Robust Control for Uncertain MDPs with Temporal Logic Specifications

Eric Wolff[1], Ufuk Topcu[2], and Richard Murray[1]

CDC 2012

12/11/12

[1]Caltech + [2]UPenn

Outline

- Motivation
- System: uncertain MDP
- Tasks: temporal logic
- Problem statement
- Solution
  - Combine system + task
  - Robust dynamic programming
- Example

Motivation

- **Goal**: Naturally specify tasks for autonomous systems

- Reality enters:
  - Autonomous systems must deal with uncertainty
  - System models are not perfect

Our Contribution

- Generalize previous results
  - MDPs [de Alfaro, Ding + Belta]
  - Interval MDPs [Chatterjee]
  - Robust dynamic programming [Nilim + El Ghaoui]
- Robustness almost for free
  - $O(\log(1/\varepsilon))$ times more effort

Specification language (LTL)

Want to specify properties such as:

- Response: always SIGNAL after a REQUEST arrives
- Liveness: always eventually PICKUP
- Safety: always remain SAFE
- Priority: do JOB1 until JOB2
- Guarantee: eventually reach GOAL

Linear temporal logic (LTL):

- A logic for reasoning about how properties change over time
- Reason about infinite sequences $\sigma = s_0 s_1 s_2 \ldots$ of states
- Propositional logic: $\wedge$ (and), $\vee$ (or), $\implies$ (implies), $\neg$ (not)
- Temporal operators: $\mathcal{U}$ (until), $\bigcirc$ (next), $\square$ (always), $\diamond$ (eventually)

Specification language (LTL)

Want to specify properties such as:

- Response: $\Box$(REQUEST $\implies$ SIGNAL)
- Liveness: $\Box\Diamond$ PICKUP
- Safety: $\Box$ SAFE
- Priority: JOB1 $\mathcal{U}$ JOB2
- Guarantee: $\Diamond$ GOAL

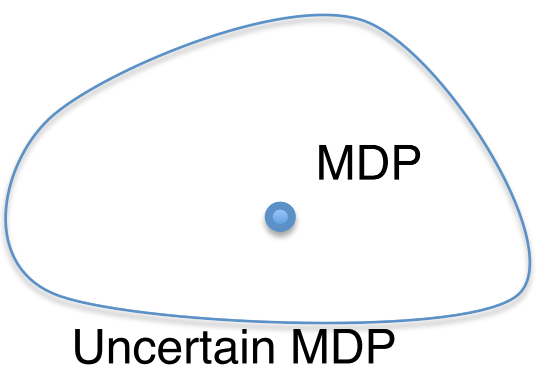
Linear temporal logic (LTL):

- A logic for reasoning about how properties change over time
- Reason about infinite sequences $\sigma = s_0 s_1 s_2 \ldots$ of states
- Propositional logic: $\wedge$ (and), $\vee$ (or), $\implies$ (implies), $\neg$ (not)
- Temporal operators: $\mathcal{U}$ (until), $\bigcirc$ (next), $\Box$ (always), $\Diamond$ (eventually)

System model (uncertain MDP)

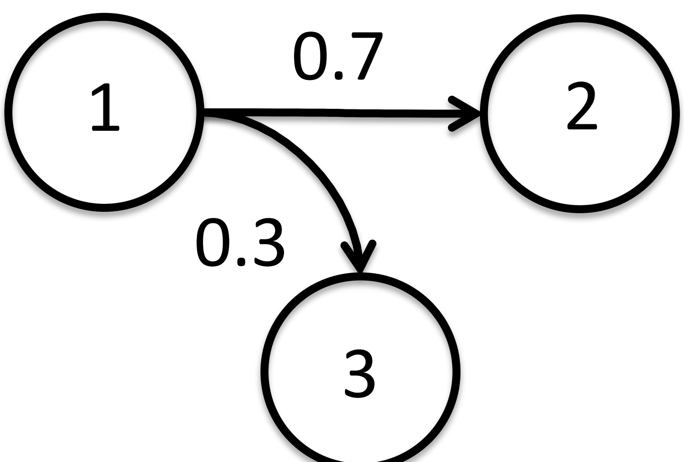
- An **MDP** M is a tuple M = (S, A, P, s_0, AP, L), where
  - S is a finite set of states,
  - A is a finite set of actions (e.g., motion primitives),
  - $P : S \times A \times S \rightarrow [0,1]$ is the transition probability function,
  - s_0 is the initial state,
  - AP is a finite set of atomic propositions, and
  - $L : S \rightarrow 2^{AP}$ is a labeling function.

