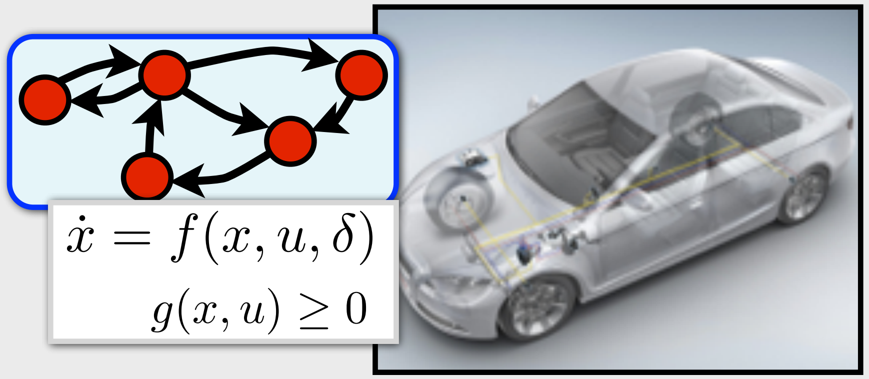
Automata Theory and Dynamic Programming

Ivan Papusha
Postdoctoral Fellow
Institute for Computational Engineering and Sciences
University of Texas at Austin

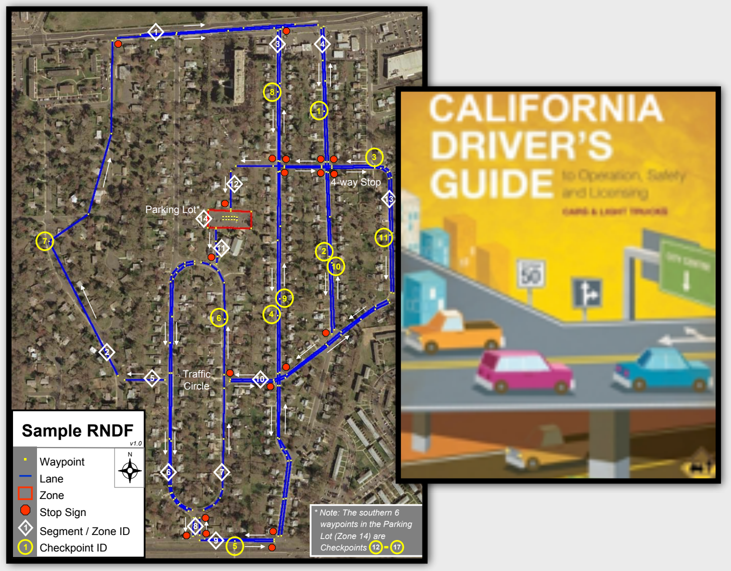
A Synthesis Problem

Given:

- ***System model***
  - both continuous & discrete evolution
  - actuation limitations
  - modeling uncertainties & disturbances



- ***Specifications***
  - high-level requirements
  - optimality criteria



Automatically synthesize a control protocol that

- manages the system behavior and
- is **provably correct** with respect to the specifications and **optimal**.

Outline

1. Abstraction-based synthesis

2. Approximate Dynamic Programming

3. Learning from expert demonstrations

Detour: Specifying Behavior with Temporal Logic

(only a dialect in a large family of languages)

Propositional Logic	$\wedge$ (and) $\vee$ (or) $\rightarrow$ (implies) $\neg$ (not)
+	
Temporal Operators	$\Diamond$ (eventually) $\Box$ (always) $\mathcal{U}$ (until)

Detour: Specifying Behavior with Temporal Logic

(only a dialect in a large family of languages)

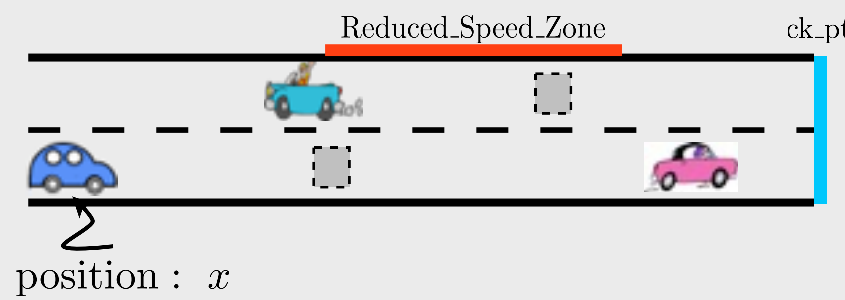
Propositional Logic

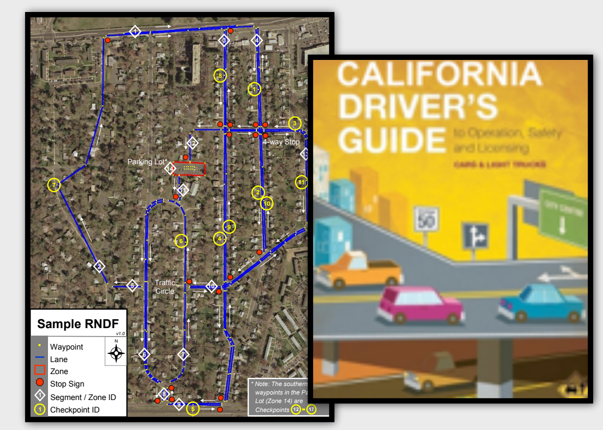
$\wedge$ (and)
$\vee$ (or)
$\rightarrow$ (implies)
$\neg$ (not)

+

Temporal Operators

$\Diamond$ (eventually)
$\Box$ (always)
$\mathcal{U}$ (until)





Traffic rules:

- No collision $\Box\,(\mathrm{dist}(x, \mathrm{Obs}) \geq X_{\mathrm{safe}} \wedge \mathrm{dist}(x, \mathrm{Loc}(\mathrm{Veh})) \geq X_{\mathrm{safe}})$
- Obey speed limits $\Box\,((x \in \mathrm{Reduced\_Speed\_Zone}) \rightarrow (v \leq v_{\mathrm{reduced}}))$
- Stay in travel lane unless blocked
- Intersection precedence & merging, stop line, passing,...

Goals:

- Eventually visit the check point $\Diamond(x = \mathrm{ck\_pt})$
- Every time check point is reached, eventually come to start $\Box((x = \mathrm{ck\_pt}) \rightarrow \Diamond(x = \mathrm{start}))$

Detour: Specifying Behavior with Temporal Logic

(only a dialect in a large family of languages)

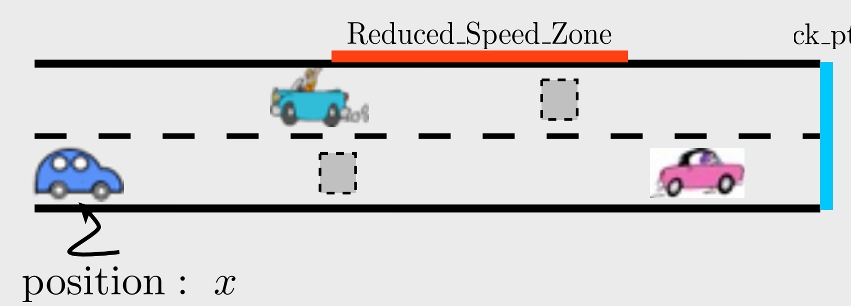
Propositional Logic

+

Temporal

$\wedge$ (and)
$\vee$ (or)
$\rightarrow$ (implies)
$\neg$ (not)

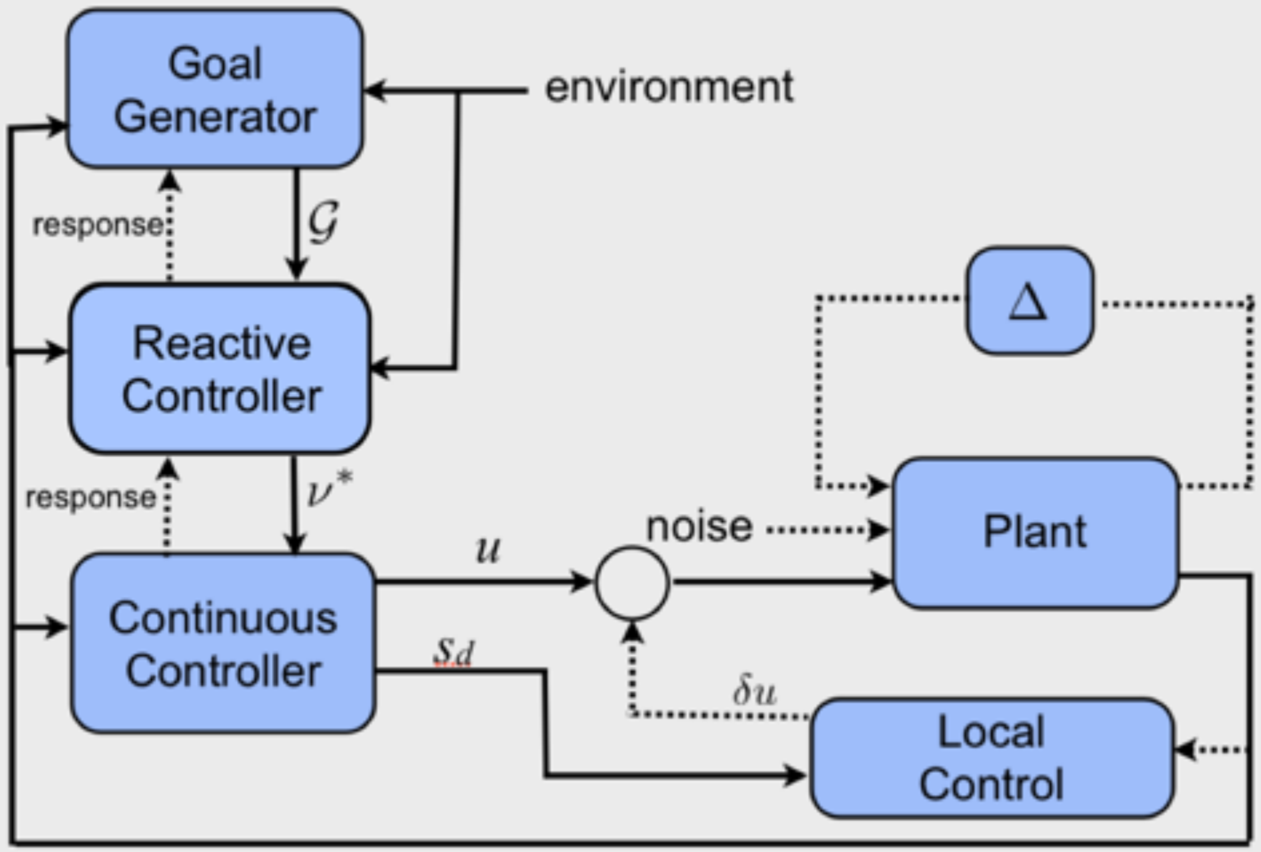
$\Diamond$ (eventually)



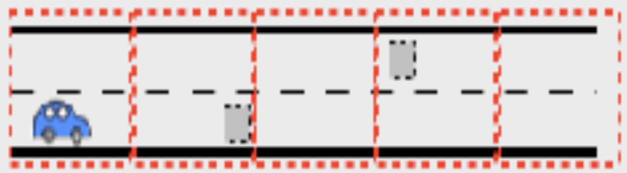
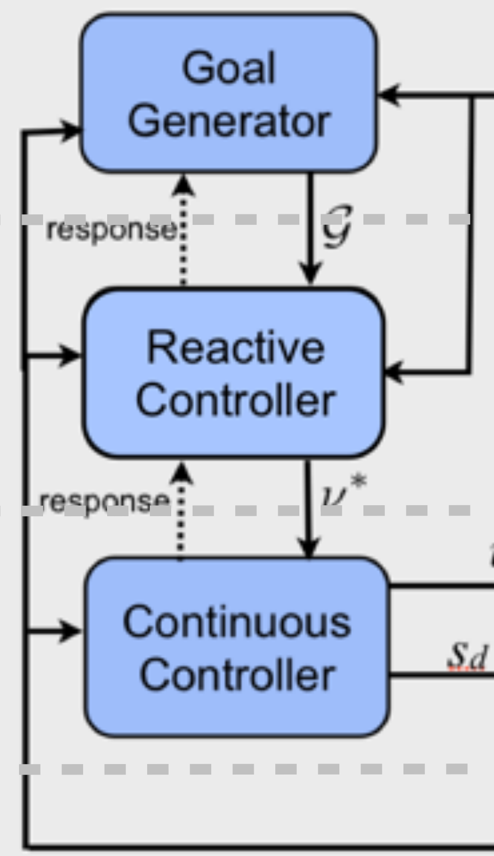
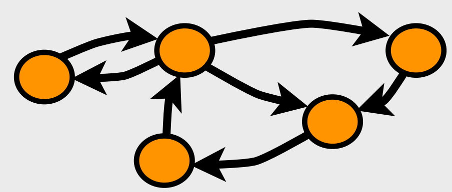
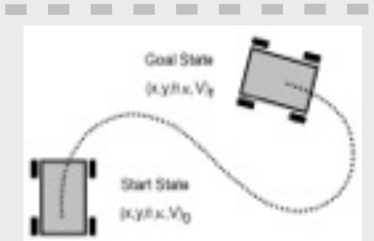
VMS_global.spc — Edited

```
        [](((s.a0=1 & s.a1=1) & (s.h0=1 & s.h1=0 & s.h2=0) & (e.w0=0 & e.w1=0)) -> ((s.pf0=1 & s.pf1=0))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=1 & s.h1=0 & s.h2=0) & (e.w0=1 & e.w1=0)) -> ((s.pf0=0 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=1 & s.h1=0 & s.h2=0) & (e.w0=0 & e.w1=1)) -> ((s.pf0=0 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=0 & s.h1=1 & s.h2=0) & (e.w0=0 & e.w1=0)) -> ((s.pf0=1 & s.pf1=0))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=0 & s.h1=1 & s.h2=0) & (e.w0=1 & e.w1=0)) -> ((s.pf0=0 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=0 & s.h1=1 & s.h2=0) & (e.w0=0 & e.w1=1)) -> ((s.pf0=1 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=1 & s.h1=1 & s.h2=0) & (e.w0=0 & e.w1=0)) -> ((s.pf0=0 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=1 & s.h1=1 & s.h2=0) & (e.w0=1 & e.w1=0)) -> ((s.pf0=0 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=1 & s.h1=1 & s.h2=0) & (e.w0=0 & e.w1=1)) -> ((s.pf0=1 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=0 & s.h1=0 & s.h2=1) & (e.w0=0 & e.w1=0)) -> ((s.pf0=0 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=0 & s.h1=0 & s.h2=1) & (e.w0=1 & e.w1=0)) -> ((s.pf0=1 & s.pf1=1))) &
        [](((s.a0=1 & s.a1=1) & (s.h0=0 & s.h1=0 & s.h2=1) & (e.w0=0 & e.w1=1)) -> ((s.pf0=1 & s.pf1=1))) &
        [](((s.h0=0 & s.h1=0 & s.h2=0) & (s.a0=0 & s.a1=0)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0))) &
        [](((s.h0=0 & s.h1=0 & s.h2=0) & (s.a0=1 & s.a1=0)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0))) &
        [](((s.h0=0 & s.h1=0 & s.h2=0) & (s.a0=0 & s.a1=1)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0))) &
        [](((s.h0=0 & s.h1=0 & s.h2=0) & (s.a0=1 & s.a1=1)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0))) &
        [](((s.h0=1 & s.h1=0 & s.h2=0) & (s.a0=0 & s.a1=0)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0) | (s.h0=1 & s.h1=1 & s.h2=0))) &
        [](((s.h0=1 & s.h1=0 & s.h2=0) & (s.a0=1 & s.a1=0)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0) | (s.h0=1 & s.h1=1 & s.h2=0))) &
        [](((s.h0=1 & s.h1=0 & s.h2=0) & (s.a0=0 & s.a1=1)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0) | (s.h0=1 & s.h1=1 & s.h2=0))) &
        [](((s.h0=1 & s.h1=0 & s.h2=0) & (s.a0=1 & s.a1=1)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0))) &
        [](((s.h0=0 & s.h1=1 & s.h2=0) & (s.a0=0 & s.a1=0)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0) | (s.h0=1 & s.h1=1 & s.h2=0) | (s.h0=0 & s.h1=0 & s.h2=1))) &
        [](((s.h0=0 & s.h1=1 & s.h2=0) & (s.a0=1 & s.a1=0)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0) | (s.h0=1 & s.h1=1 & s.h2=0) | (s.h0=0 & s.h1=0 & s.h2=1))) &
        [](((s.h0=0 & s.h1=1 & s.h2=0) & (s.a0=0 & s.a1=1)) -> next ((s.h0=0 & s.h1=0 & s.h2=0) | (s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 &
s.h1=1 & s.h2=0) | (s.h0=1 & s.h1=1 & s.h2=0) | (s.h0=0 & s.h1=0 & s.h2=1))) &
        [](((s.h0=0 & s.h1=1 & s.h2=0) & (s.a0=1 & s.a1=1)) -> next ((s.h0=1 & s.h1=0 & s.h2=0) | (s.h0=0 & s.h1=1 & s.h2=0) | (s.h0=1 &
```

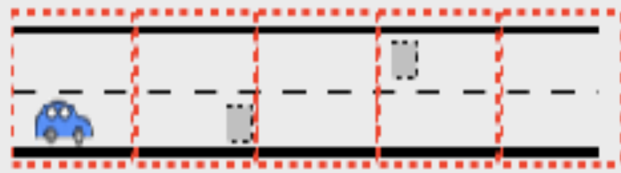
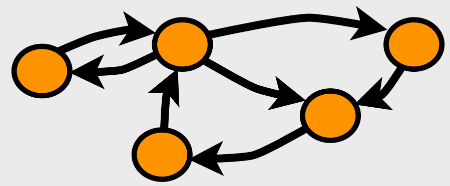
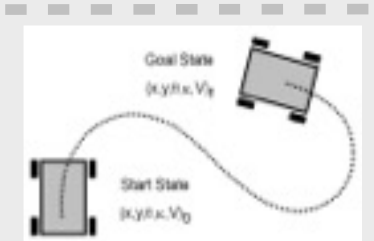
A widely explored approach



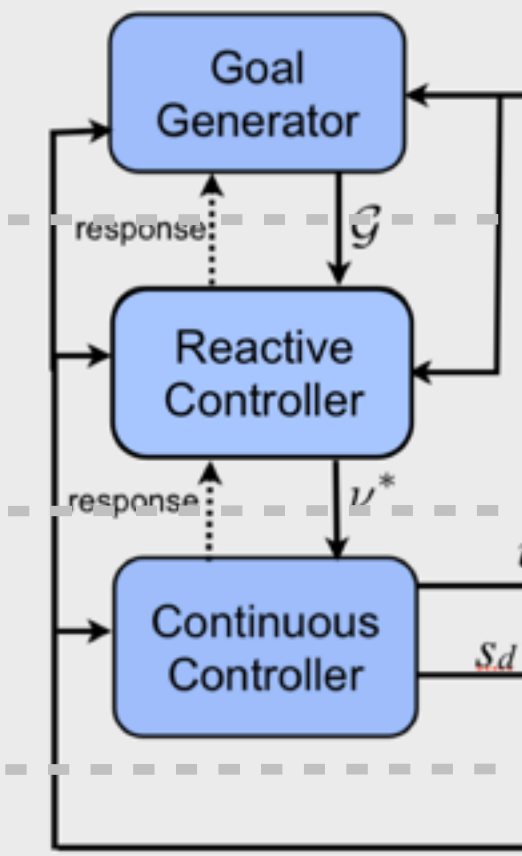
A widely explored approach

Different views		Multi-scale models	Synthesis method	Control protocol
	long-horizon specifications			
	short-horizon specifications			
	constraints on continuous state + input	$x_{t+1} = f(x_t, w_t, u_t)$ $x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$		

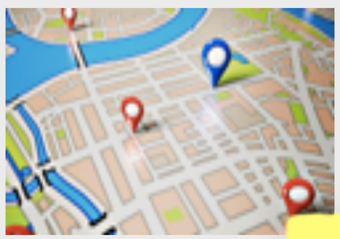
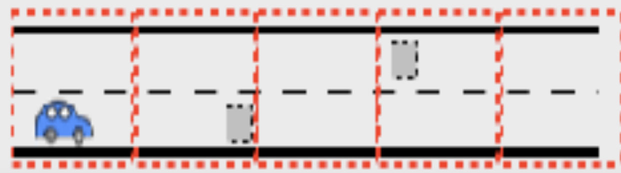
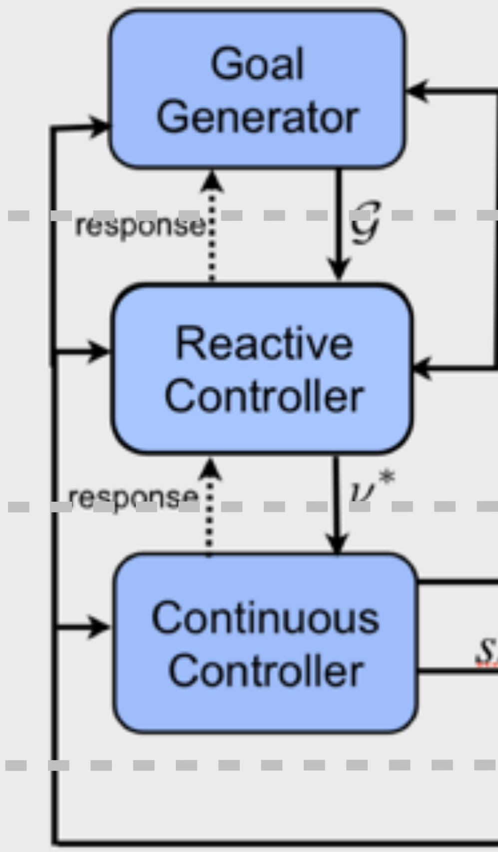
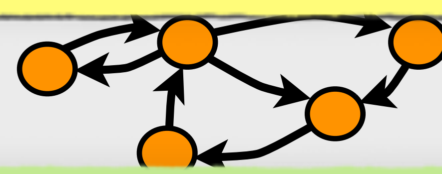
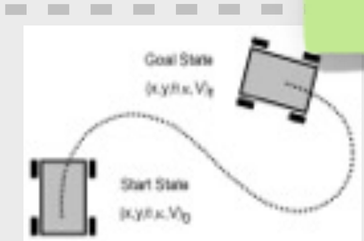
A widely explored approach

Different views		Multi-scale models	Synthesis method
	long-horizon specifications		Iterative graph search
	short-horizon specifications		Two-player, turn-based graph game
	constraints on continuous state + input	$x_{t+1} = f(x_t, w_t, u_t)$ $x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$	Constrained, finite-horizon optimal control

Control protocol



A widely explored approach

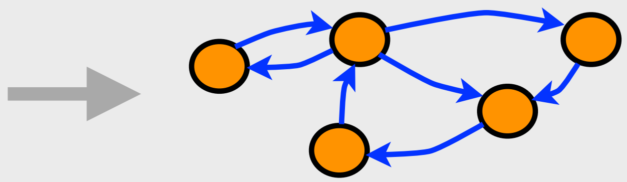
Different views		Multi-scale models	Synthesis method	Control protocol
	long-horizon specifications		Iterative graph search	
	short-horizon specifications		Two-player, turn-based graph game	
	constraints on continuous state + input	$x_{t+1} = f(x_t, w_t, u_t)$ $x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$	Constrained, finite-horizon optimal control	

Abstraction with "simulation" relation

(Finite-state) abstraction with "simulation" relation

Finite-state abstraction with "simulation" relations

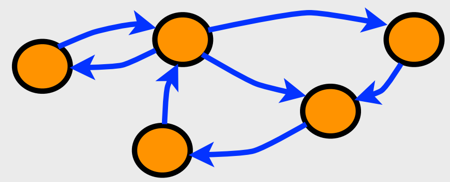
$$x_{t+1} = f(x_t, w_t, u_t)$$
$$x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$$

