Efficient Reactive Controller Synthesis for a Fragment of Temporal Logic

Eric M. Wolff[1]

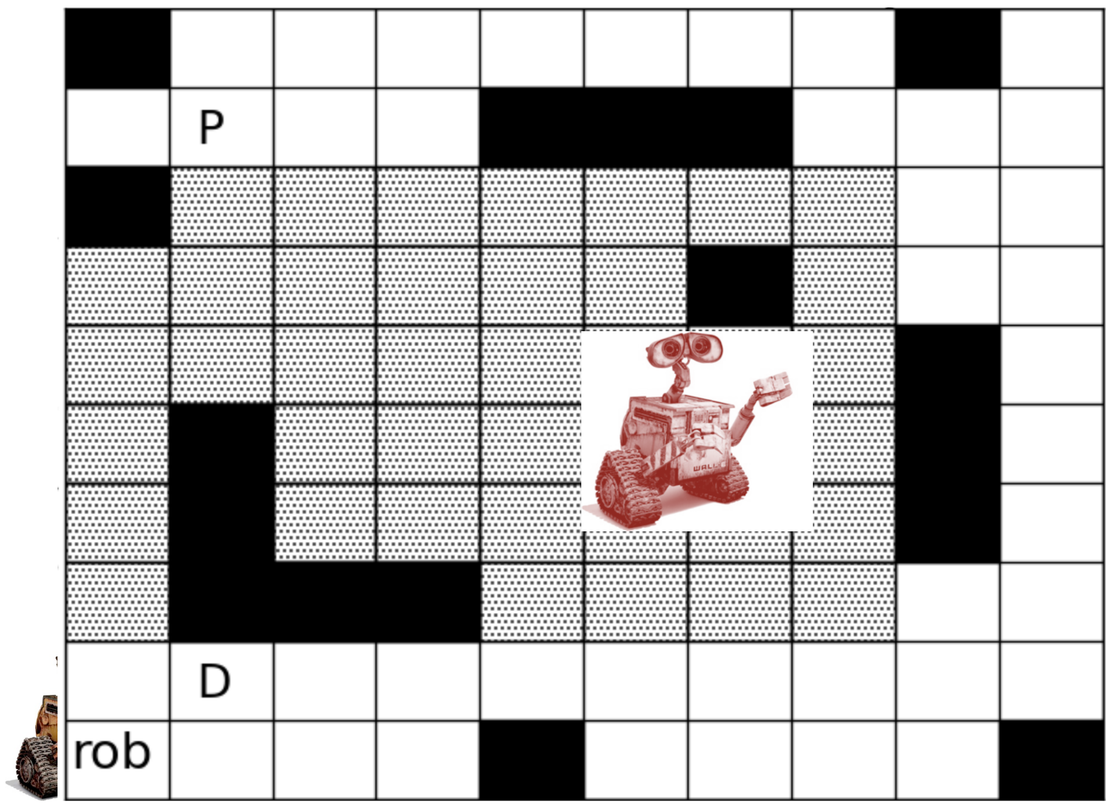
Ufuk Topcu[2] and Richard M. Murray[1]

[1]Caltech and [2]UPenn

ICRA 2013

Planning in a Dynamic Environment

- Dynamic obstacles
- Safe navigation and repetitive tasks



Our Contributions

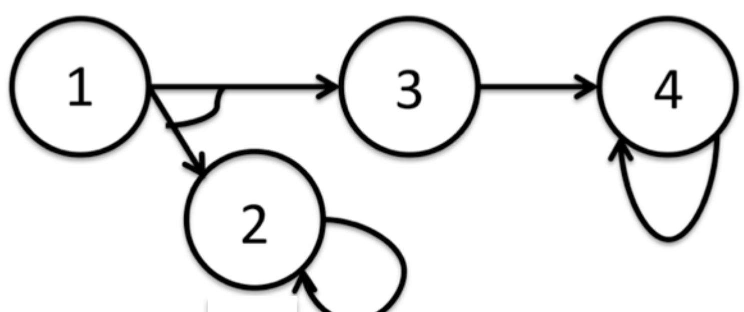
- Introduce expressive and efficient fragment of linear temporal logic
  - 2-EXP improvement over LTL
  - Non-deterministic and stochastic systems
  - Simple and extensible framework (e.g. optimality)

Outline

- Motivation
- Preliminaries
- Feasible control policies
- Examples

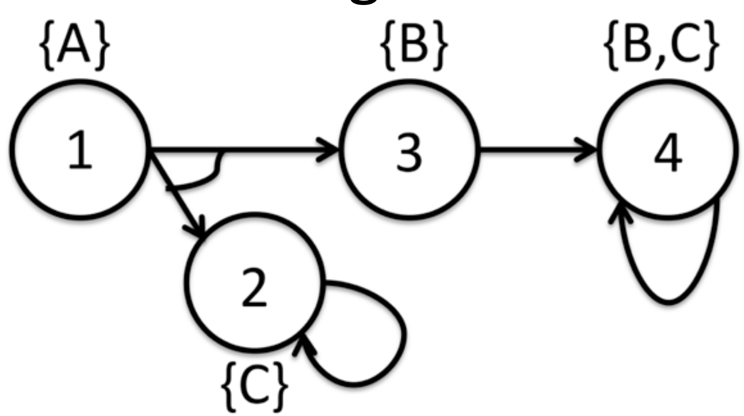
Non-deterministic Transition Systems

- A **non-deterministic transition system (NTS)** is a tuple $T = (S, A, R, s_0, AP, L)$ where
  - S is a finite set of **states**,
  - A is a finite set of **actions** (e.g., motion primitives),
  - $R: S \times A \rightarrow 2^S$ is the **transition** function,
  - s_0 is the initial state,



