Optimal Control of Non-deterministic Systems for a Fragment of Temporal Logic

Eric M. Wolff[1]

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

[1]Caltech and [2]UPenn

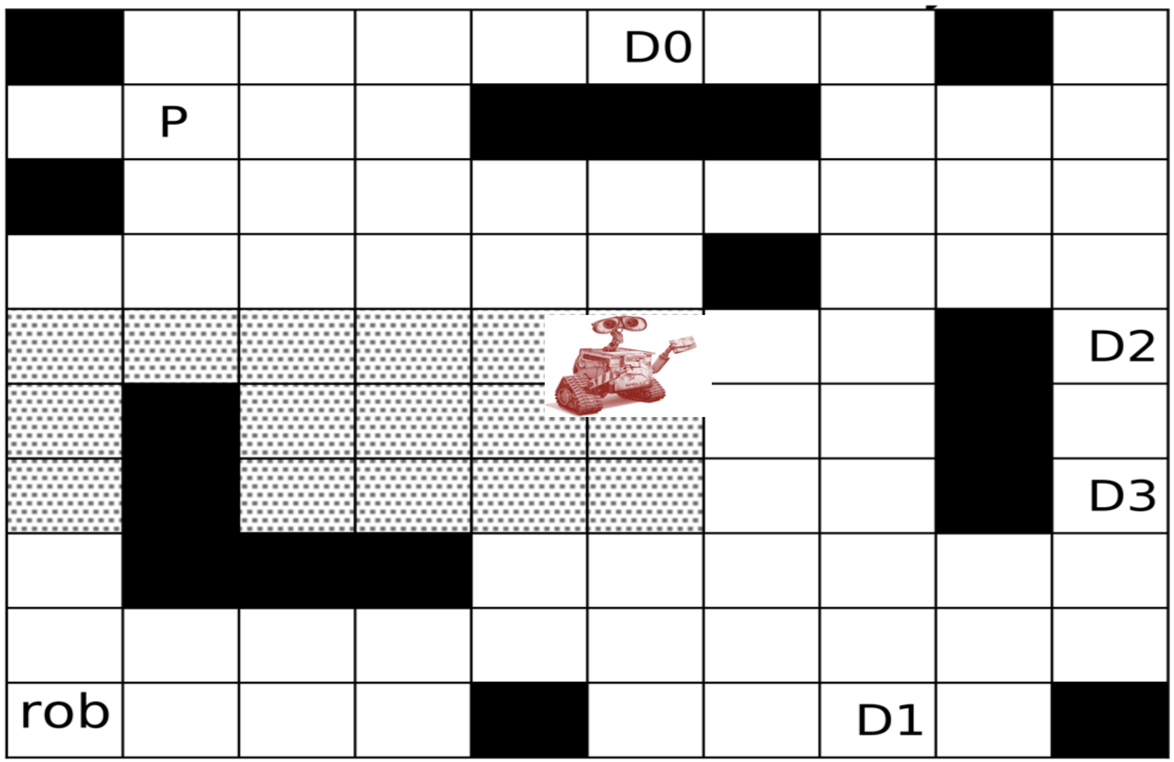
SYNT

July 13, 2013

Autonomous Systems in the Field

Planning in a Dynamic Environment

- Dynamic obstacles
- Safe navigation and repetitive tasks



Our Contributions

- Introduce **expressive** and **efficient** fragment of linear temporal logic
  - Optimal control
  - Non-deterministic and stochastic systems
  - Simple and extensible framework

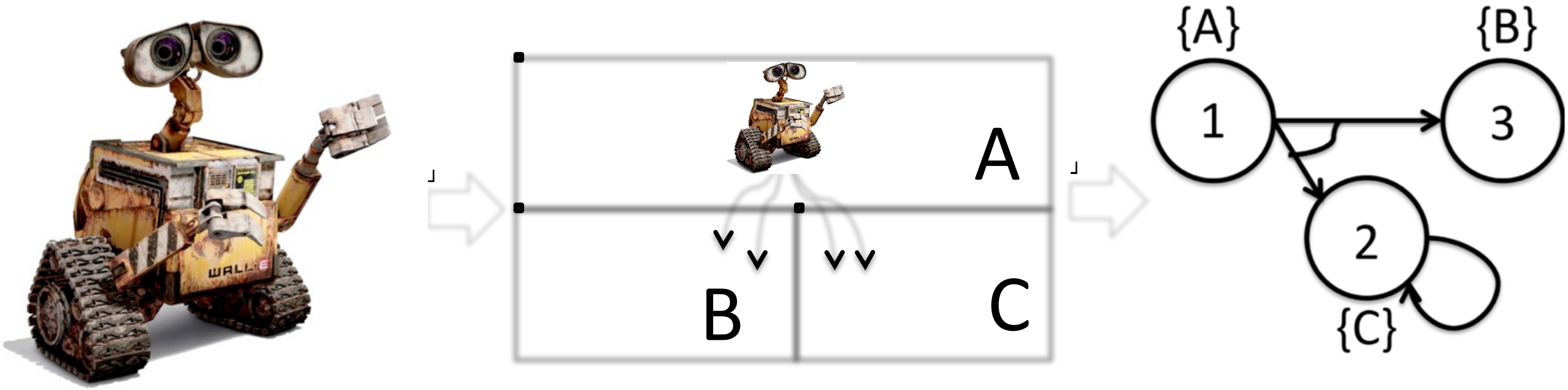
Outline

- Preliminaries
- Feasible control policies
- Optimal control policies
- Examples
- Future directions

Outline

- Preliminaries
- Feasible control policies
- Optimal control policies
- Examples
- Future directions

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, HabetsCS06, KaramanF09, KloetzerB08, WongpiromsarnTM12, and more

Discrete Abstractions

- dx/dt = f(x,u)
  - x = system state
  - u = control input

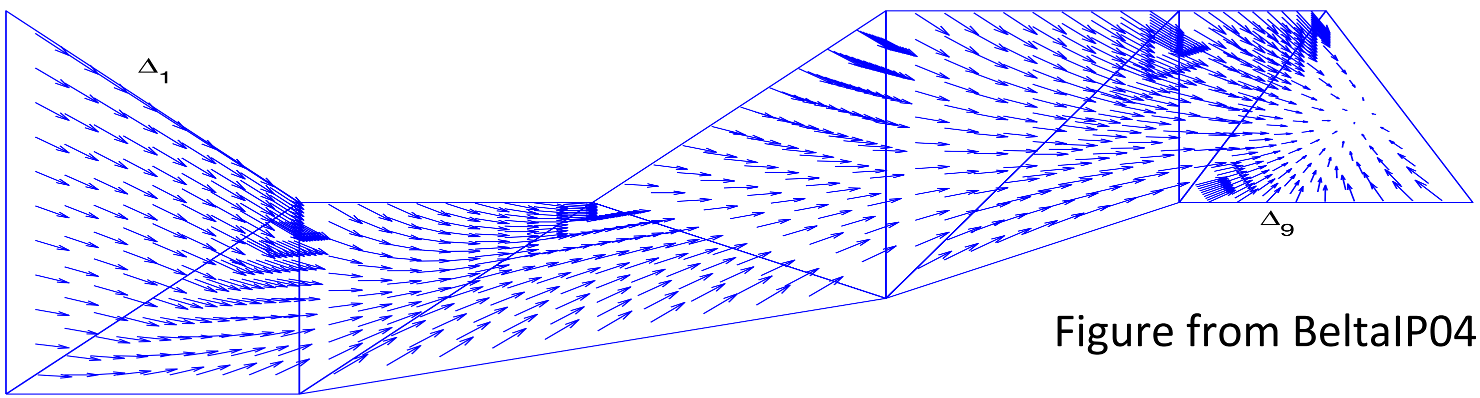
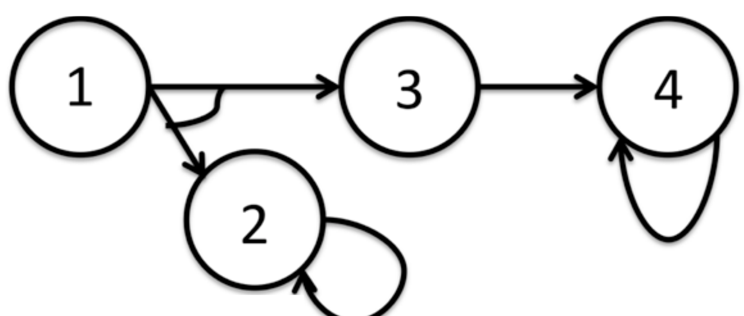


