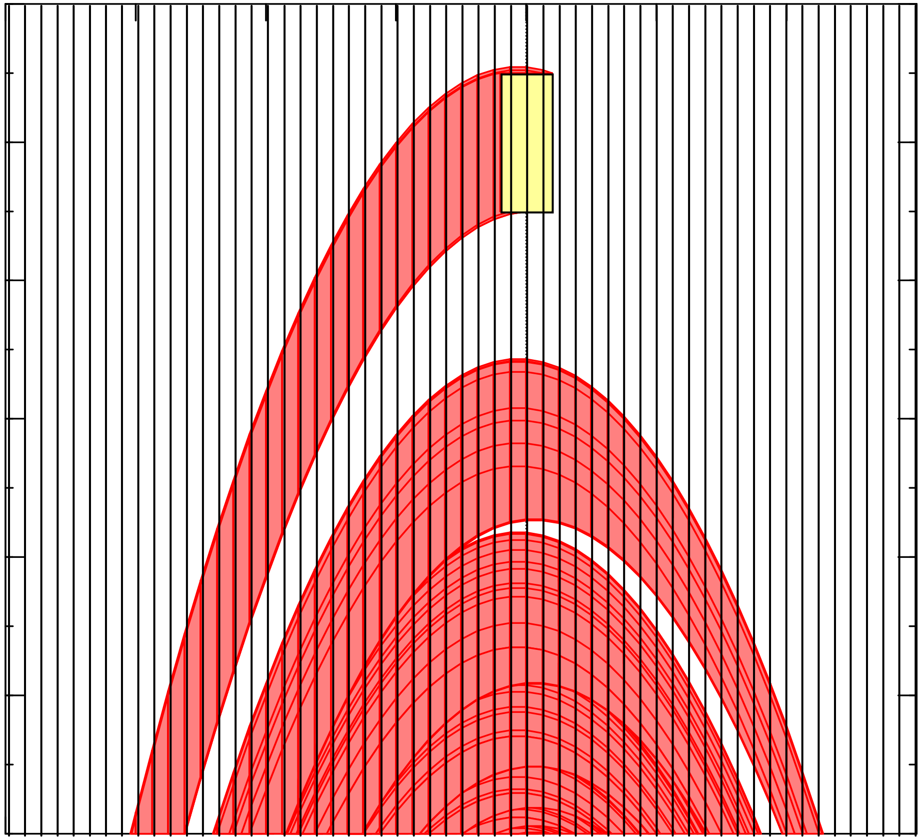
- **Arbitrary accuracy for small enough K**

$$\dot{x} \in \{v \pm 4/K\}$$

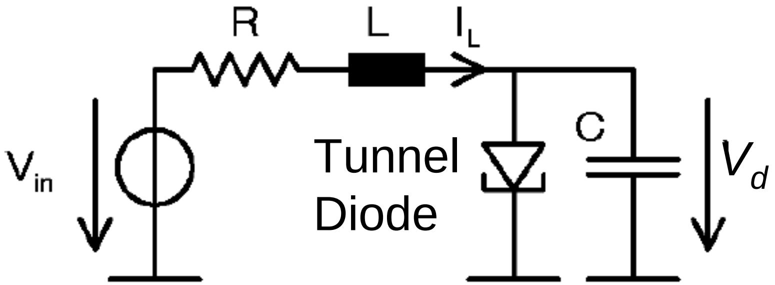
$$K \to \infty \quad \Rightarrow \quad \dot{x} \to v$$

Hybridization with LHA

- **Bouncing Ball – Reachable states for $K{=}64$:**



Tunnel Diode Oscillator

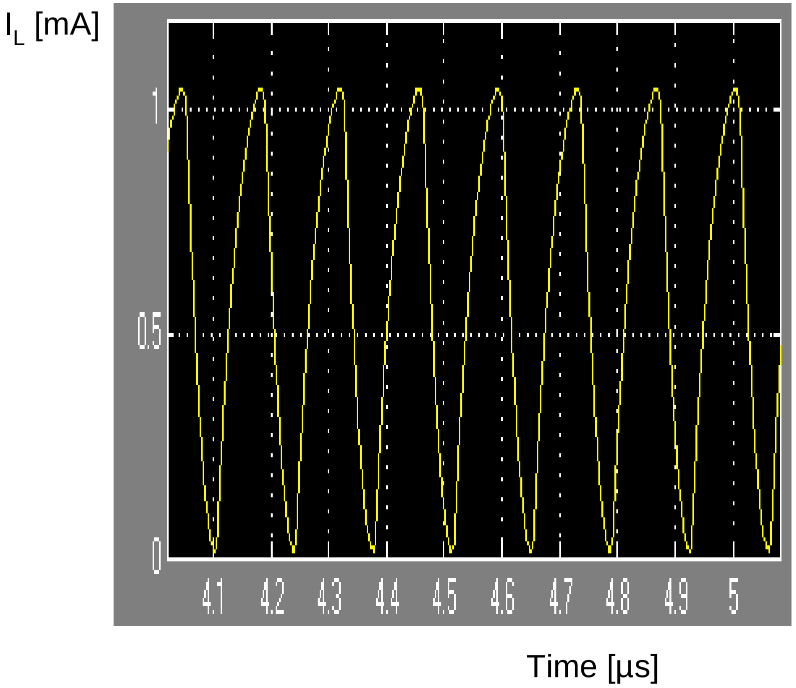


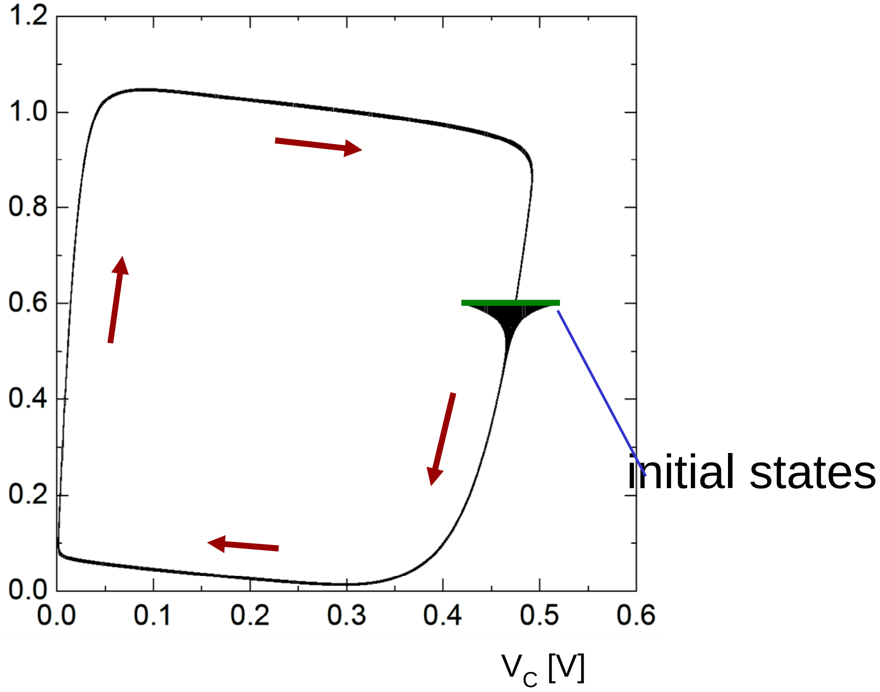
$$\dot{V}_C = \tfrac{1}{C}\left(-I_d(V_C) + I_L\right)$$
$$\dot{I}_L = \tfrac{1}{L}\left(-V_C - RI_L + V_{in}\right)$$

- **What are good parameters?**
  - startup conditions
  - parameter variations
  - disturbances