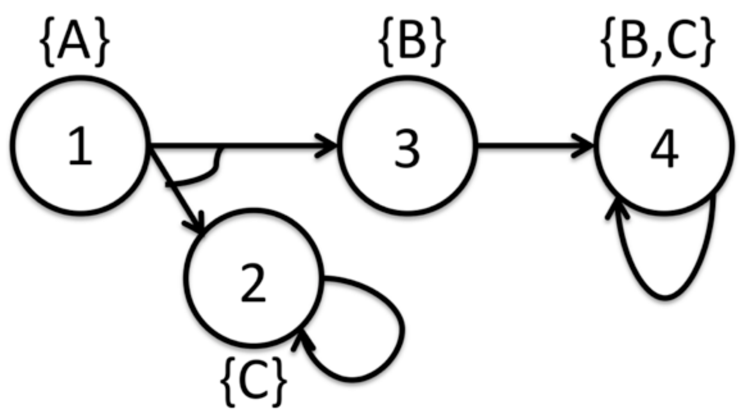
Non-deterministic Transition Systems

- A **non-deterministic transition system (NTS)** is a tuple $T = (S, A, R, s_0, AP, L)$ where
  - S is a finite set of **states**,
  - A is a finite set of **actions** (e.g., motion primitives),
  - $R: S \times A \rightarrow 2^S$ is the **transition** function,
  - s_0 is the initial state,
  - AP is a finite set of atomic propositions,
  - $L : S \rightarrow 2^{AP}$ is a labeling function



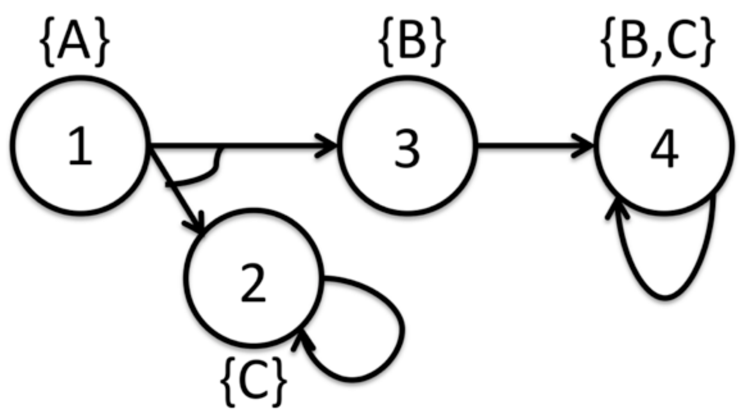
Control Policies

- **Control policy:**
  - μ: S → A (memoryless)
  - μ: S x M → A x M (finite-memory, M = memory set)



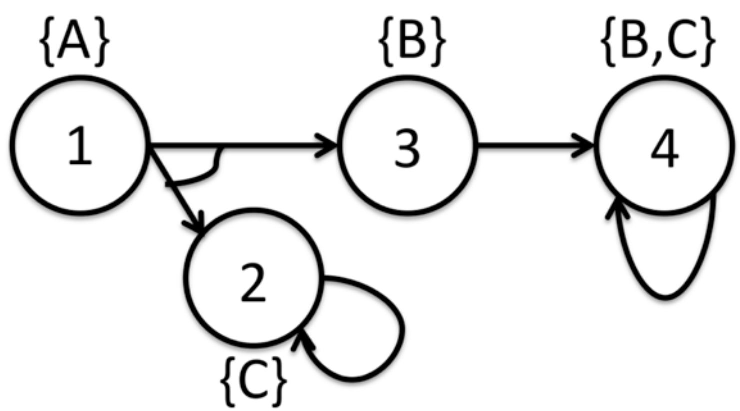
Control Policies

- **Control policy:**
  - μ: S → A (memoryless)
  - μ: S x M → A x M (finite-memory, M = memory set)
- **Two-player game**:
  - 1) **System** picks action using control policy
  - 2) **Environment** picks next state



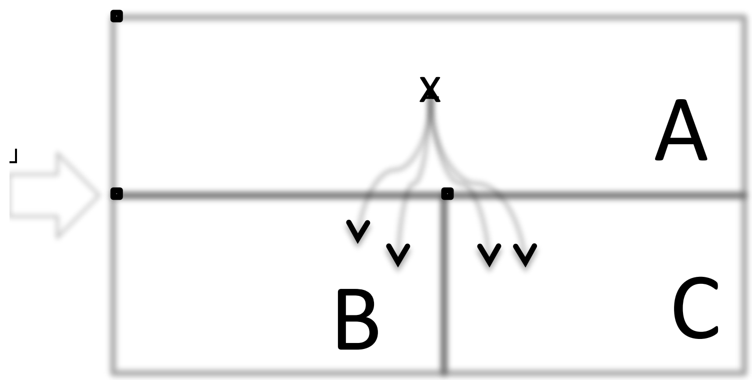
Control Policies

- **Control policy:**
  - $\mu: S \rightarrow A$ (memoryless)
  - $\mu: S \times M \rightarrow A \times M$ (finite-memory, M = memory set)
- **Two-player game**:
  - 1) **System** picks action using control policy
  - 2) **Environment** picks next state
- $T^{\mu}(s)$: set of runs from state s under policy μ

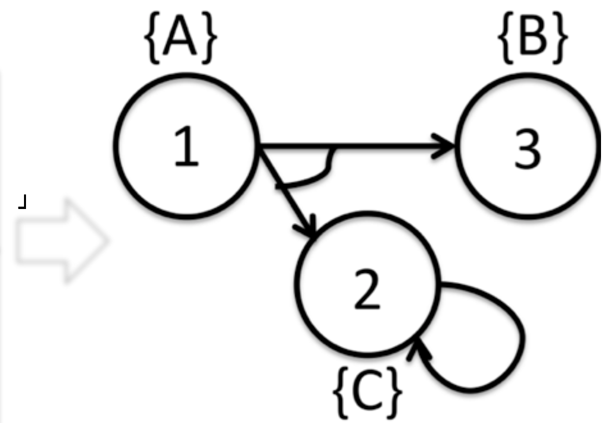


Hierarchical Control Architecture

Dynamical system



Discrete abstraction[1]



Non-deterministic transition system

- We focus on the discrete planning layer
- Discrete plan is executed at continuous level

