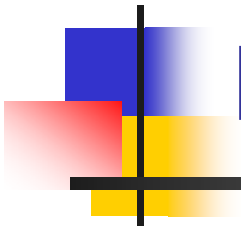
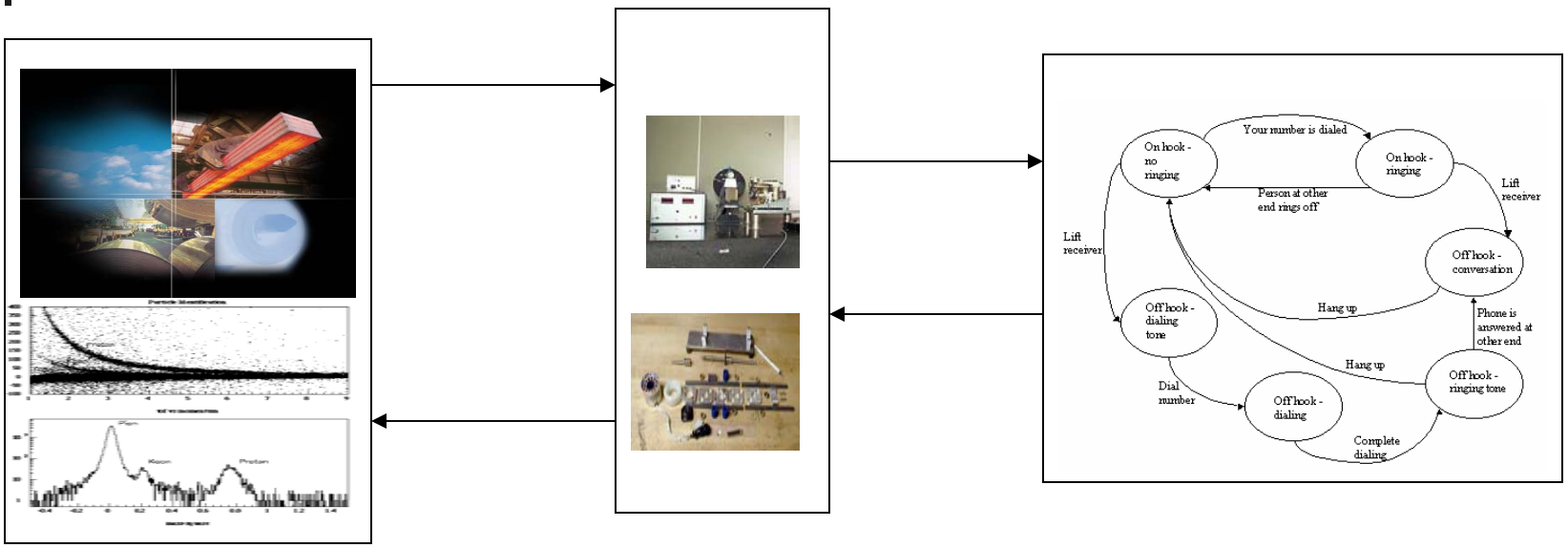


Symbolic Reachability Analysis of Lazy Linear Hybrid Automata

Susmit Jha, Bryan Brady
and
Sanjit A. Seshia

Presentation by: Susmit Jha

A typical hybrid system



HW Continuous Plant

SW Data Structure as envisioned by algorithm (integers, reals)

Sensor/Actuators with finite precision, non-zero delay

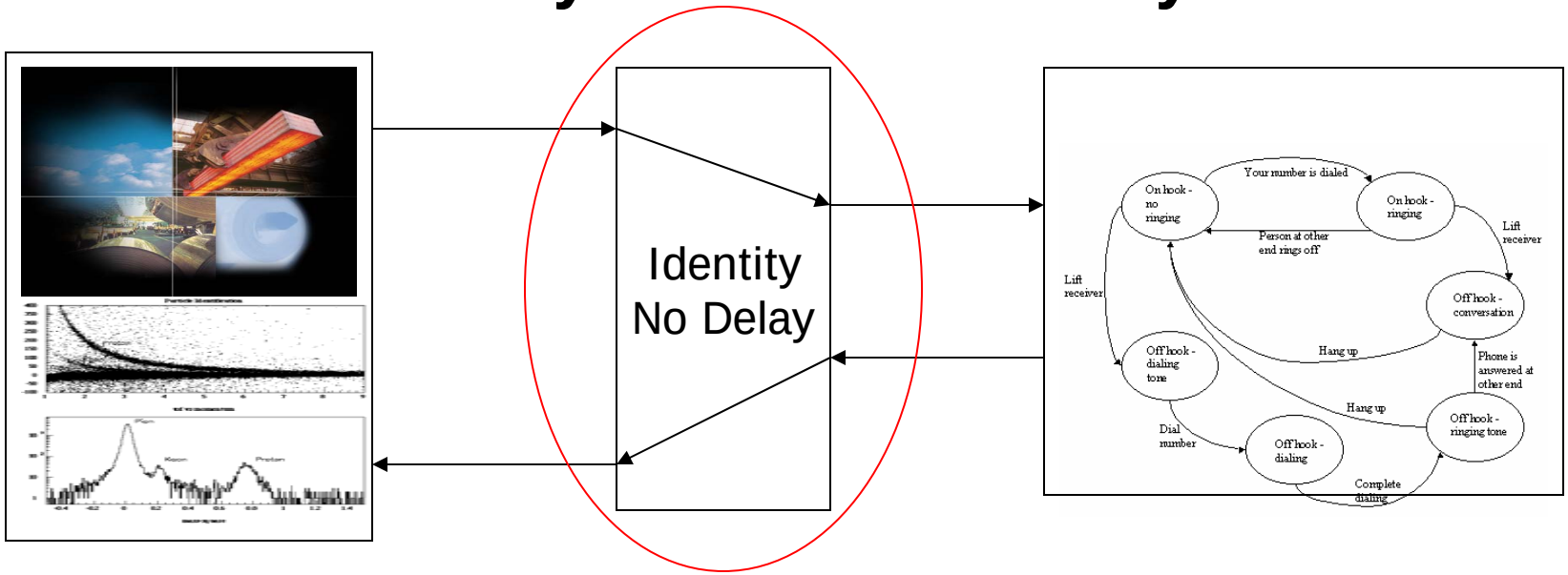
Data Structure mapped to finite registers as bit-vectors

Discrete Controller

Discrete Control Flow

Traditional Hybrid Automata

A model must abstract away ONLY the redundant features of system to make analysis easier !



Hybrid automata abstracts the intermediate discretization for such hybrid systems by approximating **high precision with continuum and ignoring delays.**

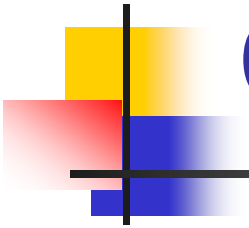
Most generalizations of initialized rectangular HA proved to be undecidable (Henzinger et al).

Alternative models

- Discrete Hybrid Automata (Torrisi et al) – Consists of a finite state machine communicating with a switched affine system through mode selector and event generator.

- Linear and Polynomial Hybrid Automata (Franzle et al) – Semi-decidable in most cases barring some pathological cases in which safety depends on complete absence of noise.

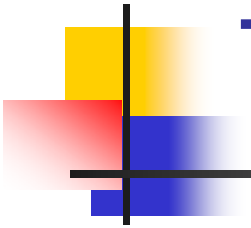
- Lazy Linear Hybrid Automata (LLHA) (Agrawal and Thiagarajan) – Models the inertial delays as well as finite precision of sensors and actuators. Reachability in LLHA is decidable.



Contributions

Goal: To develop a scalable technique for reachability analysis of LLHA

- Establishing an abstraction hierarchy for LLHA which can be used in a CEGAR framework
- A symbolic implementation of BMC for LLHA enhanced with k-induction
- Demonstration of scalability of our approach on examples like TCAS and AHS.



