SAT-Reach : A Bounded Model Checker for Affine Hybrid Systems

Atanu Kundu

Joint work with : Sarthak Das and Dr. Rajarshi Ray

Indian Association for the Cultivation of Science
Kolkata

July 4-5, 2022

Outline :

Introduction

- Hybrid systems consist of both continuous and discrete dynamics.

  Ex: Autonomous car, Robots, Aircraft flight control system, etc.

- Hybrid systems are safety-critical.
- Traditional approach can't handle such systems.
- Bounded Model Checking (BMC) searches for a CE of length at most k.

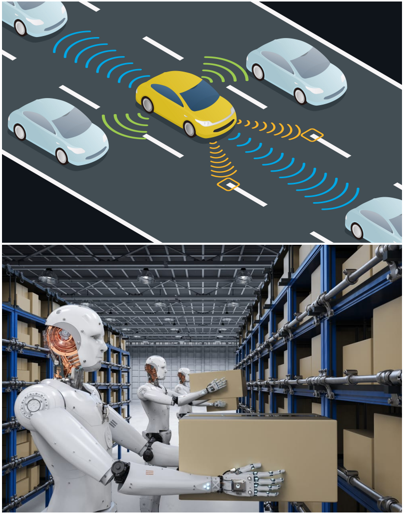

Figure: Safety-critical systems.

Preliminaries

Definition

A *hybrid automaton* ($\mathcal{H}$) is a six tuple ($\mathcal{V}$, $\mathcal{X}$, Inv, $Init$, $Flow$, $Trans$) where:

- $\mathcal{V} = \{v_0, ..., v_\ell\}$ is a finite set of locations or modes of the $\mathcal{H}$.
- $\mathcal{X} = \{x_1, ..., x_n\}$ is a finite set of real-valued variables of the $\mathcal{H}$.
- $Inv(v)$ be a invariant function that maps a location to a subset of $\mathbb{R}^n$.
- $Init(v)$ is a function called the initial set of the location v.
- $Flow(v, x)$ defines the evolution of real-valued variables x in v.
- Each transition $\delta \in Trans$, is a 4-tuple $(v, \mathcal{G}, Asgn, v')$ where:
  - v and v' is the source and target locations.
  - $\mathcal{G}$ and $Asgn$ be the guard and assignment of the transition δ.

- Affine hybrid system is a class of hybrid system.
  - $Flow(v, x) = A_v.x + u$, $u \in \mathcal{U}_v$ where $A_v \in \mathbb{R}^{n\times n}$ and $\mathcal{U}_v \subseteq \mathbb{R}^n$
  - $\mathcal{U}_v$, $Init(v)$, $Inv(v)$ and $\mathcal{G}(\delta)$ are all convex sets and $Asgn(\delta)$ is a linear transformation.

Preliminaries...

- A symbolic state of a hybrid automaton $\mathcal{H}$ is a pair of $(v, x) \in \mathcal{V} \times \mathbb{R}^n$
- Reachability analysis can be performed using two operators:
  - $post_c(v, x)$ denotes the set of reachable states from (v, x) by arbitrary timed transitions.
  - $post_d(v_i, x_i)$ denotes the set of reachable states from (v_i, x_i) by a discrete transition δ.
- A path π is an alternating sequence of locations and transitions:

$$\pi = v_0, e_1, v_1, e_2, \ldots\ldots, v_{unsafe}$$

- A path π is said to be feasible if the forbidden state is reachable.

Motivating Example

- Let $(v_1, \mathcal{C}_1)$ is the initial symbolic state and $(v_{17}, \mathcal{C}_{17})$ is the forbidden symbolic state in the verification problem with bound of analysis 7.
- BFS/DFS-based exploration computes 49 $post_c$ operations in worst-case and 17 in the bast case.

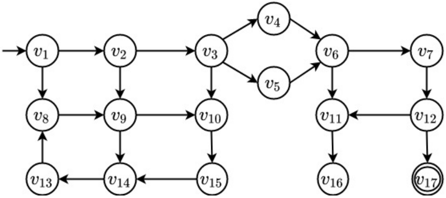

Figure: Discrete structure of an HA model.

- SAT-based path-guided exploration computes only 13 $post_c$ operations in the worst case and 9 in the best case.

- Our aim is to reduce the number of $post_c$ computations so that we can deduce the safety of the given verification problem faster.