1. AlurHLP00, BeltaH06, BhatiaLV10, HabetsCS06, KaramanF09, KloetzerB08, and more

Specification Language

- We consider formulas of the form:

$$\varphi = \varphi_{\text{safe}} \ \wedge \ \varphi_{\text{resp}} \ \wedge \ \varphi_{\text{per}} \ \wedge \ \varphi_{\text{task}} \ \wedge \ \varphi_{\text{resp}}^{\text{ss}},$$

where

$$\varphi_{\text{safe}} := \square\psi_1, \qquad \textbf{Safety}$$

Specification Language

- We consider formulas of the form:

$$\varphi = \varphi_{\text{safe}} \;\wedge\; \varphi_{\text{resp}} \;\wedge\; \varphi_{\text{per}} \;\wedge\; \varphi_{\text{task}} \;\wedge\; \varphi_{\text{resp}}^{\text{ss}},$$

where

$$\varphi_{\text{safe}} := \Box\psi_1,$$ **Safety**

$$\varphi_{\text{resp}} := \bigwedge_{j\in I_2} \Box(\psi_{2,j} \implies \bigcirc\phi_{2,j}),$$ **Response**

Specification Language

- We consider formulas of the form:

$$\varphi = \varphi_{\text{safe}} \;\wedge\; \varphi_{\text{resp}} \;\wedge\; \varphi_{\text{per}} \;\wedge\; \varphi_{\text{task}} \;\wedge\; \varphi_{\text{resp}}^{\text{ss}},$$

where

$$\varphi_{\text{safe}} := \Box\psi_1,$$ **Safety**

$$\varphi_{\text{resp}} := \bigwedge_{j\in I_2} \Box(\psi_{2,j} \implies \bigcirc\phi_{2,j}),$$ **Response**

$$\varphi_{\text{per}} := \Diamond\,\Box\,\psi_3,$$ **Persistence (stability)**

Specification Language

- We consider formulas of the form:

$$\varphi = \varphi_{\text{safe}} \;\wedge\; \varphi_{\text{resp}} \;\wedge\; \varphi_{\text{per}} \;\wedge\; \varphi_{\text{task}} \;\wedge\; \varphi_{\text{resp}}^{\text{ss}},$$

where

$$\varphi_{\text{safe}} := \Box\psi_1,$$ **Safety**

$$\varphi_{\text{resp}} := \bigwedge_{j\in I_2} \Box(\psi_{2,j} \implies \bigcirc\phi_{2,j}),$$ **Response**

$$\varphi_{\text{per}} := \Diamond\,\Box\,\psi_3,$$ **Persistence (stability)**

$$\varphi_{\text{task}} := \bigwedge_{j\in I_4} \Box\,\Diamond\,\psi_{4,j},$$ **Repeated tasks**

Specification Language

- We consider formulas of the form:

$$\varphi = \varphi_{\text{safe}} \;\wedge\; \varphi_{\text{resp}} \;\wedge\; \varphi_{\text{per}} \;\wedge\; \varphi_{\text{task}} \;\wedge\; \varphi_{\text{resp}}^{\text{ss}},$$

where

$$\varphi_{\text{safe}} := \Box \psi_1 ,$$ **Safety**

$$\varphi_{\text{resp}} := \bigwedge_{j \in I_2} \Box(\psi_{2,j} \implies \bigcirc \phi_{2,j}),$$ **Response**

$$\varphi_{\text{per}} := \Diamond \Box \psi_3 ,$$ **Persistence (stability)**

$$\varphi_{\text{task}} := \bigwedge_{j \in I_4} \Box \Diamond \psi_{4,j} ,$$ **Repeated tasks**

$$\varphi_{\text{resp}}^{\text{ss}} := \bigwedge_{j \in I_5} \Diamond \Box (\psi_{5,j} \implies \bigcirc \phi_{5,j}).$$ **Steady-state response**

Related Work

- GR(1) [Piterman06]

$$(\Box\Diamond p_1 \wedge \cdots \wedge \Box\Diamond p_m) \quad \rightarrow \quad (\Box\Diamond q_1 \wedge \cdots \wedge \Box\Diamond q_n)$$

- Related logics: AlurT04, Ehlers11, MalerPS95

- How this work differs:
  - More system guarantees than GR(1)
  - No environment liveness assumptions

Problem Statement

- **Given**:
  - Non-deterministic system
  - Temporal logic specification φ (in fragment)

- **Problem**: Create control policy μ such that that $T^{\mu}(s_0)$ satisfies φ

Value Function and Reachability

- $V^c_B(s)$: minimum cost to reach set B from state s under all resolutions of the non-determinism

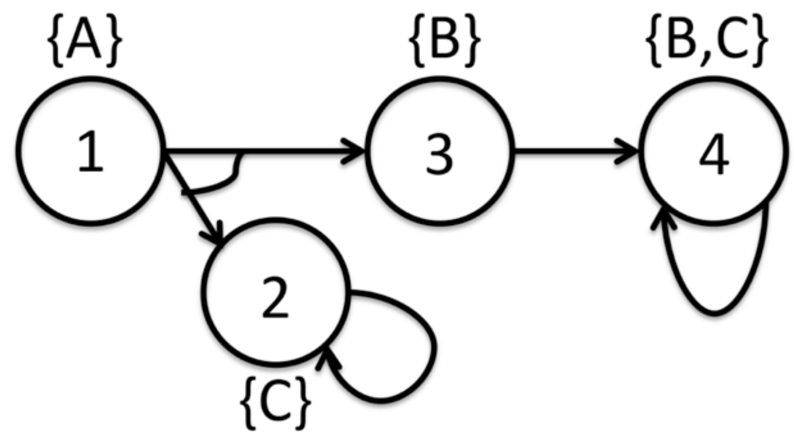
$$V^c_{B,\mathcal{T}}(s) = \min_{a \in A(s)} \max_{t \in R(s,a)} V^c_{B,\mathcal{T}}(t) + c(s,a,t)$$

Value Function and Reachability

- $V^c_B(s)$: minimum cost to reach set B from state s under all resolutions of the non-determinism

$$V^c_{B,\mathcal{T}}(s) = \min_{a \in A(s)} \max_{t \in R(s,a)} V^c_{B,\mathcal{T}}(t) + c(s,a,t)$$

- Example
  - $V^c_4(1) = \infty$
  - $V^c_4(2) = \infty$
  - $V^c_4(3) = 1$
  - $V^c_4(4) = 0$
  - CPre(4) = { 3, 4 }