Talk Outline

- Background: Lazy Linear Hybrid Automata (LLHA)
- Overall Tool Flow
- Abstraction Hierarchy for LLHA
- Symbolic BMC of LLHA and K-Induction
- Case Studies and Comparison
- Future Work

Lazy Linear Hybrid Automata

LLHA is a tuple (X,V,flow,inv,init,E,jump,Σ,syn, D,ε,B,P)

X-Continuous Variables
V-Control Modes / Locations
Flow- Constant rates of change
Inv –Invariants at control modes
E - Control mode switches
Jump - Guards over switches
Σ – reset actions
Syn – synchronization labels

Lazy Linear Hybrid automata

LLHA is a tuple

$(X, V, flow, inv, init, E, jump, \Sigma, syn, D, \varepsilon, B, P)$

Corresponding to the interface

D corresponds to delay elements of the sensor-actuator interface

g, δ_g, h, δ_h (bounded delays)

Such that $g \leq$ actuation delay $\leq g+\delta_g$

$h \leq$ sensing delay $\leq h+\delta_h$

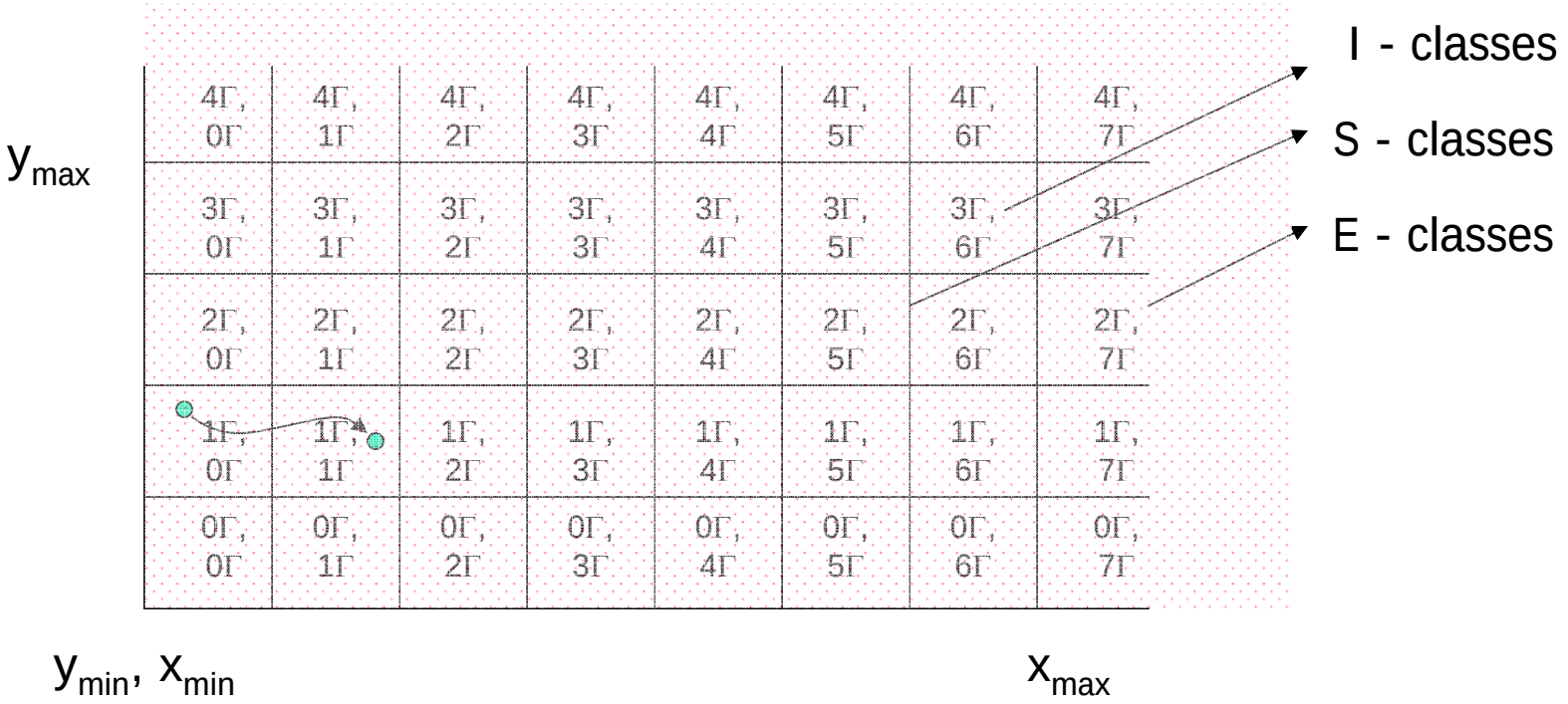
The continuous variables are observed by the controller with precision ε and are expected to be in a range $B = [B_{min}, B_{max}]$

The controller samples the values of variables at intervals of period P. For simplicity, we assume it to be 1.

Reachability in LLHA (AT-HSCC05)

Interface defines an equivalence relation

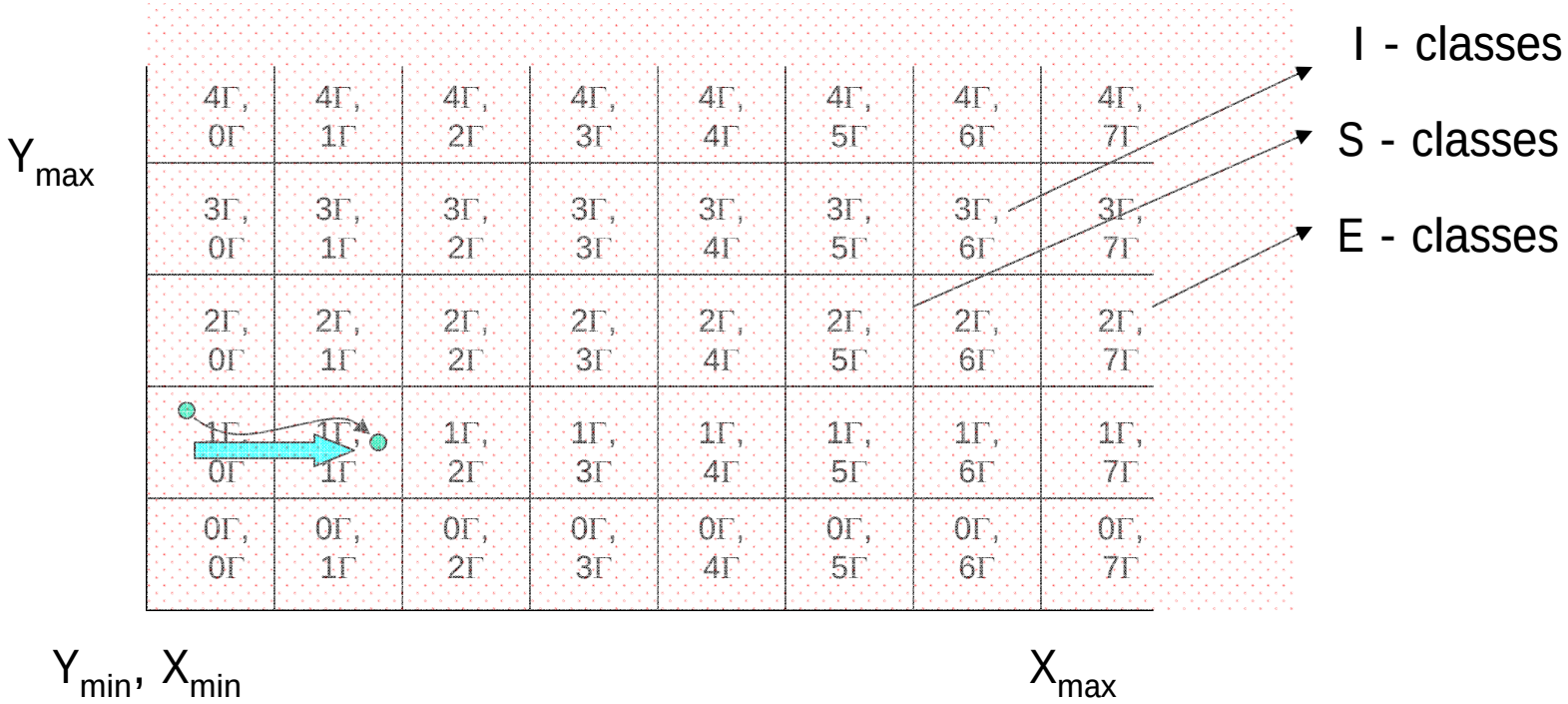
Let Δ = GCD(P,g,δg,h,δh) and Γ = GCD(RΔ, ε, Bmax, Bmin)
Γ used to construct an equivalence class partitioning.