Figure from BeltaIP04

- Discrete states = sets of continuous states

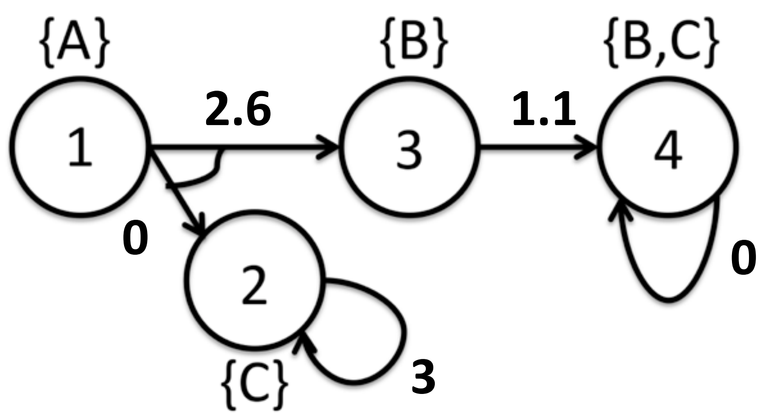
Non-deterministic Transition Systems

- A **non-deterministic transition system (NTS)** is a tuple T = (S, A, R, s_0, AP, L) where
  - **states** S,
  - **actions** A,
  - **transition** function R: S x A $\rightarrow 2^S$,
  - initial state s_0,



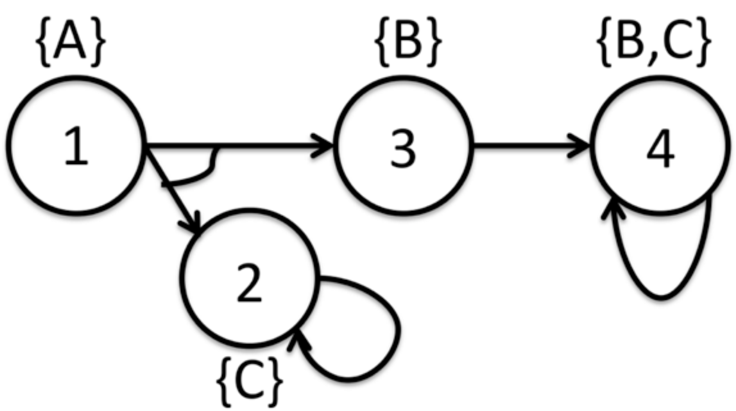
Non-deterministic Transition Systems

- A **non-deterministic transition system (NTS)** is a tuple T = (S, A, R, s_0, AP, L) where
  - **states** S,
  - **actions** A,
  - **transition** function R: $S \times A \rightarrow 2^S$,
  - initial state s_0,
  - atomic propositions AP,
  - labeling function L : $S \rightarrow 2^{AP}$, and
  - cost function c : $S \times A \times S \rightarrow \Re$.



Control Policies

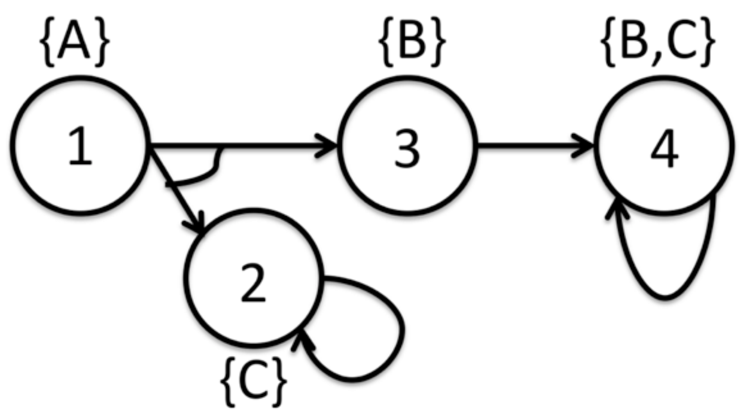
- **Finite-memory control policy:** μ: S x M → A x M



Control Policies

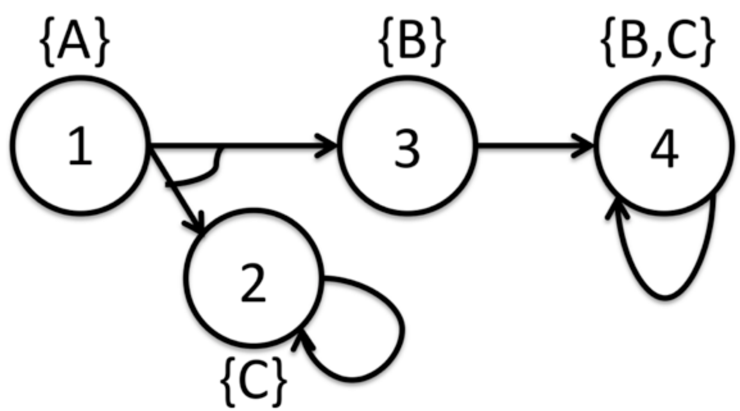
- **Finite-memory control policy:** $\mu: S \times M \rightarrow A \times M$

- **Two-player game**:
  - 1) **System** picks action using control policy
  - 2) **Environment** picks next state



Control Policies

- **Finite-memory control policy:** $\mu: S \times M \rightarrow A \times M$
- **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



Specification Language

- We consider formulas of the form:

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

where

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

Specification Language

- We consider formulas of the form:

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

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:

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

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

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

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

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

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

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

Related Work

- GR(1) [PitermanPS06, BloemJPPS12]

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

- GRabin(1) [Ehlers11]
- Related logics: AlurT04, MalerPS95

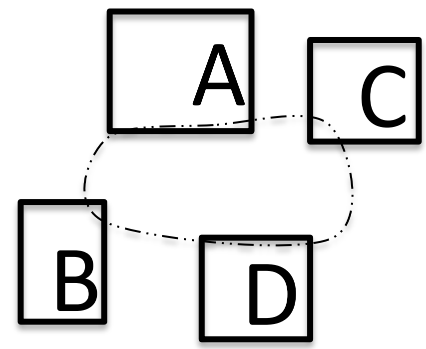
- How this work differs:
  - More system guarantees than GR(1)
  - No environment liveness assumptions

GR1 system + persistence

Cost Functions

- Average cost-per-task-cycle

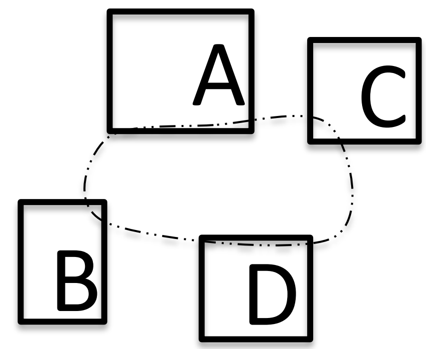
$$J'_{TC}(\sigma, \mu(\sigma)) := \limsup_{n \to \infty} \frac{\sum_{t=0}^{n} c(\sigma_t, \mu(\sigma_t), \sigma_{t+1})}{\sum_{t=0}^{n} I_{TC}(t)}$$



Cost Functions

- Average cost-per-task-cycle

$$J'_{TC}(\sigma, \mu(\sigma)) := \limsup_{n \to \infty} \frac{\sum_{t=0}^{n} c(\sigma_t, \mu(\sigma_t), \sigma_{t+1})}{\sum_{t=0}^{n} I_{TC}(t)}$$



- Not discussed today
  - Minimax (bottleneck) cost
  - Average cost

Problem Statement

- **Given**:
  - Non-deterministic transition system T
  - Temporal logic specification φ of the form

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

  - Cost function J

- **Problem**: Create control policy μ such that that the set of runs $T^{\mu}(s_0)$ satisfies φ and minimizes J