Proposed Method

- **Input:** Affine Hybrid automata $\mathcal{H}$ and bound of analysis k.
- **Output:** Terminates with one of the followings:
  - SAFE.
  - UNSAFE + Concrete CE.
  - UNKNOWN.
- It combines -
  - SAT-based path enumeration.
  - Pathwise Reachability analysis.
  - Search for CE.
- We propose -
  - Path pruning.
  - Reachability optimization.

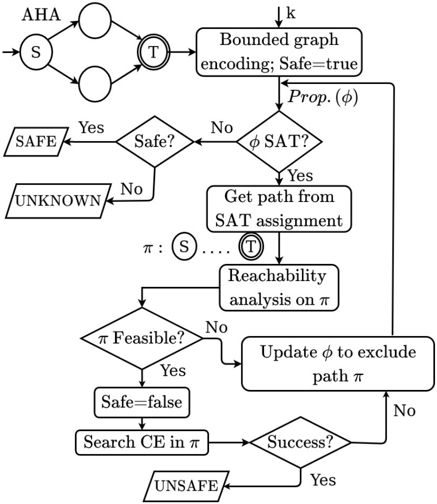

Figure: SAT-assisted BMC procedure.

SAT-based path enumeration

- **Goal:** Enumerate all paths from the AHA that satisfy the given configurations.
- Graph is encoded using two propositional logic formulas:
  - $\varphi_{\mathcal{H}}^{k}$ for retrieving a sequence of locations L.
  - $\Phi_{\mathcal{H}}^{k}(L)$ for retrieving a path π.
- Four basic constraints are used to get L :
  - **Initial constraint :**

$$\varphi_{Init}(\mathcal{V}_{Init}) := \bigvee_{v_i \in \mathcal{V}_{Init}} (\varphi_{Init}(v_i))$$

where $\varphi_{Init}(v_0) := v_0^0 \wedge \varphi_{Excl}(v_0^0)$

  - **Exclusivity constraint:**

$$\varphi_{Excl}(v_i^j) := \left(v_i^j \implies \bigwedge_{v_m \in \mathcal{V} \wedge v_m \neq v_i} (\neg v_m^j) \right)$$

  - **Transition constraint:**

$$\varphi_{Trans}(v_i^j) := \left(v_i^j \implies \bigvee_{(v_i, v_m) \in E} (v_m^{j+1}) \right)$$

SAT-based path enumeration

- **Destination constraint:**

$$\varphi_{Dest}(\mathcal{V}_{unsafe}, j) := \bigvee_{v_i \in \mathcal{V}_{unsafe}} (v_i^j)$$

- Another constraint for retrieving a path $\pi = v_0, e_1, v_1, e_2 \ldots v_k$.
  - **Parallel constraint:**

$$\Phi_{\mathcal{H}}^k(L) = \bigwedge_{0 \le i < (len(L)-1), 1 \le j \le k} (\Phi_{Para\_trans}(v_i, v_{i+1}, j))$$

  where $\Phi_{Para\_trans}(v_i, v_{i+1}, j) = \bigvee_{p \in Tid(v_i, v_{i+1})} (e_p^j)$

- **Negation constraint:** $\Phi_{Neg}(\pi) := (\neg e_1^1 \vee \neg e_2^2 \vee \ldots\ldots \vee \neg e_k^k)$

$$\varphi_{Neg}(L) := (\neg v_0^0 \vee \neg v_1^1 \vee \cdots \vee \neg v_k^k)$$

- we will discuss some constraints on path pruning section.

Reachability Analysis over a path

- Consider a path $\pi = v_0, e_1, v_1, e_2 \dots v_{unsafe}$ returned by the previous routine.
- Goal: The path π is feasible or not.

Definition

An *execution* σ of a hybrid automaton $\mathcal{H}$ is an alternating sequence of timed and discrete transitions:

$$\sigma : (v_0, x_0) \xrightarrow{\tau_0} (v_0, y_0) \xrightarrow{\delta_0} (v_1, x_1) \xrightarrow{\tau_1}, \dots, \xrightarrow{\delta_{k-1}} (v_k, x_k) \xrightarrow{\tau_k} (v_k, y_k)$$

such that (1) $x_0 \in Init(v_0)$, (2) $y_i \in \mathcal{G}(\delta_i)$, $x_{i+1} = Asgn(\delta_i)(y_i)$ for every $0 \leq i < k$.

