Carnegie Mellon

Reachability Analysis and Verification

Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania – USA
krogh@ece.cmu.edu

1:48

Carnegie Mellon

Reachability Analysis and Verification

Lecture 1: Transition Systems & Verification
- Verification of Transition Systems
- Simulation and Bisimulation
- Application to Hybrid Systems

Lecture 2: Hybrid System Reachability
- Polyhedral Approximations
- CheckMate (a tool)
- Low-Order Representations

Lecture 3: Linear Hybrid Automata
- LHA Reachability
- Approximating Richer Dynamics
- PHAVer (a tool)
- Iterative Relaxation Abstractions

2:48

Lecture 1: Transition Systems & Verification

Bruce H. Krogh
Carnegie Mellon University
krogh@ece.cmu.edu

3:48

Outline

- Verification of Transition Systems
- Simulation and Bisimulation
- Application to Hybrid Systems

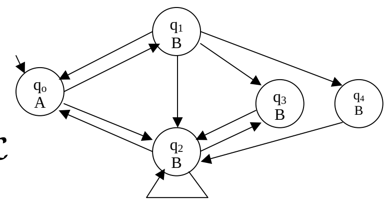
4:48

CarnegieMellon

Transition System (TS)

$T = (Q, \rightarrow, Q_0, \mathcal{L}, L)$

- states Q
- transitions $\rightarrow \subseteq Q \times Q$
- initial states $Q_0 \subseteq Q$
- labels (atomic propositions) $\mathcal{L}$
- labeling function $L : Q \rightarrow 2^{\mathcal{L}}$



Note: We assume the transition relation is total, i.e., $\forall\ q \in Q, \exists\ q' \in Q\ \ni\ q \rightarrow q'$.

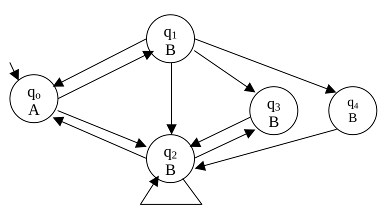
5:48

CarnegieMellon

Paths & Runs

path: $\pi = q_0 q_1 \ldots \in Q^\omega,\ q_i \rightarrow q_{i+1}\ \forall\ i \geq 0$

run: a path for which $q_0 \in Q_0$



e.g., $\pi = q_0 q_1 (q_3 q_2)^\omega$ is a run.

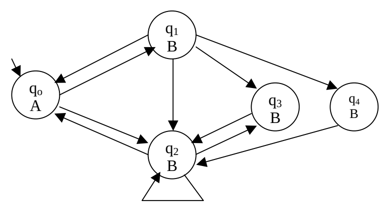
6:48

Predecessors (Pre) and Successors (Post)

For $P \subseteq Q$

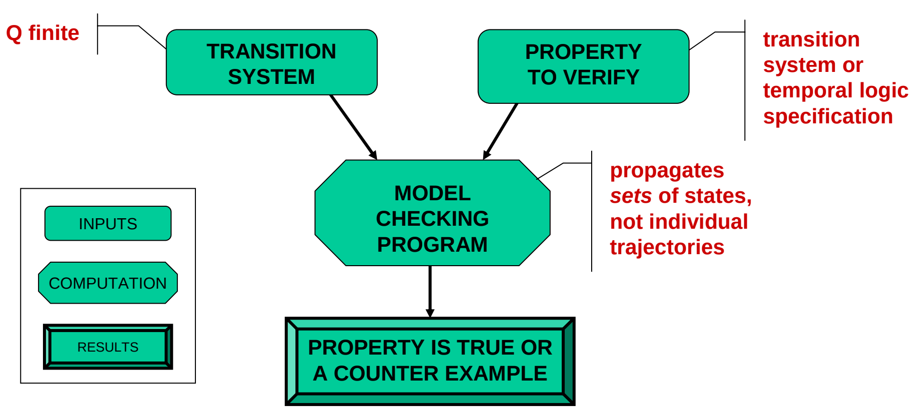
predecessors: $Pre(P) = \{q \in Q \mid \exists\, p \in P, q \to p\}$

successors: $Post(P) = \{q \in Q \mid \exists\, p \in P, p \to q\}$



7:48

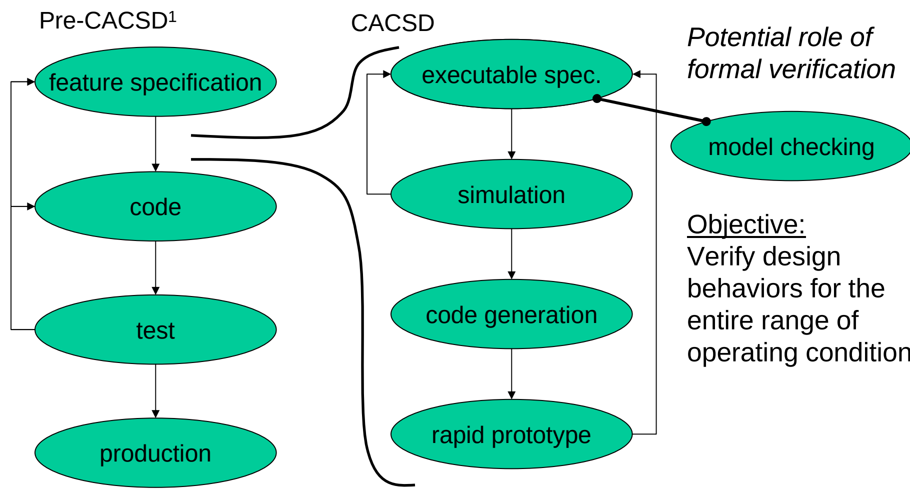
Formal Verification - Model Checking



Model checking is *algorithmic* (guaranteed to terminate).

8:48

Where does verification fit in the control system design flow?



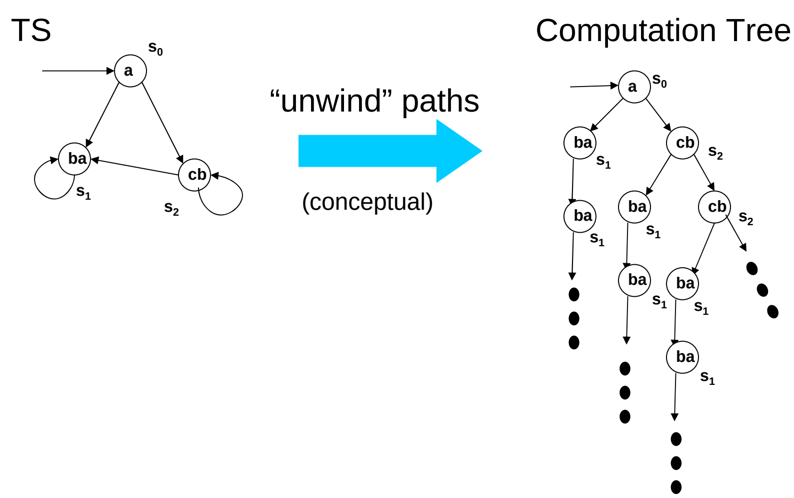
Potential role of formal verification

Objective:
Verify design behaviors for the entire range of operating conditions.

[1]CACSD = computer aided control system design

9:48

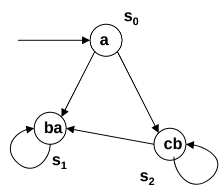
The Computation Tree

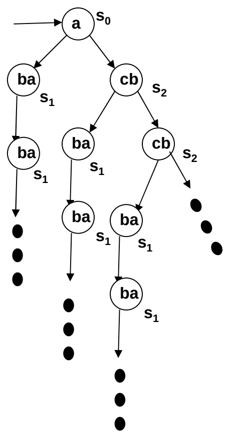


10:48

Path Formulas

- Temporal Operators (along a path)
  - **G** : for all states (globally)
  - **F** : at some future state
  - **X** : next state
  - **U** : until (f U g, f is true until g is true)
- Example: G(a → X(b))

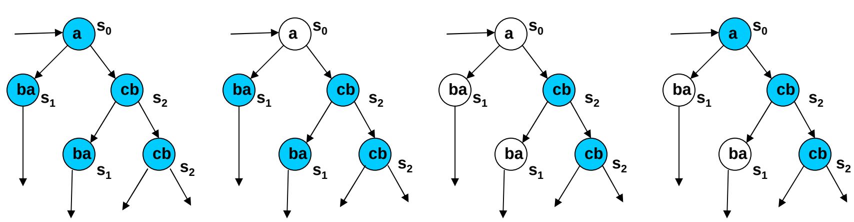




Branching Time Logic - CTL

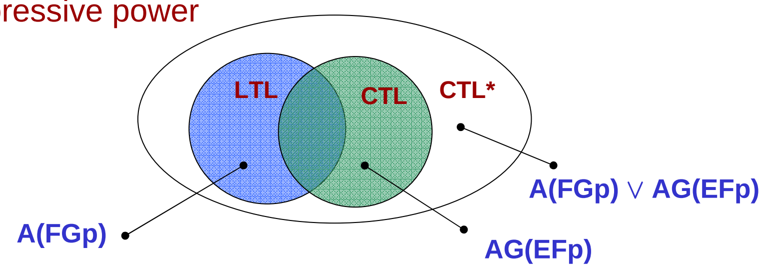
- Path Quantifiers (from a state)
  - **A** : For all computation paths (universal quantification)
  - **E** : There exists a computation path (existential quantification)
- CTL (computation tree logic)
  - a temporal operator *must* be preceded by a path quantifier

AG(a ∨ b) AF(b) EF(cb) EX(EG(cb))



LTL and CTL*

- LTL (linear temporal logic)
  - includes only path formulas
  - applies to all paths starting from initial states (implicit A before the path formulas)
- CTL*
  - negations, conjunctions and disjunctions of CTL and LTL formulas (sufficient set: ¬, ∨, **X**, **U**, **E**)
- Expressive power



CTL Model Checking

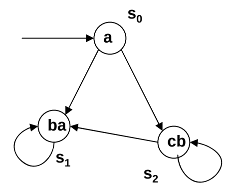
Problem: Given a TS T and a CTL formula f, determine if *f* is true for all initial states Q_0.

Solution: Compute *predicate* $P_f = \{ q \in Q \mid q \vDash f \}$ and see if $Q_0 \subseteq P$.

- *Symbolic Model Checking*
  - identify basic CTL operators with greatest fixpoint (gfp) or least fixpoint (lfp) of predicate transformers
  - apply gpf, lfp, and set operations as needed inductively over subformulas of f to obtain P_f

Predicate Transformers and Fixpoints[1]

- predicate: equate CTL expressions with predicates over Q
  - e.g. $P = ab'c' \vee bc = \{s_0, s_2\}$ (where $' \equiv \neg$)



- predicate transformer: $\tau : 2^Q \rightarrow 2^Q$
  - e.g. : $\tau(P) = P \wedge c'$
  - for $P = ab'c' \vee bc$, $\tau(P) = (ab'c' \vee bc) \wedge c' = ab'c' = \{s_0\}$
- fixpoint : $P = \tau(P)$
  - e.g. $P=ab'c'$ is a fixpoint of $\tau(P) = P \wedge c'$

[1] Fixpoint is a shortened form of the more precise term, *fixed point*.

Carnegie Mellon

Greatest and Least Fixpoints

- greatest fixpoint of τ:
  - gfp Z $[\tau(Z)] \triangleq$
    $P \subseteq Q \ni P = \tau(P)$ and if $P' = \tau(P')$, $P' \subseteq P$.
- least fixpoint of τ:
  - lfp Z $[\tau(Z)] \triangleq$
    $P \subseteq Q \ni P = \tau(P)$ and if $P' = \tau(P')$, $P \subseteq P'$.
- τ monotonic $\Rightarrow$ gfp Z $[\tau(Z)]$ and lfp Z $[\tau(Z)]$ exist.

lfp and gfp algorithms

For τ monotonic:

```
function lfp(τ)
    P := false (i.e., ∅)
    P′ = τ(P)
    while (P ≠ P′) do
        P := P′
        P′ := τ(P)
    endwhile
    return(P)
end
```

```
function gfp(τ)
    P := true (i.e., Q)
    P′ = τ(P)
    while (P ≠ P′) do
        P := P′
        P′ := τ(P)
    endwhile
    return(P)
end
```

$P \uparrow$ lfp $Z\,[\tau(Z)]$ $\qquad$ $P \downarrow$ gfp $Z\,[\tau(Z)]$

For Q finite, maximum number of steps = |Q|

17:48

Fixpoint Characterizations for CTL Operators

- AG(p) = gfp Z [p $\wedge$ A**X** Z]
- EG(p) = gfp Z [p $\wedge$ **EX** Z]
- AF(p) = lfp Z [p $\vee$ A**X** Z]
- EF(p) = lfp Z [p $\vee$ **EX** Z]
- A(p_1 U p_2) = lfp Z [$p_2 \vee (p_1 \wedge$ A**X** Z)]
- E(p_1 U p_2) = lfp Z [$p_2 \vee (p_1 \wedge$ **EX** Z)]

Intuitively:
gfp corresponds to properties that should always hold, lfp corresponds to eventualities.

18:48

ACTL: Universal Properties

- When approximations are used to prove properties of a system (*abstractions* or *simulations*), only universal properties can be shown (properties true for all paths in the computation tree).
- ACTL $\triangleq$ CTL with
  - only universal path quantification (**A:** for all paths)
  - negations applied only to atomic propositions to avoid implicit existential path quantification (i.e., $\neg$ A is not permitted)

Reachability

Specification: No "bad states" are reached.

Solution: Atomic proposition b $\triangleq$ bad state, f = AG($\neg$b).

- $Q_0 \overset{?}{\subseteq} AG(\neg b) = \text{gfp } Z\, [\neg b \wedge A\mathbf{X}\, Z]$

```
function gfp(τ)
    P := Q
    P′ = ¬b ∧ AX P
    while (P ≠ P′) do
        P := P′
        P′ := ¬b ∧ AX P
    endwhile
    return(P)
end
```

$\{q \in Q \mid \neg b \wedge Post(q) \cap P = P\}$

"backward" reachability: eliminates all paths to bad states one transition at time.

Alternative Solution: Forward Reachability

P := Q_0
while true do
  if P $\cap$ $\neg$b return "unsafe"
  if Post(P) $\subseteq$ P return "safe"
  P := P $\cup$ Post(P)
end while

This is the approach used by *explicit state* model checkers.

Outline

- Verification of Transition Systems
- Simulation and Bisimulation
- Application to Hybrid Systems

Simulation Relations

Def. T_2 simulates T_1 ($T_2 \succeq T_1$) if there is a simulation relation between T_1 and T_2.

$T_i = (Q_i, \rightarrow_i, Q_{i0}, \mathcal{L}, L_i)$, $i = 1, 2$. $\psi \subseteq Q_1 \times Q_2$ is a simulation relation between T_1 and T_2 if:

i. $\forall\, q_{10} \in Q_{10},\ \exists\, q_{20} \in Q_{20} \ni (q_{10}, q_{20}) \in \psi$

(each initial state in T_1 has a corresponding initial state in T_2)

ii. if $(q_1, q_2) \in \psi$

a. $L_1(q_1) = L_2(q_2)$ (corresponding states have the same labels)

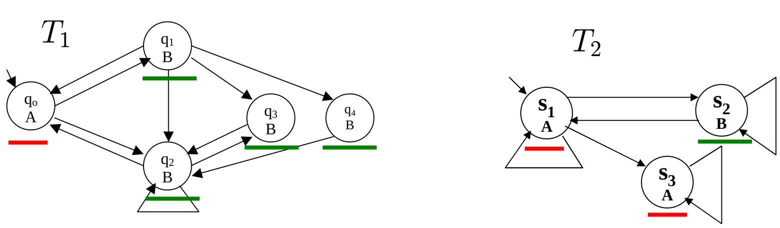
b. $q_1 \rightarrow_1 q_1' \Rightarrow \exists\, q_2' \in Q_2 \ni q_2 \rightarrow_2 q_2' \wedge (q_1', q_2') \in \psi$

(each transition in T_1 has a corresponding trasition in T_2)

Simulation & Path Correspondence

Proposition. If $T_2 \succeq T_1$, then for any path π_1 in T_1 there exists a corresponding path π_2 in T_2, where π_2 depends on the particular simulation relation ψ between T_1 and T_2.

$$\pi_1 = q_0 q_1 q_2 \ldots \text{ corresponds to } \pi_2 = q_0' q_1' q_2' \ldots \iff \forall\, i = 0, 1, 2, \ldots (q_i', q_i') \in \psi.$$



$$\psi = \{(q_o, s_1), (q_o, s_3), (q_1, s_2), (q_2, s_2), (q_3, s_2), (q_4, s_2)\}$$

Note: Corresponding paths have the same label sequence.

Application of Simulation

If $T_2 \succeq T_1$:

- ACTL properties (*universal properties*) true for the set of *all* paths in for T_2 are true for all label sequences for T_1.

Why do we care?

- it may be easier to check an ACTL property for T_2 than for T_1 (especially if T_2 has a *finite* number of states and T_1 has an *infinite* number of states!)

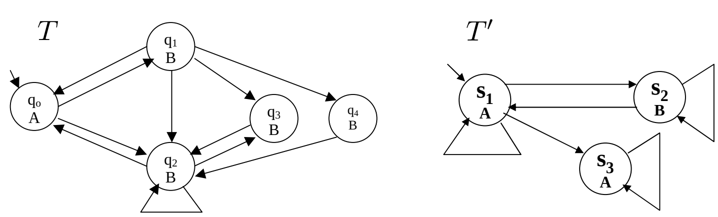
Basic approach to verification: Given a TS T_1 and an ACTL property p, construct a TS $T_2 \succeq T_1$ for which p can be checked efficiently.

Note: An ACTL property *not* true for T_2 may still be true for T_1 (since T_1 has a smaller set of paths). *Counterexamples* for an ACTL property in T_2 (paths violating the property) that satisfy the property for T_1 are called *spurious counterexamples* for T_1

25:48

Verification Using Simulation: Example

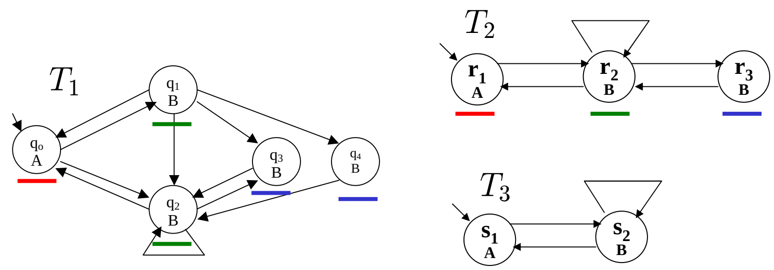
- ACTL property true for $T' \Rightarrow$ true for T
  - e.g., **AG**(**AX**(A∨B)), A or B is always true in the next state
- universal property *not* true for T' may be true for T
  - e.g., **AF**(B), B is always eventually true



26:48

Bisimulation

T_1 and T_2 are bisimulation equivalent (denoted $T_1 \equiv T_2$) if $T_1 \succeq T_2$ and $T_2 \succeq T_1$.

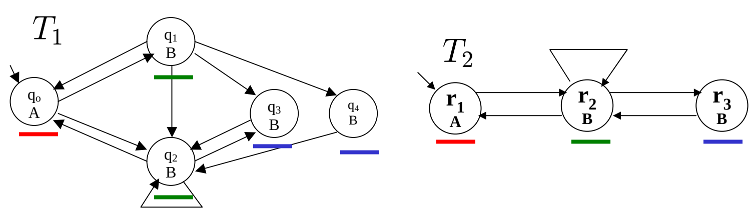


$T_1 \equiv T_2$ but $T_1 \not\equiv T_3$ (Why?)

Bisimulation

Bisimulation equivalance is established by the existence of a bisimulation relation $B \subseteq Q_1 \times Q_2$, where B is a simulation relation between T_1 and T_2 and B^{-1} is a simulation relation between T_2 and T_1.

E.g. $B = \{(q_o, r_1), (q_1, r_2), (q_2, r_2), (q_3, r_3), (q_4, s_3)\}$.



Application of Bisimulation

If $T' \equiv T$:

- CTL properties (*universal* & *existential*) are true for T' $\Leftrightarrow$ they are true for T.

Again ...

- it may be easier to check a CTL property for T' than for T (especially if T' has a *finite* number of states and T has an *infinite* number of states!)

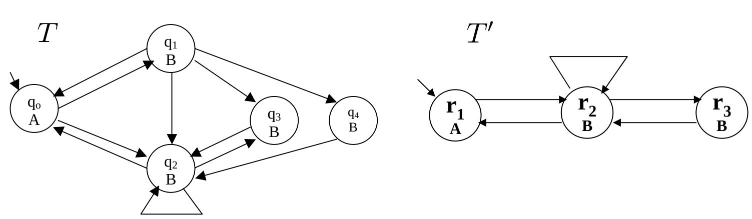
So having a bisimulation is better than simulation, BUT ...

the *basic approach* (finding a *bisimulation* for which verification is efficient) may not be possible.

More to come on this issue.

Verification Using Bisimulation: Example

- existential property true for T' $\Leftrightarrow$ true for T
  - e.g., **EF**(**AX**(¬A)), there exists a state such that A is not true for all next states
- universal property true for T' $\Leftrightarrow$ true for T
  - e.g., **AF**(B), B is eventually true



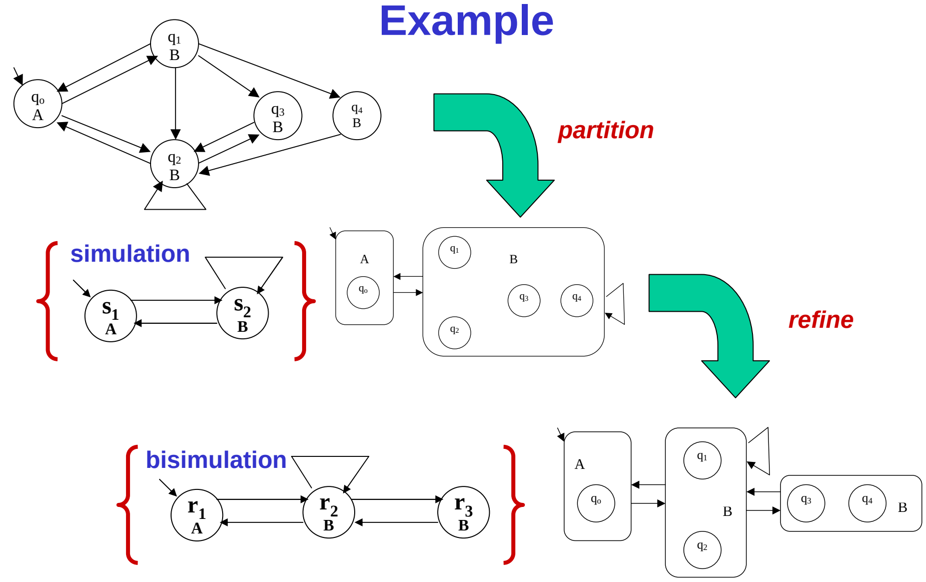
Constructing a Bisimulation: Quotient Transition Systems

The basic idea:

- partition ("quotient") the set of states, grouping states with the same labels (a *consistent* partition)
- construct a transition relation between the partitions reflecting the underlying transition relation
- if necessary, refine the partition until a bisimulation is reached

Note: The quotient transition system (QTS) created on each iteration simulates the original labeled transition system.

