Motion Planning for LTL Specifications: A Satisfiability Modulo Convex Optimization Approach

Yasser Shoukry
UC Berkeley, UCLA, and UPenn

Joint work with

Pierluigi Nuzzo (UC Berkeley), Indranil Saha (IIT Kanpur),
Alberto Sangiovanni-Vincentelli (UC Berkeley), Sanjit A. Seshia (UC Berkeley),
George Pappas (UPenn), and Paulo Tabuada (UCLA)

Motion Planning Problem

Given:

Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \overline{x}, \qquad \|u_t\|_\infty \le \overline{u} \quad \forall t \in \mathbb{N}$$

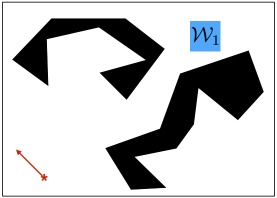
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \overline{x}, \qquad \|u_t\|_\infty \leq \overline{u} \quad \forall t \in \mathbb{N}$$

- Workspace $\mathcal{W}$:



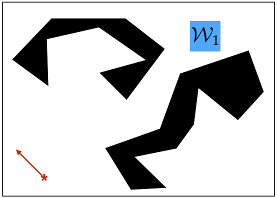
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \overline{x}, \qquad \|u_t\|_\infty \leq \overline{u} \quad \forall t \in \mathbb{N}$$

- Workspace $\mathcal{W}$: all obstacles are assumed to be unions of polyhedra.



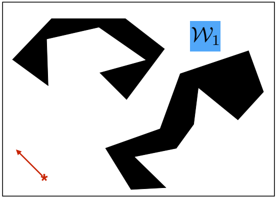
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \overline{x}, \qquad \|u_t\|_\infty \leq \overline{u} \quad \forall t \in \mathbb{N}$$

- Workspace $\mathcal{W}$: all obstacles are assumed to be unions of polyhedra.
- Atomic propositions $\Pi = \{\pi_1, \ldots, \pi_m\}$: defined over the free-workspace.



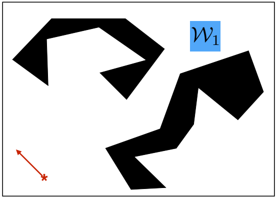
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \overline{x}, \qquad \|u_t\|_\infty \leq \overline{u} \quad \forall t \in \mathbb{N}$$

- Workspace $\mathcal{W}$: all obstacles are assumed to be unions of polyhedra.
- Atomic propositions $\Pi = \{\pi_1, \ldots, \pi_m\}$: defined over the free-workspace.
- LTL Specification Φ:



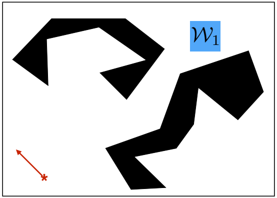
Motion Planning Problem

Given:

- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \bar{x}, \qquad \|u_t\|_\infty \leq \bar{u} \quad \forall t \in \mathbb{N}$$

- Workspace $\mathcal{W}$: all obstacles are assumed to be unions of polyhedra.
- Atomic propositions $\Pi = \{\pi_1, \ldots, \pi_m\}$: defined over the free-workspace.
- LTL Specification Φ:
  - for simplicity, I will focus on reach-avoid problems: $\Diamond \pi_1 \wedge \Box \neg \pi_0$.



Motion Planning Problem

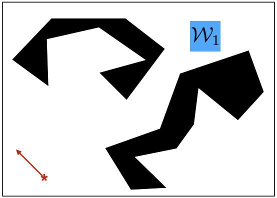
Given:

- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \overline{x}, \qquad \|u_t\|_\infty \leq \overline{u} \quad \forall t \in \mathbb{N}$$

- Workspace $\mathcal{W}$: all obstacles are assumed to be unions of polyhedra.
- Atomic propositions $\Pi = \{\pi_1, \ldots, \pi_m\}$: defined over the free-workspace.
- LTL Specification Φ:
  - for simplicity, I will focus on reach-avoid problems: $\Diamond \pi_1 \wedge \Box \neg \pi_0$.