Reachability in LLHA (AT-HSCC05)

Interface defines an equivalence relation

This equivalence relation is stable with respect to transitions.
[E(P1,P2) ∧ P1 -> Q1] => ∃ Q2 s.t. [P2 -> Q2 ∧ E(Q1,Q2)]



Reachability in LLHA (AT-HSCC05)

- Reachability of lazy linear hybrid automata is decidable. Several relaxations of LLHA like non-linear but computable guards are also decidable.

- The finite quotient space generated is finite with size

  $O(|Q|^4\, 2^{2n}\, K^{3n})$

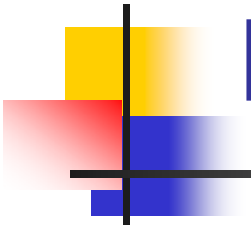
  Where Q = number of locations

  n = number of continuous variables

  $K = B_{max}/\Gamma - B_{min}/\Gamma$

  This can be very large !

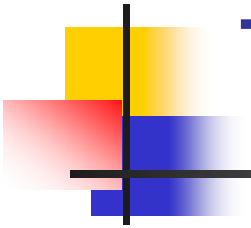
  For just 4 variables, 4 control modes and K as 10, the above bound is 1.6777216×10^{19}



Exploring Huge State Space

- Symbolic Bounded Model Checking –
  - Similar to Zone automata construction from the Region automata (Rajeev Alur)
  - Explicit enumeration avoided
  - Uses bit-vector decision procedure tool UCLID

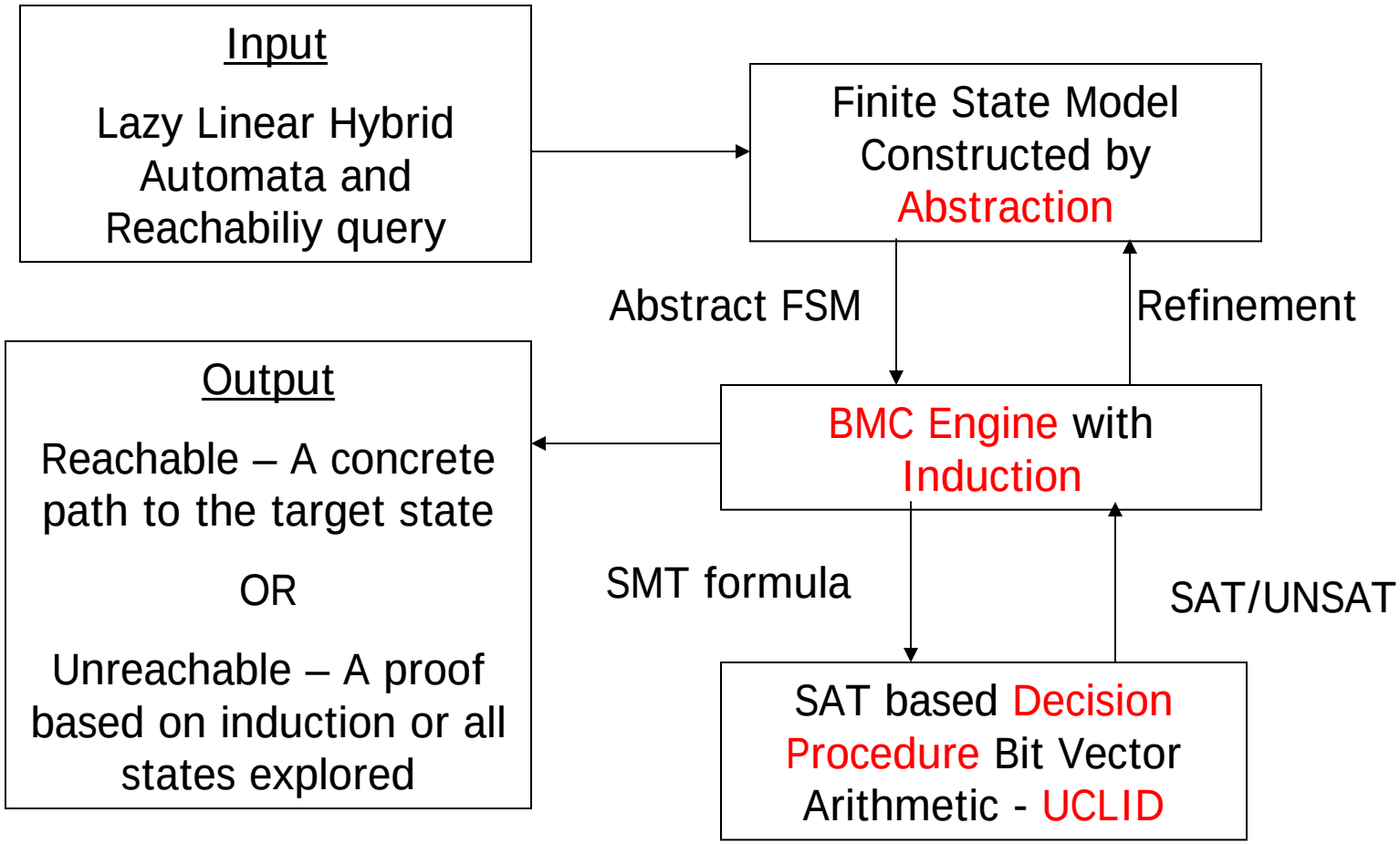
- Abstraction Refinement –
  - Reducing the value K in the above formula by looking at larger quantums Γ
  - Establish a hierarchy of sound abstractions with respect to safety properties.

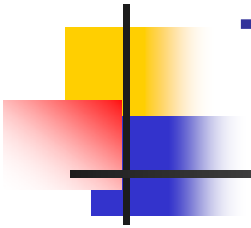


Talk Outline

- Background: Lazy Linear Hybrid Automata (LLHA)
- Overall Tool Flow
- Abstraction Hierarchy for LLHA
- Symbolic BMC of LLHA and K-Induction
- Case Studies and Comparison
- Future Work