- A state (v, x) is reachable if there is an *execution*.
- Reachable states are computed for a location v_i by the union of convex sets: $\Omega = \Omega_0, \Omega_1, ..., \Omega_{N-1}$

Reachability Analysis over a path

- We adopt the idea of computing reachable states from [1].
- The convex set Ω_j is computed by the following formula over $[j\Delta t, (j+1)\Delta t]$

$$\begin{aligned} \Omega_j &= e^{A(j\Delta t)}\Omega_0 \oplus \Psi_j \\ \Psi_{j+1} &= \Psi_j \oplus e^{A(j\Delta t)}\Psi_{\Delta t} \end{aligned} \quad (1)$$

- Reachable states that take the transition (δ_i) must satisfy the

$$\begin{aligned} \Omega' &= \Omega \cap \mathcal{G}(\delta) \cap Inv(v_i) \\ \Omega_{new} &= Asgn(\delta)(\Omega') \cap Inv(v_{i+1}) \end{aligned} \quad (2)$$

- This Ω_{new} forms the initial set of the location v_{i+1}.

Reachability Analysis algorithm

Input: $\pi = v_0, e_1, v_1, e_2, v_2, \ldots.., e_k, v_k$, and hybrid automaton $\mathcal{H}$.

Output: Returns true with the path π when π is feasible, otherwise returns false with the smallest infeasible sub_path.

Initialization: $sub\_path \leftarrow v_0$; $C_0 = Init(v_0)$

for $i \leftarrow 0$ to $k - 1$ **do**
    $\mathcal{F} = post_c(v_i, C_i)$
    if $(i = k)$ **then**
        return π, true
    end if
    $sub\_path \leftarrow sub\_path$.push$(v_{i+1})$
    if $(\mathcal{F} \cap Inv(v_i) \cap \mathcal{G}(e_{i+1}) \neq \emptyset)$ **then**
        $\mathcal{A} = Asgn(\delta(e_{i+1}))(\mathcal{F} \cap Inv(v_i) \cap \mathcal{G}(e_{i+1}))$
        if $(\mathcal{A} \cap Inv(v_{i+1}) \neq \emptyset)$ **then**
            $C_{i+1} = \mathcal{A} \cap Inv(v_{i+1})$
        else
            return *sub_path*, false
        end if
    else
        return *sub_path*, false
    end if
end for

Reachability Optimization

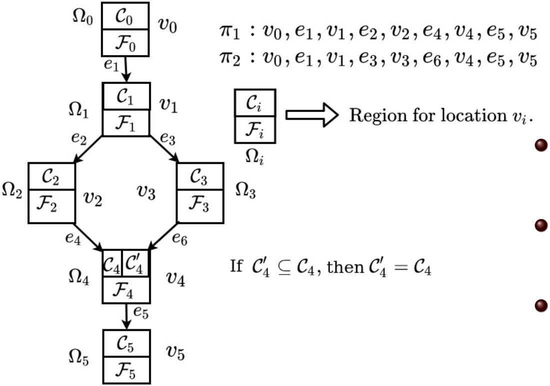

Figure: Demonstration of our optimization idea.

- Ω_i is the union of $\mathcal{C}_i$ and $\mathcal{F}_i$.
- $(v_i, \mathcal{C}_i)$ is calculated using equ. 3.
- $\mathcal{F}_i = post_c((v_i, \mathcal{C}_i))$

$$(v_i, \mathcal{C}_i) = post_d(post_c((v_{i-1}, \mathcal{C}_{i-1})), (v_{i-1}, v_i)) \quad (3)$$

Optimization Rules

$$\frac{}{(v_0, \mathcal{C}_0),\ \mathcal{C}_0 = Init(v_0)}$$

$$\frac{\textit{initial-to-flow}\big[(v_i, \mathcal{C}_i)\big] = \emptyset}{\mathcal{F}_i = post_c((v_i, \mathcal{C}_i)),\ \textit{initial-to-flow}\big[(v_i, \mathcal{C}_i)\big] = \mathcal{F}_i}$$

$$\frac{\textit{initial-to-flow}\big[(v_i, \mathcal{C}_i)\big] \neq \emptyset}{\mathcal{F}_i = \textit{initial-to-flow}\big[(v_i, \mathcal{C}_i)\big]}$$

$$\frac{\textit{flow-trans-to-init}\big[(\mathcal{F}_i, \delta)\big] = \emptyset, \text{ where } \delta = (v_i, \mathcal{G}, Asgn, v_{i+1})}{(v_{i+1}, \mathcal{C}_{i+1}) = post_d(\mathcal{F}_i, (v_i, v_{i+1}))}$$

$$\frac{\textit{flow-trans-to-init}\big[(\mathcal{F}_i, \delta)\big] \neq \emptyset, \text{ where } \delta = (v_i, \mathcal{G}, Asgn, v_{i+1})}{(v_{i+1}, \mathcal{C}_{i+1}) = \textit{flow-trans-to-init}\big[(\mathcal{F}_i, \delta)\big]}$$