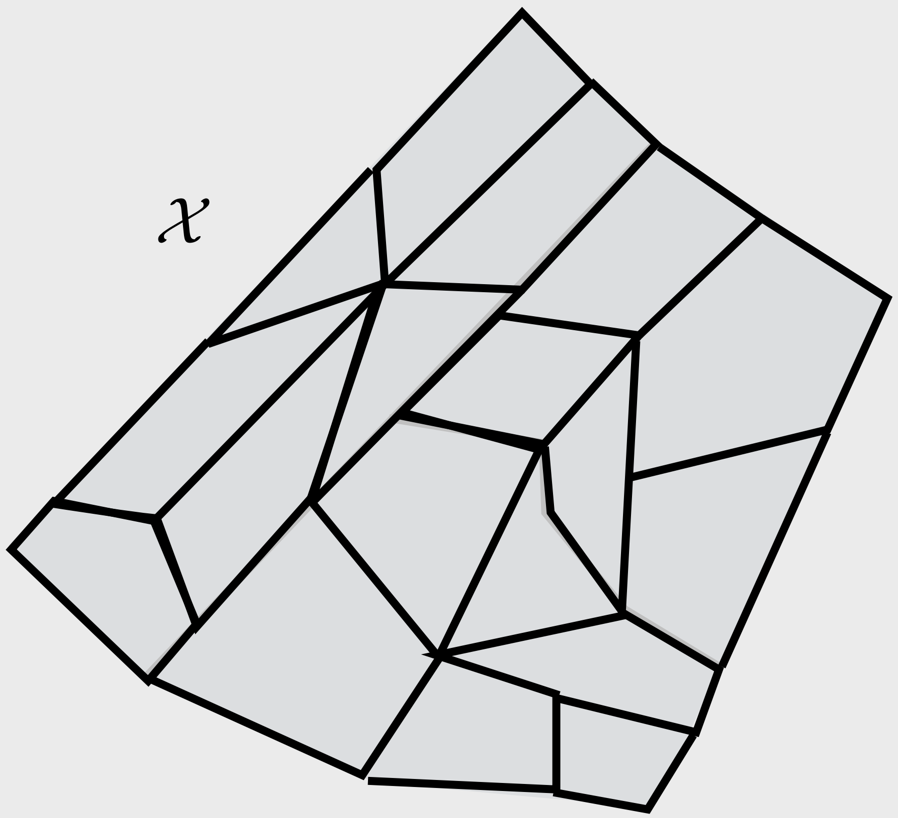


Every discrete transition can be "executed" under the continuous dynamics

Finite-state abstraction with "simulation" relations

$$x_{t+1} = f(x_t, w_t, u_t)$$
$$x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$$

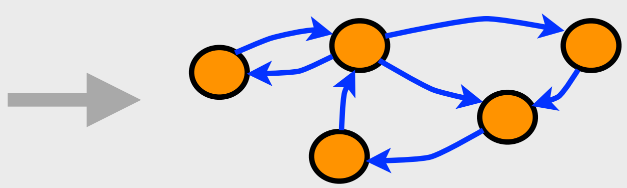


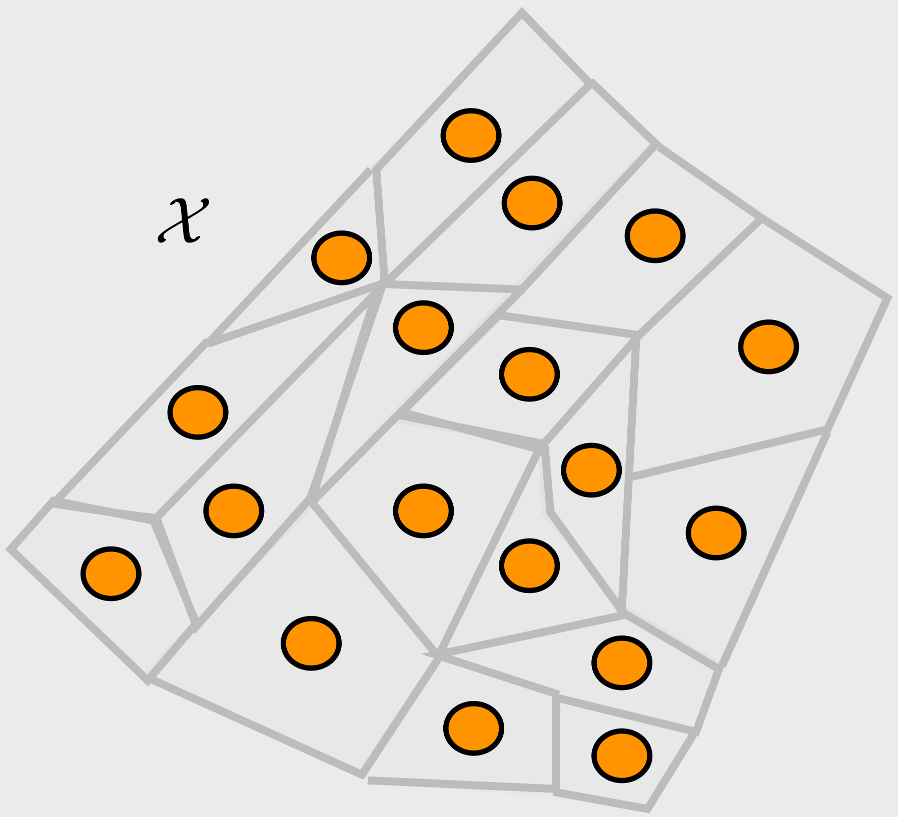


Every discrete transition can be "executed" under the continuous dynamics

Finite-state abstraction with "simulation" relations

$$x_{t+1} = f(x_t, w_t, u_t)$$
$$x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$$

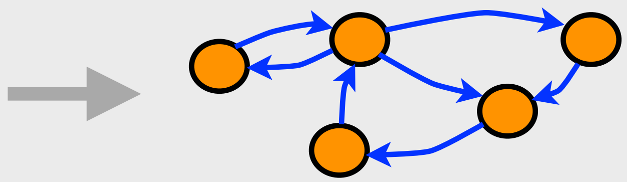


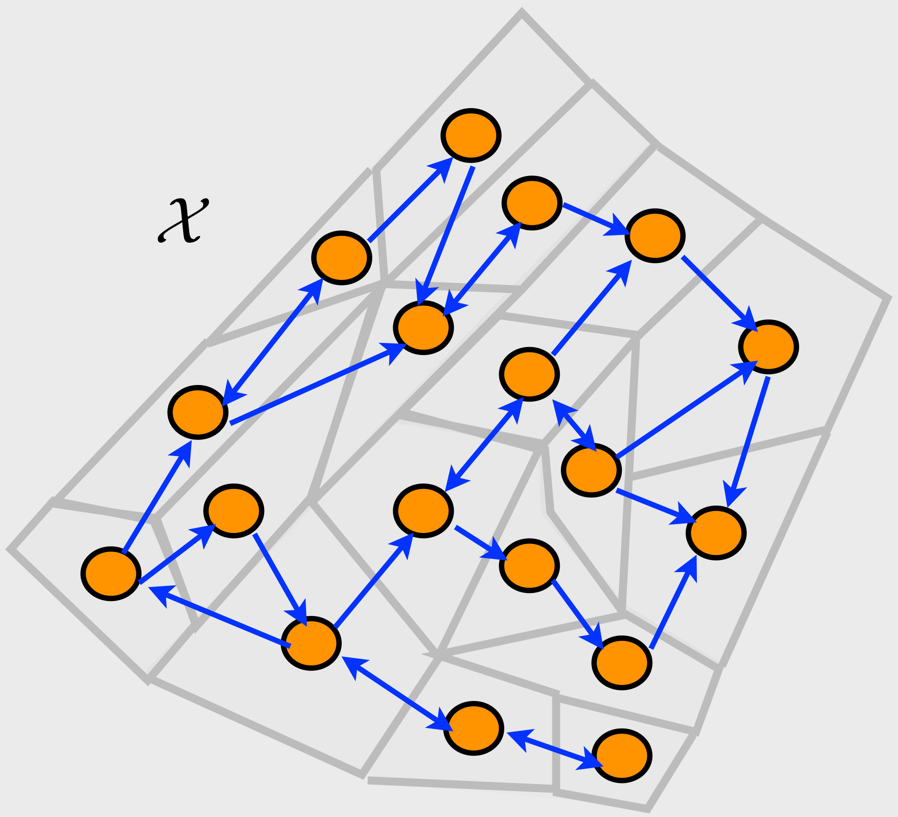


Every discrete transition can be "executed" under the continuous dynamics

Finite-state abstraction with "simulation" relations

$$x_{t+1} = f(x_t, w_t, u_t)$$
$$x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$$

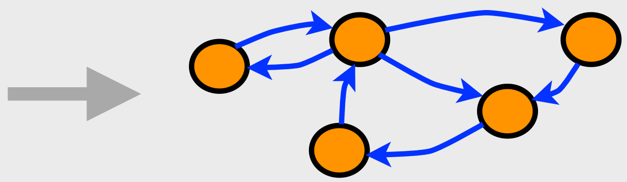


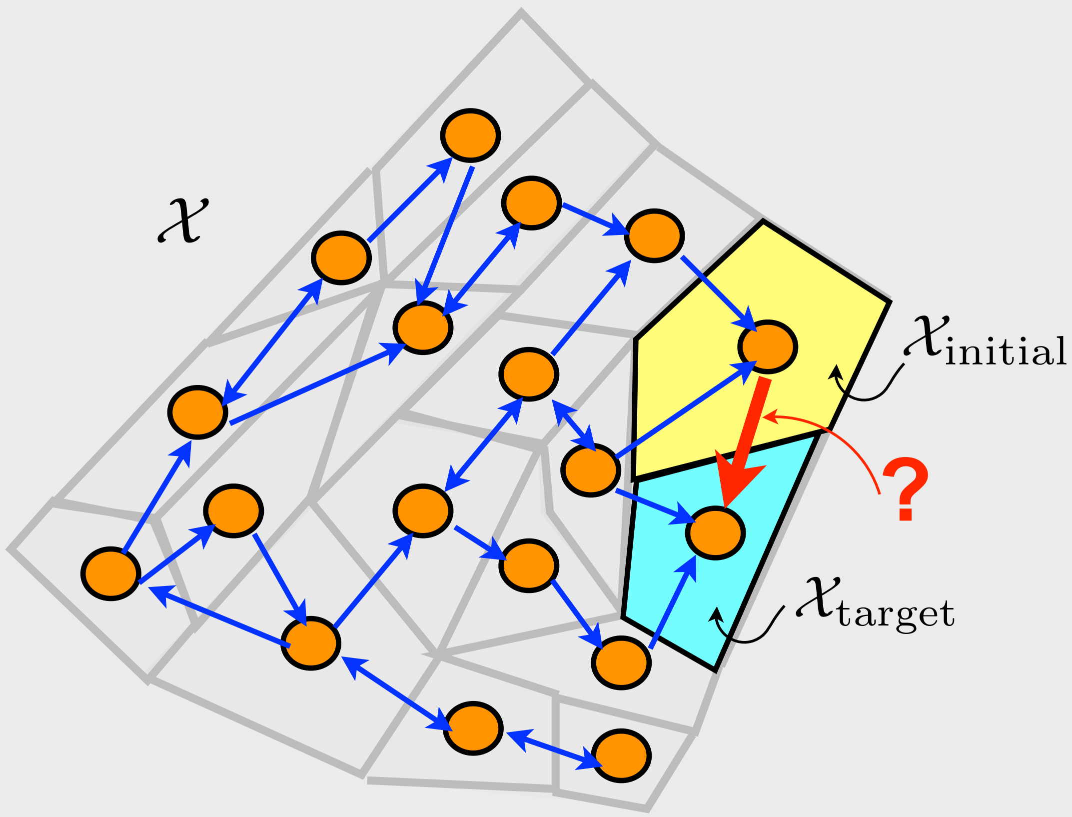


Every discrete transition can be "executed" under the continuous dynamics

Finite-state abstraction with "simulation" relations

$$x_{t+1} = f(x_t, w_t, u_t)$$
$$x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$$

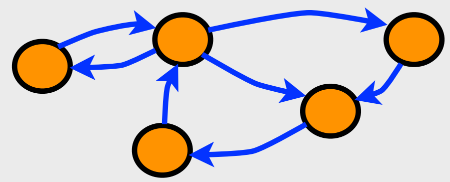


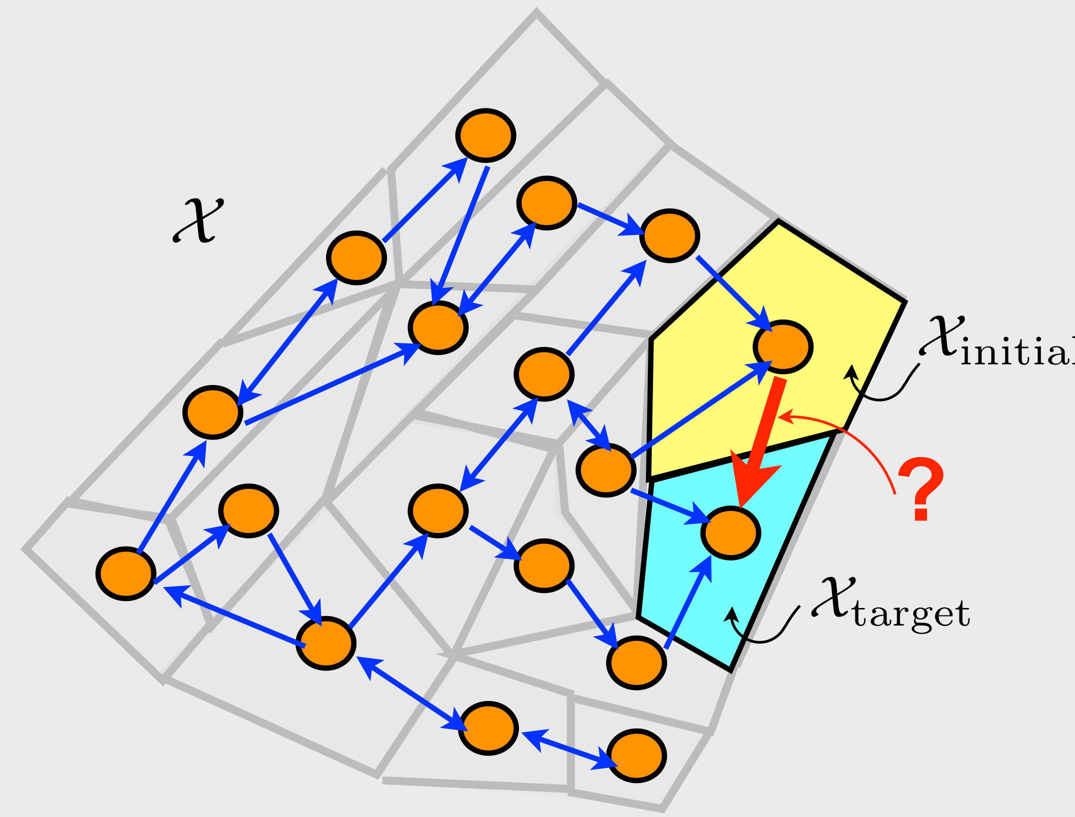


Every discrete transition can be "executed" under the continuous dynamics

Finite-state abstraction with “simulation” relations

$$x_{t+1} = f(x_t, w_t, u_t)$$
$$x \in \mathcal{X}, u \in \mathcal{U}, w \in \mathcal{W}$$

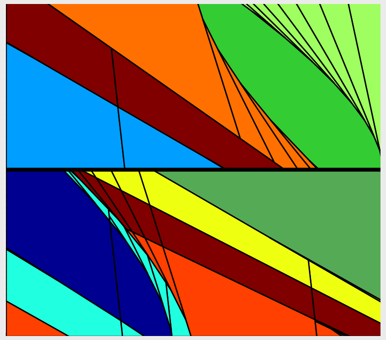




Every discrete transition can be “executed” under the continuous dynamics

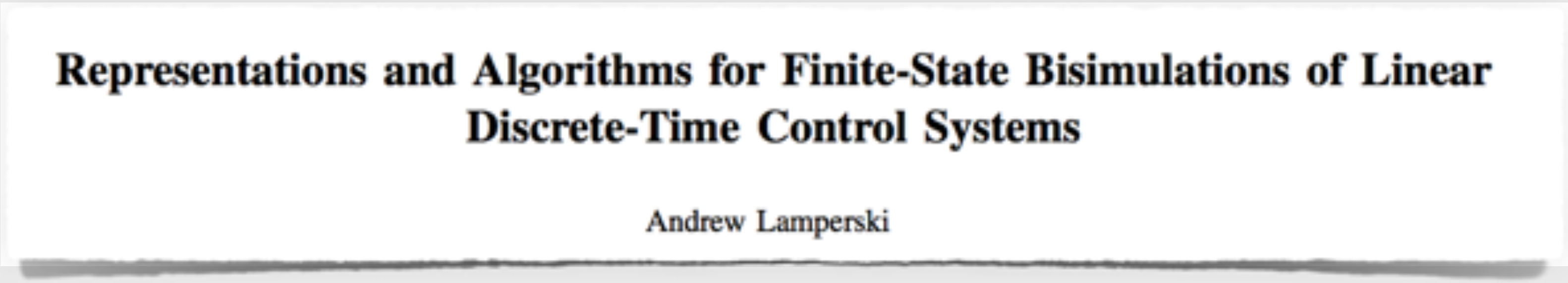
Why is discretization not necessarily a good idea?

Practically:
Complex partitions are needed.



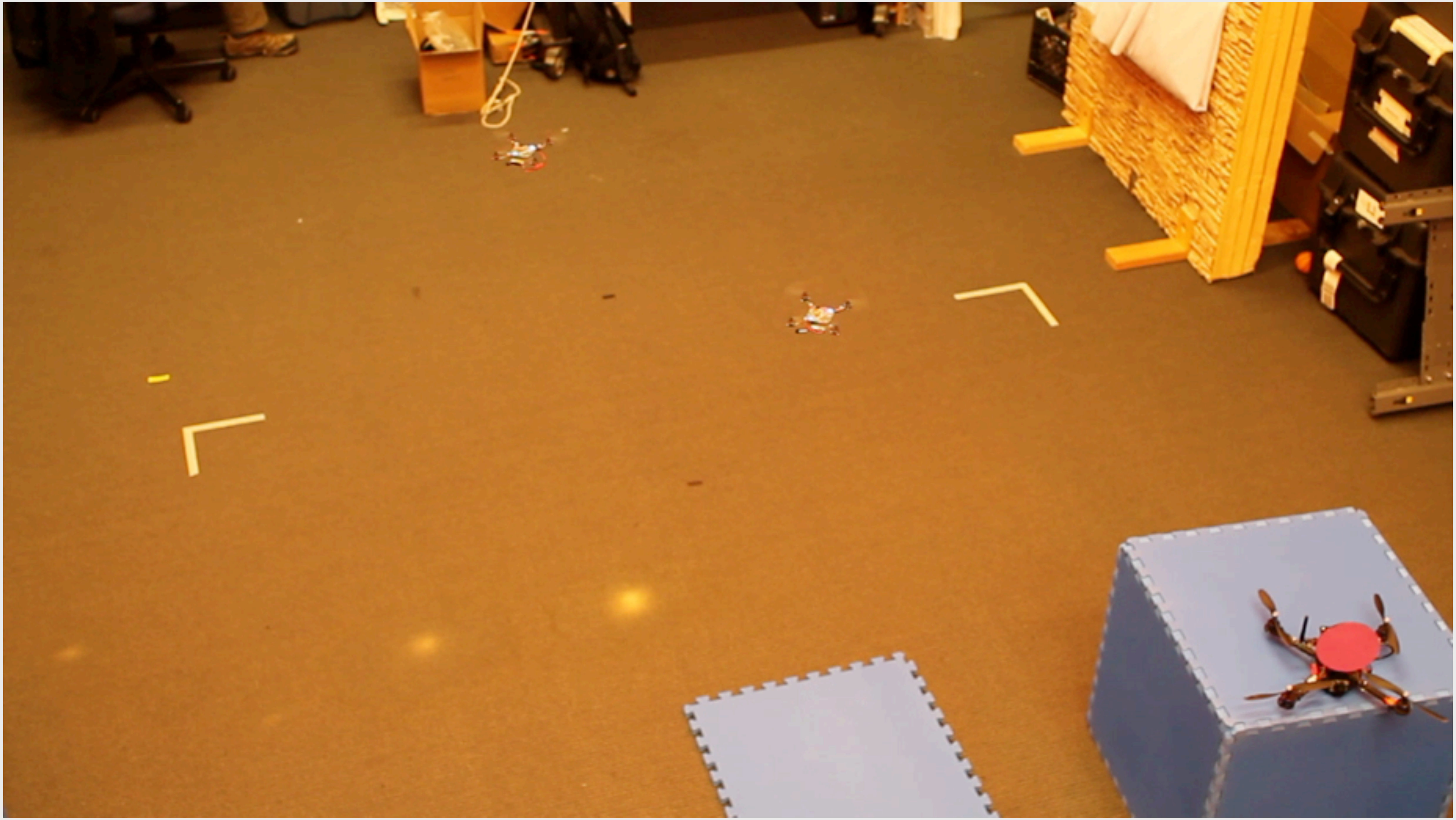
Theoretically:
Finite yet humongous discrete state spaces may be needed.

$$2^{2^{\cdot^{\cdot^{\cdot^{2^p}}}}}$$

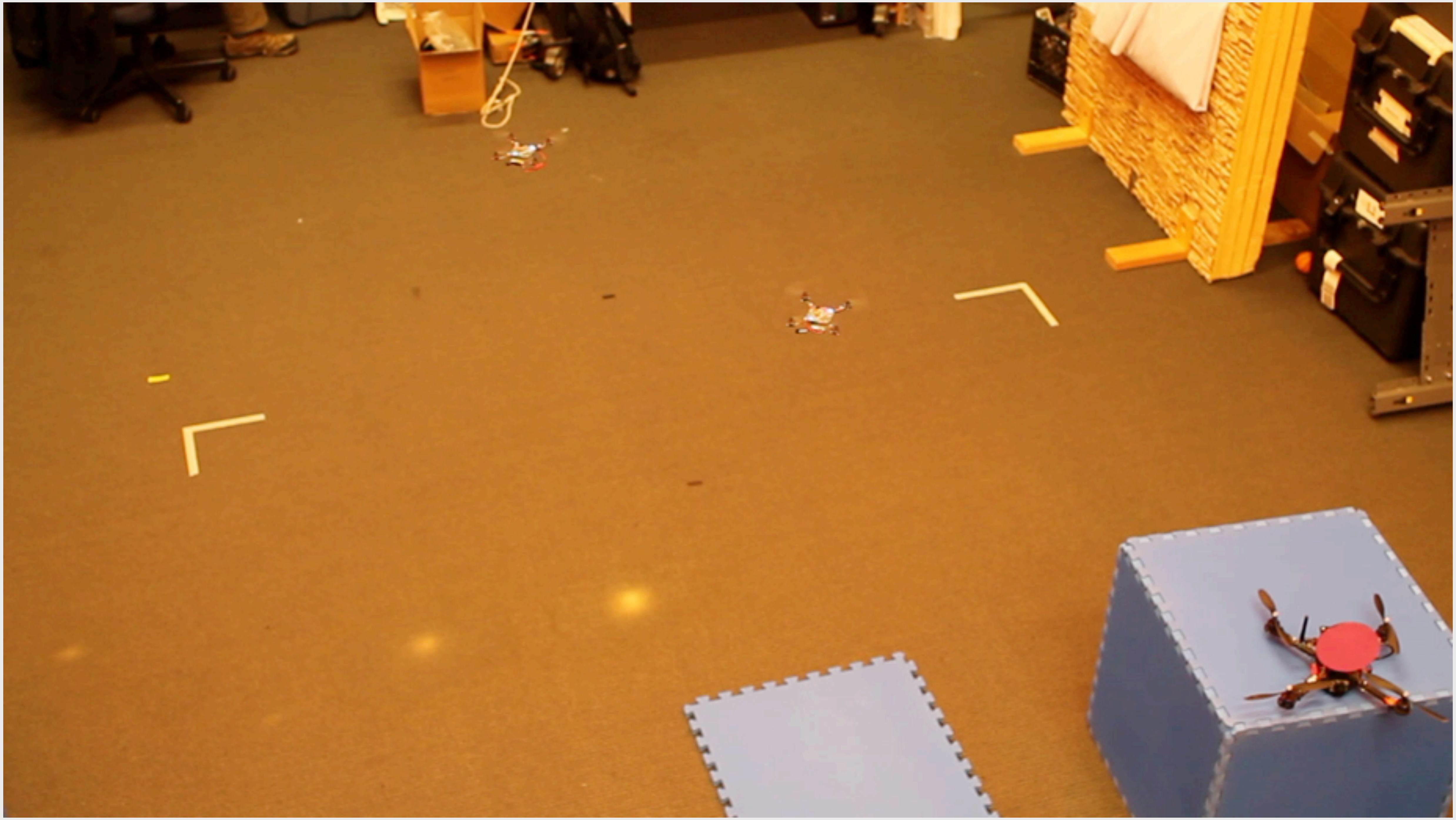

Representations and Algorithms for Finite-State Bisimulations of Linear Discrete-Time Control Systems

Andrew Lamperski

Experiments



Experiments



An alternative to explicit discretization:
no explicit discretization

An alternative to explicit discretization:
no explicit discretization

CDC 2016

Automata Theory Meets Approximate Dynamic Programming: Optimal Control with Temporal Logic Constraints

Ivan Papusha[†] Jie Fu[*] Ufuk Topcu[‡] Richard M. Murray[†]

An alternative to explicit discretization:
no explicit discretization

CDC 2016

Automata Theory Meets Approximate Dynamic Programming: Optimal Control with Temporal Logic Constraints

