Temporal logic motion planning for mobile robots

Georgios E. Fainekos, Hadas Kress-Gazit and George J. Pappas

GRASP Laboratory
Departments of CIS and ESE University of Pennsylvania

{fainekos,hadaskg,pappasg}@grasp.upenn.edu

http://www.seas.upenn.edu/~fainekos/

ICRA 2005

Penn
University of Pennsylvania

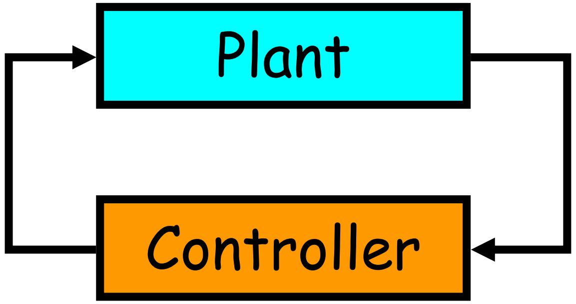
GRASP

Why temporal logic for motion planning?

How do we navigate a robot (*even with simple dynamics*)
in a (*complicated*) environment???

Continuous

- ✓ Design Controller
- ✓ Verify



Discrete

- ✓Discretize environment
- ✓Ignore robot dynamics

Complex dynamics but …
NO complicated environments!

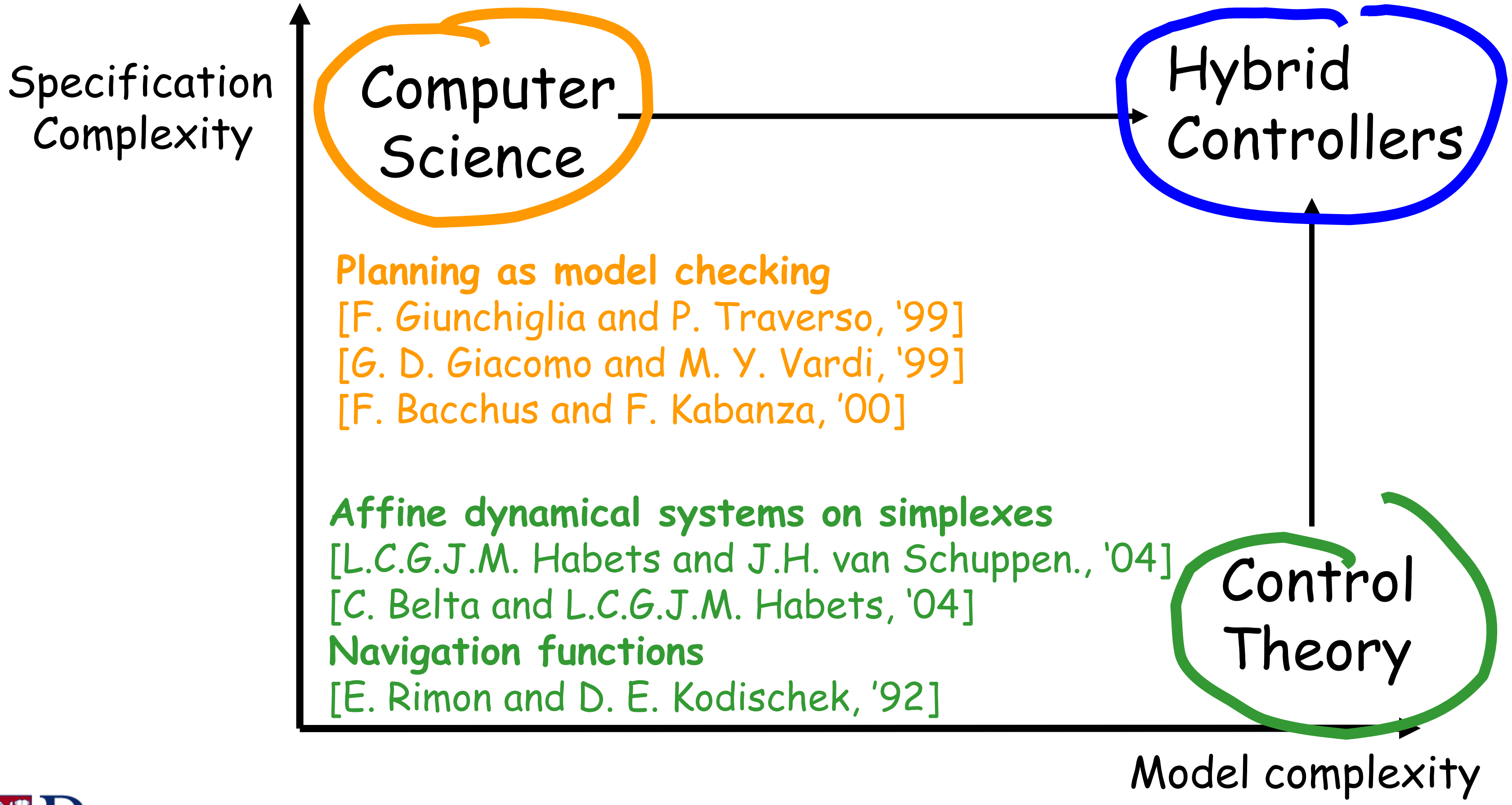
Complicated world but …
Can robot do it??

Can we combine the two approaches? Yes

Spatial and temporal specifications?
Use Temporal Logics

Control and computer science

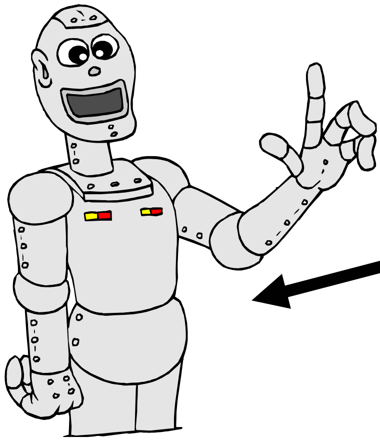
G. E. Fainekos, H. Kress-Gazit and G. J. Pappas, Temporal Logic Motion Planning for Mobile Robots, to appear in the Proceedings of the International Conference on Robotics and Automation, Barcelona, April 2005



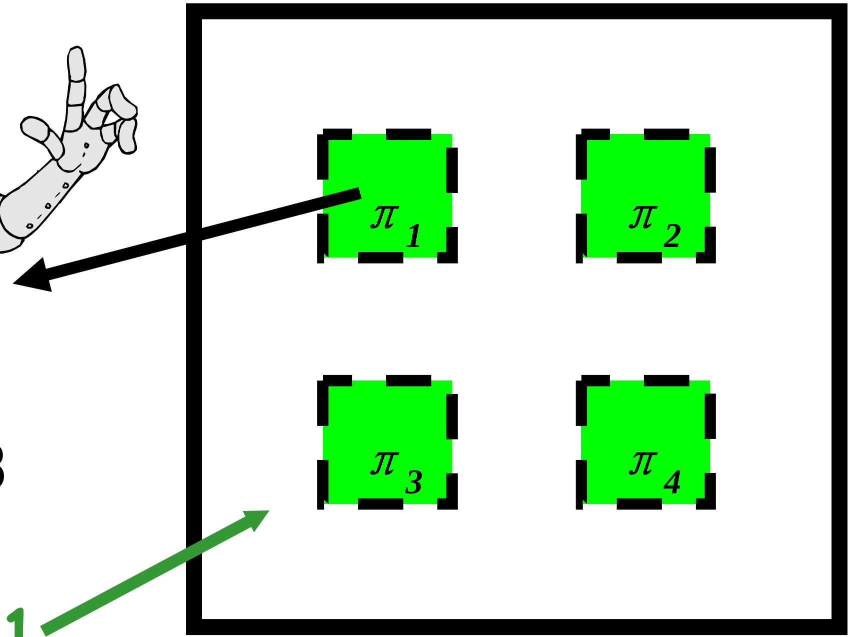
A simple example to guide us through …

Consider a robot that is moving in a square environment with four areas of interest denoted by π_1, π_2, π_3 and π_4. Initially, the robot is placed somewhere in the region labeled by π_1. The desired Specification for the robot given in natural language is: "Visit area π_2 then area π_3 then area π_4 and, finally, return to region π_1 while avoiding areas π_2 and π_3"