Optimization Rules...

$$\frac{\begin{array}{c}\exists(v_{i+1},\mathcal{C}'_{i+1})\Big[\big(\textit{initial-to-flow}\ [(v_{i+1},\mathcal{C}'_{i+1})] \neq \emptyset \wedge (\mathcal{C}_{i+1} \subseteq \mathcal{C}'_{i+1})\big) \bigwedge \\ \forall(v_{i+1},\mathcal{C}^*_{i+1})\Big(\big(\textit{initial-to-flow}[(v_{i+1},\mathcal{C}^*_{i+1})] \neq \emptyset\big) \implies \\ \Big\{\big\{(\mathcal{C}'_{i+1} \not\subseteq \mathcal{C}^*_{i+1}) \wedge (\mathcal{C}'_{i+1} \not\supseteq \mathcal{C}^*_{i+1})\big\} \vee \big\{(\mathcal{C}_{i+1} \subseteq \mathcal{C}^*_{i+1}) \implies (\mathcal{C}'_{i+1} \subseteq \mathcal{C}^*_{i+1})\big\}\Big\}\end{array}}{\mathcal{C}_{i+1} = \mathcal{C}^{'}_{i+1},\ \textit{flow-trans-to-init}\ [(\mathcal{F}_i,\delta)] = (v_{i+1},\mathcal{C}_{i+1})}$$

$$\frac{\forall(v_{i+1},\mathcal{C}'_{i+1})\Big[\big(\textit{initial-to-flow}[(v_{i+1},\mathcal{C}'_{i+1})] \neq \emptyset \wedge (\mathcal{C}_{i+1} \not\subseteq \mathcal{C}'_{i+1})\big)\Big]}{\textit{flow-trans-to-init}\big[(\mathcal{F}_i,\delta)\big] = (v_{i+1},\mathcal{C}_{i+1})}$$

Searching for a CE

- Goal: Splicing trajectory segments to get a concrete CE.

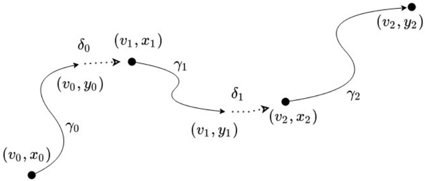

Figure: An example of a segmented trajectory having three trajectory segments.

- We used trajectory splicing [2] algorithm to find a concrete counterexample.

Definition (Segmented Trajectory)

A segmented trajectory (Γ) of an automaton is a finite sequence of trajectory segments γ_i given as:

$$\Gamma = \begin{pmatrix} \gamma_0 : & (v_0, x_0) \xrightarrow{\tau_0} (v_0, y_0) \\ \gamma_1 : & (v_1, x_1) \xrightarrow{\tau_1} (v_1, y_1) \\ \vdots & \ldots \\ \gamma_k : & (v_k, x_k) \xrightarrow{\tau_k} (v_k, y_k) \end{pmatrix} \quad (4)$$

where $(v_i, x_i) \xrightarrow{\tau_i} (v_i, y_i)$ is a timed transition in the location $v_i \in \mathcal{V}$, for all $0 \leq i \leq k$.

Searching for a CE

- Splicing these trajectory segments using a nonlinear optimization problem.

$$\min_{x_0,\ldots,x_k,\tau_0,\ldots,\tau_k} \sum_{i=0}^{k-1} \mathrm{COST}(\gamma_i, \gamma_{i+1}) \tag{5}$$

subject to:

$$\mathrm{COST}(\gamma_i, \gamma_{i+1}) = \mathrm{dist}(Asgn(\delta_i)(y_i), x_{i+1}) \tag{6}$$

$$x_0 \in Init(v_0) \tag{7}$$

$$x_i \in \mathcal{C}'_i, \qquad \forall i : 1 \leq i \leq k \tag{8}$$

$$y_i \in \mathcal{G}(\delta_i) \cap \mathcal{C}_i, \qquad \forall i : 0 \leq i \leq k-1 \tag{9}$$

$$y_k \in \mathcal{C}_k \cap \mathcal{C}_{unsafe} \tag{10}$$

$$0 \leq \tau_i \leq T, \qquad \forall i : 0 \leq i \leq k \tag{11}$$

- dist() is the euclidean distance between end point of γ_i and starting point of γ_{i+1}.