Constructing a Bisimulation: Example



Quotient Transition System (QTS)

Given a labeled transition system $T = (Q, \to, Q_0, \mathcal{L}, L)$ and a consistent partition[1] $\mathcal{P}$ of Q, the quotient transition system of T is defined as $T/\mathcal{P} = (\mathcal{P}, \to_{\mathcal{P}}, Q_0/\mathcal{P}, \mathcal{L}, L_{\mathcal{P}})$, where

i. $P \to_{\mathcal{P}} P' \iff \exists\, q \in P, q' \in P' \ni q \to q'$

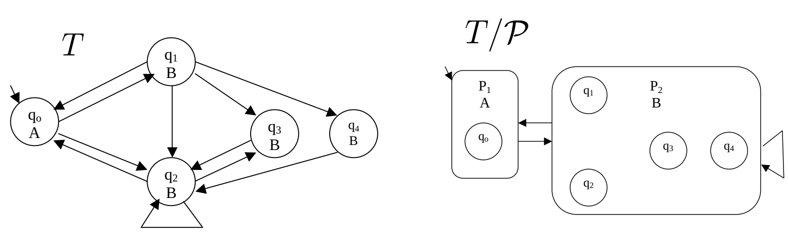
ii. $Q_0/\mathcal{P} = \{P \in \mathcal{P} \mid P \subseteq Q_0\}$

iii. $\forall\, P \in \mathcal{P}$, $L_{\mathcal{P}}(P) = L(q)$ for $q \in P$.

[1] $\mathcal{P}$ is consistent if and only if $\forall\, P \in \mathcal{P}$ and $\forall\, q, q' \in P$, $L(q) = L(q')$ and $q \in Q_0 \iff q' \in Q_0$.

QTS and Simulation

Lemma. Given a consistent partition $\mathcal{P}$, $T/\mathcal{P} \succeq T$.
pf. Let $\psi = \{(q, P) \in Q \times \mathcal{P} \mid q \in P\}$. Suppose $(q, P) \in \psi$ and $q \to q'$. $\mathcal{P}$ is a partition of Q, so $\exists P' \in \mathcal{P} \ni q' \in P'$. $P \to_{T/\mathcal{P}} P'$, since $q \to q'$. Therefore, ψ is a simulation relation between T and $T/\mathcal{P}$.

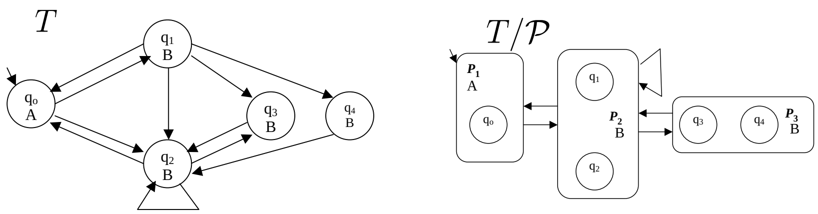


$$\psi = \{(q_o, P_1), (q_1, P_2), (q_2, P_2), (q_3, P_2), (q_4, P_2)\}$$

QTS and Bisimulation

Proposition. $\psi = \{(q, P) \in Q \times \mathcal{P} \mid q \in P\}$ is a *bisimulation* relation between T and $T/\mathcal{P}$ $\iff \forall\ P, P' \in \mathcal{P},\ P \cap Pre(P') \in \{\emptyset, P\}$.

If there is a transition from *any* state in P to P', there is a transition from *every* state in P to P'.



E.g., $P_1 \cap Pre(P_2) = P_1$ and $P_1 \cap Pre(P_3) = \emptyset$.

35:48

Proof of Bisimulation Condition

Proposition. $\psi = \{(q, P) \in Q \times \mathcal{P} \mid q \in P\}$ is a *bisimulation* relation between T and $T/\mathcal{P}$ $\iff \forall\ P, P' \in \mathcal{P},\ P \cap Pre(P') \in \{\emptyset, P\}$.

pf. $\Rightarrow$ Suppose ψ above is a bisimulation between T and $T/\mathcal{P}$. For any $P, P' \in \mathcal{P} \ni P \cap Pre(P') \neq \emptyset$, $P \rightarrow_{T/\mathcal{P}} P'$. Therefore, if $q \in P$ (i.e., $(q, P) \in \psi$), there must be a $q' \ni (q', P') \in \psi$ (i.e., $q' \in P'$) and $q \rightarrow q'$. Therefore, $Pre(P') \supseteq P$.
$\Leftarrow$ We already showed ψ is a simulation relation between T and T/P. Suppose $P \in \mathcal{P}$, $q \in P$, and $P \rightarrow_{T/P} P'$. By the definition of $\rightarrow_{T/P}$, $P \cap Pre(P') \neq \emptyset$. Hence, $P \cap Pre(P') = \emptyset$, which implies $\exists q' \in P' \ni q \rightarrow q'$. Therefore, ψ is a bisimulation between T and $T/\mathcal{P}$.

36:48

Computing Bisimulations

Bisimulation Procedure (BP)

% given an inital consistent partition $\mathcal{P}_0$

$\mathcal{P} := \mathcal{P}_0$

% build the transition relation

$\forall\ P \in \mathcal{P}, Post(P) := \{P' \in \mathcal{P} | Post(P) \cap P' \neq \emptyset\}$

% termination condition

while $\exists\ P, P' \in \mathcal{P} \ni\ \ P \cap Pre(P') \notin \{\emptyset, P\}$

$\{$ % refine partition (split P)

$P_1 := P \cap Pre(P')\ ;\ P_2 := P - Pre(P')$

$\mathcal{P} := (\mathcal{P} - \{P\}) \cup \{P_1, P_2\}$

% update the transition relation

$Post(P_1) := Post(P \cap Pre(P'))$

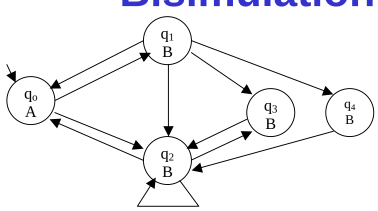
$Post(P_2) := Post(P - Pre(P'))$

$\}$

Note: Context implies wheter $Pre/Post$ operators apply to T or $T/\mathcal{P}$.

37:48

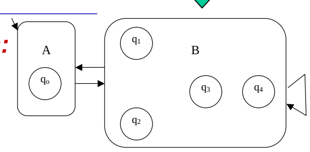
Bisimulation Procedure: Example



initial partition:

$\mathcal{P}_0 = \{Q_0, Q - Q_0\})$

termination condition fails:

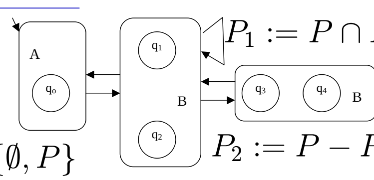


for $P = \{q_1, q_2, q_3, q_4\}$

$P' = \{q_0\}$

$P \cap Pre(P') = \{q_1, q_2\} \notin \{\emptyset, P\}$

refine



$P_1 := P \cap Pre(P')$

$P_2 := P - Pre(P')$

termination condition satisfied:

$\forall\ P, P' \in \mathcal{P},\ \ P \cap Pre(P') \in \{\emptyset, P\}$

38:48

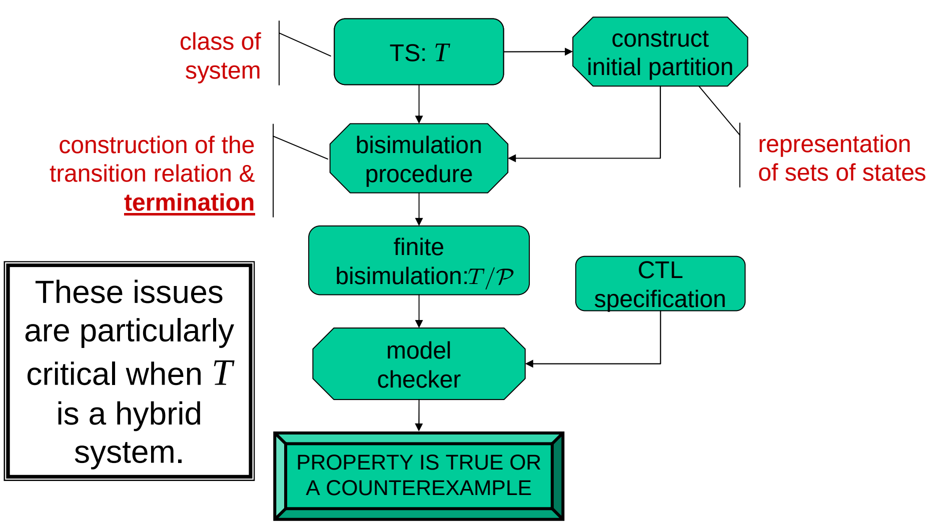
A Sufficient Condition for Bisimulation

Proposition. If $\forall\, P, P' \in \mathcal{P}, Post(P) \cap P' = \{\emptyset, Post(P)\}$, then $T \equiv T/\mathcal{P}$.

If there is a transition from P *to any* state in P', the *all* transitions from P go to P'.

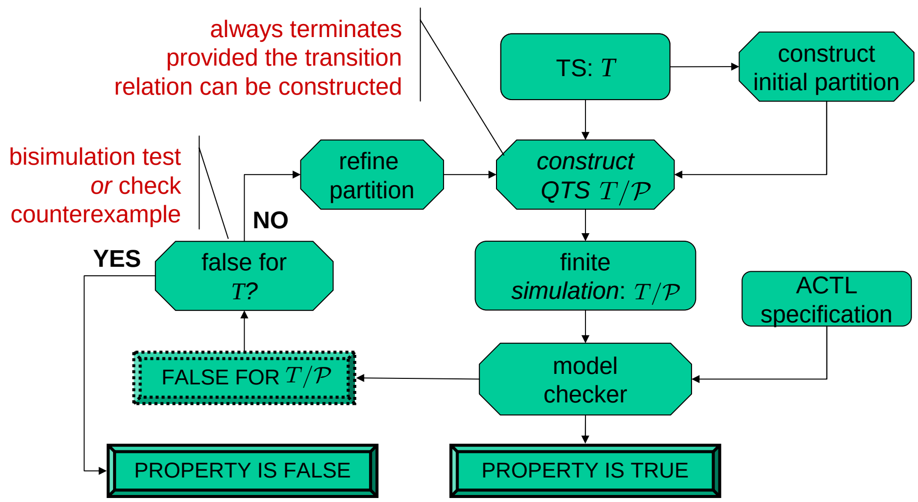
pf. $Post(P) \cap P' = \emptyset \Rightarrow P \cap Pre(P') = \emptyset$;
$Post(P) \cap P' = Post(P) \Rightarrow P \cap Pre(P') = P$.
The result follows from the previous proposition.

Verification Using Bisimulation



CarnegieMellon

Verification Using Simulation



41:48

CarnegieMellon

Outline

- Verification of Transition Systems
- Simulation and Bisimulation
- Application to Hybrid Systems

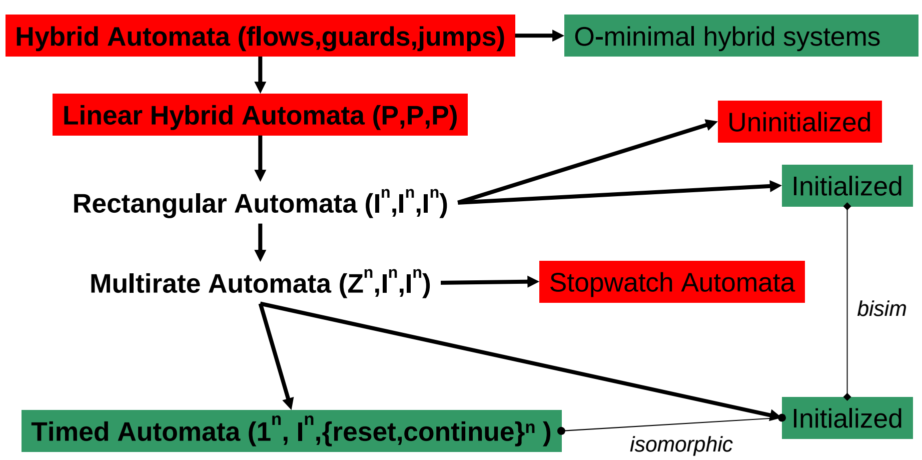
42:48

Applying Model Checking to Hybrid Systems

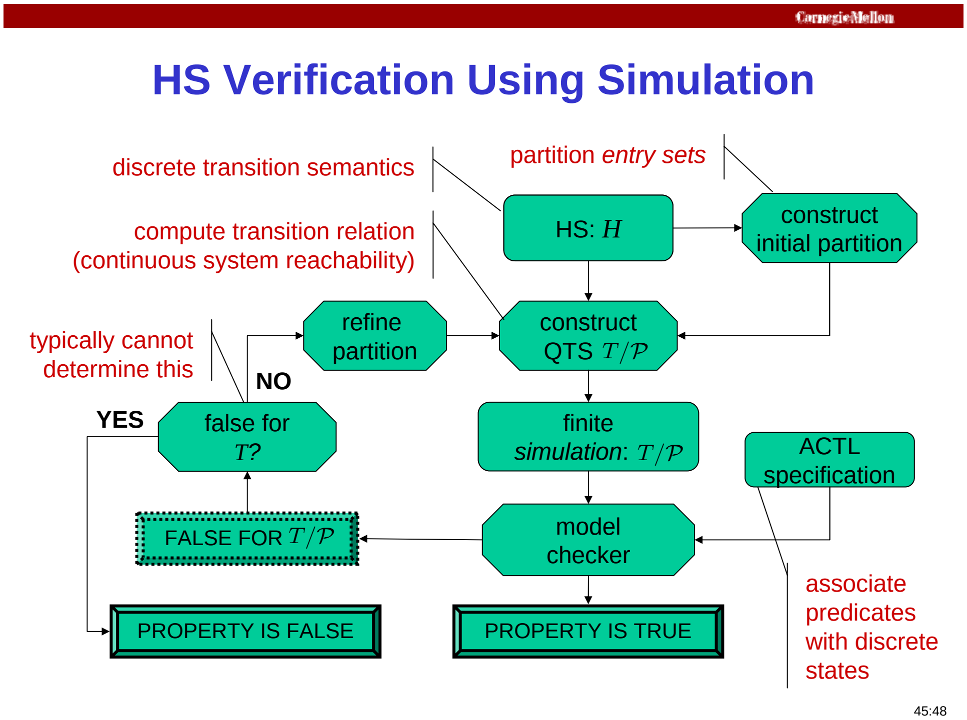
- interpret a hybrid system as a transition system (with an infinite state space)
- compute a finite-state quotient transition system (bisimulation *or* simulation)
- perform model checking on the finite-state system

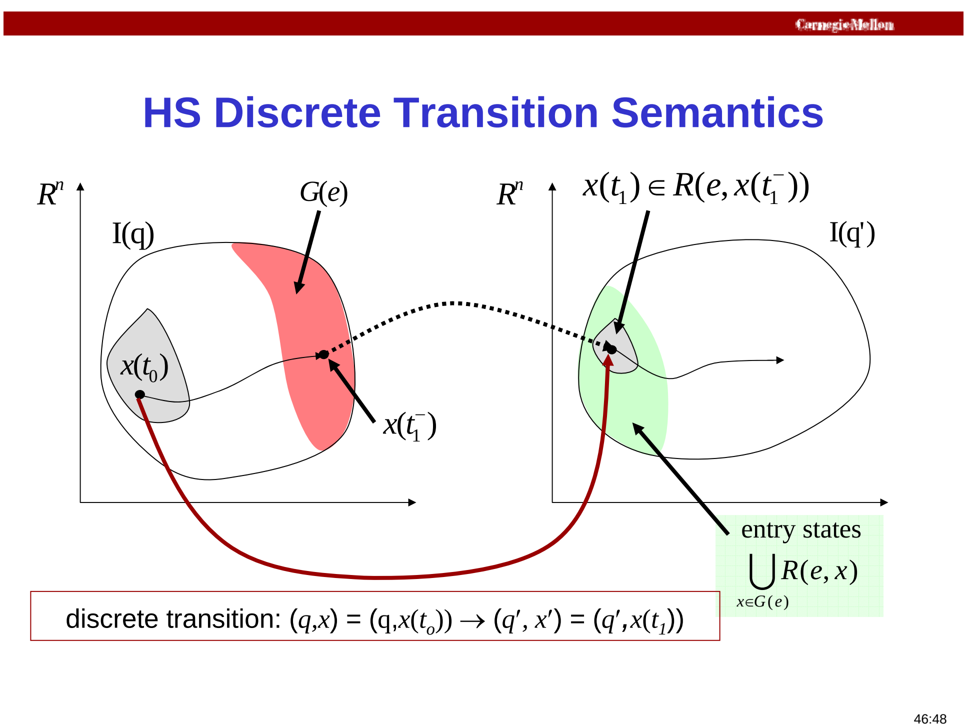
Is this approach feasible?

Termination of BP - Decidability



P = polyhedra, I = intervals, Z = integers, 1 = {1}, reset = {0}


Carnegie Mellon
HS Verification Using Simulation
discrete transition semantics
partition entry sets
compute transition relation
(continuous system reachability)
HS: H
construct
initial partition
typically cannot
determine this
refine
partition
construct
QTS T/𝒫
NO
YES
false for
T?
finite
simulation: T/𝒫
ACTL
specification
FALSE FOR T/𝒫
model
checker
associate
predicates
with discrete
states
PROPERTY IS FALSE
PROPERTY IS TRUE
45:48


Carnegie Mellon
HS Discrete Transition Semantics
R^n
$G(e)$
I(q)
$x(t_0)$
$x(t_1^-)$
R^n
$x(t_1) \in R(e, x(t_1^-))$
I(q')
entry states
$\bigcup_{x \in G(e)} R(e,x)$
discrete transition: $(q,x) = (q,x(t_o)) \rightarrow (q', x') = (q', x(t_1))$
46:48

CarnegieMellon

HS Verification Using Simulation

Primary Issues
- representation of sets of continuous states
- computation of the QTS transition relation
- termination

Next lecture: HS Reachability

47:48

CarnegieMellon

Principal References

E. M. Clarke, O. Grumberg, D. A. Peled, *Model Checking*, MIT Press, 2000.

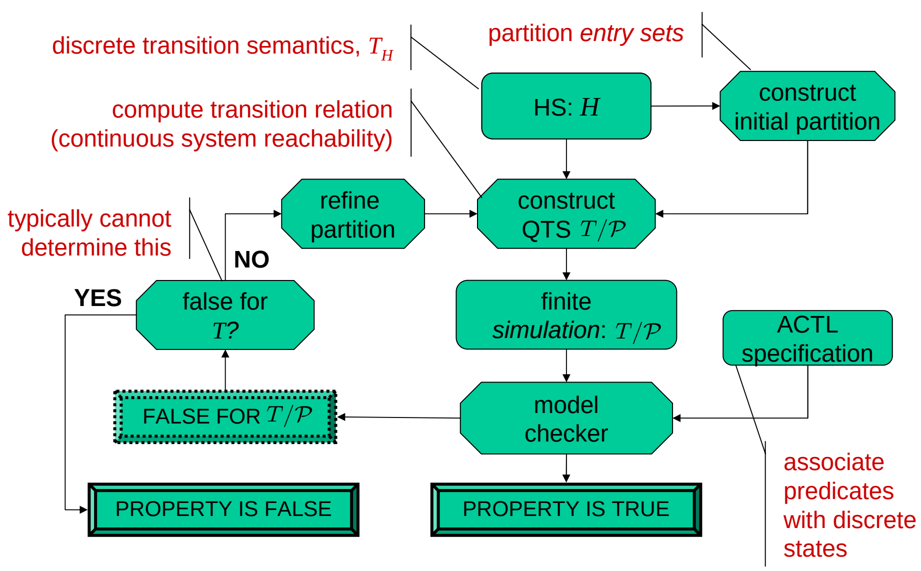
R. Alur, T. A. Henzinger, G. Lafferriere, G. J. Pappas, Discrete abstractions of hybrid systems, *Proceedings of the IEEE*, vol. 88, No. 7, July 2000, pp. 971-984.

48:48

Lecture 2: Hybrid System Reachability

Bruce H. Krogh
Carnegie Mellon University
krogh@ece.cmu.edu

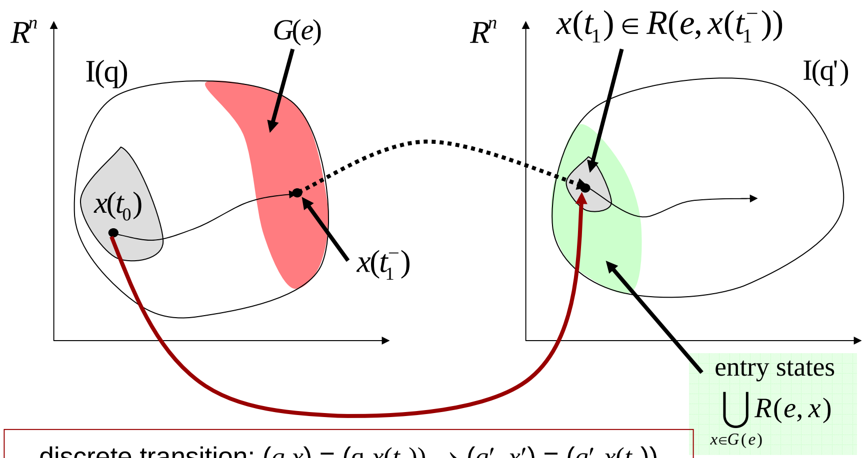
HS Verification Using Simulation



Outline

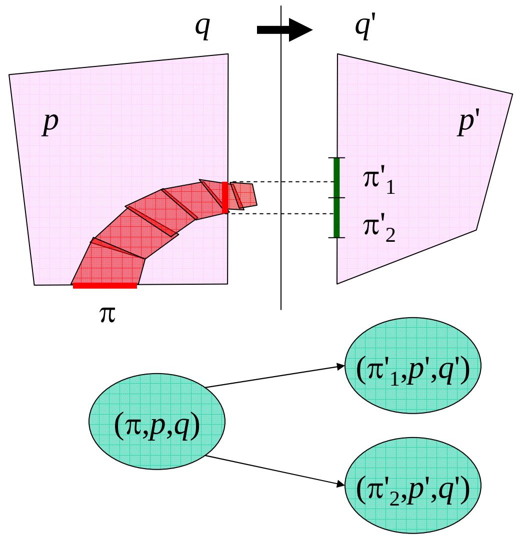
- Polyhedral Approximations
- CheckMate
- Low-Order Representations

HS Discrete Transition Semantics



discrete transition: $(q,x) = (q, x(t_o)) \rightarrow (q', x') = (q', x(t_1))$

Approximating Transitions in T_H/P



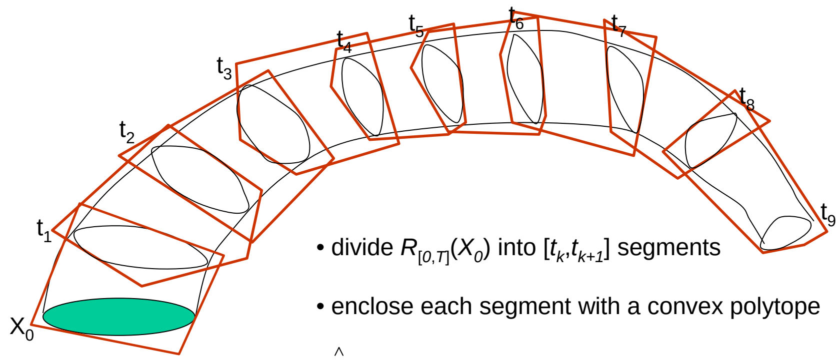
Reachability for Continuous Dynamics

- Given a continuous dynamic system,

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

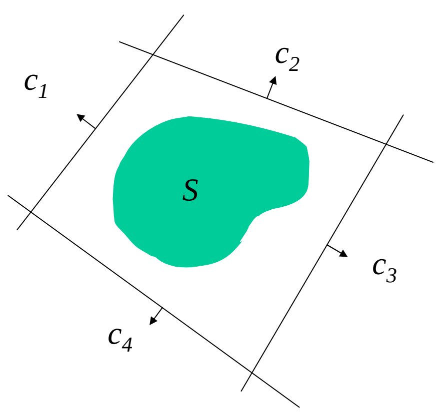
  and a set of initial states, X_0,
- conservatively approximate the set of reachable states $R_{[0,T]}(X_0)$ from time $t = 0$ to $t = T$.

