HIGH LEVEL TASKS TO LOW LEVEL CONTROLLERS

ECE584: Embedded System Verification

Lecture 21

slides from: Hadas Kress-Gazit

hadaskg@grasp.upenn.edu

lecturer : Sayan Mitra

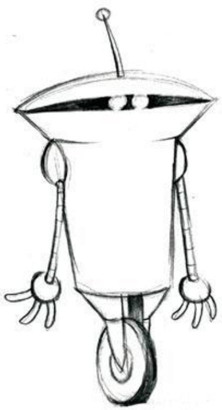
mitras@crhc.uiuc.edu

Synthesis Problem for Hybrid Systems

- How does one describe symbolic, high level tasks and transform them automatically into sensing and control while obtaining formal guarantees of correctness?

Problem

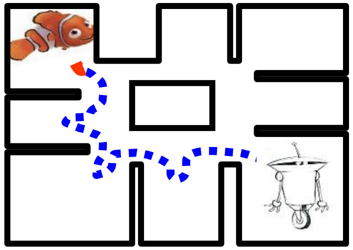
task specification



robot model

?

dynamic environment



correct robot motion and action

previous work: planning in AI and control

- *Schoppers*. **Universal plans** for reactive robots in unpredictable environments. IJCAI 1987.

- *LaValle*. Planning Algorithms. Cambridge University Press, Cambridge, 2006

- *Burridge, Rizzi, and Koditschek*, **Sequential composition** of dynamically dexterous robot behaviors, *J. of Robotics Research*, 1999.

- *Choset, Lynch, Hutchinson, Kantor, Burgard, Kavraki & Thrun*. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston, 2005.

- *Frazzoli, Dahleh, & Feron*, **Maneuver-based motion planning** for nonlinear systems with symmetries, IEEE Trans. Robot., 2005.

planning with hybrid systems

1. Kress-Gazit, Fainekos, Pappas: Where's Waldo? Sensor-Based **Temporal Logic Motion Planning**. ICRA 2007.

2. Quottrup, Bak and Izadi-Zamanabadi. **“Multi-robot planning : a timed automata approach”**. *ICRA*, 2004.

3. Kloetzer and Belta. “**A fully automated framework for control of linear systems from LTL specifications”**. *HSCC*, 2006.

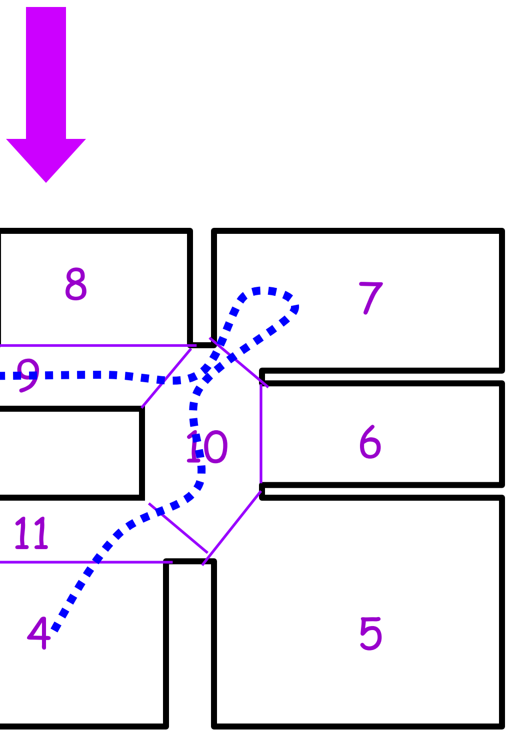
4. Delmotte, Mehta, & Egerstedt, “**Modebox a software tool for obtaining hybrid control strategies from data**,” IEEE Robot. Automat. Mag., 2008.

outline

- planning for static environments
- planning for dynamic environments
- complex dynamics
- distributed robotics
- case studies

static environments

starting in corridor 12, go to Rooms 1, 7 and 2 in any order, then to Room 8 and finally, go to either Room 4 or 5 without going through corridor 12



static environments

robot 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:

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

specification: linear temporal logic (**LTL**) formula ϕ

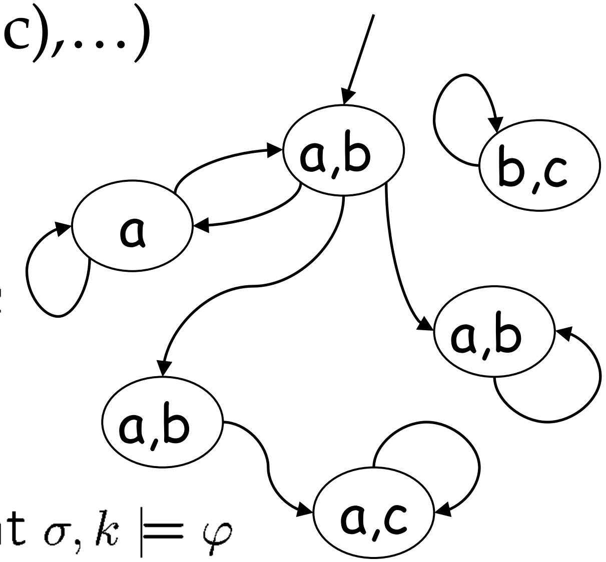
problem: given robot model, environment P, initial condition p(0), and an LTL formula ϕ, find control input u(t) such that **p(t) satisfies ϕ**.

Linear Temporal Logic (LTL)

Syntax:

$$\varphi ::= \pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi\,\mathcal{U}\,\varphi$$

Semantics: Truth is evaluated along infinite computation paths σ ((a,b),a,a,a… (a,b),(a,b),(a,c),(a,c),…)



$\sigma, i \models \pi$ iff $\pi \in \sigma(i)$

$\sigma, i \models \neg\varphi$ if $\sigma, i \not\models \varphi$

$\sigma, i \models \varphi_1 \vee \varphi_2$ if $\sigma, i \models \varphi_1$ or $\sigma, i \models \varphi_2$

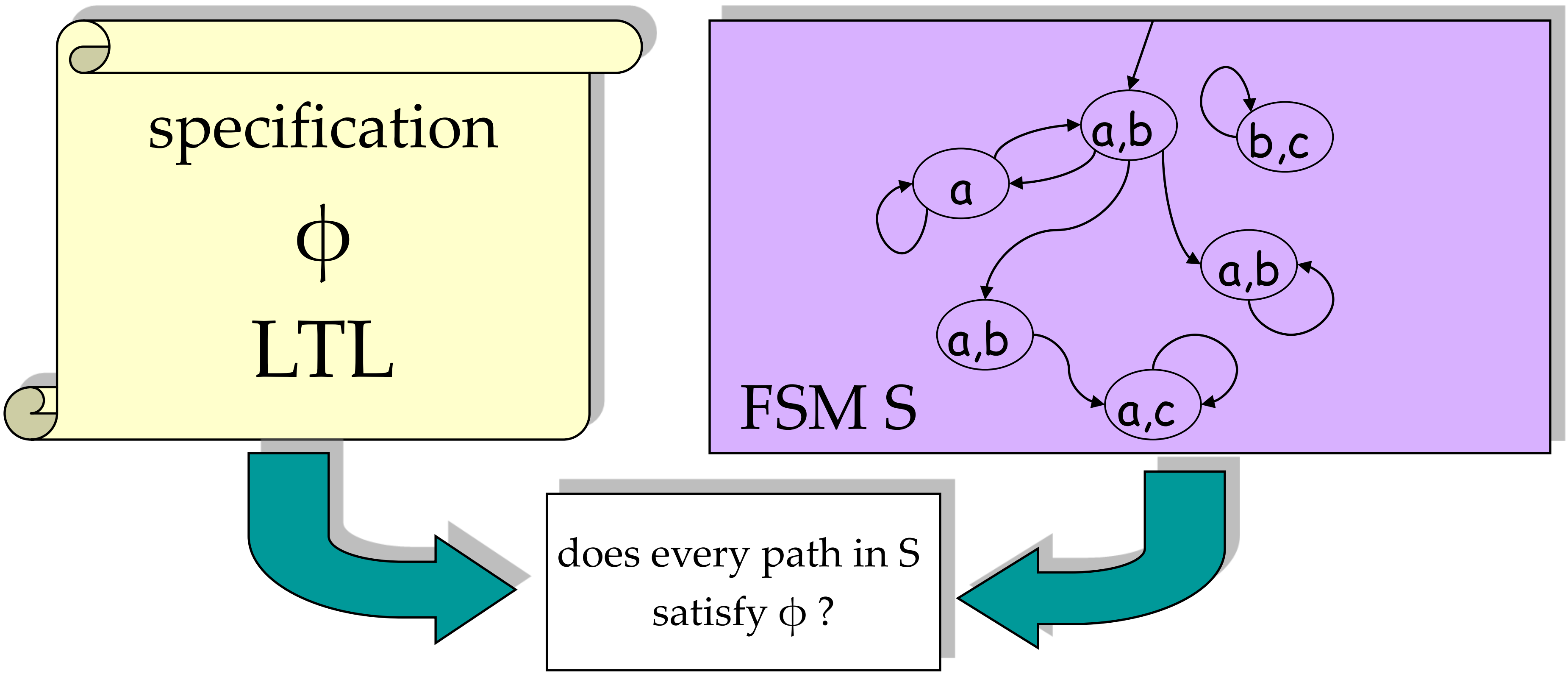
"next" $\sigma, i \models \bigcirc\varphi$ if $\sigma, i+1 \models \varphi$

"always" $\sigma, i \models \Box\varphi$ if for all $k \geq i$ $\sigma, k \models \varphi$

"eventually" $\sigma, i \models \Diamond\varphi$ if there exists $k \geq i$ such that $\sigma, k \models \varphi$

"until" $\sigma, i \models \varphi_1 \mathcal{U} \varphi_2$ if there exists $k \geq i$ such that $\sigma, k \models \varphi_2$, and for all $i \leq j < k$ we have $\sigma, j \models \varphi_1$

model checking



- Complexity of LTL model checking $|S| 2^{|\phi|}$
- guaranteed to terminate with the correct answer.
- if not satisfied, a counter-example path is given

task specifications in LTL

- “visit rooms 1,2,3 while avoiding corridor 1”:

  [] ¬(corridor1) ∧ ◊(room1) ∧ ◊(room2) ∧ ◊(room3)