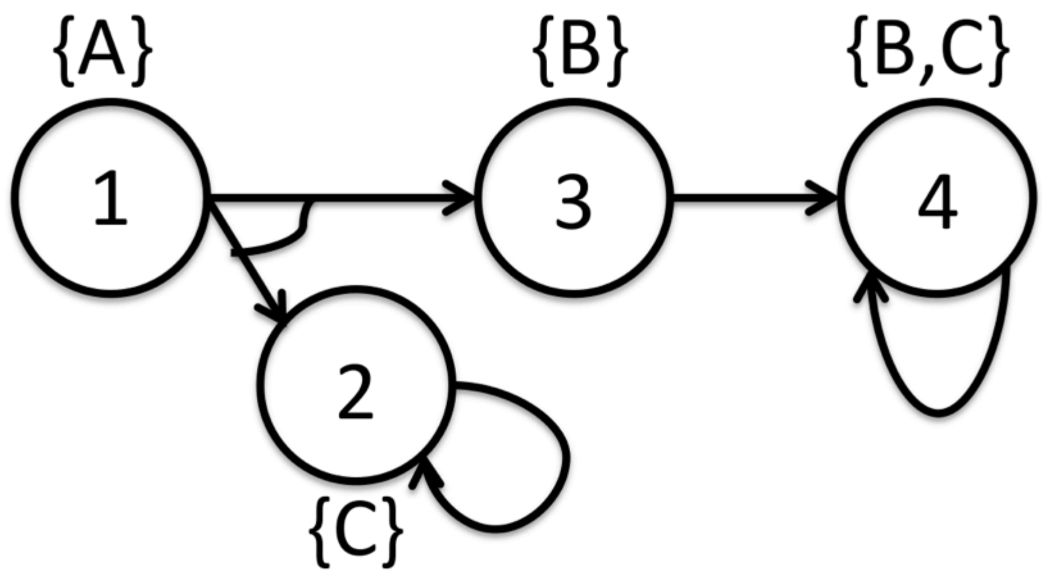


Outline

- Motivation
- Preliminaries
- Feasible control policies
  - Create “safe” subgraph
  - Compute task cycle (liveness)
- Examples

Removing Unsafe Behaviors

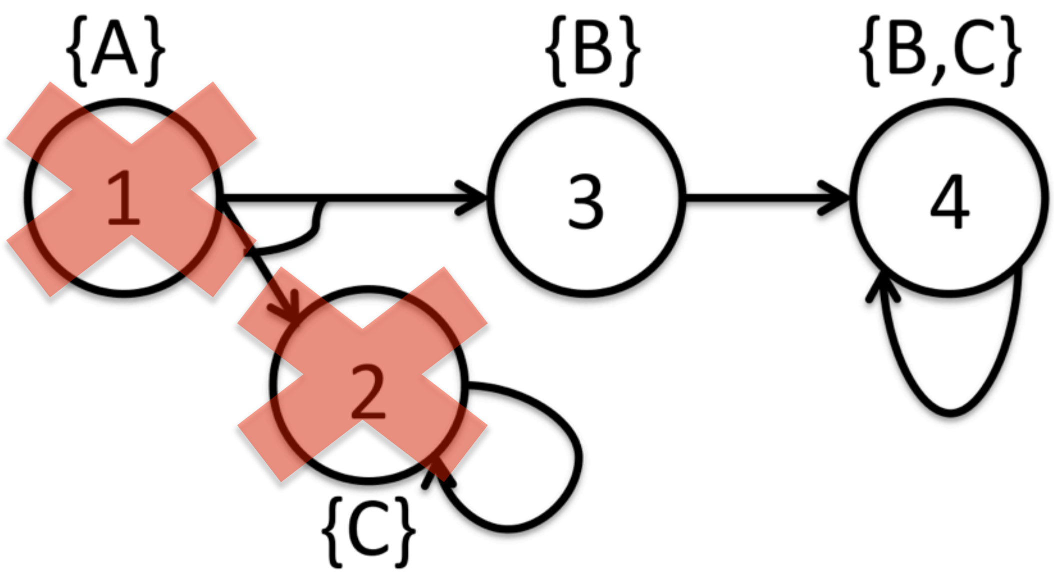
1. Remove unsafe states from T (φ_{safe}, φ_{per})
2. Remove unsafe transitions from T (φ_{resp}, $\varphi^{ss}{}_{resp}$)



Removing Unsafe Behaviors

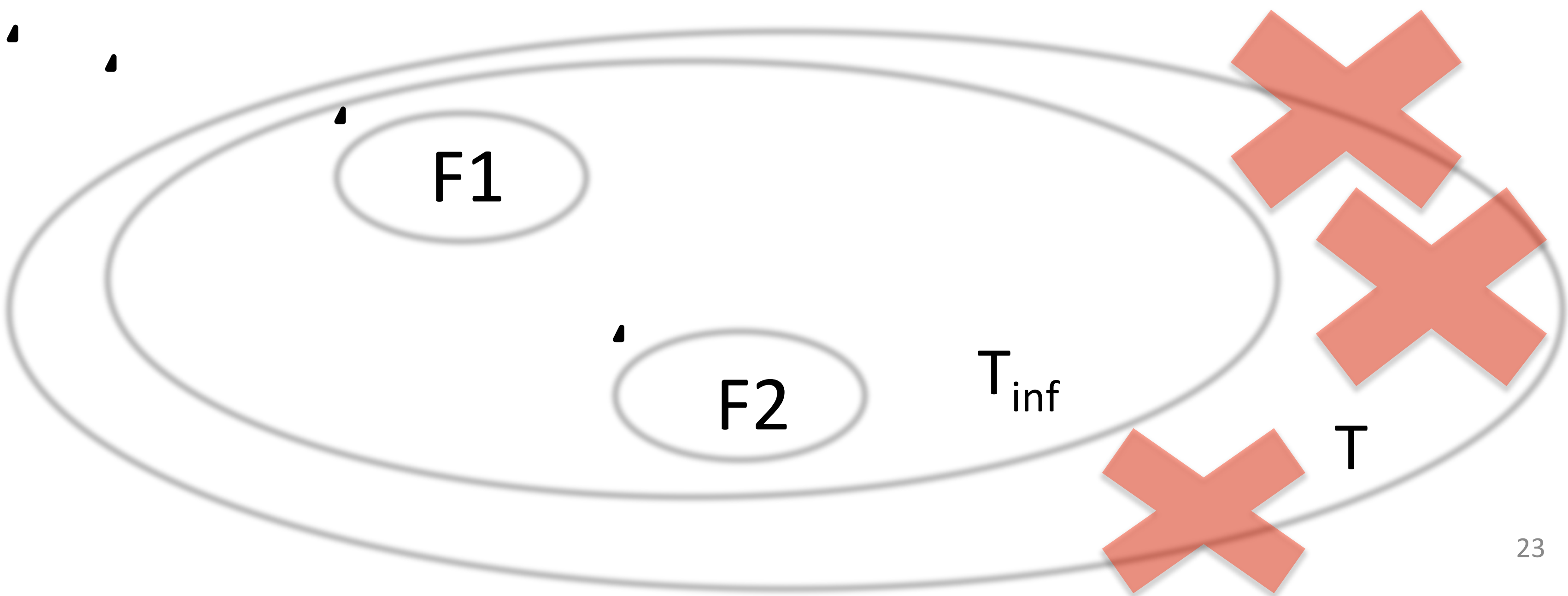
1. Remove unsafe states from T (φ_{safe}, φ_{per})
2. Remove unsafe transitions from T (φ_{resp}, $\varphi^{ss}{}_{resp}$)