Polyhedral Flow Pipe Approximations



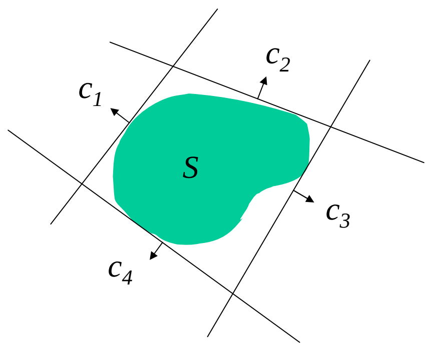
- divide $R_{[0,T]}(X_0)$ into $[t_k, t_{k+1}]$ segments
- enclose each segment with a convex polytope
- $\hat{R}_{[0,T]}(X_0)$ = union of polytopes

Wrapping Hyperplanes Around a Set (1)



Step 1:
- Choose normal vectors, $c_1,\ldots,c_m$

Wrapping Hyperplanes Around a Set (2)



Step 2:

- Adjust each hyperplane so that it just touches *S*
- By solving for each *i* optimization problem

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

Wrapping a Flow Pipe Segment

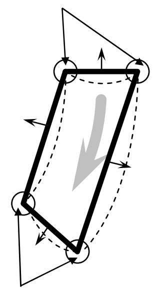
- Given normal vectors c_i, "shrink wrap" $R_{[t_k, t_{k+1}]}(X_0)$ in a polytope by solving for each *i*

$$\begin{aligned} d_i = \quad & \max_{x_0, t} \quad c_i^T x(t, x_0) \\ & s.t. \quad x_0 \in X_0 \\ & \qquad\quad t \in [t_k, t_{k+1}] \end{aligned}$$

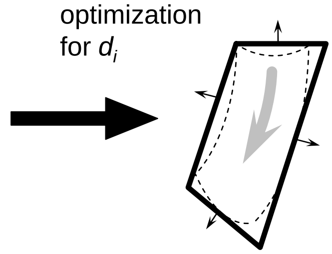
- Embed simulation into objective function computation routine

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
$\{ x \mid c_i^T x \le d_i, \forall i \}$

Example 1: Van der Pol Equation

Van der Pol Equation

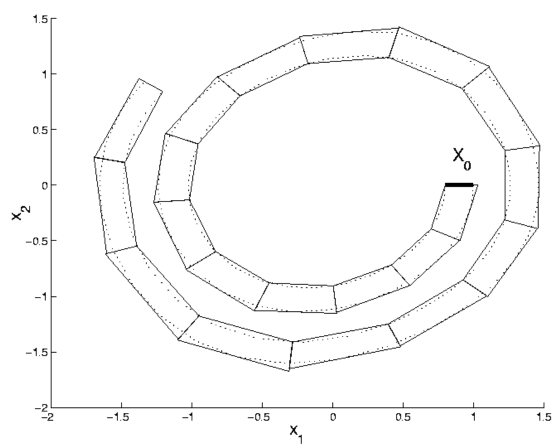
$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -0.2(x_1^2 - 1)x_2 - x_1$$

Initial Set

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

Uniform time step

$\Delta t_k = 0.5$



Improvements for Linear Systems

- $\dot{x} = Ax + b$ $\Rightarrow$ analytical solution

$$x(t, x_0) = e^{At} x_0 + e^{At} \int_0^t e^{-A\tau} b d\tau$$

- Flow pipe segment computation depends only on time step Δt
- A segment can be obtained by applying affine transformation to another segment with the same Δt

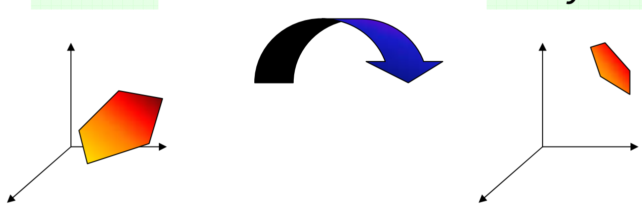
$$\hat{R}_{[t,t+\Delta t]}(X_0) = e^{At} \hat{R}_{[0,\Delta t]}(X_0) + e^{At} \int_0^t e^{-A\tau} b d\tau$$

- No longer need to embed numerical integration into optimization

Transforming a Polytope

Polytope P | Polytope $TP + v$

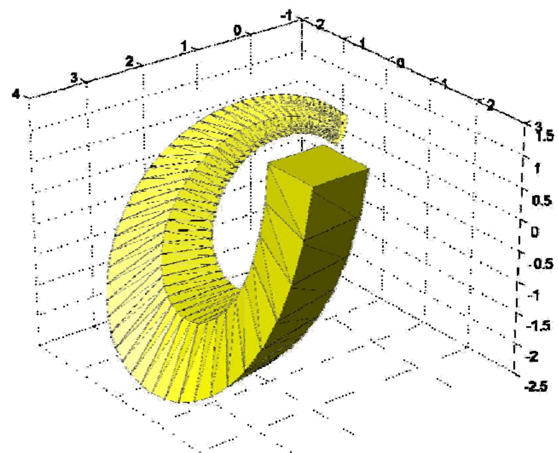
$Cx \le d$ | $y = Tx+v$ | $CT^{-1}y \le d + CT^{-1}v$



$$\hat{R}_{[t,t+\Delta t]}(X_0) = e^{At} \hat{R}_{[0,\Delta t]}(X_0) + e^{At} \int_0^t e^{-A\tau} b d\tau$$

T, P, v

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
$\Delta t_k = 0.1$

- Compute first segment
- Then transform it with $e^{A\Delta t}$ 49 times

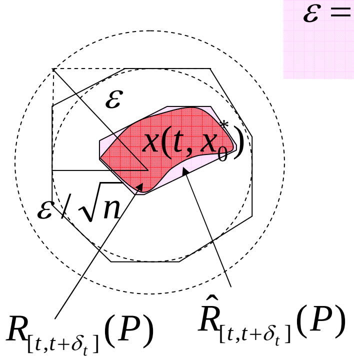
15:74

Approximation Error

- Can be made arbitrarily small for each segment

$$dist(\hat{R}_{[t,t+\delta_t]}(P), R_{[t,t+\delta_t]}(P)) \le \varepsilon$$

$$\varepsilon = \sqrt{n}\left(\frac{\|f(x(t,x_0^*))\|}{L}\left(e^{L\delta_t}-1\right)+e^{L(t+\delta_t)}\delta_{x_0}\right)$$



- Time step
- Size of X_0
- *Lipschitz* constant
- Vector field
- Dimension

16:74

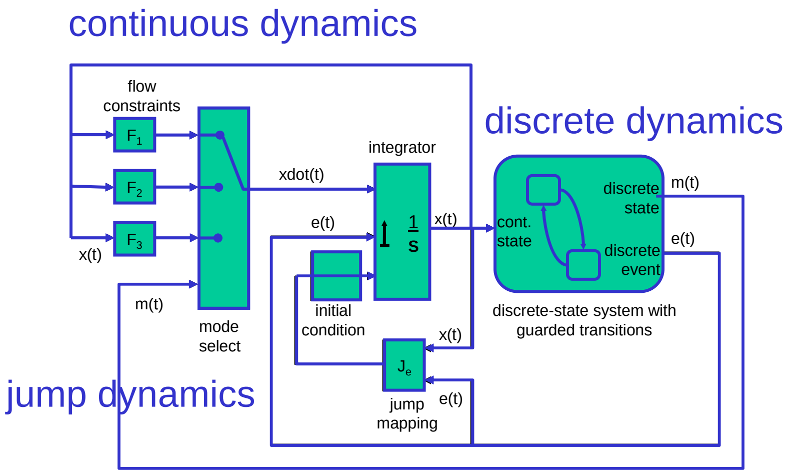
Flow Pipe Approximation

- Applies in arbitrary dimensions
- Approximation error does not accumulate from previous time step
- Approximation error can be made arbitrarily small by bounds
  - δ_t - size of segment time step
    - independent of the starting time for the segment
  - δ_{x0} - size of initial set partition
    - depends on the starting time for the segment

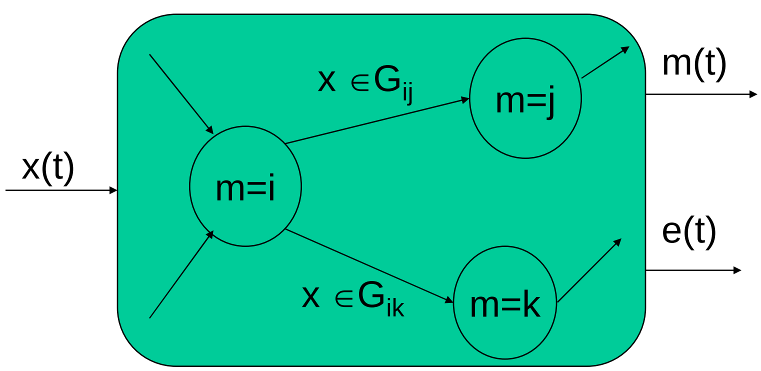
Outline

- Polyhedral Approximations
- CheckMate
- Low-Order Representations

Simulink Diagram of Hybrid System Dynamics



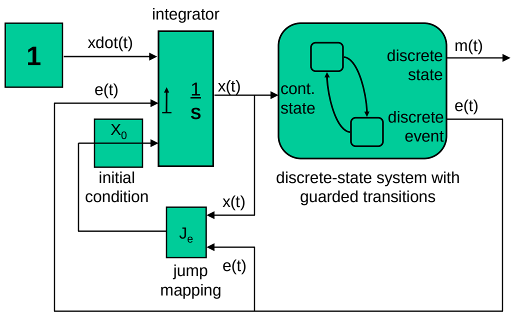
Discrete Transiation Guards



- forced vs. unforced transitions
- implied invariants for discrete states

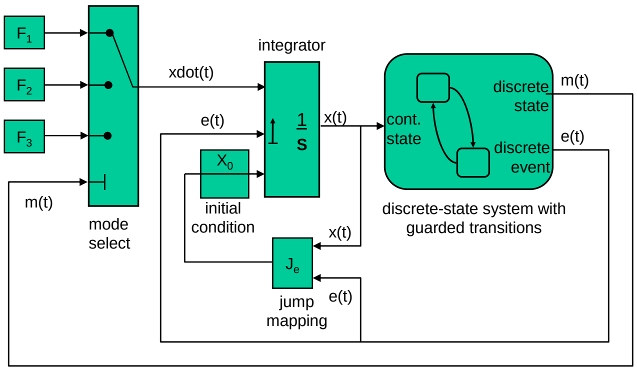
Timed Automata

- continuous dynamics = clocks
- guards are independent intervals on clock values
- jump conditions usually let clocks run or reset to zero

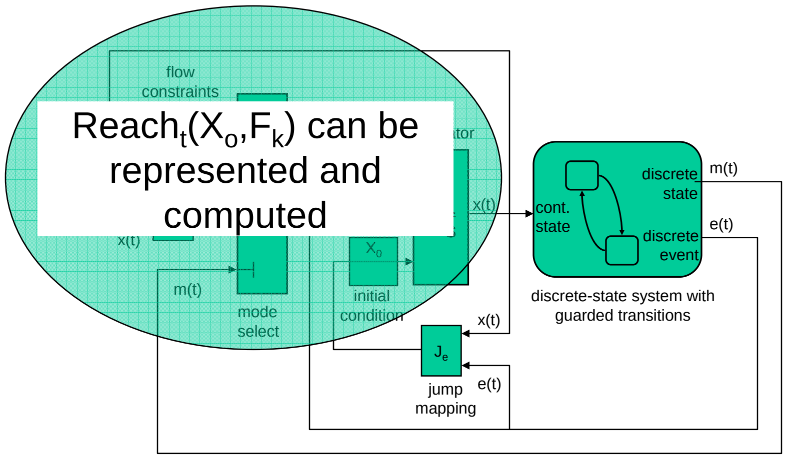


Linear Hybrid Automata

- F_k (flow constraints), J_e (jump mappings), and G_{jk} (guards) are convex polyhedra
- F_k are independent of x(t)

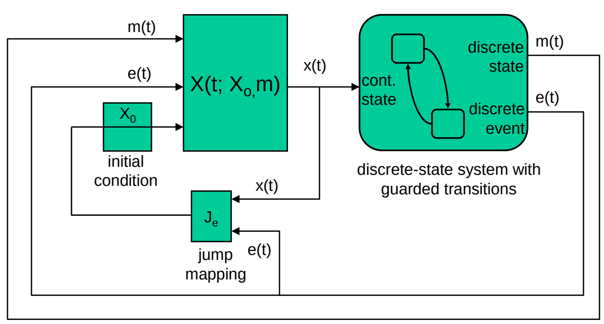


Piecewise-Trivial Hybrid Systems[1]



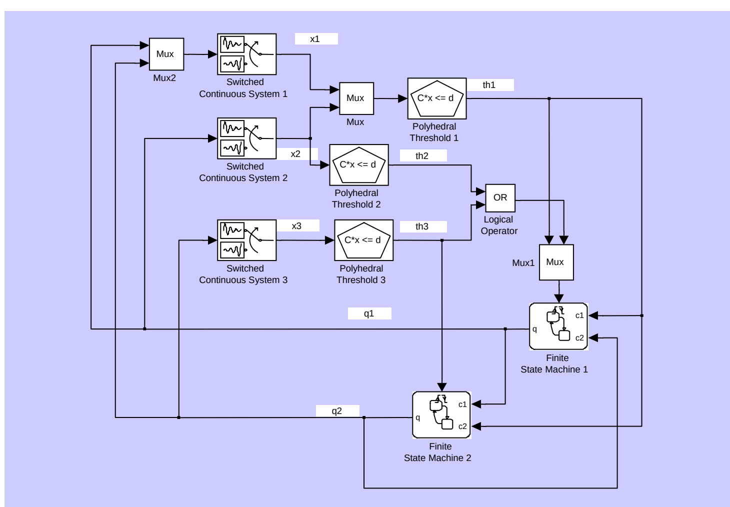
[1]Dang & Maler, HS'98

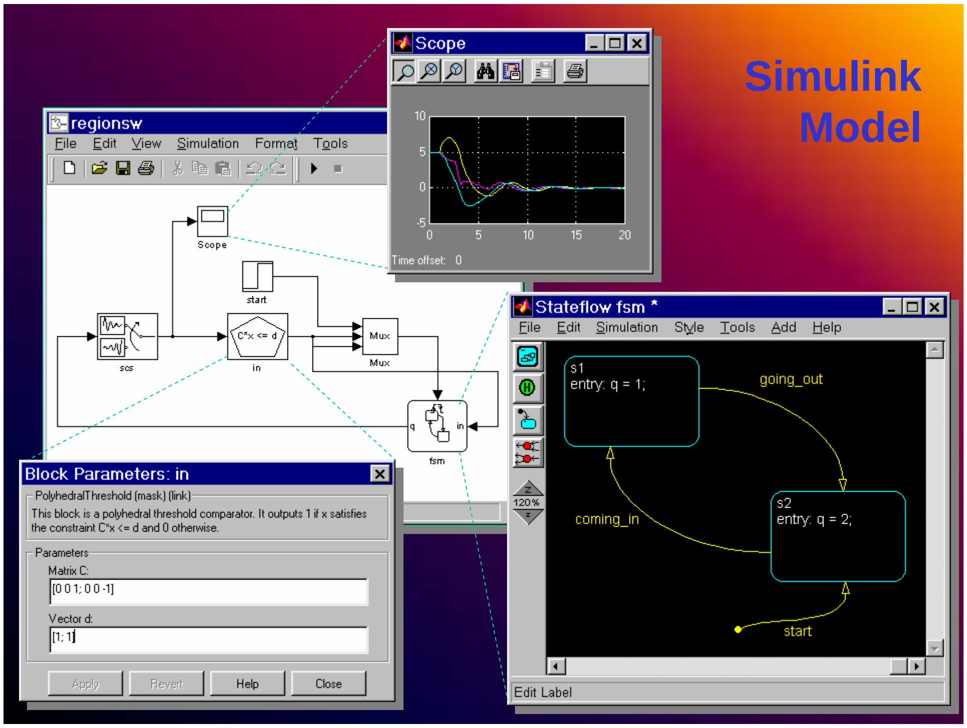
Piecewise-Trivial Hybrid Systems (PTHS)



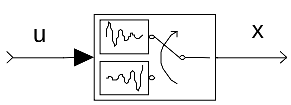
www.ece.cmu.edu/~webk/checkmate/

CheckMate Block Diagram





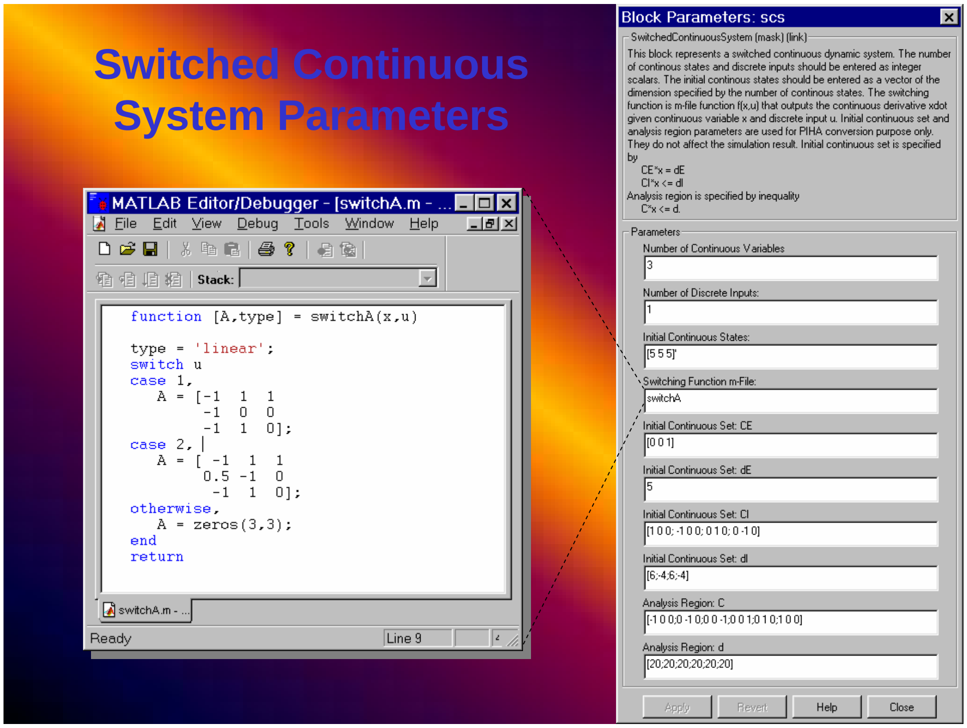
Switched Continuous System



Switched
Continuous System

- **Parameter:** Switching function *f*
- **Input:** Discrete condition signal *u*
- **Output:** Continuous state vector *x*
- **Description:** Continuous dynamics selected by discrete input signal

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

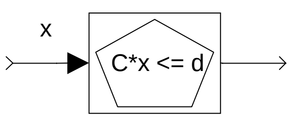


Polyhedral Threshold

- **Parameters:** *C*,*d*
- **Input:** Continuous state vector *x*
- **Output:** Boolean signal

$$\begin{cases} 1 \text{ if } Cx \le d \\ 0 \text{ otherwise} \end{cases}$$

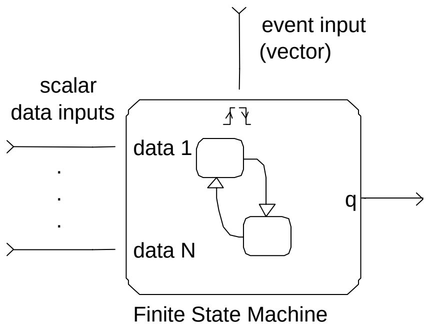
- **Description:** Outputs Boolean signal indicating whether continuous state variable x is in polyhedron $Cx \le d$



Polyhedral Threshold

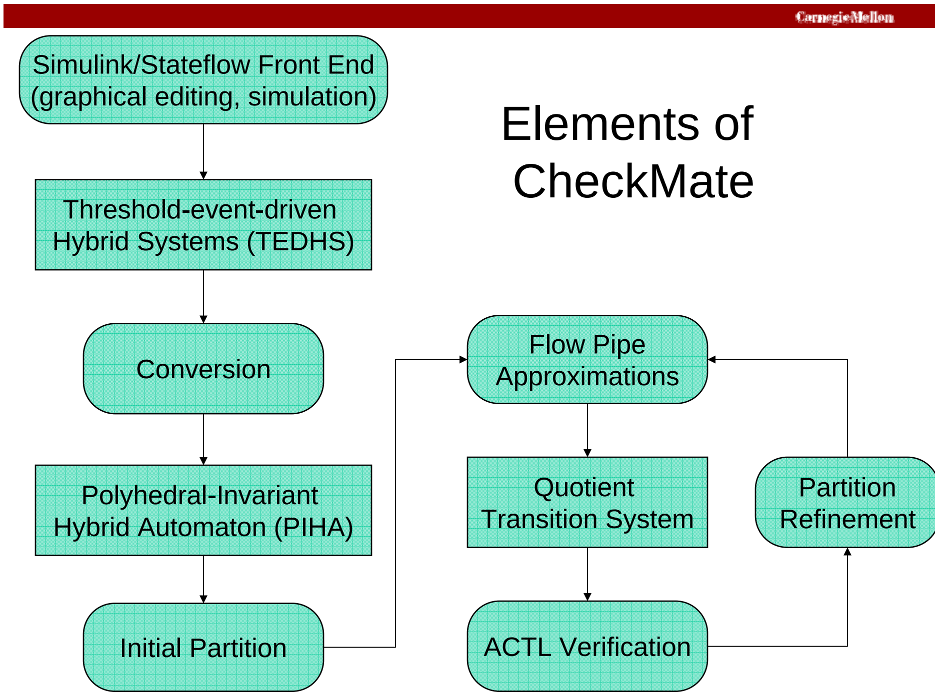
30:74

Finite State Machine (Stateflow)



- **Inputs:**
  - *Data:* Boolean condition signals, functions of PTHB and FSMB outputs
  - *Event:* Transition edges of Boolean condition signals, are functions of PTHB outputs
- **Output:** Discrete signal (integer) indicating active state of FSM

Elements of CheckMate



CheckMate Application: Automotive Engine Control in Cut-off Mode

Control law: Decide when to inject air/fuel for torque to minimize acceleration peaks during the cut-off operation.

Problem: Verify the event-driven implementation of a control law designed in continuous time.

