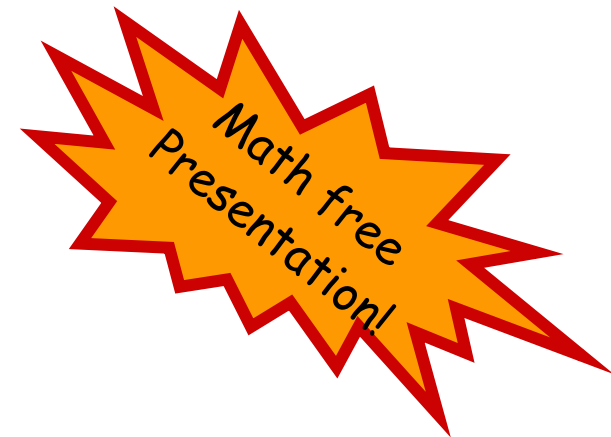


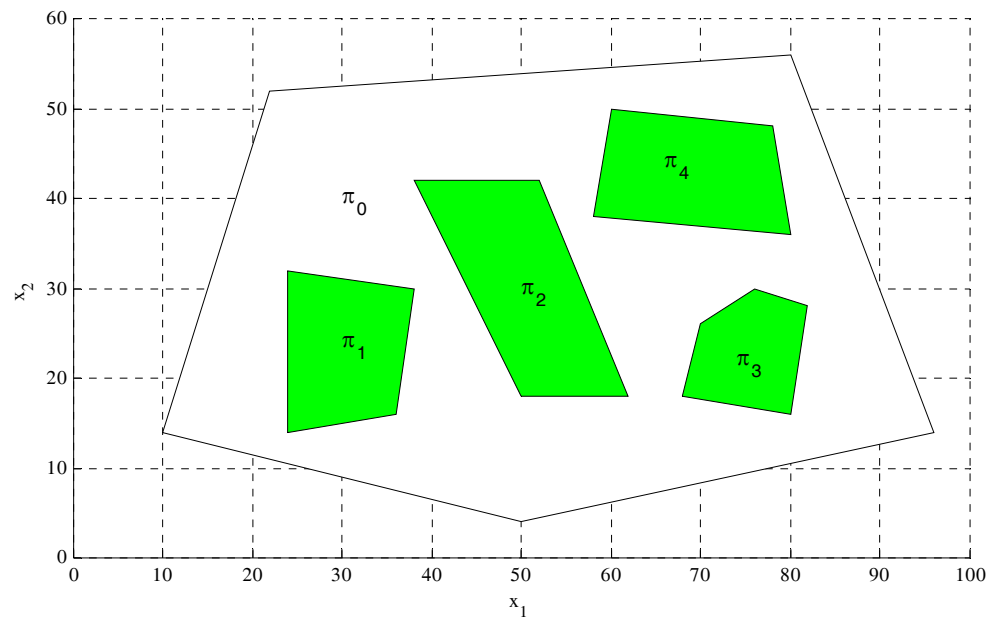
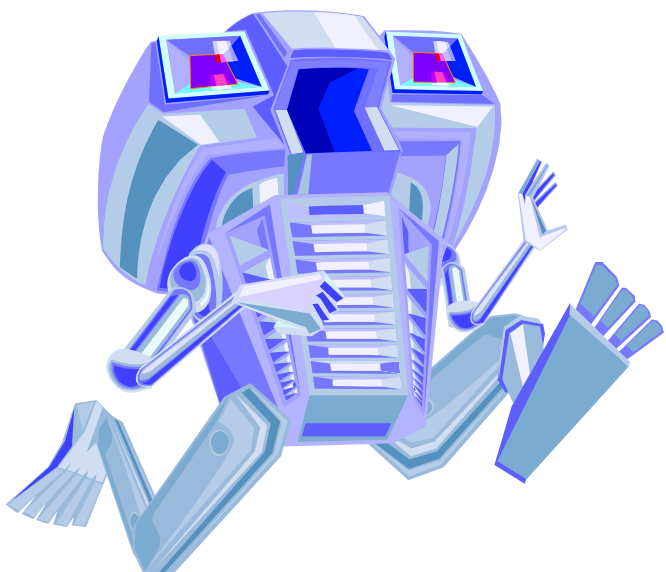
(Towards) *Translating Temporal Logic to Controller Specifications*

Georgios E. Fainekos, Savvas G. Loizou and George J. Pappas



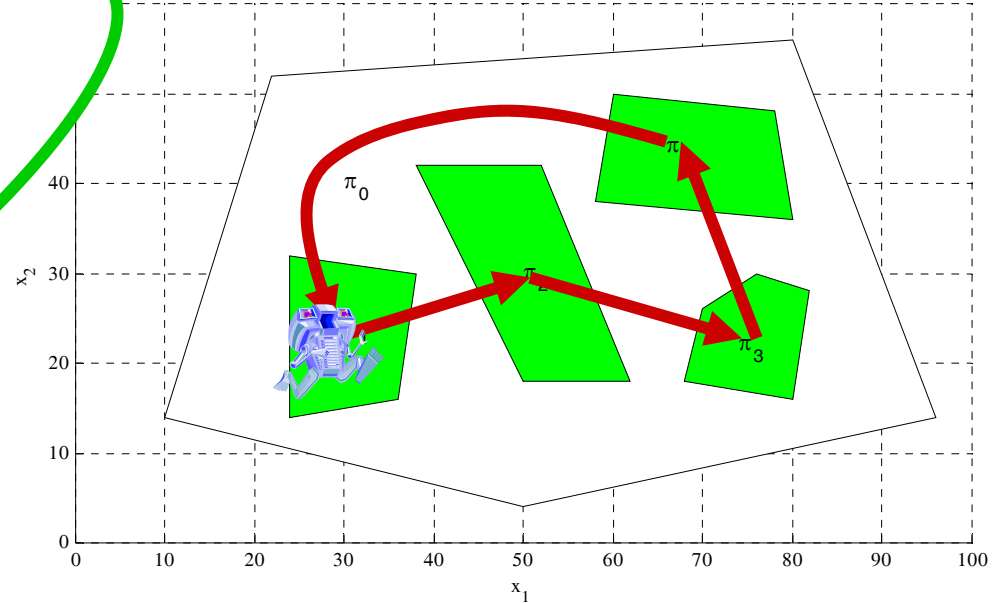
GRASP Lab
Departments of CIS, MEAM and ESE
University of Pennsylvania

Motivation - Motion Planning



Motivation - Motion Planning

SUCCESS



Task: "Stay always in π_0 and visit area π_2 , then area π_3 , then area π_4 and, finally, return to and stay in region π_1 , while avoiding areas π_2 and π_3 "

Good news

RTL: $\square \pi_0 \wedge \diamond (\pi_2 \wedge \diamond (\pi_3 \wedge \diamond (\pi_4 \wedge (\neg \pi_2 \wedge \neg \pi_3) \cup \pi_1)))$

Problem Definition

Temporal Logic Controller Synthesis

Given a dynamical system Σ , a set of initial conditions X_0 and a flat RTL formula φ over Π , construct a closed-loop system in the form of a hybrid automaton H_φ such that the resulting system trajectories $x(t)$ starting at some point $x(0) \in X_0$ satisfy the formula φ .

$$\Sigma: \dot{x}(t) = f(x(t), u(t)) \quad x(t) \in X \subseteq \mathbb{R}^n \quad u(t) \in U \subseteq \mathbb{R}^n$$

In the past ...