Tunnel Diode Oscillator

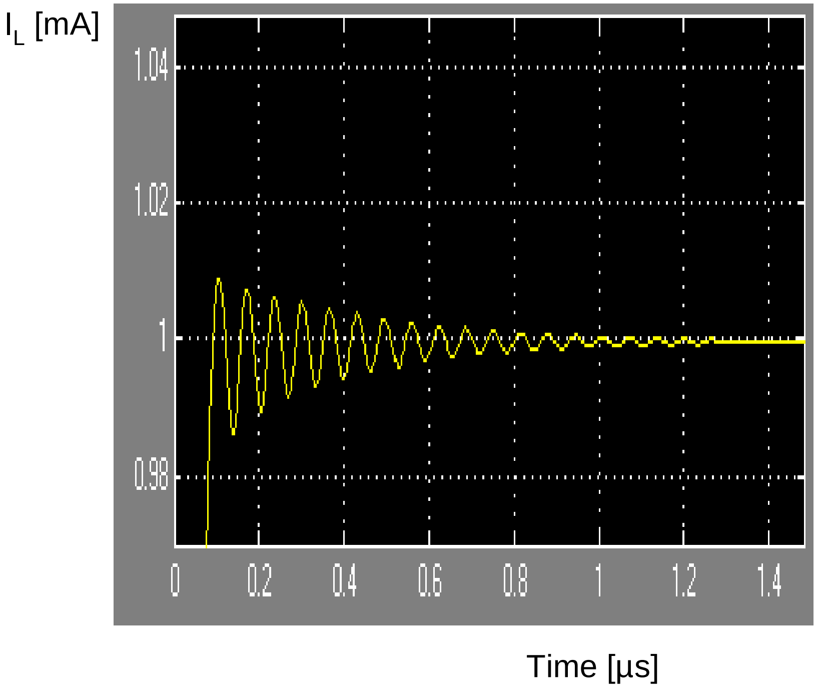
R=0.20Ω ⇒ Oscillation

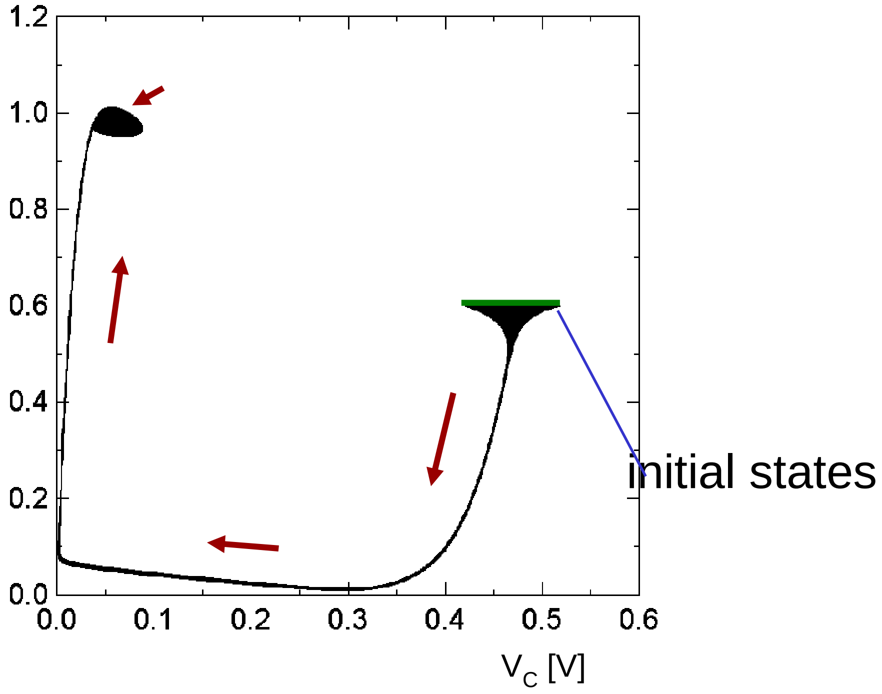




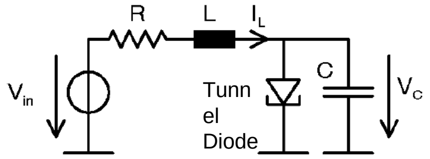
Tunnel Diode Oscillator

R=0.24Ω ⇒ Stable equilibrium

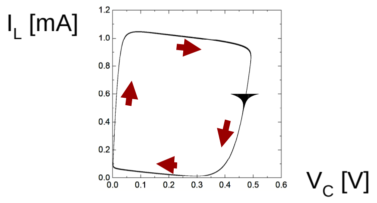




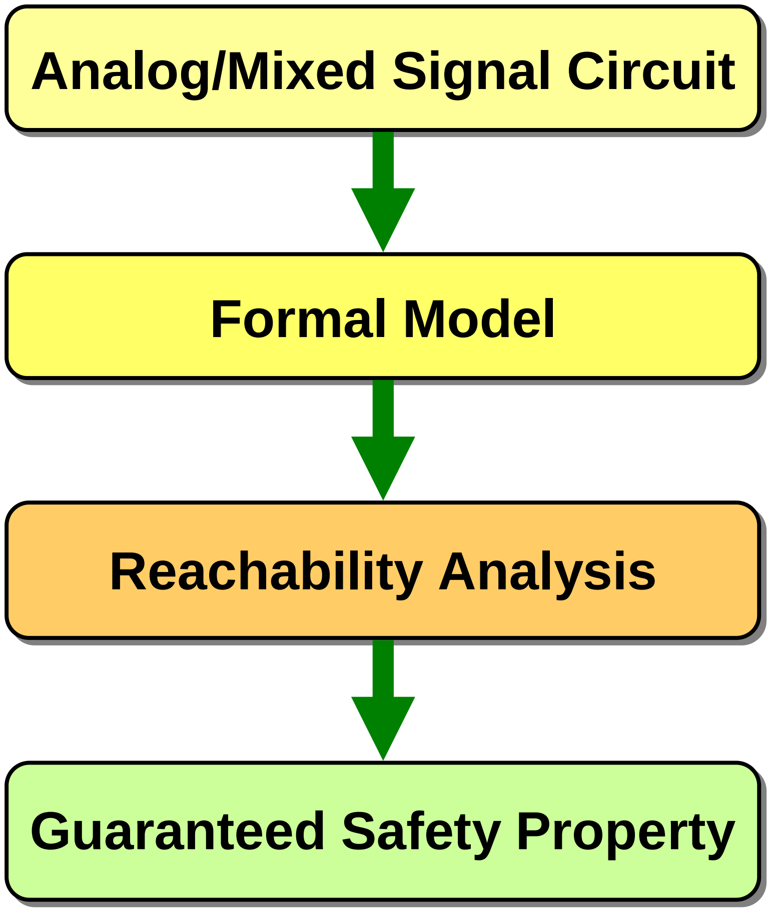
Tunnel Diode Oscillator



$$\dot{V}_C = \tfrac{1}{C}\left(-I_d(V_C) + I_L\right)$$
$$\dot{I}_L = \tfrac{1}{L}\left(-V_C - RI_L + V_{in}\right)$$



- **Oscillation**
- **Jitter**
- **…**



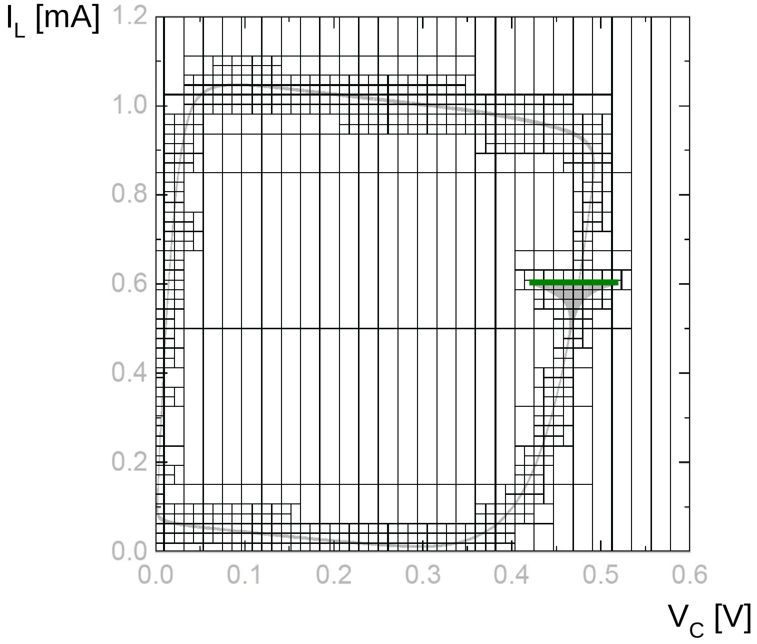
Analog/Mixed Signal Circuit

Formal Model

Reachability Analysis

Guaranteed Safety Property

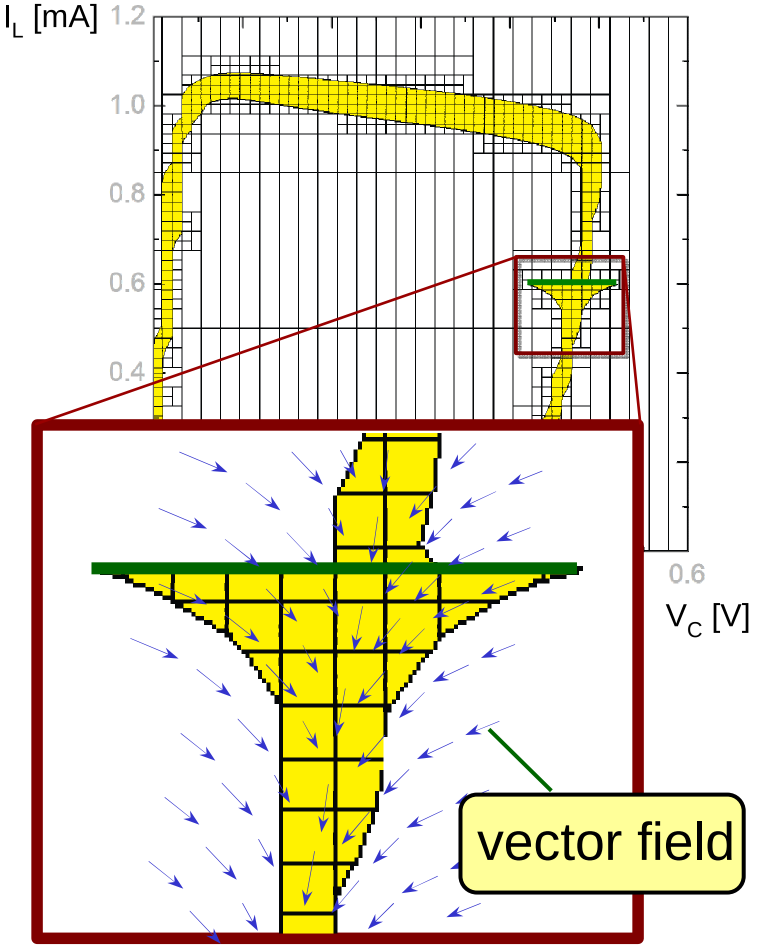
Reachability Analysis



1. **Hybridization**
   - Partition State Space (on the fly)
   - Switching between