φ_{safe} = [] B



Repeated Tasks

1. Remove unsafe states from T (φ_{safe}, φ_{per})
2. Remove unsafe transitions from T (φ_{resp}, φ^{ss}_{resp})
3. Compute task cycle (generalized Büchi) on T_{inf}



Generalized Büchi Game

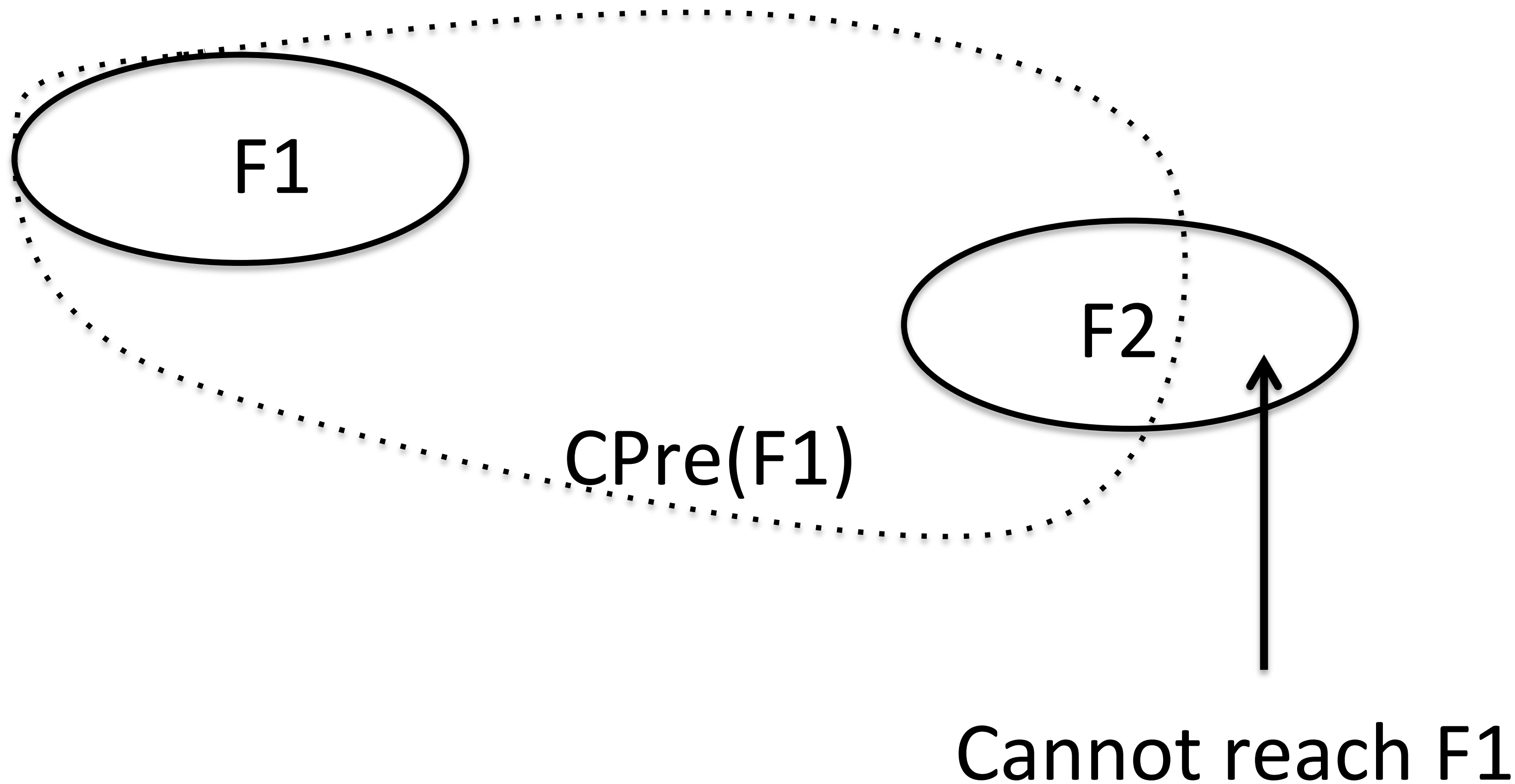
φ_{task} = []<> F1 & []<> F2

Iterate until sets stop changing.

ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> A & []<> B



ChatterjeeHP06, McNaughton93

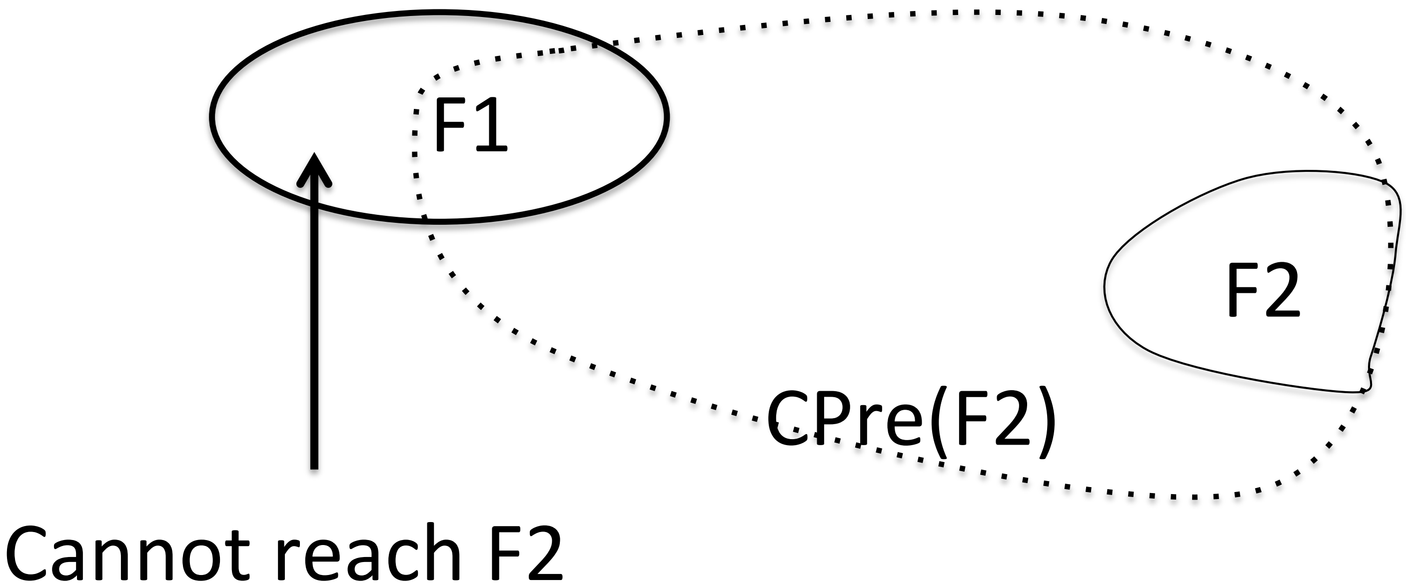
Generalized Büchi Game

φ_{task} = []<> A & []<> B

ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> A & []<> B



ChatterjeeHP06, McNaughton93

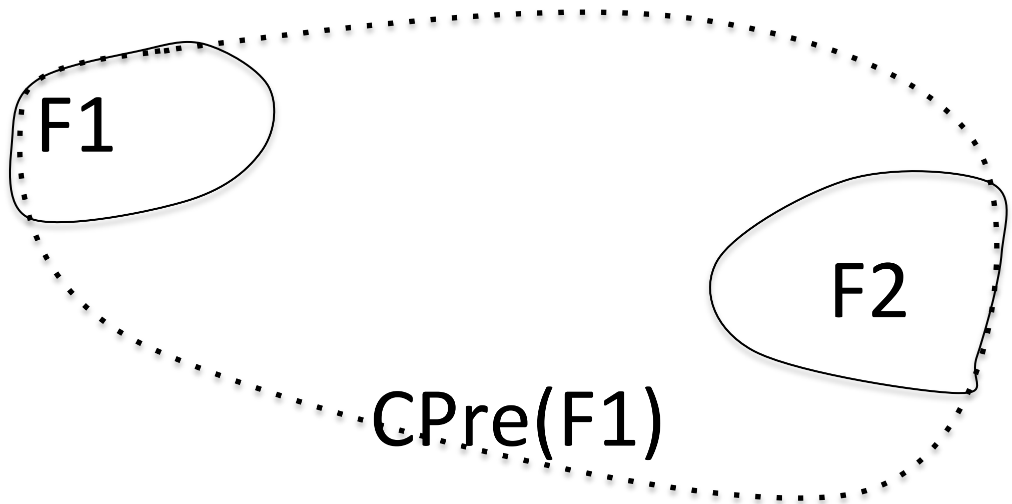
Generalized Büchi Game

φ_{task} = []<> A & []<> B

ChatterjeeHP06, McNaughton93

Generalized Büchi Game

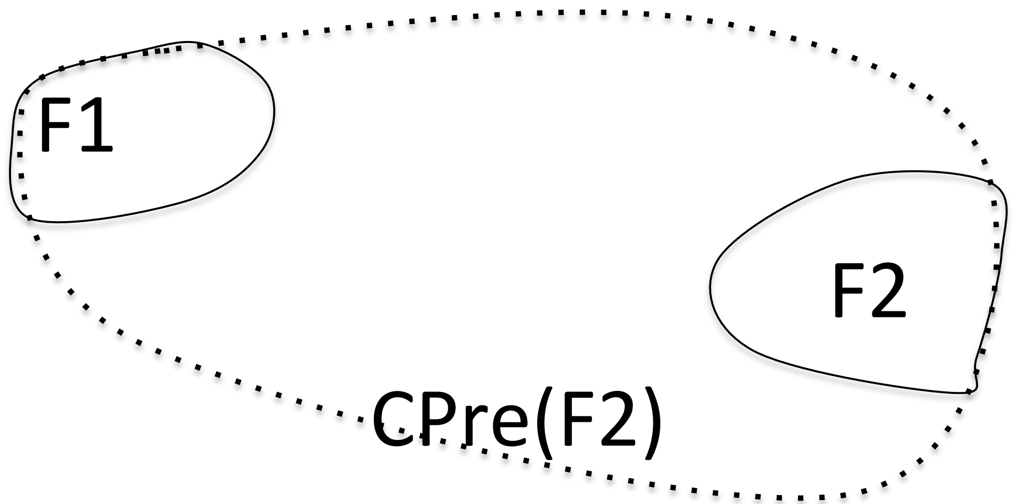
φ_{task} = []<> A & []<> B



ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> A & []<> B



ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> A & []<> B

DONE!