Approaches to synthesis of hybrid systems:

- Given controllers + switching rules \implies verify that HS satisfies spec
- Given controllers + spec \implies keep necessary controllers
- Given closed-loop dynamics + spec \implies design switching rules

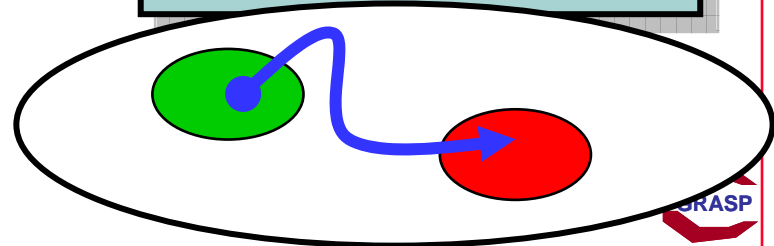
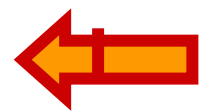
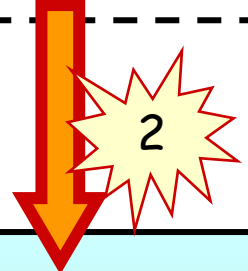
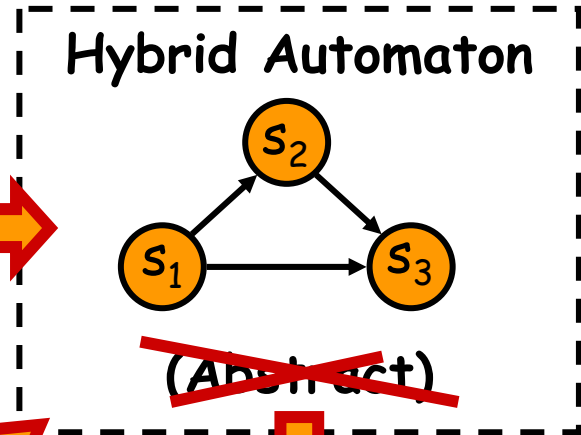
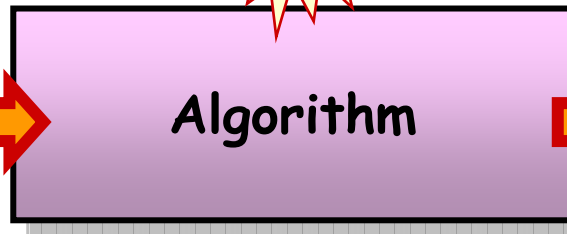
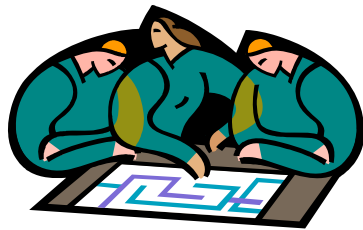
Researchers: Alur, Henzinger, Tabuada, Davoren, Moor, Koutsoukos, Antsaklis, Stiver, Lemmon, Tomlin, Lygeros, Sastry, Habets, van Schuppen, Asarin, Bournez, Dang, Maler, Pnueli, Bemporad, Morari, Giorgetti, Lafferriere, Kloetzer, Belta, Kyriakopoulos, Kress-Gazit, Loizou, Pappas, Fainekos and many others ...

- **Given spec \implies design controllers**

In this presentation ...

A top-down approach:

Input:
Software Specification



Hybrid Automaton

A hybrid automaton is a tuple

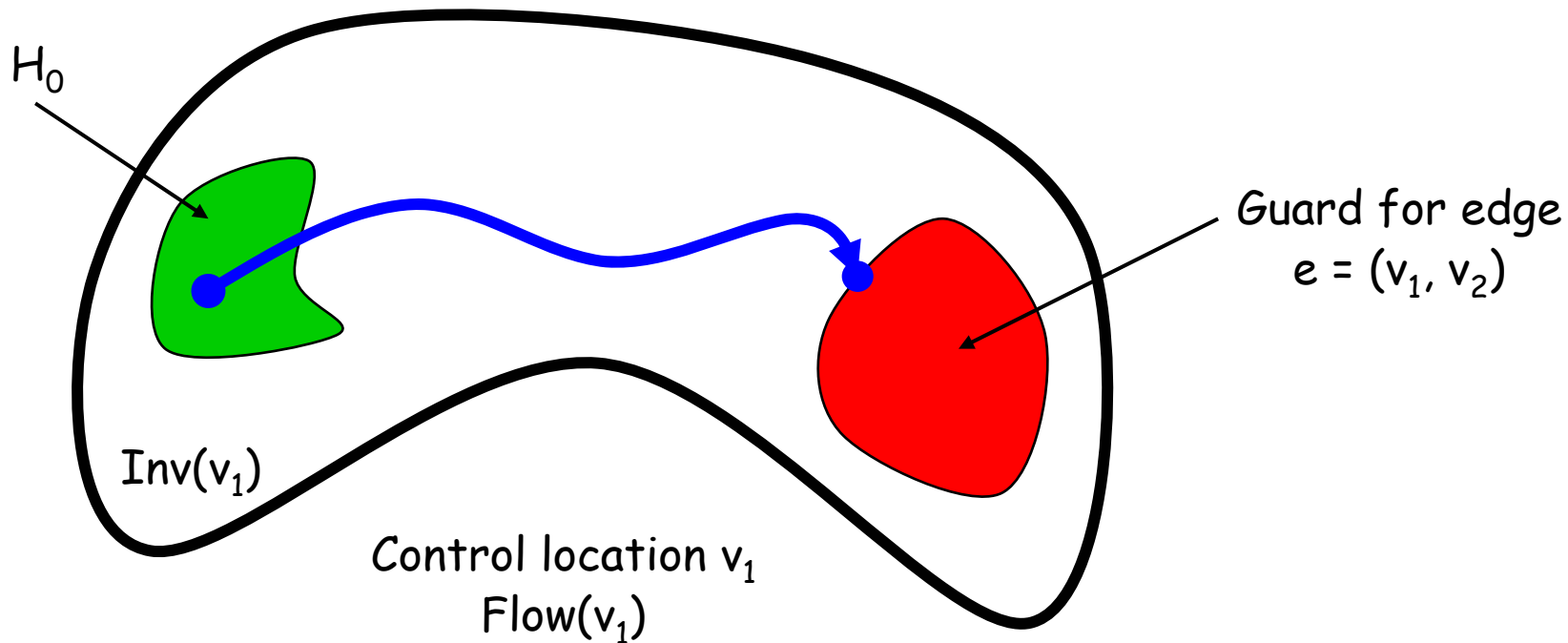
$$H = (X, V, E, \text{Inv}, \text{Flow}, \text{Init}, \text{Guard}, F)$$