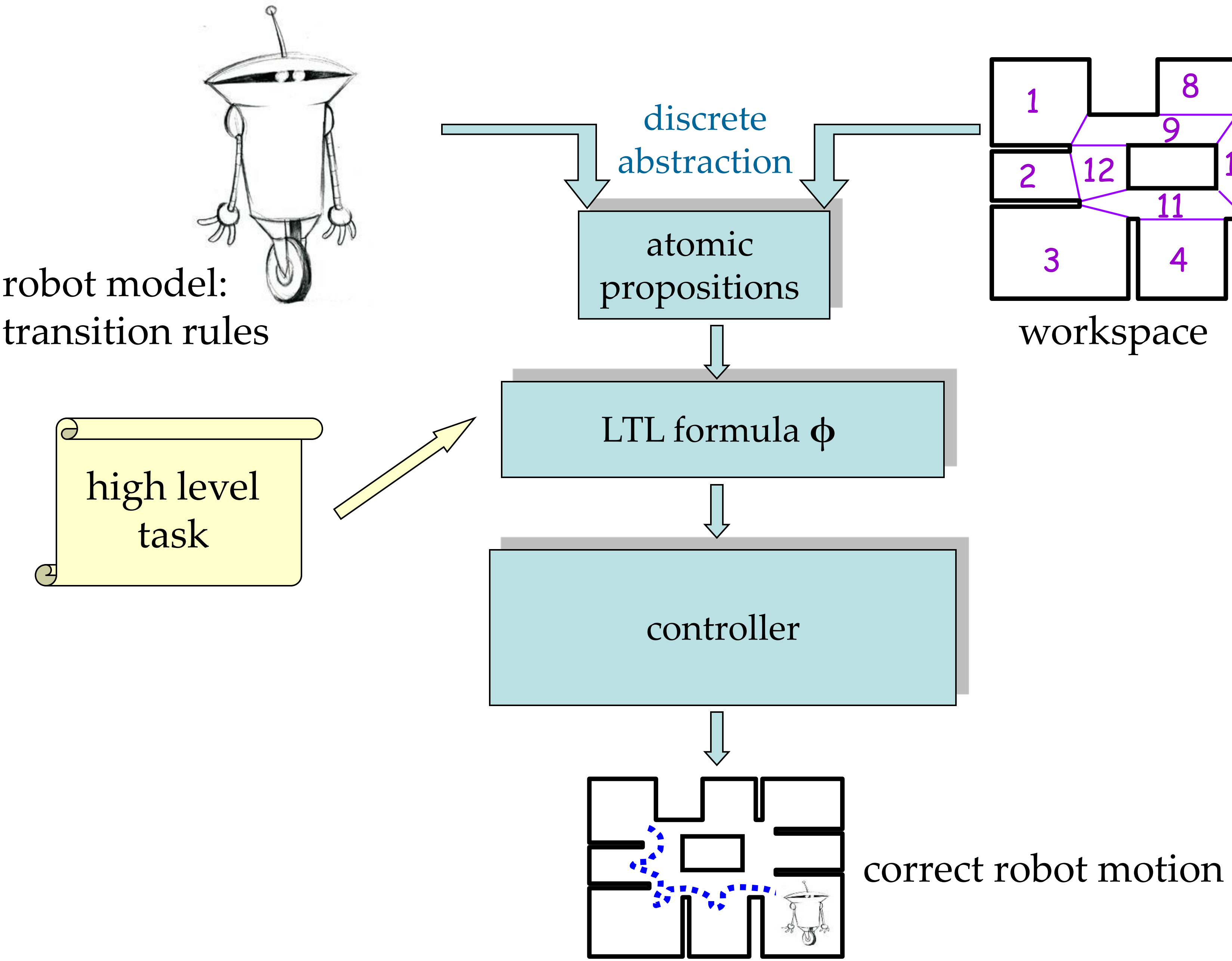
- “ if the light is on, visit rooms 1 and 2 infinitely often”:

  []((LightOn) -> ([]◊(room 1) ∧ []◊(room 2)))

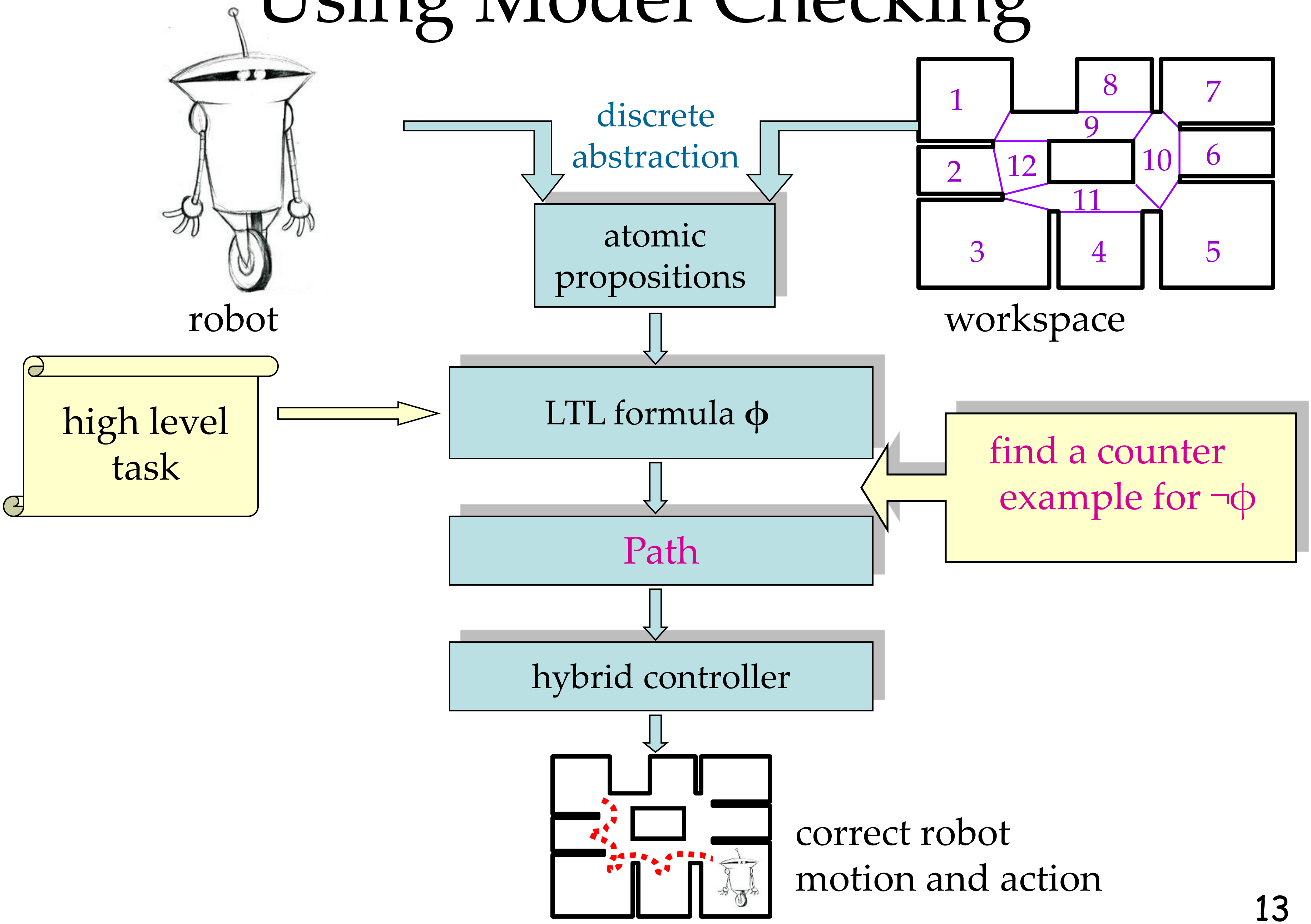
- “if you are in room 3 and Mika is there, beep”

  []((room3) ∧ (SeeMika) -> (Beep))

- and much more…


discrete
abstraction
atomic
propositions
1
8
7
9
2
12
10
6
11
3
4
5
workspace
robot model:
transition rules
LTL formula ϕ
high level
task
controller
correct robot motion

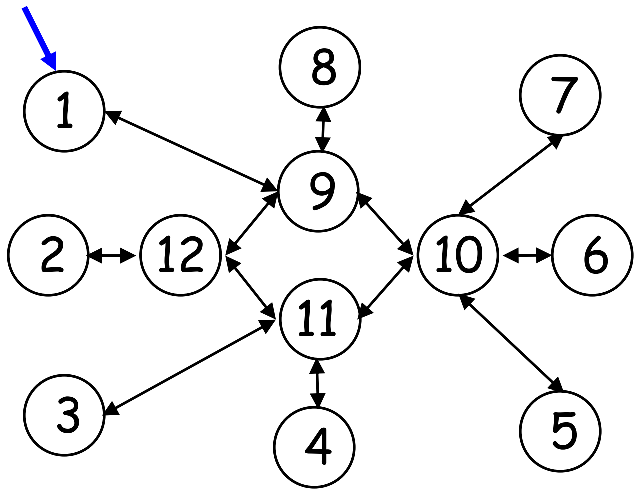
Using Model Checking

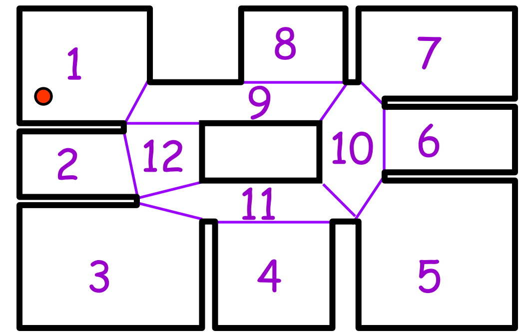


- Example

“Go to room 4”

ϕ = ◊(room4)





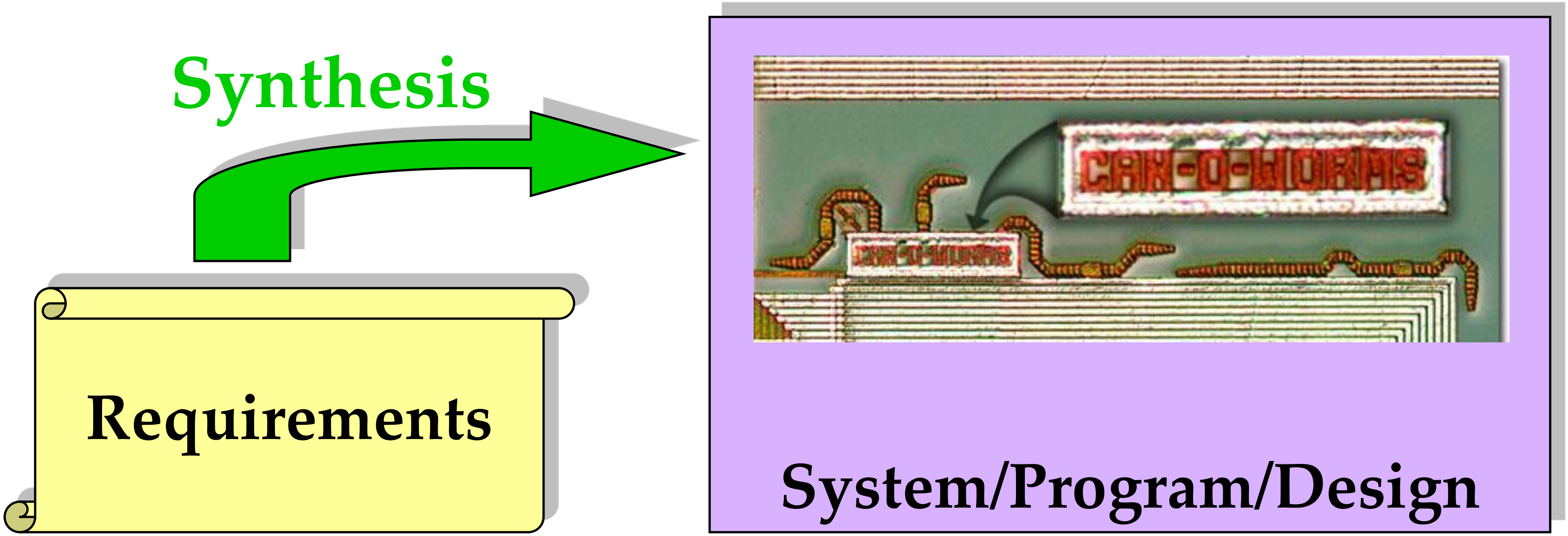
Model check the formula **¬ϕ**

¬◊(room4)