Input 2: "Visit area π_2 then area π_3 then area π_4 and, finally, return to region π_1 while avoiding areas π_2 and π_3"



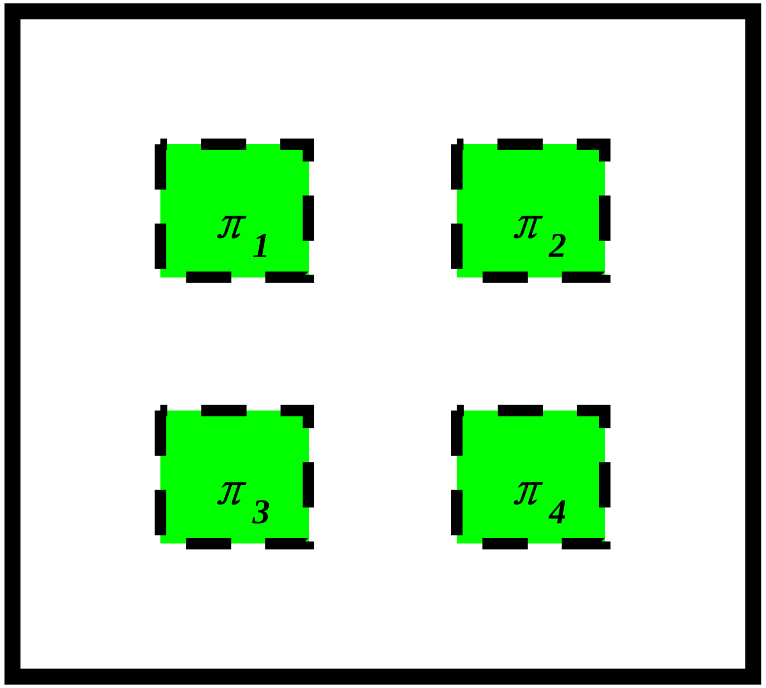
Input 3



Output: A hybrid controller that satisfies the specification by construction.

Input 1

What can we express with temporal logics?



Go to goal (reachability)

$$\varphi = \Diamond \pi_2$$

Coverage

$$\varphi = \Diamond \pi_2 \wedge \Diamond \pi_3 \wedge \Diamond \pi_4$$

Sequencing

$$\varphi = \Diamond(\pi_2 \wedge \Diamond \pi_3)$$

Reachability with avoidance

$$\varphi = \neg(\pi_2 \vee \pi_3) U \pi_4$$

Recurrent Sequencing

$$\varphi = [] \Diamond(\pi_2 \wedge \Diamond \pi_3)$$

The simple example: "Visit area π_2 then area π_3 then area π_4 and, finally, return to region π_1 while avoiding areas π_2 and π_3"

$$\varphi = \Diamond(\pi_2 \wedge \Diamond(\pi_3 \wedge \Diamond(\pi_4 \wedge (\neg\pi_2 \wedge \neg\pi_3) U \pi_1)))$$

Penn University of Pennsylvania

GRASP

Problem formulation

Model: We consider a fully actuated, planar model of robot motion operating in a polygonal environment *P*. The motion of the robot is expressed as:

$$\dot{x}(t) = u(t) \quad x(t) \in P \subseteq R^2 \quad u(t) \in U \subseteq R^2$$

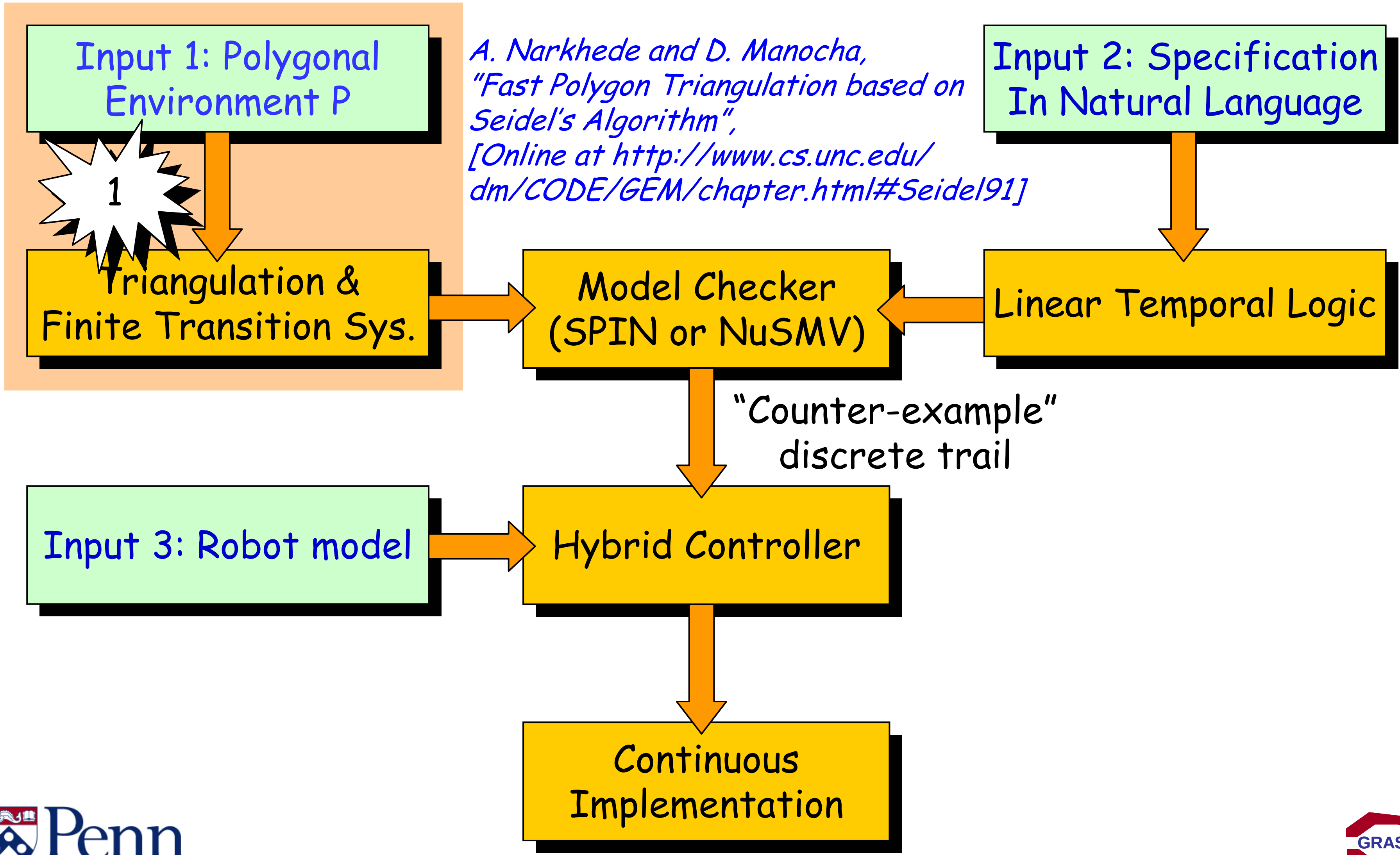
Specification: A linear temporal logic (LTL_{-X}) formula φ that captures the robots' desired behavior.

Problem: Given robot model, environment P, initial condition x(0), and an LTL_{-X} temporal logic formula φ, find control input u(t) such that x(t) satisfies φ.