where

- X is the state space of the system Σ
- V is the set of control locations,
- $E \subseteq V \times V$ is the set of control switches
- $\text{Inv} : V \rightarrow P(X)$ assigns an invariant set to each location
- $\text{Flow} : V \times X \rightarrow P(\mathbb{R}^n)$ constraints the time derivative of the continuous part of the state
- $\text{Init} : V \rightarrow P(X)$ assigns to each control location a set of initial conditions,
- $\text{Guard} : E \rightarrow P(X)$ is the guard condition that enables a control switch e in E
- $F \subseteq V$ is the set of final locations

Trajectories of the Hybrid Automaton

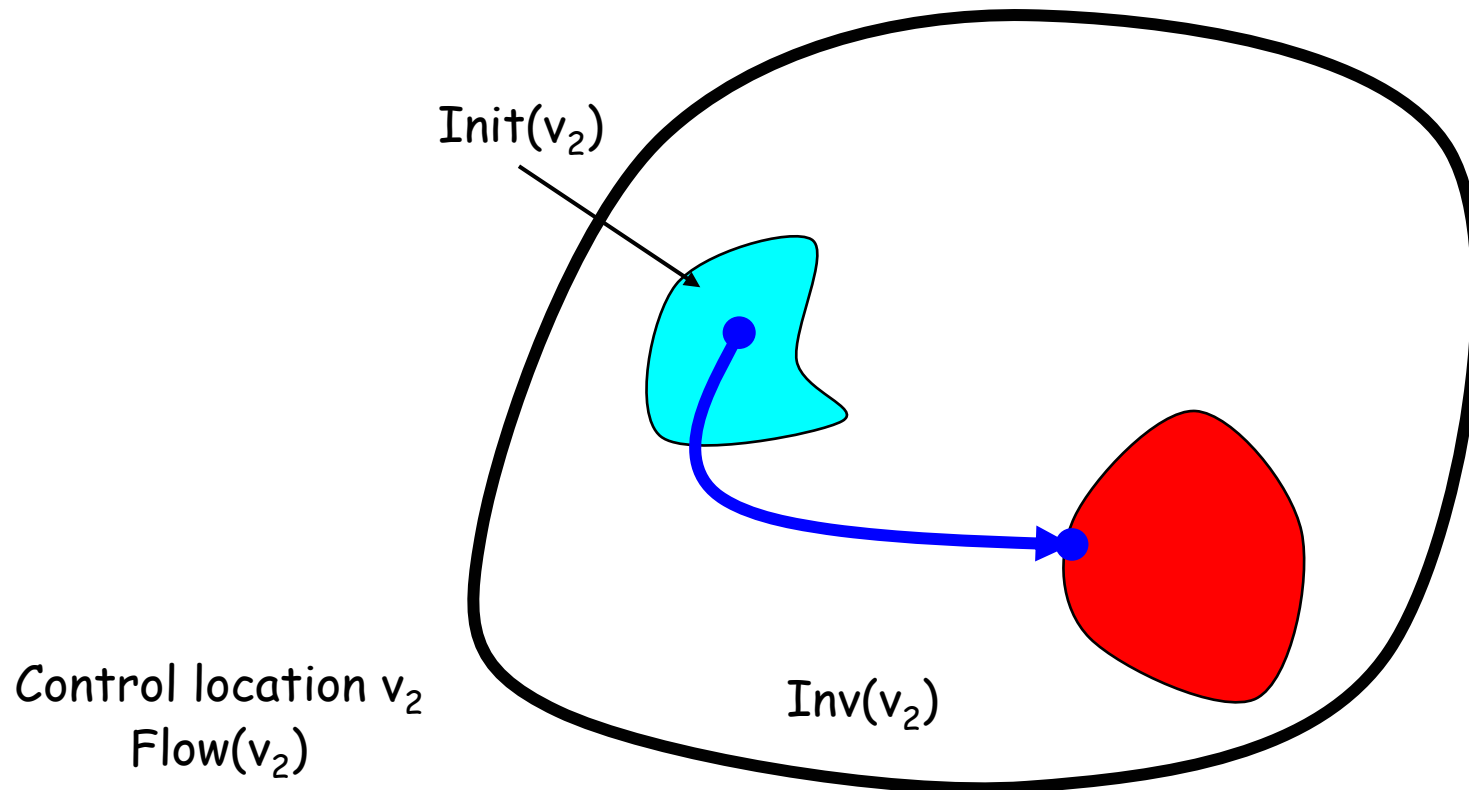
Formally, the semantics of a hybrid automaton are given in terms of timed transition systems $T_H = (H, H_0, \rightarrow)$.