The formula is False and the counter example is:

room1, room9, room12, room11, room4

Gives a path to room 4

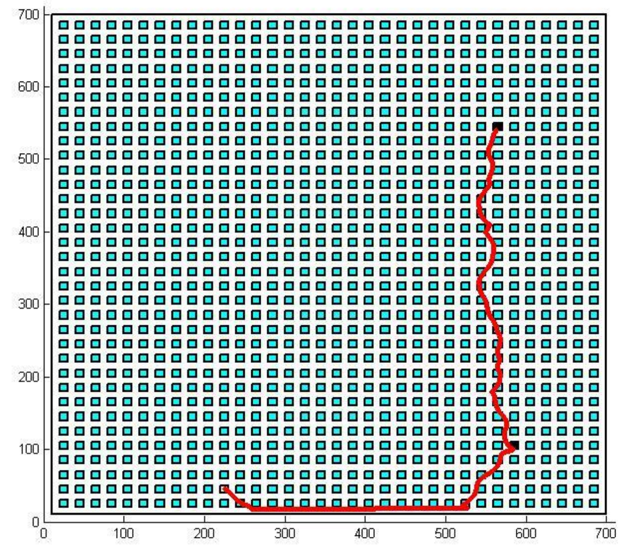


- given a formula, create the system

- synthesis of the full LTL is double exponential in the size of the formula

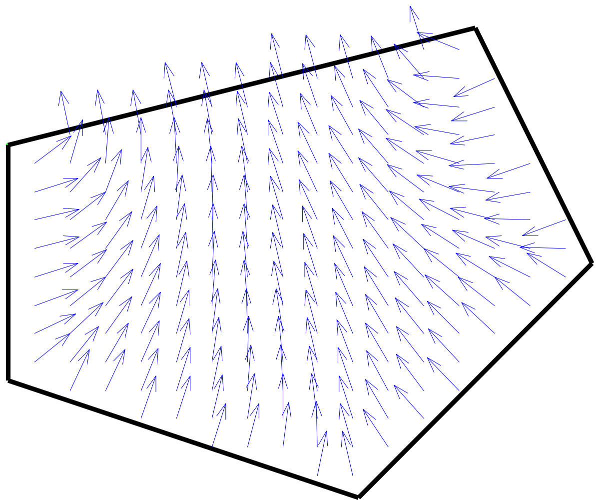
- for a specific fragment, it is polynomial in the state space

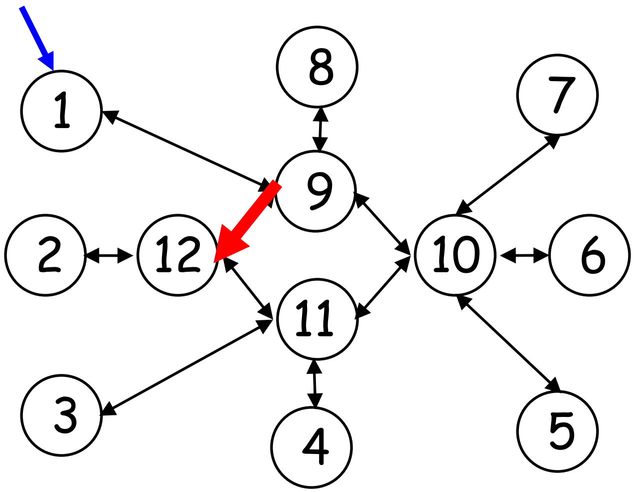
- Advantages
  - Can handle large problems – "Symbolic model checking: 10^{20} states and beyond" (Burch, Clarke, McMillan,Dill,Hwang)
  - Complex motion behaviors:
    - "Go to X or Y while avoiding Z"
    - "If you go through W then go to X too"
    - "Go to X,Y,W and Z in any order"
  - Many tools (NuSMV, SPIN…)



- Disadvantages
  - Paths are not optimal
  - Result is a path – not a plan, so we can't do reactive tasks.

From FSM to Hybrid controller





- need continuous controllers to “match” the discrete transitions
- design “atomic” feedback controllers to mimic the transitions
- bisimilar by construction

Hybrid controller

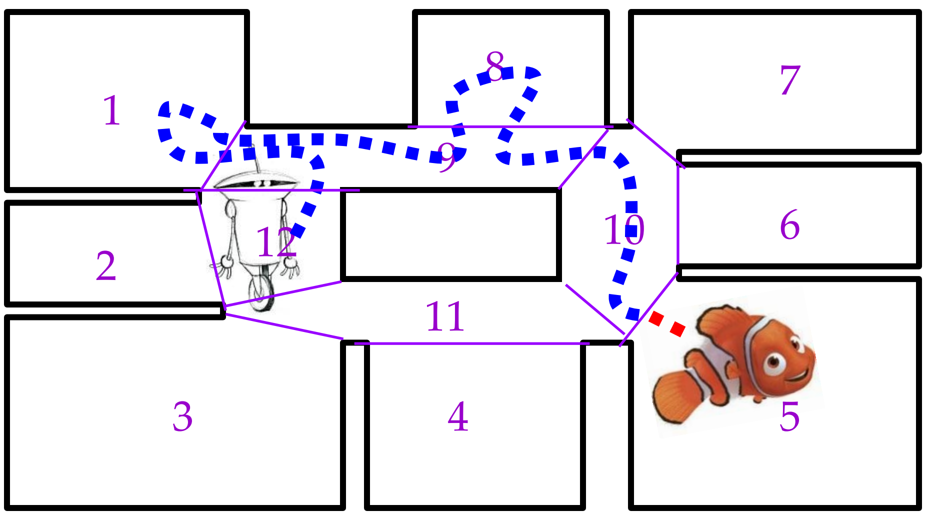
- We compose a set of "atomic" feedback controllers that drive the robot based on its dynamics.

Guarantee

Given a workspace decomposition and a set of atomic controllers, if the specification φ can be satisfied by the discrete abstraction, a hybrid controller will be generated such that p(t) satisfies ϕ

Reactive planning for dynamic environments

"Nemo may be sitting in one of rooms 1, 3, 5 and 8. Starting in corridor 12, look for him in these rooms. If at some point you see him, stop and beep"



Dynamic environments

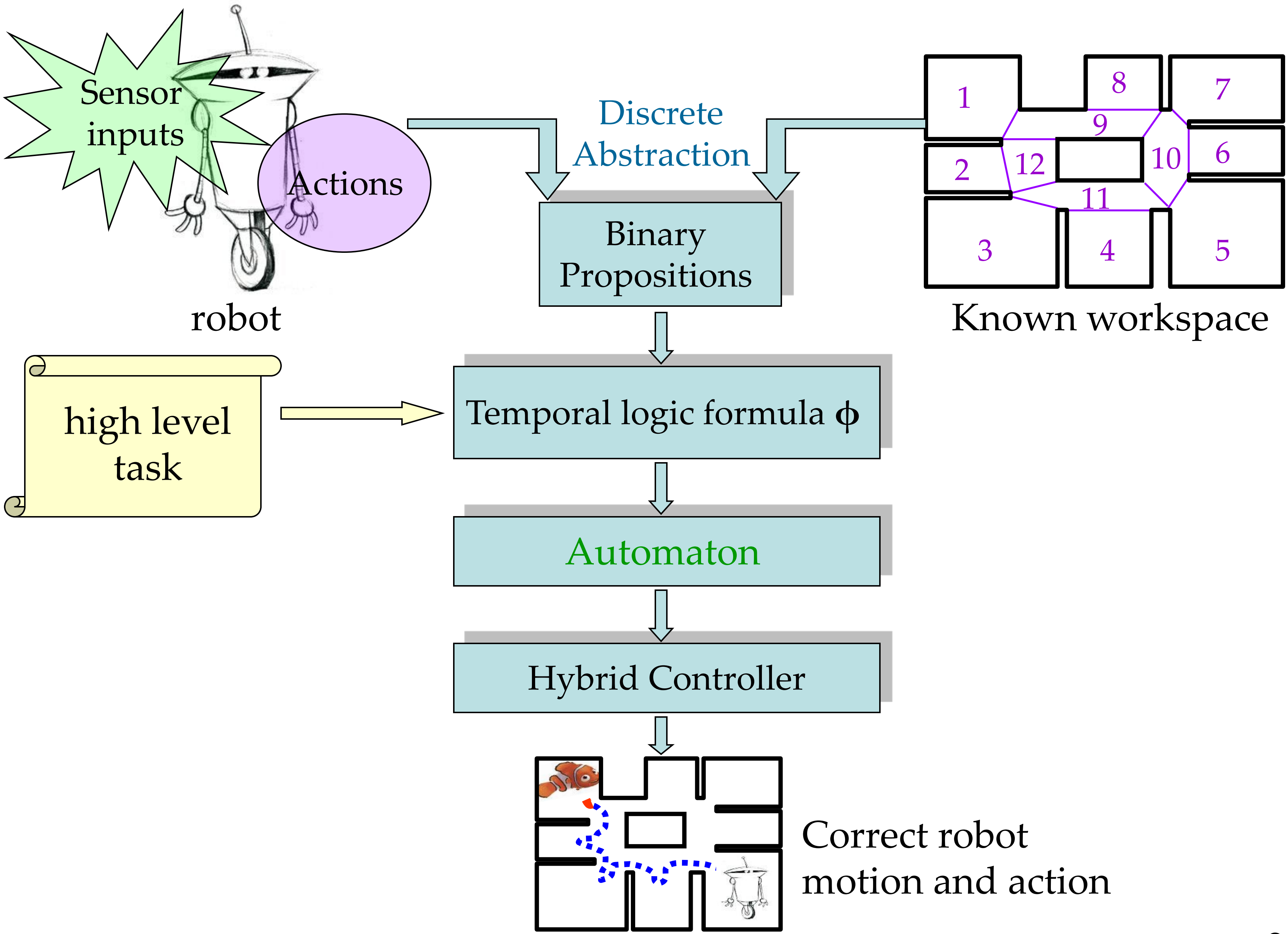
Model: We consider a fully actuated, planar model of robot motion operating in a polygonal environment *P*. The motion of the robot is expressed as:

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

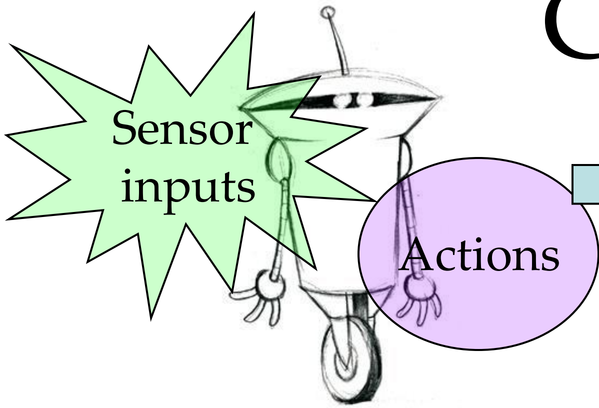
In addition, the robot has binary sensor inputs and actions

Specification: A linear temporal logic (**LTL**) formula ϕ that captures assumptions about the environment and the robot's reactive behavior.