   ⇒ Hybrid System

Reachability Analysis

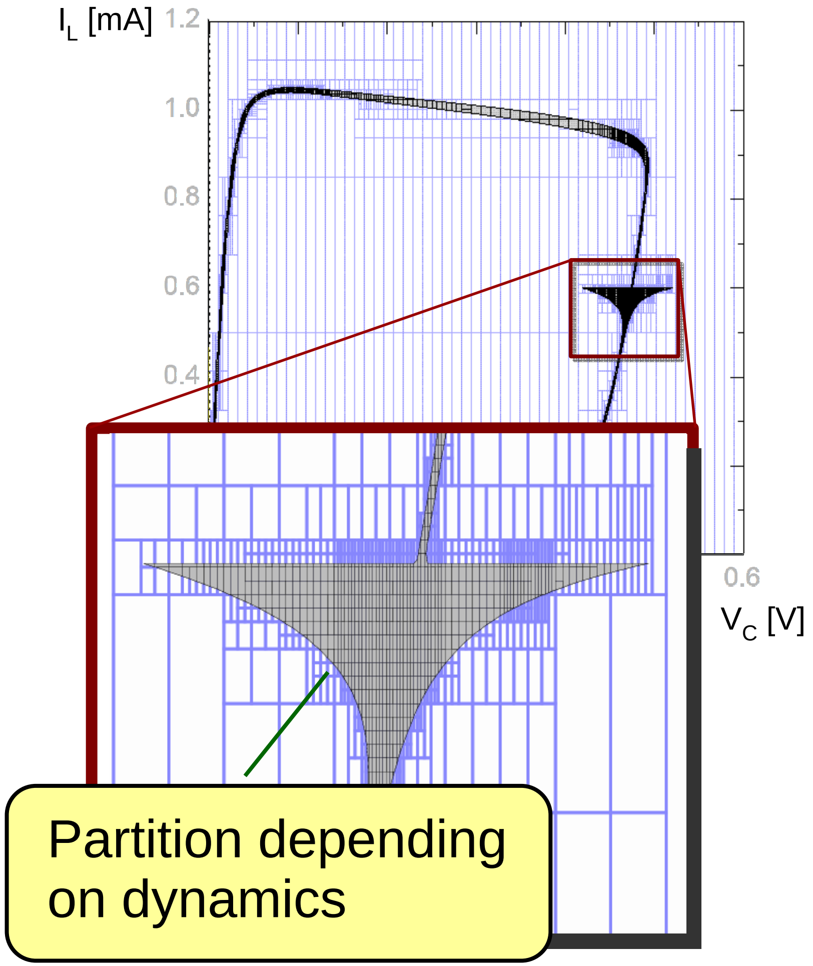


1. **Hybridization**
   - Partition State Space (on the fly)
   - Switching between

   ⇒ Hybrid System
2. **Overapproximation**
   - Linear Hybrid Automata

⇒ **Polyhedral enclosure of actual trajectories**

Reachability Analysis



Partition depending on dynamics

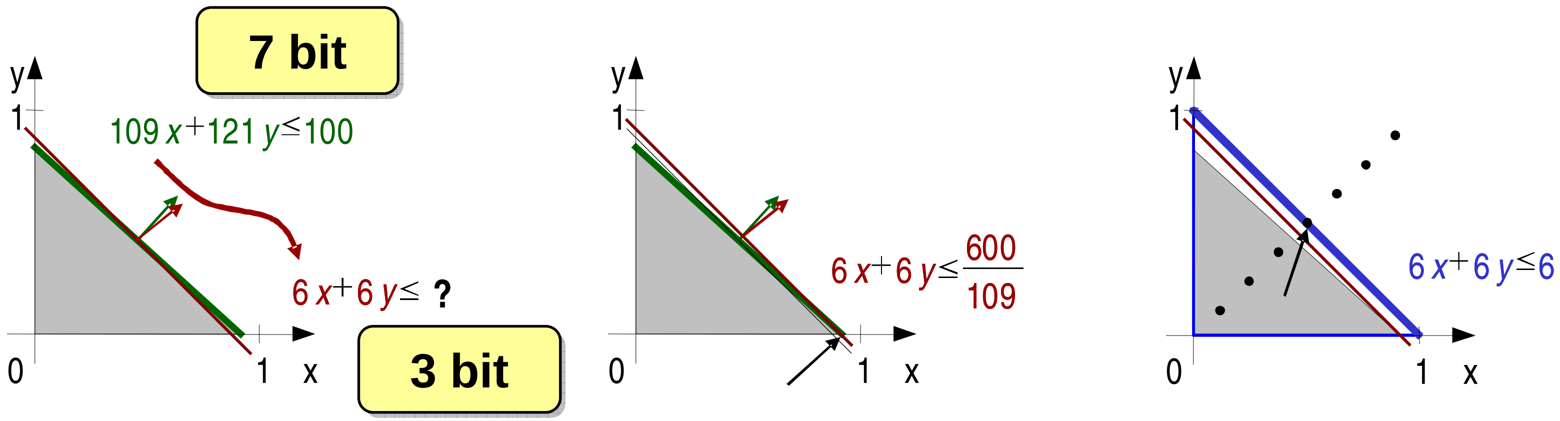
- **Efficiency through**
  - adapting partitions to dynamics
  - overapproximation of complex polyhedra with simplified polyhedra
- **Good performance**
  - Reachability with high accuracy in 72s, 127MB

Hybridization with LHA

- **Problems with high accuracy**
  - requires small partitions
  - small partitions $\rightarrow$ small fractional coefficients $\rightarrow$ large integer representations
  - complex dynamics $\rightarrow$ complex fixpoint
- **Simplification of polyhedra needed**
  - must be overapproximations

Limiting the Number of Bits

1. truncate bits of coefficients
2. push plane to outside (solve LP)
3. snap to next largest integer



7 bit

$109\,x+121\,y\leq 100$

$6\,x+6\,y\leq$?

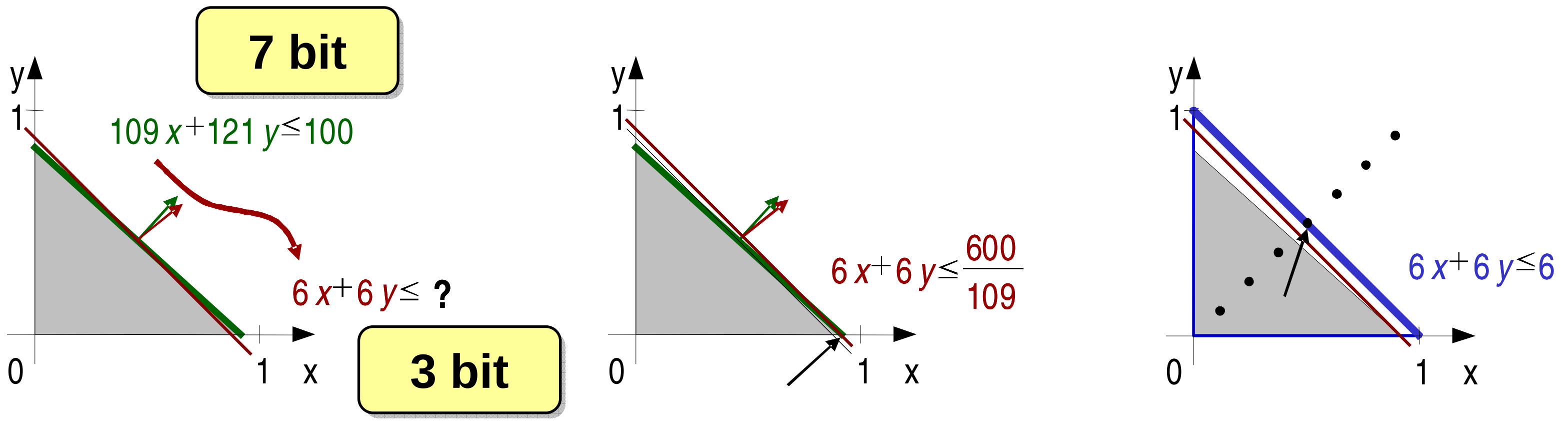
3 bit

$6\,x+6\,y\leq\frac{600}{109}$

$6\,x+6\,y\leq 6$

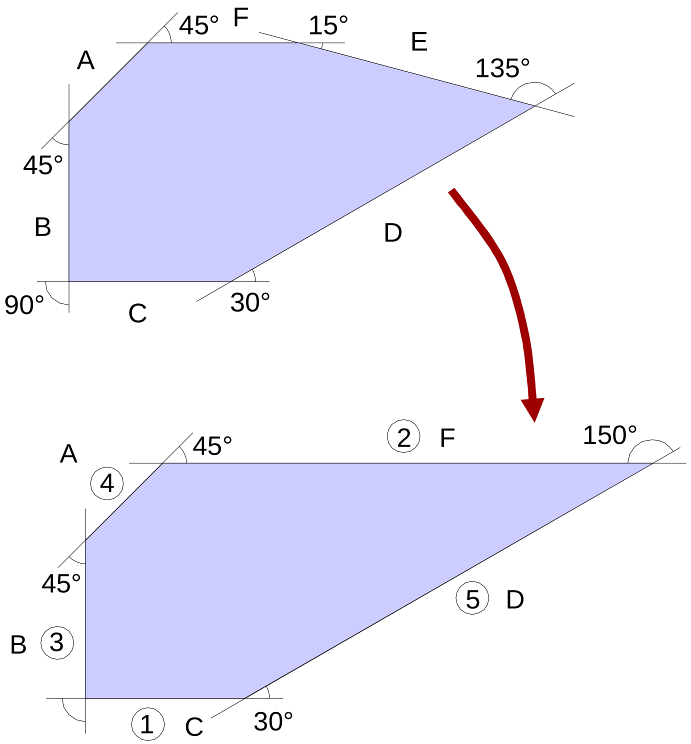
Limiting the Number of Bits

1. truncate bits of coefficients
2. push plane to outside (solve LP)
3. snap to next largest integer



- **in practice large problems infeasible without**
- **guarantees termination**
  - finite number of possible constraints
- **but: unbounded error**