Penn
UNIVERSITY of PENNSYLVANIA

GRASP

Overview of the Algorithm for Open-Loop Temporal Logic Motion Planning



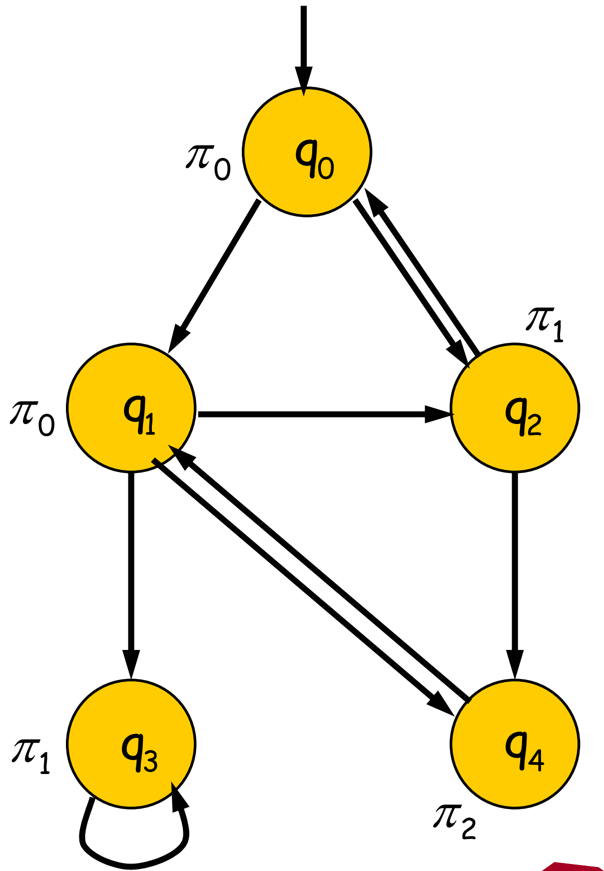
Finite Transition Systems (FTS)

A transition system

$$D = (Q, q_0, \rightarrow, \Pi, h_D)$$

consists of

- A set of states Q
- An initial state $q_0 \in Q$
- The transition relation $q_i \rightarrow q_j$
- A set of observations Π
- The observation map $h_D(q_i) = \pi_k$

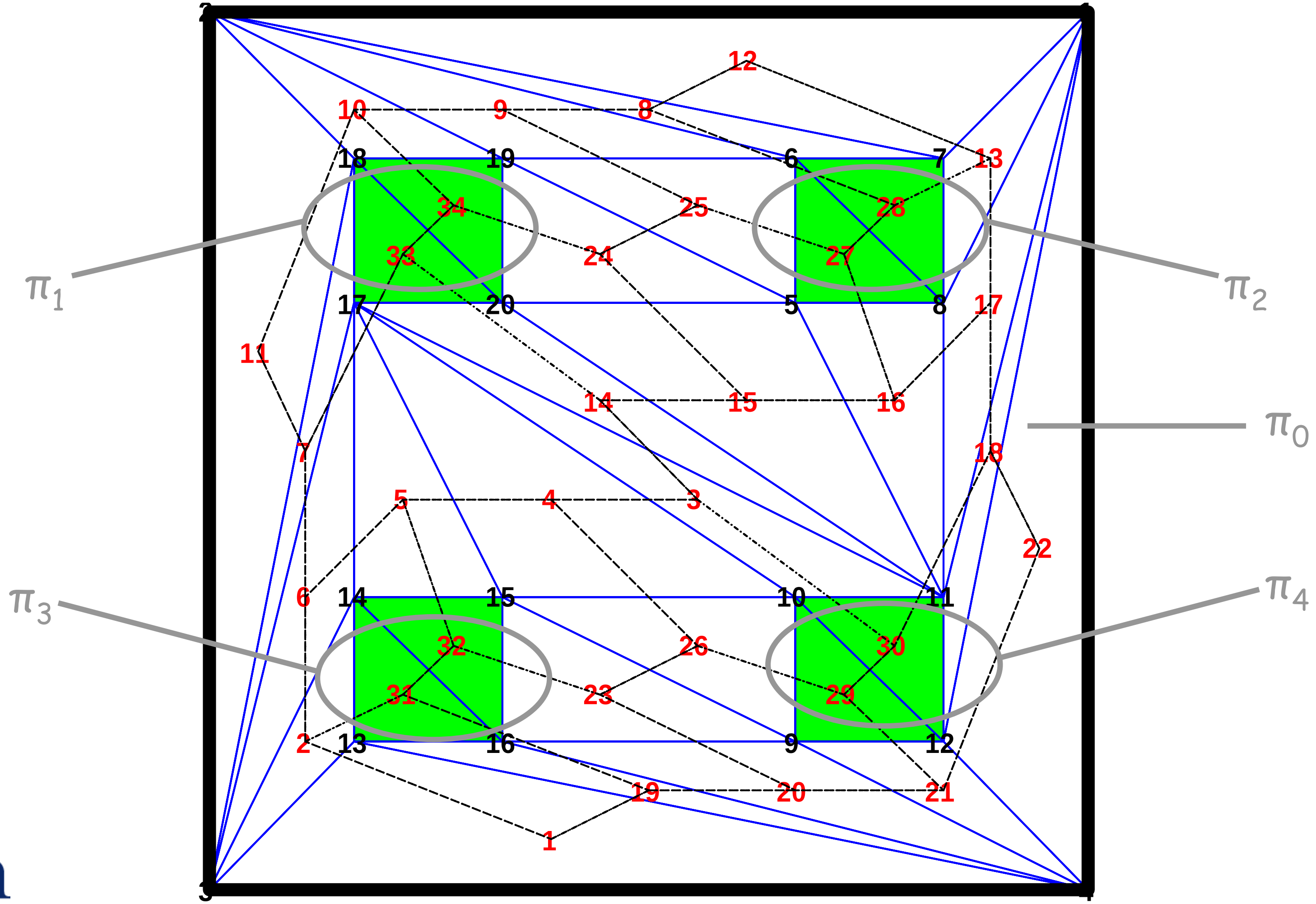


The language L(D) of D is the set of all the sequences of observations i.e. $s = \pi_0\pi_1\pi_2\pi_0 \in L(D)$

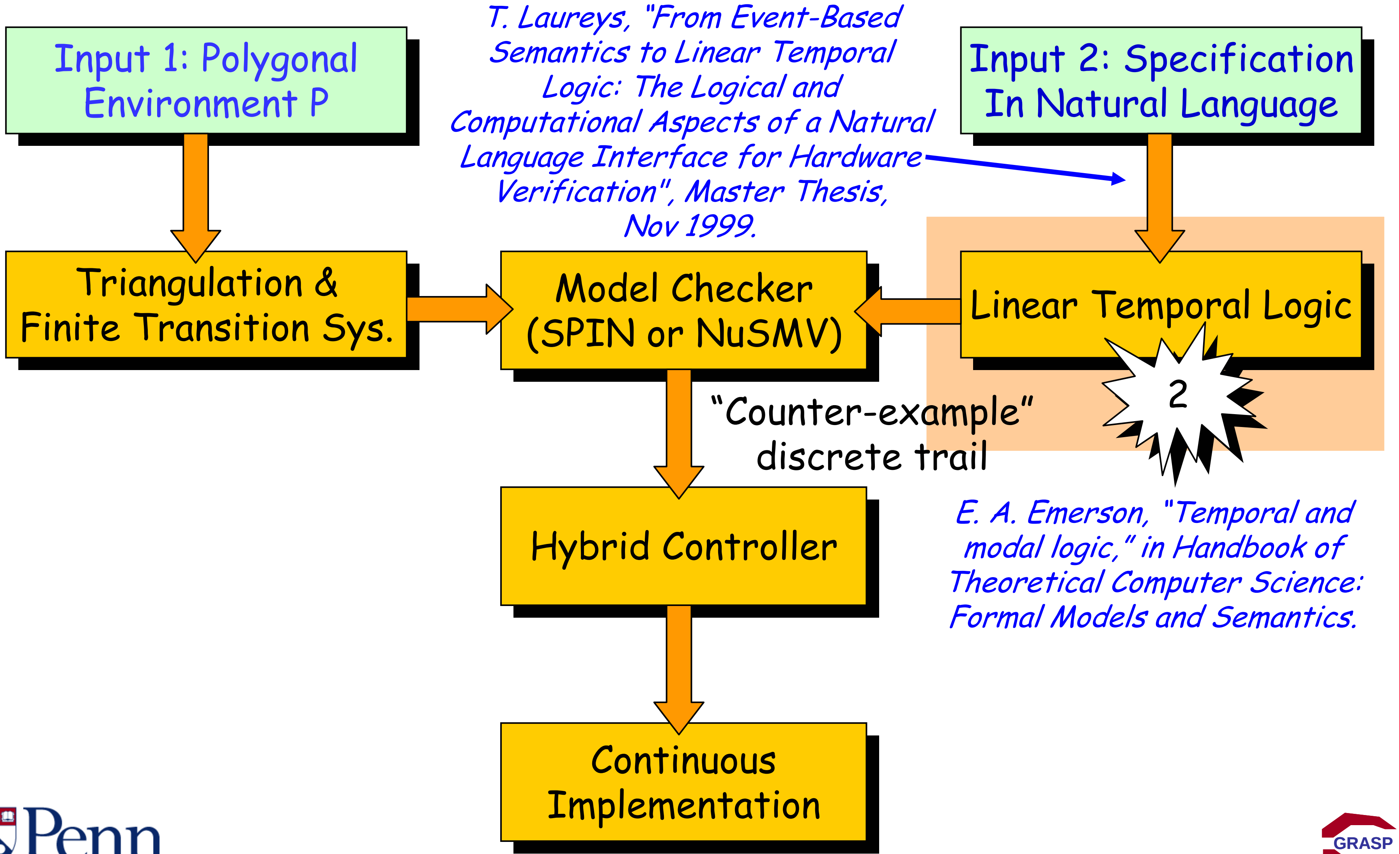
Discrete abstraction by triangulation

Partition the environment, Obtain discrete abstraction (FTS)