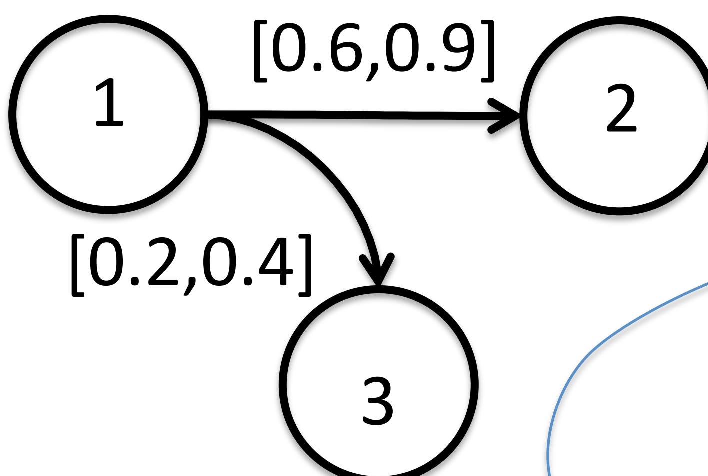
- **Control policy:**
  - $\pi: S \rightarrow A$
  - Induces Markov chain

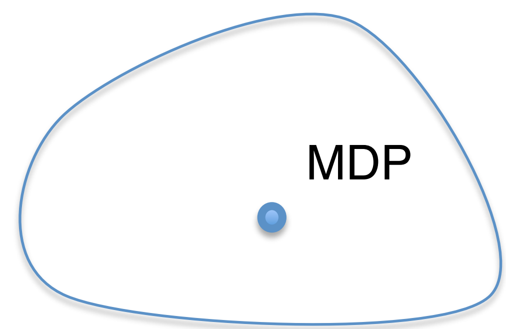


System model (uncertain MDP)

- Uncertainty **set** for MDP transitions (likelihood, entropy, MAP, interval, scenario, …)
- Control picks action, environment picks transition
- Nominal   Uncertain