Note: the reset map is the identity, transitions are forced

Trajectories of the Hybrid Automaton

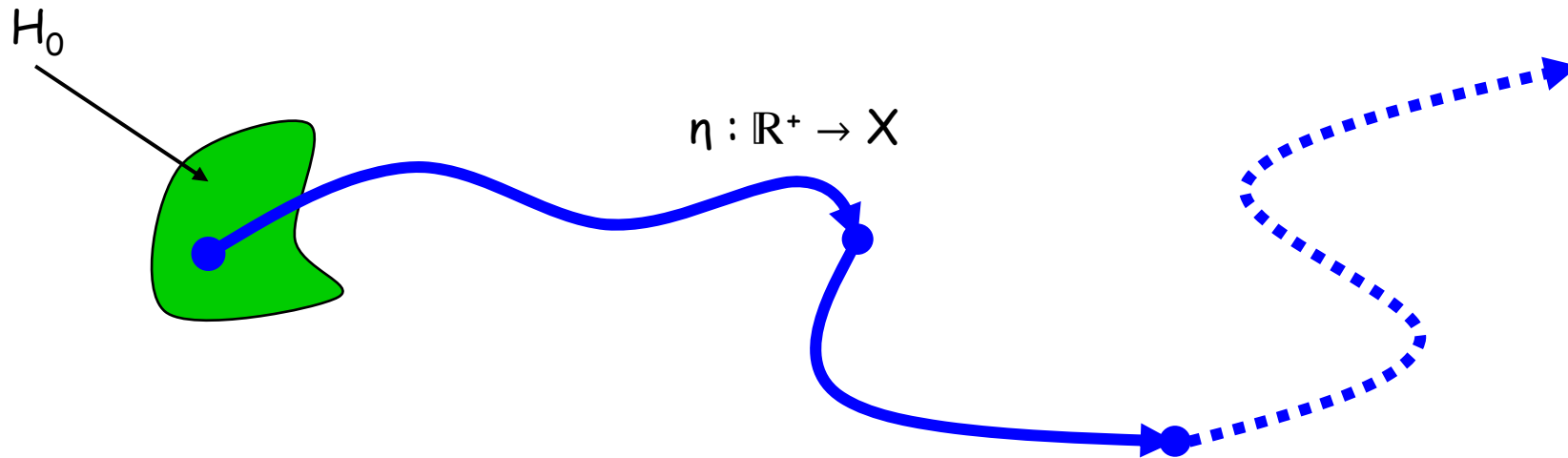
Formally, the semantics of a hybrid automaton are given in terms of timed transition systems $T_H = (H, H_0, \rightarrow)$.



Note: the reset map is the identity, transitions are forced

Language of the timed transition system

The set of all trajectories η of T_H starting from a state in H_0 is the language $L(T_H)$ of the timed transition system T_H .



Linear Temporal Logic: Continuous Time

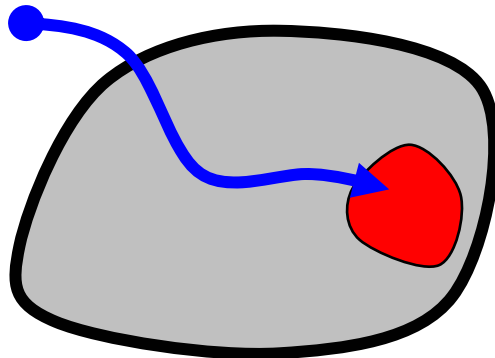
Like propositional logic, but also reasoning wrt time ...

Basic operators:

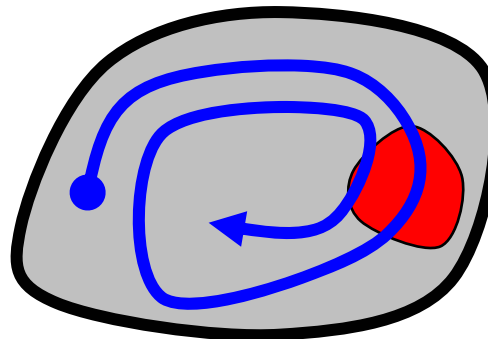
Propositional: \wedge , \vee , \neg

Temporal: \diamond , \square , U , R

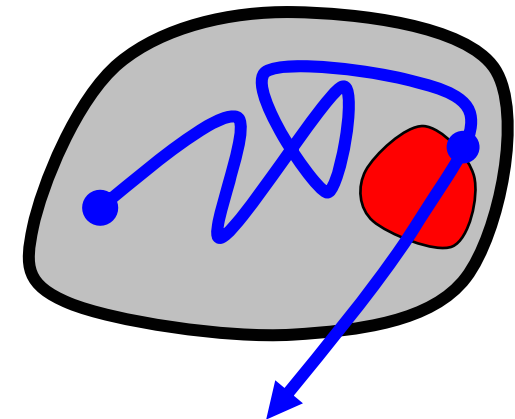
$\diamond(\text{red})$
Eventually red



$\square(\text{gray})$
Always gray



$(\text{gray}) U (\text{red})$
gray Until red



Flat LTL with continuous time semantics

Let Π be a set of atomic propositions and Π^* be the set of Boolean combinations of atomic propositions, i.e. $(\pi_1 \wedge \pi_2) \vee \neg \pi_3$.

Define the denotation $[[\cdot]] : \Pi \rightarrow P(X)$ which extends naturally over Π^* as:

$$[[\pi_1 \wedge \pi_2]] = [[\pi_1]] \cap [[\pi_2]] \text{ and } [[\neg \pi]] = [[\pi]]^c$$

Syntax in NNF: $\varphi ::= \pi^* \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \pi^* U \varphi_2 \mid \pi^* R \varphi_2$

Semantics:

$$\eta \models \pi^* \text{ iff } \eta(0) \in [[\pi^*]]$$

$$\eta \models \varphi_1 \wedge \varphi_2 \text{ iff } \eta \models \varphi_1 \text{ and } \eta \models \varphi_2$$

$$\eta \models \varphi_1 \vee \varphi_2 \text{ iff } \eta \models \varphi_1 \text{ or } \eta \models \varphi_2$$

$$\eta \models \pi^* U \varphi \text{ iff } \exists t \geq 0 \text{ s.t. } \eta|_t \models \varphi \text{ and } \forall s \in [0, t] \eta(s) \in [[\pi^*]]$$

$$\eta \models \pi^* R \varphi \text{ iff } \forall t \geq 0 \eta|_t \models \varphi \text{ or } \exists s \in [0, t] \text{ s.t. } \eta(s) \in [[\pi^*]]$$

Some notation: $\eta|_t(s) = \eta(t+s)$ $LTL(op_1, \dots, op_n)$

$$\diamond \varphi = TU \varphi \qquad \square \varphi = FR \varphi$$

LTL(\cup, \vee, \wedge) to HA

1

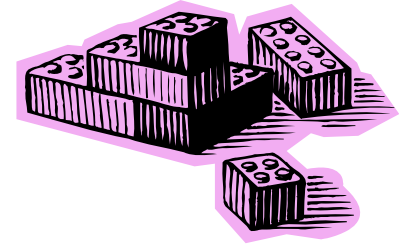
Algorithm

Modular construction using the structure of φ .
(\equiv proof by induction)

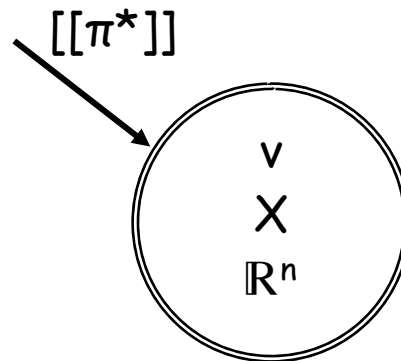
Syntax in NNF: $\varphi ::= \pi^* \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \pi^* \cup \varphi$