Ensure that triangulation preserves regions of interest



Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning

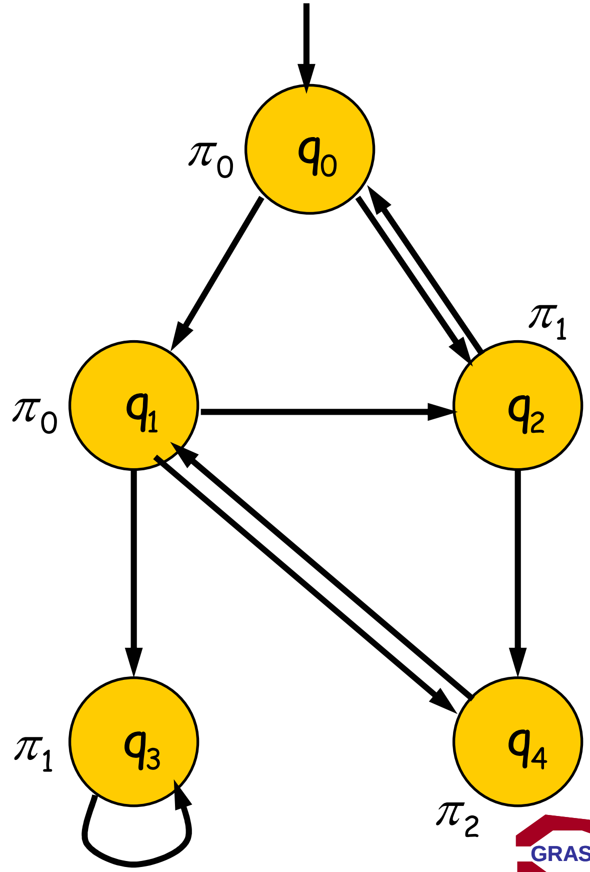


Linear Temporal Logic LTL_{-X}

(informally - discrete semantics)

The propositional formulas are formed using the traditional operators of *conjunction* ($\wedge$), *disjunction* ($\vee$), *negation* ($\neg$), *implication* ($\Rightarrow$), and *equivalence* ($\Leftrightarrow$). LTL formulas are obtained from the standard propositional logic by adding temporal operators such as *eventually* ($\Diamond$), *always* ($\Box$), and *until* (U).

Informally	Syntax
Eventually π_2 $\pi_0\pi_1\pi_2$	$\Diamond\pi_2$
Eventually Always π_1 $\pi_0\pi_0\pi_1\pi_1\pi_1\pi_1\ldots$	$\Diamond\Box\pi_1$
π_0 until π_2 $\pi_0\pi_0\pi_2$	π_0 U π_2



LTL expresses temporal specifications along sequences of states

Linear Temporal Logic LTL_{-X}

(formally – continuous semantics)

The input LTL_{-X} formulas are interpreted over continuous mobile robot trajectories. $x[t]$ denotes the flow of x(s) under the input u(s) for $t \leq s$. Proposition $\pi \in \Pi$ represents an area of interest in the environment which can be characterized by a convex set of the form:

$$P_i = \{x \in R^2 \mid \bigwedge_k a_k^T x + b_k \leq 0,\ a_k \in R^2, b_k \in R\}$$

$$x[t] \models \pi \quad \text{iff} \quad h_C(x(t)) = \pi$$

$$x[t] \models \neg\varphi_1 \quad \text{iff} \quad x[t] \not\models \varphi_1$$

$$x[t] \models \varphi_1 \vee \varphi_2 \quad \text{iff} \quad x[t] \models \varphi_1 \quad \text{or} \quad x[t] \models \varphi_2$$

$$x[t] \models \varphi_1\, U\, \varphi_2 \quad \text{iff}$$

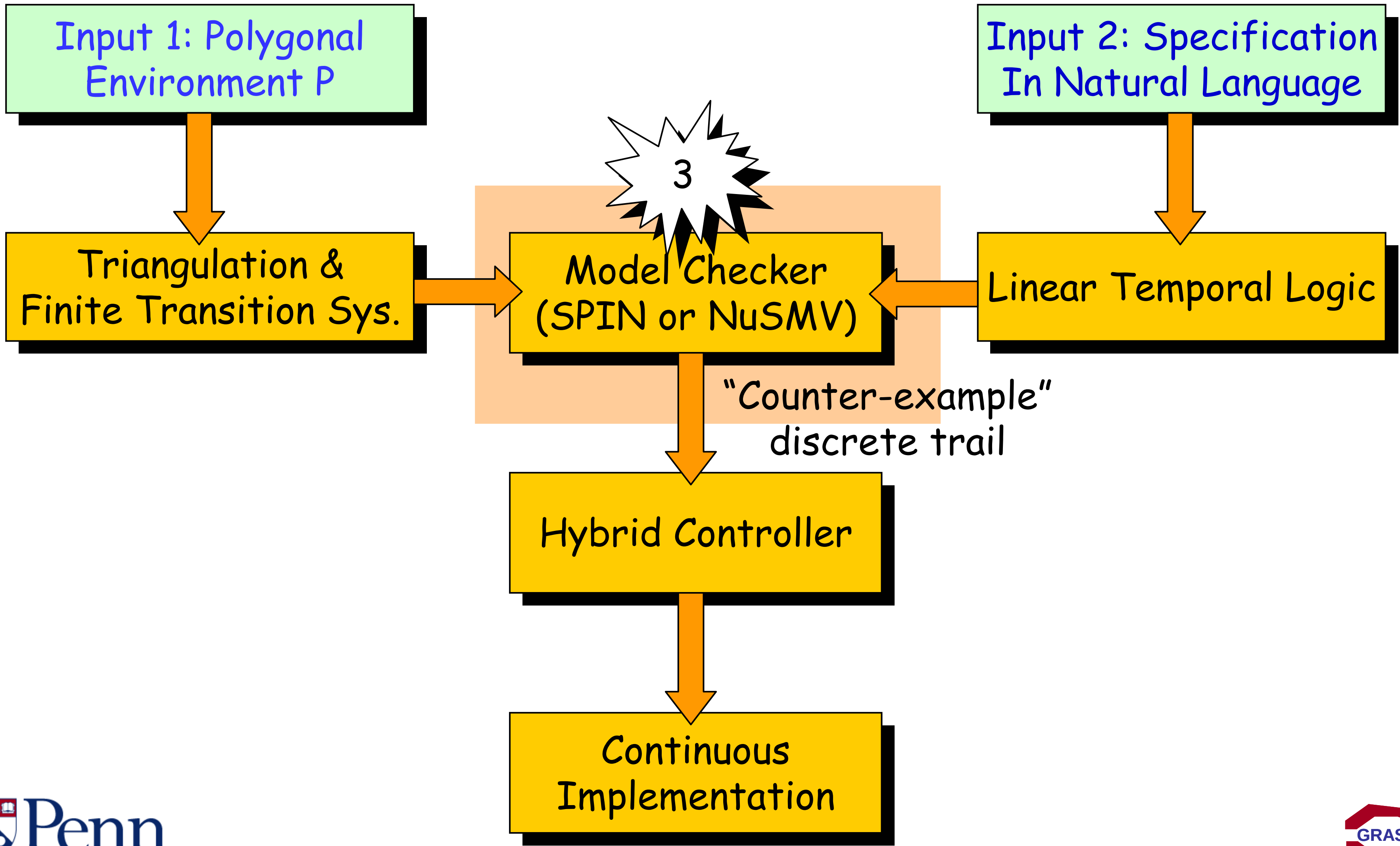
$$\exists s \geq t\ \ x[s] \models \varphi_2 \quad \text{and} \quad \forall\, t \leq t' < s \quad x[t'] \models \varphi_1$$

A trajectory x satisfies the specification φ and we write:

$$x \models \phi \quad \text{iff} \quad x[0] \models \varphi$$

where h_C is a function that maps the current state of the robot trajectory to an atomic proposition in the set Π, i.e. $h_C : P \rightarrow \Pi$

Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning



Planning via Model Checking

Model checking is the algorithmic procedure for testing whether a specification formula holds over some semantic model. The model of the system is usually given in the form of a finite transition system. The specification formula is usually in the form of the temporal logics LTL or CTL.