Value (Rank) Function and Reachability

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

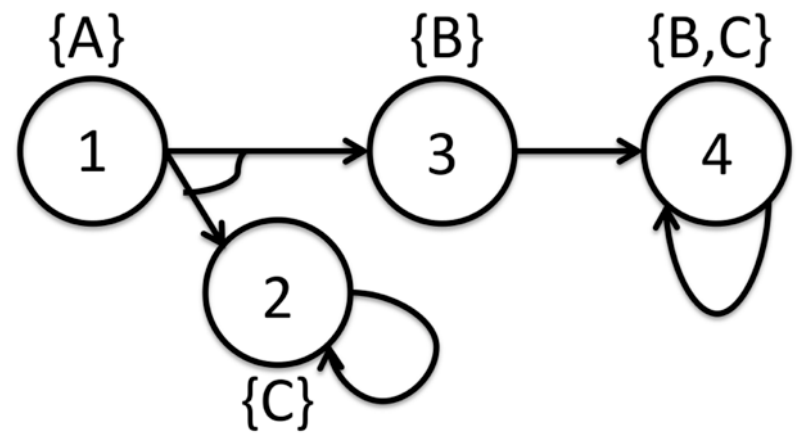
$$V^c_{B,\mathcal{T}}(s) = \min_{a \in A(s)} \max_{t \in R(s,a)} V^c_{B,\mathcal{T}}(t) + c(s,a,t)$$

Value (Rank) 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 } (attractor)

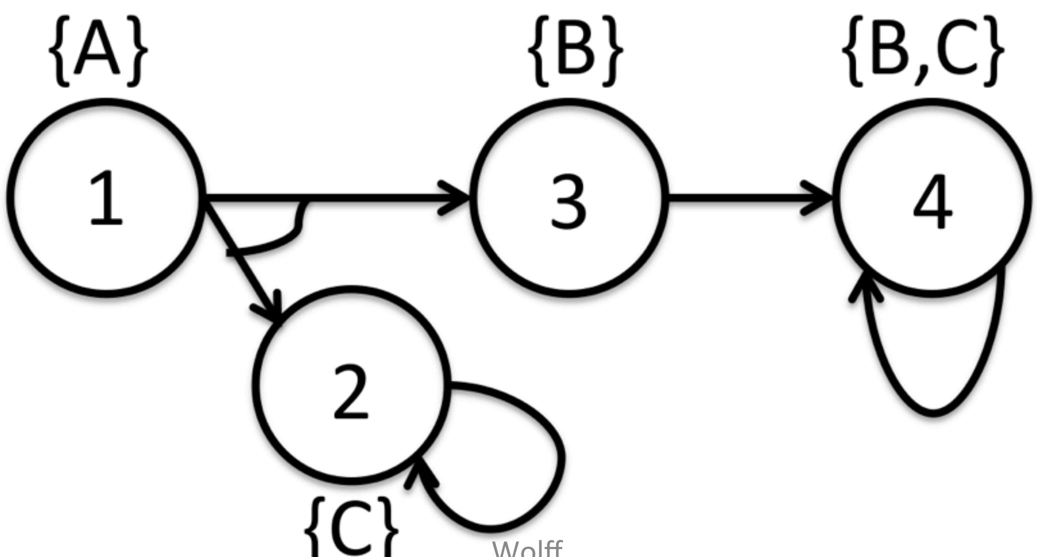


Outline

- Preliminaries
- Feasible control policies
- Optimal control policies
- Examples
- Future directions

Removing Unsafe Behaviors

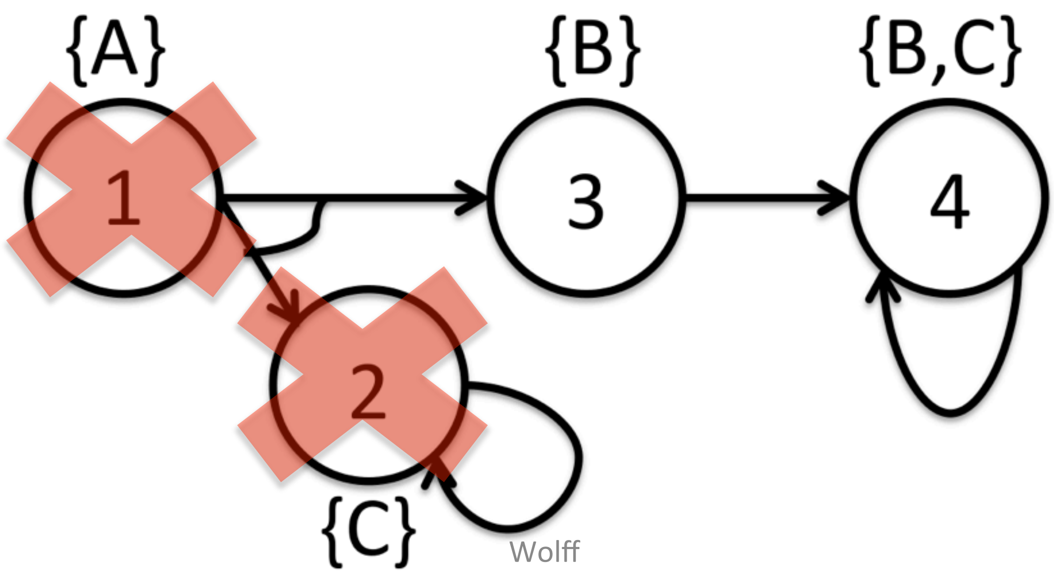
1. Remove unsafe states from T (φ_{safe}, φ_{per})
2. Remove unsafe transitions from T (φ_{resp}, $\varphi^{ss}{}_{resp}$)



Removing Unsafe Behaviors

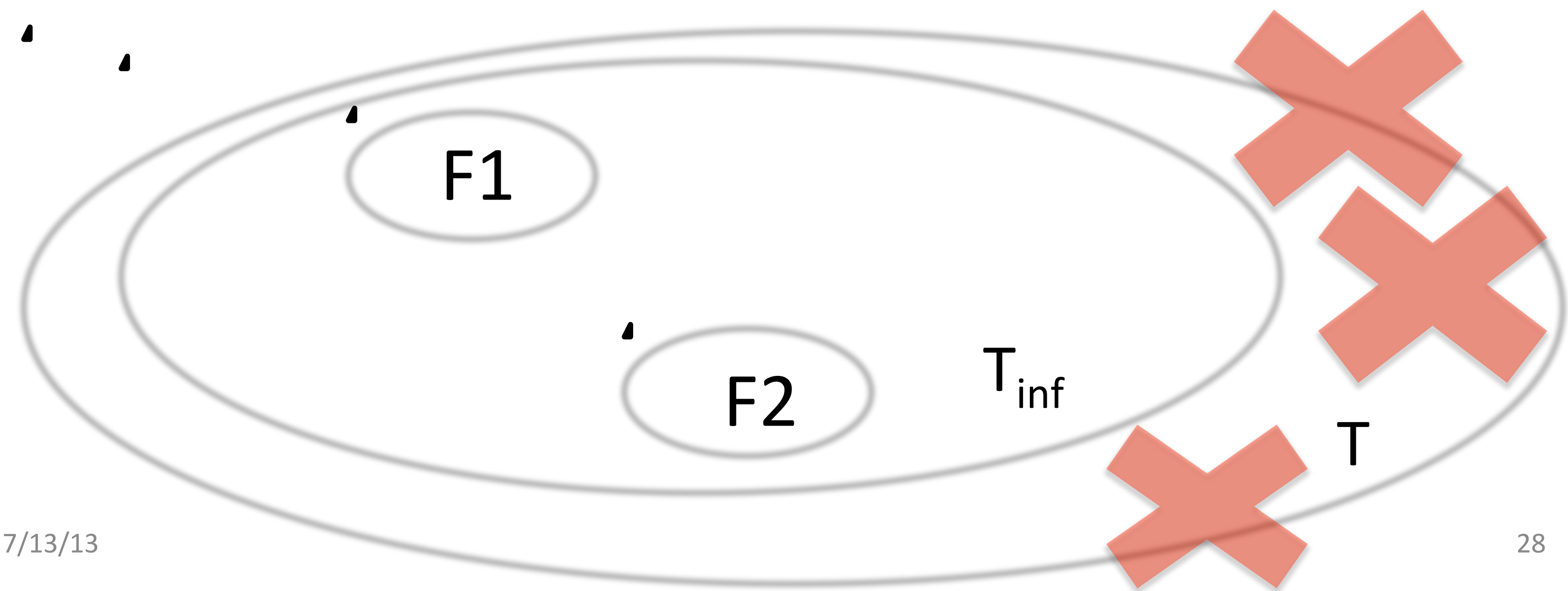
1. Remove unsafe states from T (φ_{safe}, φ_{per})
2. Remove unsafe transitions from T (φ_{resp}, $\varphi^{ss}{}_{resp}$)