ChatterjeeHP06, McNaughton93

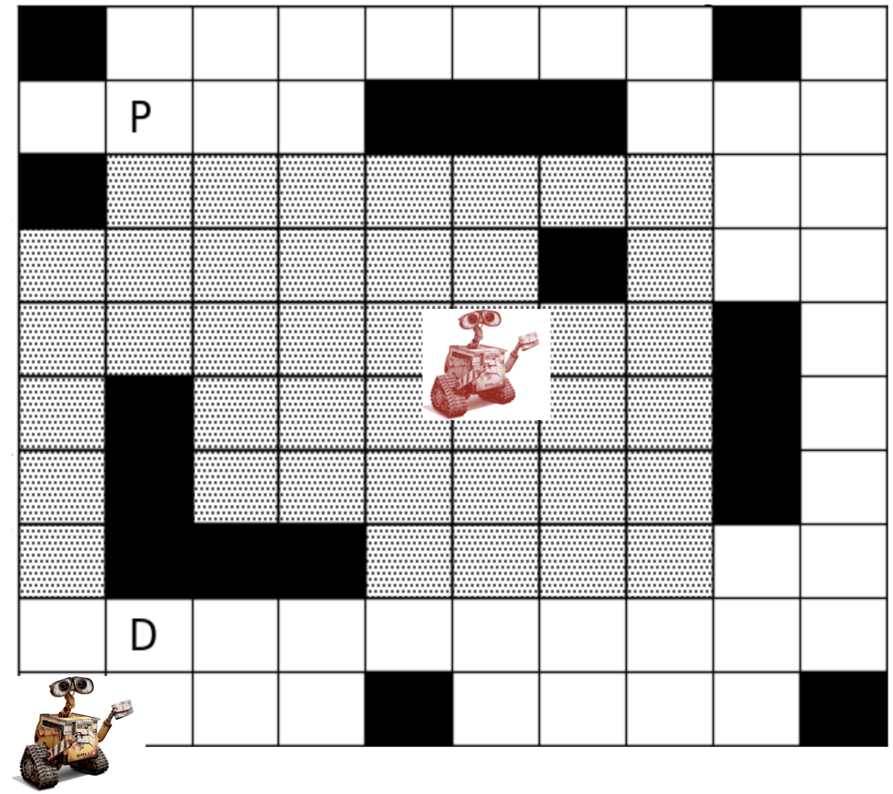
Reactive Synthesis Summary

- What have we computed?
  - Largest possible task sets F_j
  - Largest possible winning set W
  - Finite-memory control policies
- Time complexity:
  - S = # states, R = # transitions, φ = # specs

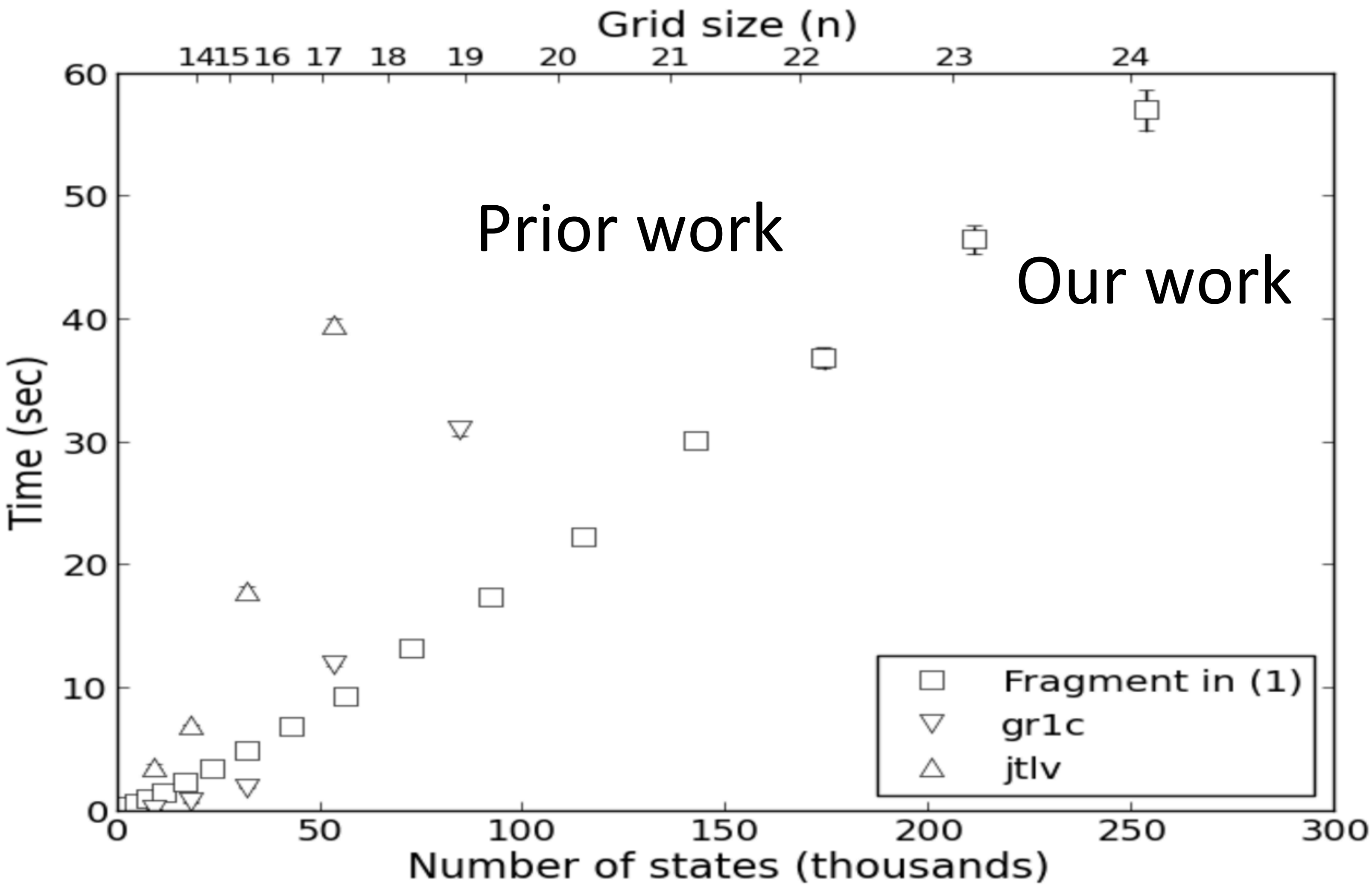
Language	DTS	NTS
Our method	$O(\lvert\varphi\rvert(\lvert S\rvert+\lvert R\rvert))$	$O(\lvert\varphi\rvert F_{\min}(\lvert S\rvert+\lvert R\rvert))$
GR(1)	$O(\lvert\varphi\rvert\lvert S\rvert\lvert R\rvert)$	$O(\lvert\varphi\rvert\lvert S\rvert\lvert R\rvert)$
LTL	$O(2^{(\lvert\varphi\rvert)}(\lvert S\rvert+\lvert R\rvert))$	$O(2^{2^{(\lvert\varphi\rvert)}}(\lvert S\rvert+\lvert R\rvert))$