Model checking problem

$$\forall p, p \models \varphi$$

Dual Problems

Planning problem

$$\exists p, p \models \varphi$$

Model Checking problem

$$\forall p, p \models \neg\varphi$$

Generate discrete trajectory, originating at the initial condition, satisfying the temporal formula φ, using model checking tools.

SPIN: http://spinroot.com/spin/whatispin.html

LTL model checking

Automata theoretic approaches

NuSMV: http://nusmv.irst.itc.it/

CTL (and LTL) model checking

Symbolic based (BDD) approaches

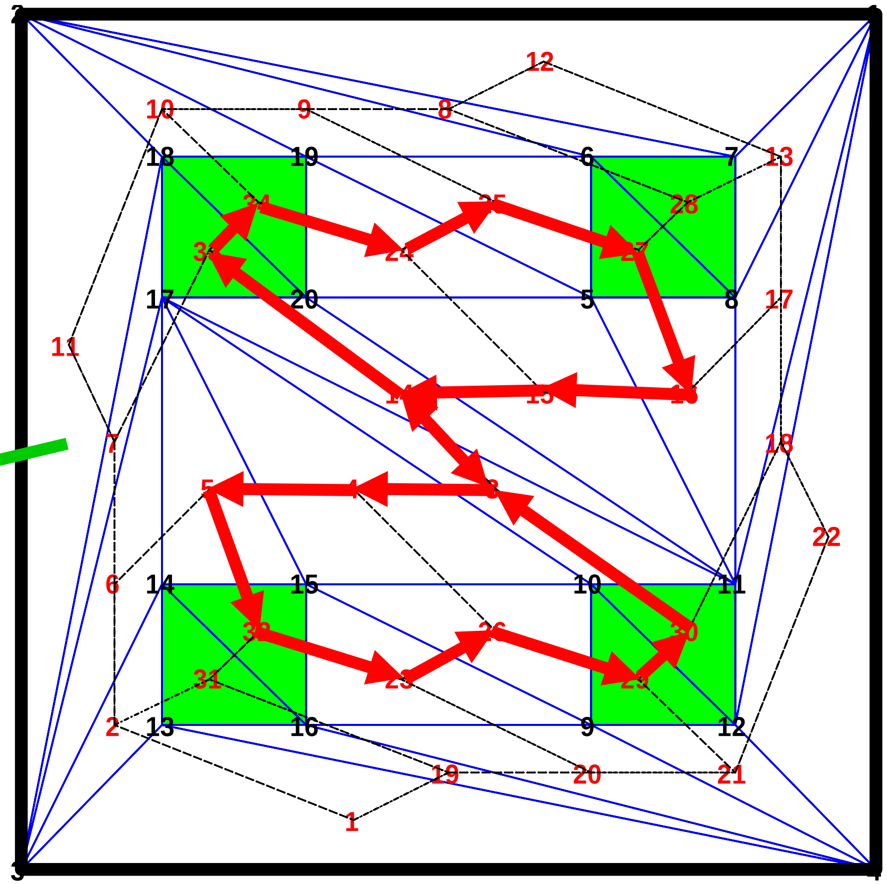
Penn
University of Pennsylvania

GRASP

NuSMV Model

$$\varphi = \neg \diamond (\pi_2 \wedge \diamond (\pi_3 \wedge \diamond (\pi_4 \wedge (\neg\pi_2 \wedge \neg\pi_3) U \pi_1)))$$

```
MODULE main
    VAR
        robot_1 : process robot(33);
    LTLSPEC ( ! F ( (pi2) & F ( (pi3) & F ( (pi4) & ( !(pi2) & ! (pi3) U (pi1)))))
MODULE robot(init_cond)
    VAR
        node : 0 .. 34;
    ASSIGN
    init(node) := init_cond;
    next(node) := case
        node=1 :  {2,19};
        node=2 : {1,6,31};
        node=3 : {4,14,30};
        node=4 : {3,5,26};
        node=5 : {4,6,32};
        node=6 : {2,5,7};
        node=7 : {6,11,33};
        ...
        node=32 : {5,23,31};
        node=33 : {7,14,34};
        node=34 : {10,24,33};
     esac;
```



The trajectory generated by NuSMV, satisfying this formula is:
33, 34, 24, 25, 27, 16, 15, 14, 3, 4, 5, 32, 23, 26, 29, 30, 3, 14, 33

Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning

Input 1: Polygonal Environment P

Input 2: Specification In Natural Language

Triangulation & Finite Transition Sys.

Model Checker (SPIN or NuSMV)

Linear Temporal Logic

"Counter-example" discrete trail

4

Hybrid Controller

Continuous Implementation

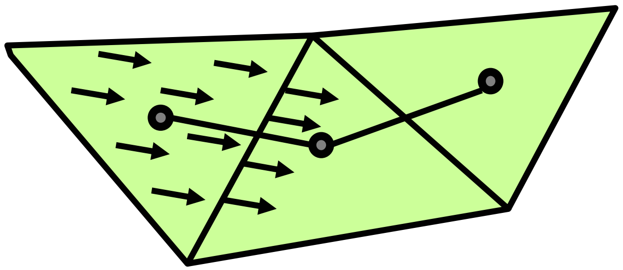
C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control," IEEE Transactions on Robotics, Accepted for publication.

D. C. Conner, A. Rizzi, and H. Choset, "Construction and automated deployment of local potential functions for global robot control and navigation," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-03-22, November 2003.

Hybrid Controller Implementation

Is the partition (triangulation) consistent with dynamics ?
Yes, if partition is a **bi-simulation.**

A triangulation is a **bisimulation** if the system can move between any two adjacent triangles regardless of the initial state. For each triangle, we design three controllers ensuring that system exits the triangle from the desired facet to the adjacent triangle.



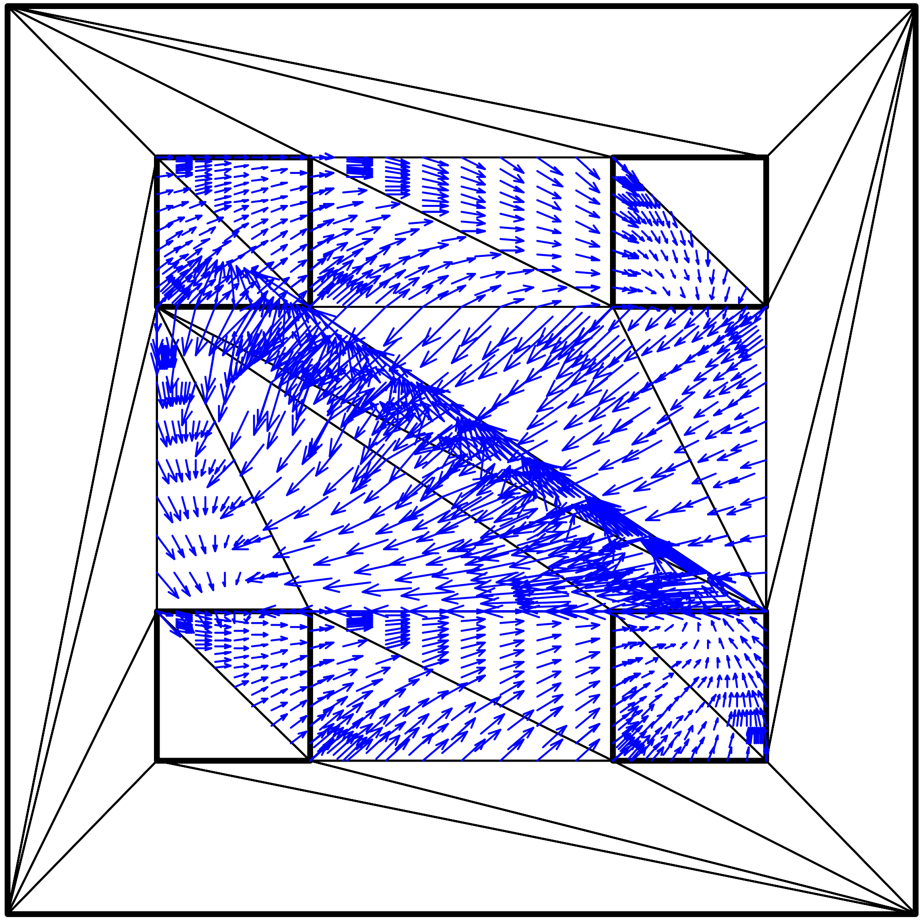
Thm: There exist (many) affine vector fields

$$\dot{x}_P = u_P \qquad u_P = Ax + b \in U_P$$

on any triangle, satisfying the bisimulation property.