Objective:



Motion Planning Problem

Given:

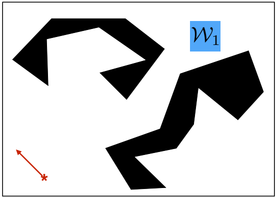
- Robot Dynamics (with input constraints):

$$x_{t+1} = Ax_t + Bu_t, \qquad x_0 = \overline{x}, \qquad \|u_t\|_\infty \leq \overline{u} \quad \forall t \in \mathbb{N}$$

- Workspace $\mathcal{W}$: all obstacles are assumed to be unions of polyhedra.
- Atomic propositions $\Pi = \{\pi_1, \ldots, \pi_m\}$: defined over the free-workspace.
- LTL Specification Φ:
  - for simplicity, I will focus on reach-avoid problems: $\Diamond \pi_1 \wedge \Box \neg \pi_0$.

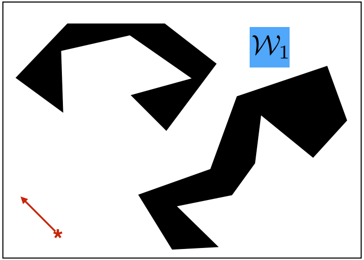
Objective:

- Generate the input sequence $u_0, u_1, \ldots, u_L$ such that the trajectory of the robot satisfies Φ.

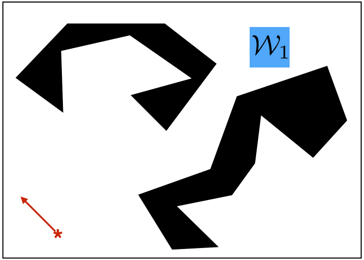


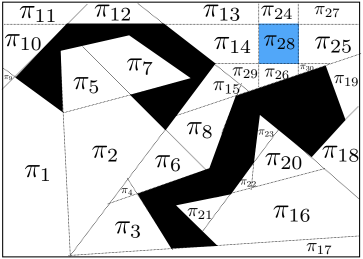
Solver Architecture [0/3]

- **Step 0:** Compute coarse, obstacle-based, discretization of the free space.

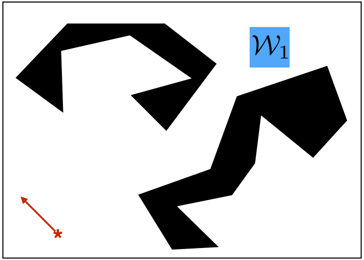


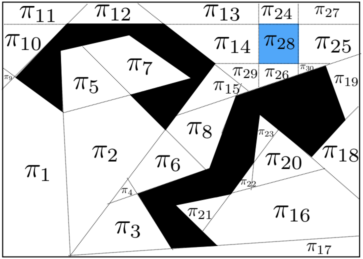
- **Step 0:** Compute coarse, obstacle-based, discretization of the free space.





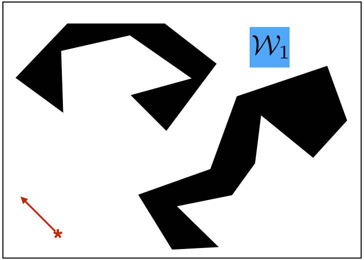
- **Step 0:** Compute coarse, obstacle-based, discretization of the free space.
  - Avoid discrete state-space explosion (no abstraction of continuous dynamics, no grid-based abstractions).

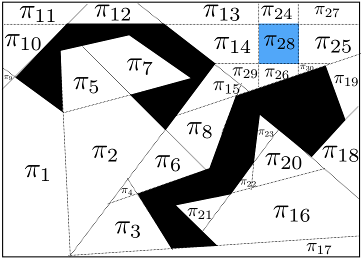


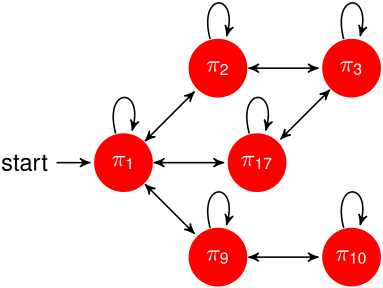


Solver Architecture [0/3]

- **Step 0:** Compute coarse, obstacle-based, discretization of the free space.
  - Avoid discrete state-space explosion (no abstraction of continuous dynamics, no grid-based abstractions).