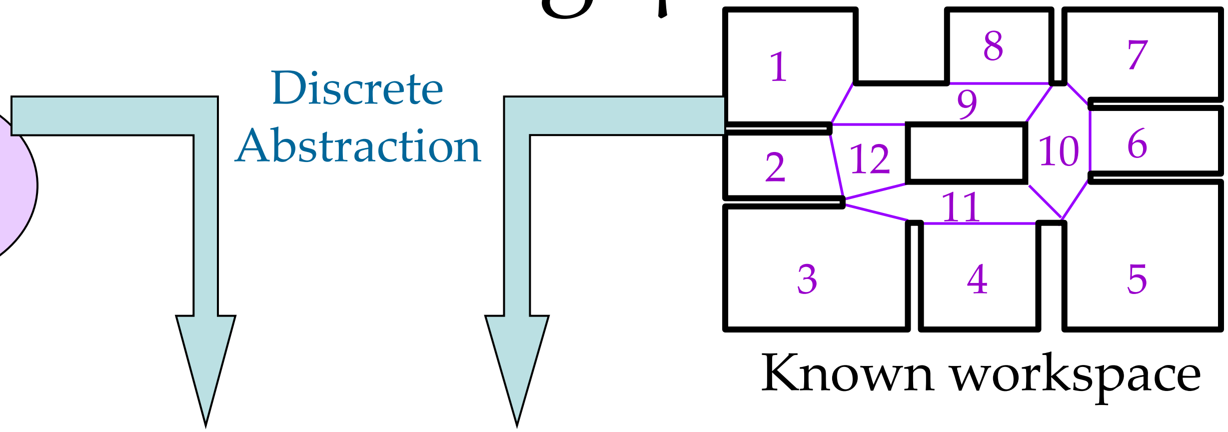
Problem: Given robot model, environment P, set of initial conditions, and an LTL formula ϕ, find control input u(t) such that **p(t) satisfies ϕ**, in any admissible environment


Sensor inputs
Actions
robot
Discrete Abstraction
Binary Propositions
1
2
3
4
5
6
7
8
9
10
11
12
Known workspace
high level task
Temporal logic formula ϕ
Automaton
Hybrid Controller
Correct robot motion and action

Constructing ϕ



robot



Known workspace

Sensor (Input) propositions:

X = {SenseNemo, SenseFire, HearBaby,… }

= $\{s_{Nemo}\}$

Robot (Output) propositions:

Y = {Room1, Room2,…,Room12, Beep, RecordVideo,… }

= $\{r_1, r_2, \ldots, r_{12}, \text{Beep}\}$

LTL fragment

We consider LTL formulas of the form:

$$\varphi = \overline{\varphi_e} \Rightarrow \overline{\varphi_s}$$

Assumptions about environment	Desired robot behavior

only if the **assumptions** are met the desired behavior is **guaranteed**.

Example

<u>Task</u>: "Nemo may be sitting in one of rooms 1, 3, 5 and 8. Starting in corridor 12, look for him in these rooms. If at some point you see him, stop and beep"

Sensor (Input) propositions: $X = \{s_{Nemo}\}$
Robot (Output) propositions: $Y = \{r_1, r_2, \ldots, r_{12}, Beep\}$

Environment Assumptions

Desired behavior

Initial Conditions

Transitions

Goals

$$\varphi = \left(\begin{array}{c} \neg s_{Nemo} \\ \wedge [](s_{Nemo} \rightarrow \bigcirc s_{Nemo}) \\ \wedge []\left(\left(\bigwedge_{i\in\{1,3,5,8\}} \neg r_i\right) \rightarrow \left(s_{Nemo} \leftrightarrow \bigcirc s_{Nemo}\right)\right) \\ \wedge []\Diamond(True) \end{array} \right) \rightarrow \left(\begin{array}{c} r_{12} \bigwedge_{i\neq 12} \neg r_i \wedge \neg Beep \\ \wedge [](r_1 \rightarrow \bigcirc r_1 \vee \bigcirc r_9) \\ \vdots \\ \wedge [](\bigcirc s_{Nemo} \leftrightarrow \bigcirc Beep) \\ \bigwedge_{i\in\{1,3,5,8\}} []\left(\left(r_i \wedge \bigcirc s_{Nemo}\right) \rightarrow \bigcirc r_i\right) \\ \bigwedge_{i\in\{1,3,5,8\}} []\Diamond\left(r_i \vee s_{Nemo}\right) \end{array} \right)$$

Why this structure?

- Can be synthesized into an automaton

- No significant loss of expressivity with respect to the full LTL

- Clear distinction between assumptions and desired behavior

Automaton and controller

- synthesis algorithm due to Piterman, Pnueli and Sa'ar (VMCAI 2006)

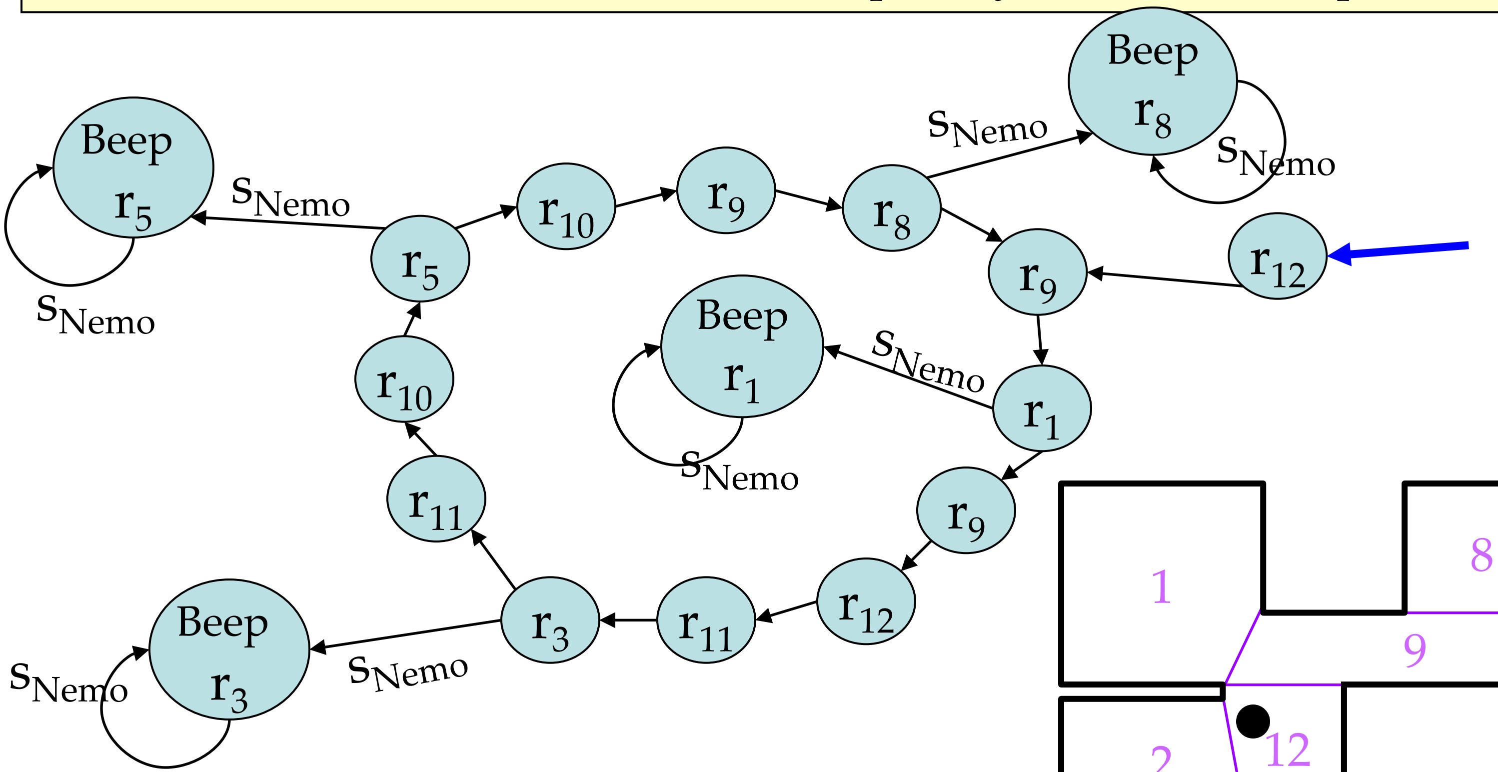
- polynomial $O(n^3)$ in the number of states (as opposed to double exponential in the length of the formula)

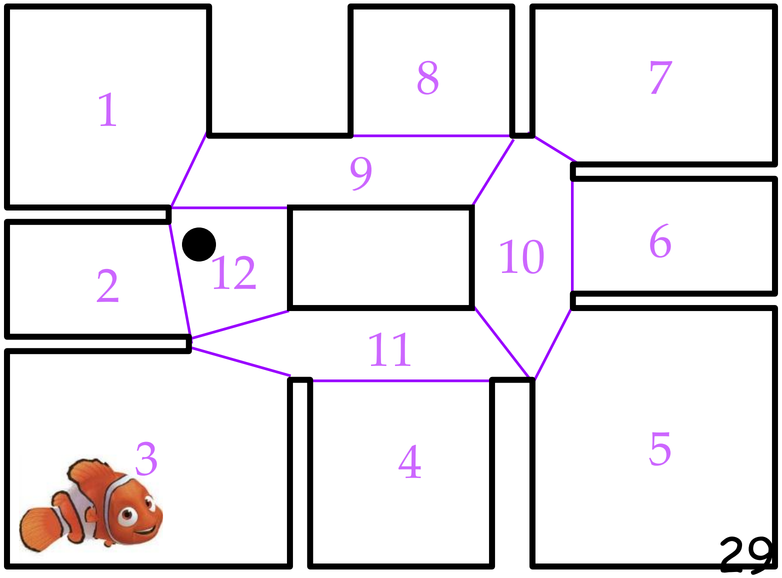
- solves a game between the environment and the robot. If the robot wins, no matter what the environment does, an automaton is extracted.