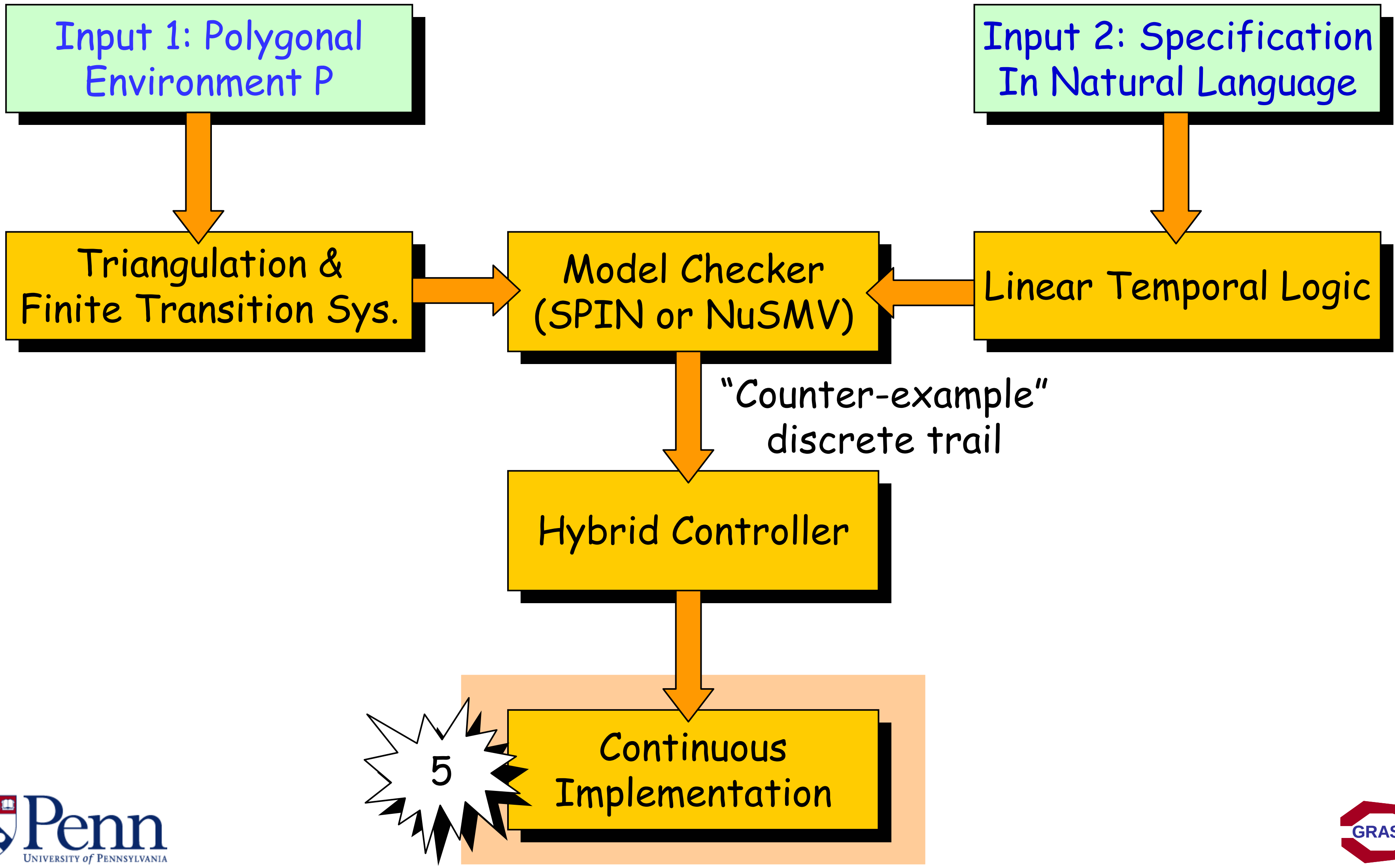
Affine functions on simplexes are uniquely defined on vertices. The set of all controllers can be parameterized by the values on the vertices.

Refinement



Based on the discrete path, we design bi-simulation controllers driving the robot from one triangle to the adjacent triangle. We can take advantage of the non-unique affine solutions by matching affine vector fields on common facets, (if possible).

Overview of the Algorithm for the Open-Loop Temporal Logic Motion Planning



Main Result - Completeness

If the robot is modeled as $\dot{x}(t) = u(t)$ and $\delta(0, \epsilon) \in U$ then

$$x \models_C \varphi \text{ iff } p \models_D \varphi$$

If the system is modeled as $\dot{x} = Ax + Bu$ and conditions (*) then

$$x \models_C \varphi \text{ iff } p \models_D \varphi$$

In any case: $x \models_C \varphi \Rightarrow p \models_D \varphi$

For the multi-robot case using the above dynamics: $p \models_D \varphi \not\Rightarrow x \models_C \varphi$

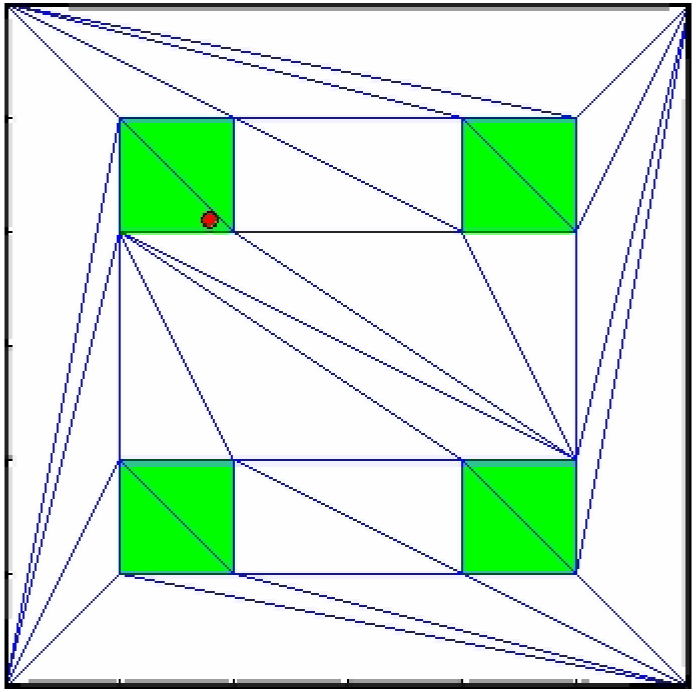
Other dynamics: $p \models_D \varphi \overset{??}{\Rightarrow} x \models_C \varphi$

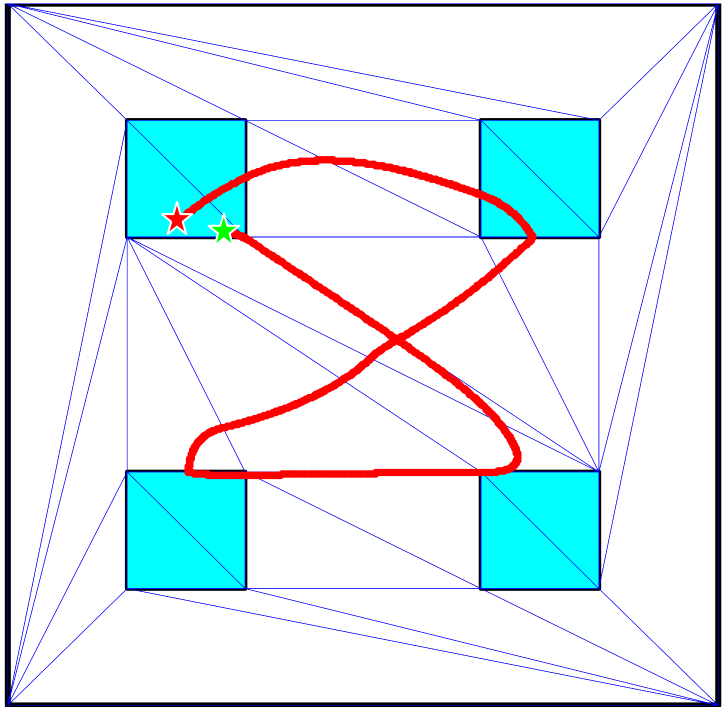
(*) L.C.G.J.M. Habets and J.H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40:21–35, 2004.