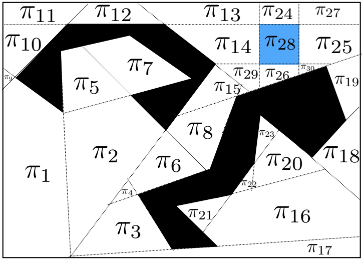


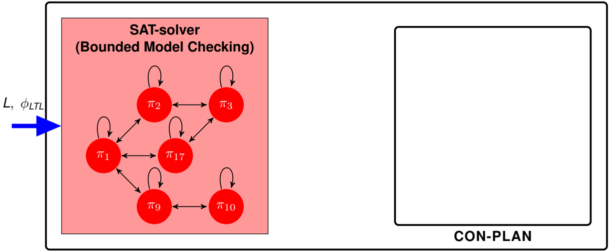




Solver Architecture [1/3]

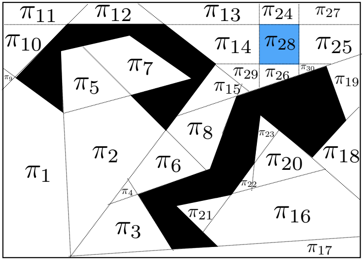
- **Step 1:** For a given horizon L, use a SAT solver to find a candidate high-level path (Bounded Model Checking) that satisfies the LTL specification.

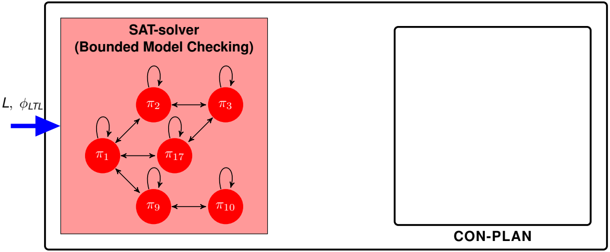




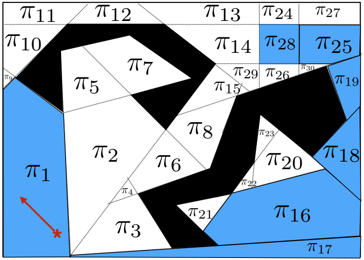
Solver Architecture [1/3]

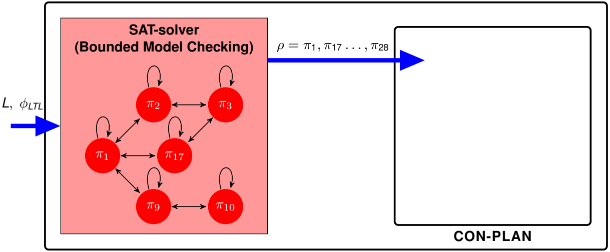
- **Step 1:** For a given horizon *L*, use a SAT solver to find a candidate high-level path (Bounded Model Checking) that satisfies the LTL specification.
  - Increase the horizon *L* until such high-level path is found.





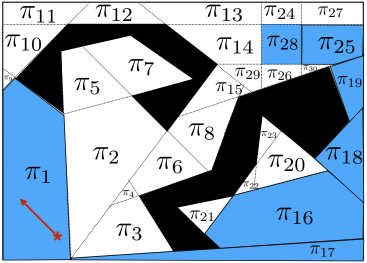
- **Step 1:** For a given horizon *L*, use a SAT solver to find a candidate high-level path (Bounded Model Checking) that satisfies the LTL specification.
  - Increase the horizon *L* until such high-level path is found.

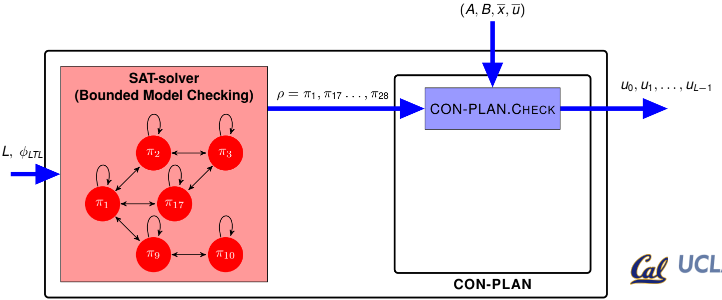




Solver Architecture [2/3]

- **Step 2:** Check whether the high-level trajectory ρ is feasible (satisfies the robot initial state, dynamics, and input constraints).
    -