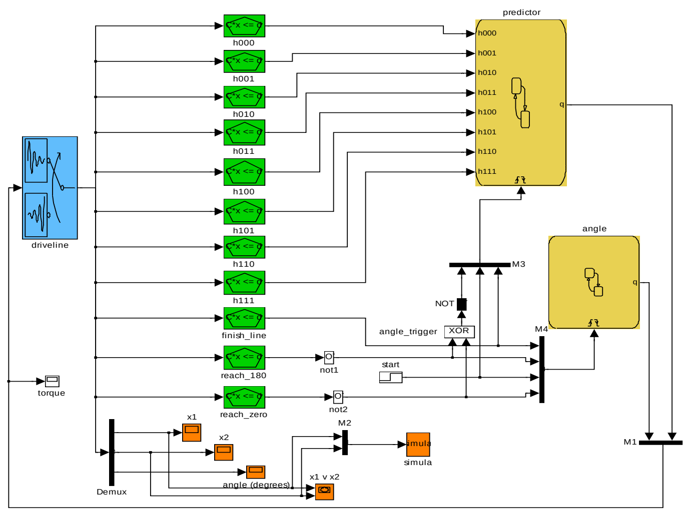
A. Balluchi et. al, Hybrid control in automotive applications: the cut-off control *Automatica* Special Issue on Hybrid Systems, vol. 35, no. 3, March 99; and CDC 97.

Automotive Powertrain Model

Model from Magneti Marelli Engine Control Division

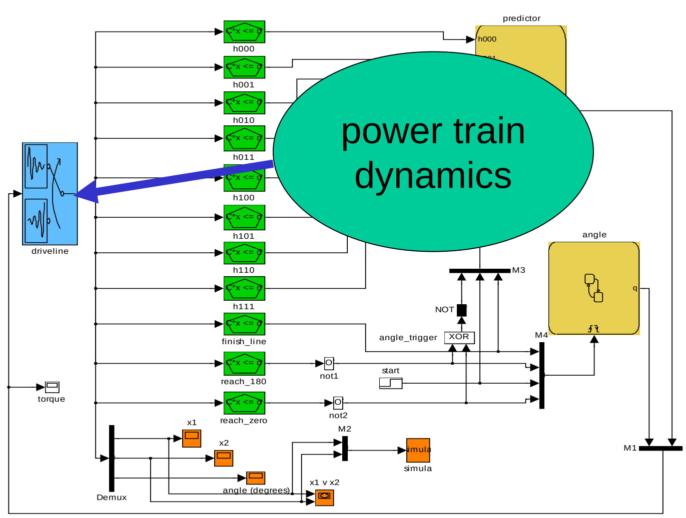
- Four-stroke, four cylinder engine
- Continuous-time powertrain model
- Hybrid model for cylinder cycles

CheckMate Model



35:74

CheckMate Model



36:74

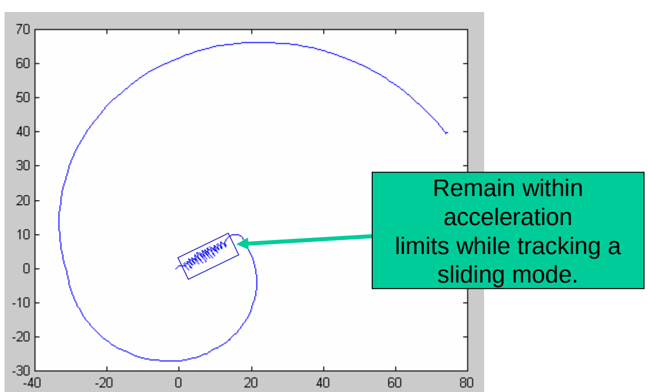
Continuous Dynamics - Initial Model

$\dot{x} = Ax + Bu$ u = 0 (no air-fuel) or 10

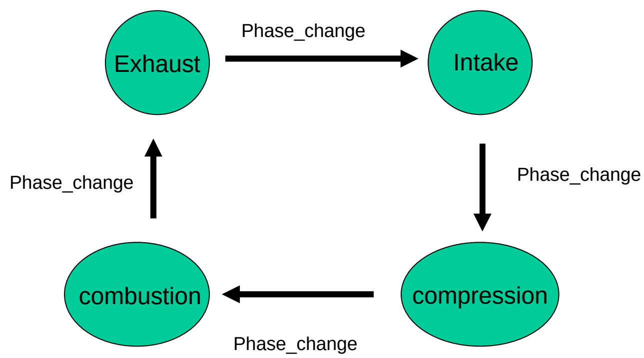
x_1 = engine block angle
x_2 = wheel revolution speed (radians)
x_3 = axle torsion angle (in radians)
x_4 = crankshaft revolution speed (rpm)
x_5 = crankshaft angle (degrees)

Controller Specification

- Sliding mode control law derived in continuous time
- Hybrid implementation due to discrete torque decisions



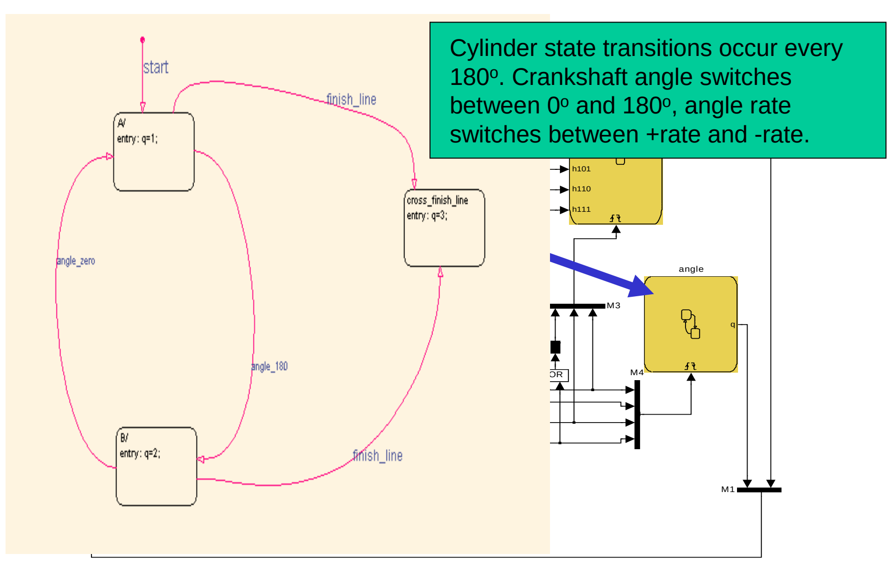
Cylinder Cycle



Control decision to apply torque on the power stroke must be made before the intake stroke ⇒ three step lookahead.

39:74

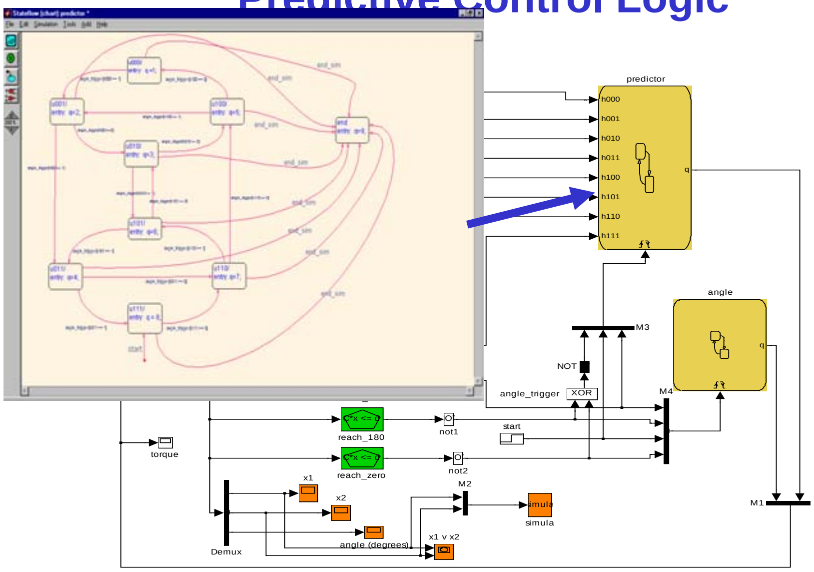
Crankshaft Angle Rate Logic



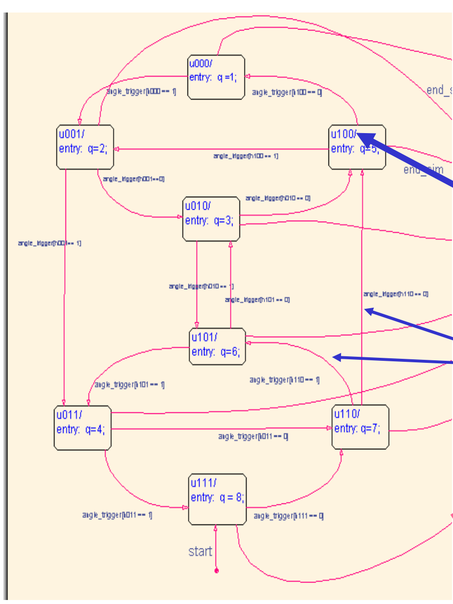
Cylinder state transitions occur every 180º. Crankshaft angle switches between 0º and 180º, angle rate switches between +rate and -rate.

40:74

Predictive Control Logic



41:74



Predictive Control Logic

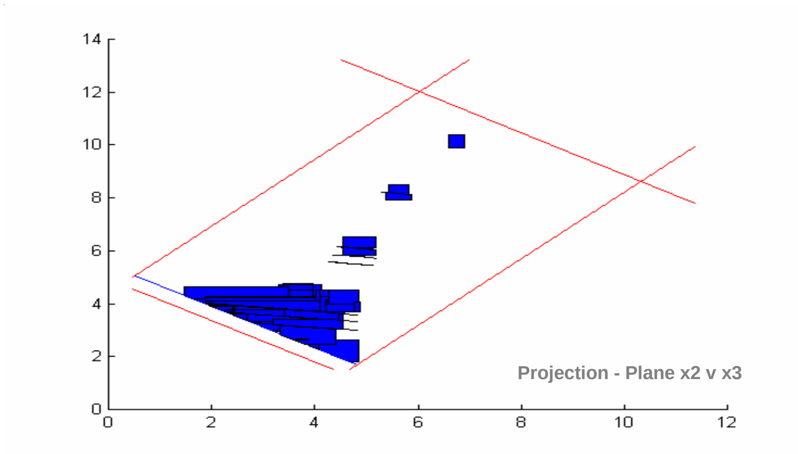
The discrete state indicates the torque decisions for the current and next two power strokes (i.e., for three of the four cylinders).

Transitions from each state depend on whether predicted state for the next power stroke is closer to the sliding mode with or without torque.

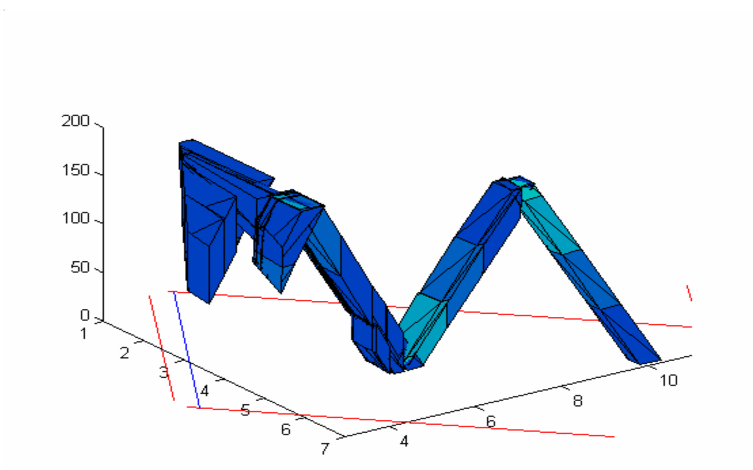
The 9th state (not shown) is the “end simulation” state--reachable from any of the other 8 states.

42:74

Reachable States in T^M/P



Flowpipe for One Discrete Sequence



Outline

- Polyhedral Approximations
- CheckMate
- Low-Order Representations

Reachability Analysis for Affine Systems

Objective: Use low-dimensional polytopes to compute the reach set for affine dynamic systems

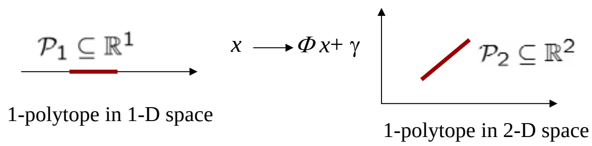
$$\dot{x} = Ax + b,\ x(0) \in \mathcal{X}_0 \text{ and } dim(\mathcal{X}_0) \ll n$$

Affine Representations for Polytopes

A d-polytope in $\mathbf{R}^n$ is the image of d-polytope in $\mathbf{R}^d$ via an affine mapping $\mathbf{R}^d \rightarrow \mathbf{R}^n : x \rightarrow \Phi x + \gamma$

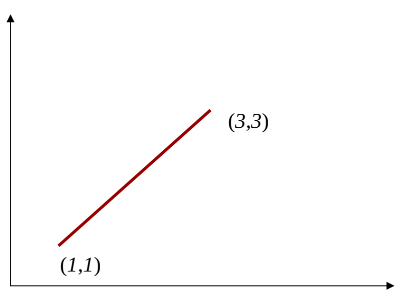
$$\mathcal{P}_n = < \Phi, \gamma, \mathcal{P}_d > \subseteq \mathbb{R}^n$$
$$:= \{x | x = \Phi w + \gamma,\ w \in \mathcal{P}_d\},$$

$\mathcal{P}_d$ is full-dimensional,
$\mathbf{0} \in \mathcal{P}_d$ and $\forall w \in \mathcal{P}_d : \|w\| \leq 1$



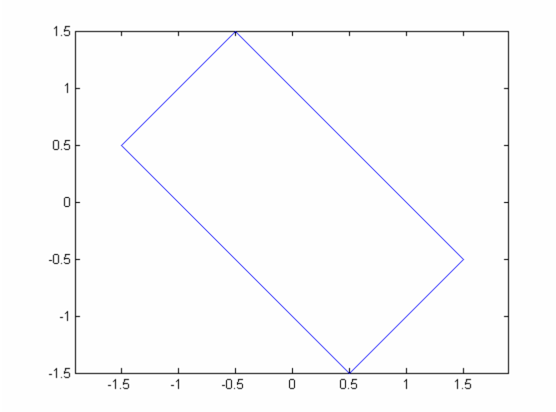
Example 1. Line Segment

(1-D polytope)



$$\langle \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}2\\2\end{bmatrix}, [-1,1](\text{ a closed interval })\rangle$$
$$= \{x | x = \begin{bmatrix}1\\1\end{bmatrix} w + \begin{bmatrix}2\\2\end{bmatrix}, -1 \leq w \leq 1\}$$

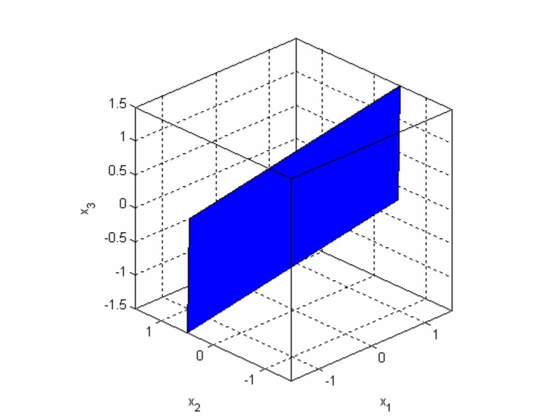
Example 2. Oriented Rectangle (full-dimensional)



$$\langle \begin{bmatrix} 1/2 & 1 \\ 1/2 & -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, CH(\{\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\}) \rangle$$

CarnegieMellon

Example 3. 2-polygon in 3-D



$$\langle \begin{bmatrix} 1/2 & 1 \\ 1/2 & -1 \\ 1/2 & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, CH(\{\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\}) \rangle$$

Computation Using Affine Representations

- Affine transformation:

$$A < \Phi, \gamma, \mathcal{P}_w > +b =< A\Phi, A\gamma + b, \mathcal{P}_w >.$$

- Minkowski sum:

$$< \Phi_1, \gamma_1, \mathcal{P}_{w1} > \oplus < \Phi_2, \gamma_2, \mathcal{P}_{w2} >=$$

$$< \begin{bmatrix}\Phi_1 & \Phi_2\end{bmatrix}, \gamma_1 + \gamma_2, \mathcal{P}_{w1} \otimes \mathcal{P}_{w2} >.$$

Computation Using Affine Representations

- Cartesian product:

$$< \Phi_1, \gamma_1, \mathcal{P}_{w1} > \otimes < \Phi_2, \gamma_2, \mathcal{P}_{w2} >=$$

$$< \begin{bmatrix}\Phi_1 & 0 \\ 0 & \Phi_2\end{bmatrix}, \begin{bmatrix}\gamma_1 \\ \gamma_2\end{bmatrix}, \mathcal{P}_{w1} \otimes \mathcal{P}_{w2} >.$$

- Intersection with halfspace*:

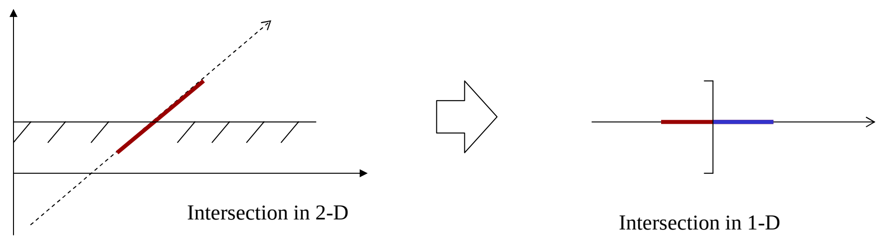
$$< \Phi, \gamma, P(\Pi_w, d_w) > \cap H(\pi^T, d) =< \Phi, \gamma, P(\begin{bmatrix}\Pi_w \\ \pi^T\Phi\end{bmatrix}, \begin{bmatrix}d_w \\ d - \pi^T\gamma\end{bmatrix}) >$$

*$H(\pi^T, d) = \{x | \pi^T x \leq d,\ x \in \mathbb{R}^n\}$, $\pi \in \mathbb{R}^n$ and $d \in \mathbb{R}$

*$P(\Pi, d) = \{x | \Pi x \leq d\} \subseteq \mathbb{R}^n$ is the $\mathcal{H}$-representation of the polytope.

Computation Using Affine Representations

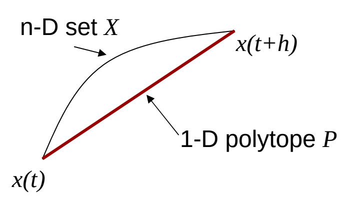
- Intersection with halfspace



Approximate Affine Representation

- If a set is 'close' to low-dimensional…

Consider the case of a segment of trajectory



Since $\mathcal{X} \approx \mathcal{P}$

,i.e., the Hausdorff distance $d(\mathcal{X}, \mathcal{P}) \leq \delta$

then $\mathcal{X} \subseteq \mathcal{P} \oplus \mathcal{B}_\delta$

Denote the set by

$$< \Phi, \gamma, \mathcal{P}_w, \delta > \equiv < \Phi, \gamma, \mathcal{P}_w > \oplus \mathcal{B}_\delta$$

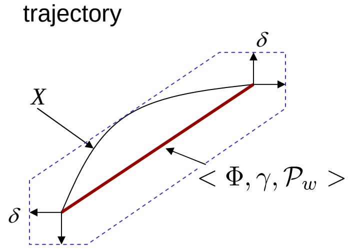
Approximate affine representation

** We consider infinity norms in this work. B_δ is the hyperbox with radius δ.

Approximate Affine Representation

Over-approximate a set by ‘bloating’.

Consider the case of a segment of trajectory



$< \Phi, \gamma, \mathcal{P}_w, \delta >$

Since $\mathcal{X} \approx \mathcal{P}$

,i.e., the Hausdorff distance $d(\mathcal{X}, \mathcal{P}) \leq \delta$

then $\mathcal{X} \subseteq \mathcal{P} \oplus \mathcal{B}_\delta$

Denote the set by

$$< \Phi, \gamma, \mathcal{P}_w, \delta > \equiv < \Phi, \gamma, \mathcal{P}_w > \oplus \mathcal{B}_\delta$$

Approximate affine representation

** We consider infinity norms in this work. B_δ is the hyperbox with radius δ.

Over-approximations With Approximate Affine Representation

Using approximate affine representation, *over-approximations* can be obtained

- Affine transformation: $A < \Phi, \gamma, \mathcal{P}_w, \delta > -b \subseteq < A\Phi, A\gamma + b, \mathcal{P}_w, \|A\|\delta >$.
- Minkowski sum: $< \Phi_1, \gamma_1, \mathcal{P}_{w1}, \delta_1 > \oplus < \Phi_2, \gamma_2, \mathcal{P}_{w2}, \delta_2 > = < \begin{bmatrix} \Phi_1 & \Phi_2 \end{bmatrix}, \gamma_1 + \gamma_2, \mathcal{P}_{w1} \otimes \mathcal{P}_{w2}, \delta_1 + \delta_2 >$.
- Cartesian product: $< \Phi_1, \gamma_1, \mathcal{P}_{w1}, \delta_1 > \otimes < \Phi_2, \gamma_2, \mathcal{P}_{w2}, \delta_2 > \subseteq < \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}, \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix}, \mathcal{P}_{w1} \otimes \mathcal{P}_{w2}, \max\{\delta_1, \delta_2\} >$.
- Intersection with halfspace : $< \Phi, \gamma, P(\Pi_w, d_w), \delta > \cap H(\pi^T, d) \subseteq < \Phi, \gamma, P(\begin{bmatrix} \Pi_w \\ \pi^T \Phi \end{bmatrix}, \begin{bmatrix} d_w \\ d - \pi^T \gamma + \|\Pi\| \delta \end{bmatrix}), \delta >$.

Reach Set Computation Procedure

Input: $\mathcal{S}$, $\mathcal{X}_0$, 0, h, t_f
Output: $Reach([0, t_f])$
Procedure REACH_AFFINE:
$t \leftarrow 0$, $k \leftarrow 1$, $\mathcal{R} \leftarrow \emptyset$
WHILE $t \leq t_f$
    Compute $\mathcal{X}_k$
    Over-approximate the linear interpolation $\mathcal{X}_{k-1,k} \subseteq \mathcal{P}_{k-1,k}$
    Compute $\delta_{k-1,k}$
    $\mathcal{R} \leftarrow \mathcal{R} \cup (\mathcal{P}_{k-1,k} \oplus \mathcal{B}_{\delta_{k-1,k}})$
    $t \leftarrow t + h$
    $k \leftarrow k + 1$
END FOR
RETURN $\mathcal{R}$

initialize

the reach set for the next step

use linear interpolation to approximate the reach segment

proceed to the next step

compute an over-approximation of the reach segment

57:74

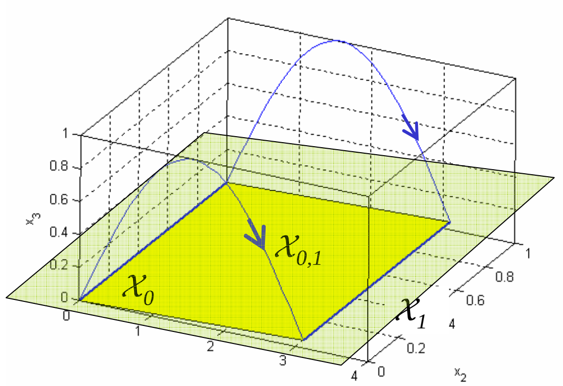
Computing CH(X_{k-1} U X_k)

1. Form the affine subspace containing X_{k-1},X_k.
2. Project the two polytopes onto the affine subspace containing the convex hull.
3. Compute the convex hull in the subspace.

***d* = 1, dim X_{k-1},X_k**

***m* = 2, dim CH**

The convex hull is computed in 2-D.