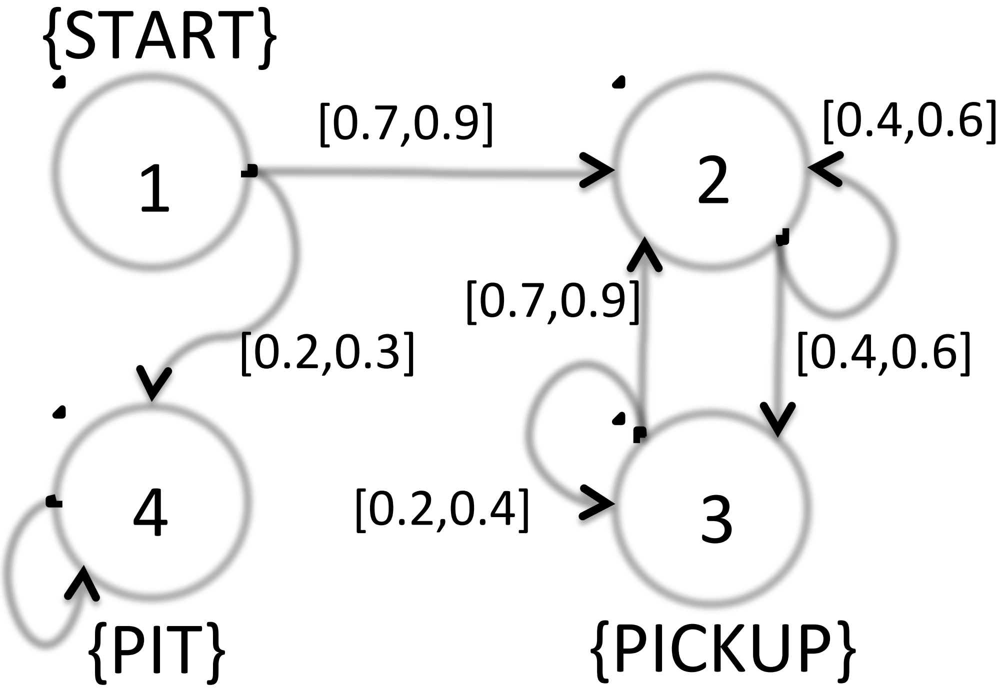
Uncertain MDP

Problem statement

- **Given**:
  - Uncertain MDP M w/ initial state s_0
  - LTL specification φ
- **Problem**: Create control policy π* that maximizes the probability of MDP M satisfying φ over uncertainty set, i.e.

$$\pi^* = \arg\max_{\substack{\pi\in\Pi \\ \uparrow \\ \text{System policies}}} \min_{\substack{\tau\in\mathcal{T} \\ \uparrow \\ \text{Env. policies (uncertainty)}}} \mathbb{P}^{\pi,\tau}(s_0 \vDash \varphi)$$

Tutorial example



Task: Repeatedly PICKUP and always avoid PIT

Solution overview

1. LTL spec φ → deterministic Rabin automaton A_φ

Solution overview

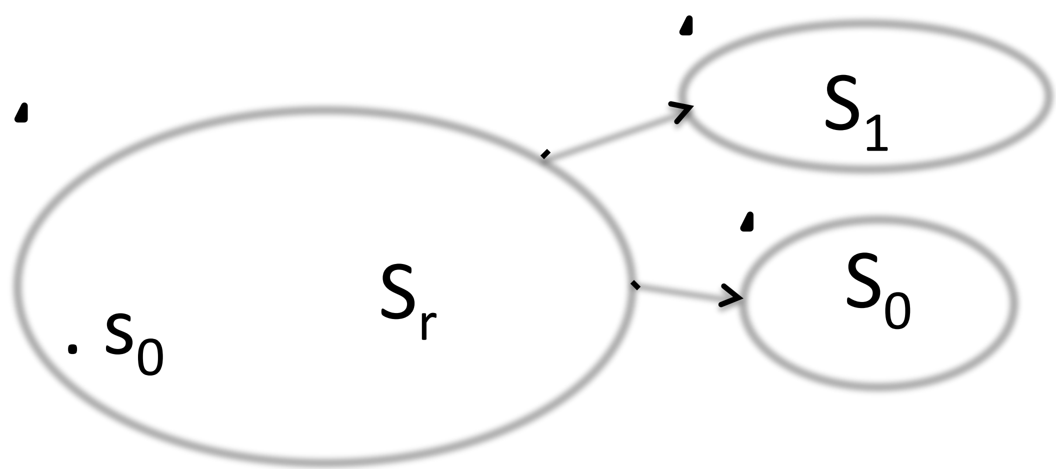
1. LTL spec φ $\rightarrow$ deterministic Rabin automaton A_φ
2. Create product MDP $M_p = M \times A_\varphi$

Solution overview

1. LTL spec φ → deterministic Rabin automaton A_φ
2. Create product MDP $M_p = M \times A_\varphi$
3. Compute winning set in M_p

Solution overview

1. LTL spec φ → deterministic Rabin automaton A_φ
2. Create product MDP $M_p = M \times A_\varphi$
3. Compute winning set in M_p
4. Compute control policy to maximize probability of reaching winning set (dynamic programming)



Solution overview

1. LTL spec $\varphi \rightarrow$ deterministic Rabin automaton A_φ
2. Create product MDP $M_p = M \times A_\varphi$
3. Compute winning set in M_p
4. Compute control policy to maximize probability of reaching winning set (dynamic programming)
5. Project policy back to the original MDP