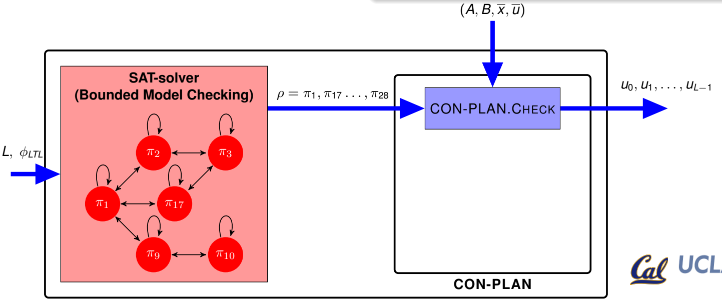




- **Step 2:** Check whether the high-level trajectory ρ is feasible (satisfies the robot initial state, dynamics, and input constraints).
  - Can be casted as an optimization problem.

Problem (CON-PLAN.CHECK)

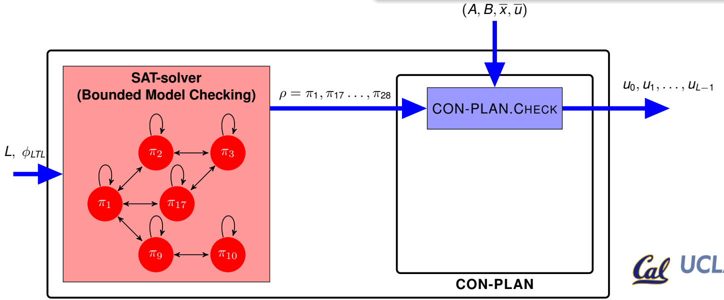
min	1	$i = 0, \ldots, L$
(initial condition)		$x_0 = \overline{x}$,
(dynamics constraints)		$x_{i+1} = Ax_i + Bu_i$
(input constraints)		$\|u_i\| \leq \overline{u}$,
(plan constraints)		$x_i \in \rho_i$



- **Step 2:** Check whether the high-level trajectory ρ is feasible (satisfies the robot initial state, dynamics, and input constraints).
  - Can be casted as a **convex** optimization problem (**Linear Program**).

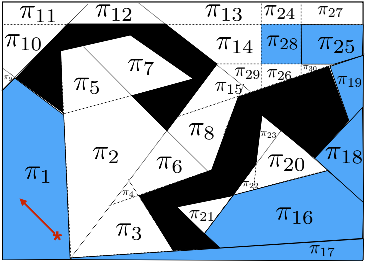
Problem (CON-PLAN.CHECK)

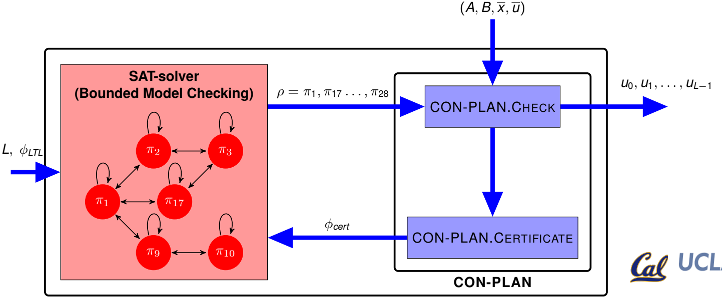
min	1	$i = 0, \dots, L$
(initial condition)		$x_0 = \overline{x},$
(dynamics constraints)		$x_{i+1} = Ax_i + Bu_i$
(input constraints)		$\lVert u_i \rVert \leq \overline{u},$
(plan constraints)		$x_i \in \rho_i$



- **Step 3:** Generate counter example.

$$\phi_{\text{triv-ce}} := \bigvee_{i=0}^{L} \neg \rho_i,$$

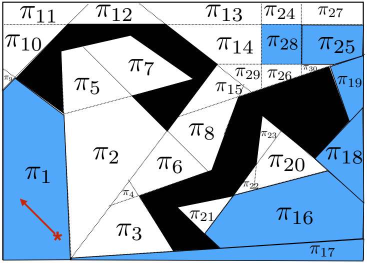


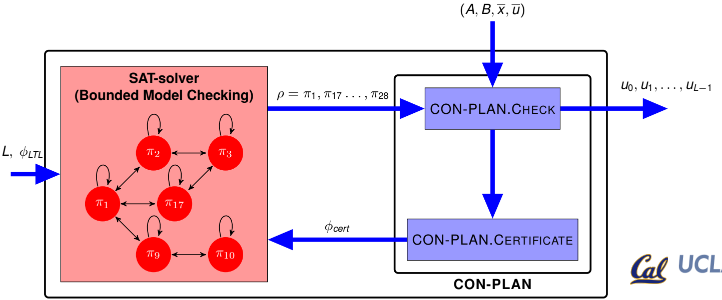


- **Step 3:** Generate counter example.

$$\phi_{\text{triv-ce}} := \bigvee_{i=0}^{L} \neg \rho_i,$$

- Repeat.