Continuous refinement

★ Start

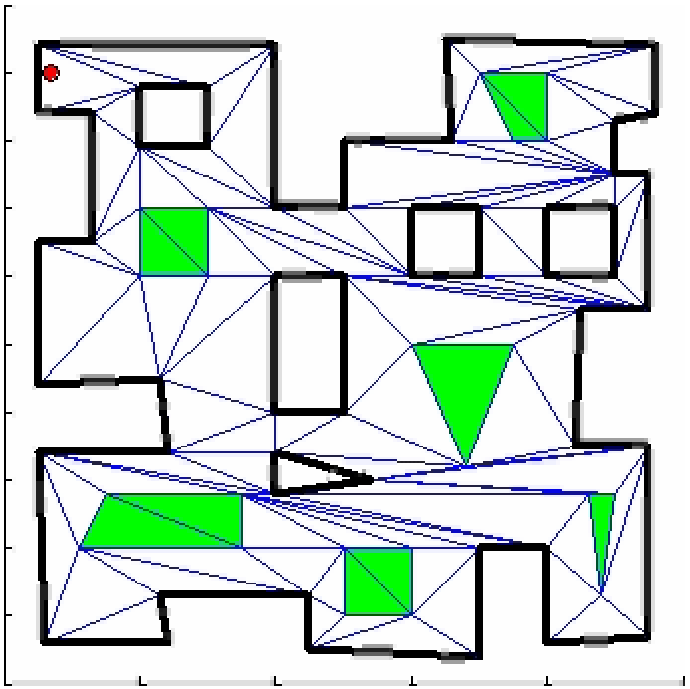
★ Goal

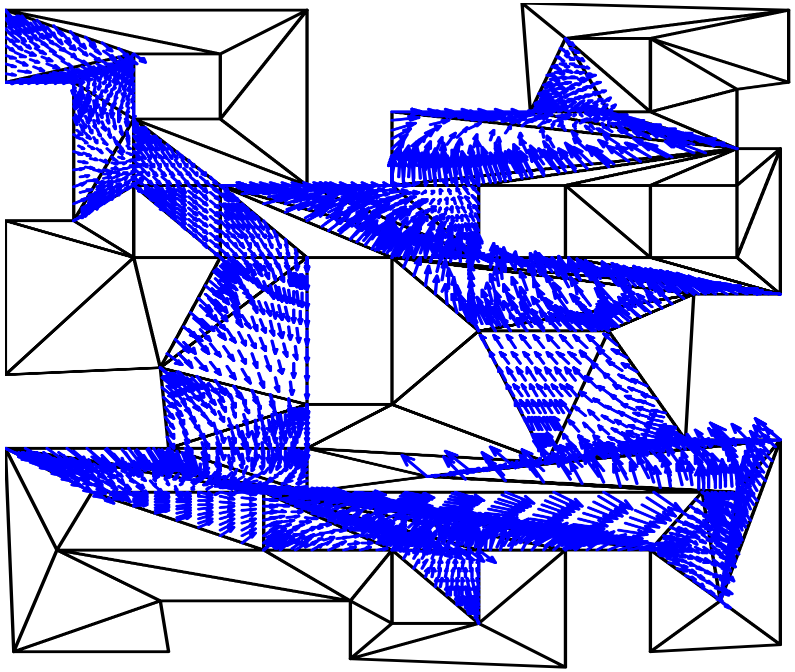


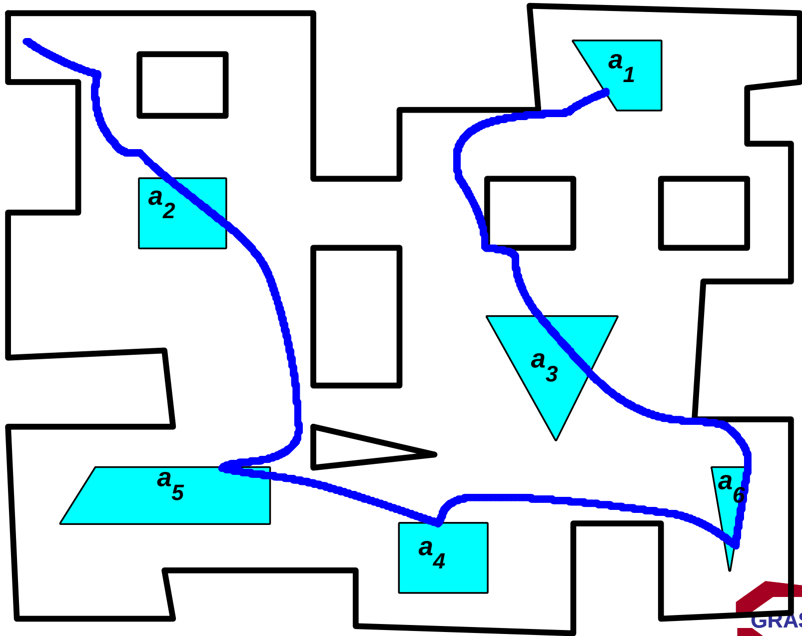


Examples

Spec: Go to areas 1, 2, 3, 4, 5, 6
in no particular order.