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

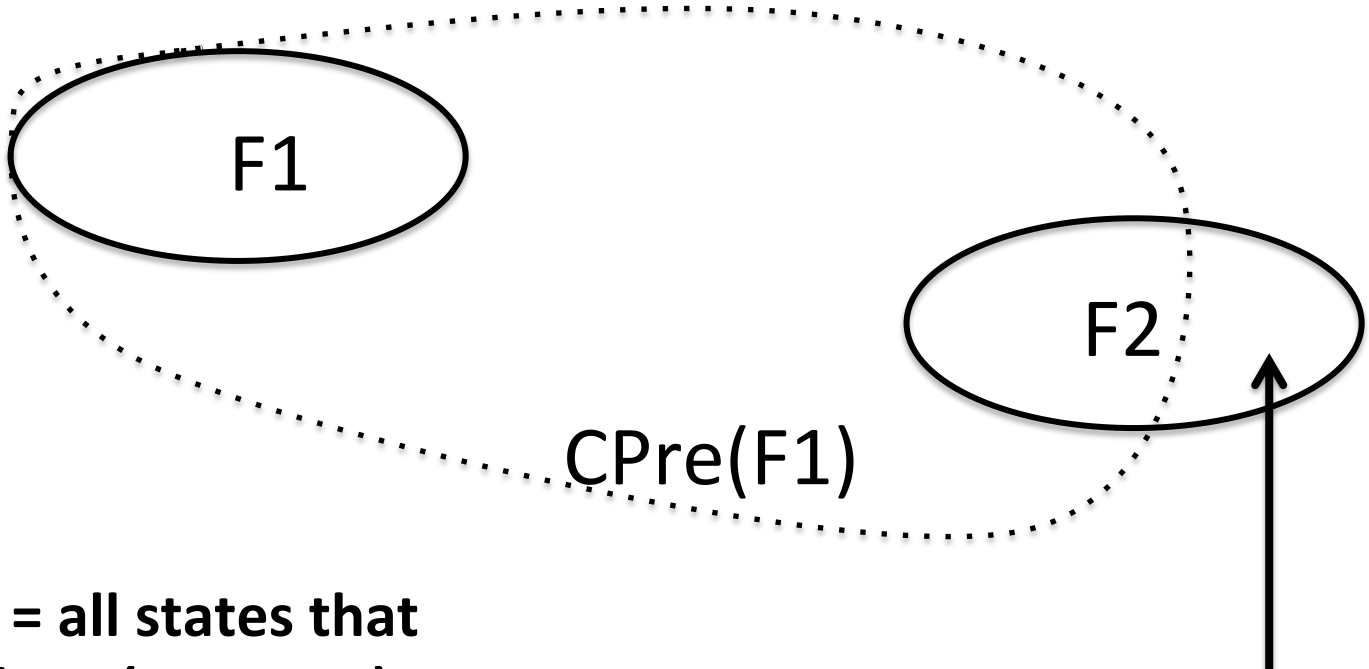
φ_{task} = []<> F1 & []<> F2

Iterate until sets stop changing.

ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> F1 & []<> F2



CPre(F1) = all states that can reach F1 (attractor)