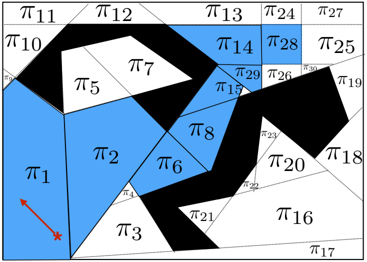


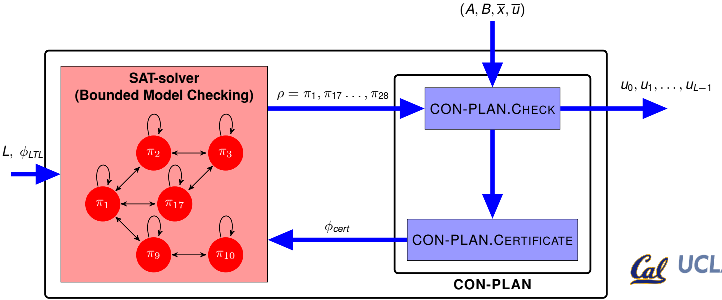


- **Step 3:** Generate counter example.

$$\phi_{\text{triv-ce}} := \bigvee_{i=0}^{L} \neg\rho_i,$$

- Repeat.



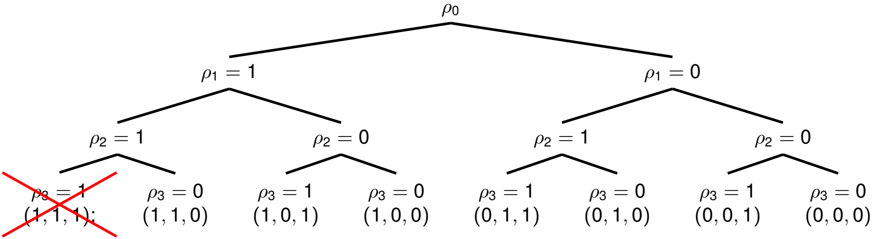


Succinct Counter-example Generation

- Detecting the earliest possible occurrence of an infeasible transition between two regions rules out a broader class of assignments to Boolean variables.

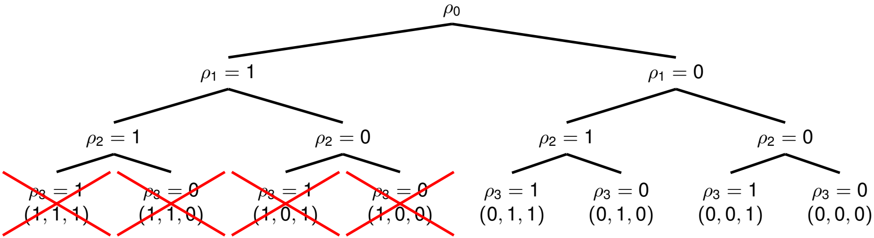
Succinct Counter-example Generation

- Detecting the earliest possible occurrence of an infeasible transition between two regions rules out a broader class of assignments to Boolean variables.
- Example 1: $\phi_{\text{counter-example}} := \neg\rho_0 \vee \neg\rho_1 \vee \neg\rho_2 \vee \neg\rho_3$



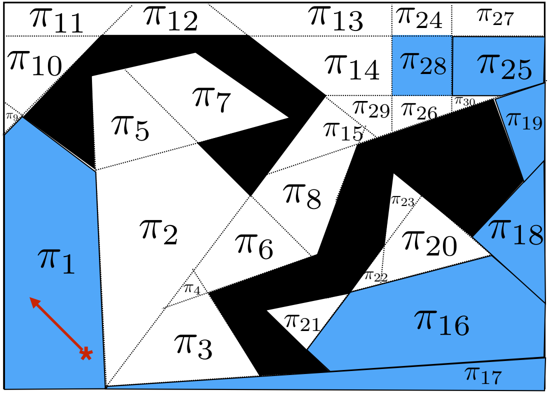
Succinct Counter-example Generation

- Detecting the earliest possible occurrence of an infeasible transition between two regions rules out a broader class of assignments to Boolean variables.
- Example 1: $\phi_{\text{counter-example}} := \neg\rho_0 \vee \neg\rho_1 \vee \neg\rho_2 \vee \neg\rho_3$
- Example 2: $\phi_{\text{counter-example}} := \neg\rho_0 \vee \neg\rho_1$