Ivan Papusha[†] Jie Fu[*] Ufuk Topcu[‡] Richard M. Murray[†]

TAC 2015

Automata Theory Meets Barrier Certificates: Temporal Logic Verification of Nonlinear Systems

Tichakorn Wongpiromsarn[*] Ufuk Topcu[†] Andrew Lamperski[‡]

Problem statement

Given

System model

$$\dot{x} = f(x, u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$

continuous time, continuous state
with assumptions on f for existence,
uniqueness and Zeno-freeness of solutions

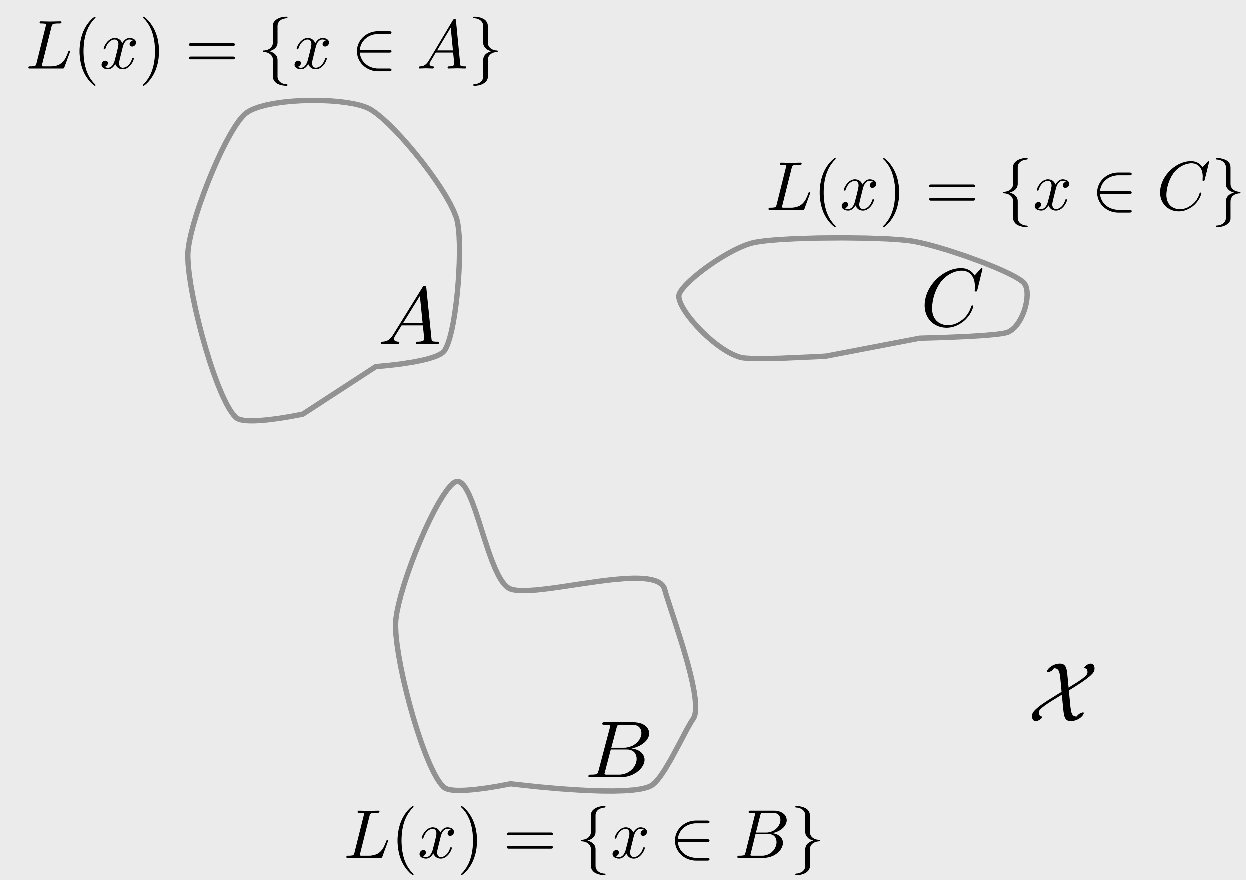
Problem statement

Given

System model

$$\dot{x} = f(x,u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$



Labeling function $\quad L : \mathcal{X} \to \Sigma = 2^{\mathcal{AP}}$

(what properties hold at a given state?)

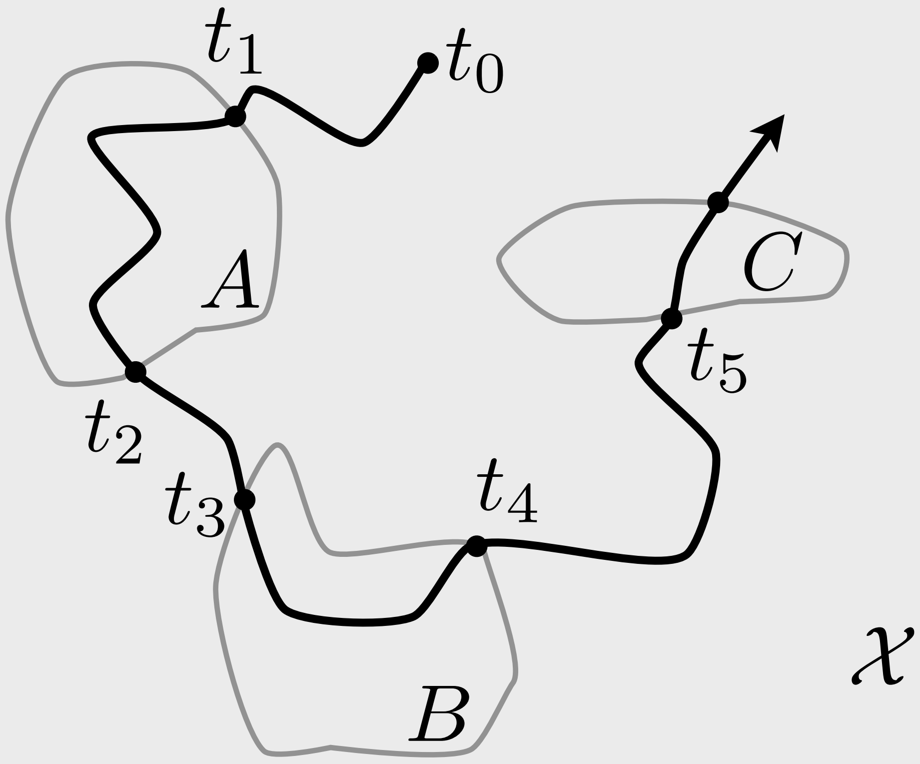
Problem statement

Given

System model

$$\dot{x} = f(x, u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \ u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$



Labeling function $L : \mathcal{X} \to \Sigma = 2^{\mathcal{AP}}$

(what properties hold at a given state?)

$$0 = t_0 < t_1 < \cdots < t_N = T$$

$$L(x(t)) = L(x(t_k)), t_k \leq t < t_{k+1}$$

$$L(x(t_k^-)) \neq L(x(t_k^+))$$

Problem statement

Given

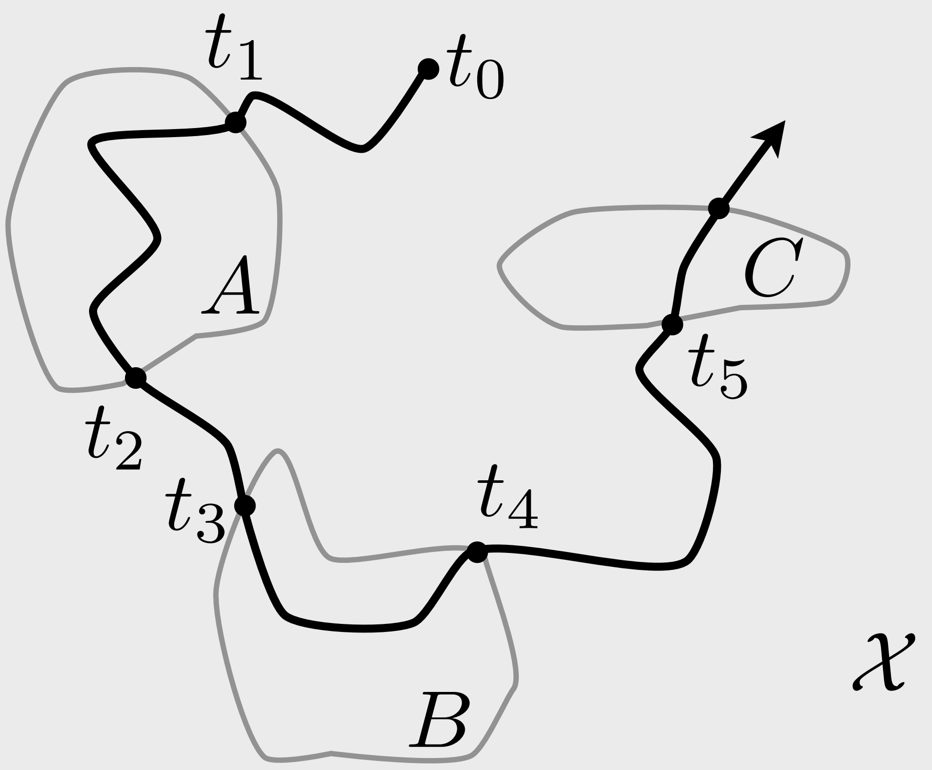
System model

$$\dot{x} = f(x, u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$

Labeling function $L : \mathcal{X} \to \Sigma = 2^{\mathcal{AP}}$

(what properties hold at a given state?)



$$0 = t_0 < t_1 < \cdots < t_N = T$$

$$L(x(t)) = L(x(t_k)), t_k \leq t < t_{k+1}$$

$$L(x(t_k^-)) \neq L(x(t_k^+))$$

“discrete” behavior: $\mathbb{B}(\phi(x_0, [0,T], u)) = \sigma_0 \sigma_1 \ldots \sigma_{N-1} \in \Sigma^*$

with $\sigma_k = L(x(t_k))$

Problem statement

Given

System model

$$\dot{x} = f(x, u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \;\; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$

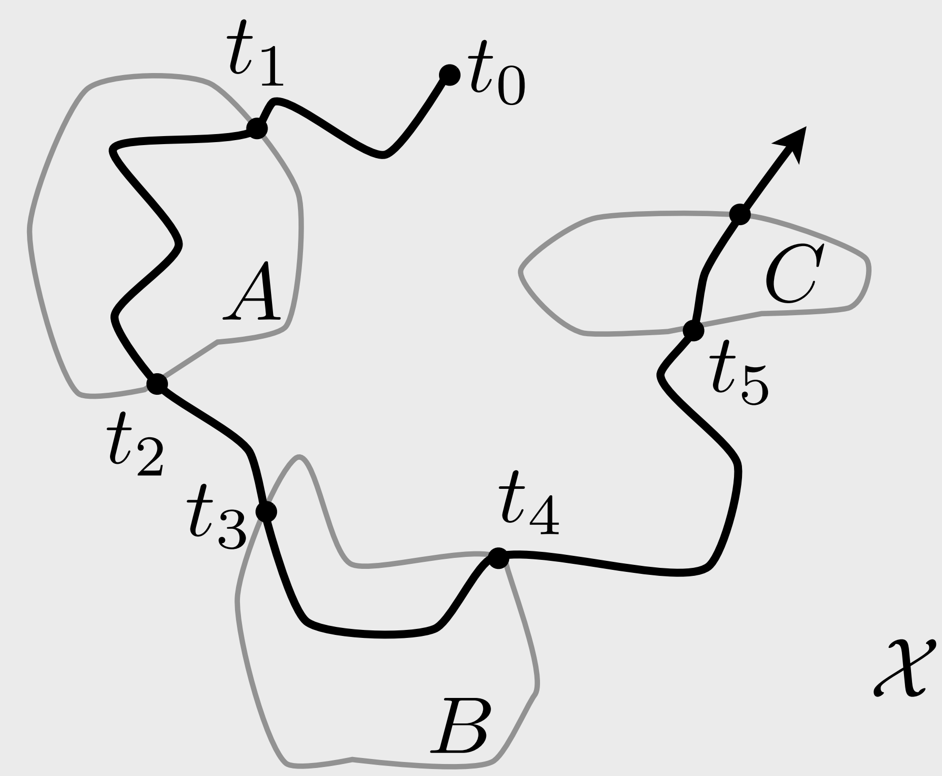
Labeling function $\quad L : \mathcal{X} \to \Sigma = 2^{\mathcal{AP}}$

(what properties hold at a given state?)

Co-safe temporal logic specification φ

(every satisfying word has a finite “good” prefix)

A final state $\quad x_f \in \mathcal{X}$ and a final time T.



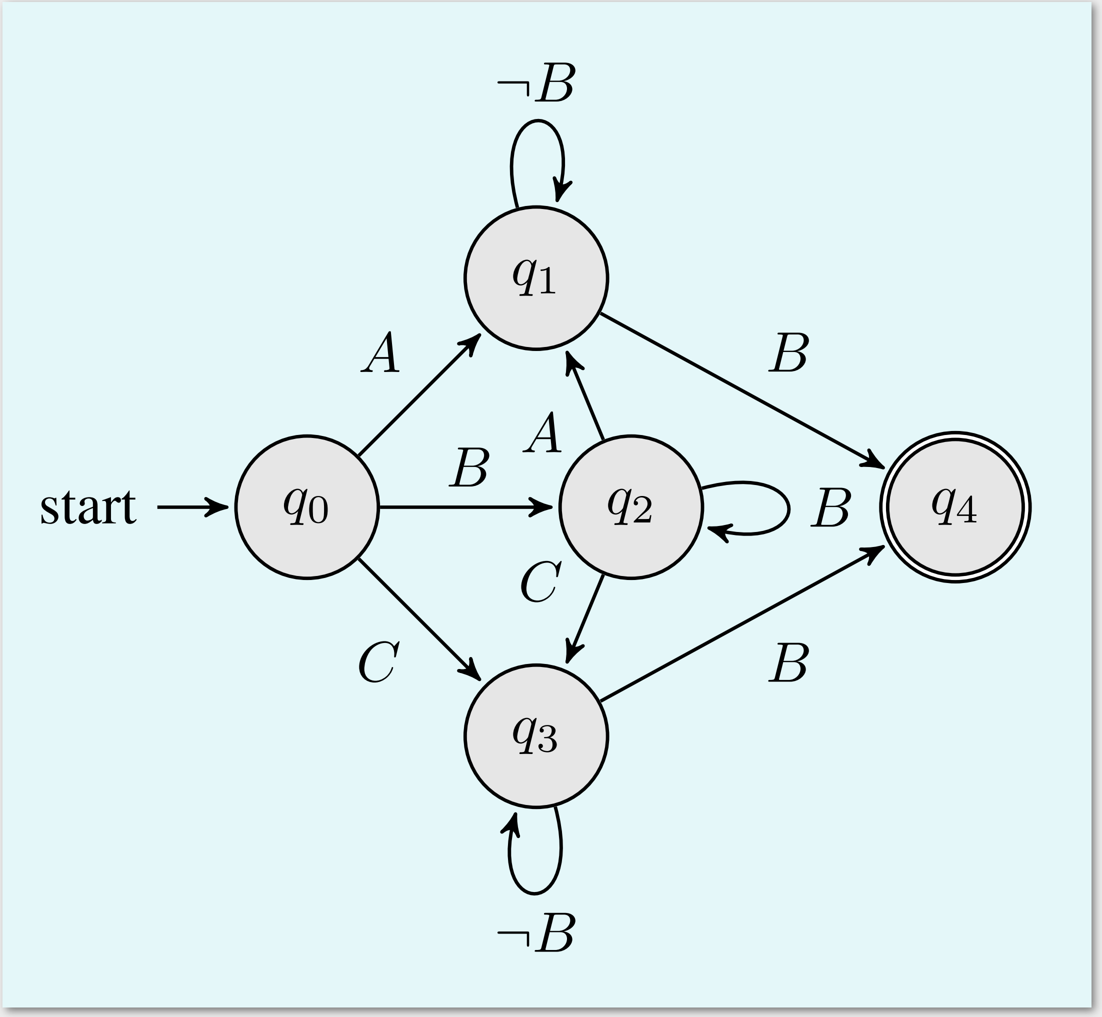
$$\mathbb{B}(\phi(x_0, [0, T], u)) = \sigma_0 \sigma_1 \ldots \sigma_{N-1} \in \Sigma^*$$

with $\sigma_k = L(x(t_k))$

De-tour: Automaton representation for temporal logic

$$(A \rightarrow \Diamond B) \wedge (C \rightarrow \Diamond B) \wedge (\Diamond A \vee \Diamond C)$$

Machine-interpretable representation of all words that satisfy the corresponding temporal logic formula



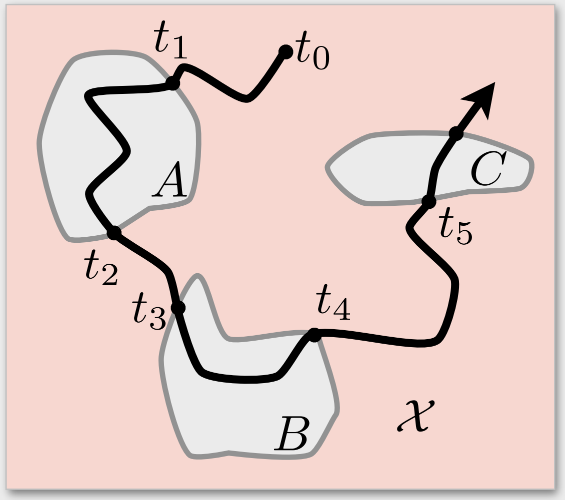
Deterministic finite automata are sufficient for co-safe linear temporal logic formulas

Problem statement (2)

Model

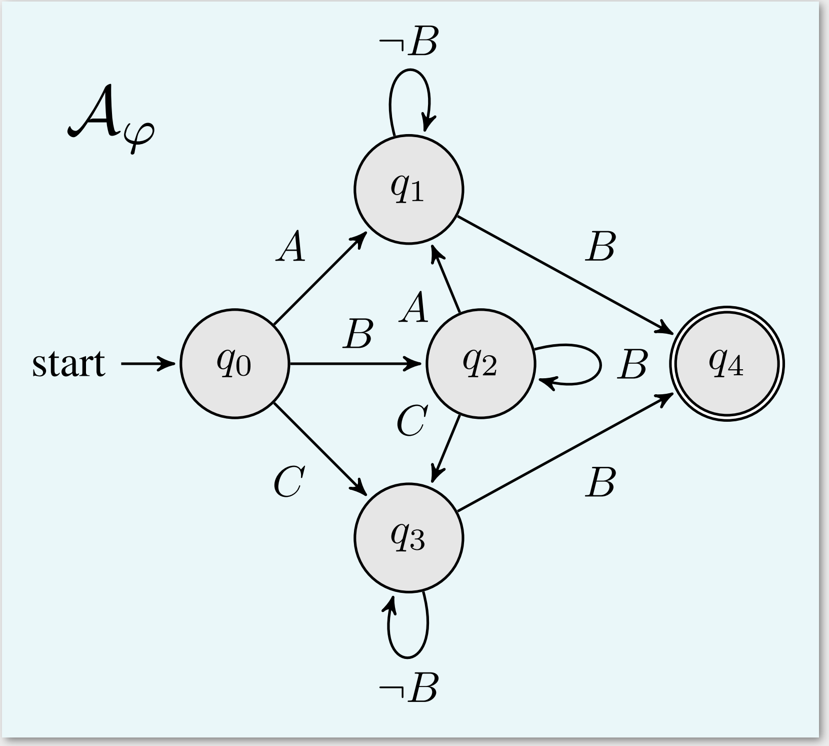
$$\dot{x} = f(x, u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$



+

Specification φ

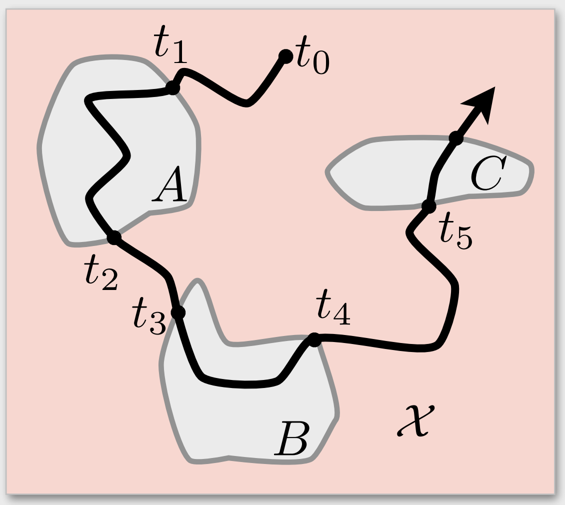


Problem statement (2)

Model

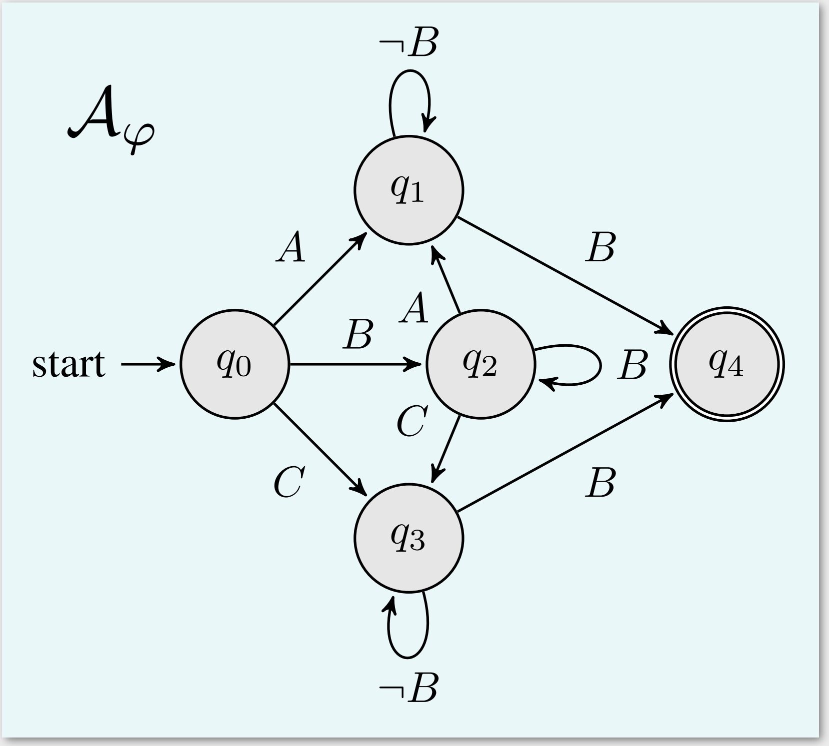
$$\dot{x} = f(x, u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$



+

Specification φ



Compute a control law u that minimizes

$$\int_0^T \ell(x(\tau), u(\tau))\, d\tau + \sum_{k=0}^{N} s(x(t_k), q(t_k^-), q(t_k^+))$$

l: loss function

s: cost of mode transition

subject to $x(T) = x_f$ and

$$\mathbb{B}(\phi(x_0, [0, T], u)) \in \mathcal{L}(\mathcal{A}_\varphi).$$

all discrete behavior satisfies the specification

Related work

$$\int_0^T \ell(x(\tau), u(\tau))\, d\tau$$

$$\dot{x} = f(x, u), \quad x(0) = x_0$$
$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \;\; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$

Temporal logic specification

$$(A \rightarrow \Diamond B) \wedge (C \rightarrow \Diamond B) \wedge (\Diamond A \vee \Diamond C)$$

restrict to simple specifications

make it a formal methods problem

Related work

$$\int_0^T \ell(x(\tau), u(\tau))\, d\tau$$

$$\dot{x} = f(x, u), \quad x(0) = x_0$$

$$x(t) \in \mathcal{X} \subseteq \mathbb{R}^n, \;\; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$$

Temporal logic specification

$$(A \rightarrow \Diamond B) \wedge (C \rightarrow \Diamond B) \wedge (\Diamond A \vee \Diamond C)$$

restrict to simple specifications

Hedlund & Rantzer
(optimal control for hybrid systems
+ convex dynamic programming)

Xu & Antsaklis
(optimal control for switched systems)

Kariotoglou, et al.
(approximate dynamic programming
for stochastic reachability)

make it a formal methods problem

Habets & Belta

Wongpiromsarn, et al.

Wolff, et al.

Fainekos, et al.

Product hybrid system

The problem can be formulated as a dynamic programming problem over a **product hybrid system**:

$$\langle\ Q,\ \mathcal{X},\ E,\ f,\ R,\ G\ \rangle$$

Product hybrid system

The problem can be formulated as a dynamic programming problem over a **product hybrid system**:

set of continuous states

continuous vector field

$$\langle\ Q,\ \mathcal{X},\ E,\ f,\ R,\ G\ \rangle$$

set of states of the specification automaton

$E \subseteq Q \times \Sigma \times Q$

set of discrete transitions

Product hybrid system

The problem can be formulated as a dynamic programming problem over a **product hybrid system**:

set of continuous states

continuous vector field

$$\langle\ Q,\ \mathcal{X},\ E,\ f,\ R,\ G\ \rangle$$

set of states of the specification automaton

$E \subseteq Q \times \Sigma \times Q$

set of discrete transitions

- The continuous state x evolves according to the vector field.
- The evolution of the discrete state q is governed by the automaton.
- A discrete transition is triggered when x crosses a boundary between two labeled regions.