Cannot reach F1

ChatterjeeHP06, McNaughton93

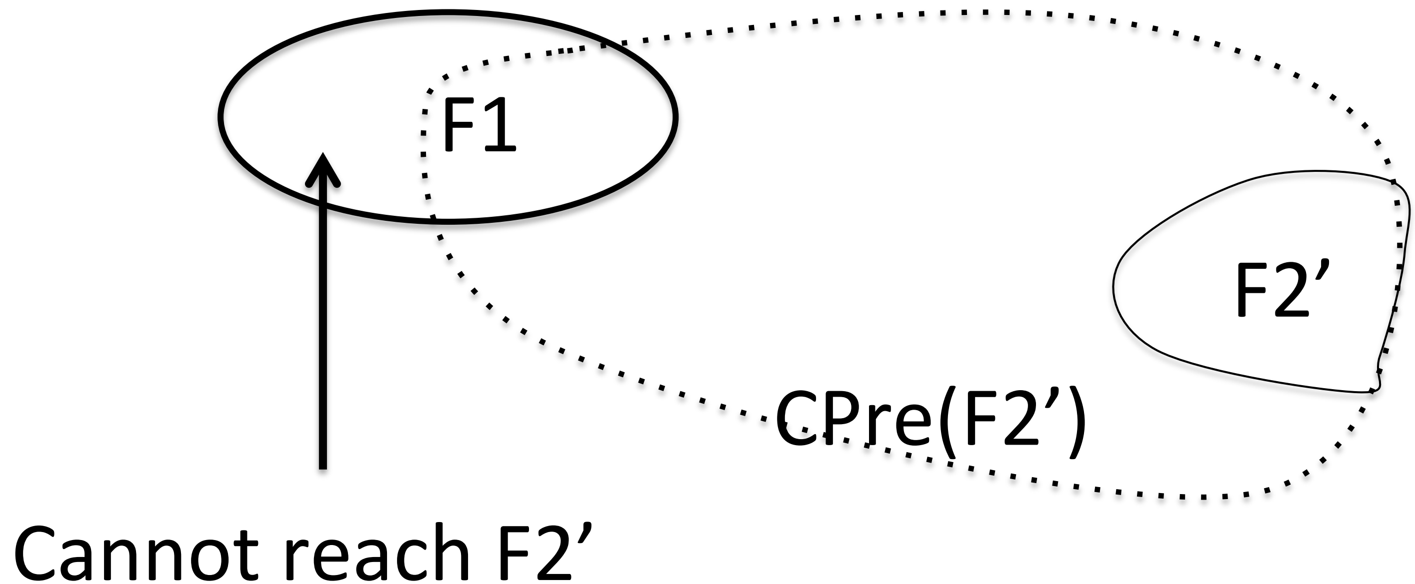
Generalized Büchi Game

φ_{task} = []<> F1 & []<> F2

ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> F1 & []<> F2



ChatterjeeHP06, McNaughton93

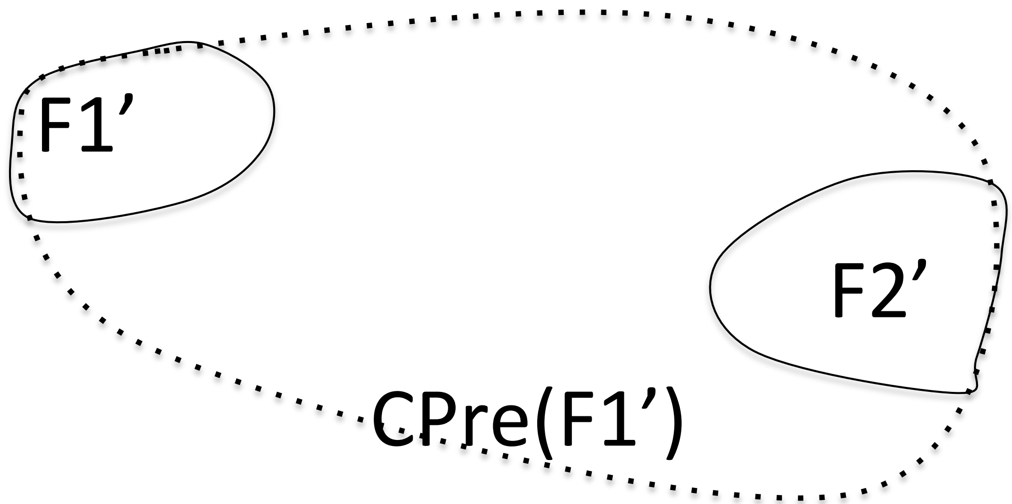
Generalized Büchi Game

φ_{task} = []<> F1 & []<> F2

ChatterjeeHP06, McNaughton93

Generalized Büchi Game

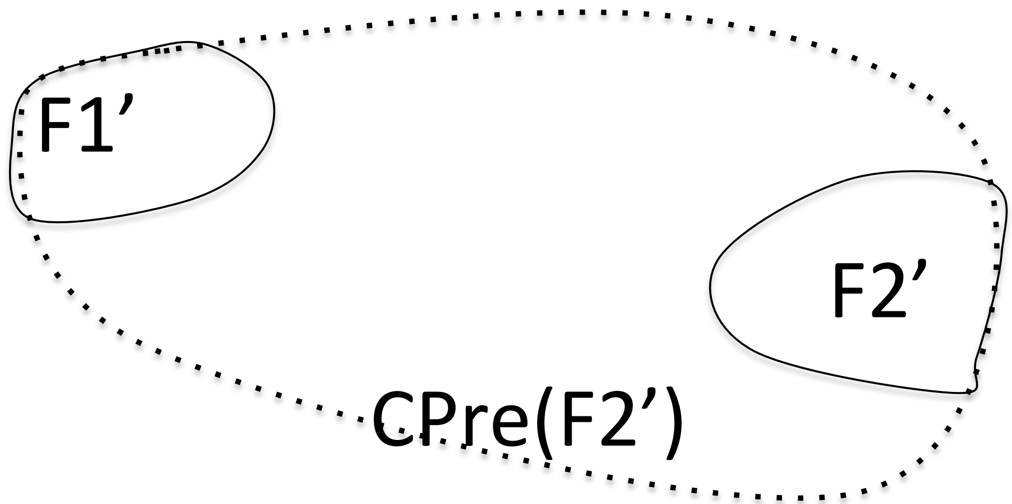
φ_{task} = []<> F1 & []<> F2



ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> F1 & []<> F2



ChatterjeeHP06, McNaughton93

Generalized Büchi Game

φ_{task} = []<> F1 & []<> F2

DONE!

Return the F_j task sets and their corresponding value functions.

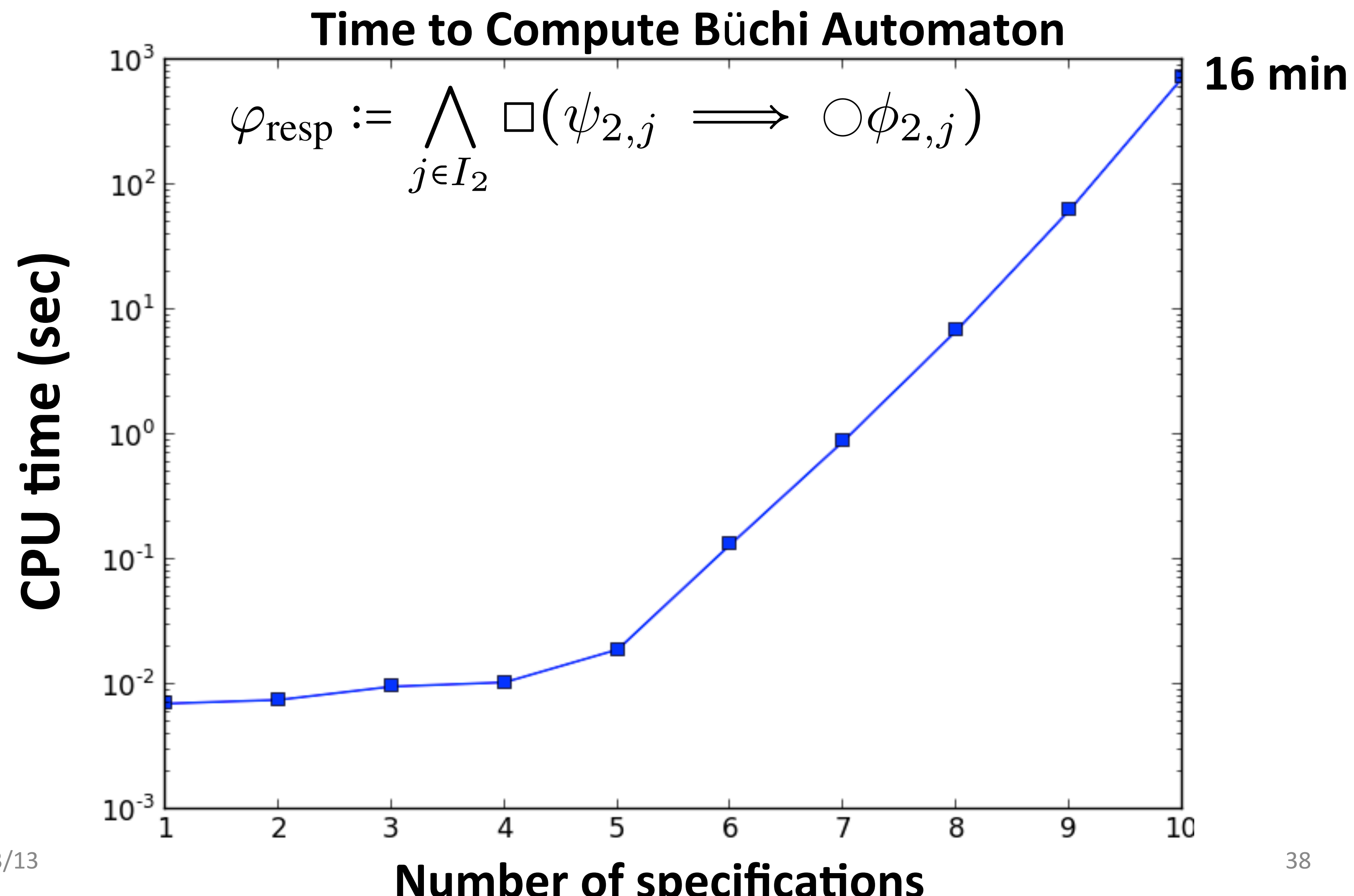
ChatterjeeHP06, McNaughton93

Feasible Synthesis Summary

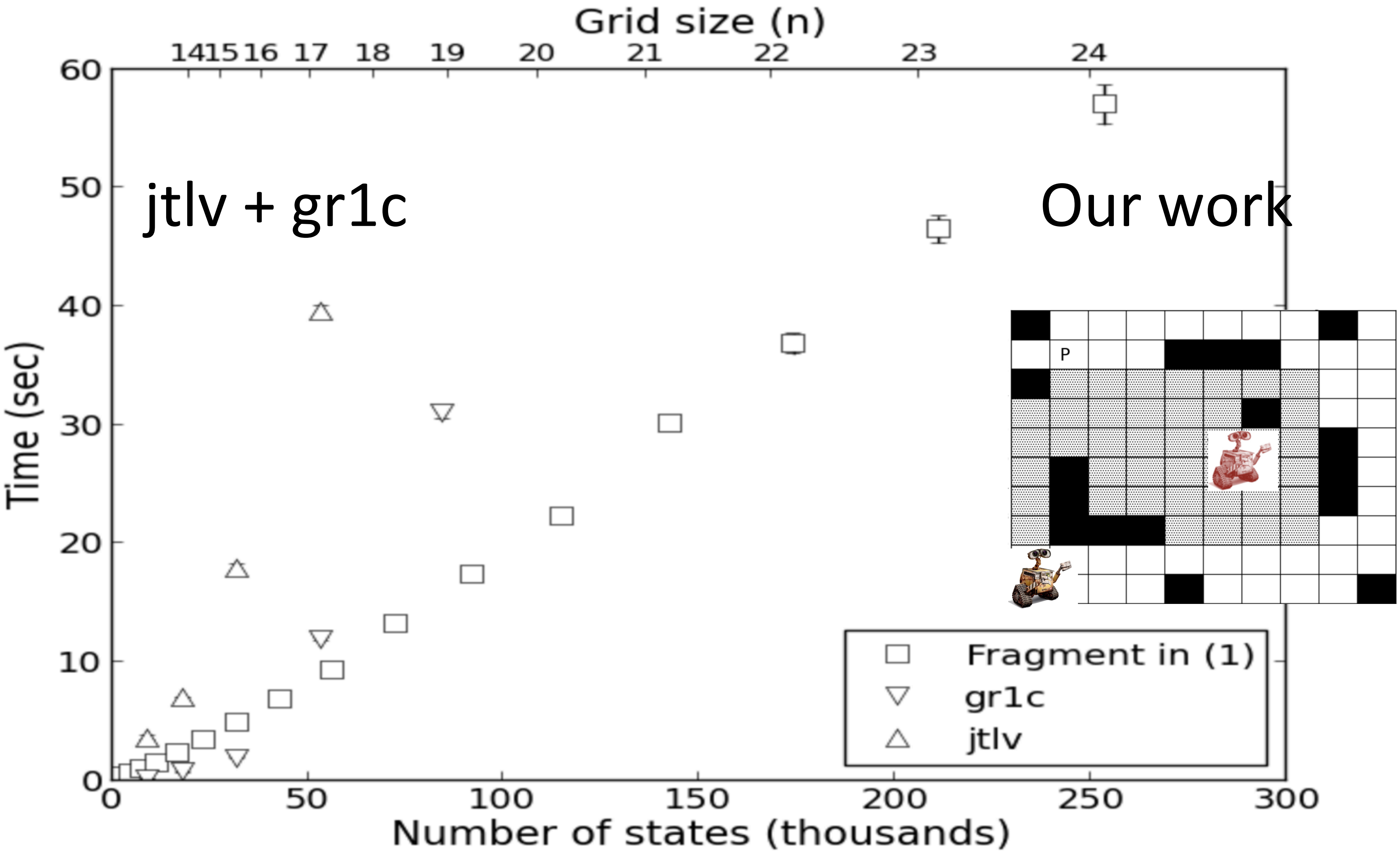
- What have we computed?
  - Largest possible task sets F_j
  - All states from which spec is feasible
  - Finite-memory control policies (from value fcn)
- Time complexity:
  - S = # states, R = # transitions, φ = # specs