- hybrid controller activates the atomic controllers and binary actions based on the sensor inputs

Example

Task: "Nemo may be sitting in one of rooms 1, 3, 5 and 8. Starting in corridor 12, look for him in these rooms. If at some point you see him, stop and beep"

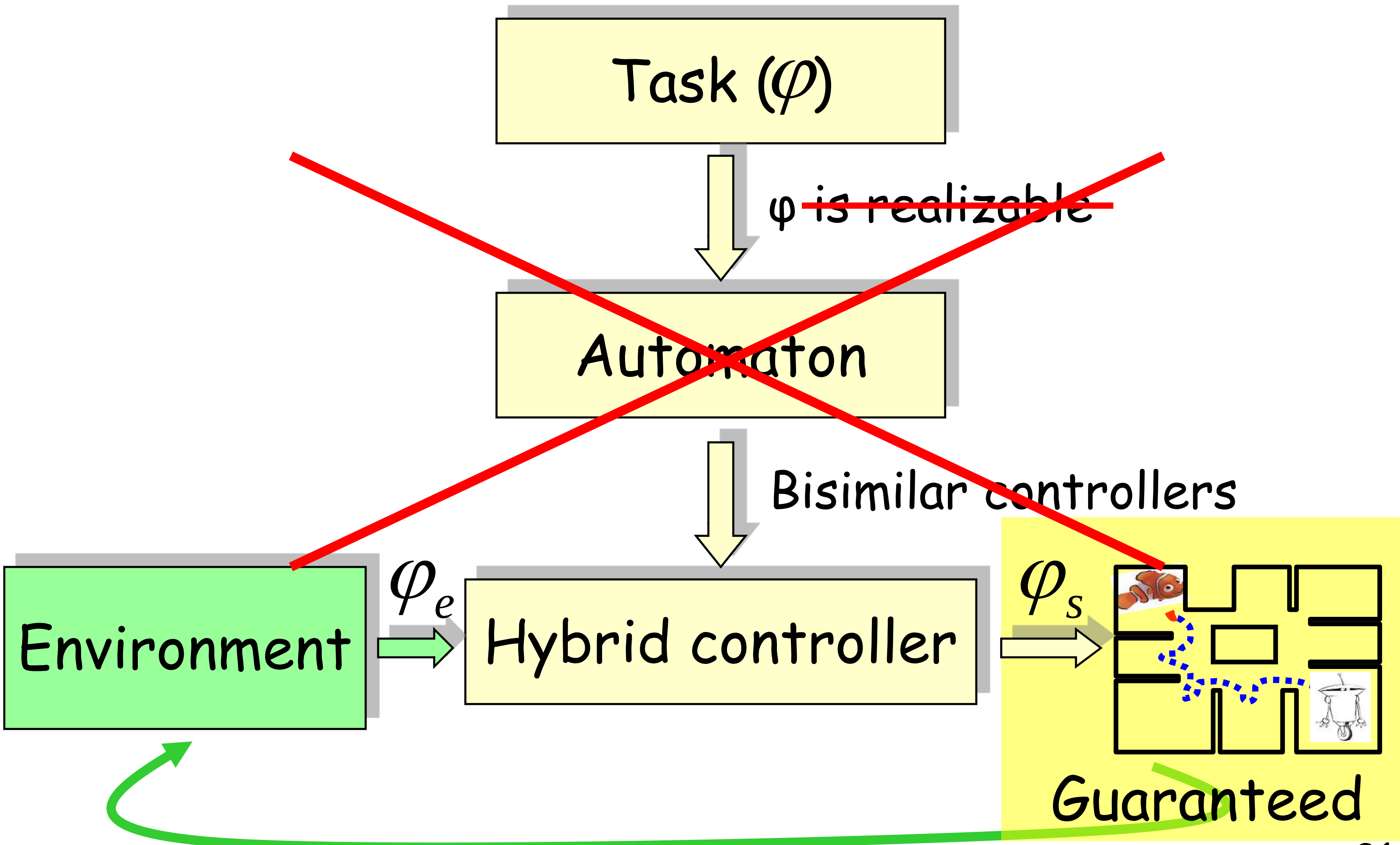




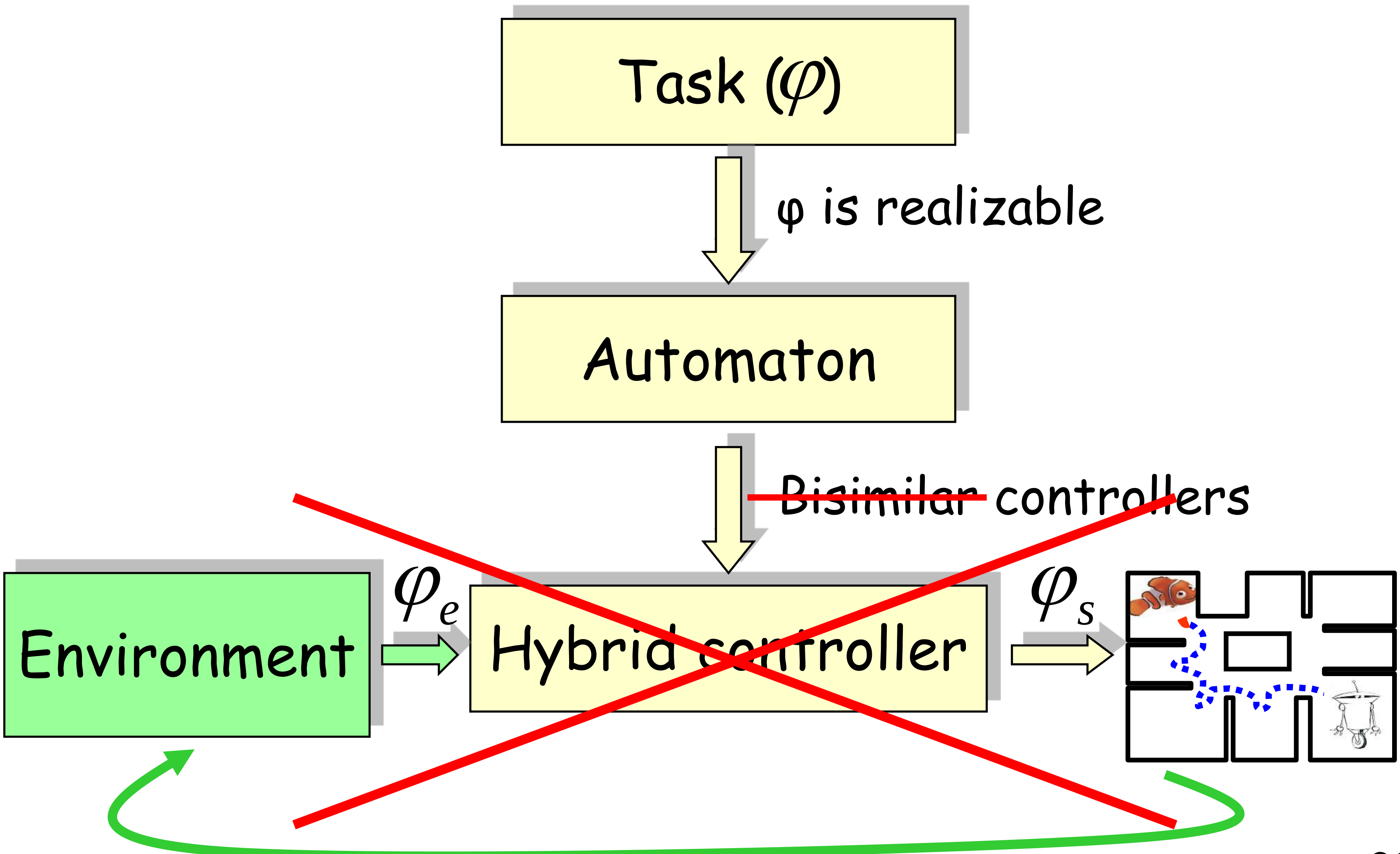
Guarantee

Given a workspace decomposition and a set of atomic controllers, if the specification can be satisfied by the discrete abstraction and the environment satisfies the assumptions made, a hybrid controller will be generated such that **p(t) satisfies ϕ**

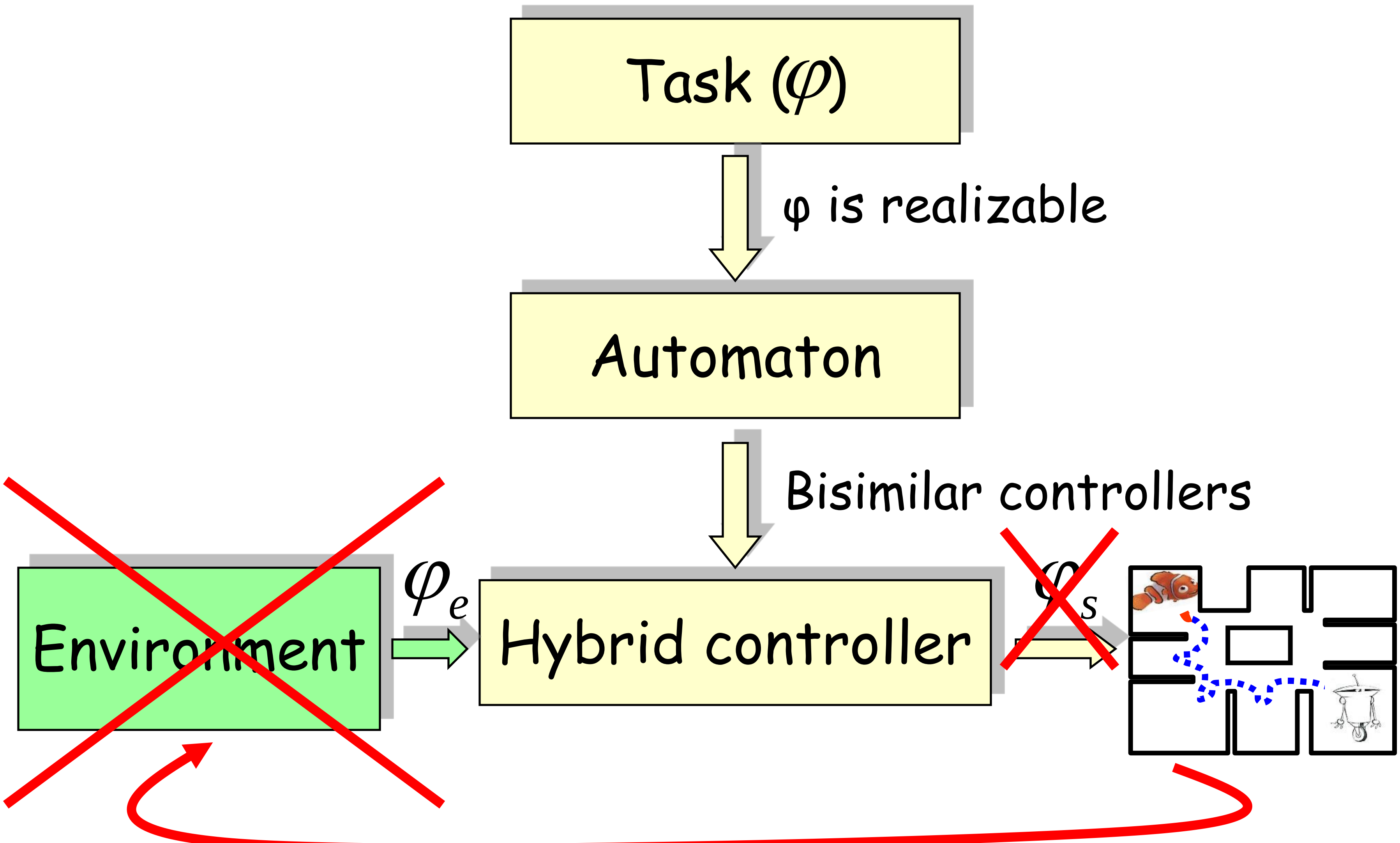
When will this break?



When will this break?



When will this break?



When will this break?