Solution overview

1. LTL spec φ → deterministic Rabin automaton A_φ
2. Create product MDP $M_p = M \times A_\varphi$
3. Compute winning set in M_p
4. Compute control policy to maximize probability of reaching winning set (dynamic programming)
5. Project policy back to the original MDP

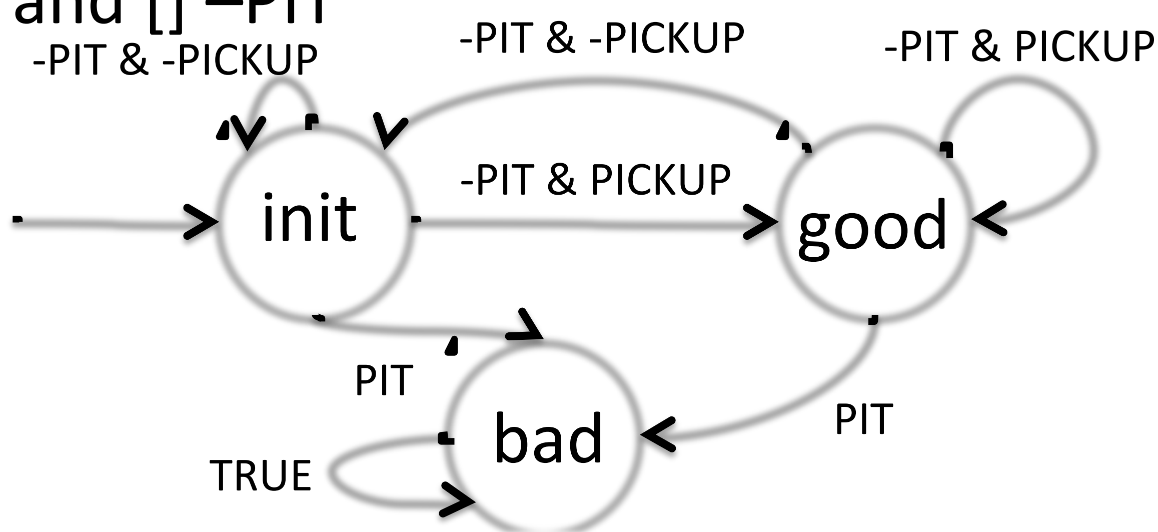
- **Our focus: Step 4**

LTL spec to automaton (1/5)

- Spec satisfaction?
  - Infinitely often visit “good” states
  - Finitely often visit “bad” states

LTL spec to automaton (1/5)

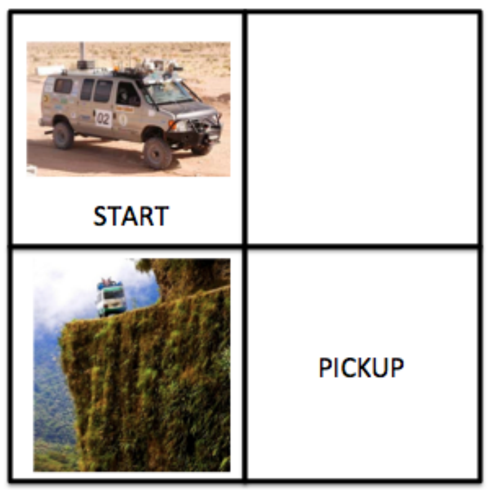
- Spec satisfaction?
  - Infinitely often visit "good" states
  - Finitely often visit "bad" states
- Ex:
  - Task: Repeatedly PICKUP and always avoid PIT
  - φ = []<> PICKUP and [] –PIT



Product automaton (2/5)