Overall Tool Flow



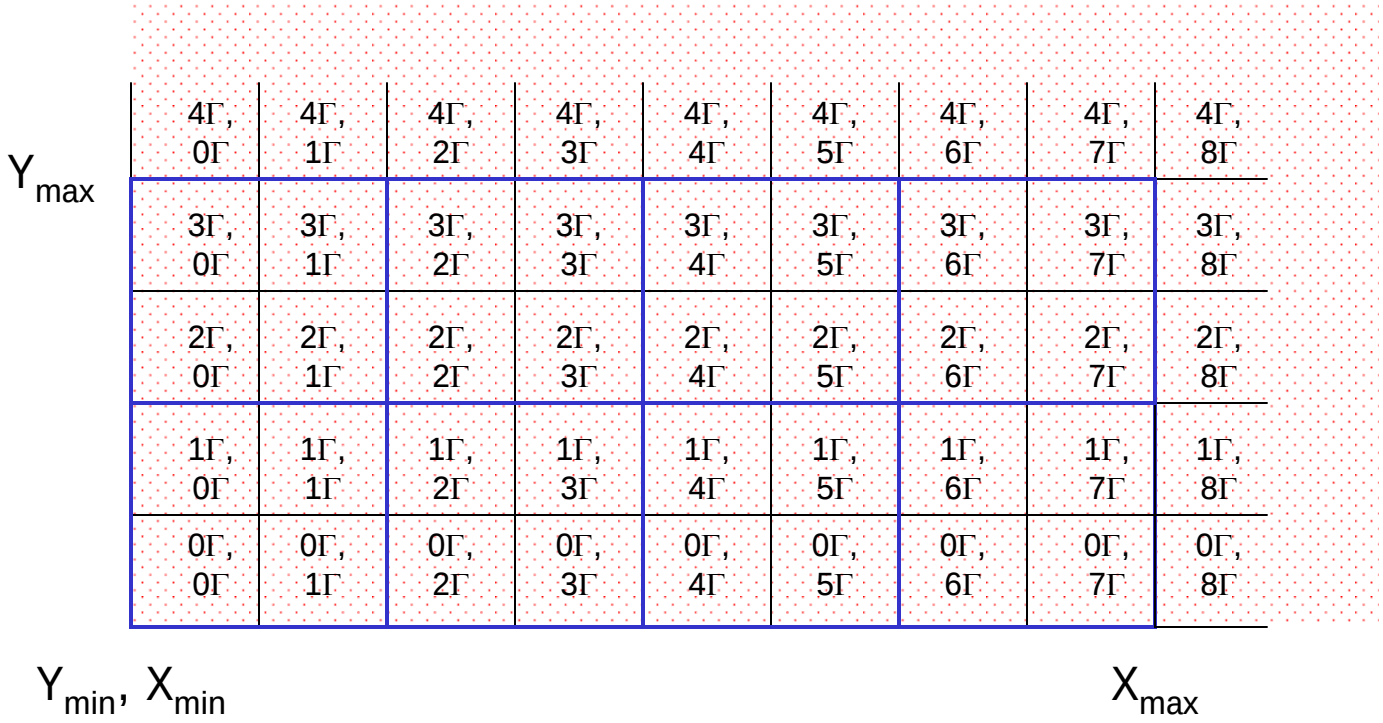


Talk Outline

- Background: Lazy Linear Hybrid Automata (LLHA)
- Overall Tool Flow
- Abstraction Hierarchy for LLHA
- Symbolic BMC of LLHA and K-Induction
- Case Studies and Comparison
- Future Work

Abstraction of States

Use $2^k\Gamma$ instead of Γ for abstraction. The abstraction so created is called k-abstraction



State space of k-abstraction would be

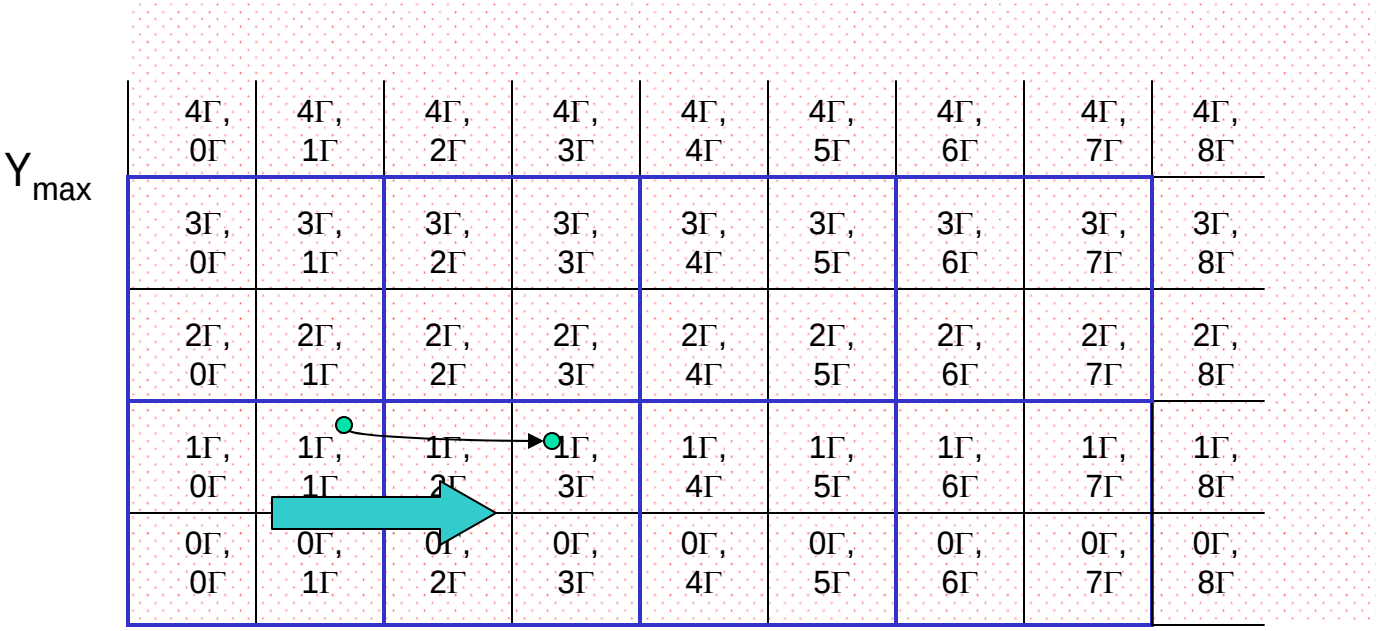
$O(|Q|^4\, 2^{2n}\, (K/2^k)^{3n})$, i.e. decrease by 2^{3kn}

Abstraction of Transitions

Transition due to switches – Guards and invariants are relaxed.

For example,

- $267(x-35)/x \leq 150$, that is, $x \leq 32 \times 267/117$.
- Let Γ be 1 and the abstraction be taken $2^5\Gamma$, $8((k-2)/k) \leq 5$, that is, $k \leq 6$, that is, $x \leq 6 \times 2^5$



Abstraction of Flows

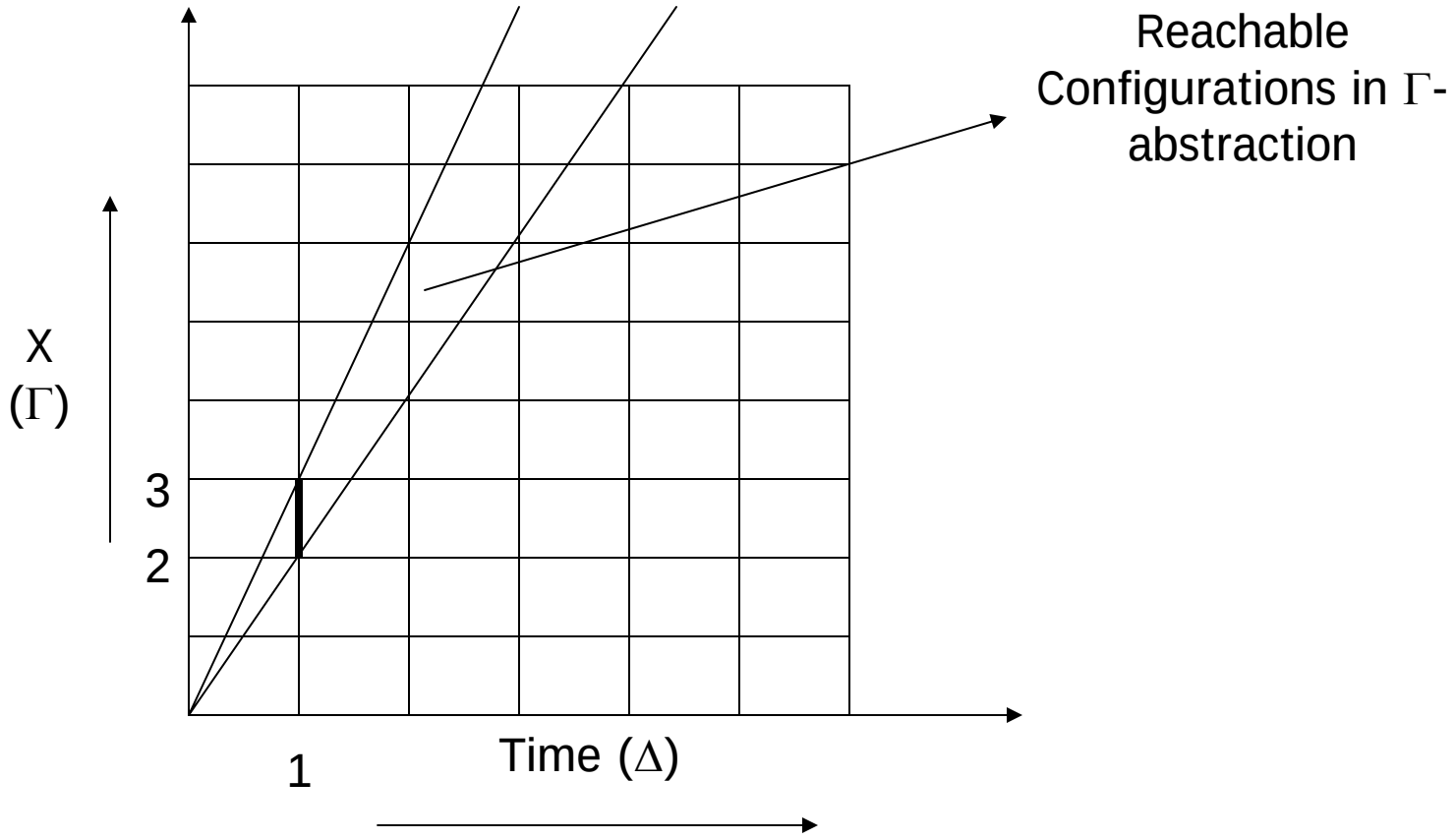
- Key Idea: Adding more flows to preserve simulation
- If rates of change of a variable X is given as the discrete set $R_x = \{r_i\}$
- The rates of change of the variable in k-abstraction is given by