- Logical inconsistency – "go to room 1 & always stay in 4"
- Topologically impossible – "go to room 5 & always avoid room 10"

→ No automaton is synthesized

- Environment behaves badly –
  - Sensors inputs contradict assumptions (φ_e is false)
  - "Violent" environment

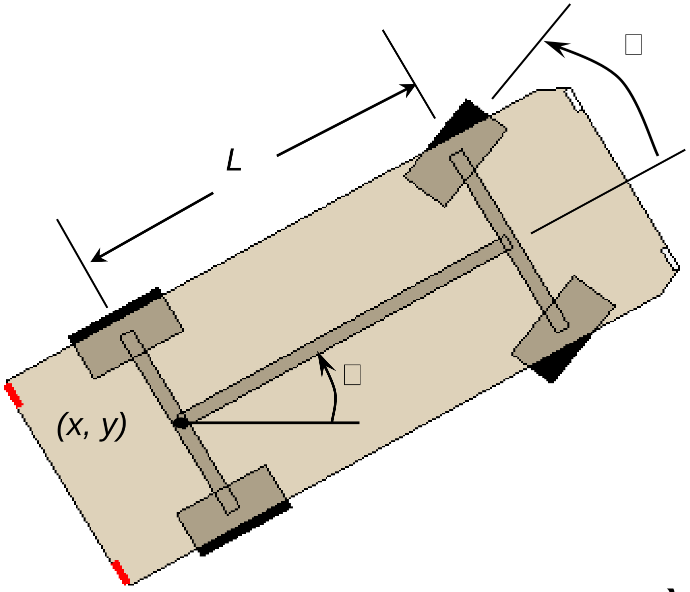
→ Execution may be incorrect or terminated prematurely

$$\dot{p}(t) = f(p(t), u(t)),\ u(t) \in U$$
$$y(t) = h(p(t))$$

Incorporating

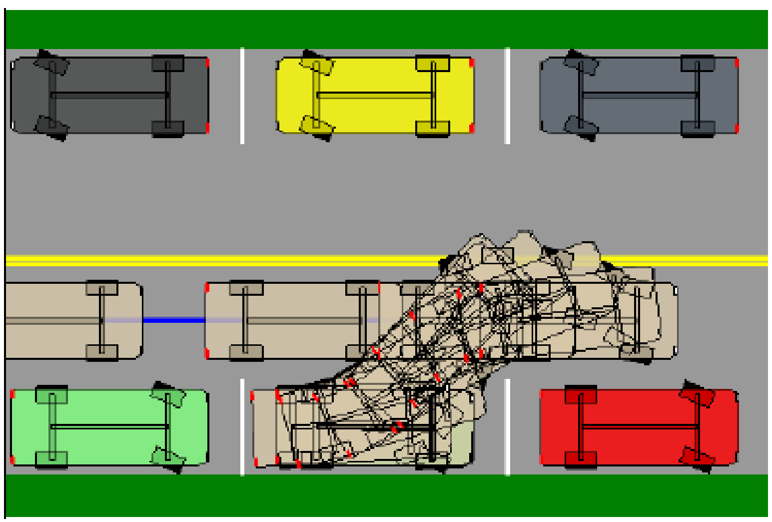
Complex dynamics

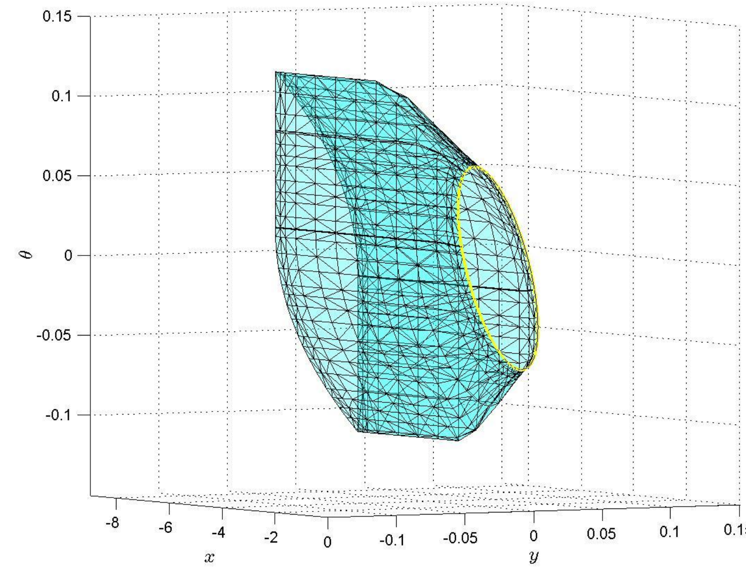
$$\ddot{p}(t) = u(t)$$
$$p(t) \in R^2, u(t) \in U$$

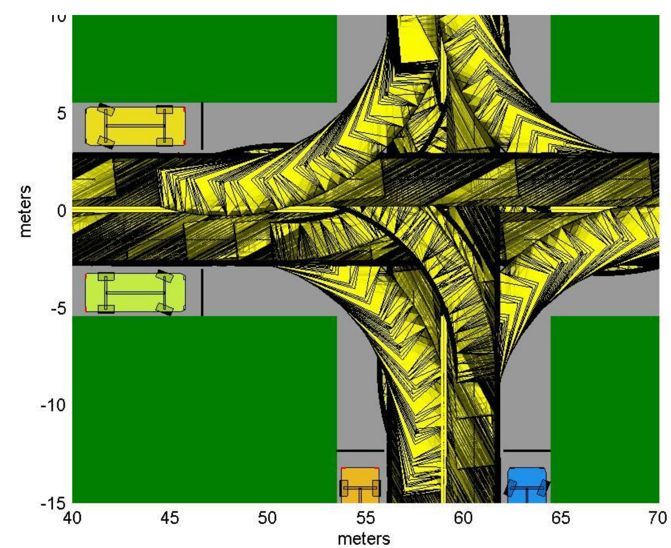


Complex controllers

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \frac{1}{L}\tan\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

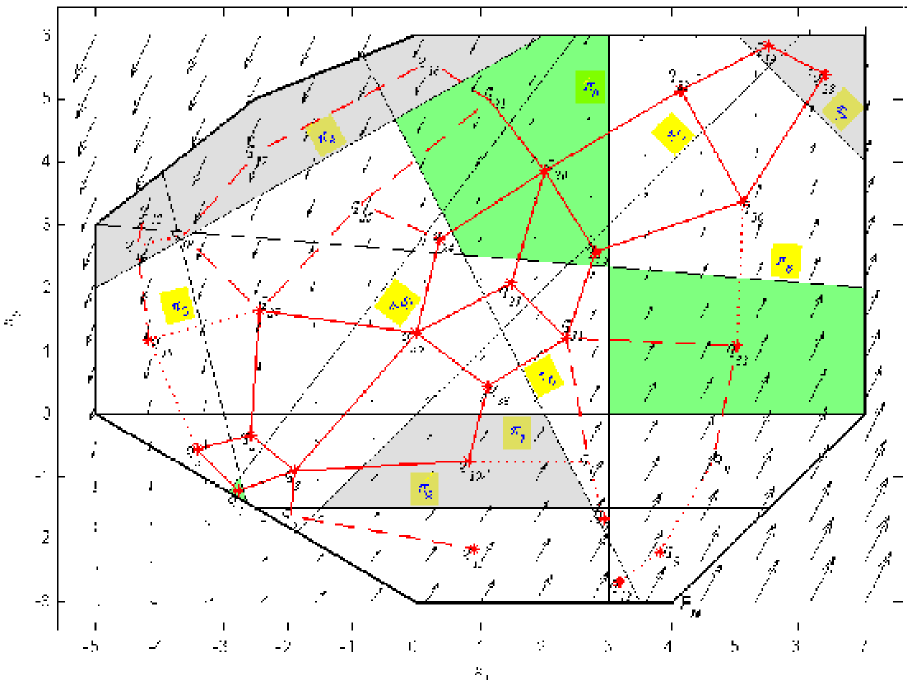


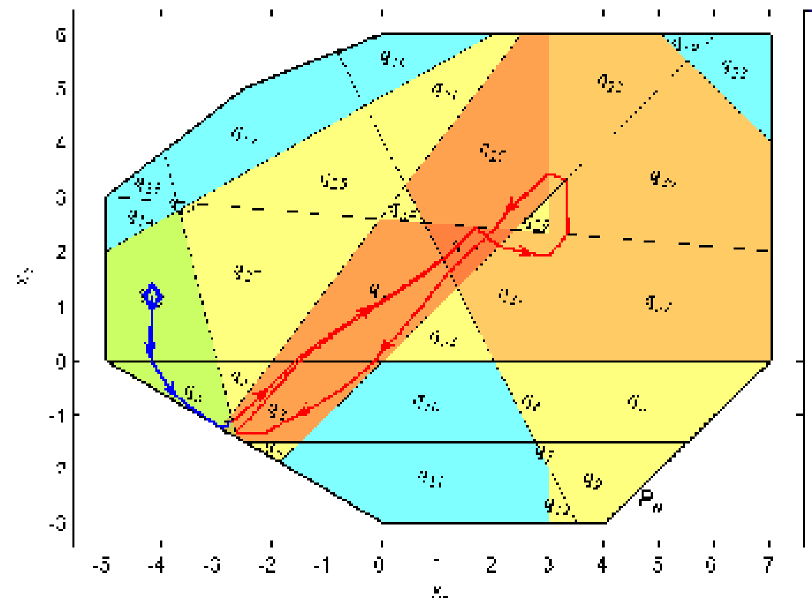




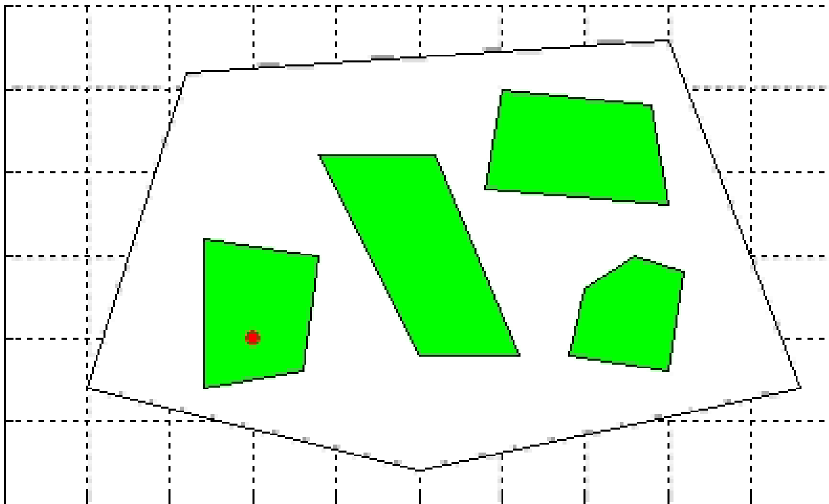
*Images courtesy of David C. Conner

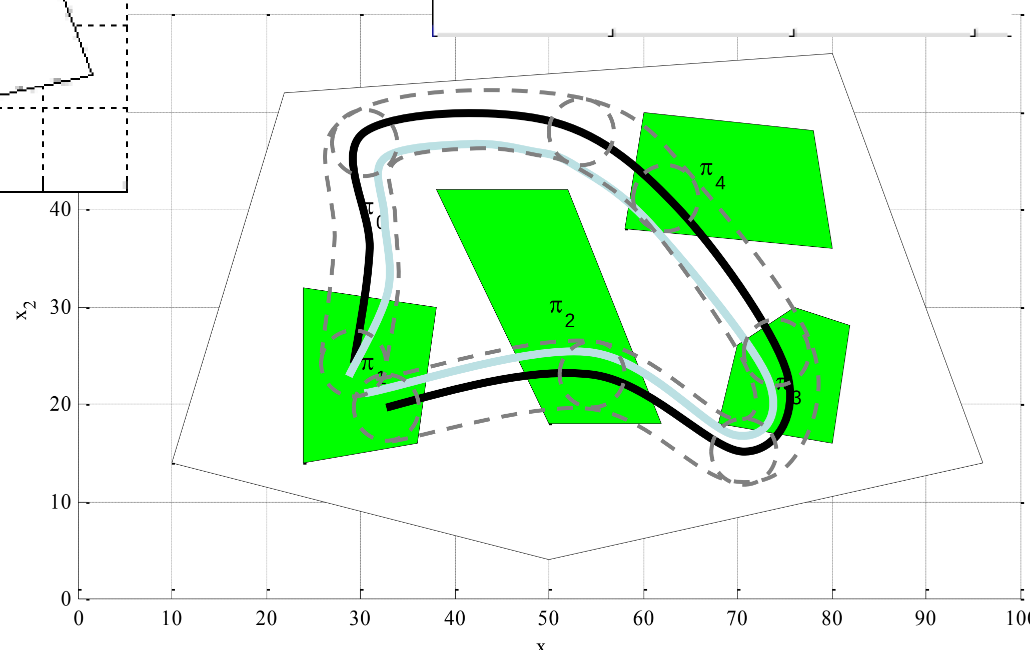
LTLCon: control of linear systems from LTL formulas over linear predicates