- M_p has behaviors that satisfy system and spec

$M_p =$

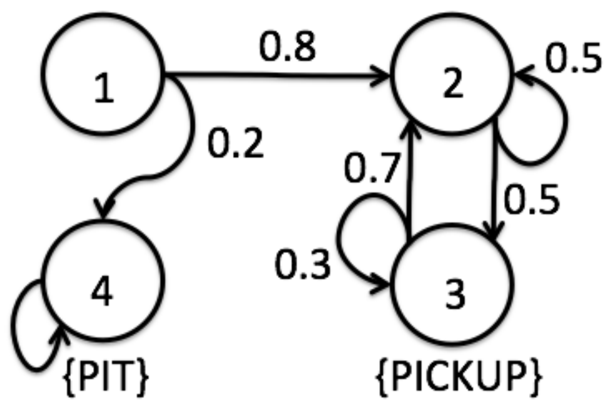


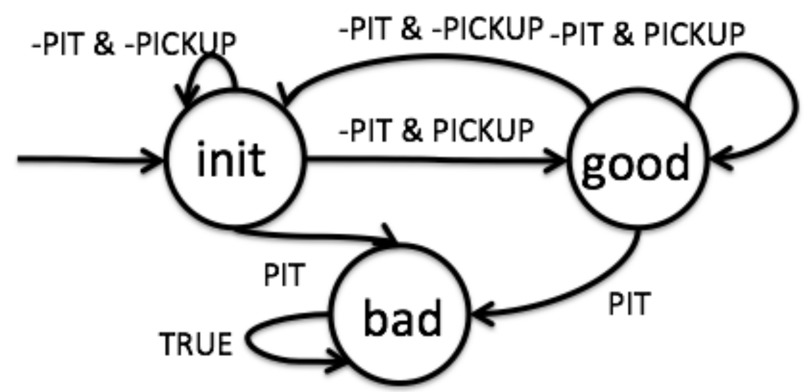
X

Task: Repeatedly PICKUP and always avoid PIT

Product automaton (2/5)

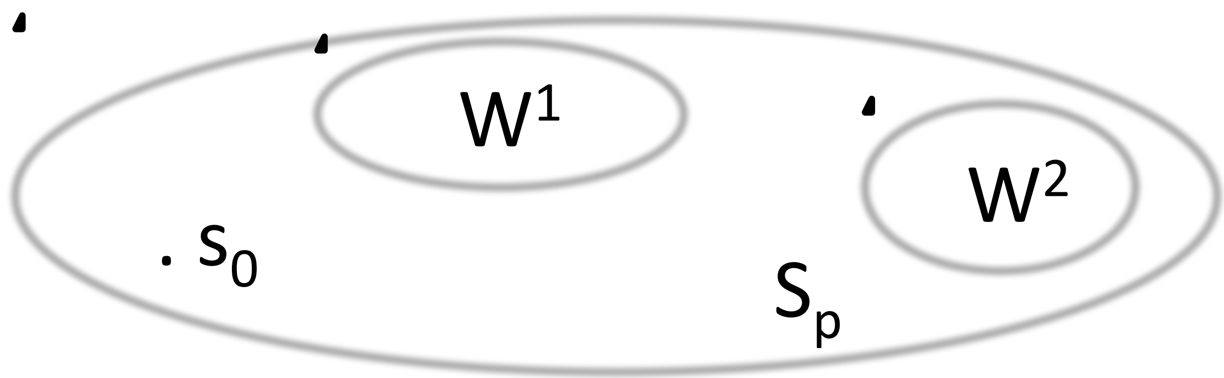
- M_p has behaviors that satisfy system and spec

$M_p =$ 

X 

Winning sets (3/5)

- Winning set:
  - System can stay in set of states forever
  - Includes “good” states
  - Excludes “bad” states