Ex. $\diamond(\pi_2 \wedge \diamond(\pi_3 \wedge \diamond(\pi_4 \wedge (\neg \pi_2 \wedge \neg \pi_3) \cup \pi_1)))$

LTL(U, V, \wedge) to HA

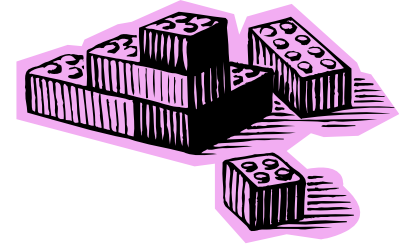


Proposition 2 (base case): Let $\varphi = \pi$, then
 $H_{\pi^*} = (X, \{v\}, \emptyset, X, \mathbb{R}^n, [[\pi^*]], \emptyset, \{v\})$



proof immediate from definition: $\eta \models \pi^*$ iff $\eta(0) \in [[\pi^*]]$

Basic Building Blocks



$$\varphi = \varphi_1 \wedge \varphi_2 \text{ or } \varphi = \varphi_1 \vee \varphi_2$$

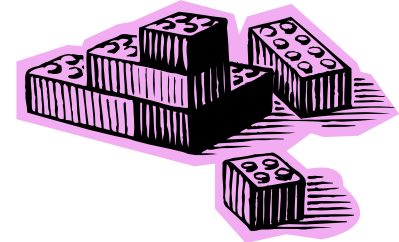
Proposition 1 [Henzinger 95]:

Let H_1 and H_2 be hybrid automata.

If $H = H_1 \cup H_2$, then $L(T_H) = L(T_{H_1}) \cup L(T_{H_2})$.

If $H = H_1 \cap H_2$, then $L(T_H) = L(T_{H_1}) \cap L(T_{H_2})$.

LTL(U, V, \wedge) to HA



Proposition 3: Let $\varphi = \pi^* U \psi$, then

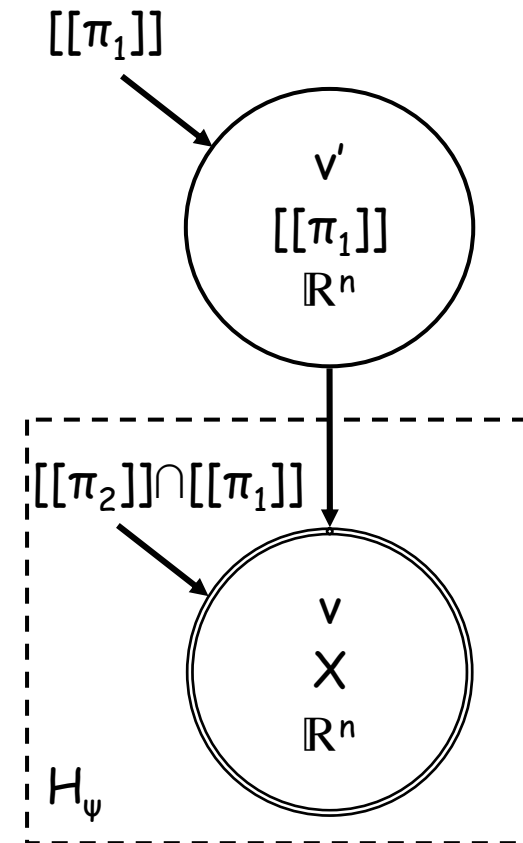
$H_\varphi = (X, V \cup \{v'\}, E', \text{Inv}', \text{Flow}', \text{Init}', \text{Guard}', F)$

Let $H_\psi = (X, V, E, \text{Inv}, \text{Flow}, \text{Init}, \text{Guard}, F)$. Then:

- ❖ $\text{Inv}'(v') = \text{Init}'(v') = [[\pi^*]]$ and $\text{Flow}(v') = \mathbb{R}^n$
- ❖ $\text{Inv}'(v) = \text{Inv}(v)$ for all $v \in V$
- ❖ $\text{Flow}'(v) = \text{Flow}(v)$ for all $v \in V$
- ❖ $\text{Init}'(v) = \text{Init}(v) \cap [[\pi^*]]$ for all $v \in V$
- ❖ $E' = E \cup \{(v', v) \mid v \in V_{\text{in}}\}$
- ❖ $\text{Guard}'(v', v) = \text{Init}'(v)$ for all $v \in V_{\text{in}}$
- ❖ $\text{Guard}'(e) = \text{Guard}(e)$ for all $e \in E$

where $V_{\text{in}} = \{v \in V \mid \text{Init}(v) \cap [[\pi^*]] \neq \emptyset\}$

Ex. $\pi_1 U \pi_2$



proof immediate from definition:

$\eta \models \pi^* U \psi$ iff $\exists t \geq 0$ s.t. $\eta \upharpoonright_t \models \psi$ and $\forall s \in [0, t] \eta(s) \in [[\pi^*]]$

LTL($\mathcal{U}, \vee, \wedge$) to HA

Algorithm 1 The LTL($\mathcal{U}, \wedge, \vee$) Fragment

Input: A formula $\phi \in \text{LTL}(\mathcal{U}, \wedge, \vee)$

Output: The hybrid automaton \mathcal{H}_ϕ

```
1: procedure LTL $\mathcal{U}$ TOHA( $\phi$ )
2:   if  $\phi = \bar{\pi}$  then
3:     return  $\mathcal{H}_{\bar{\pi}}$  ▷ Proposition 2
4:   else if  $\phi = \phi_1 \wedge \phi_2$  then
5:     return LTL $\mathcal{U}$ TOHA( $\phi_1$ )  $\cap$  LTL $\mathcal{U}$ TOHA( $\phi_2$ )
6:   else if  $\phi = \phi_1 \vee \phi_2$  then
7:     return LTL $\mathcal{U}$ TOHA( $\phi_1$ )  $\cup$  LTL $\mathcal{U}$ TOHA( $\phi_2$ )
8:   else if  $\phi = \bar{\pi}\mathcal{U}\psi$  then
9:      $\mathcal{H}_\psi \leftarrow$  LTL $\mathcal{U}$ TOHA( $\psi$ )
10:    return  $\mathcal{H}_\phi$  ▷ Proposition 3
11:  end if
12: end procedure
```