Product hybrid system

The problem can be formulated as a dynamic programming problem over a **product hybrid system**:

set of continuous states

continuous vector field

$$\langle\, Q,\ \mathcal{X},\ E,\ f,\ R,\ G\, \rangle$$

set of states of the specification automaton

$E \subseteq Q \times \Sigma \times Q$

set of discrete transitions

$R = \{R_q \mid q \in Q\}$

R_q refers to x reachable while the automaton is in or transitions to mode q

a collection of guard regions in X:
For each q, x evolves inside R_q until it enters a guard region $G_{(q,\sigma,q')}$ and a transition to q' is made

- The continuous state x evolves according to the vector field.
- The evolution of the discrete state q is governed by the automaton.
- A discrete transition is triggered when x crosses a boundary between two labeled regions.

Dynamic programming formulation

Hybrid Hamilton-Jacobi-Bellman equations over the product space

V*: optimal cost-to-go subject to the specifications

$$0 = \min_{u \in \mathcal{U}} \left\{ \frac{\partial V^\star(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \right\}$$

$$\forall x \in R_q, \ \forall q \in Q$$

$$V^\star(x,q) = \min_{q'} \{ V^\star(x,q') + s(x,q,q') \}$$

$$\forall x \in G_e, \ \forall e = (q, \sigma, q') \in E$$

Dynamic programming formulation

Hybrid Hamilton-Jacobi-Bellman equations over the product space

V*: optimal cost-to-go subject to the specifications

While the labels remain constant:

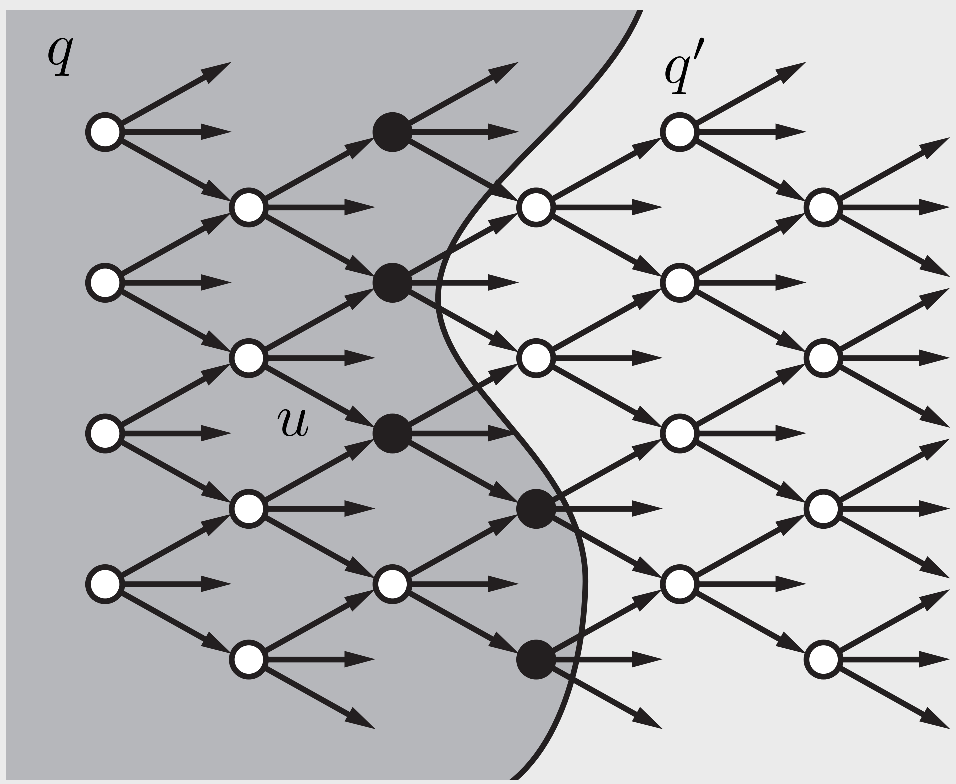
$$0 = \min_{u \in \mathcal{U}} \left\{ \frac{\partial V^\star(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \right\}$$

$$\forall x \in R_q, \ \forall q \in Q$$

Over discrete transitions:

$$V^\star(x,q) = \min_{q'} \{ V^\star(x,q') + s(x,q,q') \}$$

$$\forall x \in G_e, \ \forall e = (q, \sigma, q') \in E$$



Dynamic programming formulation

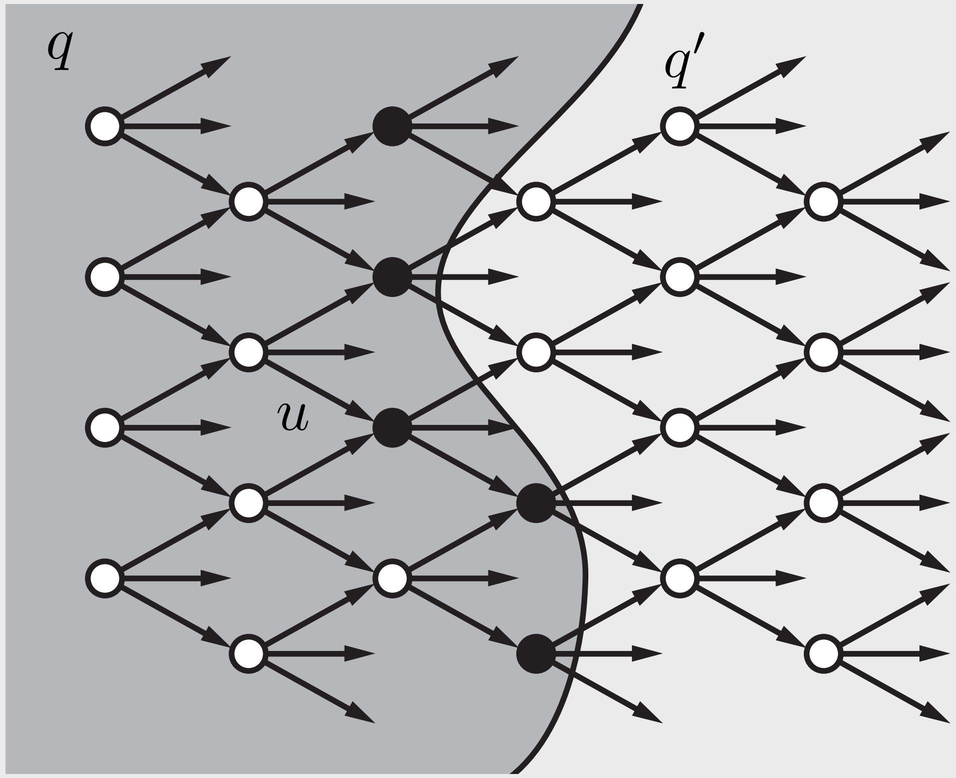
Hybrid Hamilton-Jacobi-Bellman equations over the product space

V*: optimal cost-to-go subject to the specifications

While the labels remain constant:

$$0 = \min_{u \in \mathcal{U}} \left\{ \frac{\partial V^\star(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \right\}$$

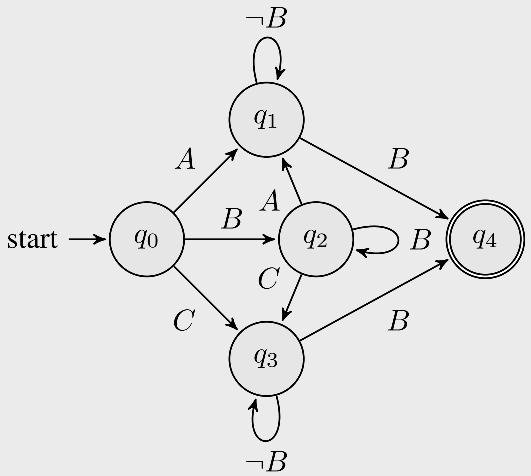
$$\forall x \in R_q,\ \forall q \in Q$$



Over discrete transitions:

$$V^\star(x,q) = \min_{q'} \{ V^\star(x,q') + s(x,q,q') \}$$

$$\forall x \in G_e,\ \forall e = (q,\sigma,q') \in E$$



At the "terminal" state:

$$0 = V^\star(x_f, q_f), \quad \forall q_f \in F$$

(Toward computable) lower bounds on the optimal cost

$$0 \leq \frac{\partial V(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \qquad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

$$0 \leq V(x,q') - V(x,q) + s(x,q,q') \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E$$

$$0 = V(x_f, q_f), \quad \forall q_f \in F$$

(Toward computable) lower bounds on the optimal cost

$$0 \leq \frac{\partial V(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \qquad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

$$0 \leq V(x,q') - V(x,q) + s(x,q,q') \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E$$

$$0 = V(x_f, q_f), \quad \forall q_f \in F$$

***V*: approximate value function**

A function V that satisfies the above conditions is an under-estimator for the optimal value function V*:

$$V(x_0,q_0) \leq V^\star(x_0,q_0)$$

(Toward computable) lower bounds on the optimal cost

$$0 \leq \frac{\partial V(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \qquad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

compare to $\quad 0 = \min_{u \in \mathcal{U}} \left\{ \frac{\partial V^\star(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \right\} \qquad \forall x \in R_q,\ \forall q \in Q$

$$0 \leq V(x,q') - V(x,q) + s(x,q,q') \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E$$

compare to $\quad V^\star(x,q) = \min_{q'} \{V^\star(x,q') + s(x,q,q')\} \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E$

$$0 = V(x_f, q_f), \quad \forall q_f \in F$$

***V*: approximate value function**

A function V that satisfies the above conditions is an under-estimator for the optimal value function V*:

$$V(x_0, q_0) \leq V^\star(x_0, q_0)$$

(Toward computable) lower bounds on the optimal cost

$$0 \leq \frac{\partial V(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \qquad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

compare to $0 = \min_{u \in \mathcal{U}} \left\{ \frac{\partial V^\star(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \right\} \qquad \forall x \in R_q,\ \forall q \in Q$

$$0 \leq V(x,q') - V(x,q) + s(x,q,q') \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E$$

compare to $V^\star(x,q) = \min_{q'} \{V^\star(x,q') + s(x,q,q')\} \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E$

$$0 = V(x_f, q_f), \quad \forall q_f \in F$$

***V*: approximate value function**

A function V that satisfies the above conditions is an under-estimator for the optimal value function V*:

$$V(x_0,q_0) \leq V^\star(x_0,q_0)$$

Intuition from purely discrete version:

$$V^* = \mathbb{T}V^*$$

$$V \leq \mathbb{T}V \Rightarrow V \leq V^*$$

Approximate value function and approximately optimal control law

Parametrize V with pre-specified basis functions ϕ:

$$V(x,q) = \sum_{i=1}^{n_q} w_{i,q}\phi_{i,q}(x)$$

basis:
function of x,
indexed by q

Search for approximate value function that maximizes $V(x_0,q_0)$.

(one of the many scalarizations)

Approximate value function and approximately optimal control law

Parametrize V with pre-specified basis functions ϕ:

$$V(x,q) = \sum_{i=1}^{n_q} w_{i,q}\phi_{i,q}(x)$$

basis:
function of x,
indexed by q

Search for approximate value function that maximizes $V(x_0,q_0)$.

(one of the many scalarizations)

Given V, an approximately optimal control law:

$$u(x,q) = \arg\min_{u\in\mathcal{U}} \left\{ \frac{\partial V(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \right\}$$

Mode switchings are autonomous, driven by the evolution of x.

Search for approximate value function

Linear system: $\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0,$

Quadratic continuous cost: $\ell(x, u) = x^T Q x + u^T R u, \quad Q \succeq 0, \quad R \succ 0$

Constant switching cost: $s(x, q, q') = \xi \cdot \mathbb{I}\left(\{(q, q') \mid q \neq q'\}\right)$

For each $q \in Q$, parametrize V by P_q, r_q, t_q: $V(x, q) = x^T P_q x + 2 r_q^T x + t_q$

Search for approximate value function

Linear system: $\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0,$

Quadratic continuous cost: $\ell(x,u) = x^T Q x + u^T R u, \quad Q \succeq 0, \quad R \succ 0$

Constant switching cost: $s(x,q,q') = \xi \cdot \mathbb{I}\left(\{(q,q') \mid q \neq q'\}\right)$

For each $q \in Q$, parametrize V by P_q, r_q, t_q: $V(x,q) = x^T P_q x + 2r_q^T x + t_q$

$$\max_{P_q, r_q, t_q} \quad V(x_0, q_0) = x_0^T P_{q_0} x_0 + 2r_{q_0}^T x_0 + t_{q_0} \qquad \text{subject to}$$

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \quad \forall x \in R_q, \ \forall u \in \mathcal{U}, \ \forall q \in Q$$

$$0 \leq \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} P_{q'} - P_q & r_{q'} - r_q \\ r_{q'}^T - r_q^T & t_{q'} - t_q + \xi \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \forall x \in G_e, \ \forall e \in E$$

$$0 = x_f^T P_{q_f} x_f + 2r_{q_f}^T x_f + t_{q_f} \qquad \forall q_f \in F$$

Search for approximate value function

Linear system: $\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0,$

Quadratic continuous cost: $\ell(x,u) = x^T Q x + u^T R u, \quad Q \succeq 0, \quad R \succ 0$

Constant switching cost: $s(x,q,q') = \xi \cdot \mathbb{I}\left(\{(q,q') \mid q \neq q'\}\right)$

For each $q \in Q$, parametrize V by P_q, r_q, t_q: $V(x,q) = x^T P_q x + 2r_q^T x + t_q$

semi-infinite optimization problem

$$\max_{P_q, r_q, t_q} \quad V(x_0, q_0) = x_0^T P_{q_0} x_0 + 2 r_{q_0}^T x_0 + t_{q_0} \qquad \text{subject to}$$

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \qquad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

$$0 \leq \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} P_{q'} - P_q & r_{q'} - r_q \\ r_{q'}^T - r_q^T & t_{q'} - t_q + \xi \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \forall x \in G_e,\ \forall e \in E$$

$$0 = x_f^T P_{q_f} x_f + 2 r_{q_f}^T x_f + t_{q_f} \qquad \forall q_f \in F$$

Solving the semi-infinite optimization problem

$$\max_{P_q, r_q, t_q} \quad V(x_0, q_0) = x_0^T P_{q_0} x_0 + 2 r_{q_0}^T x_0 + t_{q_0} \qquad \text{subject to}$$

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \quad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

$$0 \leq \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} P_{q'} - P_q & r_{q'} - r_q \\ r_{q'}^T - r_q^T & t_{q'} - t_q + \xi \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \forall x \in G_e,\ \forall e \in E$$

$$0 = x_f^T P_{q_f} x_f + 2 r_{q_f}^T x_f + t_{q_f} \qquad \forall q_f \in F$$

For *quadratically representable* R_q, G_e and U,

(1) use the S-procedure to resort to finite sufficient conditions for the semi-infinite constraints

(2) translate into a semidefinite program

Solving the semi-infinite optimization problem

$$\max_{P_q, r_q, t_q} \quad V(x_0, q_0) = x_0^T P_{q_0} x_0 + 2 r_{q_0}^T x_0 + t_{q_0} \qquad \text{subject to}$$

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \quad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

$$0 \leq \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} P_{q'} - P_q & r_{q'} - r_q \\ r_{q'}^T - r_q^T & t_{q'} - t_q + \xi \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \forall x \in G_e,\ \forall e \in E$$

$$0 = x_f^T P_{q_f} x_f + 2 r_{q_f}^T x_f + t_{q_f} \qquad \forall q_f \in F$$

"S-procedure"

$$M_0, M_1 : \mathbb{R}^n \to \mathbb{R}$$

$$\boxed{M_1 \geq 0 \Rightarrow M_0 \geq 0}$$

$$\Uparrow$$

$$\boxed{\begin{array}{c} \exists \lambda \geq 0 \text{ s.t.} \\ M_0(\zeta) - \lambda M_1(\zeta) \geq 0 \ \forall \zeta \end{array}}$$

For *quadratically representable R_q*, *G_e* and *U*,

(1) use the S-procedure to resort to finite sufficient conditions for the semi-infinite constraints

(2) translate into a semidefinite program

Solving the semi-infinite optimization problem

$$\max_{P_q, r_q, t_q} \quad V(x_0, q_0) = x_0^T P_{q_0} x_0 + 2 r_{q_0}^T x_0 + t_{q_0} \qquad \text{subject to}$$

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \quad \forall x \in R_q, \ \forall u \in \mathcal{U}, \ \forall q \in Q$$

$$0 \leq \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} P_{q'} - P_q & r_{q'} - r_q \\ r_{q'}^T - r_q^T & t_{q'} - t_q + \xi \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \forall x \in G_e, \ \forall e \in E$$

$$0 = x_f^T P_{q_f} x_f + 2 r_{q_f}^T x_f + t_{q_f} \qquad \forall q_f \in F$$

For *quadratically representable* R_q, G_e and U,

(1) use the S-procedure to resort to finite sufficient conditions for the semi-infinite constraints

(2) translate into a semidefinite program

Are R_q and G_e quadratically representable?

- Can be decided based on the atomic propositions in the specification.

Example

Linear quadratic system

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$Q = I, \quad R = 1, \quad \xi = 1,$$

$$x_f = (0, 0)$$

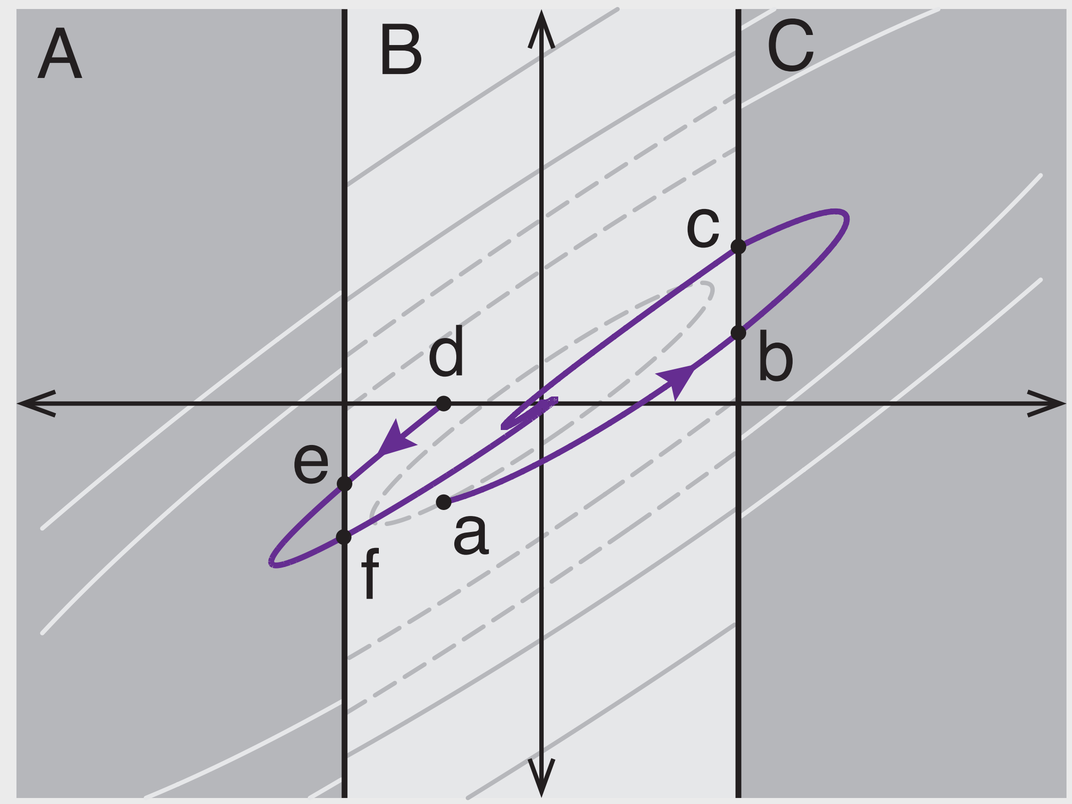
Example

Linear quadratic system

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

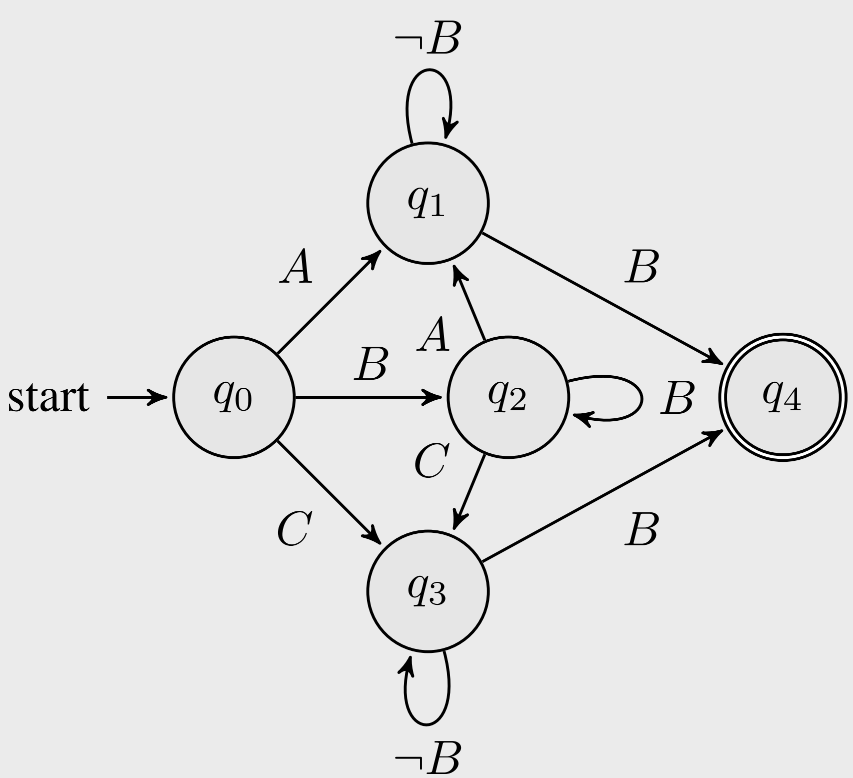
$$Q = I, \quad R = 1, \quad \xi = 1,$$

$$x_f = (0, 0)$$



Specification

$$(A \to \Diamond B) \wedge (C \to \Diamond B) \wedge (\Diamond A \vee \Diamond C)$$



Example

Linear quadratic system

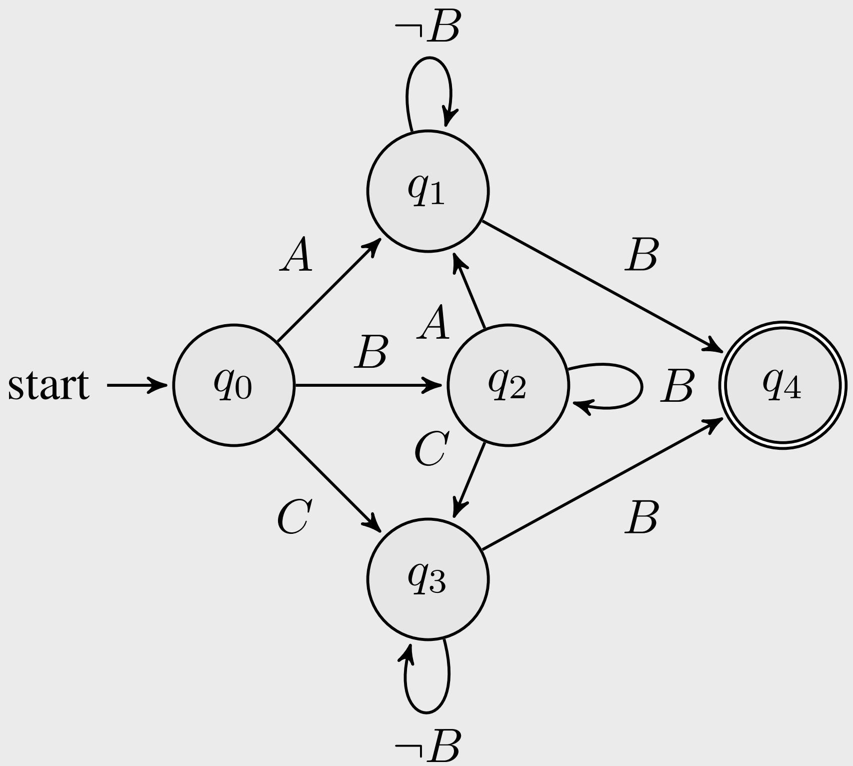
$$A = \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