  $R'_x = \cup_i \{ \lfloor r_i/2^k\Gamma \rfloor 2^k\Gamma , \lceil r_i/2^k\Gamma \rceil 2^k\Gamma \}$
- So if the rates of change were [a,a+1.....b], then the abstract rates of change is given by

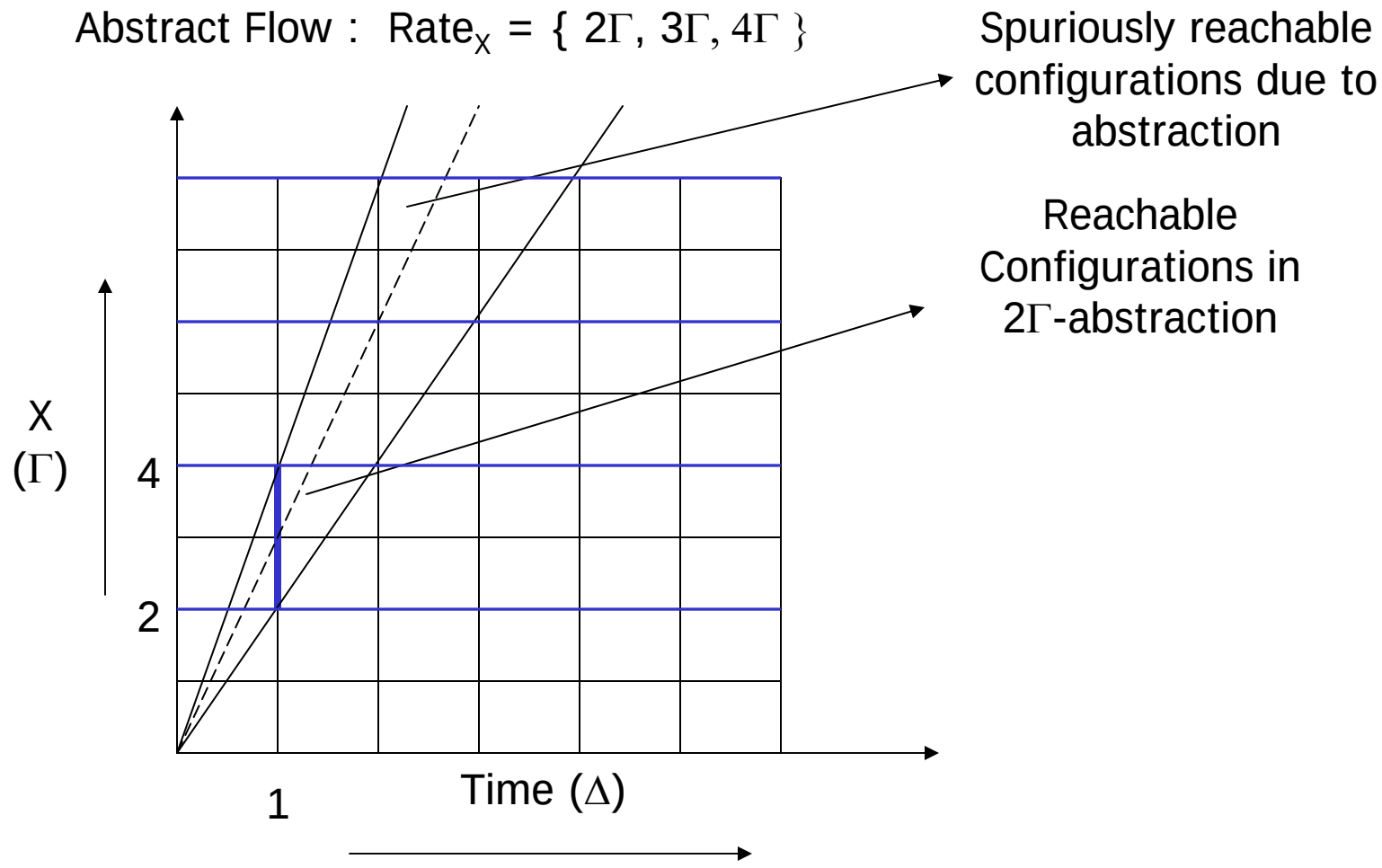
  $[\lfloor a/2^k\Gamma \rfloor 2^k\Gamma \ \ldots\ldots\ldots \ \lceil b/2^k\Gamma \rceil 2^k\Gamma]$

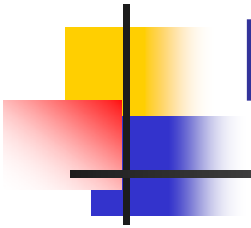
Abstraction of Flows

Flow : $Rate_X = \{ 2\Gamma, 3\Gamma \}$



Abstraction of Flows





Key Results

- Simulation Result:

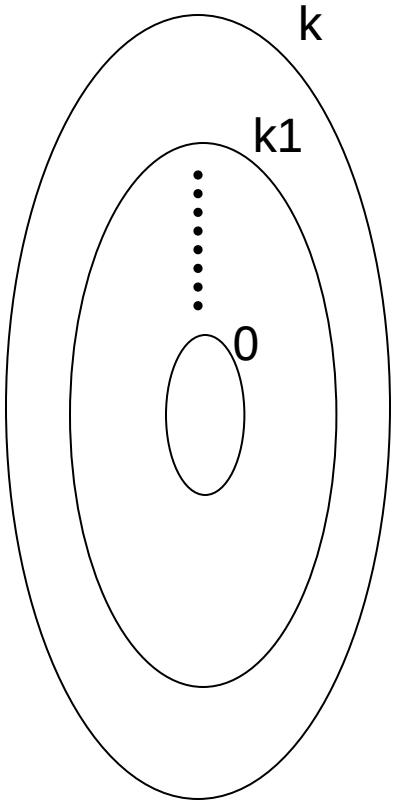
The k-abstraction defined above simulates the lazy linear hybrid automata.

- Hierarchy Result:

For any k>m, k-abstraction simulates the m-abstraction.