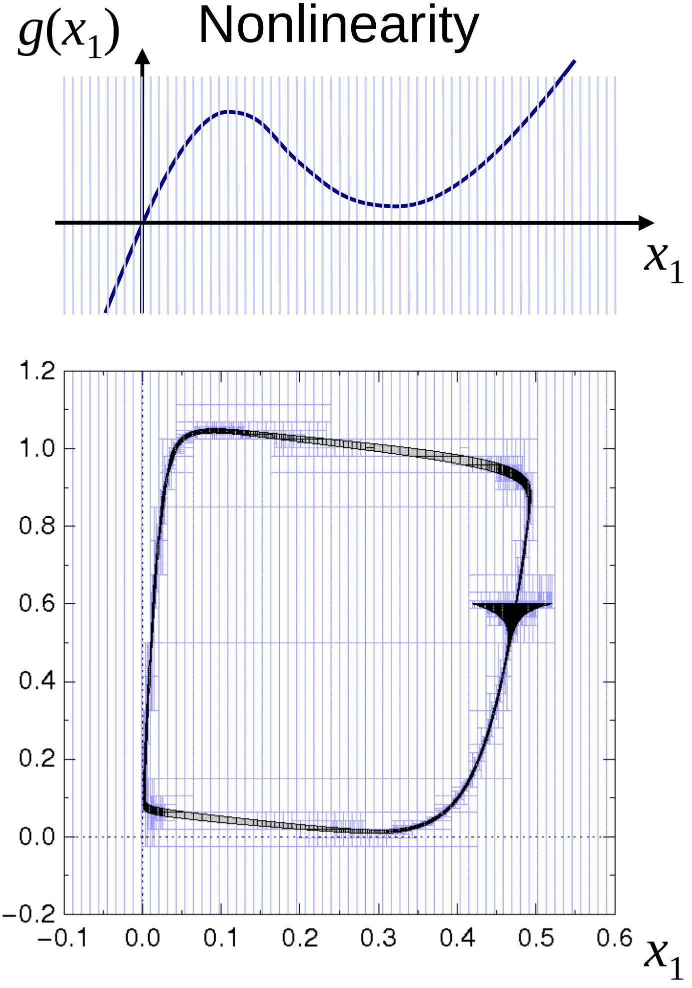
Limiting the Number of Constraints



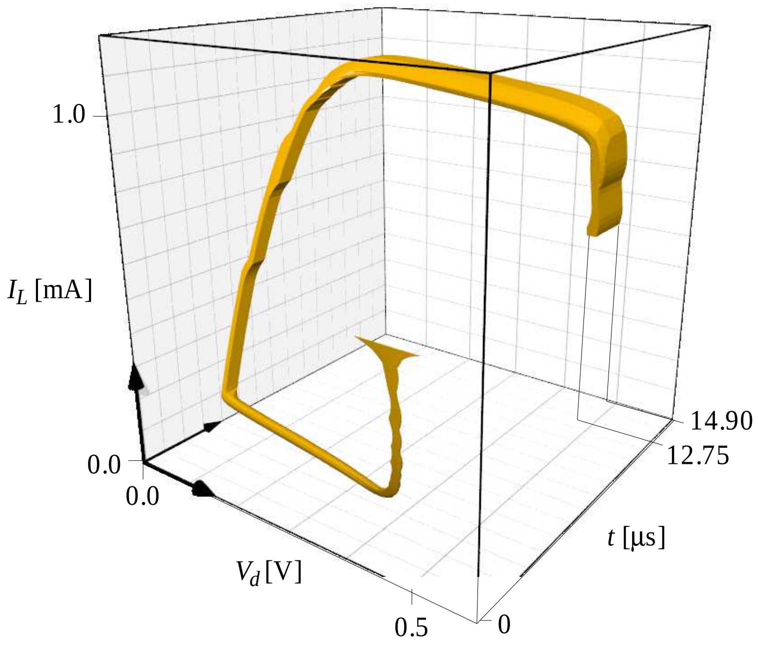
From 6 to 5 constraints

- **Reduce from *m* to *z* constraints**
- **Significance Measure *f*(*m*,*d*)**
  - Volume: exp
  - Slack: LP
  - **max. angle: m²d**
    - ⇒ $- \min_{i \neq j} a_i^T a_j$
- **Heuristics to choose constraints**
  - **deconstruction:** *drop (m-z) least significant*
  - **reconstruction:** *add z most significant*
- **Experiments: angle & reconstr.**
  - 1000 → 50 in 4 dim: < 2 sec. (1000x faster than slack)

Clocked Tunnel Diode Oscillator

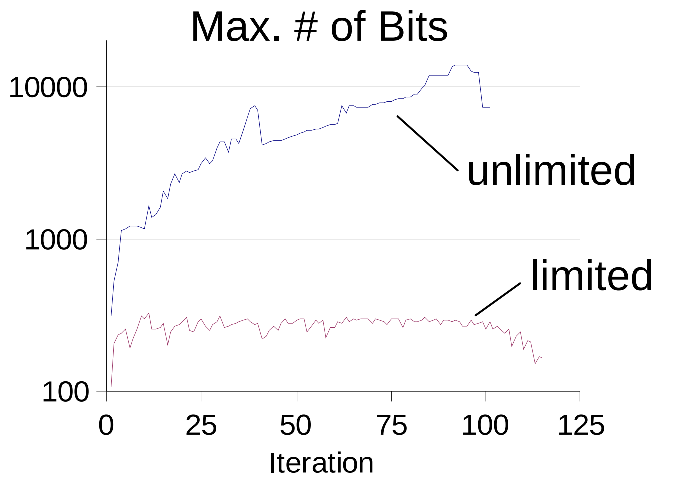


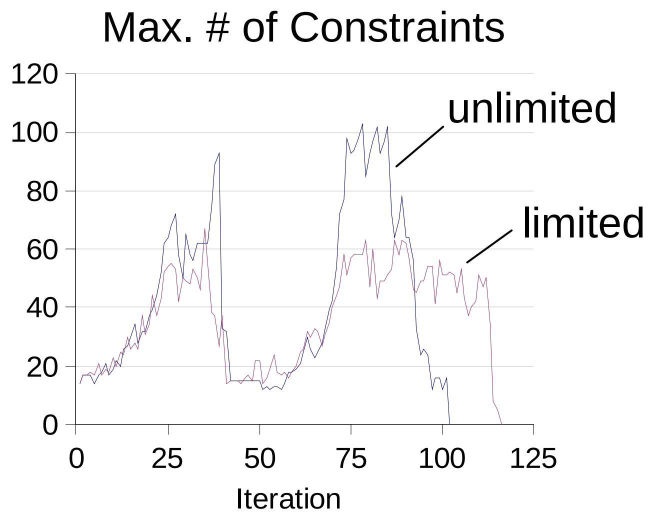
- 2-dim. oscillator
  + clock to measure bound on cycle time
  = 3-dim. system



Clocked Tunnel Diode Oscillator

- **Limiting at every iteration bad**
  - prohibitively expensive
  - convergence problems
- **Trigger limiting at threshold**
  - 300 bits ⇒ 16 bits
  - 56 constraints ⇒ 32 constraints
- **Comparison for low accuracy:**
  - **12x faster, 20% memory**
  - Loss of accuracy: **< 0.3%**

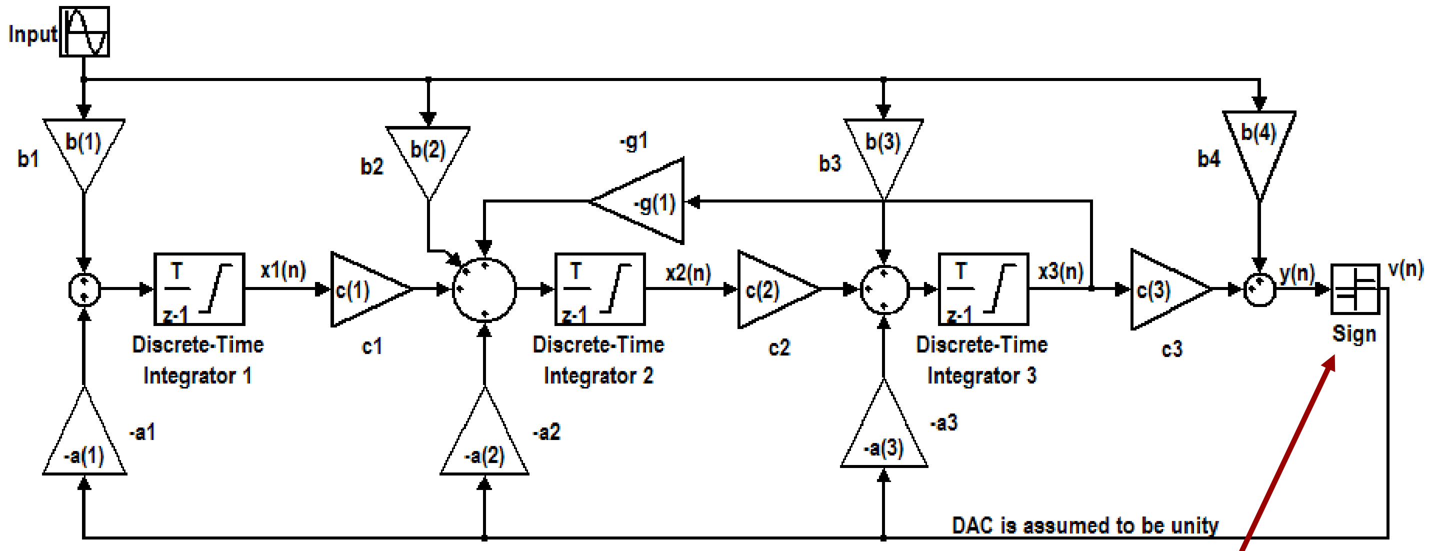




Hybridization with LHA

- **Problem with reachability computations:**
  - fixpoint may be complex
  - or even not representable by finite number of polyhedra (spirals…)
- **Apply overapproximation techniques**
- **Splitting locations can “localize” error**
  - approximation error limited to invariant
  - small invariant $\to$ small error

3rd-Order Delta Sigma Modulator



- **Analog/Digital converter**
  - linear circuit + 1-bit quantizer
  - 3 discrete-time integrators
- **To show: quantizer input in [-2,2]**

Monitor quantizer input

Richard Schreier. The delta-sigma toolbox version 6.0, January 2003.