Path Pruning

- We prune the number of candidate paths using two *overall negation constraints*:
  - *overall negation constraint* for the set of paths.

$$\Phi_{Neg}^{\Pi} := \bigwedge_{\pi \in \Pi} \big(\Phi_{Neg}(\pi)\big)$$

    - Π is a set of infeasible paths π that are not to be further enumerated.
  - *overall negation constraint* for the set of sequences of locations.

$$\varphi_{Neg}^{SL} := \bigwedge_{L \in SL} \big(\varphi_{Neg}(L)\big)$$

    - SL is a set of sequence of locations L that are not to be further enumerated.

Results for the unsafe instances

Benchmark	#Locs	#Dims	BMC Bnd (k)	dReach Time (secs)		BFS-Reach					SAT-Reach					
				Time	#Paths	#Abs	Time (secs)			#postC	#Abs	Time (secs)			#Paths	#postC
							RA	Opt.	Fal.			RA	Opt.	Fal.		
Platoon	2	10	≤ 1	Timeout	1	1	4.29	0.54	4.83	2	1	4.09	0.53	4.62	1	2
Oscillator	4	3	≤ 3	0.96	1	1	0.15	0.10	0.25	4	1	0.19	0.10	0.29	1	4
F_oscillator_4	4	7	≤ 3	5.65	1	1	0.37	0.26	0.63	4	1	0.39	0.26	0.66	1	4
F_oscillator_8	4	11	≤ 3	22.32	1	1	0.66	1.1	1.76	4	1	0.69	1.12	1.81	1	4
F_oscillator_16	4	19	≤ 3	20.88	1	1	1.69	3.45	5.14	4	1	1.78	3.47	5.25	1	4
F_oscillator_32	4	35	≤ 3	128.4	1	1	7.82	15.82	23.64	4	1	7.77	15.88	23.65	1	4
F_oscillator_64	4	67	≤ 3	Timeout	1	1	-	-	UKW	4	1	-	-	UKW	1	4
Two_tank_U1	4	3	≤ 3	129.47	2	1	0.11	0.05	0.16	4	1	0.14	0.05	0.19	3	4
NAV_U1	9	4	≤ 6	1.68	22	1	1.16	0.14	1.30	29	1	0.27	0.16	0.43	4	8
NAV_U2	9	4	≤ 7	6.71	101	1	2.49	0.15	2.64	46	1	0.47	0.15	0.62	14	18
NAV_U3	9	5	≤ 3	0.87	1	1	1.78	0.91	2.69	10	1	0.28	0.14	0.42	1	4
NAV_U4	9	5	≤ 2	1.13	1	1	0.51	0.54	1.05	4	1	0.17	0.08	0.25	1	3
NAV_U5	27	6	≤ 6	91.57	34	1	1.95	0.3	2.25	29	1	0.53	0.31	0.84	7	9
NAV_U6	27	6	≤ 5	42.86	8	1	1.13	0.22	1.35	17	1	0.36	0.0.20	0.56	4	6
NAV_U7	81	7	≤ 1	26.7	1	1	0.09	0.03	0.12	2	1	0.10	0.02	0.12	1	2
NAV_U8	81	7	≤ 8	Timeout	307	1	11.13	3.52	14.65	119	1	1.8	3.44	4.62	34	16
NAV_U9	81	7	≤ 6	1376.71	103	1	2.08	1.53	3.61	31	1	0.47	1.44	1.91	4	7
NAV_U10	81	7	≤ 12	Timeout	337	0	-	-	Timeout	3069	1	2.57	11.90	14.47	90	38
NAV_U11	81	7	≤ 9	Timeout	266	1	46.65	10.46	57.11	271	1	1.27	7.16	8.43	36	19
NAV_U12	81	7	≤ 11	Timeout	297	5	2545	254.77	2800.09	2439	2	2.94	74.26	77.20	103	45
NAV_U13	81	7	≤ 10	Timeout	277	1	171.69	8.05	179.74	676	2	1.83	66.85	68.68	57	28
NAV_U14	625	5	≤ 2	1.54	1	1	1.63	1.30	2.93	5	1	2.46	1.21	3.67	1	3
NAV_U15	625	5	≤ 12	OOM	-	0	-	-	Timeout	1386	46	-	-	Timeout	149	409
NAV_U16	625	5	≤ 9	Timeout	27	0	-	-	Timeout	1385	23	45.49	1409.79	1455.28	47	147
NAV_U17	625	5	≤ 10	Timeout	36	0	-	-	Timeout	1397	18	44.15	1104.14	1148.29	27	104
NAV_U18	625	7	≤ 10	30.42	1	1	3.32	14.63	17.95	27	1	4.27	14.23	18.5	1	11
NAV_U19	625	7	≤ 12	302.91	1	1	4.23	23.33	27.56	33	1	5.63	22.69	28.32	1	13
NAV_U20	625	7	≤ 13	Timeout	1	1	4.57	31.56	36.13	36	1	6.41	31.08	37.49	1	14
NAV_U21	625	7	≤ 15	1802.17	1	1	5.26	45.24	50.5	42	1	8.13	44.82	52.95	1	16
NAV_U22	625	7	≤ 17	Timeout	1	1	5.85	64.95	70.80	48	1	9.96	63.47	73.43	1	18
NAV_U23	625	7	≤ 18	Timeout	1	1	-	-	UKW	54	1	-	-	UKW	1	19
NAV_U1_P	9	4	≤ 6	1.16	28	1	1.30	0.15	1.45	29	1	0.31	0.15	0.46	9	8