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	$O(2^{(\lvert\varphi\rvert)}(\lvert S\rvert + \lvert R\rvert))$	$O(2^{2^{(\lvert\varphi\rvert)}}(\lvert S\rvert + \lvert R\rvert))$

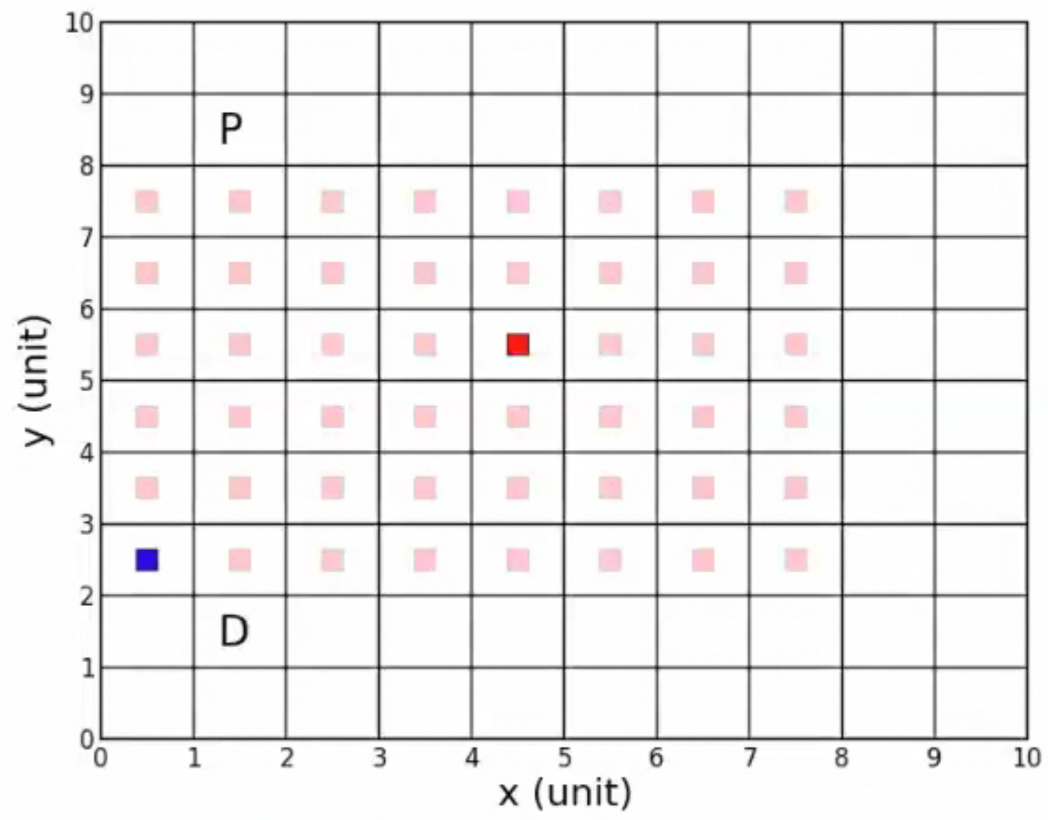
Do Standard Methods Work?



Comparison to GR(1)



An Example Run



- Task: Repeatedly visit locations P and D
- Blue = controlled robot (system)
- Red = non-deterministic obstacle (environment)

Outline

- Preliminaries
- Feasible control policies
- Optimal control policies
- Examples
- Future directions

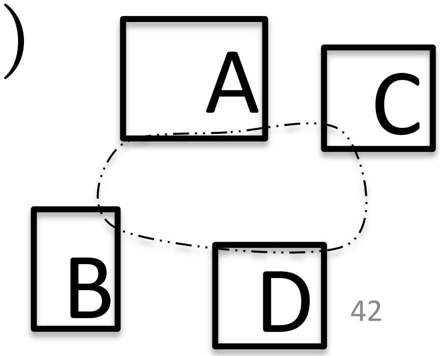
Average Cost-Per-Task-Cycle

- Average cost-per-task-cycle of run σ is

$$J'_{TC}(\sigma, \mu(\sigma)) := \limsup_{n \to \infty} \frac{\sum_{t=0}^{n} c(\sigma_t, \mu(\sigma_t), \sigma_{t+1})}{\sum_{t=0}^{n} I_{TC}(t)}$$

- The average cost-per-task-cycle cost function is

$$J_{TC}(\mathcal{T}^{\mu}(s)) := \max_{\sigma \in \mathcal{T}^{\mu}(s)} J'_{TC}(\sigma, \mu(\sigma))$$



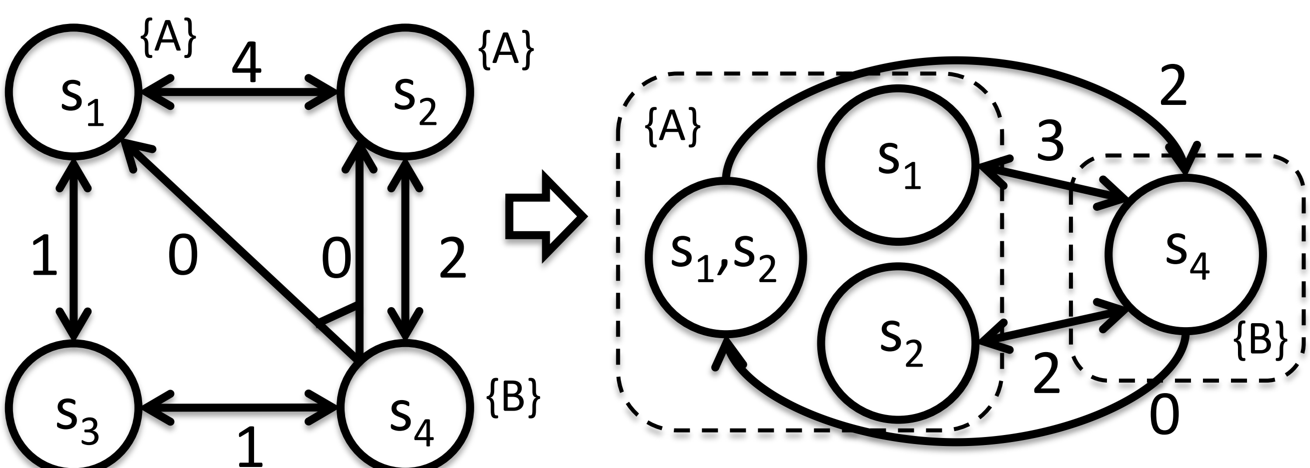
Optimality is Hard

- **Theorem**: Computing a control policy that is minimizes the average cost-per-task-cycle is NP-hard, even in the deterministic case.
- **Proof**: By constructing a generalized traveling salesman problem where tasks are nodes in the TSP graph.