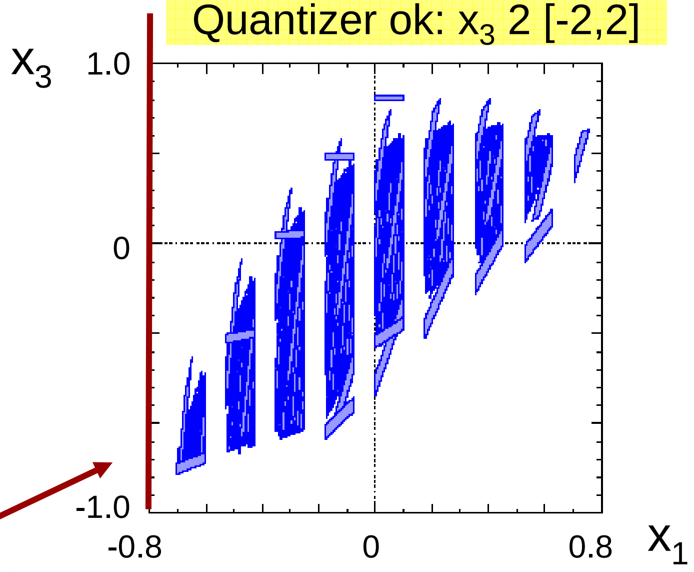
Symbolic Execution

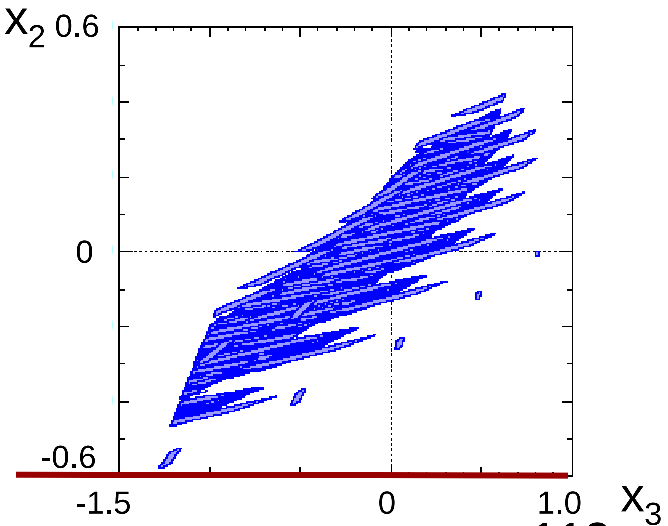
- **All runs of fixed length**
  - const. input, cont. set of initial states

	Depth	Time	Polyh.
Failure Detected			
CheckMate		3min	
PHAVer	15	0.17s	116
No Failure			
PHAVer	100	6min	189,414

- **Advantage**
  - Find errors after few time steps
- **Drawback**
  - Combinatorial explosion inherent in switching algorithm prevents longer horizons

Quantizer ok: x_3 2 [-2,2]



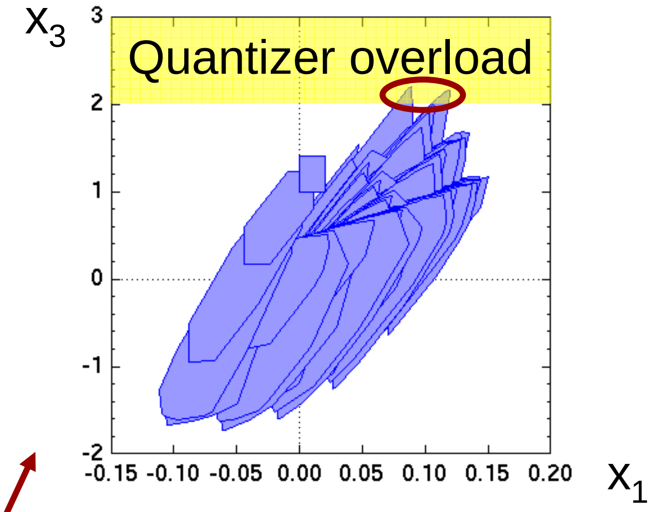


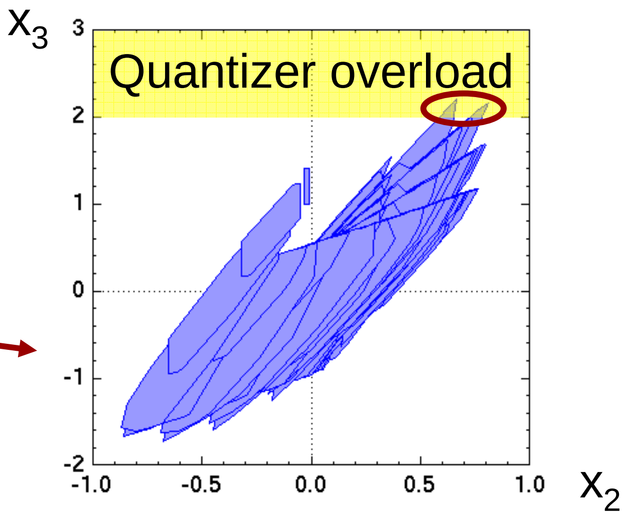
Quantizer ok: x_3 2 [-2,2]

Delta-Sigma Modulator – Variable Input

- **All runs of fixed length**
  - cont. set of initial states
- **Variable Input**
  - changing to arbitrary values at each sampling step
  - modeled using state variable $\rightarrow$ 4-dimensional system
  - greatly increased complexity

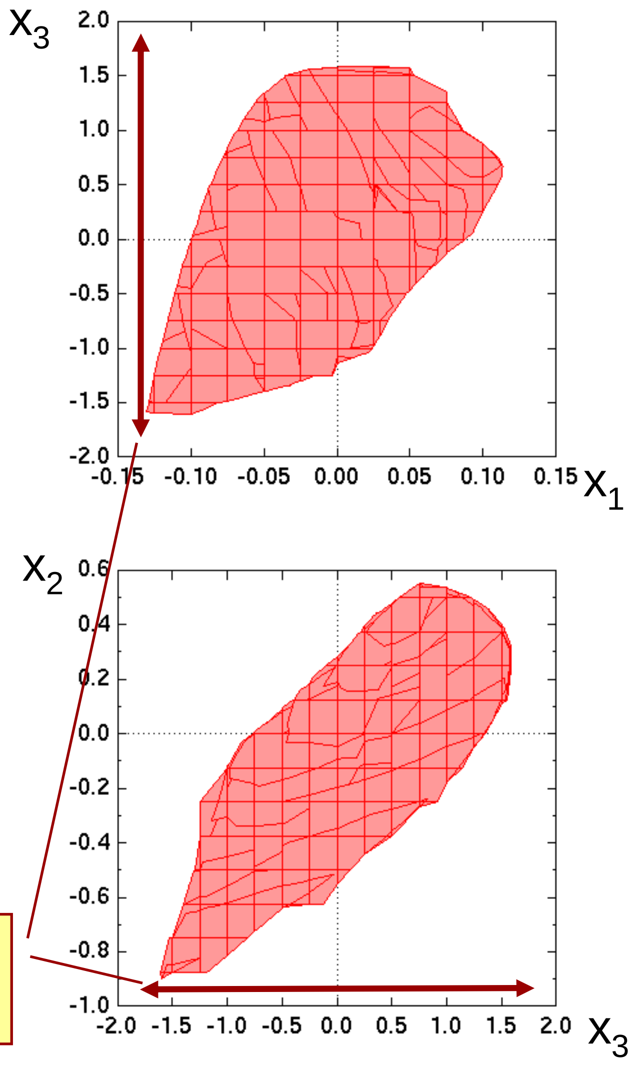
	Depth	Time	Polyh.
Failure Detected, u $\in$ [0,0.8]			
PHAVer	9	195s	464
No Failure, u $\in$ [0.5,0.6]			
PHAVer	18	12min	940

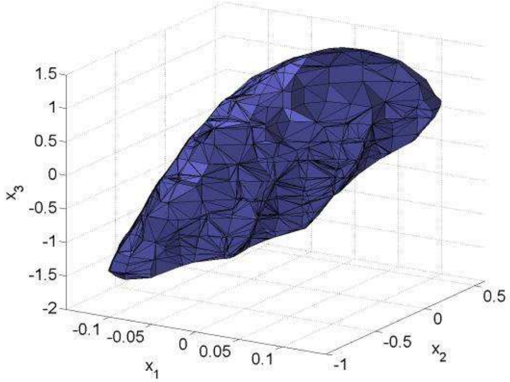




Delta-Sigma Modulator - Reachability

- **Infinite time horizon**
- Compute convex hull
  - cover state space, so eventually new states will be contained
- Limit bits + constraints
- Localize overapproximation by partitioning
  - otherwise too large in undesirable directions
- **Computation: 34min, 224MB**





Saturation Bounds guaranteed

Nonlinear Dynamics

- **Continuous Time System**

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

- **Hybridization**
  - partition state space (invariant) into small regions
  - overapproximate with simpler dynamics in each region

Nonlinear Dynamics

- **Continuous Time System**

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

- **Approximation with affine dynamics**

$$\dot{x} \quad = \quad Ax + Bu, u \in U$$

- U **modeling approximation error**
  - determine U such that

$$F(x) - Ax \in BU$$

Van der Pool Oscillator

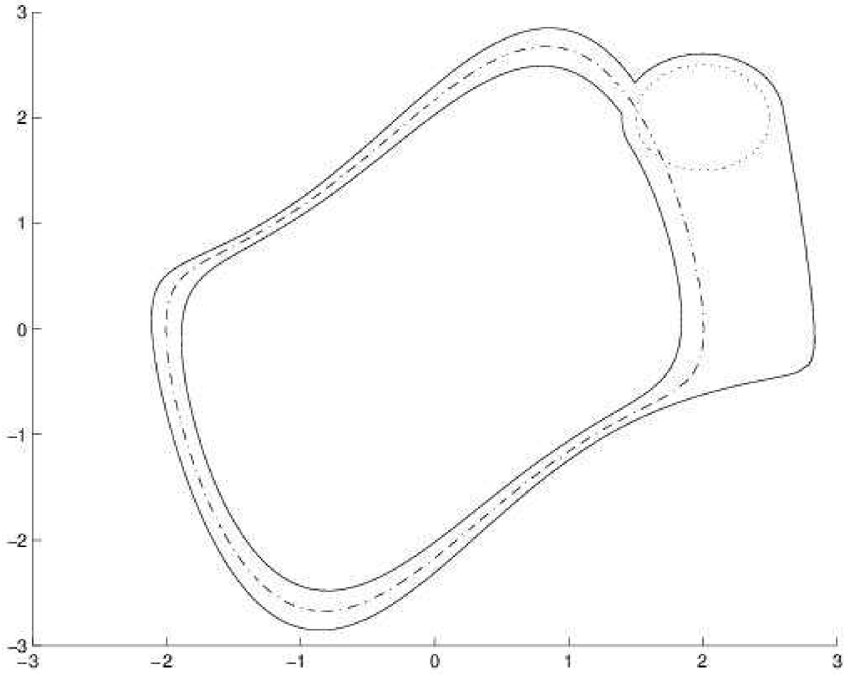
- **Nonlinear Continuous Time System**

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= y(1-x^2)-x\end{aligned}$$

- **Reachability Analysis using Hybridization**
  - approximation with piecewise affine dynamics
  - uniform triangular mesh, partition of size 0.05
  - result used in detection of limit cycle