Generated Counterexample

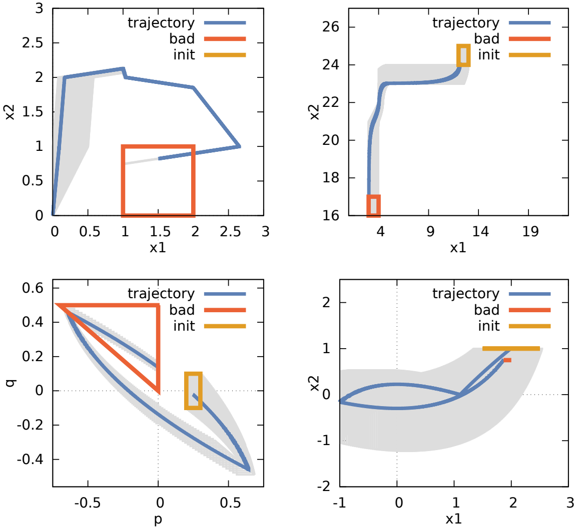

Figure: Counterexamples generated by SAT-Reach on different instances.

Results for the safe instances

Benchmark	#Locs	#Dims	BMC Bound (k)	dREACH		BFS-REACH		SAT-REACH		
				Time (secs)	#Paths	Time (secs)	#postC	Time (secs)	#Paths	#postC
Two_tank_S1	4	3	≤ 12	137.09	458	0.72	13	0.53	8	11
NAV_S1	9	4	≤ 10	543.17	4650	31.15	234	3.09	110	127
ACCS03_1	9	9	≤ 7	Timeout	4	Timeout	753	659.16	5461	5461
ACCS03_2	9	9	≤ 7	Timeout	4	Timeout	755	664.73	5461	5461
NAV_S2	27	6	≤ 6	292.04	156	2.41	33	0.69	15	16
NAV_S3	27	6	≤ 6	634.72	312	2.41	33	0.69	21	21
NAV_S4	27	6	≤ 19	Timeout	1074	Timeout	2060	1107.29	8103	5816
NAV_S5	27	6	≤ 19	Timeout	1122	Timeout	2064	412.70	5019	2700
NAV_S6	81	7	≤ 15	Timeout	341	Timeout	3072	1616.98	10231	8792
NAV_S7	81	7	≤ 12	Timeout	327	Timeout	3069	6.03	254	121
NAV_S8	81	7	≤ 15	Timeout	279	Timeout	3073	2261.32	11741	11018
NAV_S9	81	7	≤ 14	Timeout	329	Timeout	3062	197.72	3417	2927
NAV_S10	81	7	≤ 14	Timeout	267	Timeout	3062	137.49	2667	1655
NAV_S11	81	7	≤ 15	Timeout	246	Timeout	3064	873.14	8114	4591
ACCS05_1	81	13	≤ 4	OOM	1	Timeout	566	458.84	4369	4369
ACCS05_2	81	13	≤ 4	OOM	1	Timeout	565	455.48	4369	4369
NAV_S12	625	5	≤ 15	OOM	0	Timeout	1395	158.79	336	511
NAV_S13	625	5	≤ 15	OOM	1	Timeout	1396	301.20	779	671
NAV_S14	625	5	≤ 20	OOM	0	Timeout	1394	137.56	68	112
NAV_S15	625	7	≤ 16	0.60	1	5.71	47	8.78	1	12
NAV_S16	625	7	≤ 10	0.275	1	3.53	29	3.77	1	4