Cost function:	Average	Minimax	Task Cycle
Complexity:	POLY	POLY in task graph	EXP in task graph

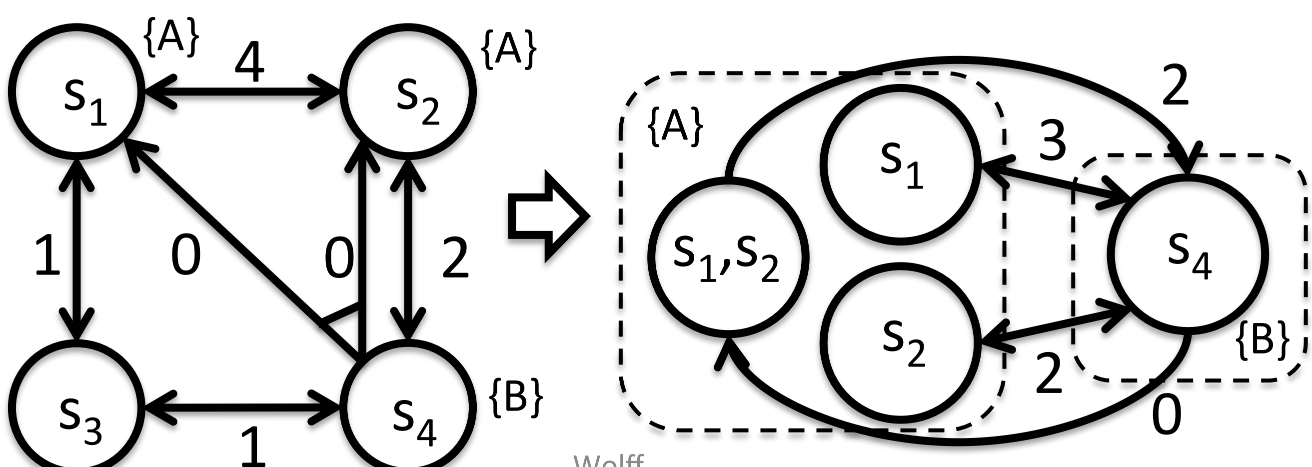
Task Graph

- Create new graph that encodes shortest paths between tasks



Task Graph

- Create new graph that encodes shortest paths between tasks
- Number of states
  - Deterministic: |F|
  - Non-deterministic: $\sum_{i \in I_4} 2^{|F_i|} - 1$



Fixed Ordering Assumption

- **Assumption**: Optimize over all fixed orderings of tasks

Fixed Ordering Assumption

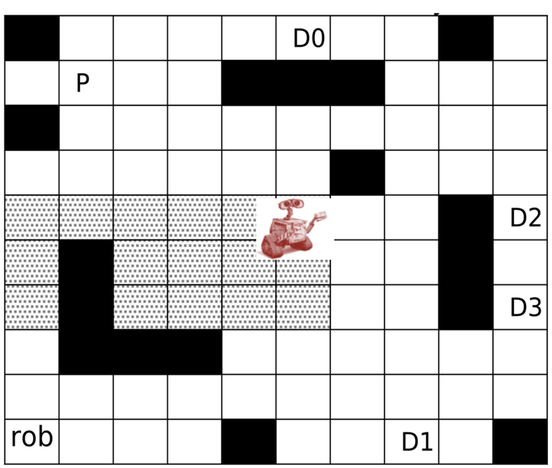
- **Assumption**: Optimize over all fixed orderings of tasks
- Solve generalized TSP on task graph
  - Use commercial solvers
  - Approximate solutions
- Solution gives optimal task ordering and subsets

Outline

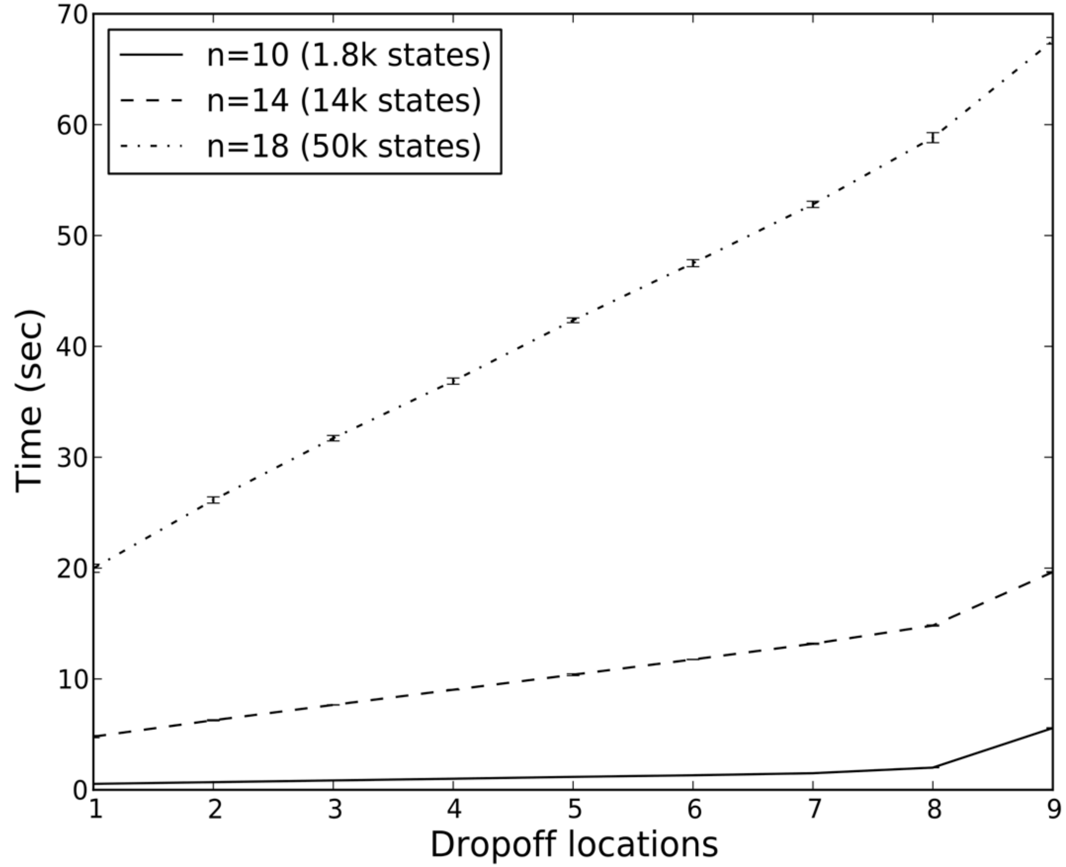
- Preliminaries
- Feasible control policies
- Optimal control policies
- Examples
- Future directions

Example: Pickup and Delivery

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



Optimal Control Policy Time

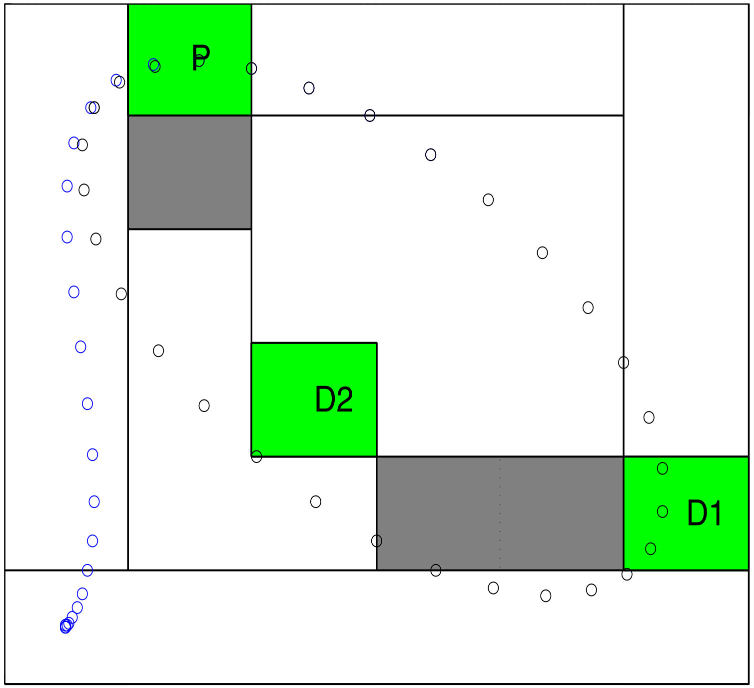


Outline

- Preliminaries
- Feasible control policies
- Optimal control policies
- Examples
- Future directions