Van der Pool Oscillator

- **Reachable states**



Outline

I. **Hybrid Automata and Reachability**

II. **Reachability for Simple Dynamics**

   a) Linear Hybrid Automata

   b) Piecewise Affine Hybrid Systems

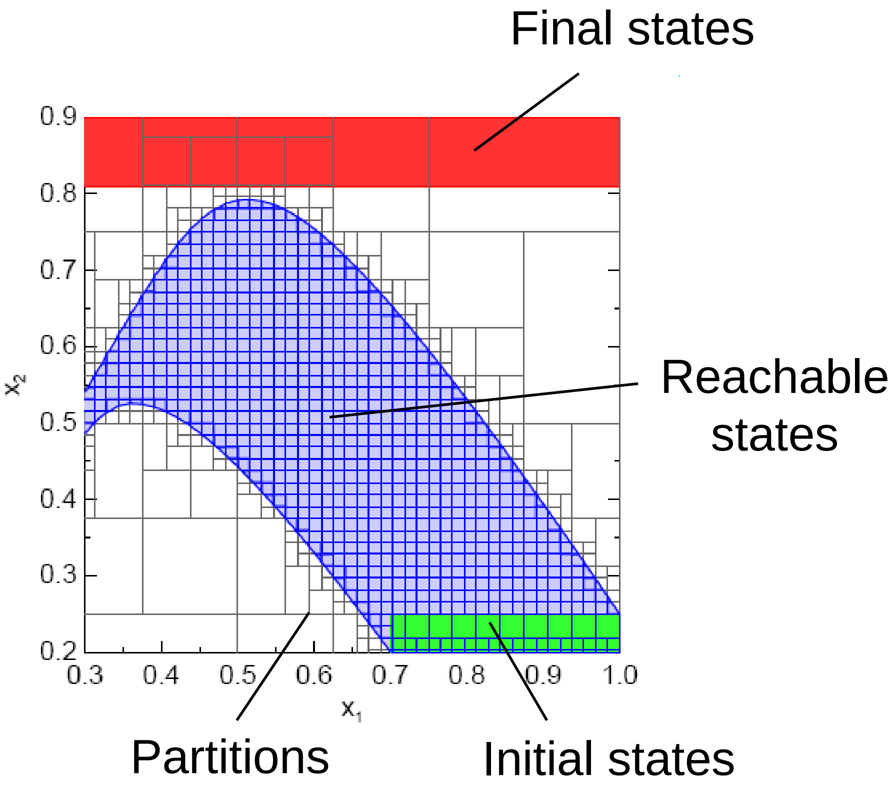
III. **Application to Complex Dynamics**

   a) Hybridization Techniques

   b) Abstraction Refinement

Forward/Backward Refinement - Principle

- **To show:**
  - bad states not reachable
- **Observation:**
  - Small partitions not leading to bad states
- **Solution:**
  - forward/backward between initial and bad states
  - smaller partitions at each step



F/B-Refinement - Example

Step 1

a) Forward reachability with coarse partition R_1

Step 2

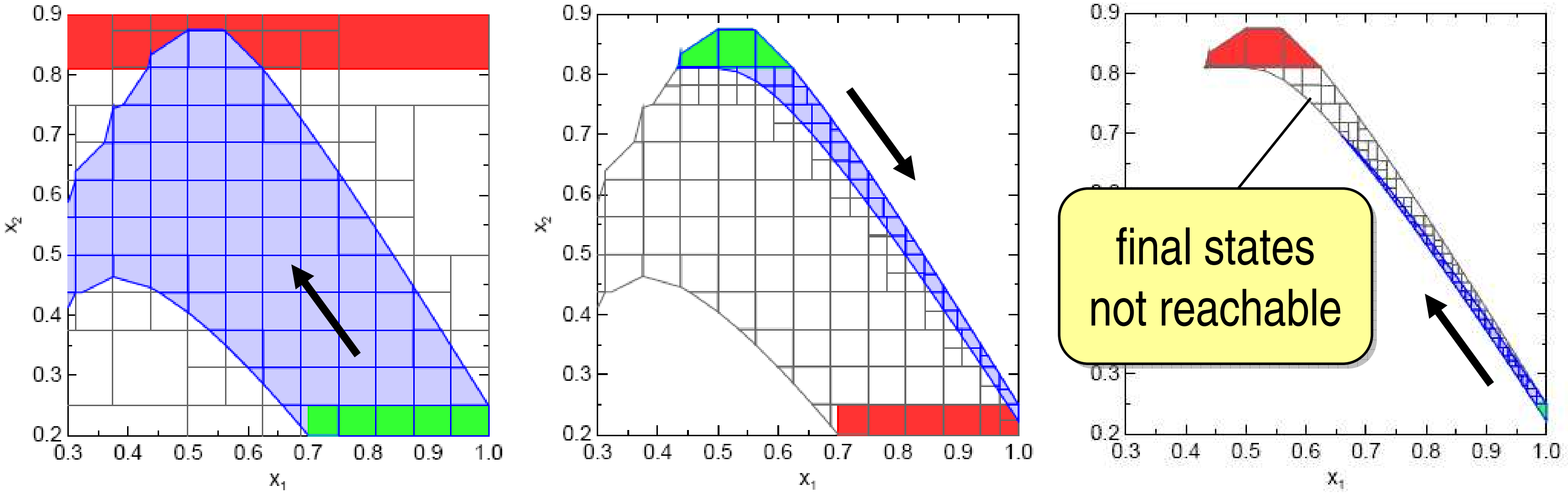
a) Restrict final states and invariants to R_1

b) Backward reachability with finer partition R_2

Step 3

a) Restrict final states and invariants to R_2

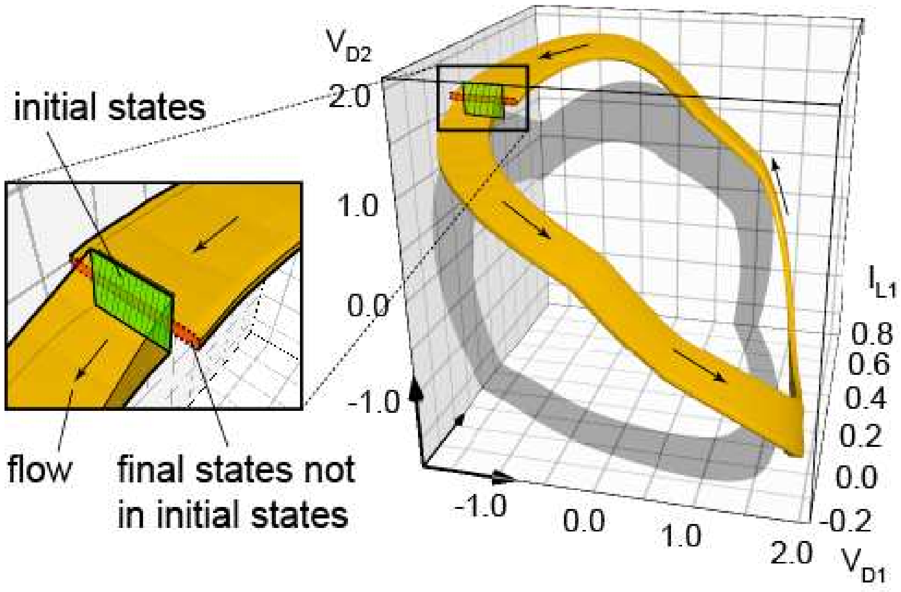
b) Backward reachability with finer partition R_3

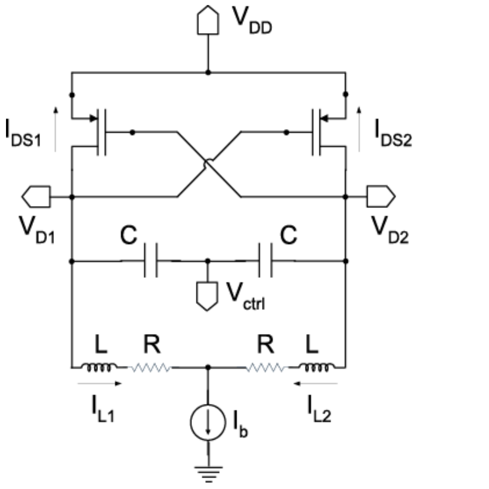


final states not reachable

Voltage Controlled Oscillator

- **3-dim. system with nonlinearity**
- **Goal: Show invariance of cycle**





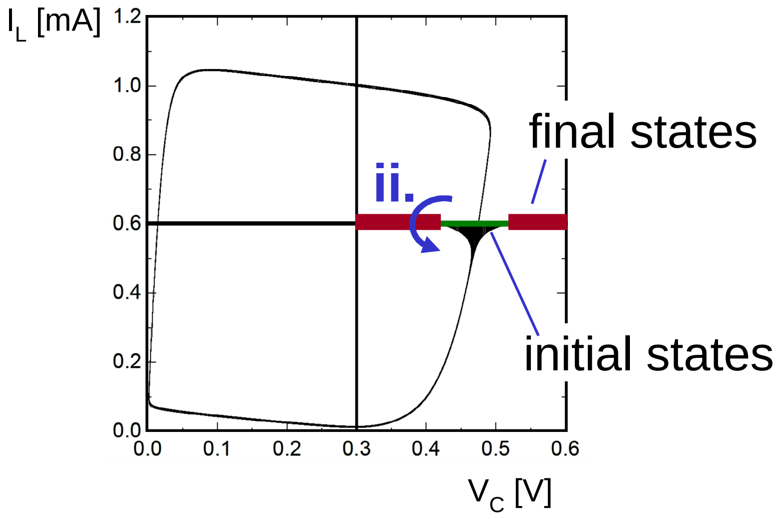
$$\dot V_{D1} = -\frac{1}{C}(I_{DS1}(V_{D2}-V_{DD}, V_{D1}-V_{DD}) + I_{L1}),$$