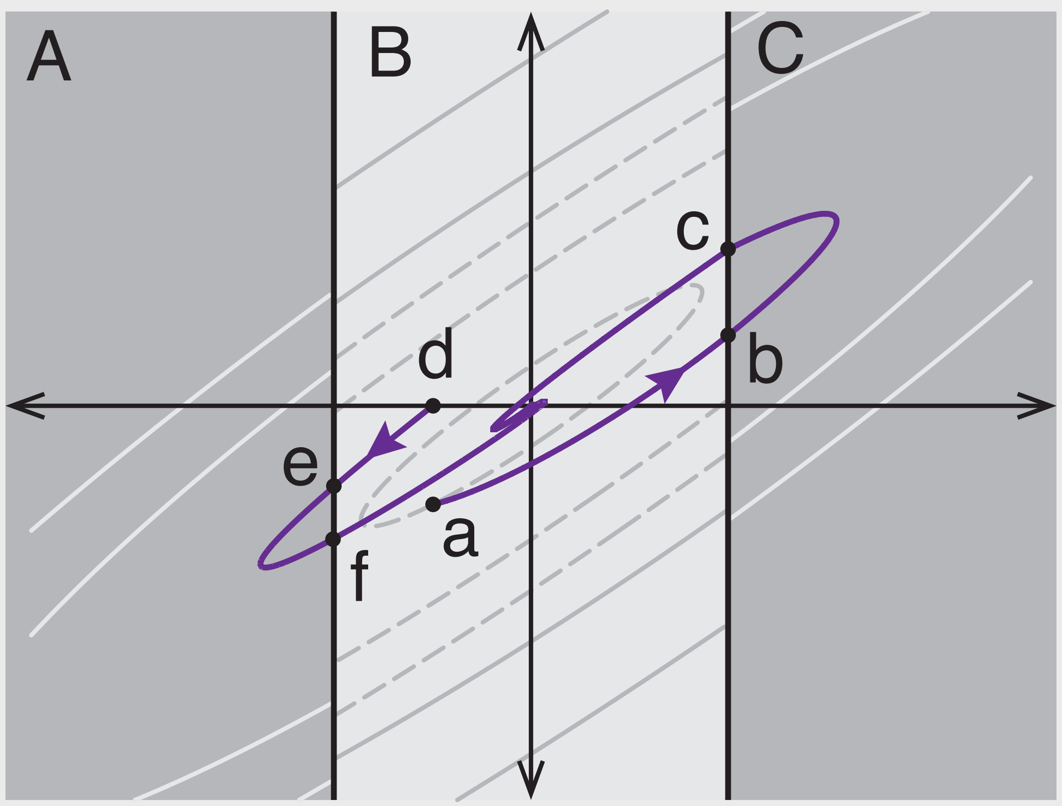
$$Q = I, \quad R = 1, \quad \xi = 1,$$

$$x_f = (0,0)$$

Specification

$$(A \to \Diamond B) \wedge (C \to \Diamond B) \wedge (\Diamond A \vee \Diamond C)$$





Compare the spectra of the closed-loop matrix in different modes

$$A_q^{\mathrm{cl}} = A - BR^{-1}B^T P_q^\star$$

$$\lambda(A_{q_0}^{\mathrm{cl}}) = \{0.786 \pm 1.144i\}$$

$$\lambda(A_{q_4}^{\mathrm{cl}}) = \{-1 \pm i\}$$

Example

Linear quadratic system

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

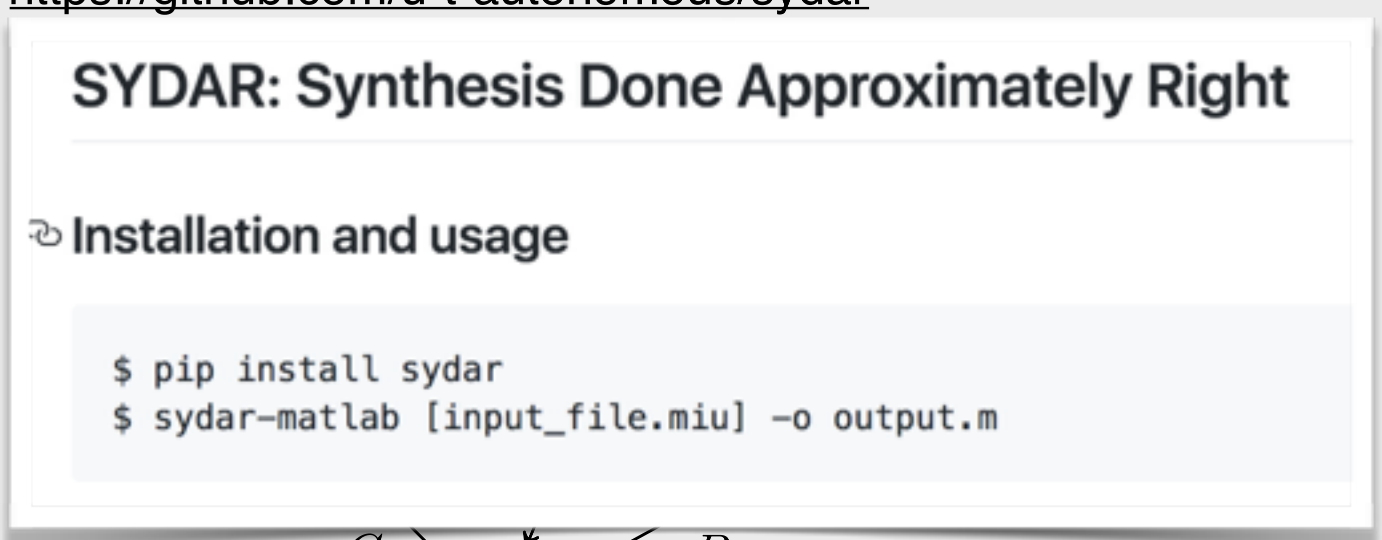
$$Q = I, \quad R = 1, \quad \xi = 1,$$

$$x_f = (0, 0)$$

Specification

$$(A \to \Diamond B) \wedge (C \to \Diamond B) \wedge (\Diamond A \vee \Diamond C)$$

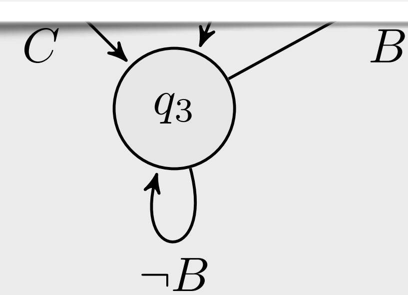
https://github.com/u-t-autonomous/sydar

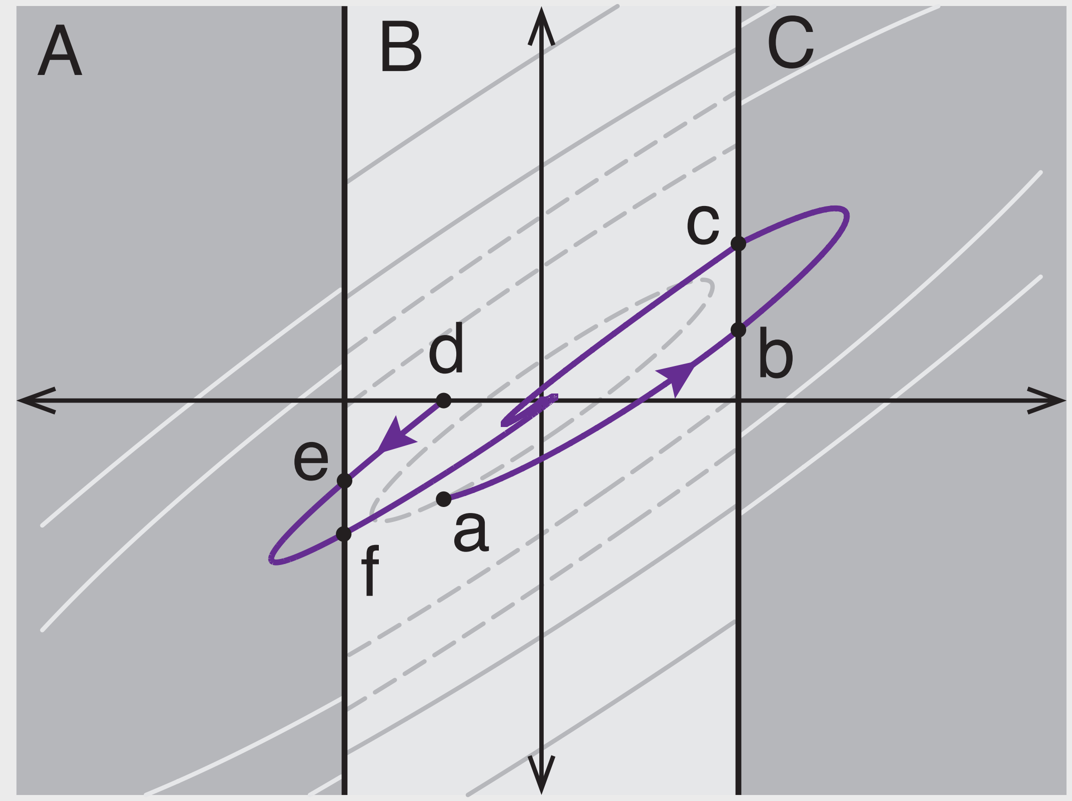


SYDAR: Synthesis Done Approximately Right

Installation and usage

```
$ pip install sydar
$ sydar-matlab [input_file.miu] -o output.m
```





Compare the spectra of the closed-loop matrix in different modes

$$A_q^{\mathrm{cl}} = A - BR^{-1}B^T P_q^\star$$

$$\lambda(A_{q_0}^{\mathrm{cl}}) = \{0.786 \pm 1.144i\}$$

$$\lambda(A_{q_4}^{\mathrm{cl}}) = \{-1 \pm i\}$$

Summary

No need for explicit finite abstraction
(w.r.t. the dynamics)

No need for expensive reachability
calculations

Summary

No need for explicit finite abstraction (w.r.t. the dynamics)

No need for expensive reachability calculations

Hope for scalability?

Scalability goal:
"Can we synthesize temporal-logic-constrained controllers for systems with **50 continuous states**?"

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}$$

$$\forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q$$

Summary

No need for explicit finite abstraction (w.r.t. the dynamics)

No need for expensive reachability calculations

Hope for scalability?

Scalability goal:
"Can we synthesize temporal-logic-constrained controllers for systems with **50 continuous states**?"

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}$$

$$\forall x \in R_q, \ \forall u \in \mathcal{U}, \ \forall q \in Q$$

Conservatism — S-procedure and basis selection

Policy is approximately optimal (bounds on sub optimality possible!)

Only co-safe temporal logic specifications (at this point)

What is next?

usual suspects

Demonstrate scalability

Reduce conservatism

Extend to broader classes dynamics — hybrid, nonlinear,…

Expand the family of specifications

new opportunities

Open up a broad set of new problems to ideas from controls and optimization

Automata Theory Meets Approximate Dynamic Programming: Optimal Control with Temporal Logic Constraints

Ivan Papusha† Jie Fu* Ufuk Topcu‡ Richard

Automata Theory Meets Barrier Certificates: Temporal Logic Verification of Nonlinear Systems

Tichakorn Wongpiromsarn* Ufuk Topcu† Andrew Lamperski‡

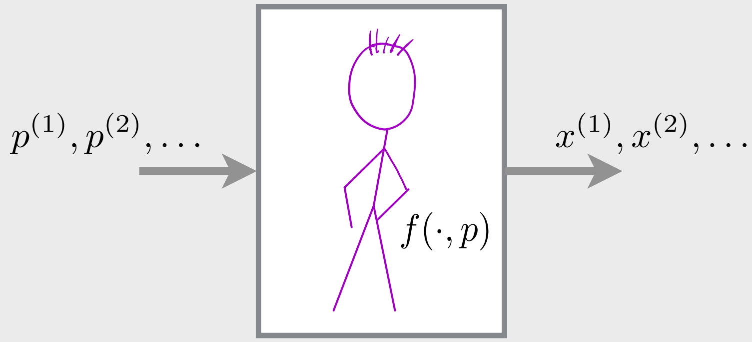
Learning from expert demonstrations

- Incorporating **expert demonstrations** into an autonomous system is difficult
- Even when expert demonstrations are somehow incorporated, generalizing to unseen scenarios can be **unsafe**.
- Can we generalize in a “safe” way using side information?
  - what does “safe” mean?
  - what kind of “side information” can be incorporated?

Learning by expert demonstration

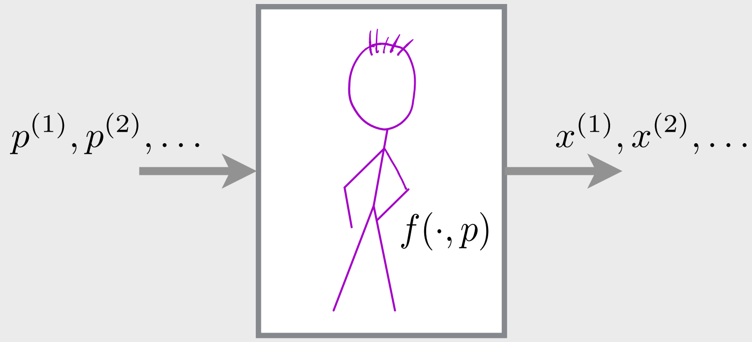
Learning by expert demonstration

1. Expert gives "optimal" demonstrations

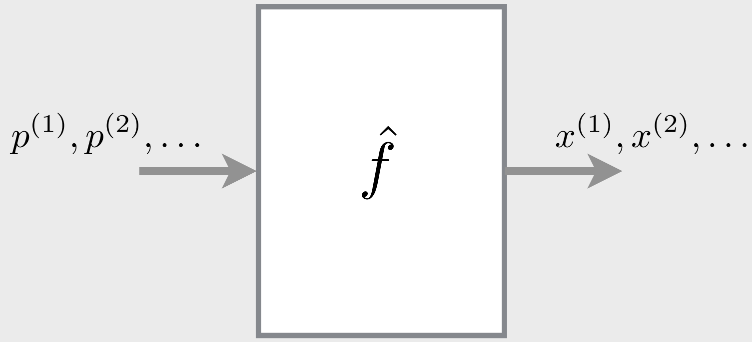


Learning by expert demonstration

1. Expert gives "optimal" demonstrations

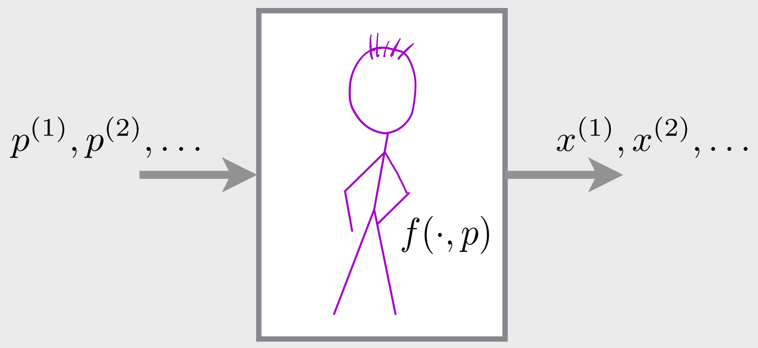


2. Demonstrations used to "learn" the expert

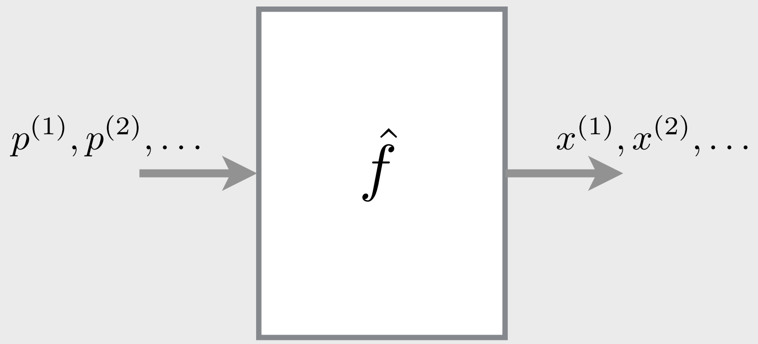


Learning by expert demonstration

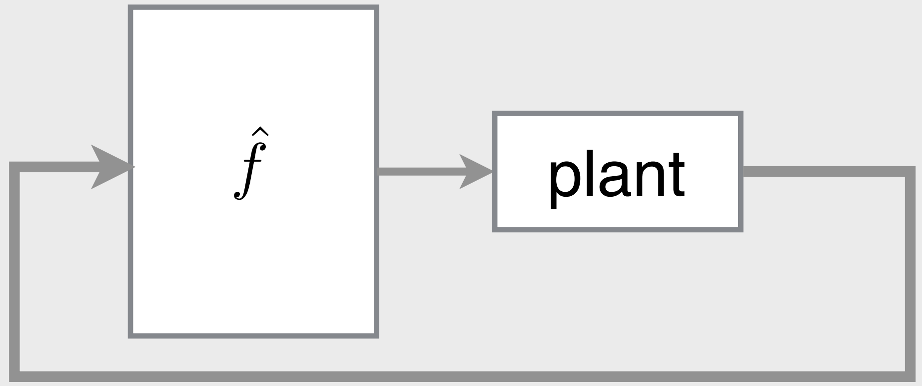
1. Expert gives “optimal” demonstrations



2. Demonstrations used to “learn” the expert

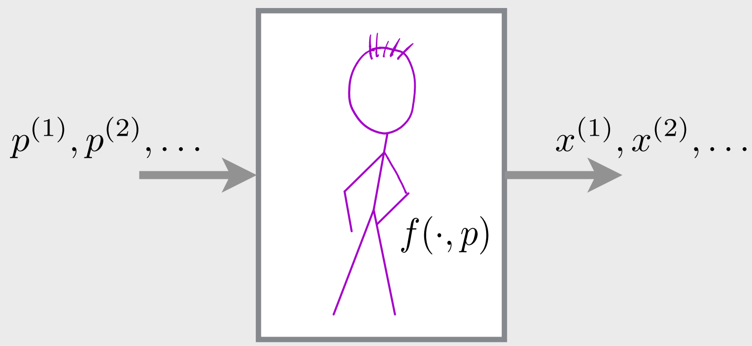


3. Learned objective (e.g. loss function) used to mimic the expert in an autonomous system

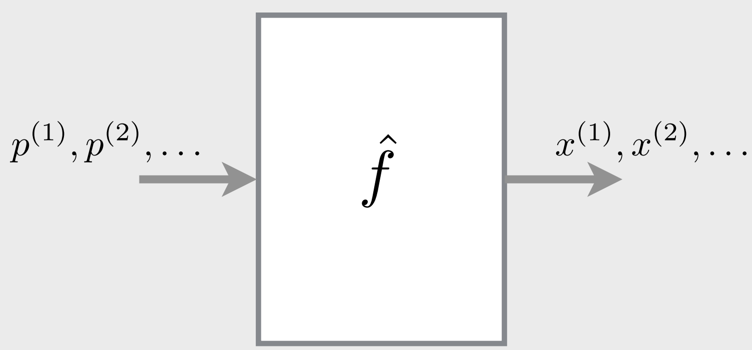


Learning by expert demonstration

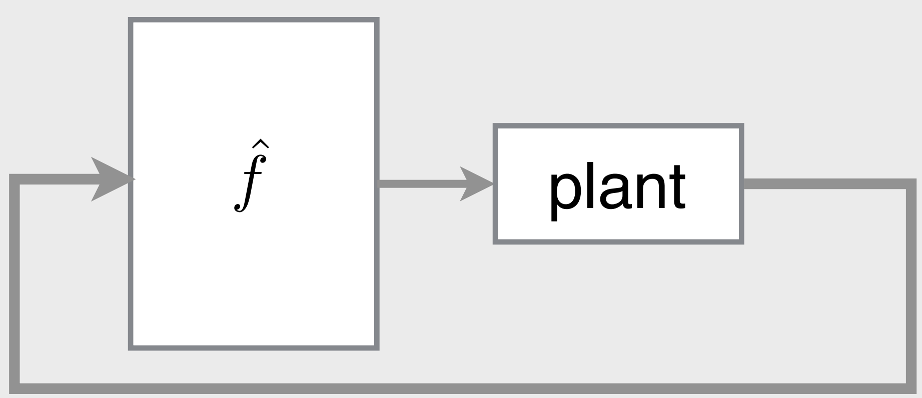
1. Expert gives “optimal” demonstrations



2. Demonstrations used to “learn” the expert



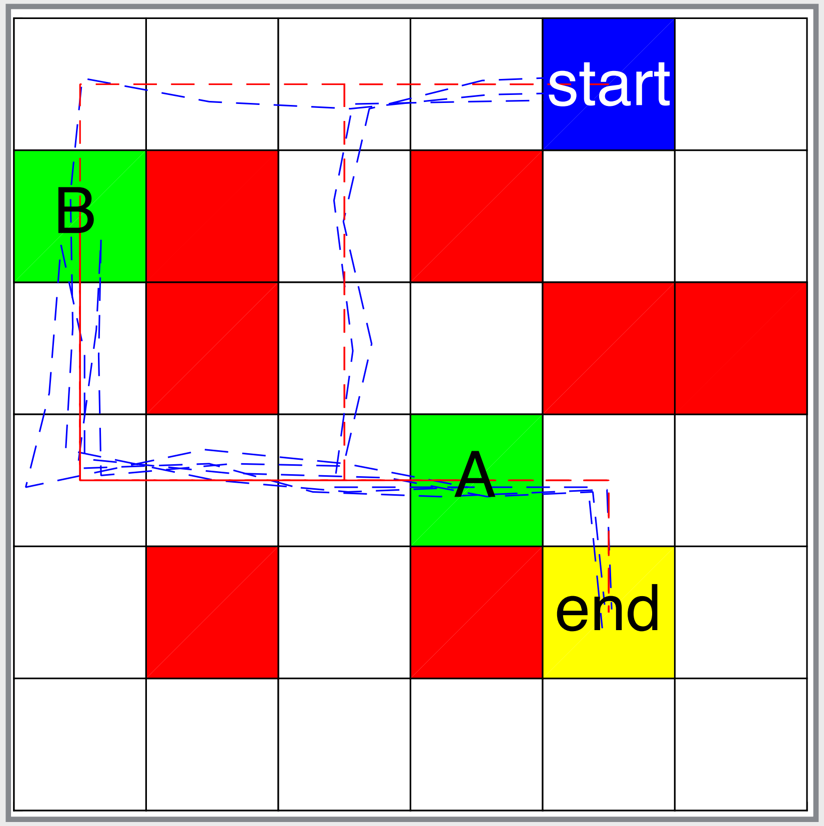
3. Learned objective (e.g. loss function) used to mimic the expert in an autonomous system



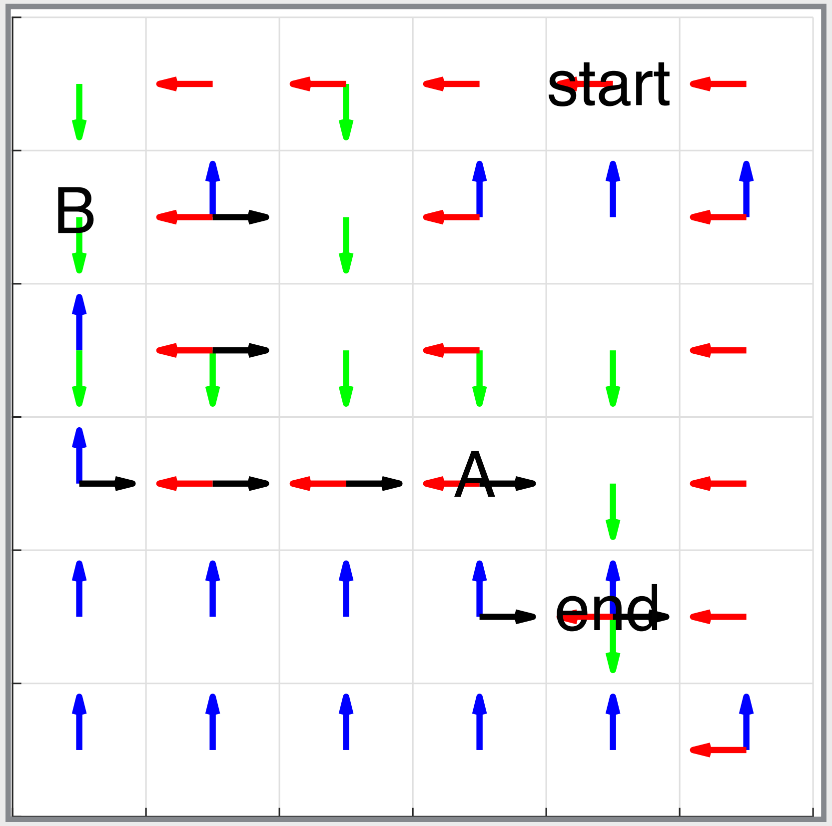
heli.stanford.edu

Learning fails when the expert is inconsistent

Expert demonstrations:



Learned policy:



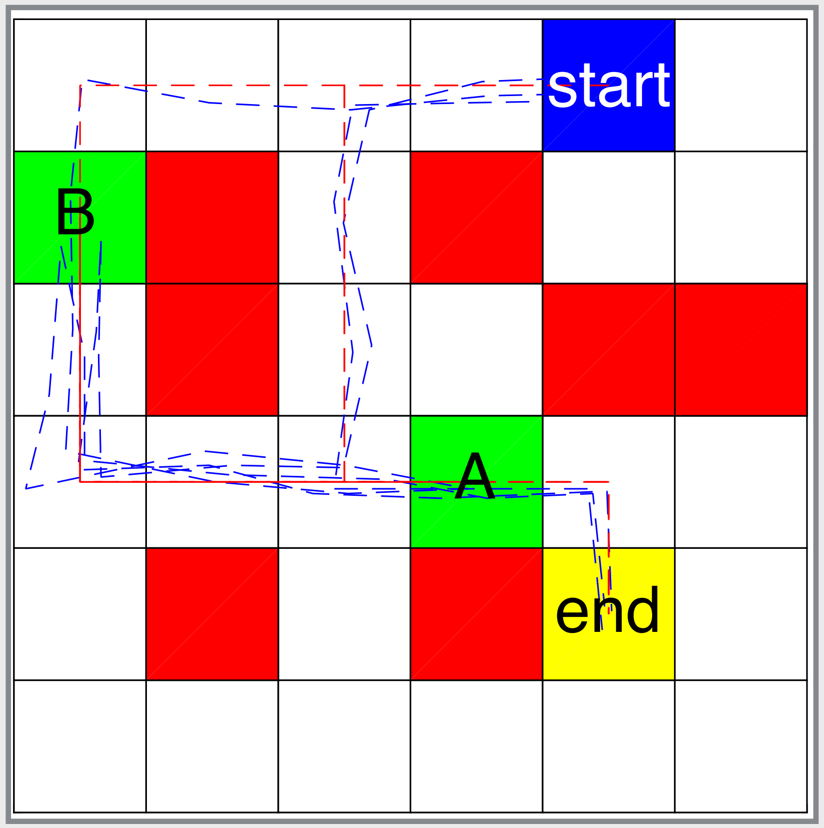
Task: get from start to end in the fewest steps, while visiting A and B in any order

- inverse optimal control applied to a grid world
- dynamics are modeled as a transition system
- learned an approximation to the optimal value function $V^\star(s)$

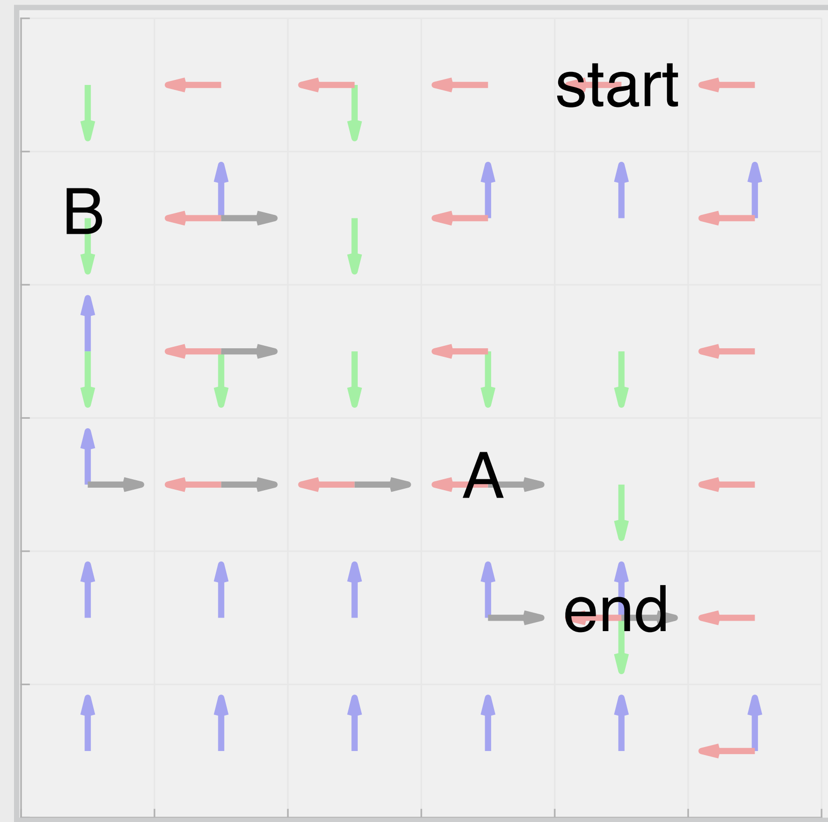
$$f(x, p) = \ell(s, s') + \hat{V}(s')$$

Learning fails when the expert is inconsistent

Expert demonstrations:



Learned policy:



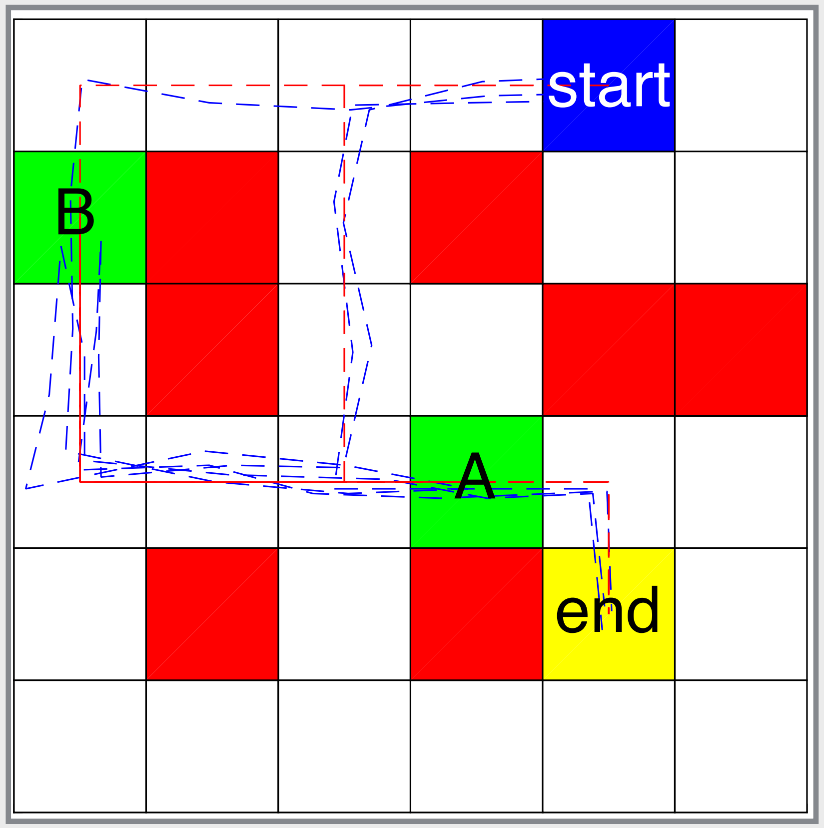
Task: get from start to end in the fewest steps, while visiting A and B in any order

- inverse optimal control applied to a grid world
- dynamics are modeled as a transition system
- learned an approximation to the optimal value function $V^\star(s)$

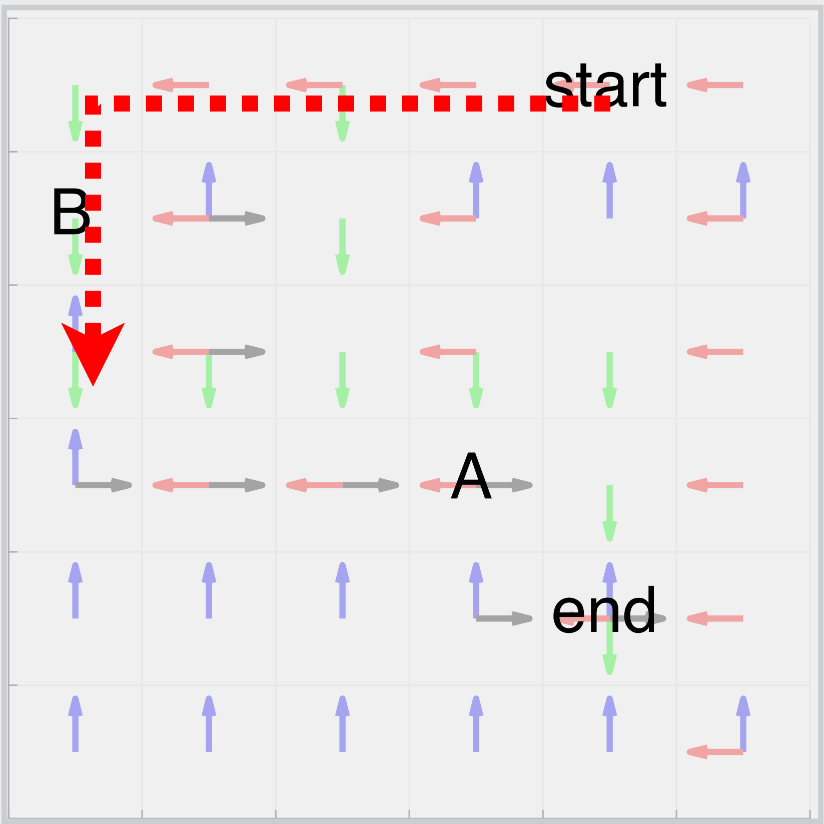
$$f(x, p) = \ell(s, s') + \hat{V}(s')$$

Learning fails when the expert is inconsistent

Expert demonstrations:



Learned policy:



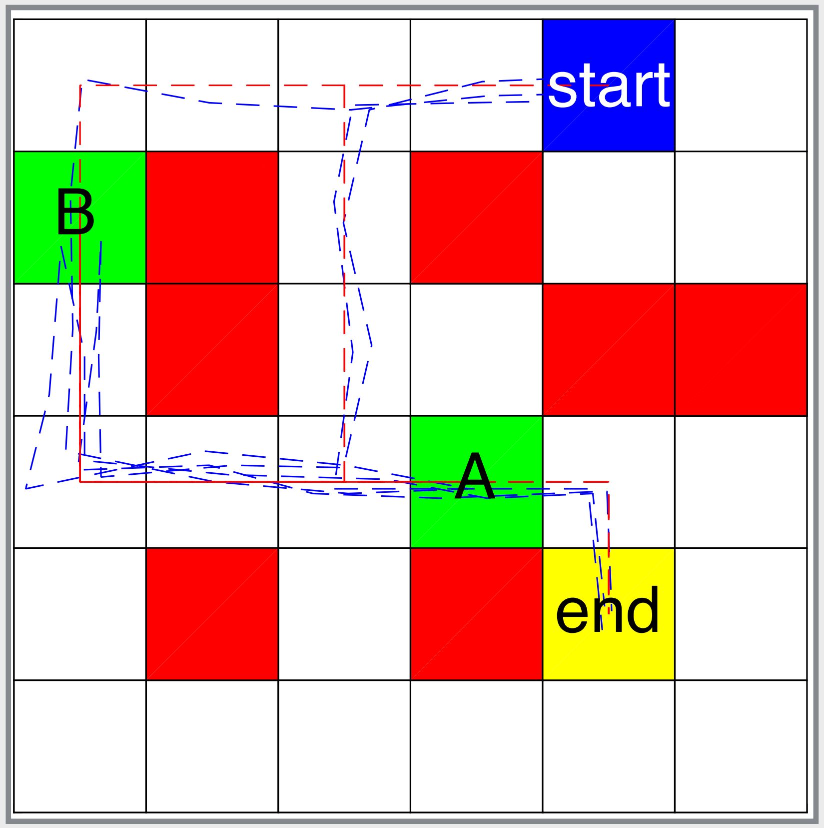
Task: get from start to end in the fewest steps, while visiting A and B in any order

- inverse optimal control applied to a grid world
- dynamics are modeled as a transition system
- learned an approximation to the optimal value function $V^\star(s)$

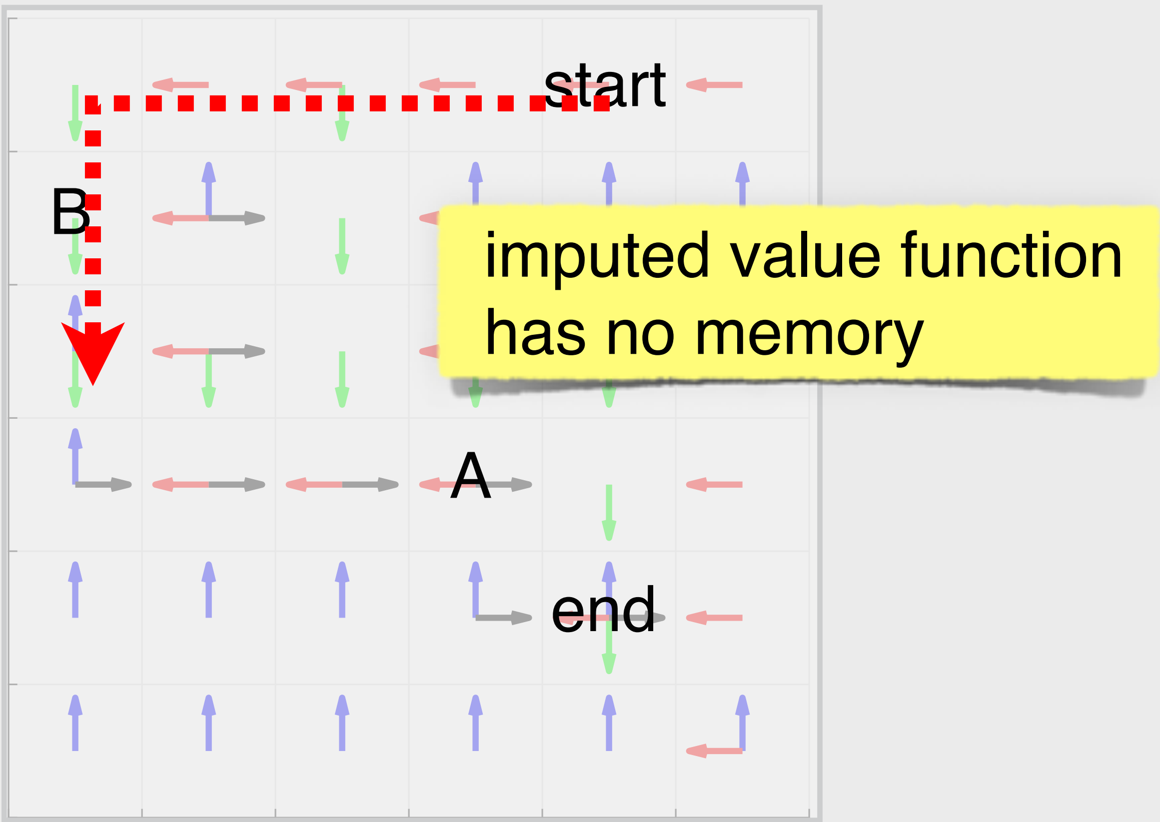
$$f(x,p) = \ell(s,s') + \hat{V}(s')$$

Learning fails when the expert is inconsistent

Expert demonstrations:



Learned policy:



Task: get from start to end in the fewest steps, while visiting A and B in any order

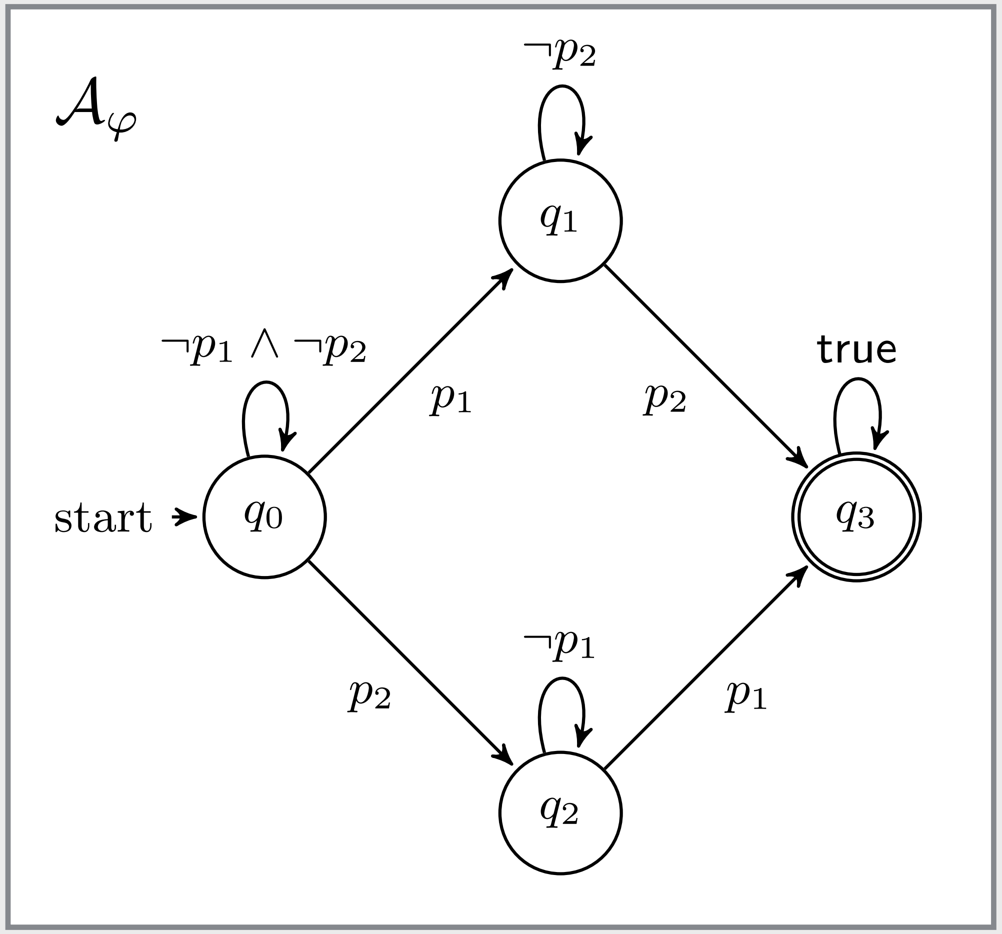
- inverse optimal control applied to a grid world
- dynamics are modeled as a transition system
- learned an approximation to the optimal value function $V^\star(s)$

$$f(x,p) = \ell(s,s') + \hat{V}(s')$$

Side information

Task: get from start to end in the fewest steps, while visiting A and B in any order

Side information automaton



time	0	1	2	3	…	t	t+1
state	s_1	s_2	s_3	s_4	…	s_t	s_{t+1}
p_1	F	F	T	F		T	F
p_2	F	F	F	F		T	F

Side information: a specification automaton that every "optimal" trajectory must satisfy.

At each time step the atomic propositions p_1 and p_2 are evaluated

- p_1 = true iff. the state is A
- p_2 = true iff. the state is B

In the inverse problem, the side information becomes a **hidden state** with (known) evolution

Data structures

Memoryless policy:

$$\mu_t^\star(s) \in \operatorname*{argmin}_{\{\alpha \mid s \xrightarrow{\alpha} s'\}} \left\{ \ell(s,\alpha,s') + V_{t+1}^\star(s') \right\}$$

Mode-varying policy:

$$\mu^\star(s,q) \in \operatorname*{argmin}_{\{\alpha \mid s \xrightarrow{\alpha} s', q' = \delta(q, L(s))\}} \{ \ell(s,\alpha,s') + V^\star(s',q') \}$$

Expert data:

$$\mathcal{D} = \left\{ \Big((\alpha^{(k)}, {s'}^{(k)}, {q'}^{(k)}),\ (s^{(k)}, q^{(k)}) \Big) \mid k = 1, \ldots, N \right\}$$

Data structures

Memoryless policy:

$$\mu_t^\star(s) \in \underset{\{\alpha | s \xrightarrow{\alpha} s'\}}{\operatorname{argmin}} \{\ell(s, \alpha, s') + V_{t+1}^\star(s')\}$$

optimizing process

Mode-varying policy:

$$\mu^\star(s, q) \in \underset{\{\alpha | s \xrightarrow{\alpha} s', q' = \delta(q, L(s))\}}{\operatorname{argmin}} \{\ell(s, \alpha, s') + V^\star(s', q')\}$$

Expert data:

$$\mathcal{D} = \left\{ \left((\alpha^{(k)}, {s'}^{(k)}, {q'}^{(k)}),\ (s^{(k)}, q^{(k)}) \right) \mid k = 1, \ldots, N \right\}$$

Data structures

Memoryless policy:

$$\mu_t^\star(s) \in \underset{\{\alpha \mid s \xrightarrow{\alpha} s'\}}{\operatorname{argmin}} \left\{ \ell(s, \alpha, s') + V_{t+1}^\star(s') \right\}$$

optimizing process

learned by IOC

Mode-varying policy:

$$\mu^\star(s, q) \in \underset{\{\alpha \mid s \xrightarrow{\alpha} s', q' = \delta(q, L(s))\}}{\operatorname{argmin}} \{ \ell(s, \alpha, s') + V^\star(s', q') \}$$

Expert data:

$$\mathcal{D} = \left\{ \Big((\alpha^{(k)}, {s'}^{(k)}, {q'}^{(k)}),\ (s^{(k)}, q^{(k)}) \Big) \mid k = 1, \ldots, N \right\}$$

Data structures

Memoryless policy:

$$\mu_t^\star(s) \in \operatorname*{argmin}_{\{\alpha \mid s \xrightarrow{\alpha} s'\}} \left\{ \ell(s,\alpha,s') + V_{t+1}^\star(s') \right\}$$

learned by IOC

optimizing process

Mode-varying policy:

$$\mu^\star(s,q) \in \operatorname*{argmin}_{\{\alpha \mid s \xrightarrow{\alpha} s',\, q'=\delta(q,L(s))\}} \left\{ \ell(s,\alpha,s') + V^\star(s',q') \right\}$$

Expert data:

$$\mathcal{D} = \left\{ \Big((\alpha^{(k)}, {s'}^{(k)}, {q'}^{(k)}),\ (s^{(k)}, q^{(k)}) \Big) \mid k = 1, \ldots, N \right\}$$

$x^{(k)}$

Data structures

Memoryless policy:

$$\mu_t^\star(s) \in \operatorname*{argmin}_{\{\alpha \mid s \xrightarrow{\alpha} s'\}} \left\{\ell(s,\alpha,s') + V_{t+1}^\star(s')\right\}$$

optimizing process

learned by IOC

Mode-varying policy:

$$\mu^\star(s,q) \in \operatorname*{argmin}_{\{\alpha \mid s \xrightarrow{\alpha} s',\, q'=\delta(q,L(s))\}} \left\{\ell(s,\alpha,s') + V^\star(s',q')\right\}$$

Expert data:

$$\mathcal{D} = \left\{ \Big((\alpha^{(k)}, {s'}^{(k)}, {q'}^{(k)}),\ (s^{(k)}, q^{(k)}) \Big) \mid k = 1,\ldots,N \right\}$$

$x^{(k)}$ $p^{(k)}$

Data structures

Memoryless policy:

$$\mu_t^\star(s) \in \operatorname*{argmin}_{\{\alpha | s \xrightarrow{\alpha} s'\}} \{\ell(s,\alpha,s') + V_{t+1}^\star(s')\}$$

optimizing process

learned by IOC

Mode-varying policy:

$$\mu^\star(s,q) \in \operatorname*{argmin}_{\{\alpha | s \xrightarrow{\alpha} s', q' = \delta(q, L(s))\}} \{\ell(s,\alpha,s') + V^\star(s',q')\}$$

Expert data:

$$\mathcal{D} = \left\{ \left((\alpha^{(k)}, s'^{(k)}, q'^{(k)}),\ (s^{(k)}, q^{(k)}) \right) \mid k = 1, \ldots, N \right\}$$

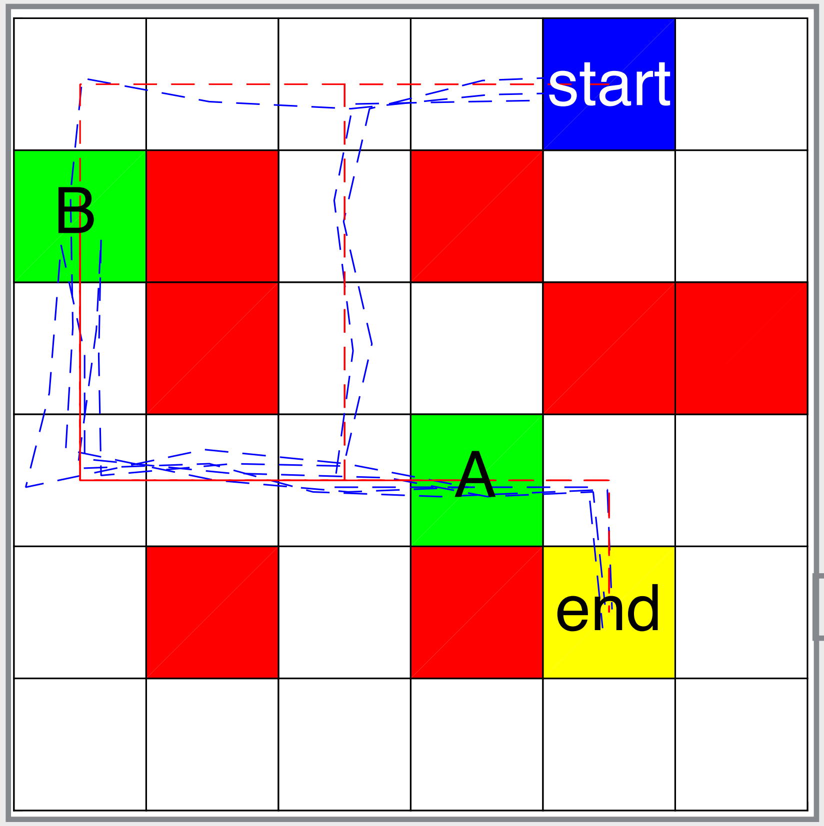
$x^{(k)}$ $p^{(k)}$

compare to memoryless case:

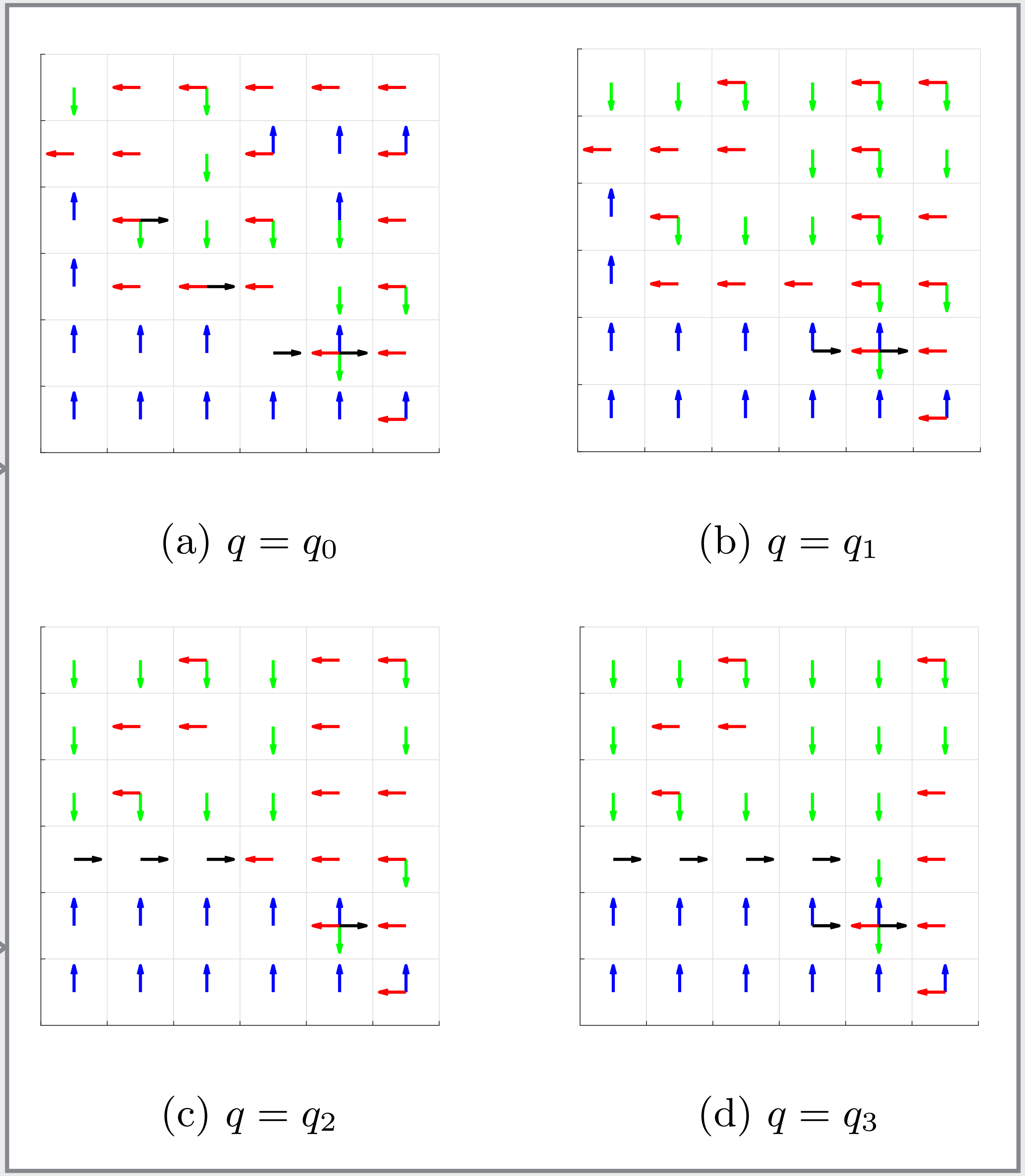
$$\mathcal{D} = \left\{ \left((\alpha^{(k)}, s'^{(k)}),\ s^{(k)} \right) \mid k = 1, \ldots, N \right\}$$

Side information-aware policy has memory

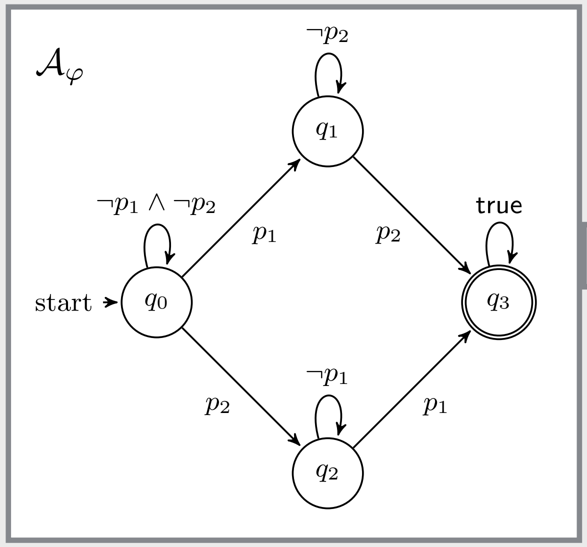
Expert demonstrations:



Learned policy:

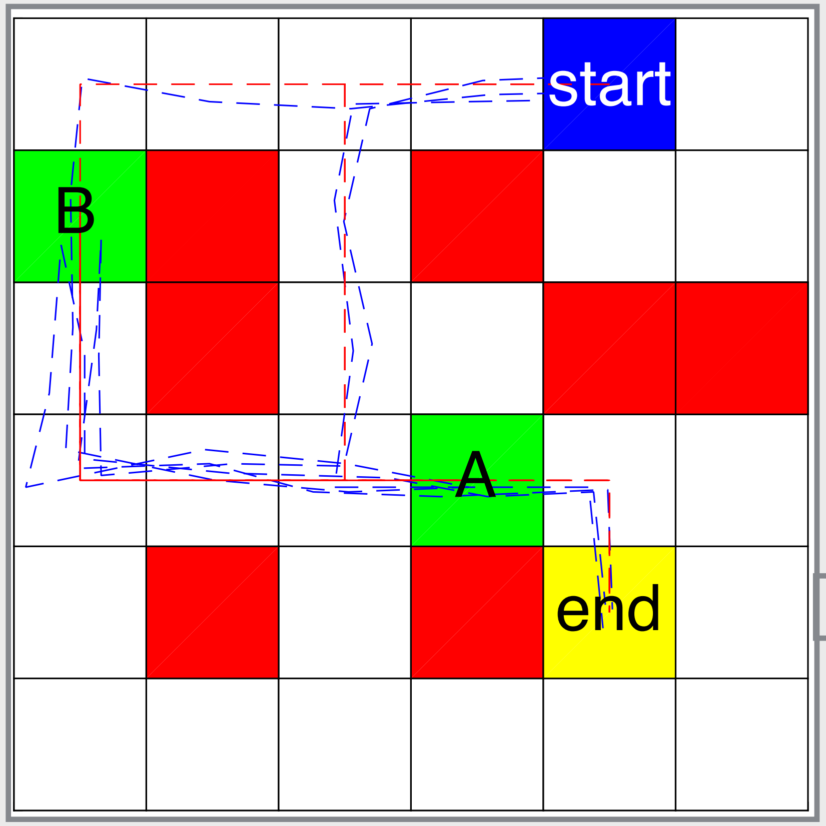


Side information:

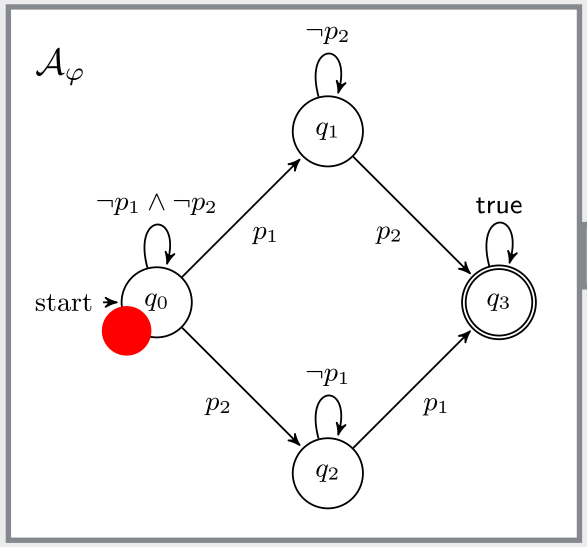


Side information-aware policy has memory

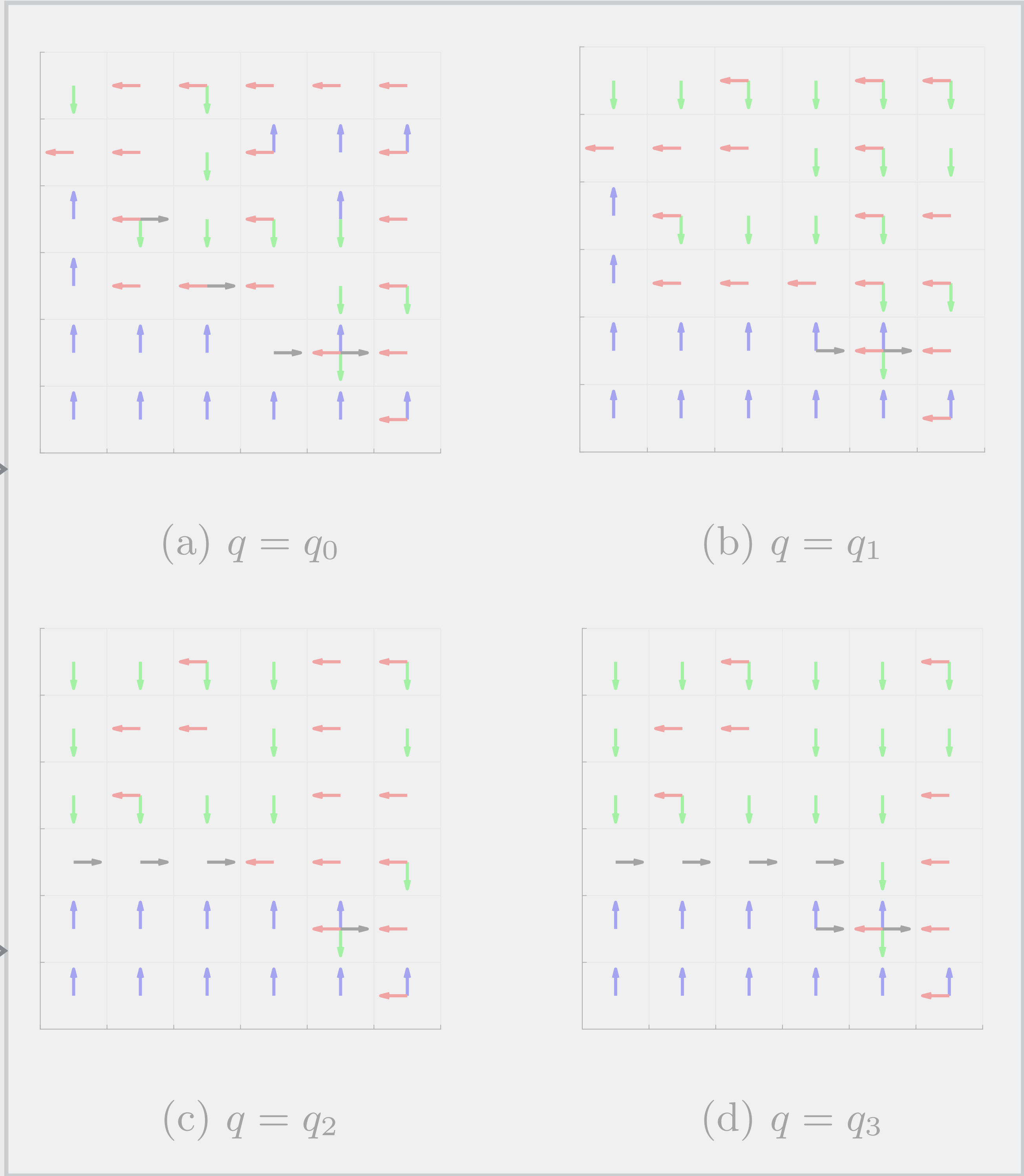
Expert demonstrations:



Side information:

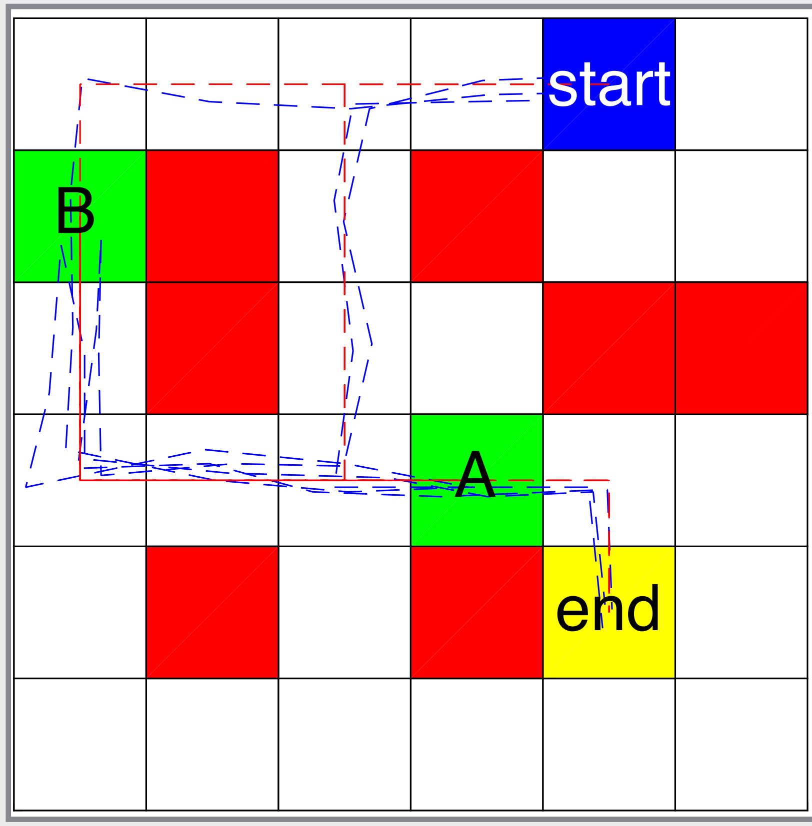


Learned policy:

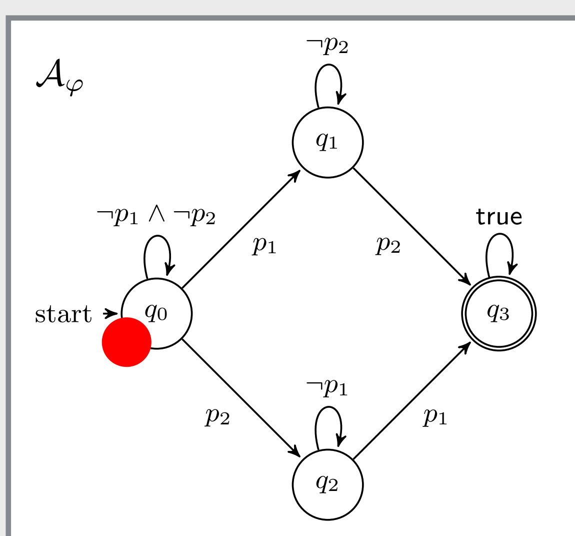


Side information-aware policy has memory

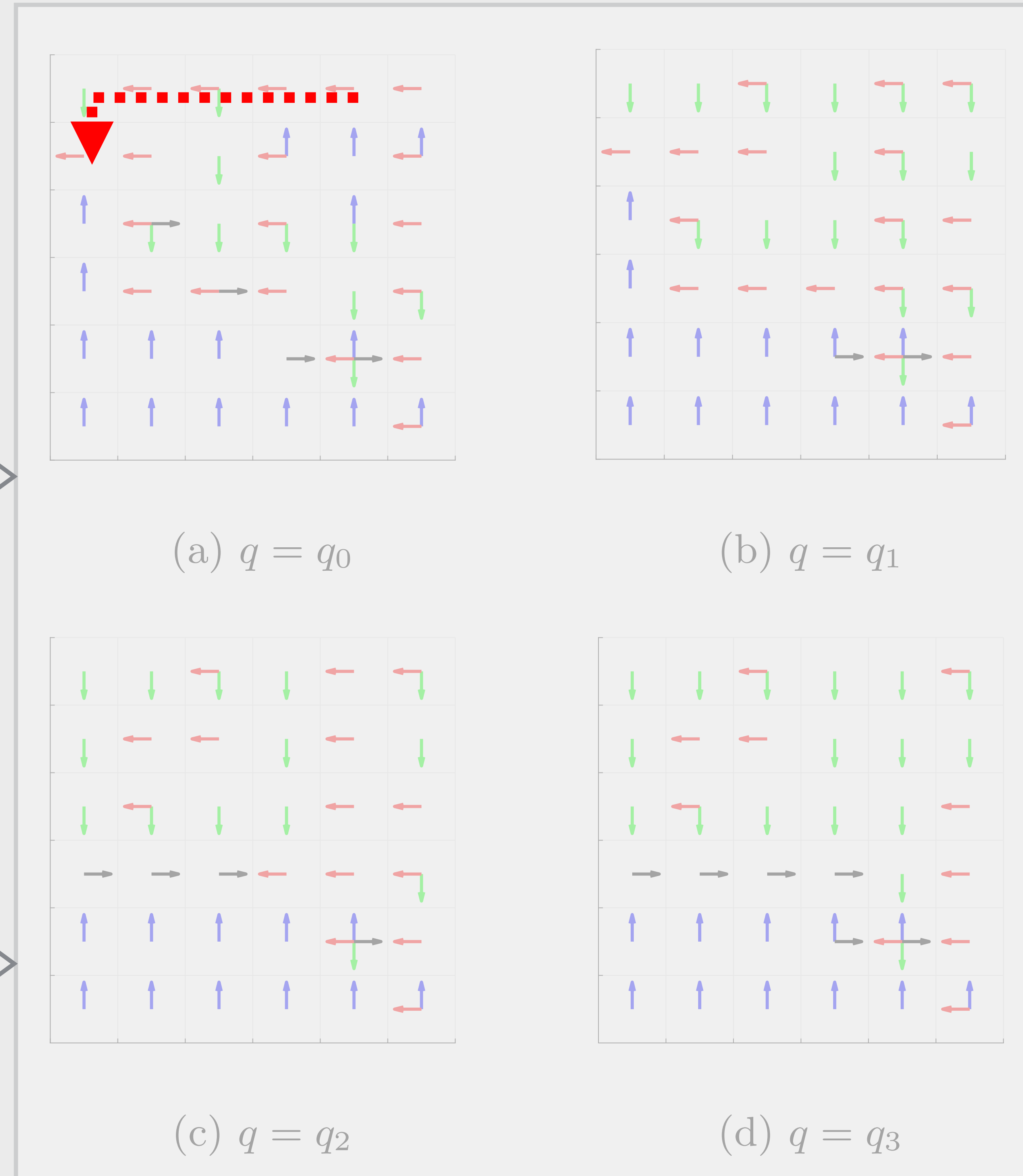
Expert demonstrations:



Side information:

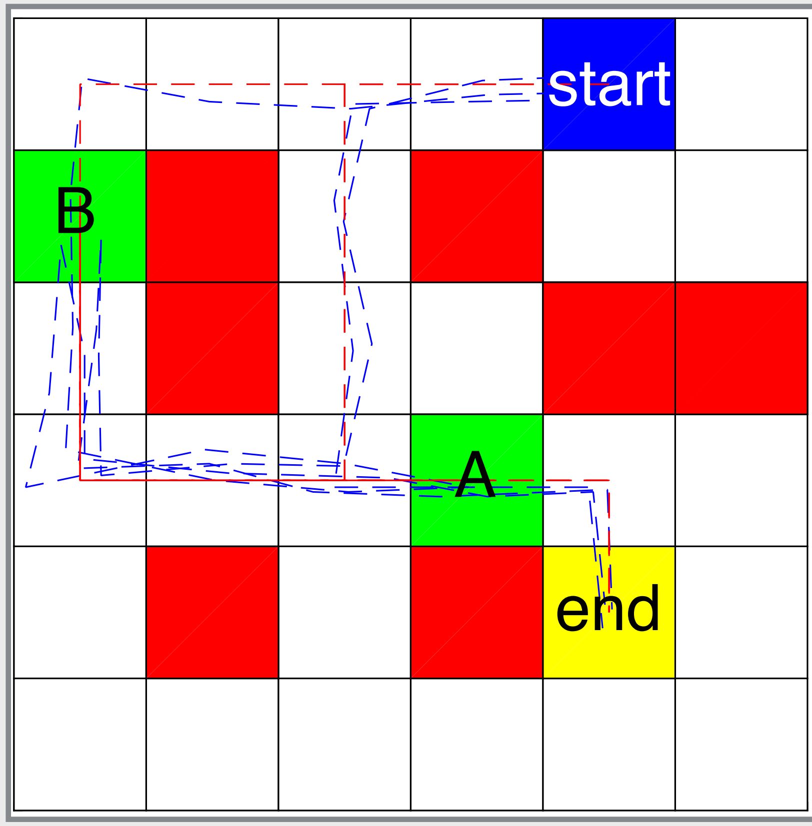


Learned policy:

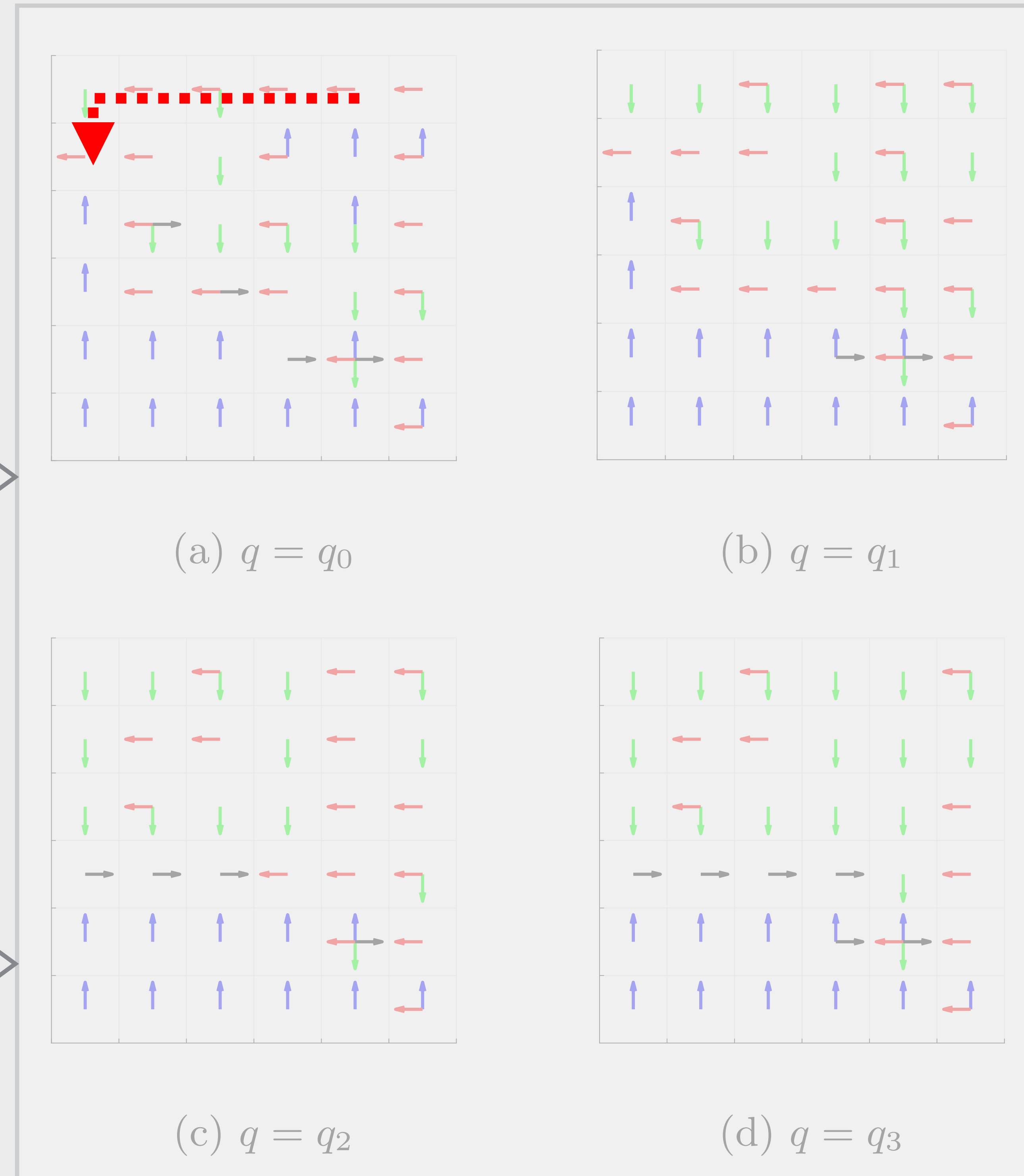


Side information-aware policy has memory

Expert demonstrations:



Learned policy:



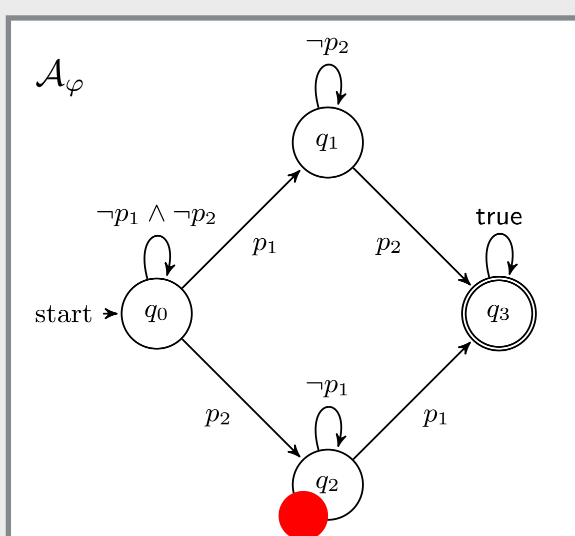
(a) $q = q_0$

(b) $q = q_1$

(c) $q = q_2$

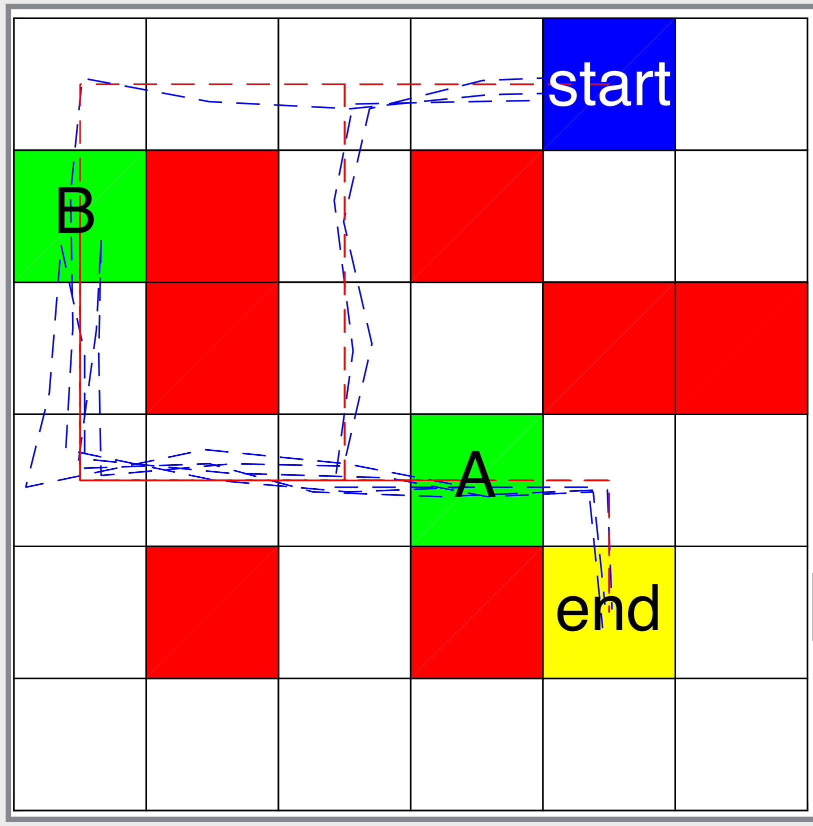
(d) $q = q_3$

Side information:

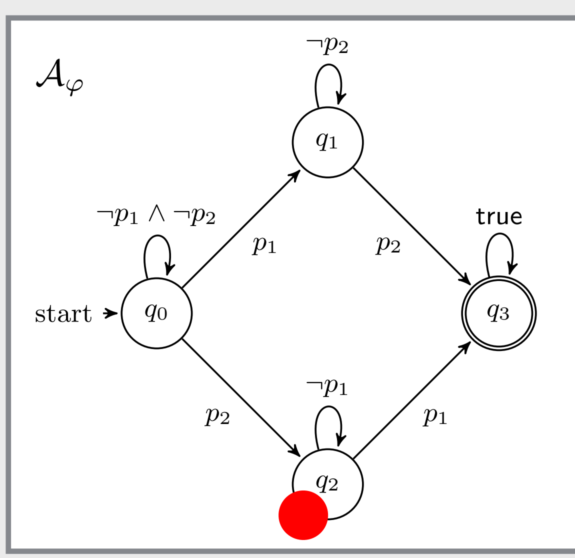


Side information-aware policy has memory

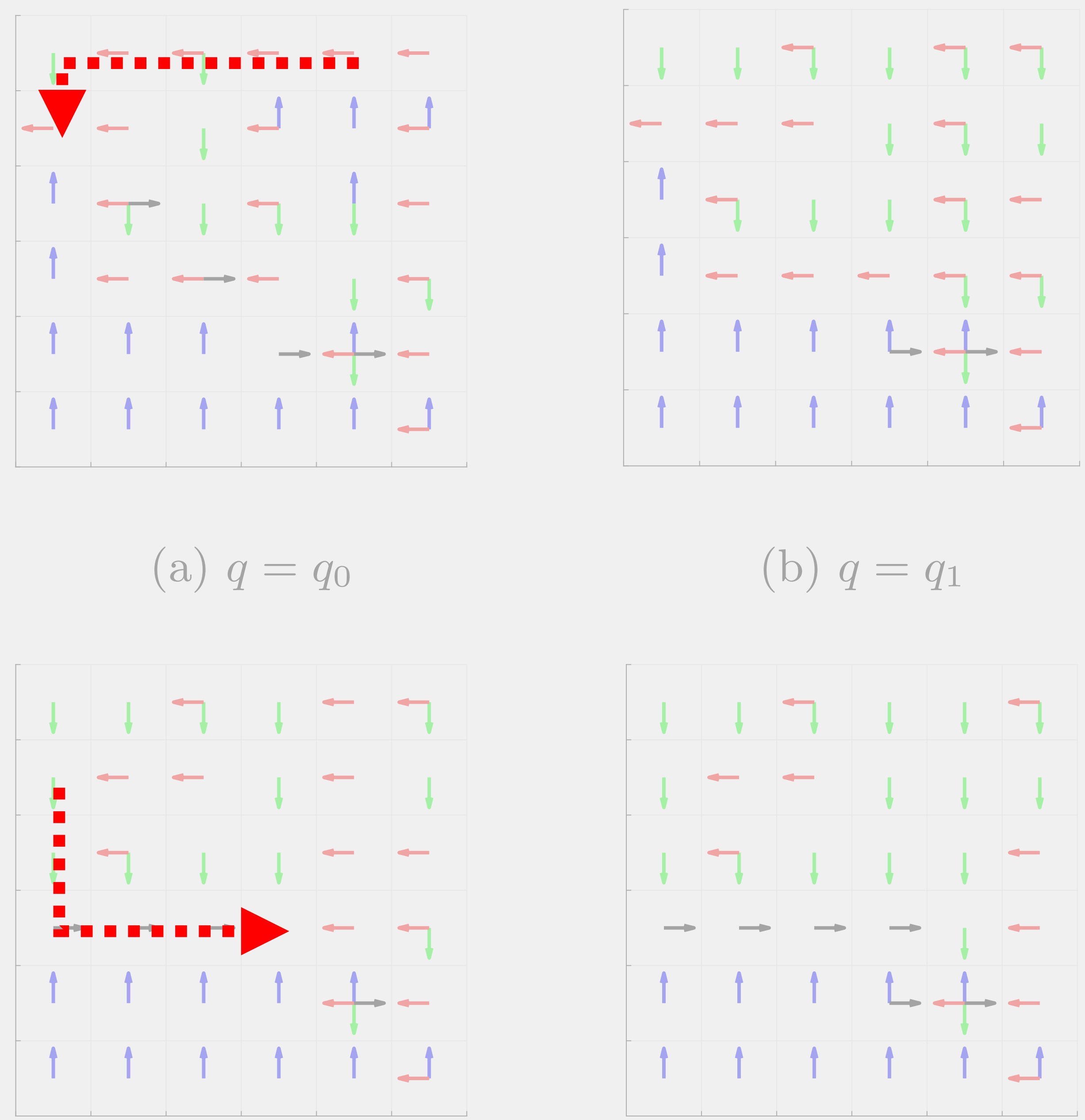
Expert demonstrations:



Side information:



Learned policy:

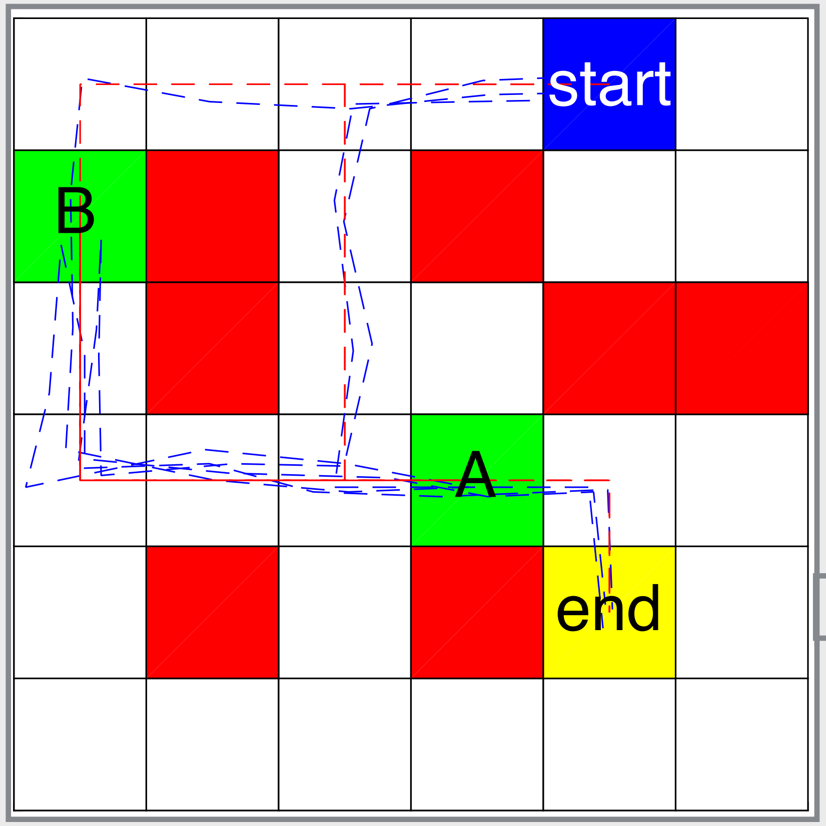


(a) $q = q_0$ (b) $q = q_1$

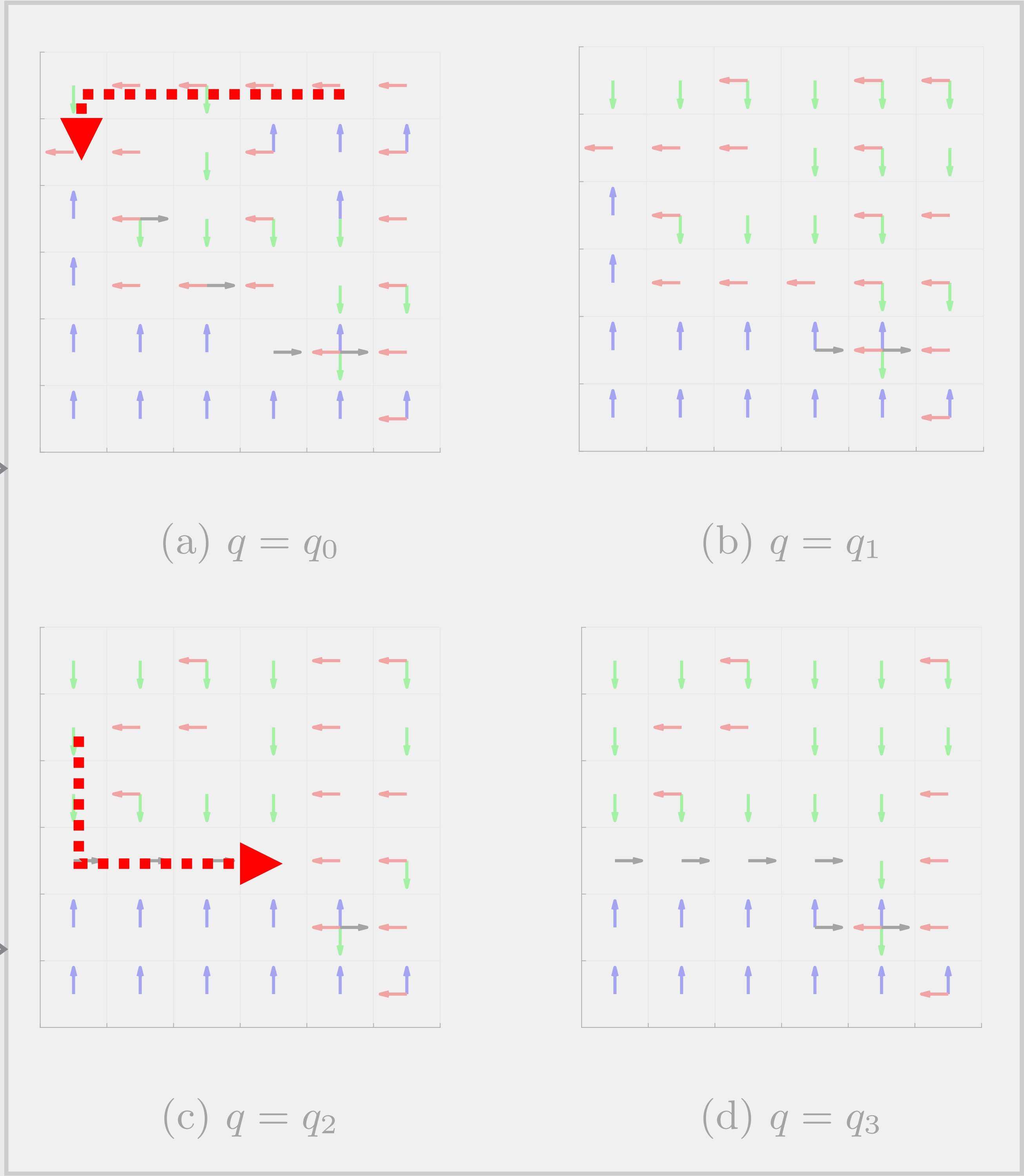
(c) $q = q_2$ (d) $q = q_3$

Side information-aware policy has memory

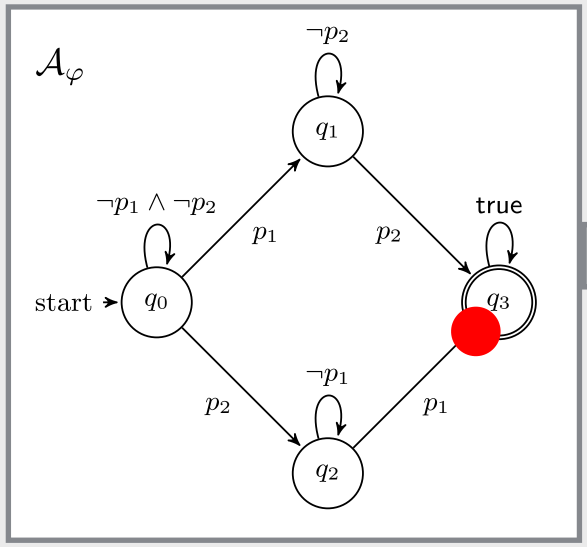
Expert demonstrations:



Learned policy:

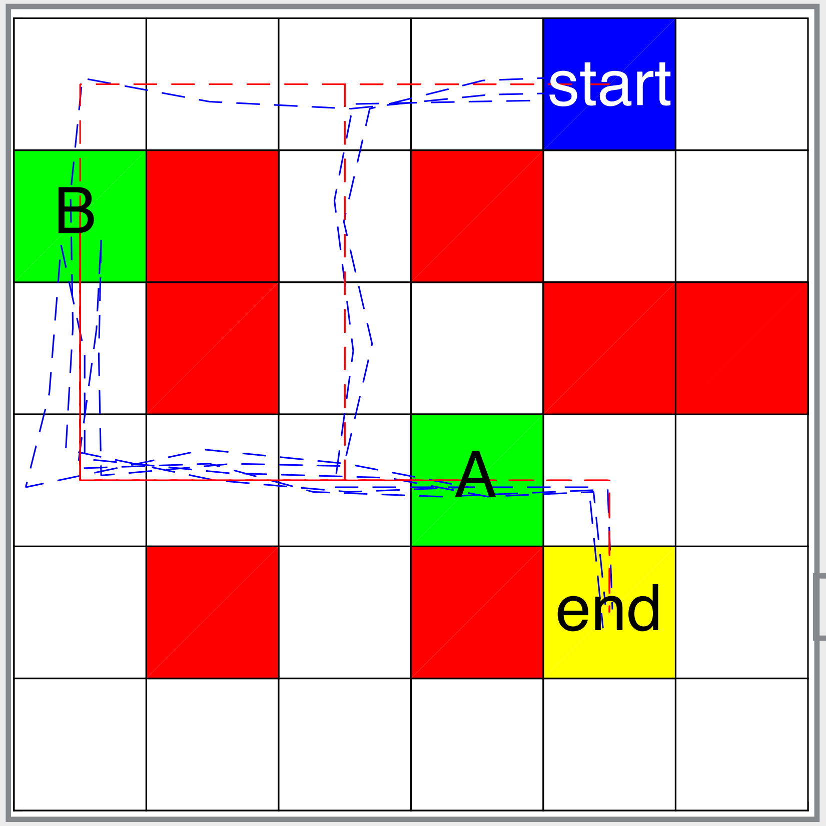


Side information:

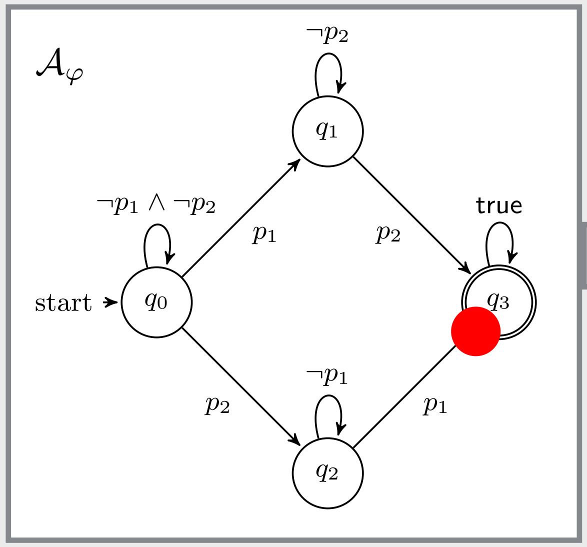


Side information-aware policy has memory

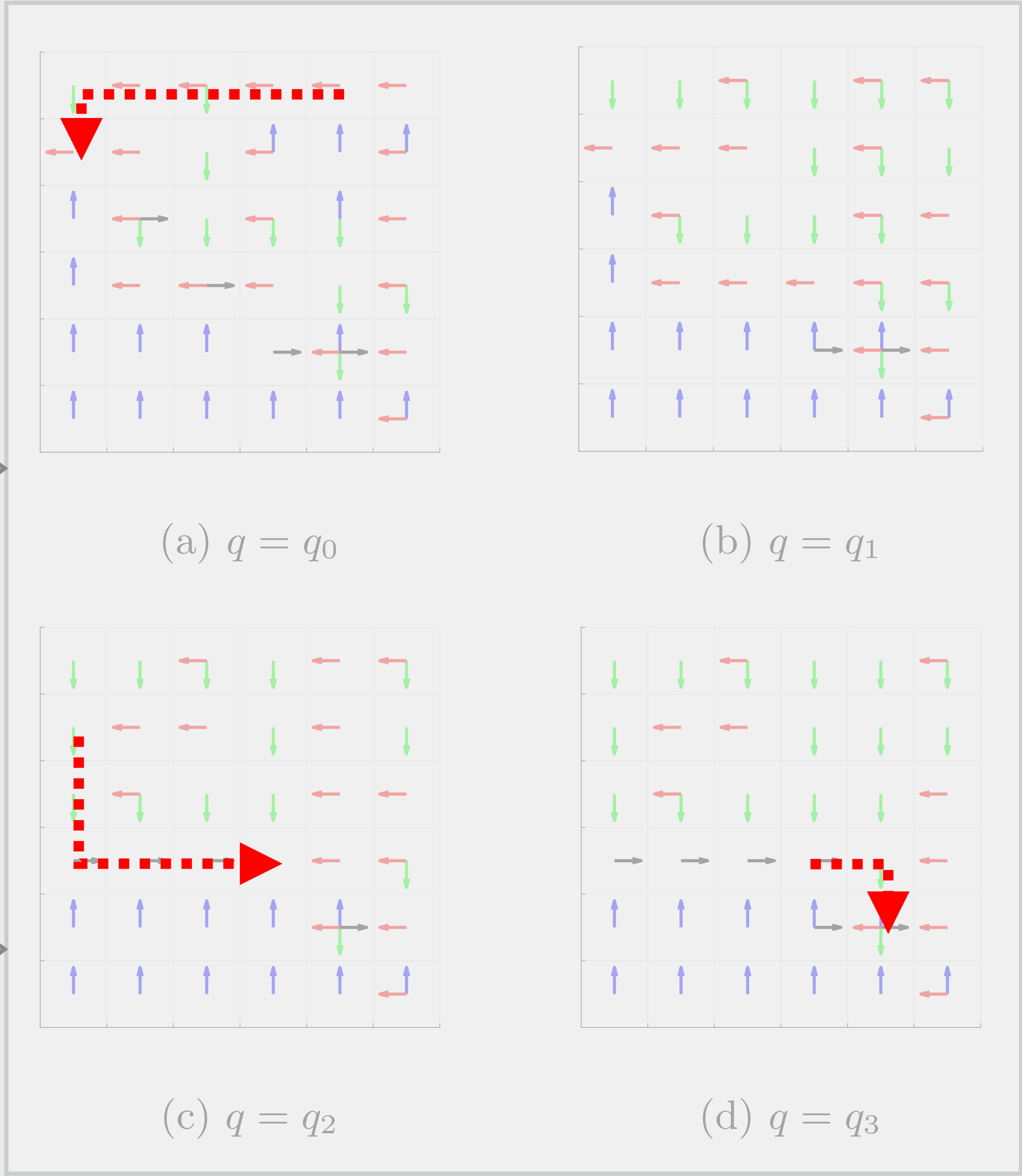
Expert demonstrations:



Side information:



Learned policy:



Summary

direct extensions

- Extend to broader dynamics classes—hybrid, nonlinear…
- Expand the family of specifications and languages
- Investigate the role of stochastic policies and partially specified side information
- Demonstrate scalability

new opportunities

- Open up a broad set of new problems to ideas from control and optimization

CDC 2016

Automata Theory Meets Approximate Dynamic Programming: Optimal Control with Temporal Logic Constraints

Ivan Papusha† Jie Fu* Ufuk Topcu‡ Richard M. Murray†

Learning from Demonstrations with High-Level Side Information

Min Wen* Ivan Papusha† Ufuk Topcu†
*University of Pennsylvania
†University of Texas at Austin

https://github.com/u-t-autonomous/sydar

SYDAR: Synthesis Done Approximately Right

Installation and usage

```
$ pip install sydar
$ sydar-matlab [input_file.miu] -o output.m
```

Acknowledgments

- Jie Fu (WPI)
- Min Wen (UPenn)
- Ufuk Topcu (UTexas)
- Richard Murray (Caltech)

WPI

Penn
UNIVERSITY of PENNSYLVANIA

TEXAS
The University of Texas at Austin

Caltech