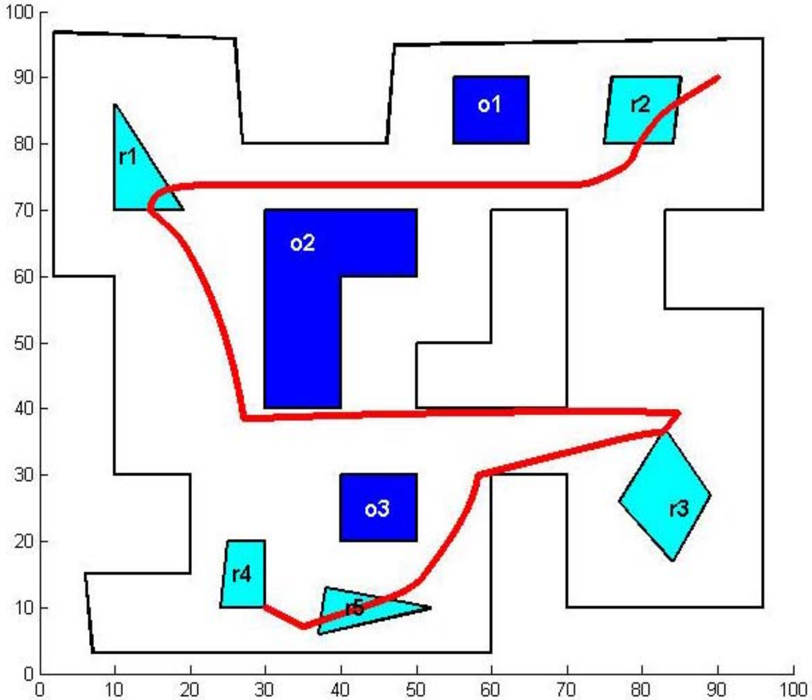




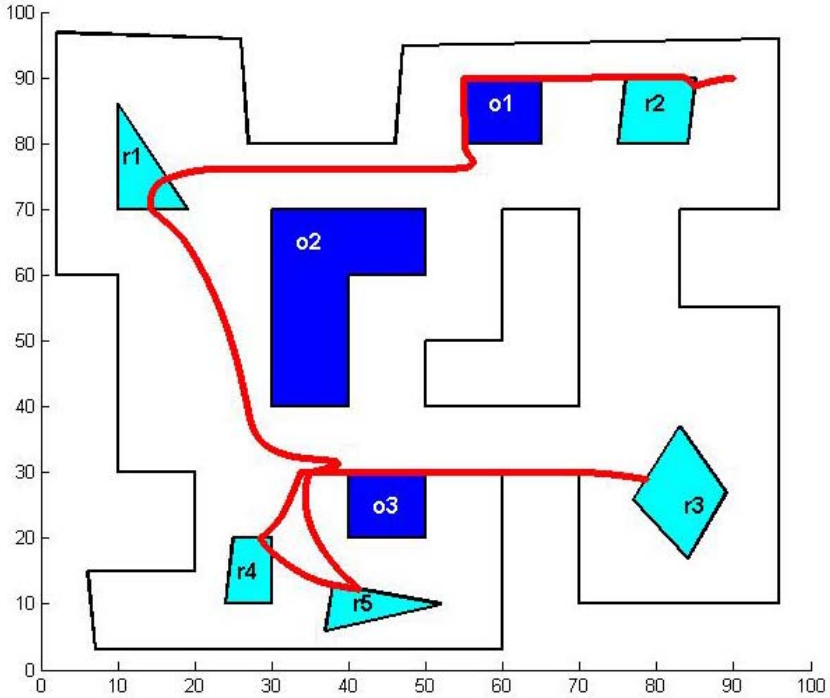
Computation time: Triangulation < 1sec,
Model checking < 1sec,
Hybrid Controller ~13sec (MATLAB)

Examples

Spec : Go to area 2, then to area 1 and then cover areas 3, 4, 5 – all this, while avoiding obstacles O_1, O_2, O_3



SPIN



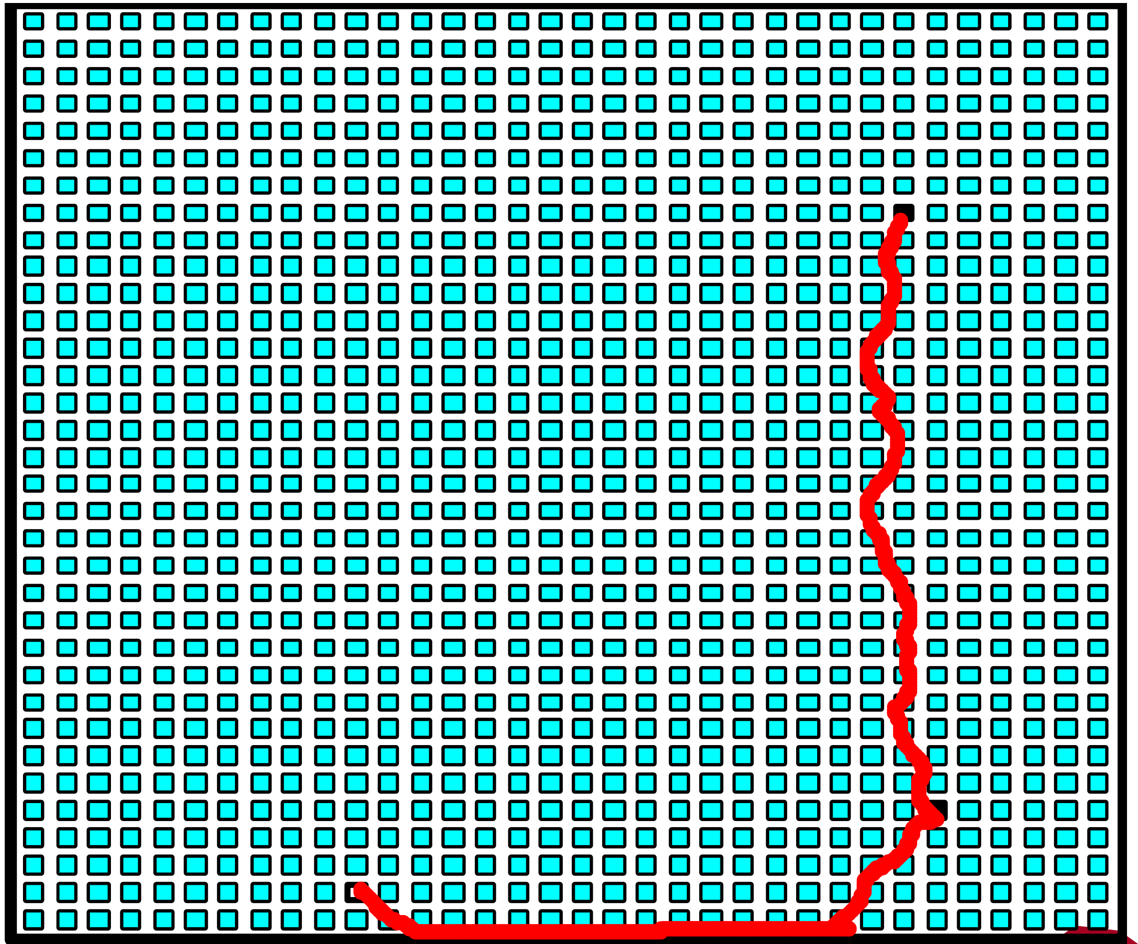
NuSMV

Larger examples

Spec : Go to the two black rooms

<u>Problem Size</u>
1156 observables
9250 triangles
Solution path : 145 triangles
145 controllers

<u>Computation time</u>
Triangulation : A few seconds
NuSMV : 55 seconds
Matlab : 90 seconds



Conclusions

- ✓ Presented a formal and compact way to capture complicated path planning specifications. Also, a unified specification language for many tasks.
- ✓ A connection between high level AI planning and low level controller design
- ✓ Completeness results (for certain cases)
- ✓ Computationally efficient approach
- ✓ Robustness with respect to the initial conditions within a class of the partition

Future Issues

More general decompositions/abstractions

- Abstraction should depend on more complicated dynamics
- Robust abstractions with respect to modeling/sensing noise

Open-loop versus closed loop planning

- Robust satisfaction of temporal formulas
- Feedback plans will result in hybrid controllers
  - G. E. Fainekos, H. Kress-Gazit and G. J. Pappas, Hybrid Controllers for Path Planning: A Temporal Logic Approach, Submitted to the 44th IEEE Conference on Decision and Control, Seville, Spain, 2005

Multi-robot logics

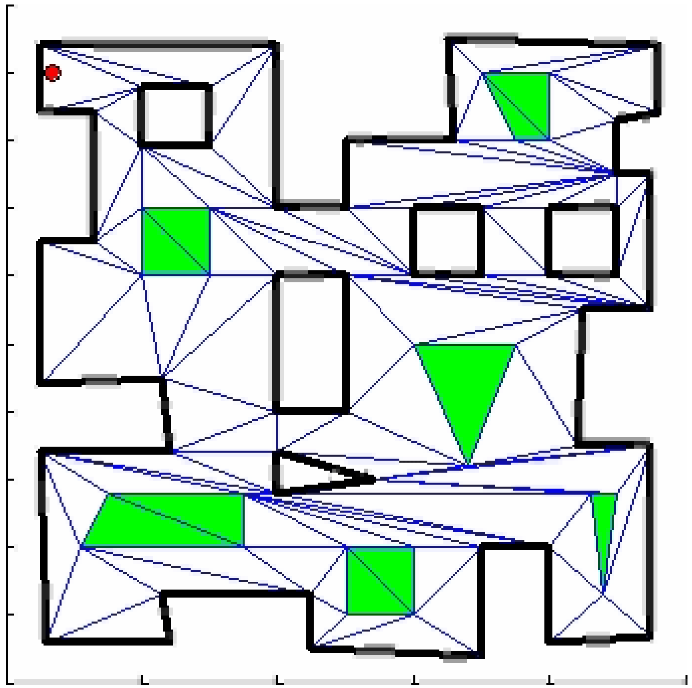
- Composition semantics for multiple robots

From natural language to robot motion

Experiments (ground vehicles and UAVs)

Thank You!

Questions?



http://www.seas.upenn.edu/~fainekos/