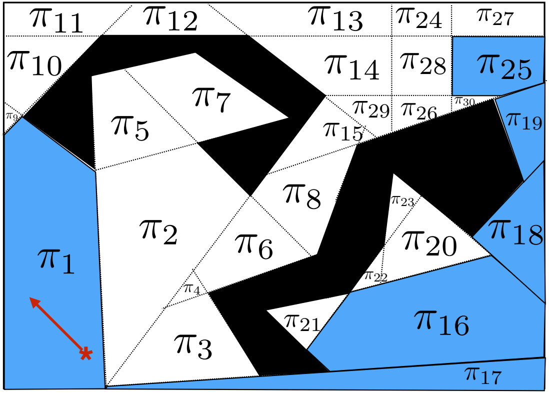
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.

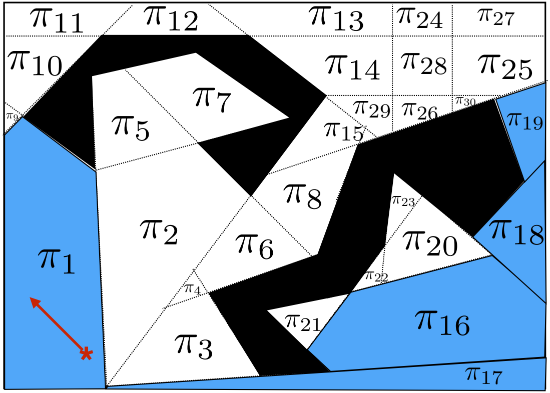


Succinct Counterexample Generation

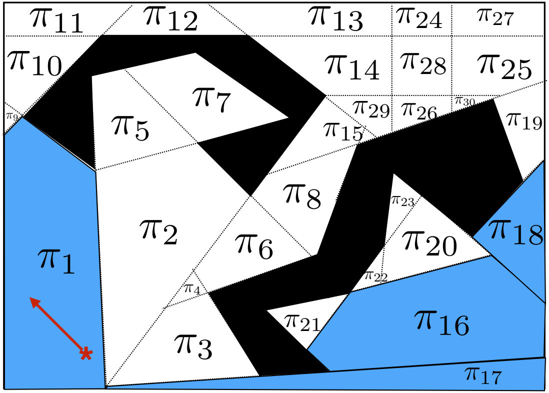
- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.



- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.

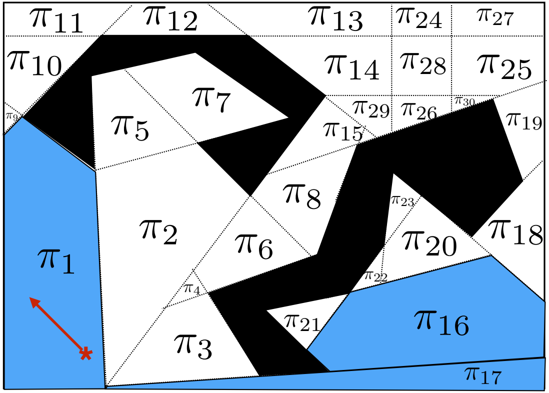


- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.



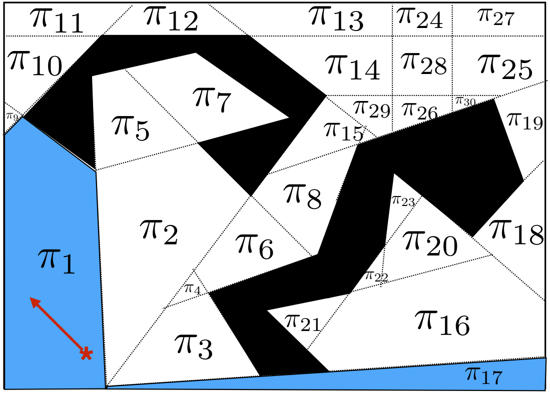
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.