LTL(U, V, \wedge) to HA

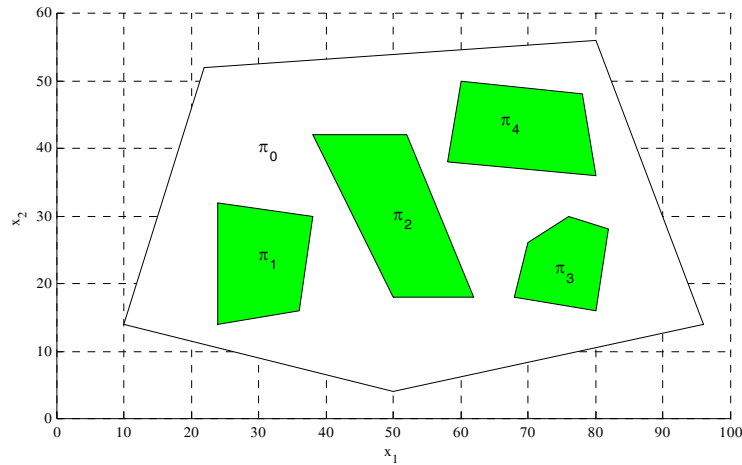
Remarks:

- ☞ "Fairness conditions"
- ☞ Computational complexity
 - ☞ Size of $H1 \cap H2$: $O(n_1 n_2)$
 - ☞ Size of final automaton $O(\exp(|\varphi|))$
 - ☞ Set intersections and complementation
 - ☞ set theoretic operations, not reachability etc.
- ☞ Note: graph is acyclic
- ☞ Final hybrid automaton might have empty guards and invariant sets
 - ☞ Then formula might be unsatisfiable

LTL_{U,□} fragment

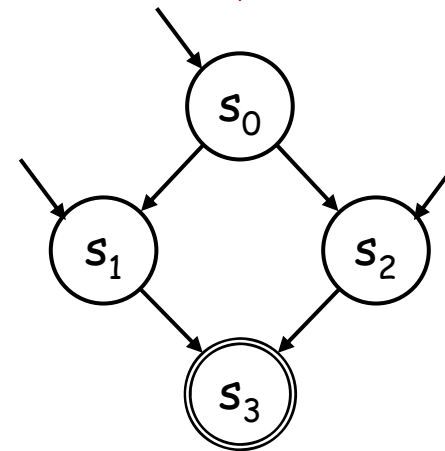
- Similar to LTL(U,V,∧) we can get an algorithm for LTL(□,V,∧)
- General solution for $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$
 - ◆ with $\varphi_1 \in \text{LTL}(U, V, \wedge)$ and $\varphi_2 \in \text{LTL}(\square, V, \wedge)$
- More general solution for any formula:
 - ◆ $\varphi ::= \pi^* \mid \varphi_\square \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \pi^* \cup \varphi$
 - ◆ $\varphi_\square \in \text{LTL}(\square, V, \wedge)$
 - ◆ Ex. $\square \pi_0 \wedge \diamond (\pi_2 \wedge \diamond (\pi_3 \wedge \diamond (\pi_4 \wedge (\neg \pi_2 \wedge \neg \pi_3) \cup \pi_1)))$

"Abstract" Hybrid Automaton

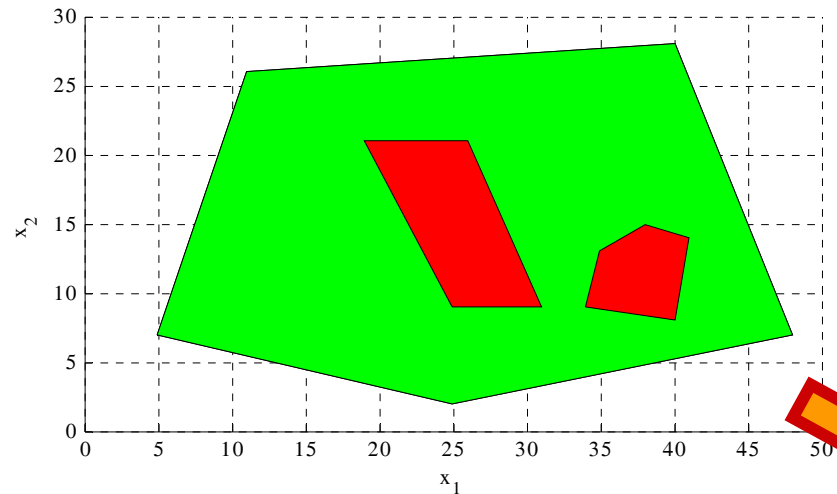


Ex. $\square \pi_0 \wedge \diamond \pi_2 \wedge \diamond \pi_3$

Algorithm

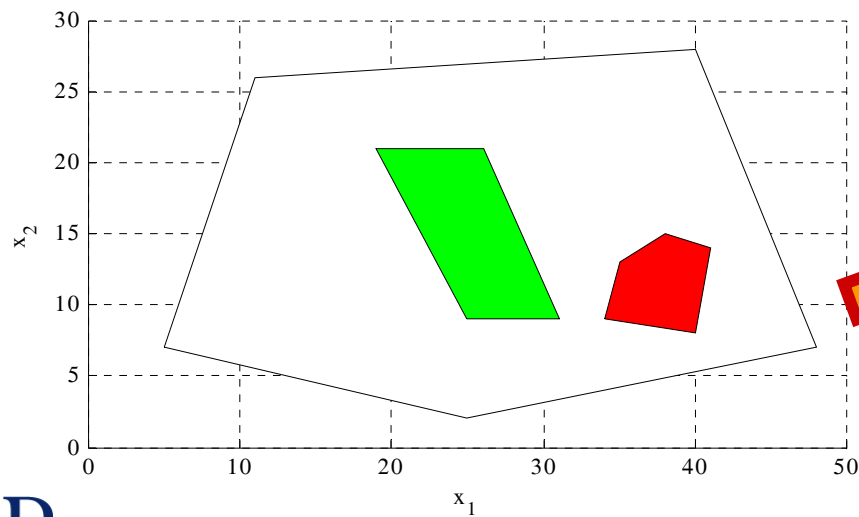


"Abstract" Hybrid Automaton



Ex. $\square \pi_0 \wedge \diamond \pi_2 \wedge \diamond \pi_3$

Algorithm



s_0

s_1

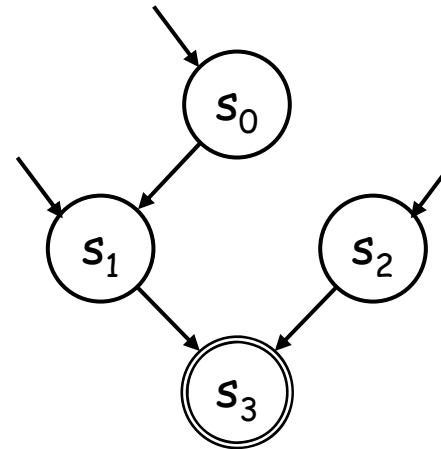
s_2

s_3

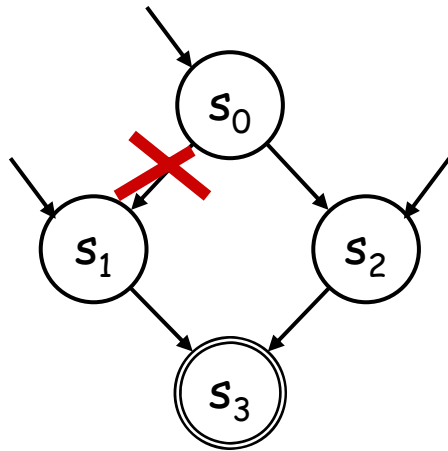
A practical solution

Backward reachability using
Breadth First Search:

- ✓ fix initial conditions
- ✓ one outgoing edge from each control location

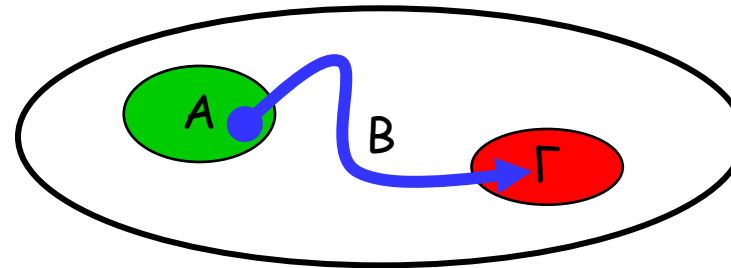


2 Derive controller constraints



Feedback control law $g_c : X \rightarrow U$
 Controller constraints $c = \{A, B, \Gamma\}$

- A initial conditions, i.e. $x(0) \in A$
- Γ final conditions, i.e. $x(t_g) \in \Gamma$
- B invariant, $\forall t \in [0, t_g]. x(t) \in B$



$$c_3 = \{ \bigcup_{e \in I_3} \text{Guard}(e), \text{Inv}(s_3), \emptyset \} \implies g_{c3}$$

$$c_1 = \{ \text{Guard}(e_{I1}) \cup \text{Init}(s_1), \text{Inv}(s_1), \text{Guard}(e_{O1}) \} \implies g_{c1}$$

$$\rightarrow c_2 = \{ \text{Init}(s_2), \text{Inv}(s_2), \text{Guard}(e_{O2}) \} \implies g_{c2}$$

~~$$c_0 = \{ \text{Init}(s_0), \text{Inv}(s_0), \text{Guard}(e_{O1}) \} \implies g_{c0}$$~~

$$c_0 = \{ \text{Init}(s_0), \text{Inv}(s_0), \text{Guard}(e_{O1}) \} \implies g_{c0}$$

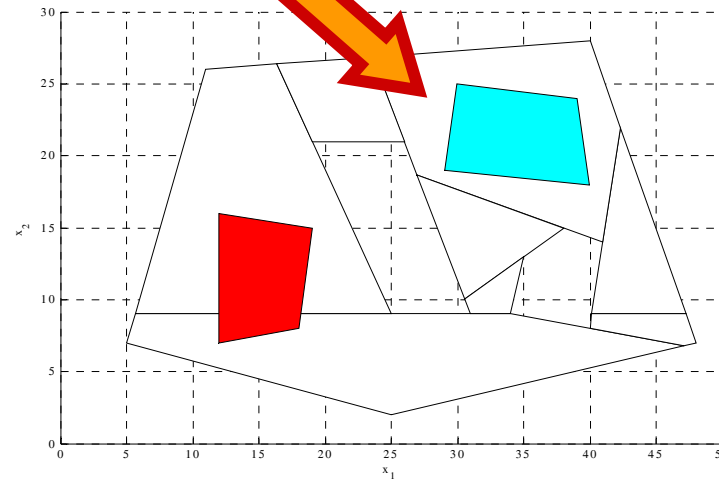
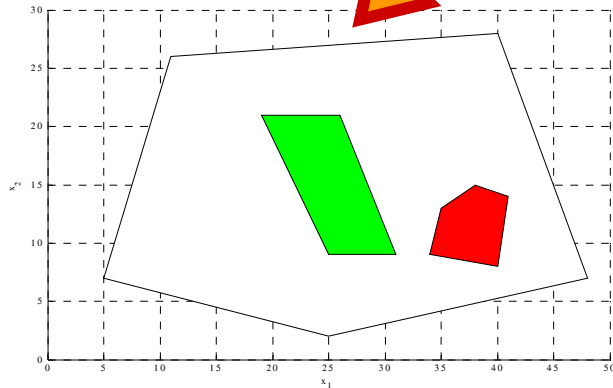
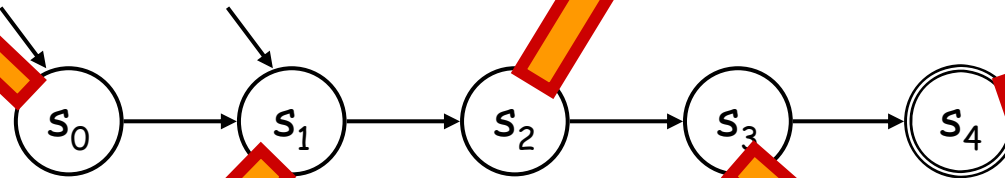
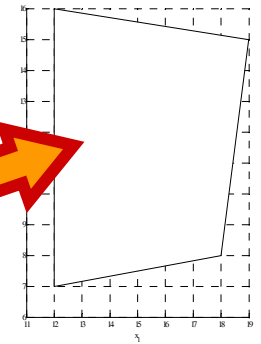
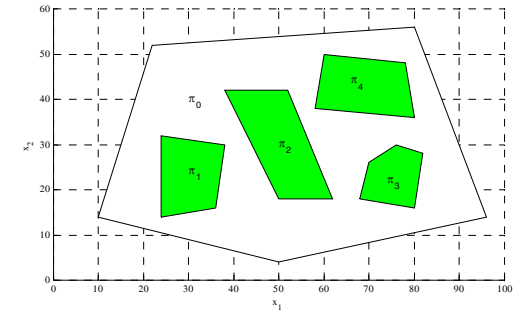
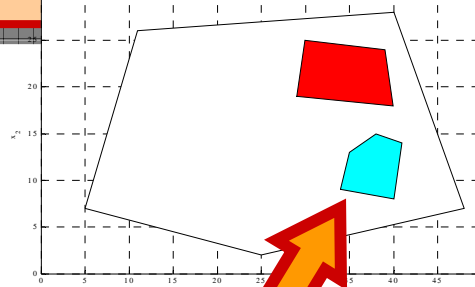
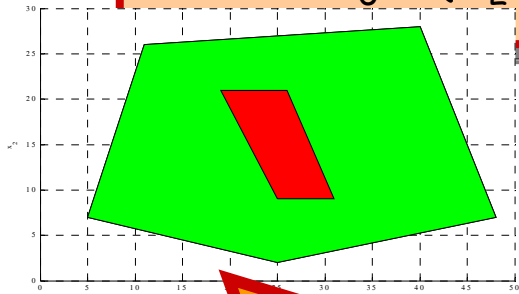
Main result