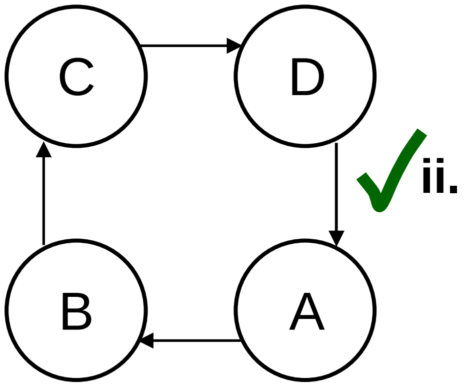
$$\dot V_{D2} = -\frac{1}{C}(I_{DS2}(V_{D1}-V_{DD}, V_{D2}-V_{DD}) + I_b - I_{L1}),$$

$$\dot I_{L1} = \frac{1}{2L}(V_{D1}-V_{D2}-R(2I_{L1}-I_b)),$$

- × No success after 20min, 1GB RAM
- × 64x accuracy needed ⇒ 20h, 64GB?

F/B-Refinement of VCO

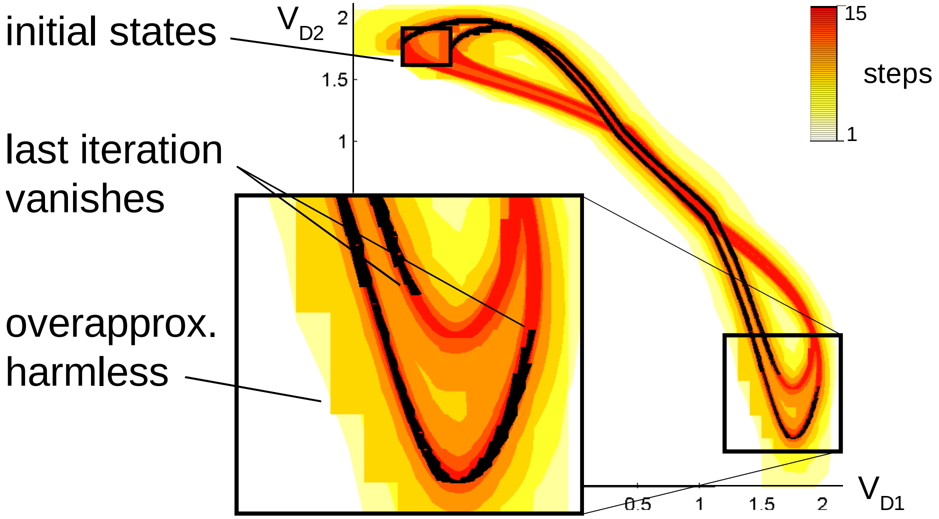


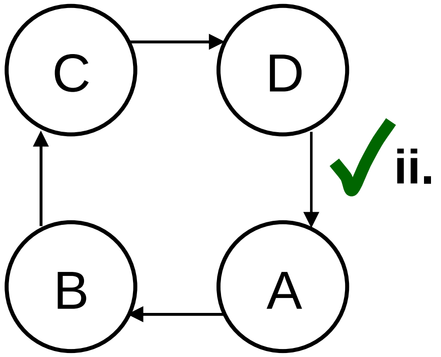


hybrid automaton

- **F/B-Refinement**
  - final (forbidden) := states outside initial
  - not reachable ⇒ any cycle passes through initial states

F/B-Refinement of VCO



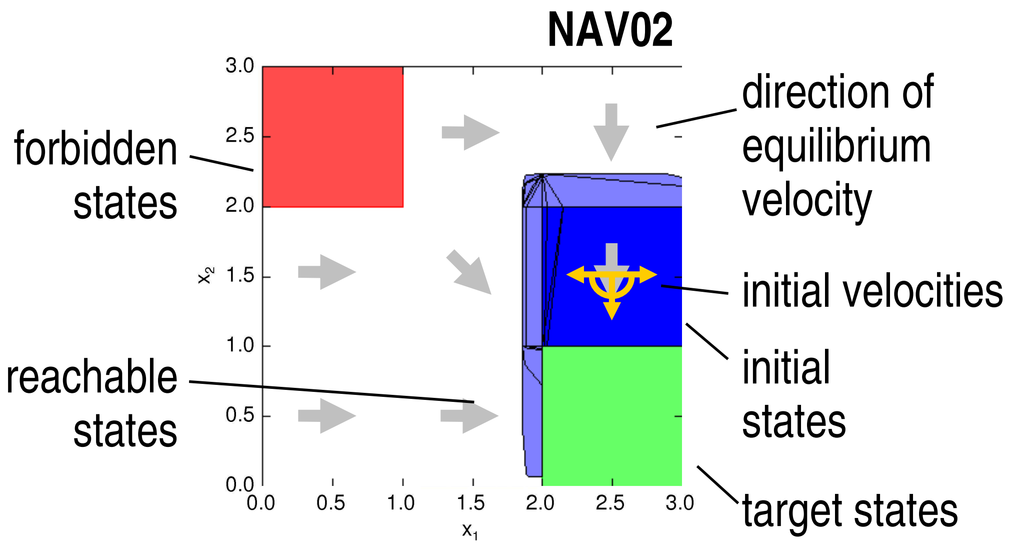


hybrid automaton

- **F/B-Refinement**
  - final (forbidden) := states outside initial
  - not reachable ⇒ any cycle passes through initial states
- **Success**
  - 11.5h, 1.7GB RAM

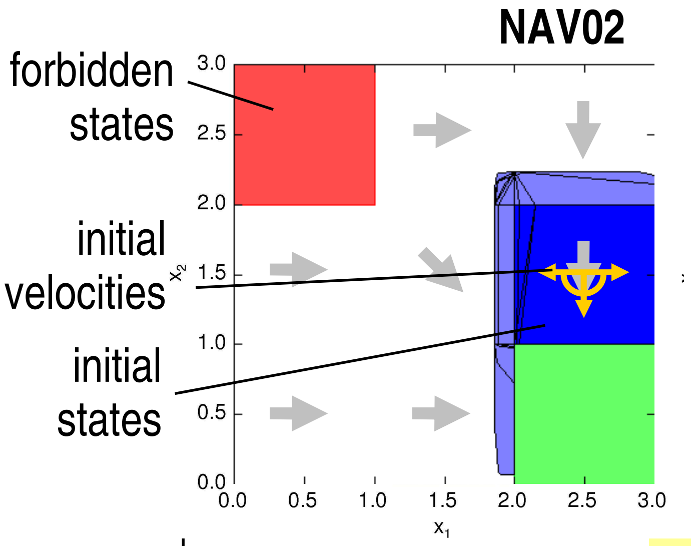
Navigation Benchmark

- **Fehnker, Ivancic. *Benchmarks for Hybrid Systems Verification.* HSCC'04**



- **“Balloon driven by wind”**
  - moving object in plane
  - 4-dimensional piecewise affine dynamics (position, velocity)
  - equilibrium velocity depends on position
- **Instances NAV01-NAV29 with increasing difficulty**
- **Verification Task: Reachability of forbidden states**

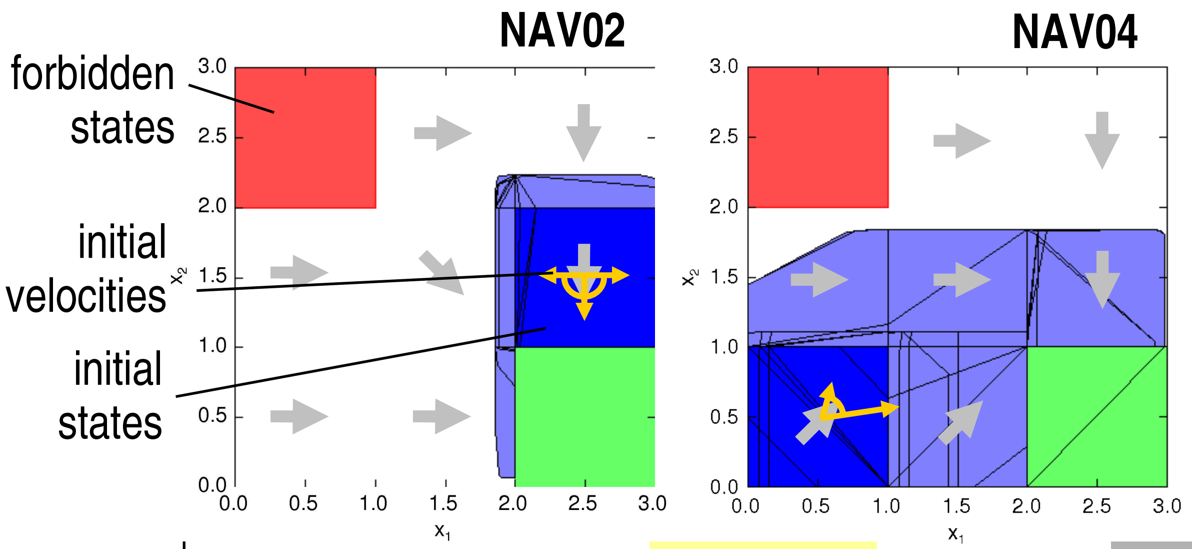
Navigation Benchmark



Tool Instance	d/dt Verimag '00	Pred. Abstr. UPenn'02 4x250MHz Sun	PHAVer '05/'06 2.8GHz P4	TimePass Stanf. '06 PIII(!)	PHAVer F/B-Ref.'05 3GHz Xeon		PHAVer F/B-Ref.'05 2.8GHz P4
NAV01	~30s	34s	5s 27MB	5s 2MB	5s	*Doyen,*	32s 59MB
NAV02	~150s	153s 68MB	6s 27MB	73s 5MB	10s	*Henzinger,*	34s 60MB
NAV03	?	152s 180MB	6s 27MB	78s 5MB	10s	*Raskin*	33s 60MB