Explored Reachable States

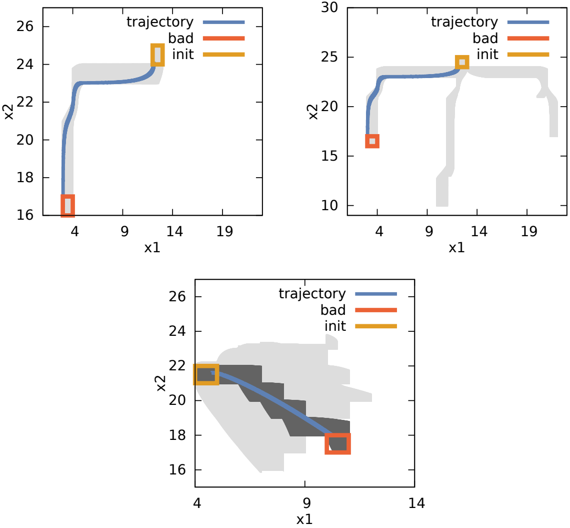

Figure: Reachable region and CE for different instances in different tools.

Comparison with Flow*

Benchmarks	#Locs	#Dims	BMC Bound	Safety Result					
				FLOW*		SAT-REACH			
				Time (secs)	Result	Time (secs)			Result
						RA	Opt	Total	
Platoon	2	10	1	0.0045	UNSAFE	4.04	0.51	4.56	UNSAFE
Oscillator	4	3	3	0.0004	UNSAFE	0.19	0.11	0.30	UNSAFE
F_oscillator_32	4	35	3	0.1900	UNSAFE	7.77	15.88	23.65	UNSAFE
F_oscillator_64	4	67	3	3.5920	UNSAFE	38.78	81.09	119.87	UNKNOWN
Two_tank_U1	4	3	3	0.0037	UNSAFE	0.13	0.05	0.18	UNSAFE
NAV_U2	9	4	7	0.6420	UNKNOWN	0.49	0.15	0.64	UNSAFE
NAV_U3	9	5	3	0.0010	UNSAFE	0.28	0.14	0.42	UNSAFE
NAV_U5	27	6	6	0.80.6293	UNKNOWN	0.47	0.30	0.77	UNSAFE
NAV_U10	81	7	12	170.4699	UNKNOWN	2.55	11.89	14.44	UNSAFE
NAV_U11	81	7	9	12.2957	UNKNOWN	1.28	7.13	8.41	UNSAFE
NAV_U12	81	7	11	64.6145	UNKNOWN	2.88	73.84	76.72	UNSAFE
NAV_U13	81	7	10	27.8765	UNKNOWN	1.75	66.65	68.40	UNSAFE
NAV_U14	625	5	2	16.3594	UNKNOWN	2.36	1.19	3.55	UNSAFE
NAV_U15	625	5	12	-	OOM	-	-	-	Timeout
NAV_U16	625	5	9	-	OOM	44.47	1407.59	1452.06	UNSAFE
NAV_U17	625	5	10	-	OOM	42.37	1102.84	1145.21	UNSAFE
NAV_U22	625	7	17	26.6224	UNKNOWN	9.46	60.43	69.79	UNSAFE
NAV_U23	625	7	18	26.5592	UNKNOWN	11.53	64.94	76.47	UNKNOWN
NAV_S1	9	4	10	2.2723	UNKNOWN	3.05	0	3.05	SAFE
NAV_S2	27	6	6	0.6325	UNKNOWN	0.65	0	0.65	SAFE
NAV_S3	27	6	6	0.6335	UNKNOWN	0.68	0	0.68	SAFE
NAV_S4	27	6	19	-	OOM	1057.82	0	1057.82	SAFE
NAV_S5	27	6	19	-	OOM	386.67	0	386.67	SAFE
NAV_S6	81	7	15	-	OOM	1547	0	1547	SAFE
NAV_S7	81	7	12	164.9963	UNKNOWN	5.82	0	5.82	SAFE