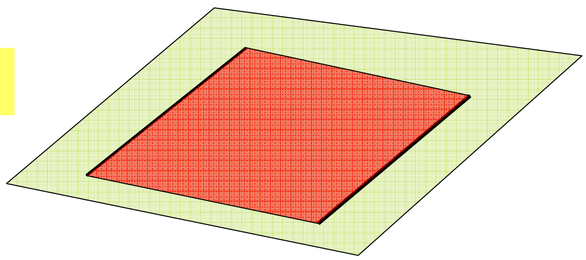
58:74

Computing CH(X_{k-1} U X_k)

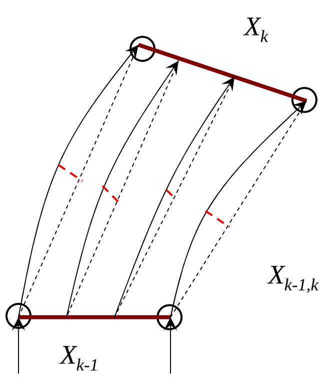
1. Form the affine subspace
2. Project the two polytopes onto the affine subspace containing the convex hull.
3. Compute the convex hull in the subspace.

d = 1,
m = 2

The convex hull is computed in 2-D.



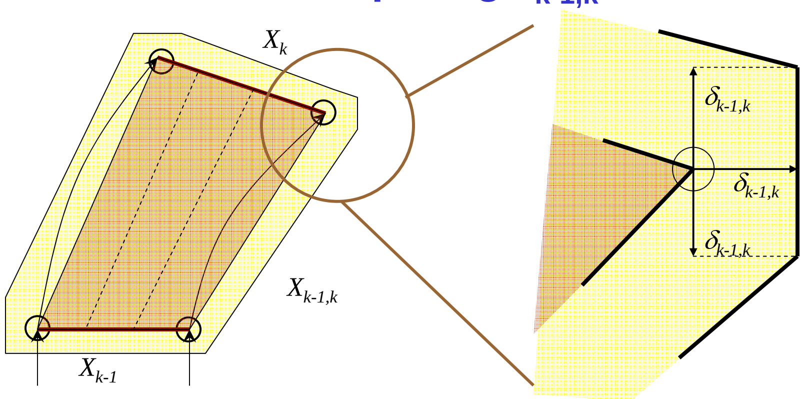
Computing $\delta_{k-1,k}$



•Every trajectory is *approximated* by its linear interpolation.

• $\delta_{k-1,k}$ is computed as an *upper-bound* on the *infinity-norm* of the approximation error of the linear interpolations over the set of trajectories.

Computing $\delta_{k-1,k}$



For $\mathcal{P}_{k-1,k} = \langle \Phi_{k-1,k}, \gamma_{k-1,k}, \mathcal{P}_m \rangle$, its δ-neighborhood *over-approximates* the reach segment.

$N(\mathcal{P}_{k-1,k}, \delta_{k-1,k}) = \mathcal{P}_{k-1,k} \oplus \mathcal{B}_{\delta k-1,k} =: \langle \Phi_{k-1,k}, \gamma_{k-1,k}, \mathcal{P}_m, \delta_{k-1,k} \rangle$

Summary of the Procedure

Input: $\mathcal{S}$, $\mathcal{X}_0$, 0, h, t_f
Output: $Reach([0, t_f])$
Procedure REACH_AFFINE:
$t \leftarrow 0$, $k \leftarrow 1$, $\mathcal{R} \leftarrow \emptyset$
WHILE $t \leq t_f$
  Compute $\mathcal{X}_k$
  Compute low-dimensional polytope $\mathcal{P}_{k-1,k}$ s.t. $\mathcal{X}_{k-1,k} \subseteq \mathcal{P}_{k-1,k}$
  Compute $\delta_{k-1,k}$
  $\mathcal{R} \leftarrow \mathcal{R} \cup (\mathcal{P}_{k-1,k} \oplus \mathcal{B}_{\delta_{k-1,k}})$
  $t \leftarrow t + h$
  $k \leftarrow k + 1$
END FOR
RETURN $\mathcal{R}$

compute matrix functions

convex hull in reduced-order subspace

δ–neighborhood of the convex hull

Handling Large-Scale Systems

- The affine representations for $\mathcal{X}_k$ are computed using
  - $\Phi_k = \varphi_0(A,t)\Phi_0$
  - $\gamma_k = \varphi_0(A,t)\ \gamma_0 + t\varphi_1(A,t)\ b$
- Computing $\boldsymbol{\varphi_0(A,t)\Phi_0}$ and $\boldsymbol{t\varphi_1(A,t)\ b}$ is difficult for *large-scale sparse* systems.
  - Note $\Phi_0 = [\phi_{01}, \phi_{02}, \ldots \phi_{0d}] \in \Re^{n\times d}$ where $d << n$, $\varphi_0(A,t)\Phi_0 = [\varphi_0(A,t)\phi_{01}, \varphi_0(A,t)\phi_{02} \ldots \varphi_0(A,t)\ \phi_{0d}]$

Matrix-vector product

63:74

The Krylov Subspace Approximations

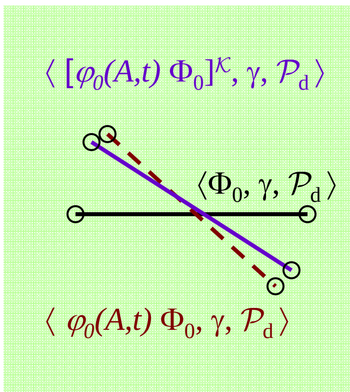
- If we are interested in computing $\varphi_0(A,t)v$ *instead of* $\varphi_0(A,t)$, the Krylov subspace approximation is an efficient way to compute it.

r-dim Krylov subspace = $\text{span}\{v, Av, A^2v, \ldots, A^{r-1}v\}$

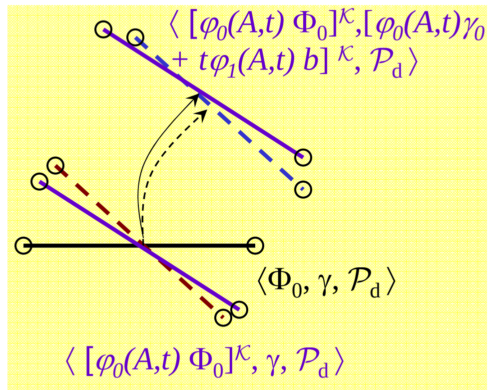
1. Y Saad, Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM Journal of Numerical Analysis*, 20(1) 209-228, 1992.
2. C Moler and C Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1) 3-49, 2003.

64:74

Using Krylov Approximations for the Computations



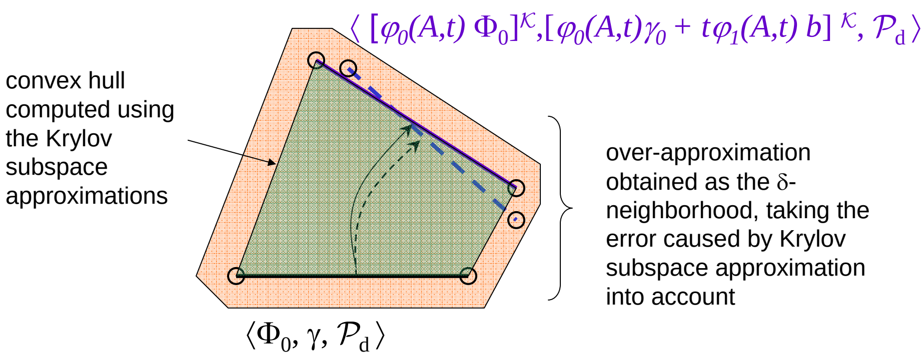
Approximate Linear Transformation



Approximate Displacement

CarnegieMellon

The Error Introduced by Krylov Method



$$\delta = \delta_{k-1,k}^{\mathcal{K}} + \delta_{AE}$$

to over-approximate reach segment

(Accumulated error) to over-approximate actual reach sets using Krylov method

The Computation Procedure

Input: $\mathcal{S}$, $\mathcal{X}_0$, 0, dt, t_f
Output: $Reach([0, t_f])$
ProcedureREACH_AFFINE$^{\mathcal{K}}$:
$t \leftarrow 0$, $k \leftarrow 1$, $\mathcal{R} \leftarrow \emptyset$
WHILE $t \leq t_f$
  IF $size(A) >$MAX_ORDER
    Choose step size h based on the error bound δ_{OSE}
    Compute $\mathcal{X}_k \leftarrow \mathcal{X}_k^{\mathcal{K}}$
  ELSE
    Choose step size $h \leftarrow dt$
    Compute $\mathcal{X}_k$
  END IF
  Compute low-dimensional polytope $\mathcal{P}_{k-1,k}$ s.t. $\mathcal{X}_{k-1,k} \subseteq \mathcal{P}_{k-1,k}$
  Compute $\delta_{k-1,k}$
  IF $\mathcal{X}_k^{\mathcal{K}}$ is used
    Compute δ_{AE} for the Krylov subspace approximation
  END IF
  $\mathcal{R} \leftarrow \mathcal{R} \cup (\mathcal{P} \oplus \mathcal{B}_{\delta_{k-1,k}+\delta_{AE}})$, $t \leftarrow t + h$, $k \leftarrow k + 1$
END FOR
RETURN $\mathcal{R}$

The Computation Procedure

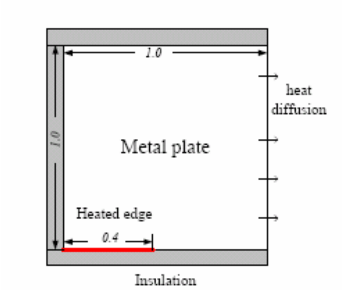
Input: $\mathcal{S}$, $\mathcal{X}_0$, 0, dt, t_f
Output: $Reach([0, t_f])$
ProcedureREACH_AFFINE$^{\mathcal{K}}$:
$t \leftarrow 0$, $k \leftarrow 1$, $\mathcal{R} \leftarrow \emptyset$
WHILE $t \leq t_f$
  IF $size(A) >$MAX_ORDER
    Choose step size h based on the error bound δ_{OSE}
    Compute $\mathcal{X}_k \leftarrow \mathcal{X}_k^{\mathcal{K}}$
  ELSE
    Choose step size $h \leftarrow dt$
    Compute $\mathcal{X}_k$
  END IF
  Compute low-dimensional polytope $\mathcal{P}_{k-1,k}$ s.t. $\mathcal{X}_{k-1,k} \subseteq \mathcal{P}_{k-1,k}$
  Compute $\delta_{k-1,k}$
  IF $\mathcal{X}_k^{\mathcal{K}}$ is used
    Compute δ_{AE} for the Krylov subspace approximation
  END IF
  $\mathcal{R} \leftarrow \mathcal{R} \cup (\mathcal{P} \oplus \mathcal{B}_{\delta_{k-1,k}+\delta_{AE}})$, $t \leftarrow t + h$, $k \leftarrow k + 1$
END FOR
RETURN $\mathcal{R}$

threshold for using Krylov

adaptive step size control via Krylov

bounds on Krylov approximation

Example. 2-D Heat Transfer Problem



(a) A heated metal plate.

Environment: 0 C

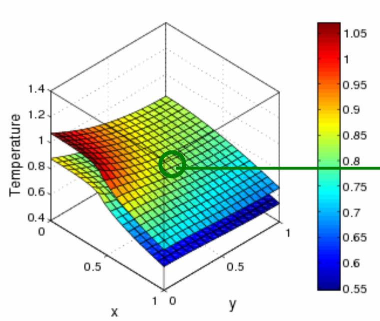
Initial Temp: 0 C

Heated Edge: [0.9,1.1] C

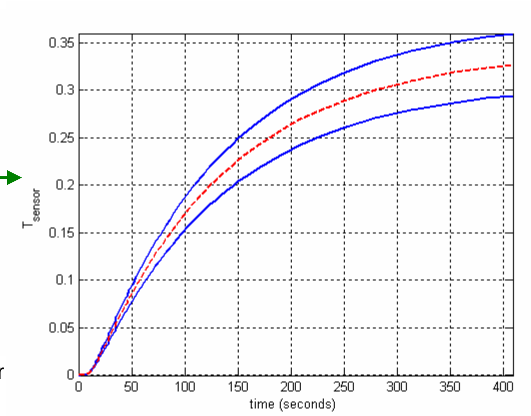
2500th-order finite-difference model

Reach set computed using 30th-order Krylov subspace reduced models.

Example. 2-D Heat Transfer Problem



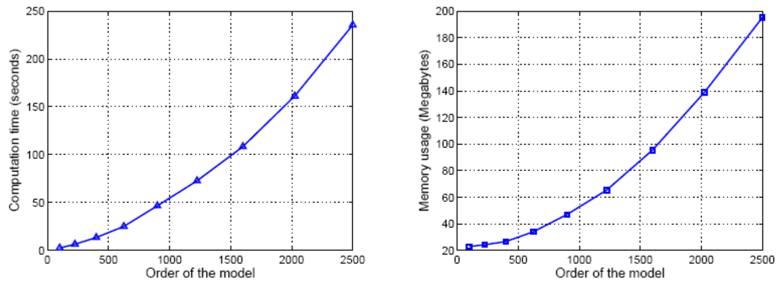
Steady-state temperature distribution for nominal in put 1 C.



Reach set vs. Time at one point

Time/Memory vs. Order

2-D heat transfer problem (100th to 2500th – order)

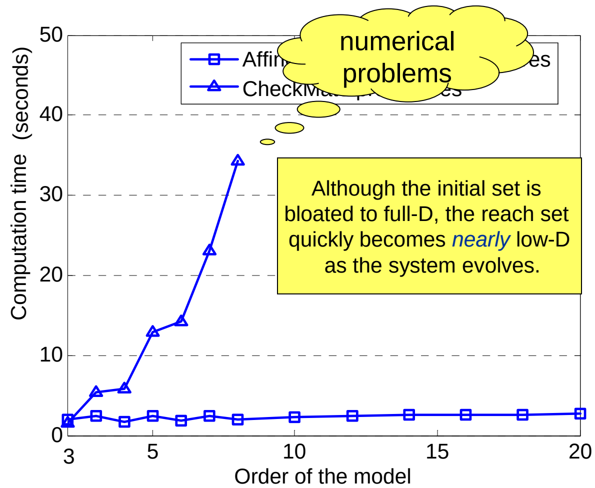


(a) Computation time as a function of model order. (b) Memory usage as a function of model order.

Preliminary Results for Hybrid System Verification

- A set of procedures developed to replace the subroutines of CHECKMATE.
- Compare the results using affine representations and CHECKMATE using hybrid system models of thermostat with various orders

Computation Time for Analyzing a Thermostat



The initial set is a 1-D polytope.

CHECKMATE always bloats the 1-D polytope to full-D.

Procedures using affine representations keep the 1-D polytopes.

Principal References

A. Chutinan and B. H. Krogh, Computational techniques for hybrid system verification, *IEEE Trans. on Automatic Control*, vol. 48, no. 1, 2003, pp. 64-75.

Z. Han and B. H. Krogh, Reachability analysis of large-scale affine systems using low-dimensional polytopes, *Hybrid Systems: Computation and Control*, 8th International Workshop, March 2006.

Next Lecture

- Using linear hybrid automata to approximate general hybrid systems

Lecture 3: Linear Hybrid Automata

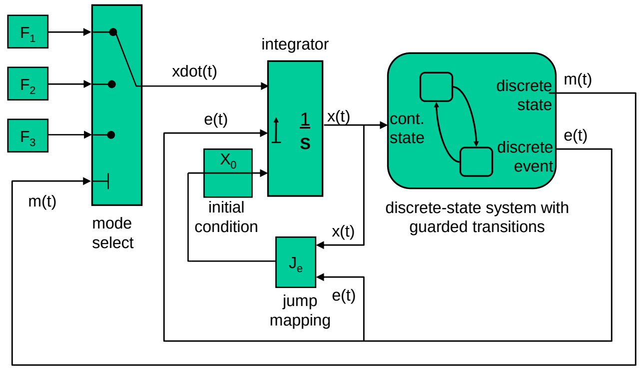
Bruce H. Krogh
Carnegie Mellon University
krogh@ece.cmu.edu

Overview

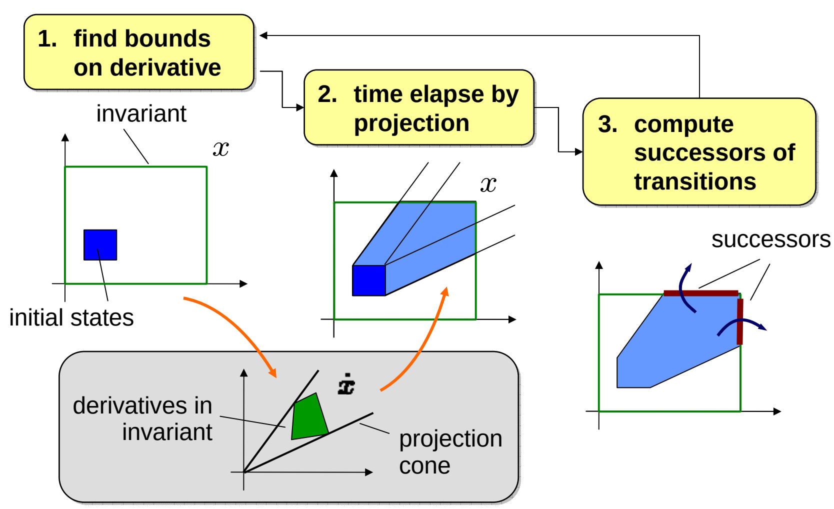
- LHA Reachability
- Approximating Richer Dynamics
- PHAVer
- Iterative Relaxation Abstractions

Linear Hybrid Automata

- F_k (flow constraints), J_e (jump mappings), and G_{jk} (guards) are convex polyhedra
- F_k are independent of x(t)



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

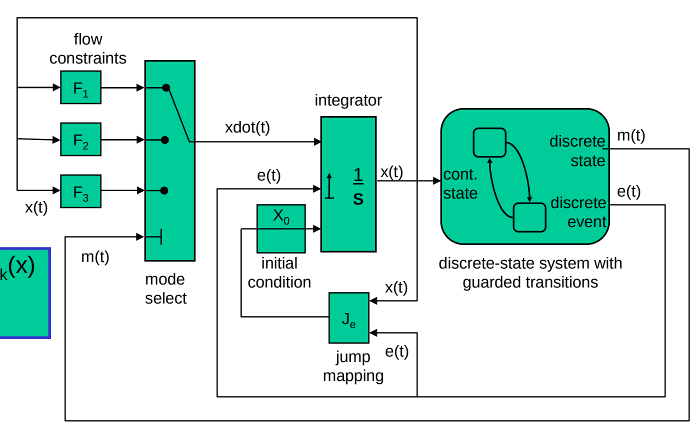


Overview

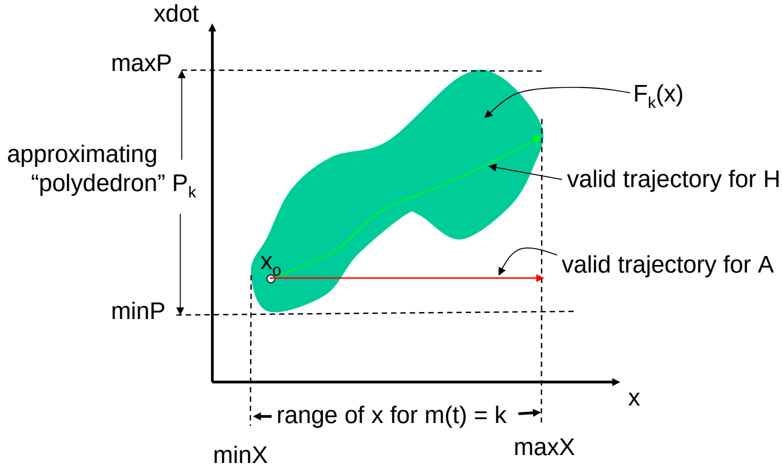
- LHA Reachability
- Approximating Richer Dynamics
- PHAVer
- Iterative Relaxation Abstractions

Approximating Hybrid Systems with Linear Hybrid Automata

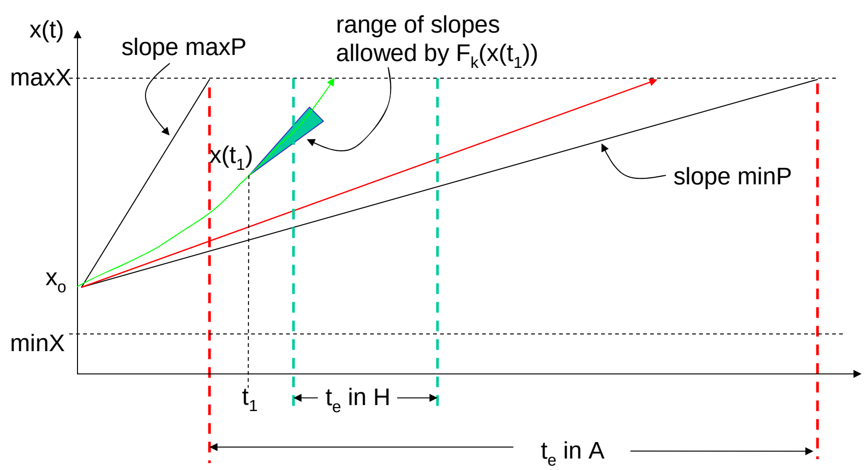
Objective: Replace $F_k(x)$ with constant convex polyhedra P_k.



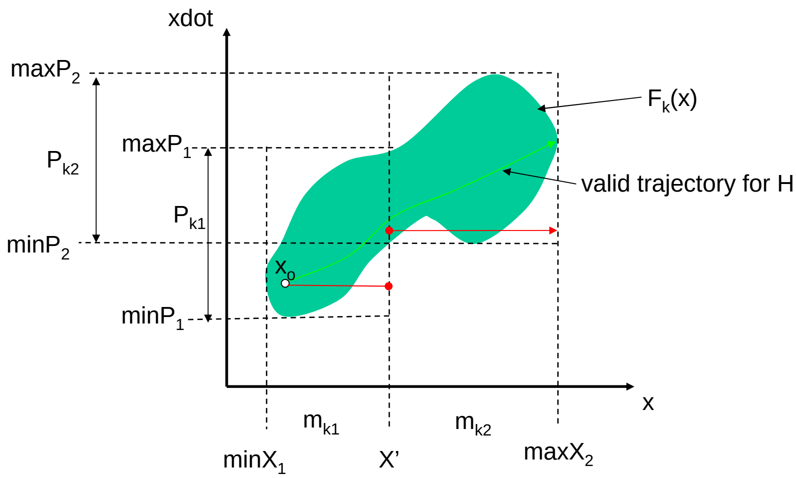
Linear Phase-Portrait Approximation



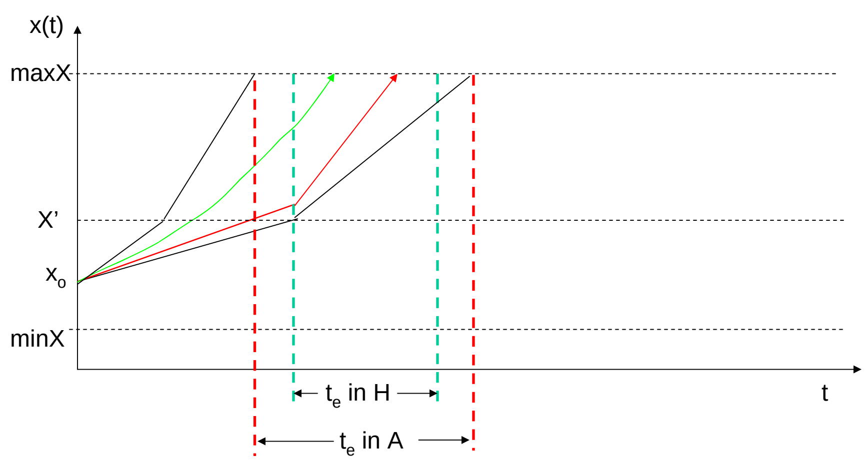
Linear Phase-Portrait Approximation: Time-Domain Implications



Improving Linear Phase-Portrait Approximations: Mode Splitting



Linear Phase-Portrait Approximation: Improved Time-Domain Approximation

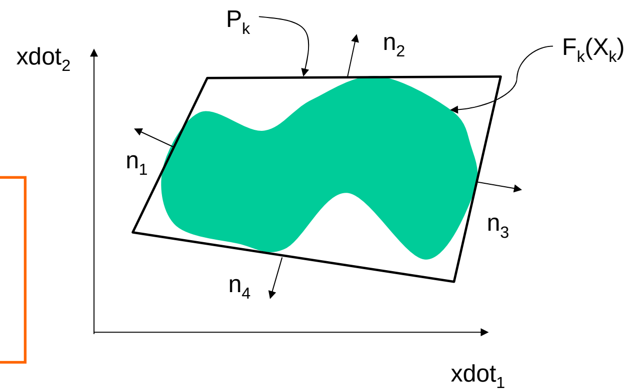


Linear Phase-Portrait Approximation: Higher Dimensions

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

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

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



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 ($n_i^T = [0\ldots1\ldots0]$)

CarnegieMellon

Overview

- LHA Reachability
- Approximating Richer Dynamics
- PHAVer
- Iterative Relaxation Abstractions

13:84

CarnegieMellon

The following slides are excerpts from the following presentation:

PHAVer: Reachability Analysis for Linear Hybrid Systems and Beyond

Goran Frehse

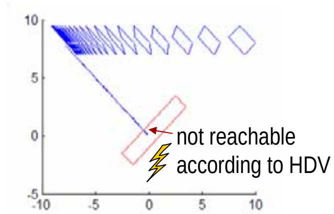
Verimag – UJF/CNRS/INPG, Grenoble

PHAVer available at http://www.cs.ru.nl/~goranf/

14:84

Yet Another Verification Tool?