Let $\varphi \in \text{LTL}_{U, \square}$ be satisfiable wrt to our model of computation. Then:

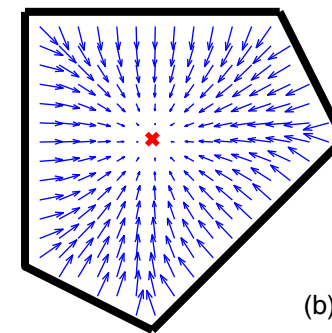
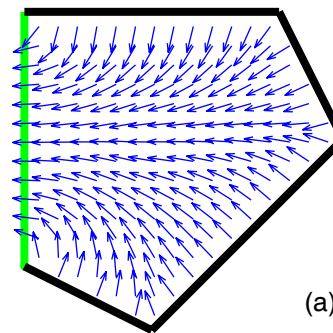
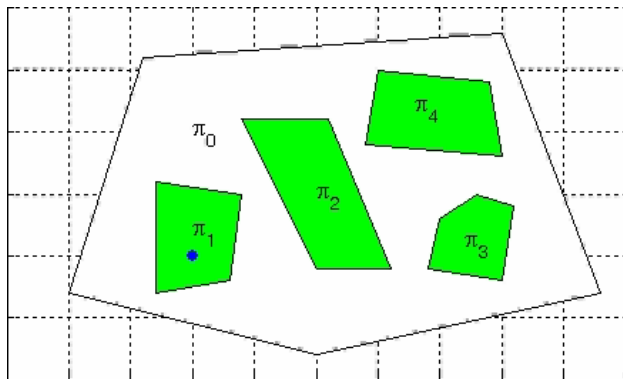
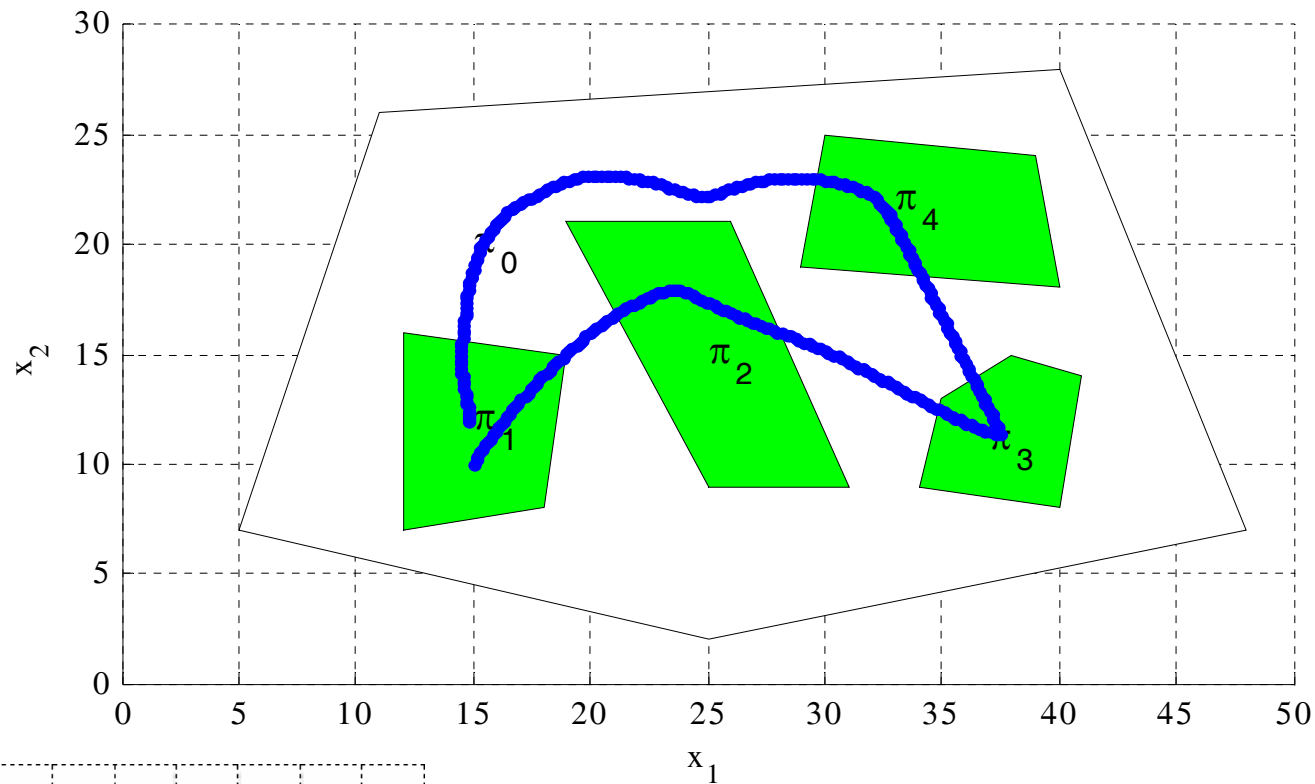
- (i) φ can be converted into list of controller specifications.
- (ii) The hybrid automaton H resulting from the closed-loop dynamics satisfies: $\forall \eta \in L(H) . \eta \models \varphi$

Back to the toy example ...

RTL: $\square \pi_0 \wedge \diamond (\pi_2 \wedge \diamond (\pi_3 \wedge \diamond (\pi_4 \wedge (\neg \pi_2 \wedge \neg \pi_3) \cup \pi_1)))$



Back to the toy example ...



Conclusions

- ✓ Top-down approach for the synthesis of HA from temporal logic
 - ✓ identified a fragment of LTL that provides a modular construction
 - ✓ algorithm is based on the closure properties of HA
- ✓ Advantages of the proposed framework:
 - ✓ Clean separation between software specification and controller design
 - ✓ Potentially smaller number of required controllers
 - ✓ The closed-loop dynamics can be designed using different methodology in each control location
 - ✓ "Real" continuous time semantics
 - ✓ Ability to distinguish between events that must hold at a point in time or for an interval

Future work

- ☑ Introduce robustness into the system
 - ☑ Fainekos, Girard, Pappas: *Hierarchical Synthesis of Hybrid Controllers from Temporal Logic Specifications*, HSCC 2007 (to appear)
- ☐ Complete the framework for the full LTL
 - ☐ i.e. add "liveness"
- ☐ Design for distributed specifications
- ☐ Built a complete toolbox with libraries of controllers

Thank You!

Any Question(s) ?