Robust LTL

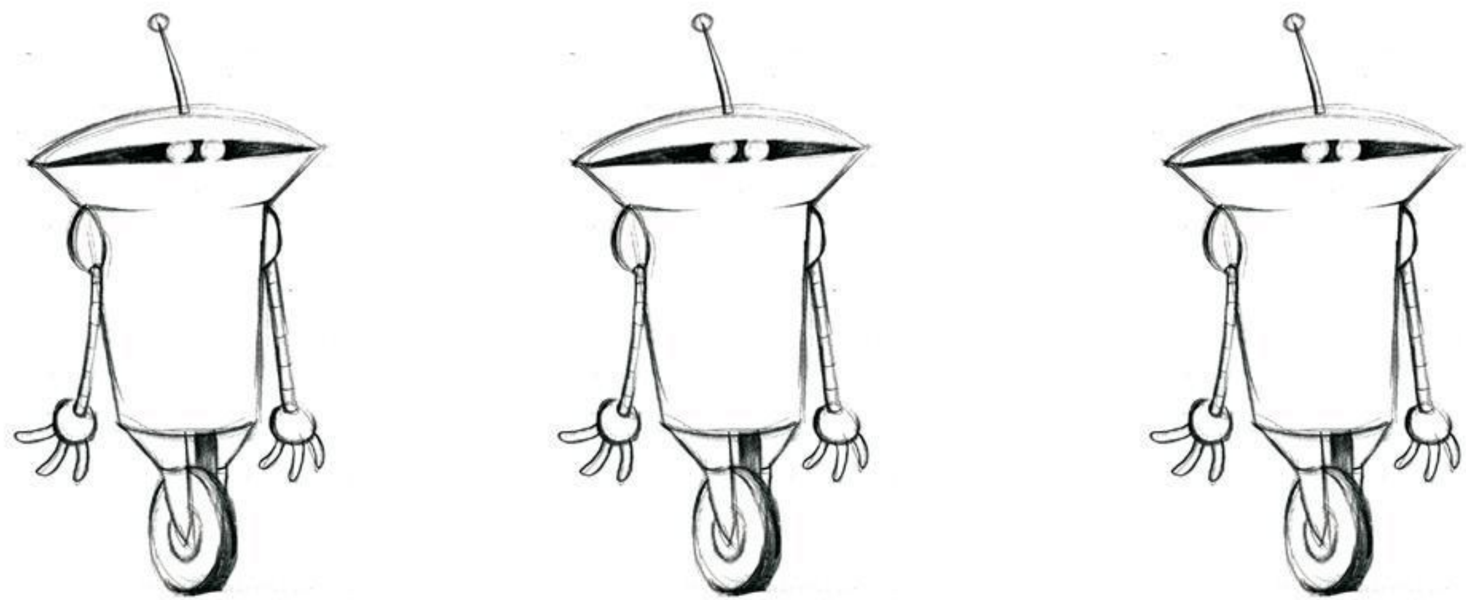




Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, George J. Pappas. *Temporal Logic Motion Planning for Dynamic Robots*, Automatica. To appear.

Handling

Multiple Robots



Decentralized

- Automaton for each robot
- Other robots are a part of the environment

- Scales well*
- Hard to provide global guarantees

Centralized

- One automaton for all robots
- Multi robot controllers

- Global guarantees
- Scales exponentially with the number of robots

Extensions

- Multi Robot
  - Naturally captured in a decentralized way
  - The environment of each robot contains all other robots

  - " If robot 2 is in the kitchen, do not go there"

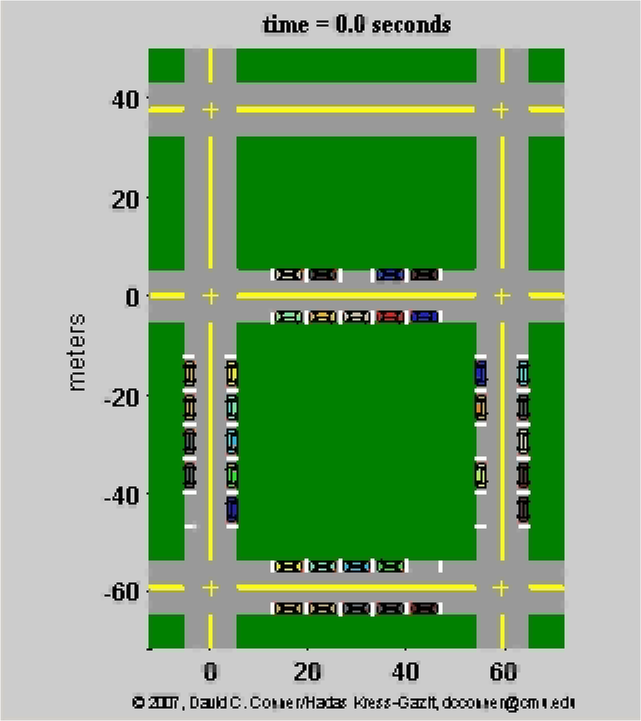
    X = {Robot2Kitchen,… }, Y = {Kitchen, Hall, Bedroom,… }

    $\ldots \wedge \square(\text{Robot2Kitchen} \rightarrow \neg\bigcirc(\text{Kitchen})) \wedge \ldots$

Extensions

- Multi Robot

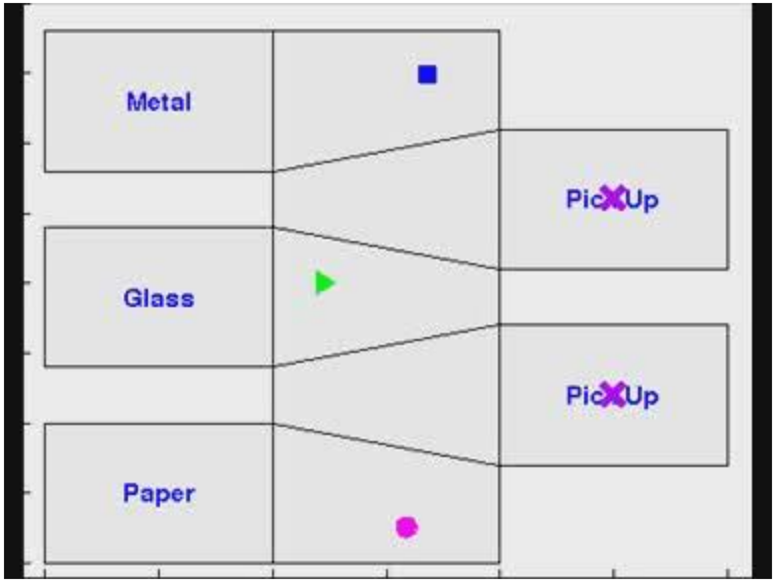
“Drive around the environment, while obeying traffic rules, until you find a free parking space, and then park”



“Leave the block, while obeying traffic rules, through Exit_i”

Multi Robot - Centralized

"Pick up items and sort them according to the material they are made of"



"Nemo may be in one of rooms 1, 3, 5 and 8.
Starting in corridor 12, look for him in these rooms.
If at some point you see him, stop and beep"

Incorporating

Language

$$\varphi = \begin{pmatrix} \neg s_{\mathrm{Nemo}} \\ \wedge [](s_{\mathrm{Nemo}} \rightarrow \mathrm{O}s_{\mathrm{Nemo}}) \\ \wedge []\left(\left(\bigwedge_{i\in\{1,3,5,8\}} \neg r_i\right) \rightarrow \left(s_{\mathrm{Nemo}} \leftrightarrow \mathrm{O}s_{\mathrm{Nemo}}\right)\right) \\ \wedge []\Diamond(True) \end{pmatrix} \rightarrow \begin{pmatrix} r_{12} \bigwedge_{i\neq 12} \neg r_i \wedge \neg Beep \\ \wedge [](r_1 \rightarrow \mathrm{O}r_1 \vee \mathrm{O}r_9) \\ \vdots \\ \wedge [](\mathrm{O}s_{\mathrm{Nemo}} \leftrightarrow \mathrm{O}Beep) \\ \bigwedge_{i\in\{1,3,5,8\}} []\left(\left(r_i \wedge \mathrm{O}s_{\mathrm{Nemo}}\right) \rightarrow \mathrm{O}r_i\right) \\ \bigwedge_{i\in\{1,3,5,8\}} []\Diamond\left(r_i \vee s_{\mathrm{Nemo}}\right) \end{pmatrix}$$

Constructing ϕ

"Nemo may be in one of rooms 1, 3, 5 and 8.
Starting in corridor 12, look for him in these rooms.
If at some point you see him, stop and beep"

(MURI SUBTLE)

Structured English

$$\varphi$$

H. Kress-Gazit, G. E. Fainekos and G. J. Pappas. Translating structured English to robot controllers. Advanced Robotics Special Issue on Selected Papers from IROS 2007. To appear.

Structured English Interface

EnvInit ::= "Environment starts with (ϕ_{env} | false | true) "

RobotInit ::= "Robot starts [in ϕ_{region}] [with ϕ_{action} | with false | with true] "

EnvSafety ::= "Always ϕ_{env}"

RobotSafety ::= "(Always | Always do | Do) ϕ_{robot}"

EnvLiveness ::= "Infinitely often ϕ_{env}"

RobotLiveness ::= "(Go to | Visit | Infinitely often do) ϕ_{robot}"

RobotGoStay ::= "Go to ϕ_{region} and stay [there]"

Conditional ::= "If *Condition* then *Requirement* " | "*Requirement* unless *Condition* " |
| "*Requirement* if and only if *Condition* "

Condition ::= "*Condition* and *Condition*" | "*Condition* or *Condition*" |
| "you (were | are) [not] in ϕ_{region} " |
| "you (sensed | did not sense | are [not] sensing) ϕ_{env} " |
| "you (activated | did not activate | are [not] activating) ϕ_{action}

Requirement ::= *EnvSafety* | *RobotSafety* | *EnvLiveness* | *RobotLiveness* | "stay [there]"

Structured English Interface

“Nemo may be in one of rooms 1, 3, 5 and 8.
Starting in corridor 12, look for him in these rooms.
If at some point you see him, stop and beep”