Example: Pickup and delivery

- System:
  - Robot and obstacle move to adjacent regions
- Specs:
  - Always avoid collisions
  - Repeatedly visit **P**ickup and **D**ropoff locations



Results



Grid size (n)
14
15
16
17
18
19
20
21
22
23
24
60
50
40
30
20
10
0
Time (sec)
0
50
100
150
200
250
300
Number of states (thousands)
Prior work
Our work
Fragment in (1)
gr1c
jtlv

Conclusions

- Main results
  - 2-EXP improvement over LTL
  - Non-deterministic and stochastic systems
  - Simple and extensible framework (e.g. optimality)
  - Promising alternative to GR1
- These are the easy tasks!

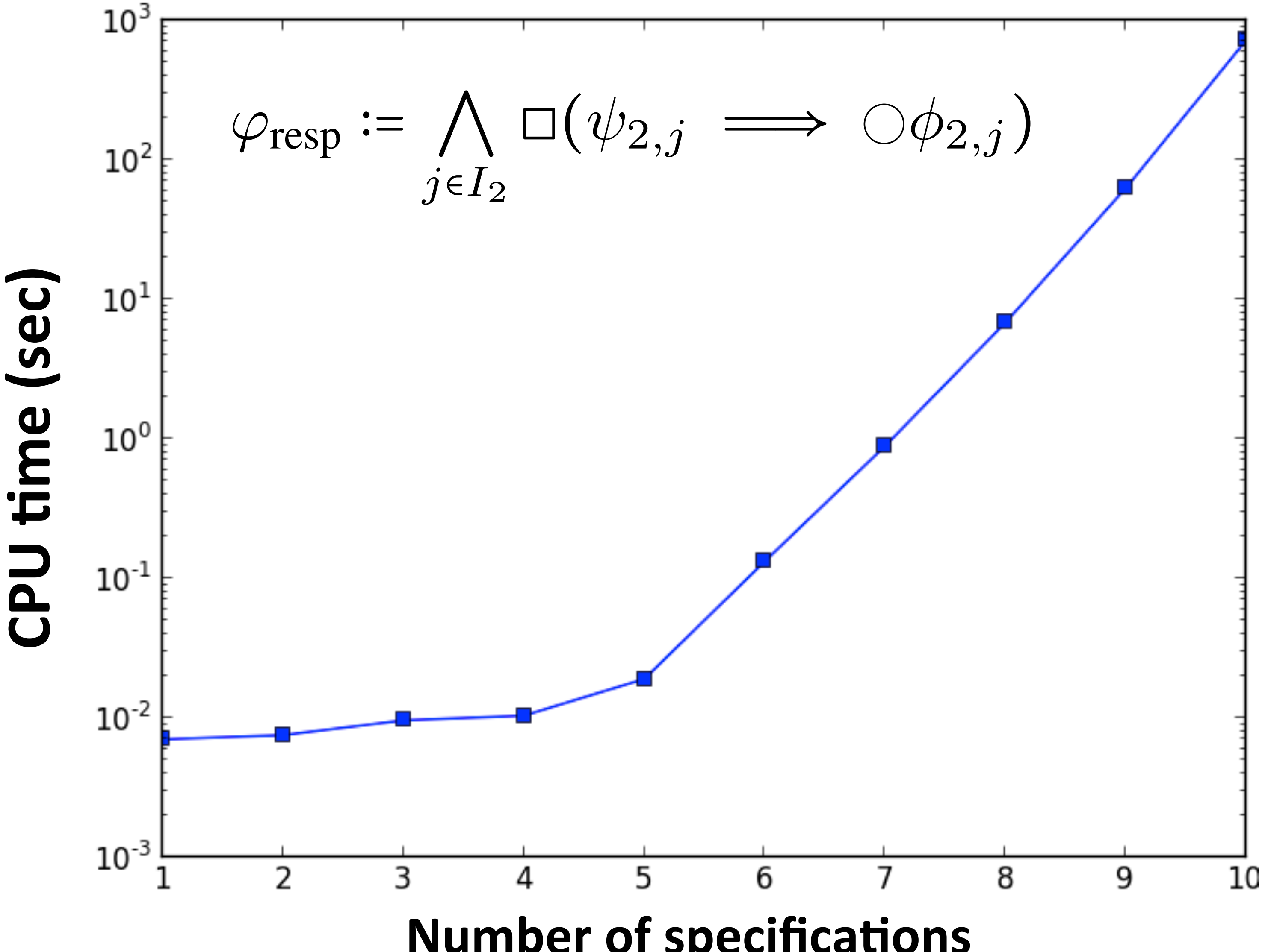
Thanks!

- Questions?

- Funding
  - NDSEG fellowship
  - Boeing
  - AFOSR

Backup

Do Standard Methods Work?



Acceptance Conditions

- Let σ = $s_0 s_1 s_2$ … be a run of the system
- Let ψ be a propositional formula. It holds at states in [[ψ]]
- Vis(σ) = set of states visited
- Inf(σ) = set of states visited repeatedly

- $\sigma \models \Box\psi$ iff $\mathit{Vis}(\sigma) \subseteq [[\psi]]$,
- $\sigma \models \Diamond \Box \psi$ iff $\mathit{Inf}(\sigma) \subseteq [[\psi]]$,
- $\sigma \models \Box \Diamond \psi$ iff $\mathit{Inf}(\sigma) \cap [[\psi]] \neq \emptyset$,
- $\sigma \models \Box(\psi \implies \bigcirc\phi)$ iff $\sigma_i \notin [[\psi]]$ or $\sigma_{i+1} \in [[\phi]]$ for all i,
- $\sigma \models \Diamond \Box (\psi \implies \bigcirc\phi)$ iff there exists an index j such that $\sigma_i \notin [[\psi]]$ or $\sigma_{i+1} \in [[\phi]]$ for all $i \geq j$.

Modeling Non-determinism

state = (UAV, car)