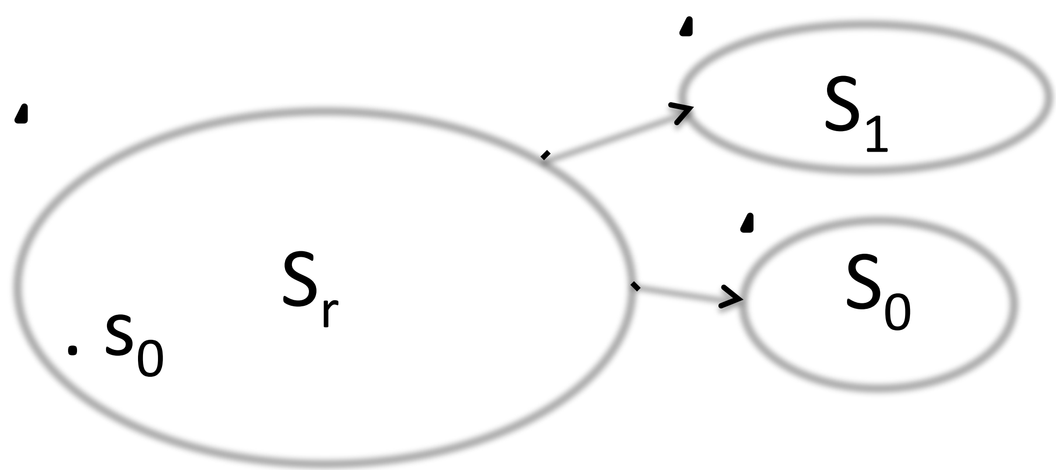


- Problem is now to reach union of these sets

Reachability problem (4/5)

- $V(s)$ is probability of satisfying spec at state s

- $V(s) = 1$ for s in winning sets ($S_1 = W^1 \cup W^2$)
- $V(s) = 0$ for s that cannot reach winning set (S_0)
- $V(s) = ???$ for s in $S_r = S - (S_0 \cup S_1)$



Robust dynamic programming (4/5)

- Undiscounted problem [compare w/ Nilim + El Ghaoui]

- Informally:
  - V maps each state to a scalar (spec. satisfaction prob.)
  - p is probability distribution environment selects
  - A(s) is the set of control actions in state s
  - r(s,a) is a scalar reward

$$(TV)(s) \quad := \quad \max_{a \in A(s)} \left[r(s,a) + \min_{p \in \mathcal{P}_s^a} \; p^T V \right]$$

Robust dynamic programming (4/5)

- **Theorem**: T operator is a contraction
  - Based on transformation of product MDP
  - Problem specific insight
  - Weighted sup norm

- Use contraction mapping theorem for existence/uniqueness of $TV^* = V^*$ fixed-point
- **Value iteration** to compute V^*

Solution overview

1. LTL spec φ → deterministic Rabin automaton A_φ
2. Create product MDP $M_p = M \times A_\varphi$
3. Compute winning set in M_p
4. Compute control policy to maximize probability of reaching winning set (dynamic programming)
5. Project policy back to the original MDP

- **Our focus: Step 4**

Complexity

- Good?
  - n, m = # states, edges in product MDP
  - ε-suboptimal policy: $O(n^2 m \log(1/\varepsilon)$ $\log(1/\varepsilon)$) [likelihood]

$$(TV)(s) \quad := \quad \max_{a \in A(s)} \left[r(s,a) + \min_{p \in \mathcal{P}_s^a} \; p^T V \right]$$