Environment starts with not Nemo

…

You start in r12

If you are sensing Nemo then stay there

Beep if and only if you are sensing Nemo

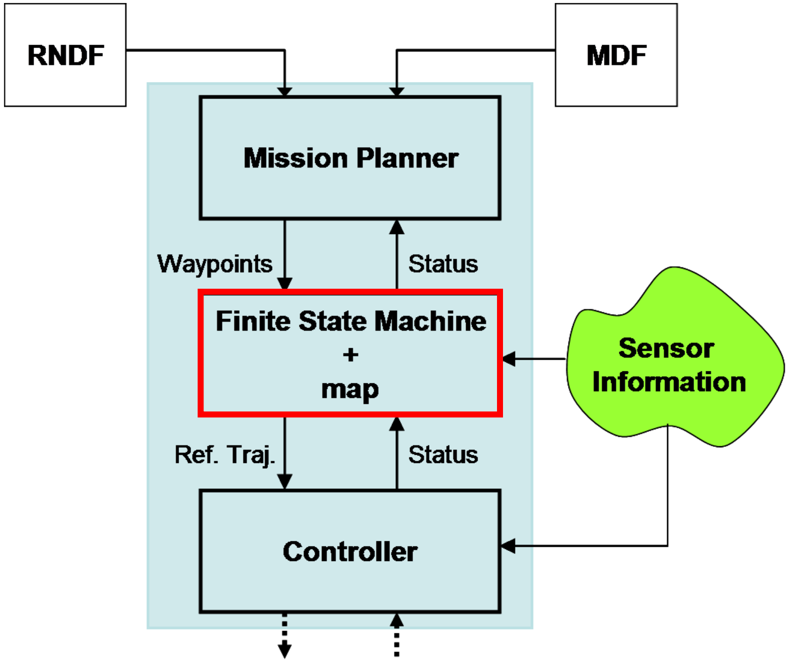
If you are not sensing Nemo then go to r1

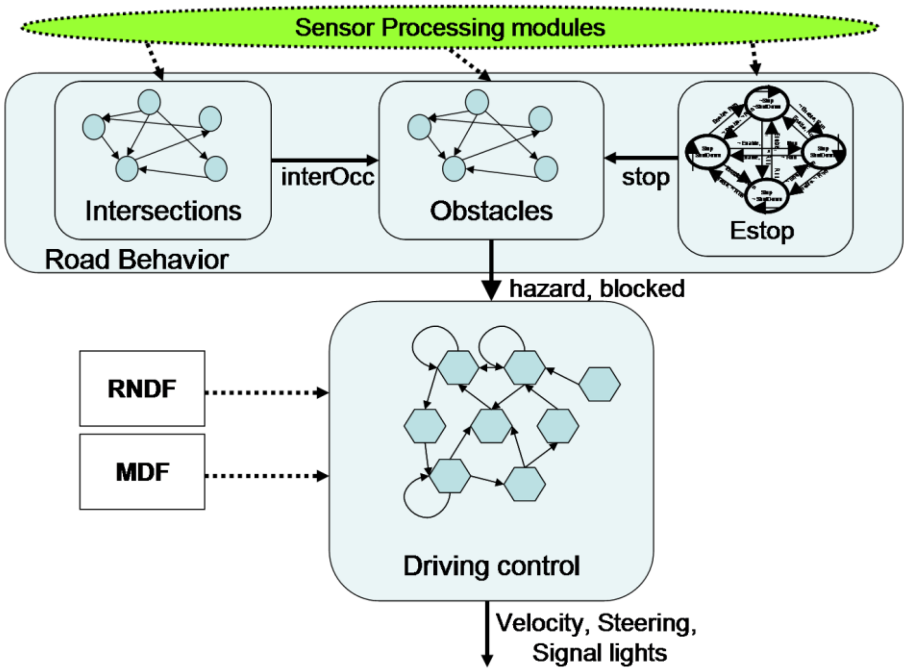
…

Case studies

DARPA's Urban Challenge

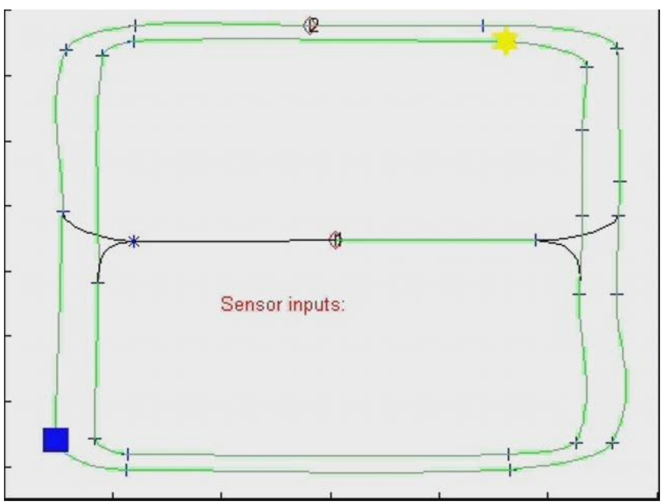
- "Reach sequence of checkpoints while observing traffic laws"





Inputs: Obstacle, leftOcc, leftMoved, Estop, timerUp,…

DARPA's Urban Challenge - NQE

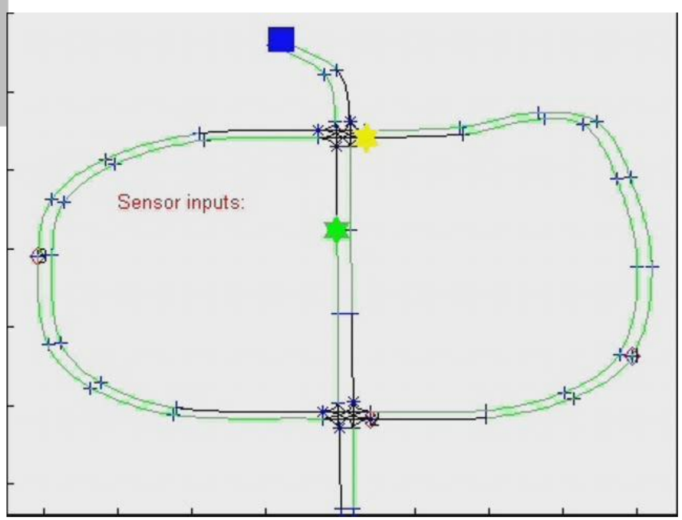


Robot moving

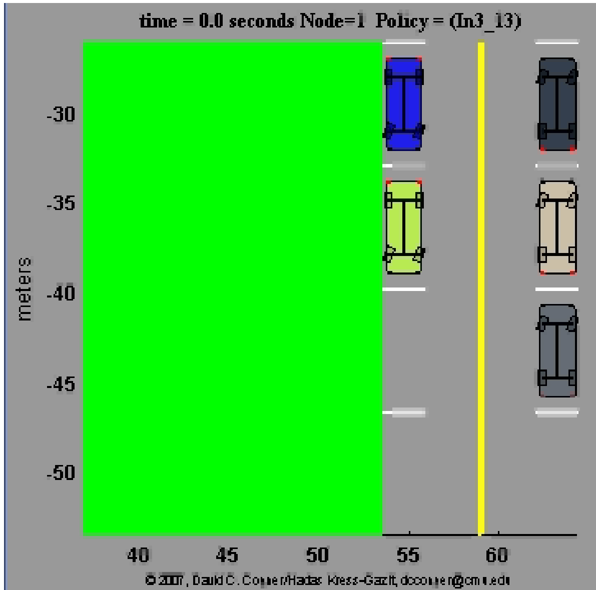
Robot stopping

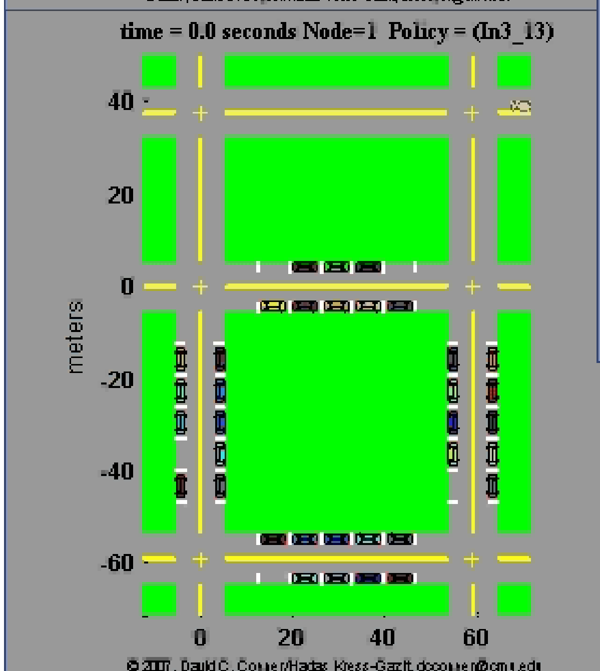
Other vehicles

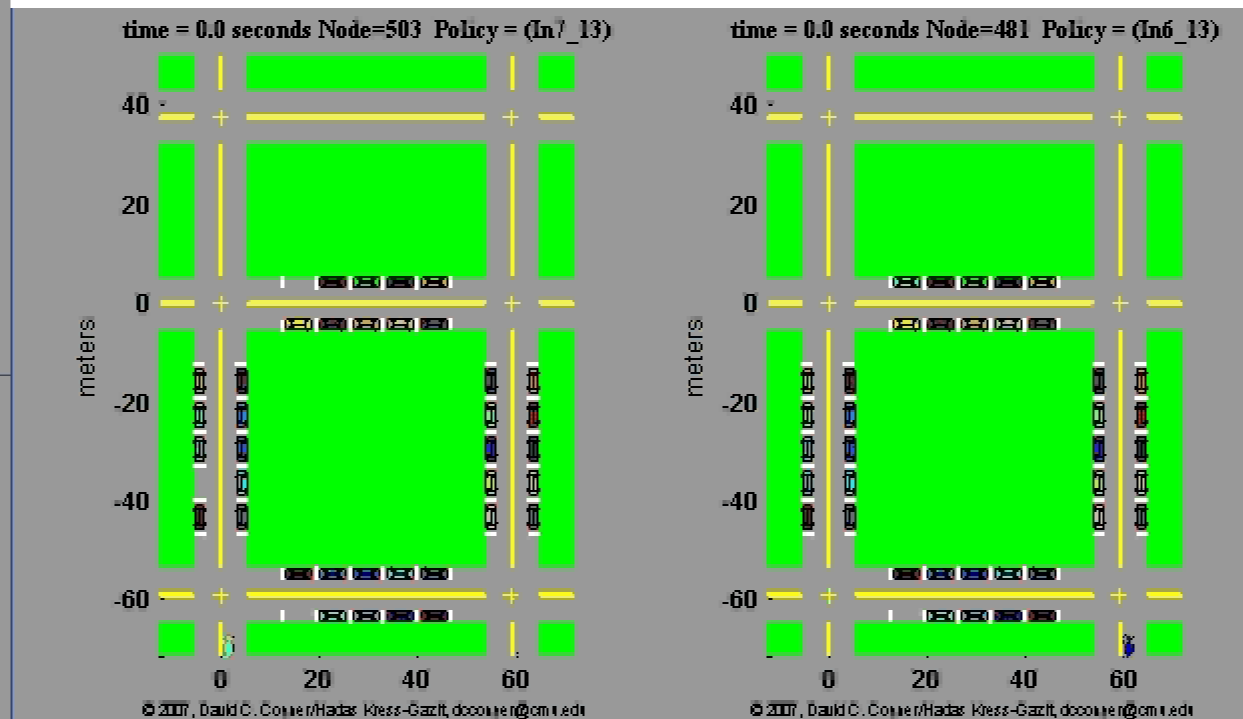
Obstacles



Valet parking







Summary

- Synthesis
- Holy grail: Natural language specifications to “correct by construction” controllers
  - Natural language to temporal logic formula that captures specification, environment assumptions and allowed automaton transitions
  - Synthesize finite state automaton satisfying specs
  - Local hybrid controllers for achieving transitions
- Generalizations to
  - More complex dynamics
  - Multi-robot models
  - Robust specifications