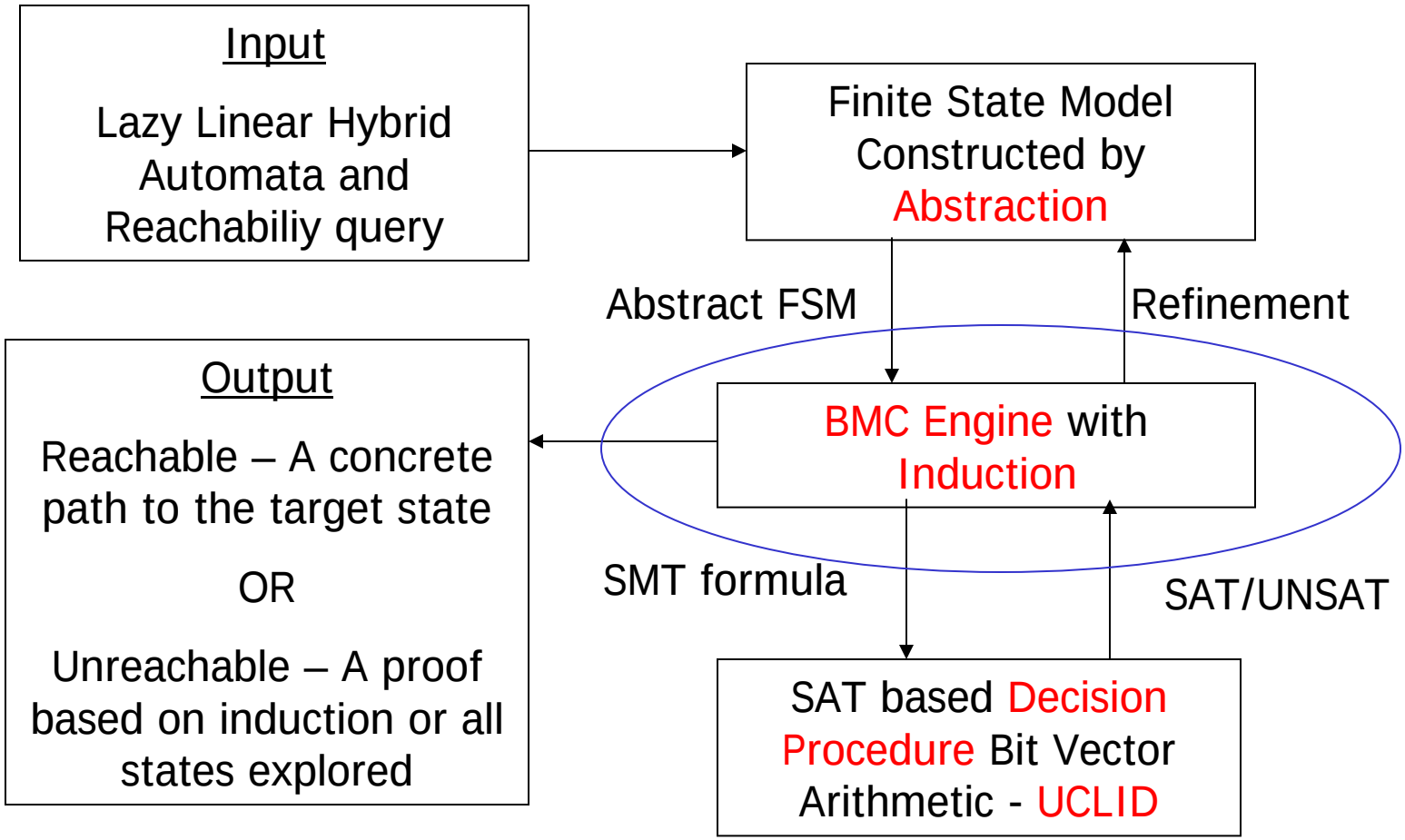
Corollary: If a configuration is not reachable in k-abstraction for some k, it is not reachable in any k'-abstraction for k' < k and is also not reachable in the lazy linear hybrid automata.

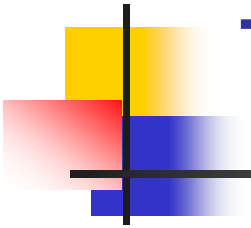
Abstraction-Refinement



- Given an LLHA, chose a “suitable” k, to construct a k-abstraction with tractable state space.
- If the target state is not reachable, then declare safe.
- If the target state is reachable, do counter-example guided refinement.
- So, sequence of considered abstraction would be k,k1,k2,..... where k>k1>k2... So, at most k iterations.
- Repeat till 0-abstraction. If target state is still reachable, then it is also reachable in LLHA since 0-abstraction bisimulates LLHA.

Overall Tool Flow





Talk Outline

- Background: Lazy Linear Hybrid Automata (LLHA)
- Overall Tool Flow
- Abstraction Hierarchy for LLHA
- Symbolic BMC of LLHA and K-Induction
- Case Studies and Comparison
- Future Work

The core of BMC algorithm

Initial State:

$Init(F_0) := (l=v_{start}) \wedge \phi_0(X)$,

where l denoted the control mode and

ϕ_0 is the initial predicate over the continuous variables.

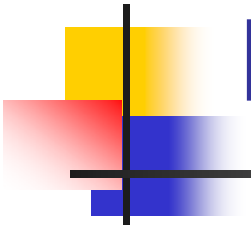
Transition Predicate:

$T(F_{k-1},F_k) := \vee_{(i,j) \in E} G_{ij}(F_{k-1},F_k) \vee \vee_{i \in V} E_i(F_{k-1},F_k)$,

where G_{ij} corresponds to switches and E_i corresponds to evolutions.

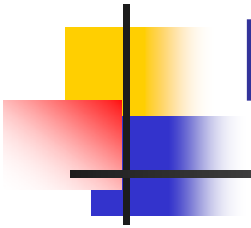
Is $Init(F_0) \wedge \vee_{0 \leq i \leq d} T(F_i,F_{i+1}) \wedge !safe(F_d)$ satisfiable ?

(Is !safe reachable in d-steps)



BMC completeness

- The maximum number of steps we need to run BMC is the length of the longest trajectory – that is the diameter of the graph.
- The diameter of the graph is bounded by the size of the graph – which we know to be finite, say D
- So, after D times, we can declare that the system satisfies the safety property.
- But D can be very huge !!



K-step induction

K-step induction

1. Prove that bad state is not reachable in 0,1,2,.....k steps (BMC till k steps)

2. Prove that if a state is not bad, no bad state is reachable from it in k+1 steps.

This is sufficient to prove that no bad state is reachable.

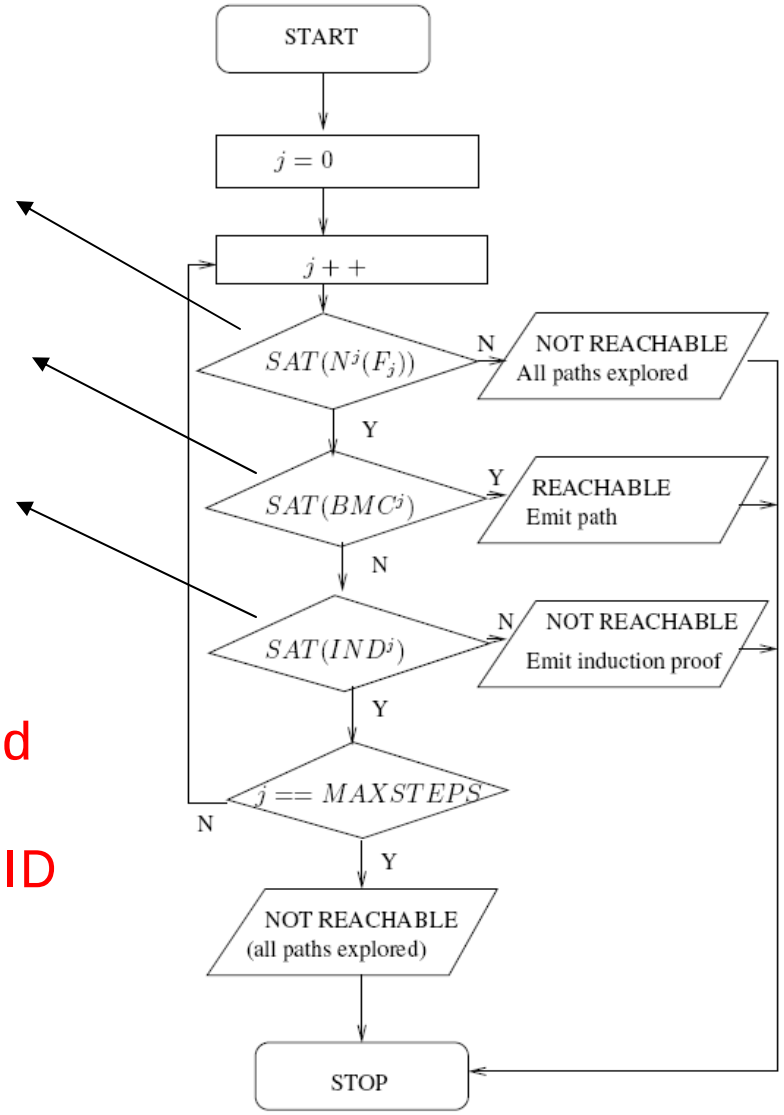
Complete IND-BMC

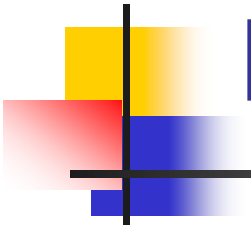
Check if there exists a simple path unexplored ?