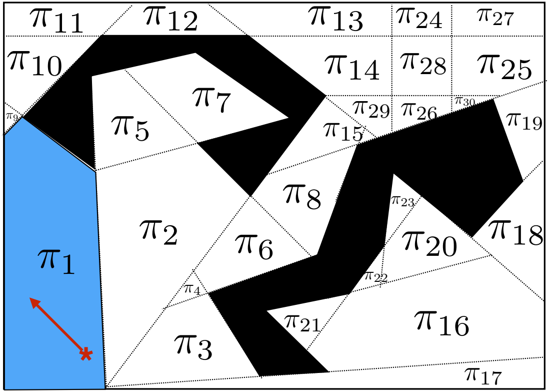
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.



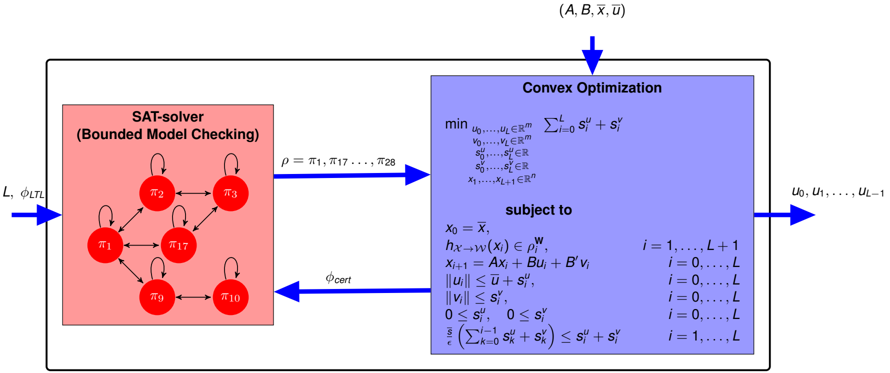
Succinct Counterexample Generation

- Searching for succinct counterexample can be performed by checking the feasibility of prefixes of ρ.
- Leads to solving multiple optimization problems.



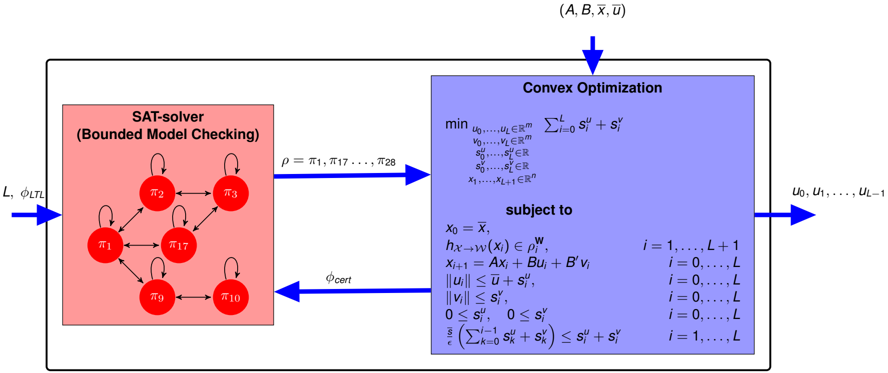
Succinct Counterexample Generation

- **Key insight:** we can check feasibility of high-level plans and generate the shortest counter example by solving a single linear program.



Succinct Counterexample Generation

- **Key insight:** we can check feasibility of high-level plans and generate the shortest counter example by solving a single linear program.
- Satisfiability modulo convex-optimization approach.



Case Study 1: Dubin's Vehicle

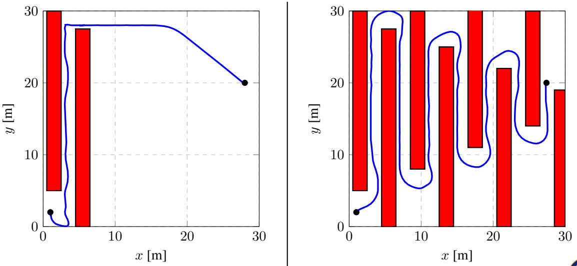
- **Robot dynamics** (Dubin car): $\dot{x} = v\cos\theta \qquad \dot{y} = v\sin\theta \qquad \dot{\theta} = \omega$

Case Study 1: Dubin's Vehicle

- **Robot dynamics** (Dubin car): $\dot{x} = v\cos\theta \qquad \dot{y} = v\sin\theta \qquad \dot{\theta} = \omega$
  - Dynamics can be transformed into a linear chain of integrators using dynamic feedback linearization.