Complexity

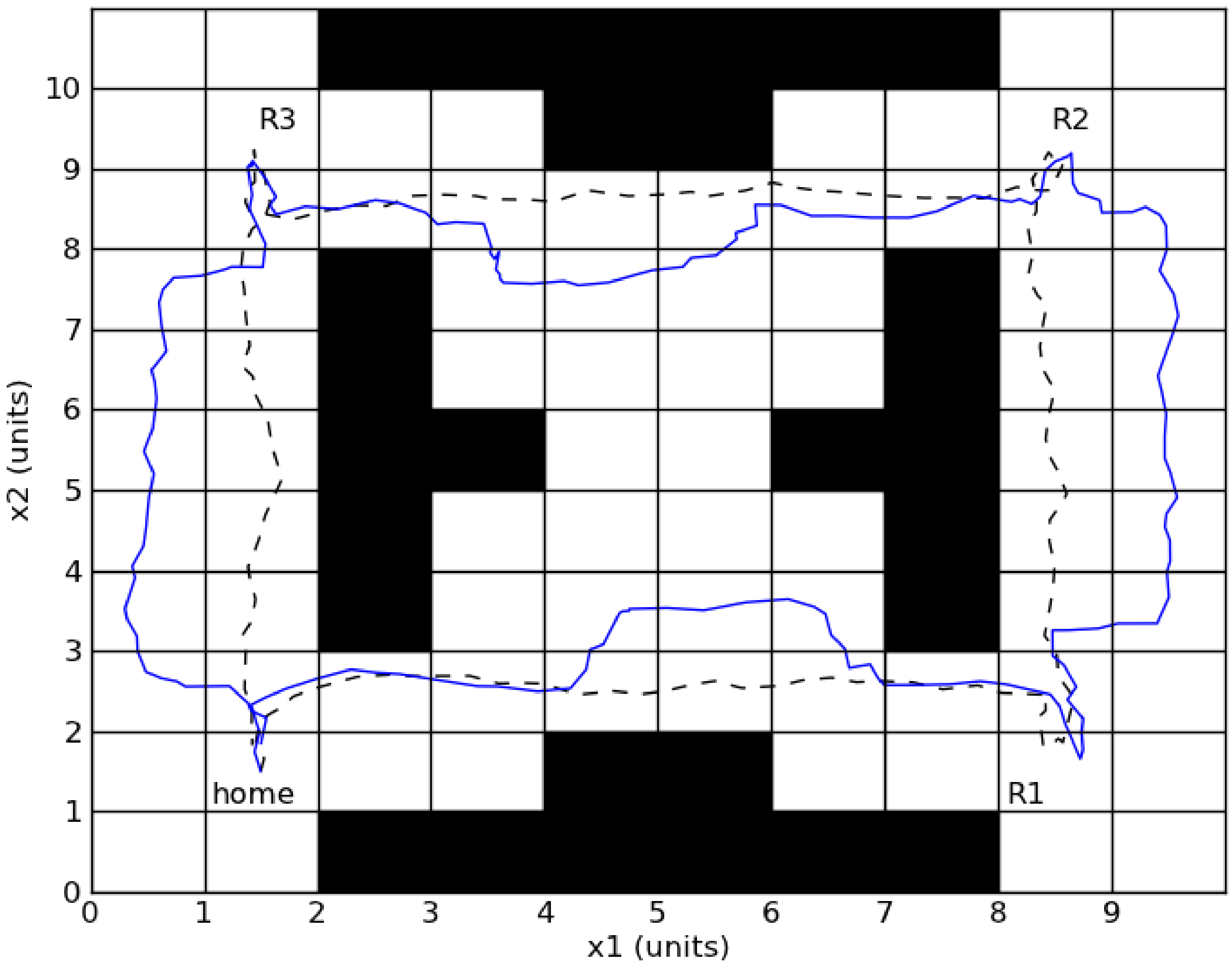
- Good?
  - n, m = # states, edges in product MDP
  - ε-suboptimal policy: $O(n^2 m \log(1/\varepsilon) \log(1/\varepsilon))$ [likelihood]
- Wait!
  - LTL to DRA: $O(2\text{-}\exp(|\varphi|))$
  - For LTL fragment: $O(\exp(|\varphi|))$ [Alur]
  - For other LTL fragment: N/A (!) [Wolff, ICRA13 sub.]