Check if the new paths found (with length=j) can reach bad state ?

Check if j-depth induction can be applied ?

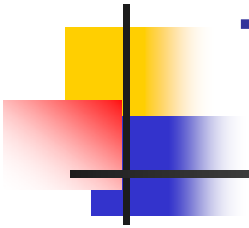
SAT function used in decision boxes correspond to calls to underlying decision procedure - UCLID





Decision Procedure Used

- SMT solver needed to decide the satisfiability of formulae in finite precision bit-vector arithmetic.
- We used Bit-vector UCLID. It can decide any formulae in the Bit-Vector SMT theory (smtlib.org)



Talk Outline

- Background: Lazy Linear Hybrid Automata (LLHA)
- Overall Tool Flow
- Abstraction Hierarchy for LLHA
- Symbolic BMC of LLHA and K-Induction
- Case Studies and Comparison
- Future Work

Case study 1: AHS

Normal cruise speed – [a,f]

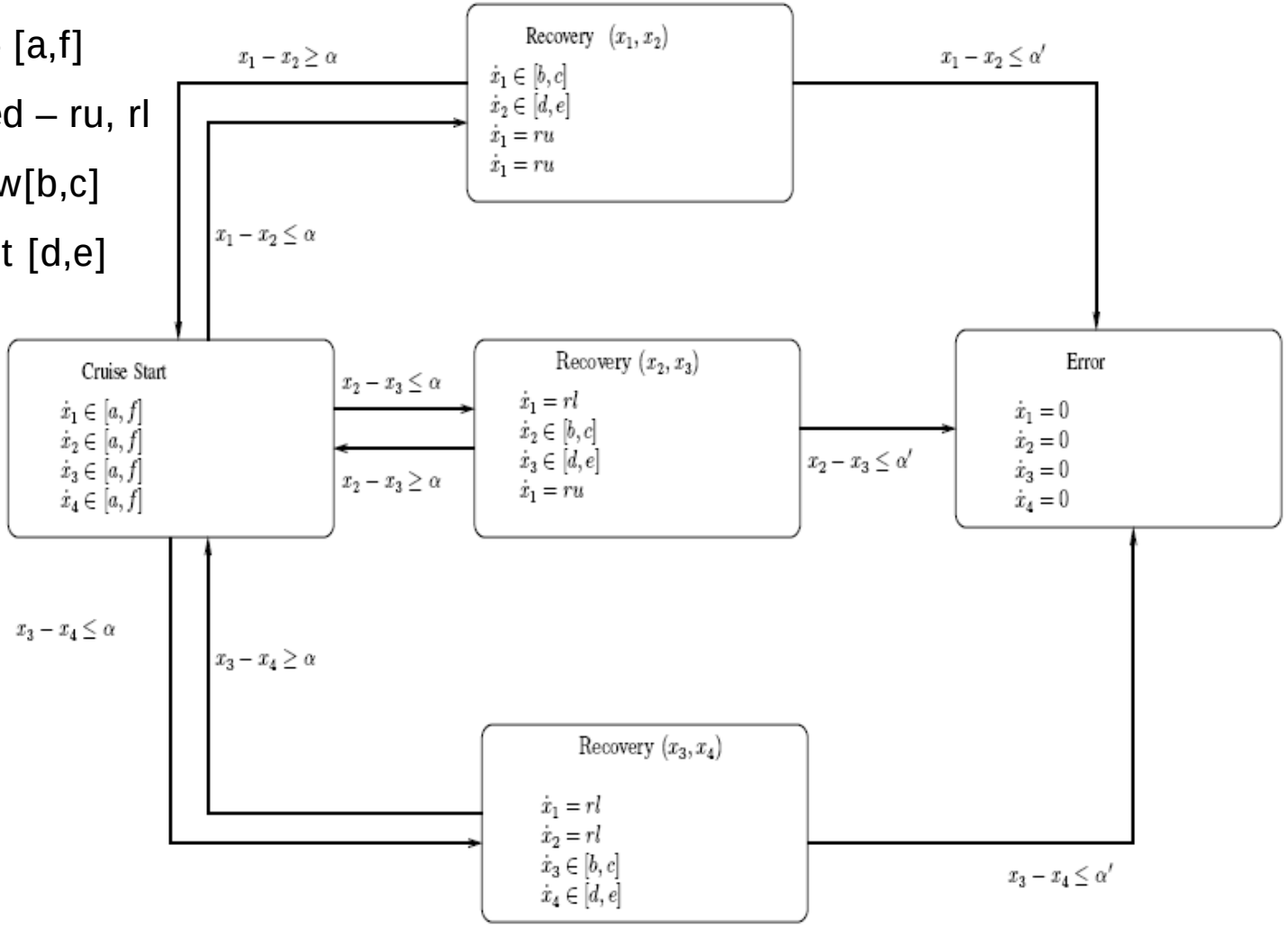
Recovery cruise speed – ru, rl

Recovery speed – slow[b,c]

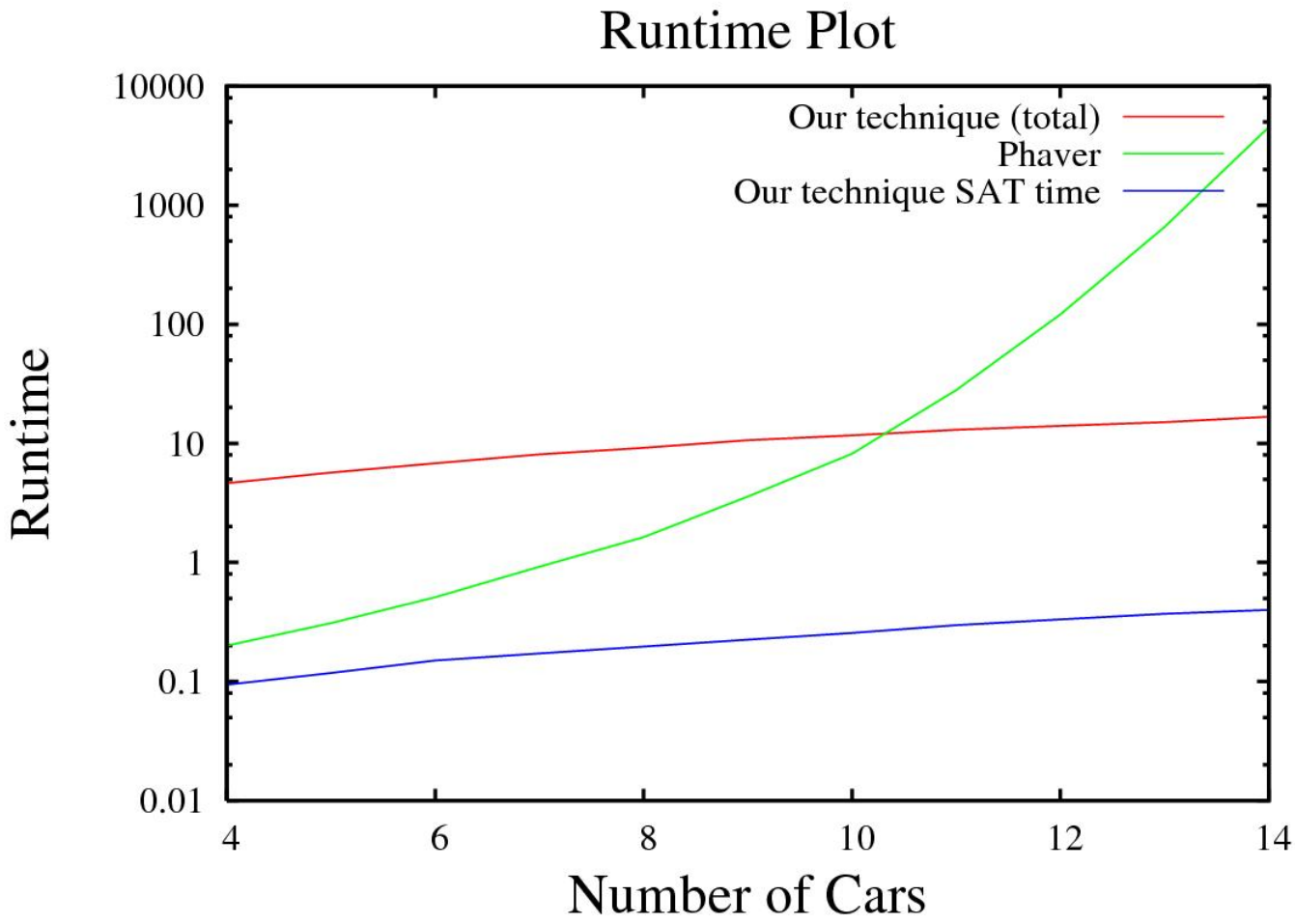
fast [d,e]