Case Study 1: Dubin's Vehicle

- **Robot dynamics** (Dubin car): $\dot{x} = v\cos\theta \qquad \dot{y} = v\sin\theta \qquad \dot{\theta} = \omega$
  - Dynamics can be transformed into a linear chain of integrators using dynamic feedback linearization.
- **Workspace:** 30m $\times$ 30m maze-like workspace.
  - We increase number of passages from 1 to 4.



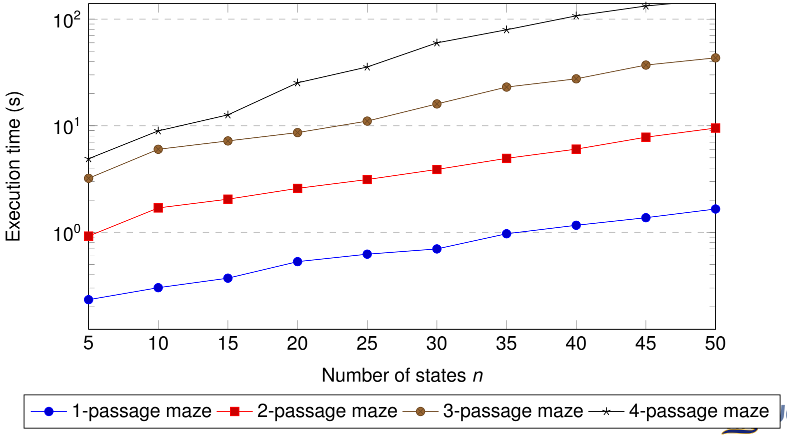
Case Study 1: Dubin's Vehicle

- **Robot dynamics** (Dubin car): $\dot{x} = v\cos\theta \qquad \dot{y} = v\sin\theta \qquad \dot{\theta} = \omega$
  - Dynamics can be transformed into a linear chain of integrators using dynamic feedback linearization.
- **Workspace:** 30m $\times$ 30m maze-like workspace.
  - We increase number of passages from 1 to 4.
- We compare execution-time against (1) standard RRT and (2) LTL OPT tool (mixed-integer linear program).

Number of passages	**SMT-Based Motion Planner [s]**			**RRT [s]**	**LTL OPT [s]**
	Discrete abstraction	DIS-PLAN	CON-PLAN		
1	1.9975	0.1360	0.2542	10.4381	>7200
2	7.1461	1.1290	0.9294	122.3017	time out
3	19.3267	3.6495	1.0053	423.6957	time out
4	43.0985	4.0913	1.9204	1002.4193	time out

Case Study 2: Scalability Results

- Maze-like workspace with increasing number of passages.
- We increase the number of states n (randomly generate the matrices A and B).
- Take average across 10 runs.



Final Remarks

- **What is still missing ?**

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT* and PRM.

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT^* and PRM.
  - We need to compare against existing benchmarks.

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.
- **SAT + Convex optimization** $\stackrel{?}{=}$ **tools**:

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.
- **SAT + Convex optimization** $\stackrel{?}{=}$ **tools**:
  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.
- **SAT + Convex optimization $\stackrel{?}{=}$ tools**:
  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.
  - Security: secure state estimation.

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT^* and PRM.
  - We need to compare against existing benchmarks.
- **SAT + Convex optimization $\stackrel{?}{=}$ tools**:
  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.
  - Security: secure state estimation.
  - Sensor Networks: localization.

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.
- **SAT + Convex optimization $\stackrel{?}{=}$ tools**:
  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.
  - Security: secure state estimation.
  - Sensor Networks: localization.
  - Motion planning (this work).

Final Remarks

- **What is still missing ?**
  - We need to compare with other methods like RRT* and PRM.
  - We need to compare against existing benchmarks.
- **SAT + Convex optimization $\stackrel{?}{=}$ tools**:
  - Probabilistic CTL (PCTL) verification of Markov Decision Processes.
  - Security: secure state estimation.
  - Sensor Networks: localization.
  - Motion planning (this work).
  - We are currently developing a comprehensive theory of Satisfiability Modulo Convex Optimization.