No results with:	HyTech ('95-'00, Henzinger)
	CheckMate ('98-'05, CMU)
	HSOLVER ('05, MPI)

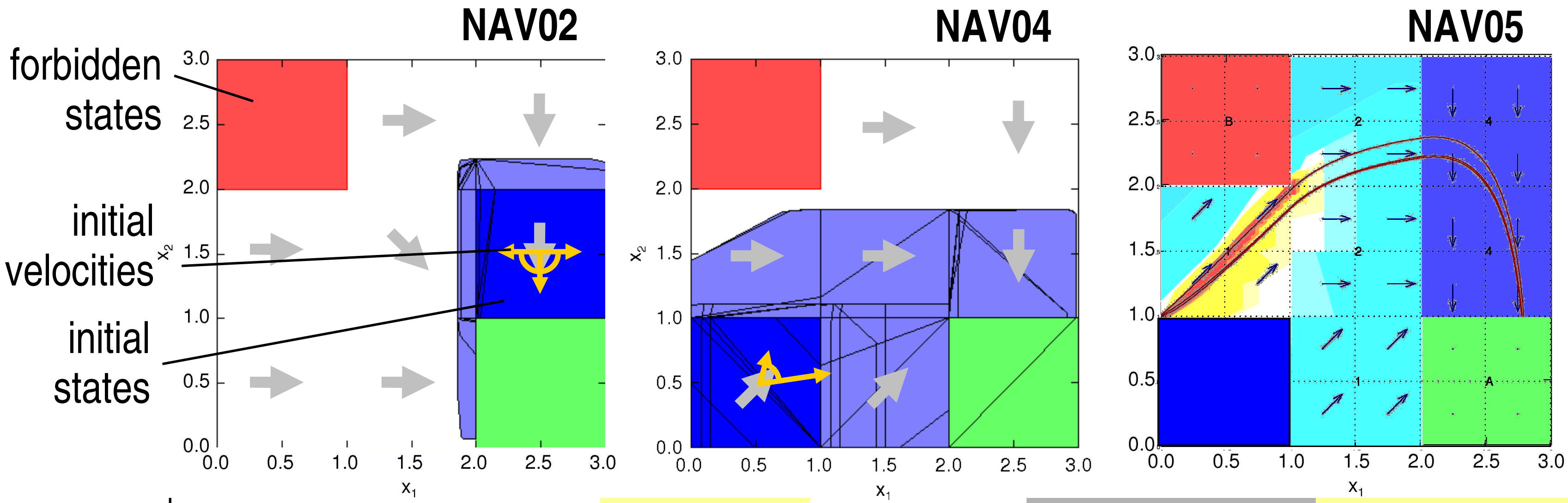
Navigation Benchmark



Tool Instance	d/dt Verimag '00	Pred. Abstr. UPenn'02 4x250MHz Sun	PHAVer '05/'06 2.8GHz P4	TimePass Stanf. '06 PIII(!)	PHAVer F/B-Ref.'05 3GHz Xeon		PHAVer F/B-Ref.'05 2.8GHz P4
NAV01	~30s	34s	5s 27MB	5s 2MB	5s	*Doyen,*	32s 59MB
NAV02	~150s	153s 68MB	6s 27MB	73s 5MB	10s	*Henzinger,*	34s 60MB
NAV03	?	152s 180MB	6s 27MB	78s 5MB	10s	*Raskin*	33s 60MB
NAV04	"	-?-	8s 48MB	1191s 16MB	75s	*Sept. '05*	81s 52MB

Only results: PHAVer & TimePass

Navigation Benchmark



forbidden states

initial velocities

initial states

NAV02

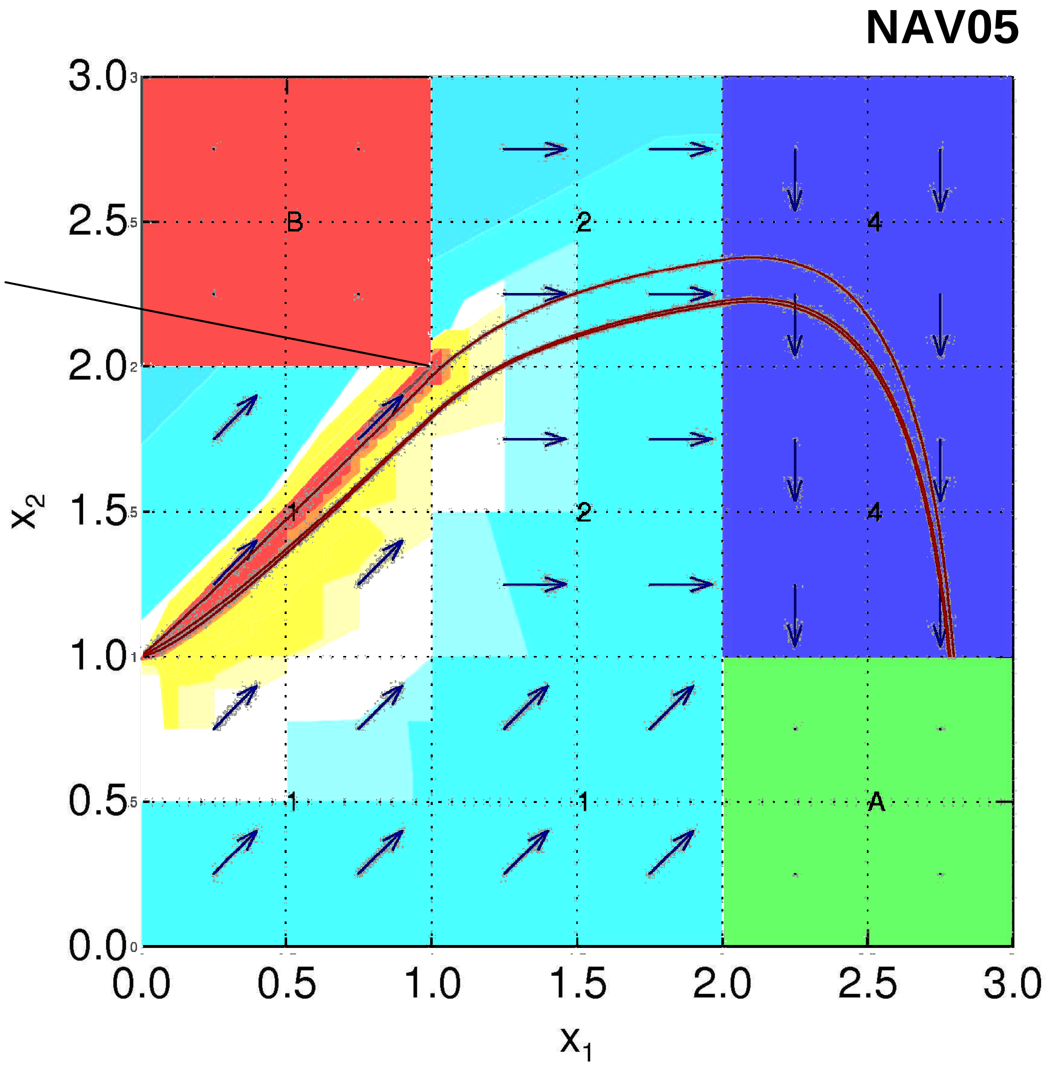
NAV04

NAV05

Tool Instance	d/dt Verimag '00	Pred. Abstr. UPenn'02 4x250MHz Sun	PHAVer '05/'06 2.8GHz P4	TimePass Stanf. '06 PIII(!)	PHAVer F/B-Ref.'05 3GHz Xeon		PHAVer F/B-Ref.'05 2.8GHz P4
NAV01	~30s	34s	5s 27MB	5s 2MB	5s	*Doyen,*	32s 59MB
NAV02	~150s	153s 68MB	6s 27MB	73s 5MB	10s	*Henzinger,*	34s 60MB
NAV03	?	152s 180MB	6s 27MB	78s 5MB	10s	*Raskin*	33s 60MB
NAV04	"	-?-	8s 48MB	1191s 16MB	75s	*Sept. '05*	81s 52MB
NAV05							46000s 529MB
NAV06							48000s 575MB

convergence problems → widening? [Halbwachs94]

Navigation Benchmark



high accuracy required

Bibliography

- **Hybrid Systems Theory**
  - Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. Theoretical Computer Science 138:3-34, 1995
  - Thomas A. Henzinger. The theory of hybrid automata. Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1996, pp. 278-292
- **Linear Hybrid Automata**
  - Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi, HyTech: The next generation. RTSS'95
  - Goran Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past HyTech. HSCC'05

Bibliography

- **Affine Dynamics**
  - E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems. HSCC’00
  - A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. HSCC’06
- **Hybridization and Nonlinear Dynamics**
  - Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. IEEE Transactions on Automatic Control 43:540-554, 1998
  - E. Asarin, T. Dang, and A. Girard. Reachability Analysis of Nonlinear Systems Using Conservative Approximation. HSCC’03
- **Forward/Backward Refinement**
  - G. Frehse, B. H. Krogh, R. A. Rutenbar. Verifying Analog Oscillator Circuits Using Forward/Backward Abstraction Refinement. DATE’06

Verification Tools for Hybrid Systems

- **HyTech: LHA**
  - http://embedded.eecs.berkeley.edu/research/hytech/
- **PHAVer: LHA + affine dynamics**
  - http://www-verimag.imag.fr/~frehse/
- **d/dt: affine dynamics + controller synthesis**
  - http://www-verimag.imag.fr/~tdang/Tool-ddt/ddt.html
- **Matisse Toolbox: zonotopes**
  - http://www.seas.upenn.edu/~agirard/Software/MATISSE/
- **HSOLVER: nonlinear systems**
  - http://hsolver.sourceforge.net/