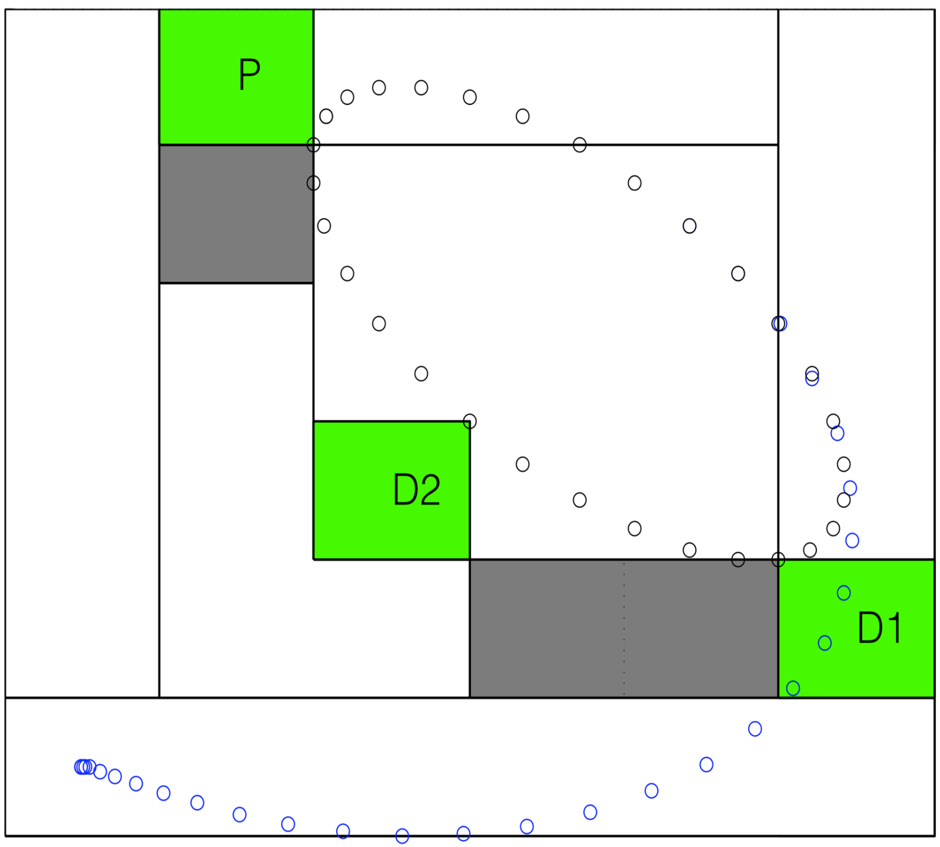
Future Directions

- **Optimal** control of highly **dynamic** systems
  - Automata-guided reachability [IROS13,accepted]
  - Encoding LTL as mixed-integer constraints [ISRR13, sub.]
- **Robust** control for **uncertain** systems

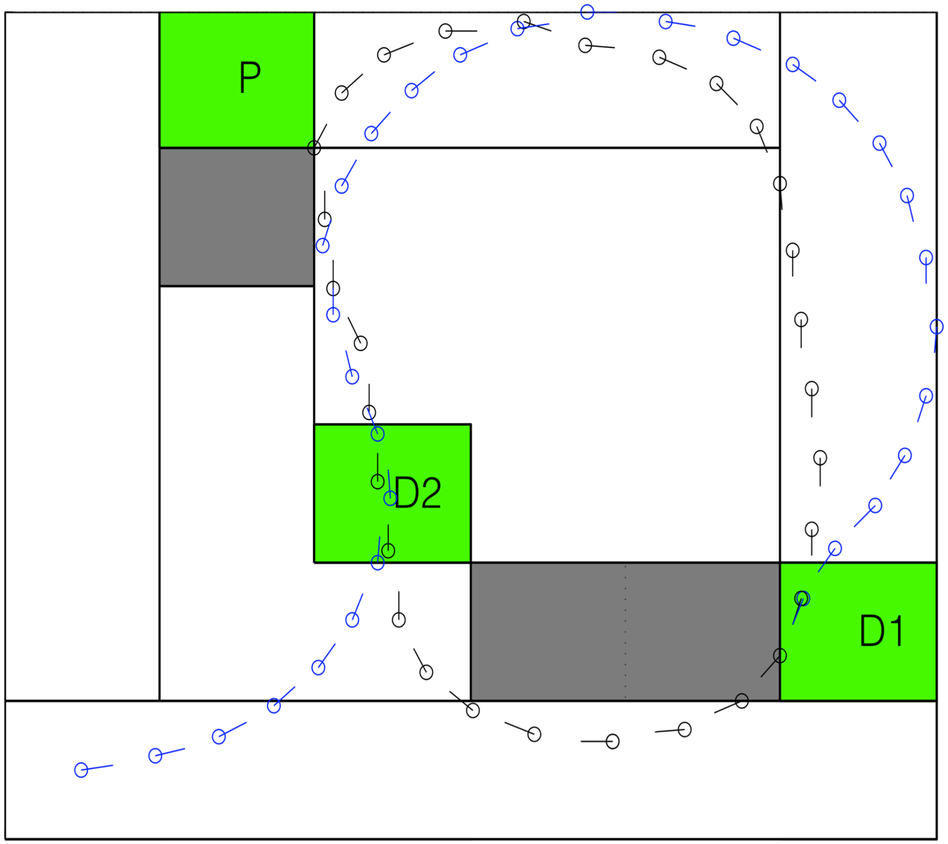


Integrator (20 dim)
Solution in 6 sec.

Future Directions



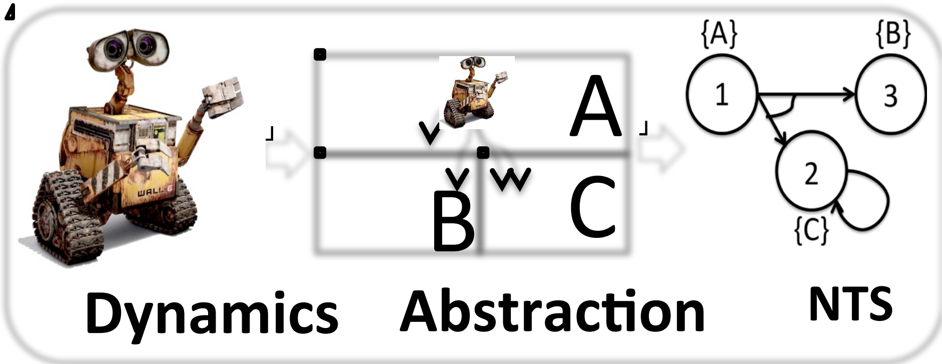
Quadrotor



Aircraft

Conclusions

- Main results
  - **Optimal control** with LTL fragment
  - **Non-deterministic** and stochastic systems
  - Simple and extensible framework



- Future work
  - Receding horizon control
  - Removing fixed-ordering assumption

Thank you!

- Contact: Eric M. Wolff
  - Email: ewolff@caltech.edu
  - Web: www.cds.caltech.edu/~ewolff/
- Funding: NDSEG fellowship, Boeing, AFOSR

References

- R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," Proc. IEEE, 2000.
- R. Alur and S. LaTorre,"Deterministic generators and games for LTL fragments," ACM Trans. Comput. Logic, vol. 5, no. 1, pp. 1–25, 2004.
- C.Belta and L.C.G.J.M.Habets,"Control of a class of nonlinear systems on rectangles," IEEE Trans. on Automatic Control, vol. 51, pp. 1749–1759, 2006,
- R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of Reactive(1) designs," Journal of Computer and System Sciences, vol. 78, pp. 911–938, 2012.
- R. Ehlers, "Generalized Rabin(1) synthesis with applications to robust system synthesis," in NASA Formal Methods, 2011
- L. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," IEEE Trans. on Automatic Control, vol. 51, pp. 938–948, 2006.
- S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in Proc. of IEEE Conf. on Decision and Control, 2009.
- M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," IEEE Trans. on Automatic Control, vol. 53, no. 1, pp. 287–297, 2008.
- O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in STACS 95.
- T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," IEEE Trans. on Automatic Control, 2012