Complexity

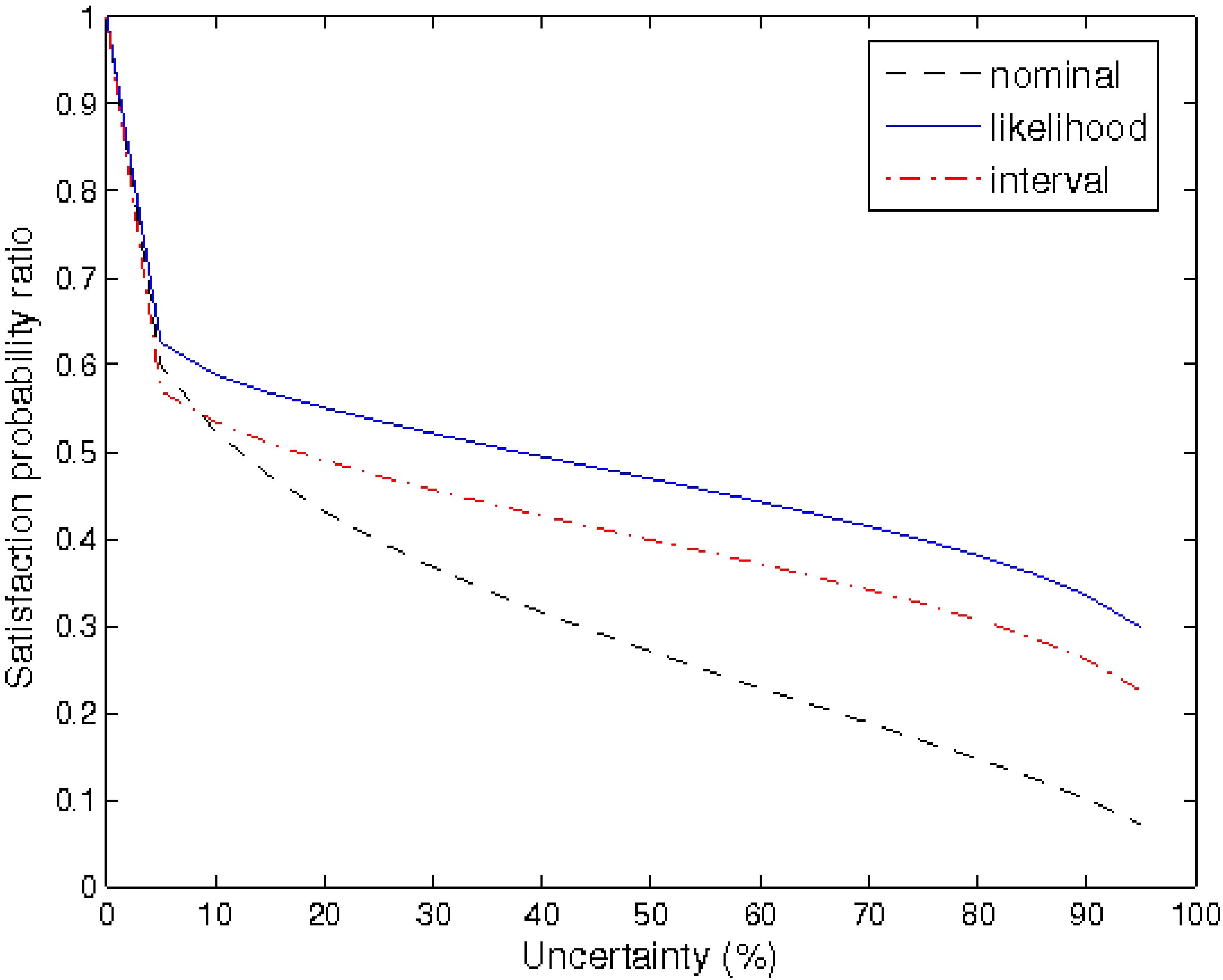
- Good?
  - n, m = # states, edges in product MDP
  - ε-suboptimal policy: $O(n^2 m \log(1/\varepsilon)$ $\log(1/\varepsilon)$) [likelihood]
- Wait!
  - LTL to DRA: O(2-exp(|φ|))
  - For LTL fragment: O(exp(|φ|)) [Alur]
  - For other LTL fragment: N/A (!) [Wolff, ICRA13 sub.]

- Takeaway: Robust policy in O(log(1/ε)) more time [Nilim + El Ghaoui results for likelihood uncert.]

Simulation Results



- **Informal task**: Start + end at HOME. Avoid OBSTACLES. Visit R1, R2, R3.
- **Sample trajectories**: nominal (0.47 sec) + robust (5.7 sec)


nominal
likelihood
interval
Satisfaction probability ratio
Uncertainty (%)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100

Conclusions

- Our approach:
  - Uncertainty sets for MDP transitions
  - LTL formulas describe complex tasks
  - Robustness almost free [O(log(1/ε)) more time]
- Current work:
  - Non-deterministic + stochastic environments
  - Multi-objective

Thanks!

- Questions?

- Funding
  - NSF graduate research fellowship
  - Boeing
  - AFOSR