- Existing not powerful enough
  - in practice only 3 - 4 dimensions
- Non-conservative floating-point tools give wrong results
  - exception: HSOLVER
- Why not use HyTech?
  - numerical problems, no easy fix (exact arithm. & 32 bit ⇒ overflow)
  - complexity explosion
  - limited class of automata (LHA)



thanks to Zhi Han, CMU

Floating-Point:
CheckMate (CMU '98)
HYSDEL (ETH Zurich '99)
d/dt (Verimag '00)
Predicate Abstraction (UPenn '02)
HDV (UPenn '04)
HSOLVER (MPI '05)

Exact Arithmetic:
HyTech (Berkeley '95)

Polyhedral Hybrid Automaton Verifyer

Model: Hybrid Automata $M\dot{x} = Ax + b$, M, A, b as intervals

↓ On-the-fly over-approximation

Analysis Engine: Linear Hybrid Automata $\sum_k \alpha_k \dot{x}_k + \beta \leq 0$

↓ Overapprox. with limited complexity

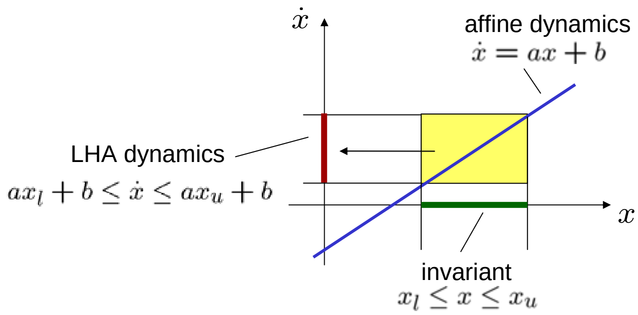
Output: Reachable States as Polyhedra

- Reachability Analysis
  - exact arithmetic
  - guaranteed overapproximation
  - complexity management
    - limiting bits & constraints
- State-of-the-Art Libraries:
  - Parma Polyhedra Library
  - Gnu MultiPrecision (GMP)
- Compositional Reasoning
  - computing simulation relations

Over-Approximation of Affine Dynamics

- From $\sum_i \alpha_i \dot{x}_i + \underbrace{\sum_k a_k x_k + b}_{} \leq 0$

  to LHA: $\sum_i \alpha_i \dot{x}_i + \beta \leq 0$



Over-Approximation of Affine Dynamics

- From $\sum_i \alpha_i \dot{x}_i + \underbrace{\sum_k a_k x_k + b}_{} \leq 0$

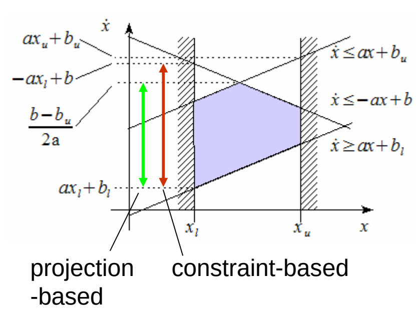
  to LHA: $\sum_i \alpha_i \dot{x}_i + \beta \leq 0$

- Solutions:

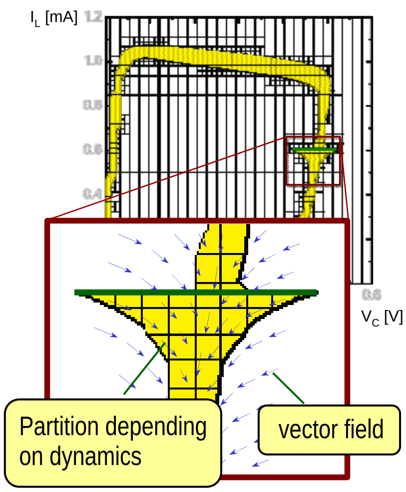
  a) project invariant ∩ flow to $\dot{x}$

  b) each constraint separately (rectangular, octagonal, etc.)

$\beta = max_{x \in Inv(loc)} \sum_k a_k x_k + b$



Reachability of Affine Dynamics



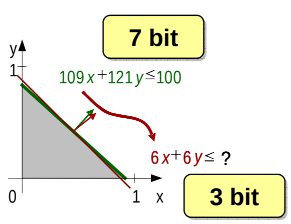
Principle:

1. Hybridization
   - Partition State Space (on the fly)
   - Switching between
   - ⇒ Hybrid System
2. Overapproximation
   - const. bounds on dynamics
     = *"Linear" Hybrid Automata*

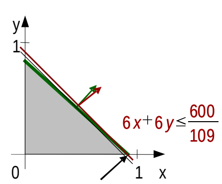
⇒ Polyhedral enclosure of actual trajectories

Limiting the Number of Bits

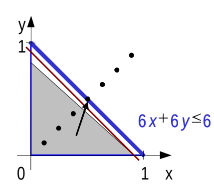
1. truncate bits of coefficients



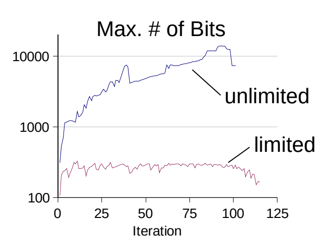
2. push plane to outside (solve LP)



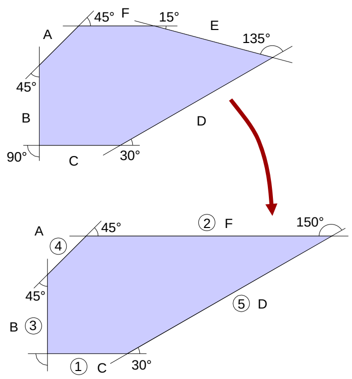
3. snap to next integer



- **Good:**
  - –large problems infeasible without
  - –with limit of constraints → termination
- **Bad:**
  - –unbounded error



Limiting the Number of Constraints



From 6 to 5 constraints

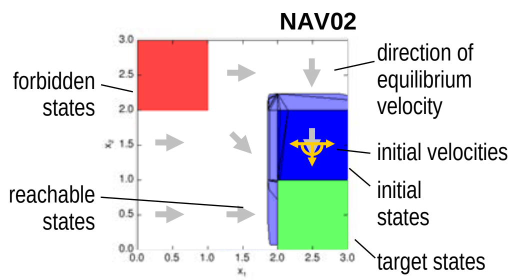
- Reduce from *m* to *z* constraints
- Significance Measure *f*(*m*,*d*)
  - Volume: exp
  - Slack: LP
  - **max. angle: m^2d**

$\Rightarrow$ $-\min_{i \neq j} a_i^T a_j$

- Heuristics to choose constraints
  - **deconstruction:** *drop (m-z) least significant*
  - **reconstruction:** *add z most significant*
- Experiments: angle & reconstr.
  - 1000 $\rightarrow$ 50 in 4 dim: < 2 sec. (1000x faster than slack)

Navigation Benchmark

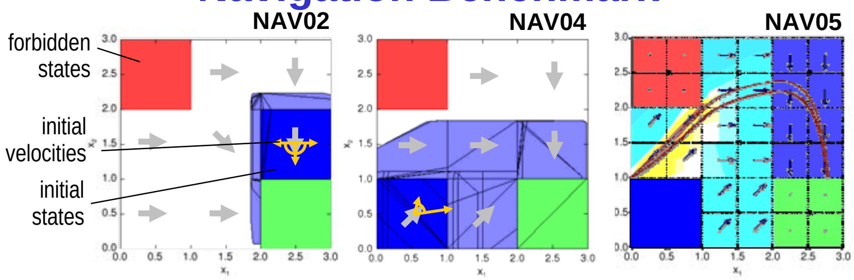
- Fehnker, Ivancic. *Benchmarks for Hybrid Systems Verification.* HSCC'04



- "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

www.cse.unsw.edu.au/~ansgar/benchmark/

Navigation Benchmark



Instance \ Tool	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

PHAVer References

- Reachability Analysis
  - PHAVer: Algorithmic Verification of Hybrid Systems past HyTech
    Frehse. HSCC'05
  - Time Domain Verification of Oscillator Circuit Properties
    Frehse, Krogh, Rutenbar, Maler. FAC'05
  - Verifying Analog Oscillator Circuits Using Forward/Backward Abstraction Refinement
    Frehse, Krogh, Rutenbar. DATE'06
- Compositional Reasoning
  - On Timed Simulation and Compositionality
    Frehse, FORMATS'06
  - Assume-Guarantee Reasoning for Hybrid I/O-Automata by Over-Approximation of Continuous Interaction
    Frehse, Han, Krogh. CDC'04

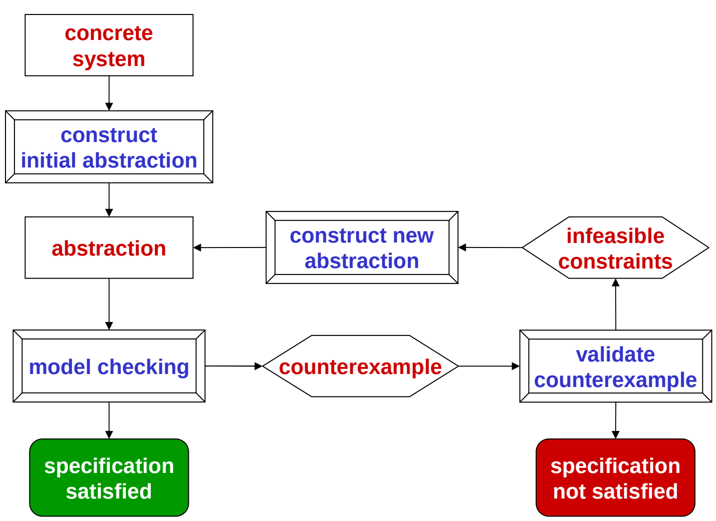
http://www.cs.ru.nl/~goranf/

Overview

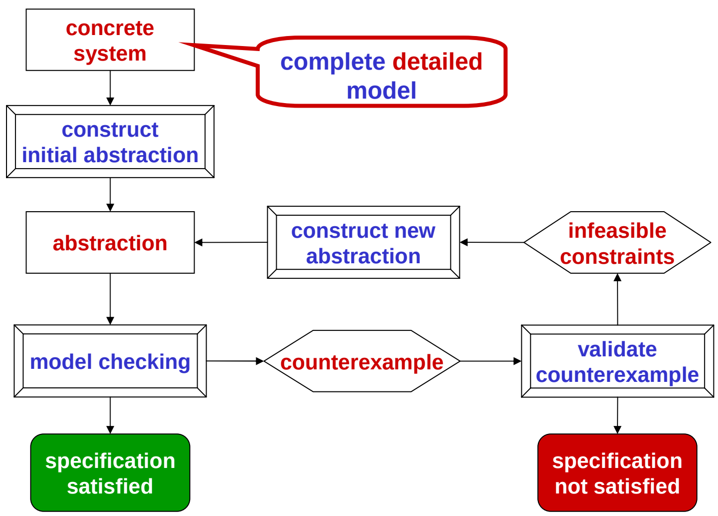
- LHA Reachability
- Approximating Richer Dynamics
- PHAVer
- Iterative Relaxation Abstraction

CEGAR

(CounterExample Guided Abstraction Refinement)

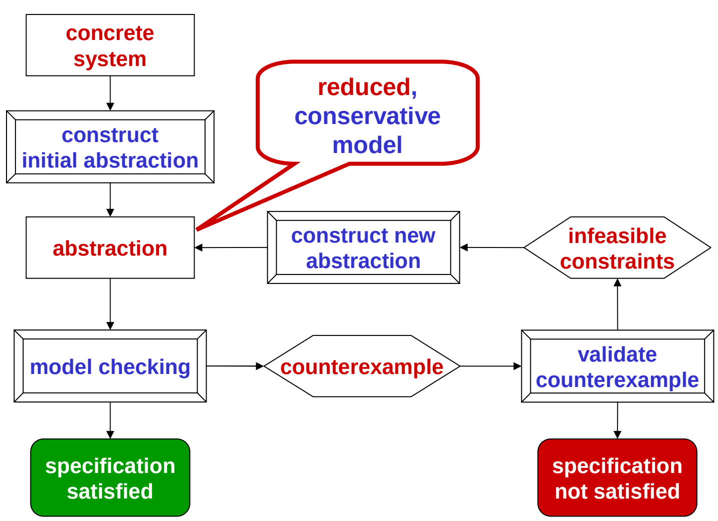


CEGAR

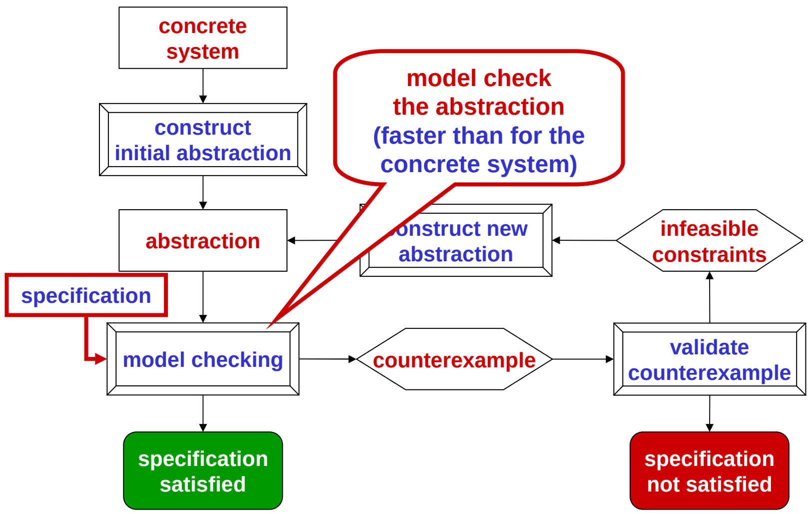


Carnegie Mellon

CEGAR

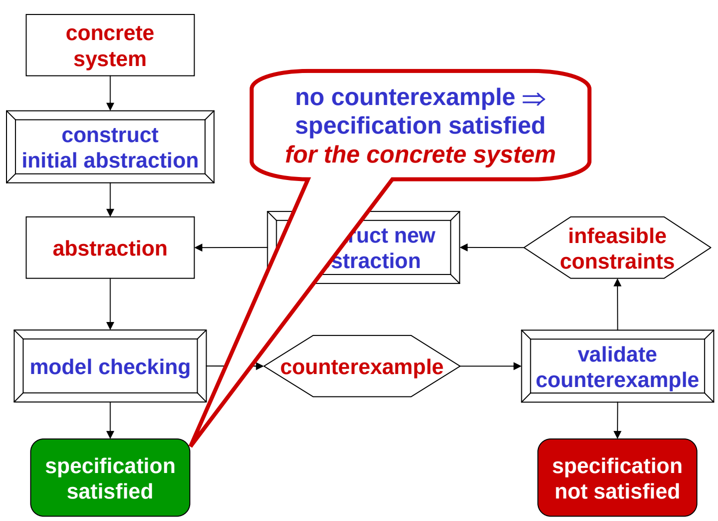


CEGAR



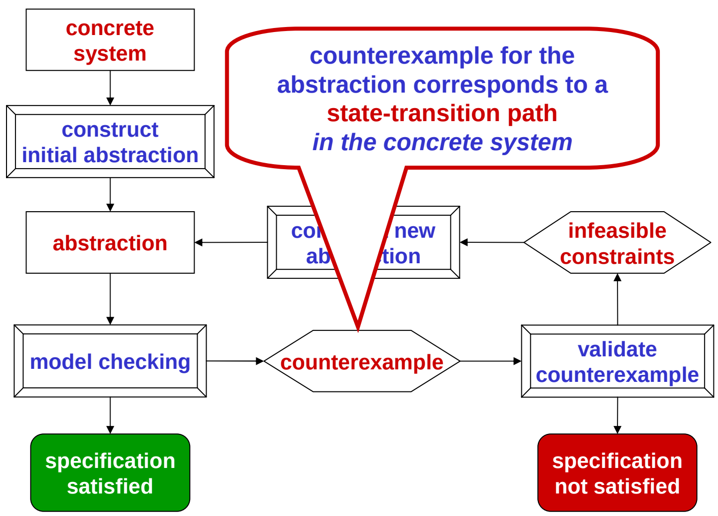
29:84

CEGAR

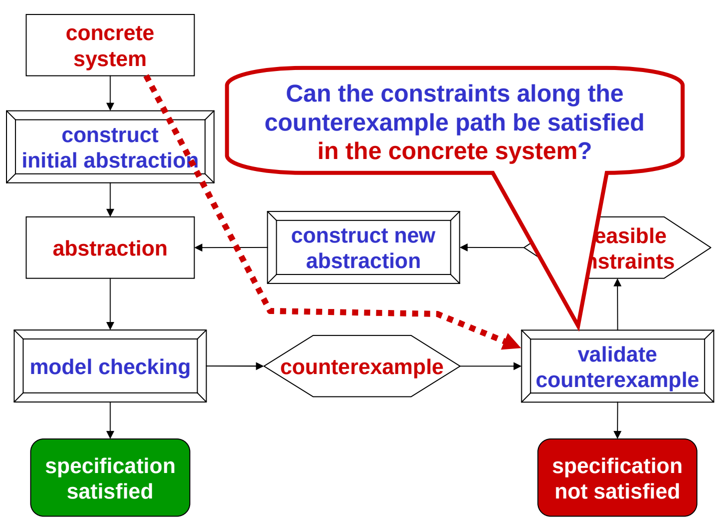


30:84

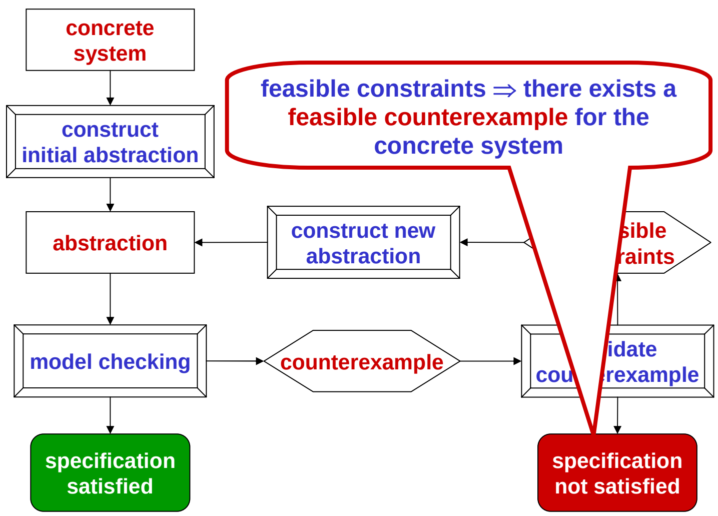
CEGAR


concrete system
construct initial abstraction
abstraction
model checking
counterexample
validate counterexample
infeasible constraints
specification satisfied
specification not satisfied
counterexample for the abstraction corresponds to a state-transition path *in the concrete system*

CEGAR

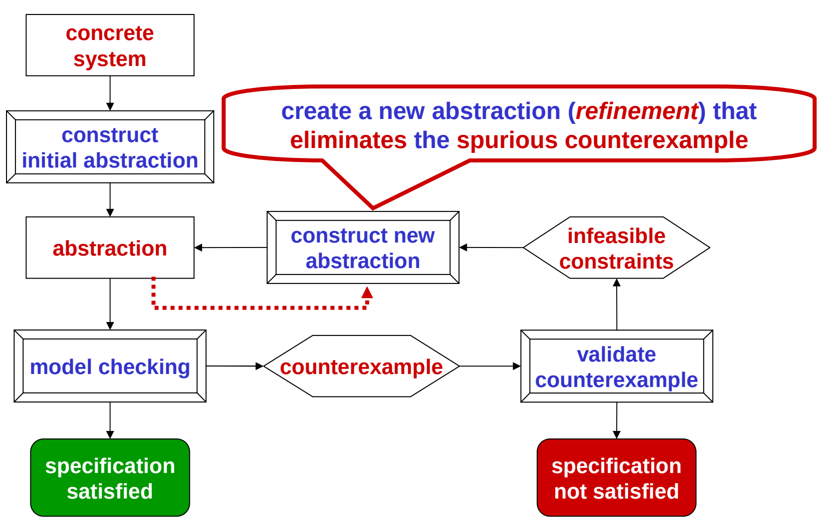

concrete system
construct initial abstraction
abstraction
construct new abstraction
model checking
counterexample
validate counterexample
specification satisfied
specification not satisfied
Can the constraints along the counterexample path be satisfied in the concrete system?