Possible collision α

Actual collision α'



Case Study 1: AHS

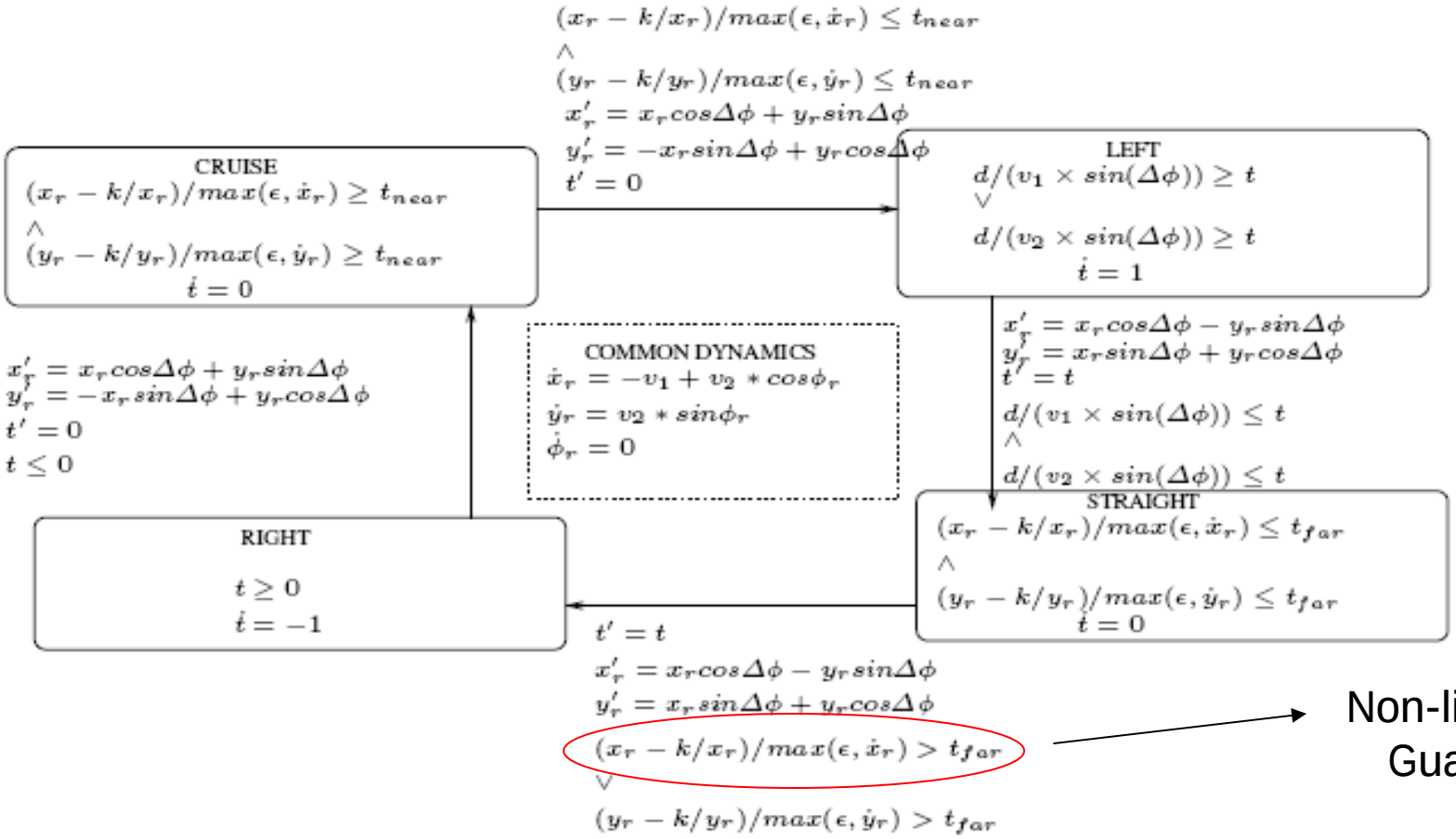


Phaver times out (>10 hours for 15 cars), our technique took less than 2 minutes for 150 cars.

Case Study 2: Simplified TCAS

Model similar to those considered by Tomlin-Pappas

The parameter values obtained from TCAS document by Avionics

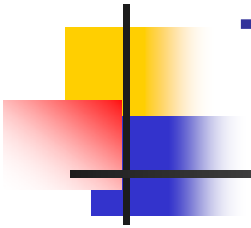


Case Study 2: Simplified TCAS

16-abstraction is 10 times faster than 0-asbtraction

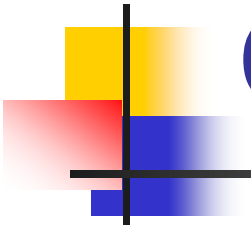
k	Runtime (sec) for different angles		
	$\Delta\phi = 30°$	$\Delta\phi = 45°$	$\Delta\phi = 60°$
0	400.50	181.47	177.49
2	300.01	732.24	253.96
4	904.76	136.39	544.97
8	117.25	101.01	55.45
16	27.45	18.21	17.64

Runtimes of our model checker for TCAS with varying angles.



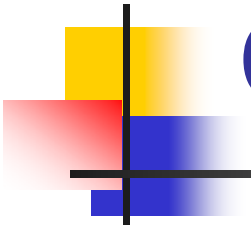
Talk Outline

- Background: Lazy Linear Hybrid Automata (LLHA)
- Overall Tool Flow
- Abstraction Hierarchy for LLHA
- Symbolic BMC of LLHA and K-Induction
- Case Studies and Comparison
- Conclusion



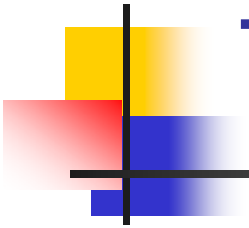
Conclusion

- Established an abstraction hierarchy for LLHA which can be used in a CEGAR framework
- A symbolic implementation of BMC for LLHA enhanced with k-induction
- Demonstration of scalability of our approach on examples like TCAS and AHS.



Ongoing Work

- Selection of an initial level of abstraction
- An intelligent strategy for refinement
- Finding an induction bound if it exists
- Incremental SMT solving for BMC and refinement



Thanks

- Any Questions ?