Comparison with SpaceEx

Benchmarks	#Locs	#Dims	#Trans	BMC Bound	Safety Result							
					SpaceEx			SAT-Reach				
					Time (secs)	#*PostC*	Result	Time (secs)			#*PostC*	Result
								RA	Opt	Total		
Platoon	2	10	2	1	3.32	2	UNSAFE	4.04	0.51	4.56	2	UNSAFE
F_oscillator_32	4	35	4	3	1.80	4	UNSAFE	7.77	15.88	23.65	4	UNSAFE
F_oscillator_64	4	67	4	3	6.88	4	UNSAFE	38.78	81.09	119.87	4	UNKNOWN
Two_tank_U1	4	3	7	3	0.17	4	UNSAFE	0.13	0.05	0.18	4	UNSAFE
NAV_U1	9	4	24	6	0.26	23	UNSAFE	0.27	0.16	0.43	8	UNSAFE
NAV_U2	9	4	24	7	0.34	32	UNSAFE	0.49	0.15	0.64	18	UNSAFE
NAV_U3	9	5	20	3	0.85	14	UNSAFE	0.28	0.14	0.42	4	UNSAFE
NAV_U4	9	5	20	2	0.60	8	UNSAFE	0.17	0.08	0.25	3	UNSAFE
NAV_U5	27	6	108	6	0.39	25	UNSAFE	0.47	0.30	0.77	9	UNSAFE
NAV_U6	27	6	108	5	0.29	17	UNSAFE	0.36	0.20	0.56	6	UNSAFE
NAV_U10	81	7	432	12	42.40	1262	UNSAFE	2.55	11.89	14.44	38	UNSAFE
NAV_U11	81	7	432	9	7.05	262	UNSAFE	1.28	7.13	8.41	19	UNSAFE
NAV_U12	81	7	432	11	25.11	802	UNSAFE	2.88	73.84	76.72	45	UNSAFE
NAV_U13	81	7	432	10	12.87	480	UNSAFE	1.75	66.65	68.40	28	UNSAFE
NAV_U15	625	5	2392	12	1877.48	4181	UNSAFE	-	-	-	409	Timeout
NAV_U16	625	5	2392	9	341.12	935	UNSAFE	44.47	1407.59	1452.06	147	UNSAFE
NAV_U17	625	5	2392	10	580.80	1581	UNSAFE	42.37	1102.84	1145.21	104	UNSAFE
NAV_U22	625	7	227	17	3.73	50	UNSAFE	9.46	60.43	69.79	18	UNSAFE
NAV_U23	625	7	227	18	4.40	54	UNSAFE	11.53	64.94	76.47	19	UNKNOWN
Two_tank_S1	4	3	7	12	0.81	13	SAFE	0.53	0	0.53	11	SAFE
NAV_S1	9	4	24	10	0.82	76	SAFE	3.05	0	3.05	127	SAFE
ACCS03_1	9	9	36	7	10.62	70	SAFE	633.02	0	633.02	5461	SAFE
NAV_S3	27	6	108	6	0.38	25	SAFE	0.68	0	0.68	21	SAFE
NAV_S5	27	6	108	19	53.86	1683	SAFE	386.67	0	386.67	2700	SAFE
NAV_S7	81	7	432	12	3.25	136	SAFE	5.82	0	5.82	121	SAFE
NAV_S10	81	7	432	14	118.18	2669	SAFE	133.86	0	133.86	1655	SAFE
ACCS05_1	81	13	1296	4	336.36	299	SAFE	457.81	0	457.81	4369	SAFE
NAV_S12	625	5	2392	15	-	-	Timeout	158.70	0	158.70	511	SAFE
NAV_S13	625	5	2392	15	-	-	Timeout	300.88	0	300.88	671	SAFE
NAV_S14	625	5	2392	20	-	-	Timeout	135.60	0	135.60	112	SAFE

Conclusion

- We demonstrate the BMC procedure for affine hybrid systems.
- Generates concrete counterexample faster than other existing procedures.
- We have shown the efficiency of our algorithm.
- In future, we plan to guide the paths so that SAT-REACH find a CE faster.
- `https://gitlab.com/Atanukundu/SAT-Reach`

References

[1] Goran Frehse et al. "SpaceEx: Scalable Verification of Hybrid Systems". In: *Proc. 23rd International Conference on Computer Aided Verification (CAV)*. Ed. by Shaz Qadeer Ganesh Gopalakrishnan. LNCS. Springer, 2011.

[2] Aditya Zutshi et al. "A trajectory splicing approach to concretizing counterexamples for hybrid systems". In: *Proceedings of the 52nd IEEE Conference on Decision and Control, CDC 2013, December 10-13, 2013, Firenze, Italy*. 2013, pp. 3918–3925. DOI: 10.1109/CDC.2013.6760488. URL: http://dx.doi.org/10.1109/CDC.2013.6760488.

Thank you for your attention!!
Any questions?