CEGAR



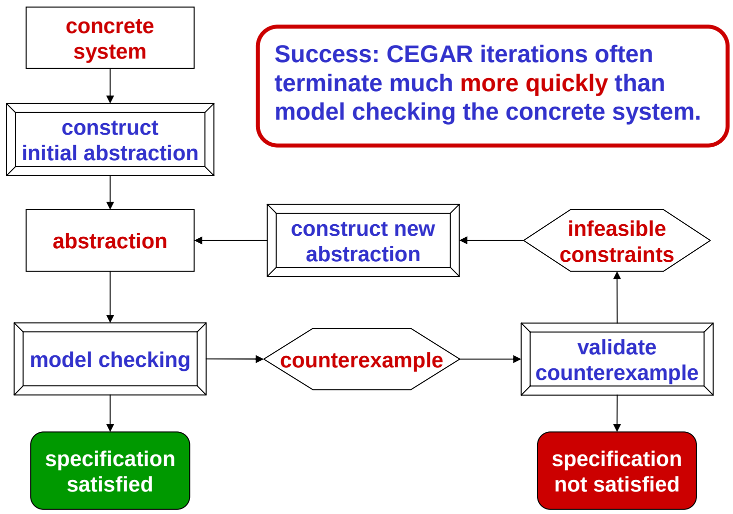
33:84

CEGAR

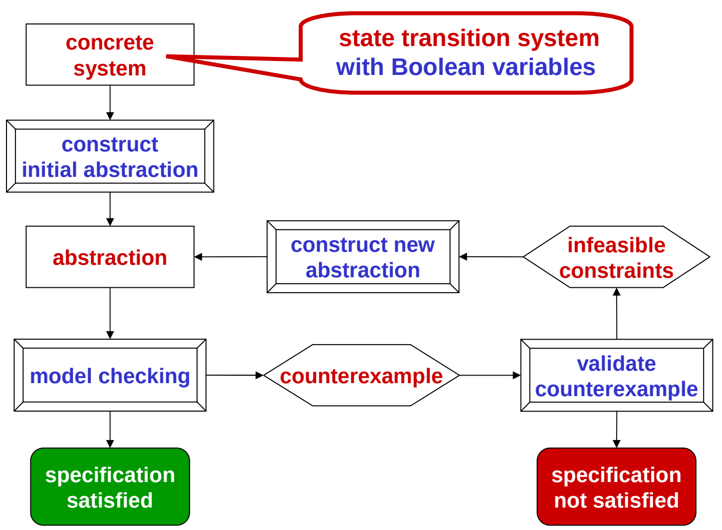


34:84

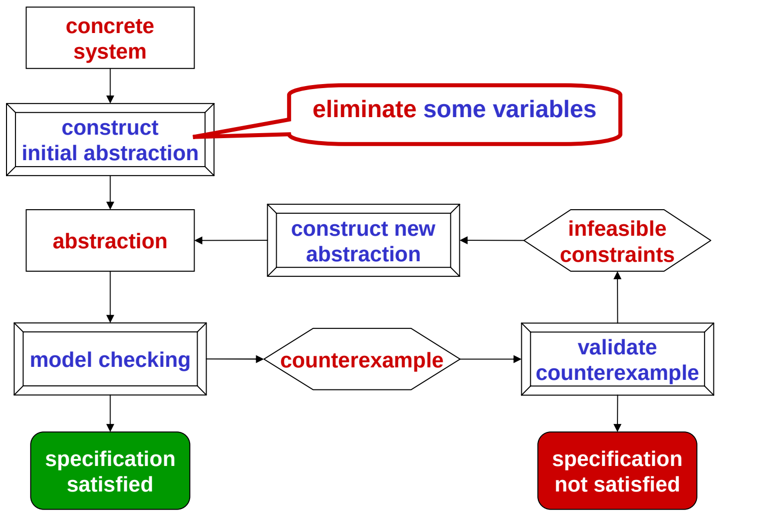
CEGAR



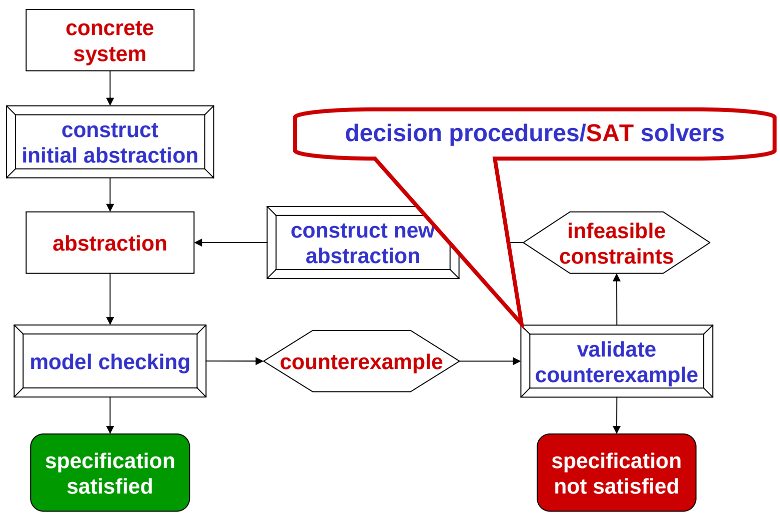
CEGAR for Discrete Systems



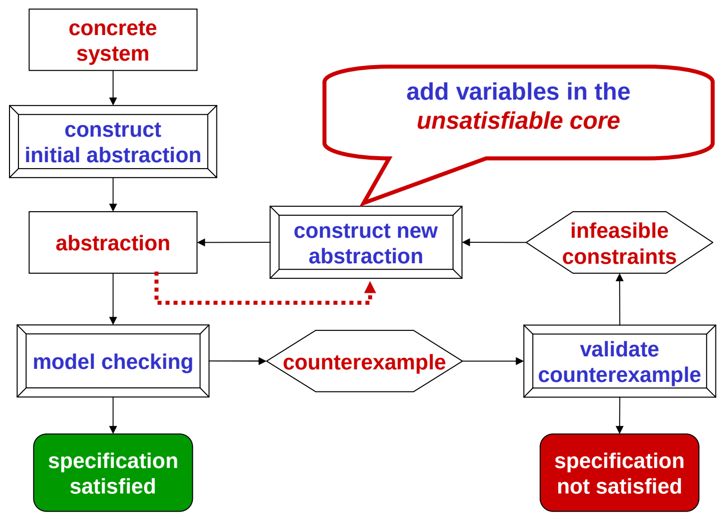
CEGAR for Discrete Systems



CEGAR for Discrete Systems



CEGAR for Discrete Systems



39:84

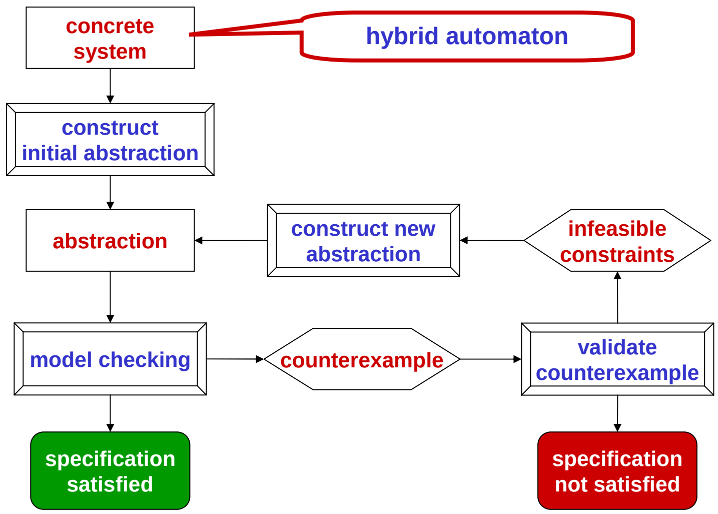
CEGAR for Discrete Systems

- Leverages
  - Power of model checking on **simpler models**
  - Power of decision procedures / SAT solvers to **validate counterexamples**
- Empirically a very powerful approach
- Many success stories
  - **SLAM** : Verifying Device Drivers at Microsoft
    - Actually ships as a commercial product **Static Driver Verifier (SDV)**
  - Many software model checkers developed
    - MAGIC, BLAST, CBMC

40:84

CarnegieMellon

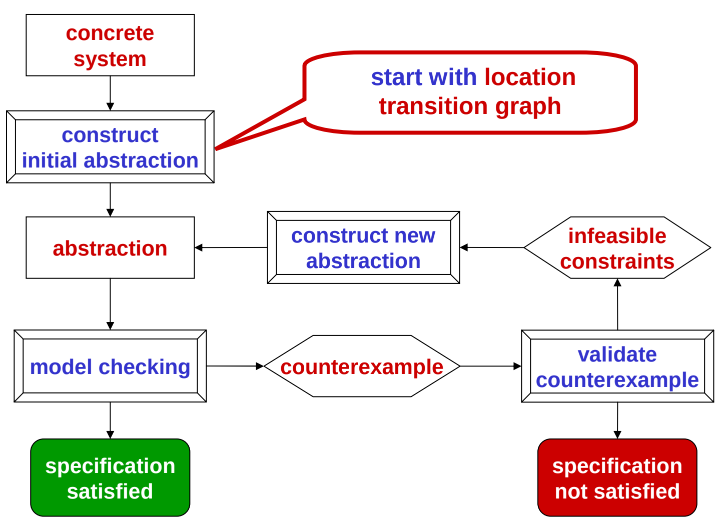
CEGAR for Hybrid Systems
(our previous work)



41:84

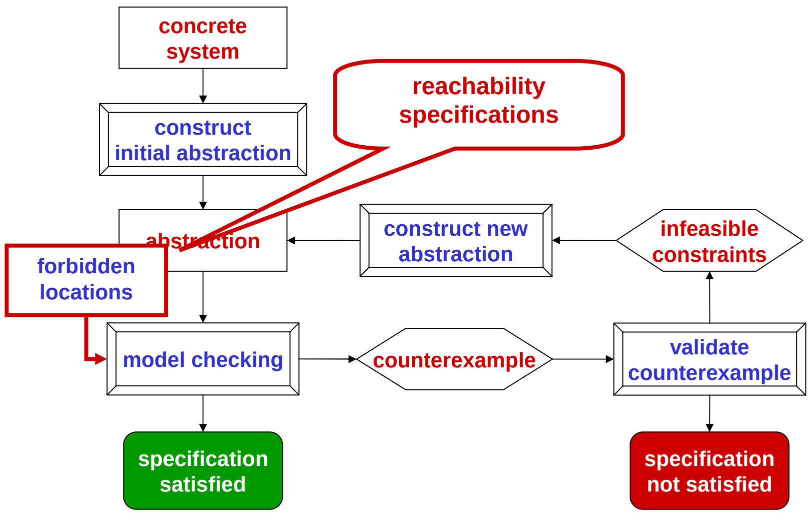
CarnegieMellon

CEGAR for Hybrid Systems

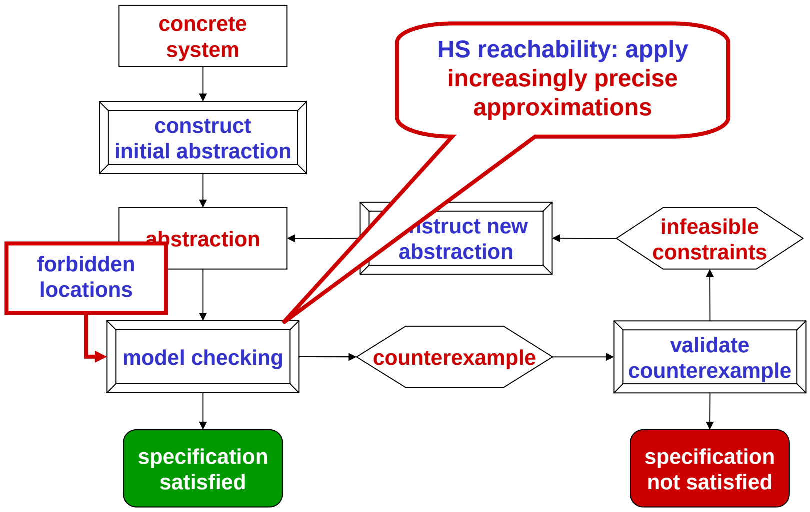


42:84

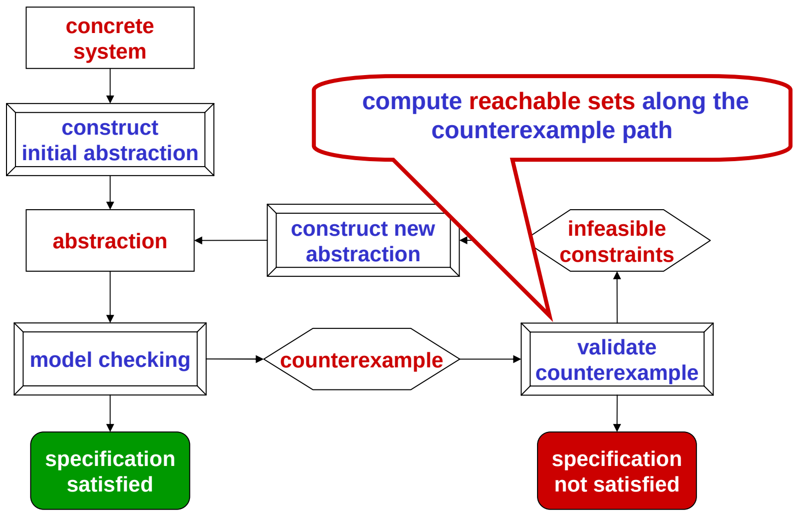
CEGAR for Hybrid Systems



CEGAR for Hybrid Systems

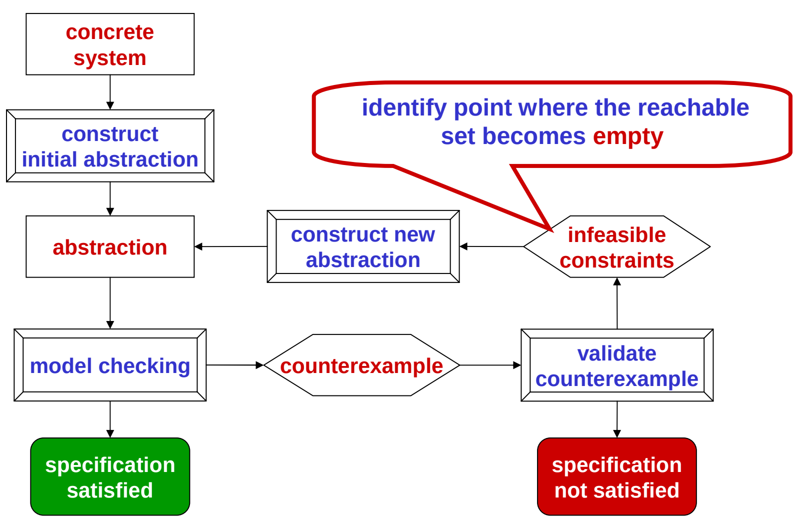


CEGAR for Hybrid Systems



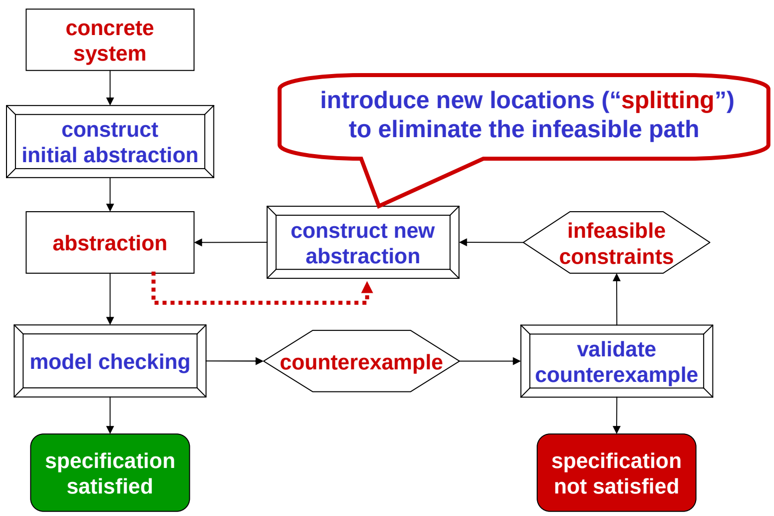
45:84

CEGAR for Hybrid Systems

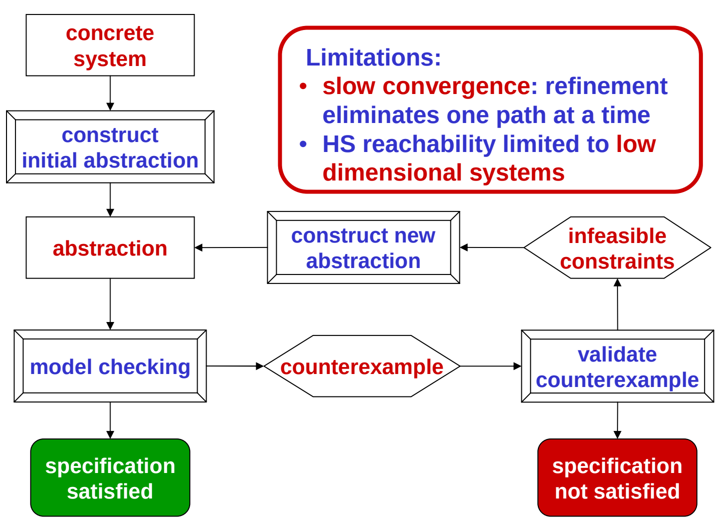


46:84

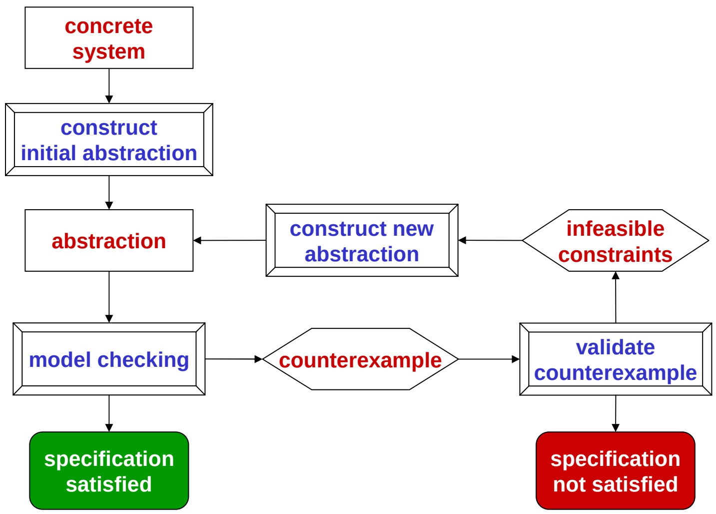
CEGAR for Hybrid Systems



CEGAR for Hybrid Systems

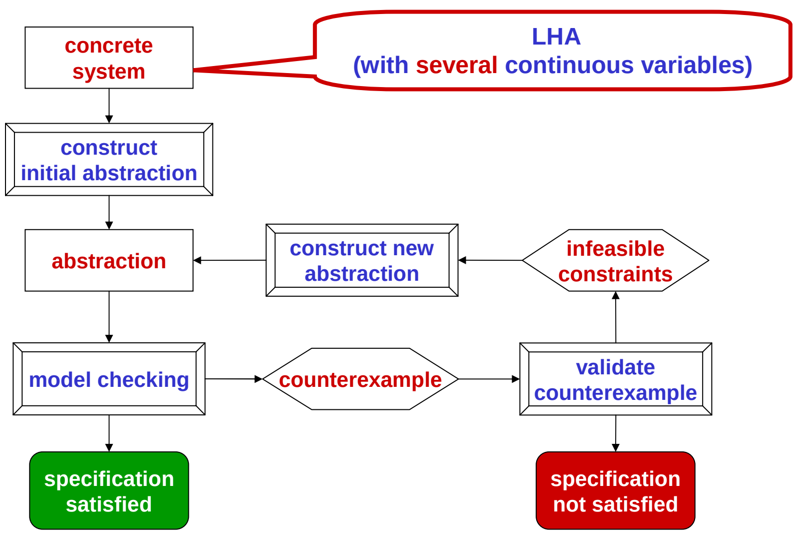


Iterative Relaxation Abstraction (IRA) for Linear Hybrid Automata (LHA)



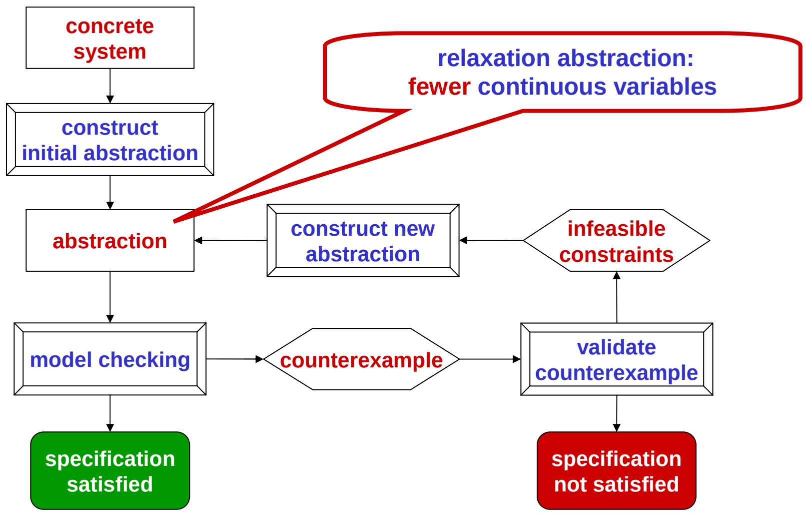
49:84

IRA for LHA



50:84

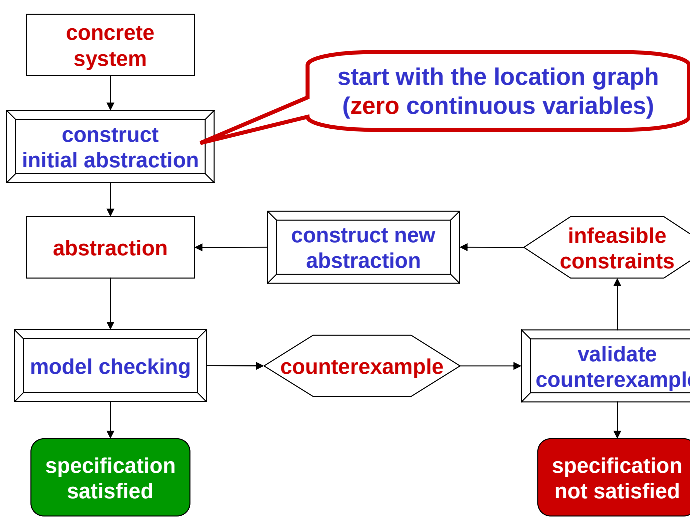
IRA for LHA



51:84

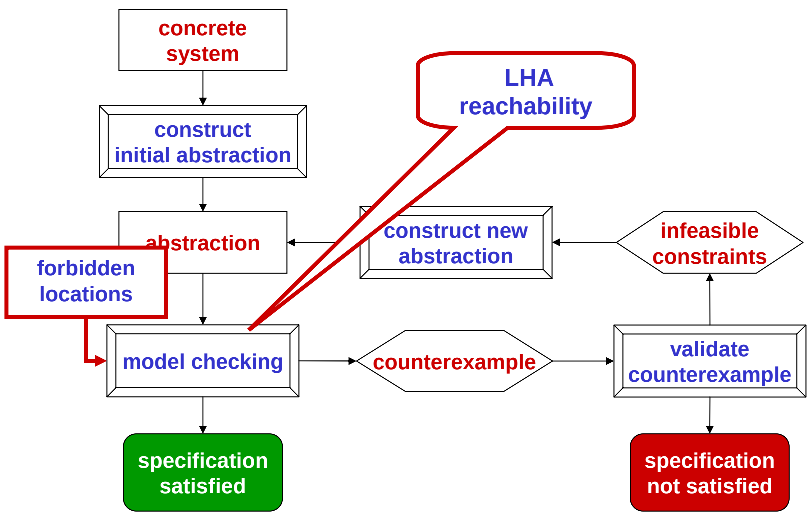
CarnegieMellon

IRA for LHA

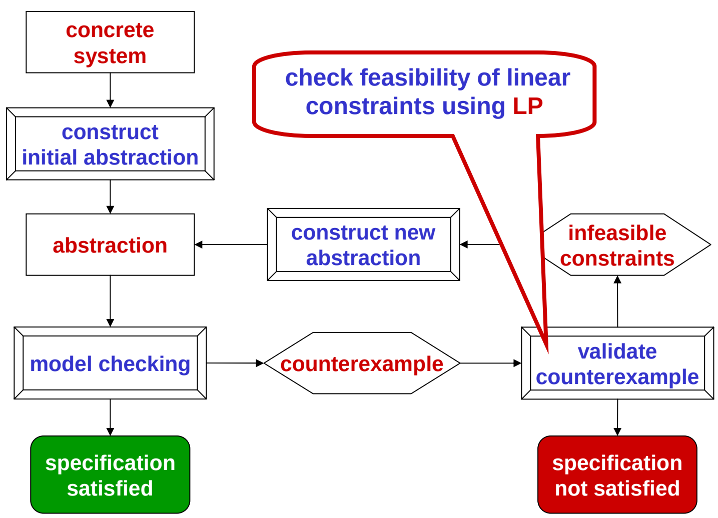


52:84

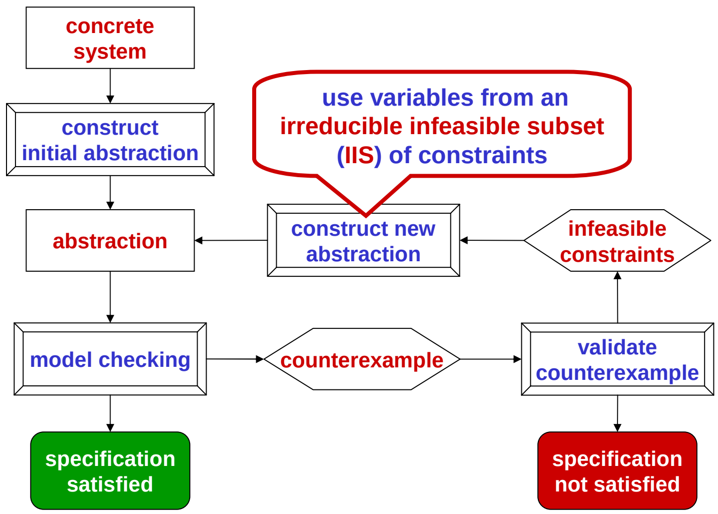
IRA for LHA



IRA for LHA

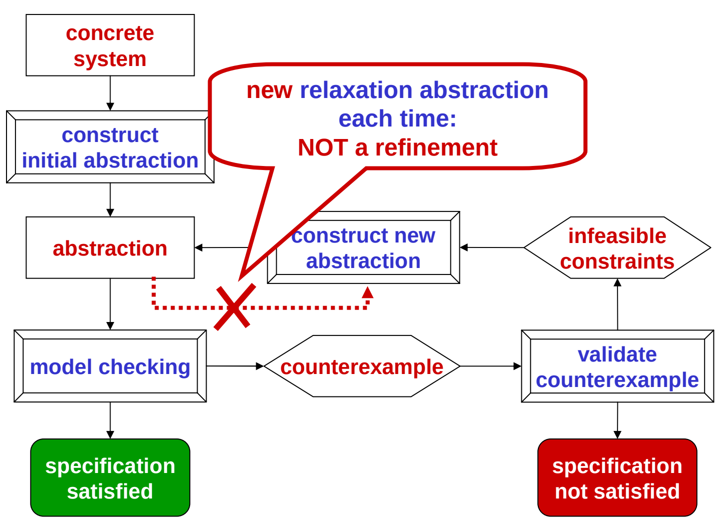


IRA for LHA



55:84

IRA for LHA



56:84

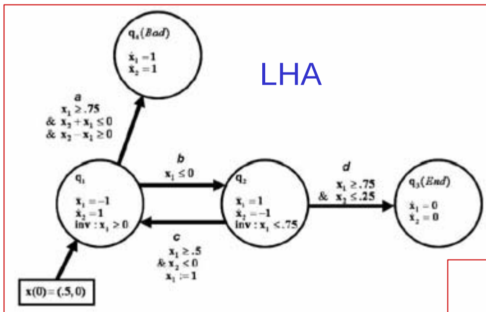
IRA for LHA – Leverages:

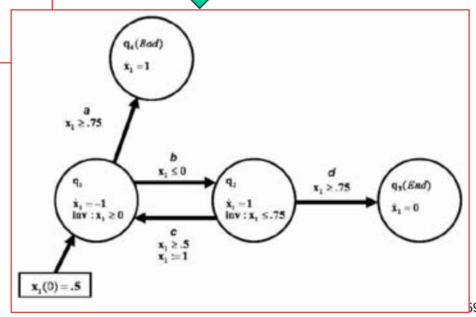
- Power of LHA reachability on low-order LHA models
- Power of LP to validate counterexamples involving huge number of continuous variables.
- Ability of a LP solver to identify an irreducible infeasible subset for an infeasible LP
- Inspired by CEGAR for discrete systems, but variables are not added to refine abstractions

Relaxation Abstractions

- *LHA*
  - discrete transition structure (locations/transitions)
  - linear constraints for invariants, guards, jumps
- *Given a subset of continuous variables V*
- *Replace linear constraints with relaxed constraints involving only variables in V*
  - $x<100 \wedge x>20 \wedge y<30 \wedge x<y$ *can be relaxed to* $x<100 \wedge x>20$
- *Not unique – various relaxations*
  - Drop constraints involving variables not in *V (***localization***)*
  - Quantifier Elimination (**Fourier-Motzkin)**

Relaxation Abstractions





Relaxation Abstraction (localization on x_1)

Counterexamples (CEs)

- *Paths in the discrete structure (sequence of locations and transitions)*
- *Key observations [Xuandong Li, Sumit Jha, Lei Bu BMC06] :*
  - *Feasible runs along a path are defined by* ***linear constraints***
  - CE exists in the concrete LHA **if and only if** the corresponding linear constraints are feasible

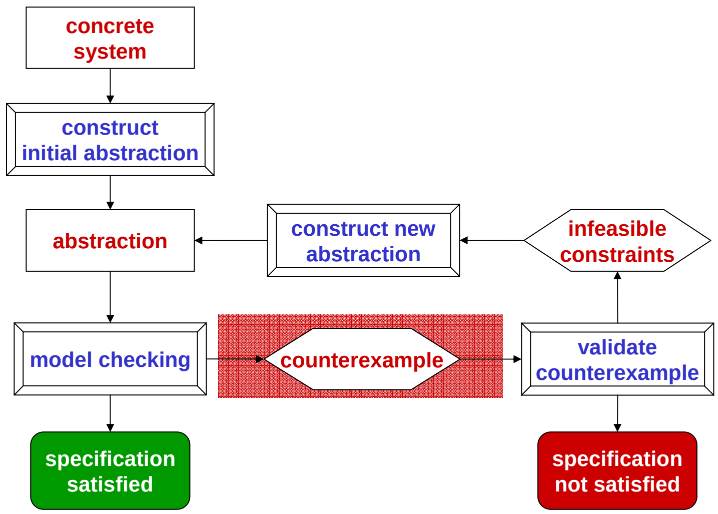
Irreducible Infeasible Subset (IIS)

- *Given a set of infeasible linear constraints (corresponding to a spurious CE).*
- *IIS: a subset of constraints such that*
  - the constraints are infeasible
  - removing one constraint makes them feasible
- *Use variables in the IIS for the next relaxation abstraction*

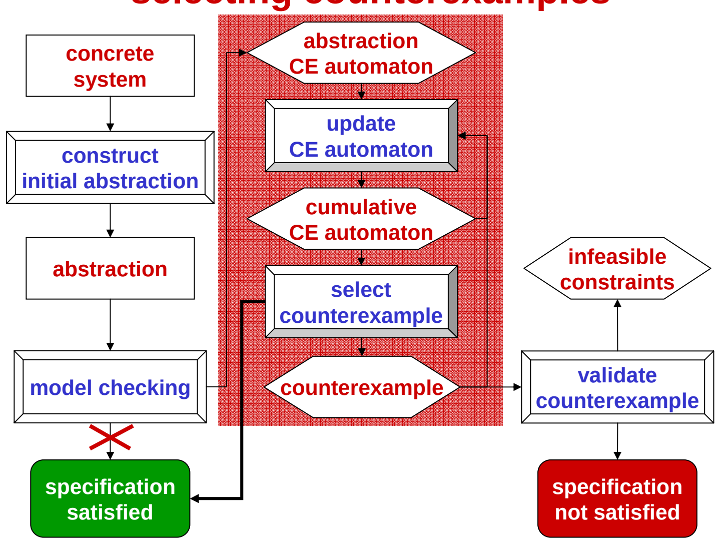
The Language of Counterexamples

- *LHA reachability gives a discrete CE automaton A for the current relaxed LHA*
  - *A string s = {s_0,s_1 ……,s_n} is in the language of the discrete CE automaton A* ***only if*** *the reachability analysis engine says that s_n* ***may be reachable*** *from s_0 using the path $s_0 \rightarrow s_1 \ldots \rightarrow \ldots \rightarrow s_n$.*
- *Intersect with the previous CE automaton*
  - *to* ***remove CE s refuted earlier*** *by other abstractions*
  - *also, remove previous CE in case reachability was too conservative*
- *Key Idea: Generate relaxation abstractions with only the most recent set of IIS variables.*

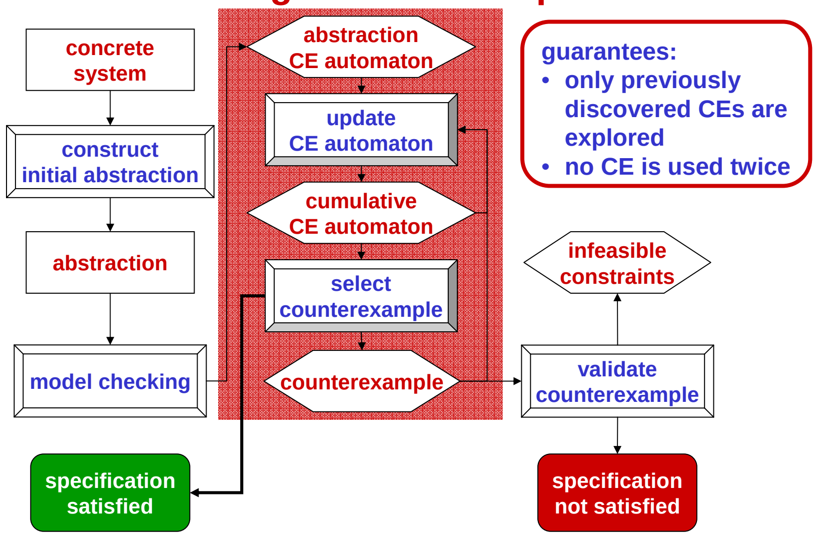
IRA for LHA
selecting counterexamples



IRA for LHA
selecting counterexamples

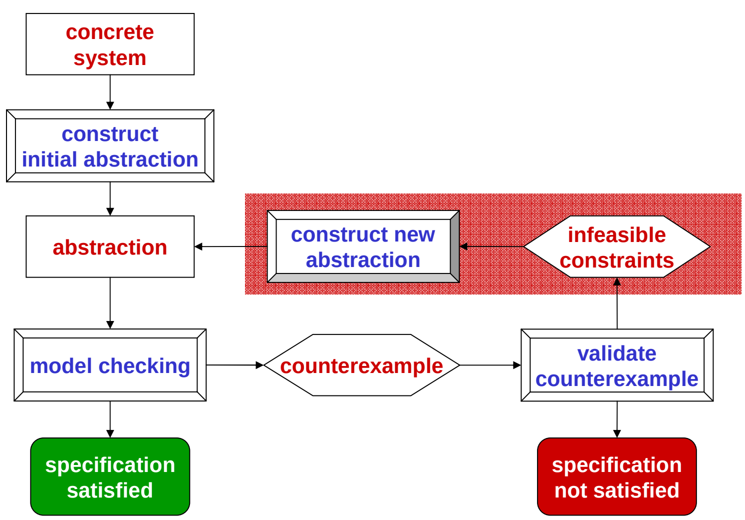


IRA for LHA
selecting counterexamples



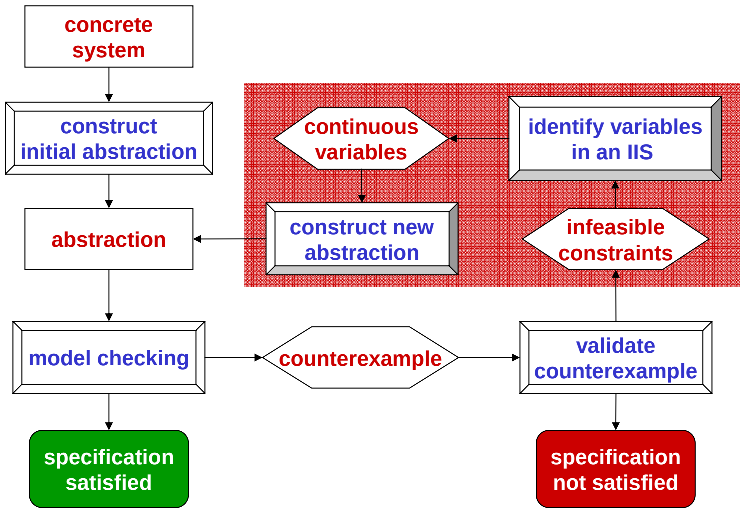
65:84

IRA for LHA
constructing new relaxation abstractions



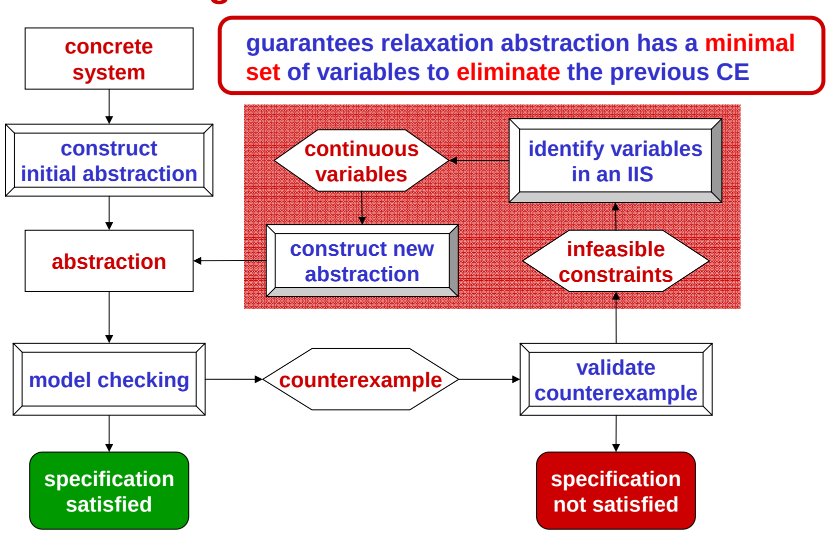
66:84

IRA for LHA
constructing new relaxation abstractions



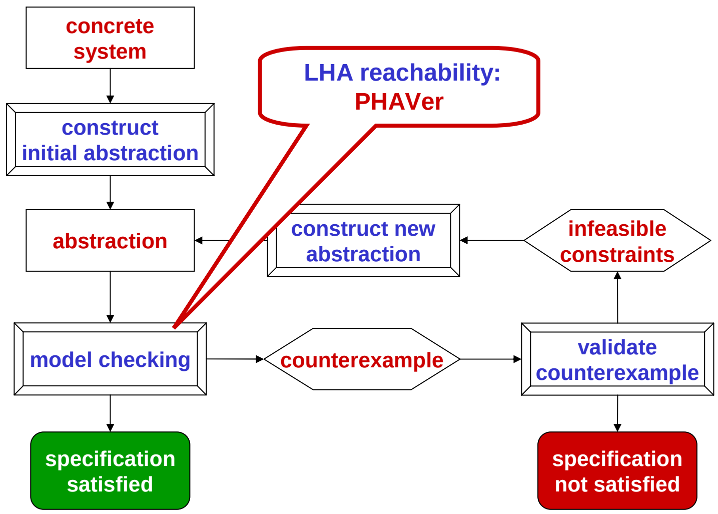
67:84

IRA for LHA
constructing new relaxation abstractions



68:84

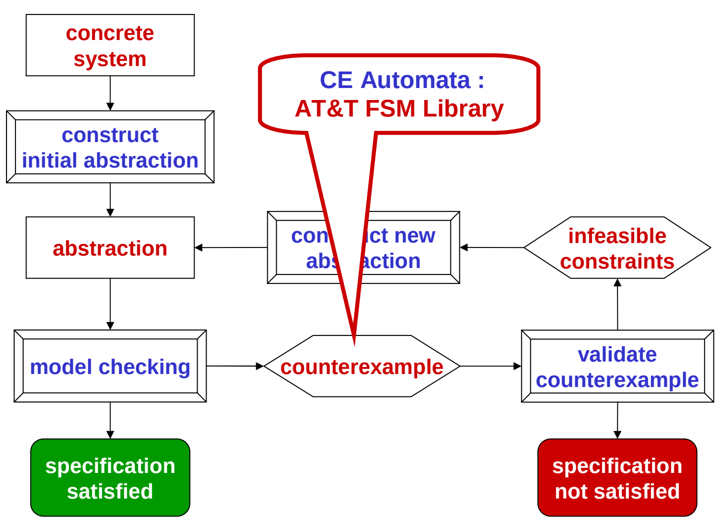
IRA for LHA
implementation



69:84

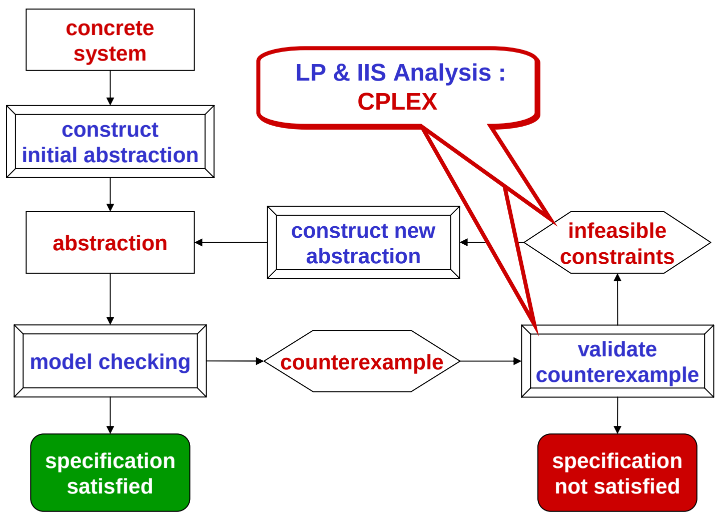
Carnegie Mellon

IRA for LHA
implementation



70:84

IRA for LHA
implementation



IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6	0.26	1.34	61.05
8	0.96	5.11	170.11
10	8.21	17.76	402.15
12	147.11	50.04	933.47
14	7007.51	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
[illegible]	[illegible]	1.34	61.05
8	0.[illegible]	5.11	170.11
10	8.21	17.76	402.15
12	147.11	50.04	933.47
14	7007.51	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

IRA becomes faster for ≥ 12 variables

73:84

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6	[illegible]	[illegible]	61.05
8	[illegible]	[illegible]	170.11
10	8.21	[illegible]	402.15
12	147.11	50.04	933.47
14	7007.51	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

IRA-FM becomes faster for ≥ 14 variables

74:84

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6	0.26	1.34	61.05
8	0.96	5.11	170.11
10	8.21	17.76	402.15
14	7007.3	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

15 Vars: 19.5 hr. (PHAVer) vs. 3 min. (IRA-LOC)

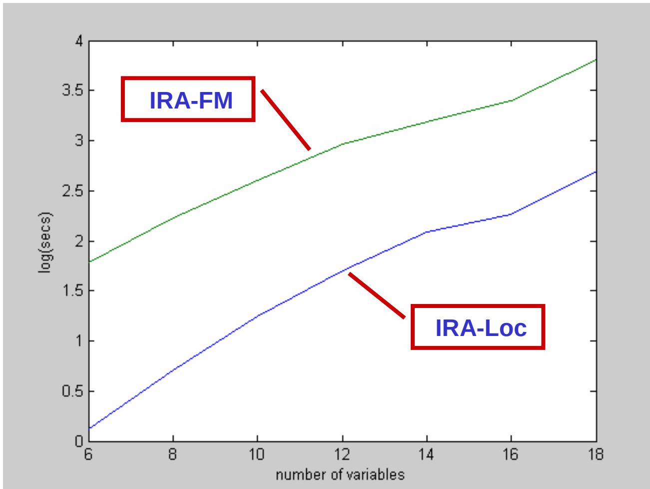
CarnegieMellon

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

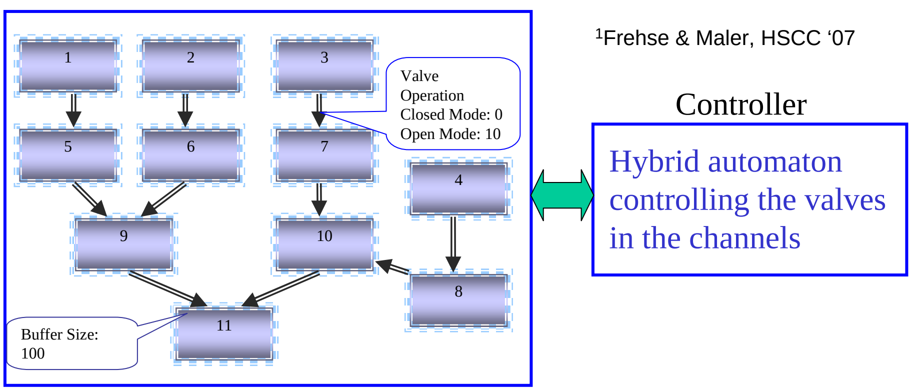
No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6	0.26	1.34	61.05
8	0.96	5.11	170.11
10			402.15
12	147.1	50.04	933.47
14	7007.	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

PHAVer fails to converge for 16 variables

IRA-Loc vs. IRA-FM



Switched Buffer Network[1]



[1]Frehse & Maler, HSCC '07

- Buffers connected by pipes with valves.
- Valves have several modes
- Controller observes buffers and to switch valve modes
- Specification: No buffer overflow

Carnegie Mellon

Switched Buffer Network

- Implemented a simple controller with three locations and 11 continuous variables
- Design: sequence of actual counterexamples from IRA used to "tune" the control parameters
- One case led to a 101 location CE in 3 iterations of the abstraction refinement loop

Final design (verified):

- PHAVer took over 12 minutes
- IRA took 23.7 seconds

79:84

Carnegie Mellon

Nuclear Power Plant Control[2]

- Temperature control
  - rods immersed to cool the reactor, withdrawn to allow reaction
  - rods controlled temperature measurements and local timers.
  - each rod can stay inside only for a certain max time limit
- Temperature should not rise beyond a critical threshold.
- Model
  - 3 control rods
  - 11 continuous variables

[2] Variation of the problem studied by Kapur and Shyamasundar (HART'97), R Alur et al (TCS'95), P. H. Ho 95 PhD thesis and others.

80:84

Nuclear Power Plant Control

Iterative Design Procedure

- First attempt:
  - simple counterexample of 3 locations
  - abstraction 3 continuous variables
  - all of variables related to control rod 1
  - clear that the rod was being inserted too late
  - changed the cutoff temperature
- Similar CEs for control rods 2 and 3

Final Design

- PHAVer verification: 16 hours
- IRA verification: 6 iterations, 30.04 seconds

Current Work

- Further empirical studies
- Use of IRA for interactive design (actually using the counterexamples!)
- Distributed computation (we have found most of the time is spent in FM quantifier elimination)
- Extensions to more general hybrid systems (outer refinement loops)

Principal References

T. A. Henzinger, P.-H. Ho and H. Wong-Toi, Algorithmic analysis of nonlinear hybrid systems, *IEEE Trans. on Automatic Control*, April 1998.

S. K. Jha, B. H. Krogh, J. E. Weimer, E. M. Clarke, Reachability for linear hybrid automata using iterative relaxation abstraction, *Hybrid Systems: Computation and Control*, April 2007.

Hybrid System Reachability: Additional Topics

- systems with inputs
  - control inputs
  - disturbances
- uncertain systems
  - unknown parameters
  - stochastic systems
- other abstractions/representations
  - predict abstraction
  - ellipsoids
  - qualitative reasoning
  - level sets
- theorem proving