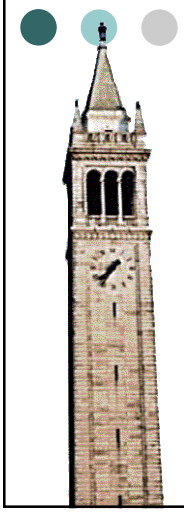


Introduction to Embedded Systems

Edward A. Lee & Sanjit A. Seshia
UC Berkeley
EECS 124
Spring 2008

Lecture 16: Controller Synthesis

Recap of Concepts from Last Time

Synthesis is a Game
between the Robot ("System") and its Environment

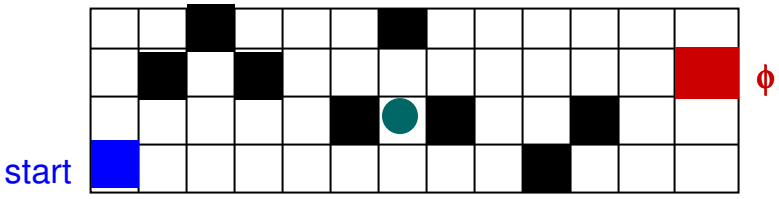
Goal: **F** ϕ
- Robot wins if it reaches ϕ
- Environment wins otherwise
- → Zero-sum game

Goal: **G** $\neg \phi$
- Environment wins if ϕ is always false
- Robot wins otherwise

Controllable State (for the robot)

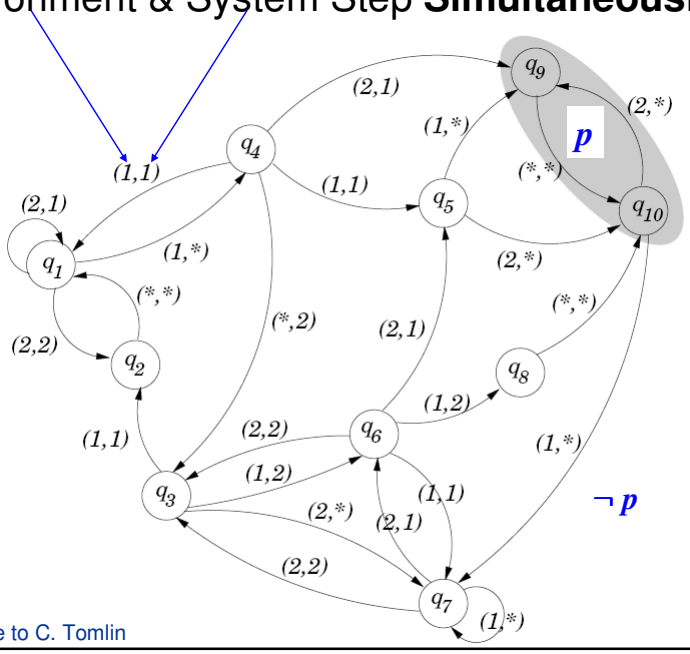
A state from which, no matter what the environment does, the robot can reach its goal.

If there is a start state that is also controllable, then the robot has a winning strategy.



Env_Moves = MAX_MOVES

Environment & System Step **Simultaneously**



Example due to C. Tomlin

Rest of today's lecture

- Discuss synthesis for G F p
- How to synthesize a continuous trajectory

Handling other kinds of Temporal Logic Goals
G F p

Example:
The iRobot must visit the charging station infinitely often

- Consider the FSM formed by
  composing the iRobot FSM with its Environment FSM
- Visualize this FSM as a directed graph
- Suppose that "visiting the charging station" is a state p in this graph

  What graph property corresponds to visiting the state p infinitely often?

Winning Strategy for **G F** p

The system must visit state p infinitely often

For benign environment (optimistic synthesis):

→ The state graph must contain a cycle with state p

→ How can we detect this if we have to build the graph on the fly?

Finding Winning Strategy for **G F** p

The system must visit state p infinitely often

For benign environment (optimistic synthesis):

→ The state graph must contain a cycle with state p

→ How can we detect this if we have to build the graph on the fly?

Two steps:

1. Check if p is reachable from the initial state
2. Check if p is reachable from itself

Winning Strategy for **G F** p: Adversarial Setting

The system must visit state p infinitely often

How do we check this for an adversarial environment?

Perform “Adversarial Reachability”!

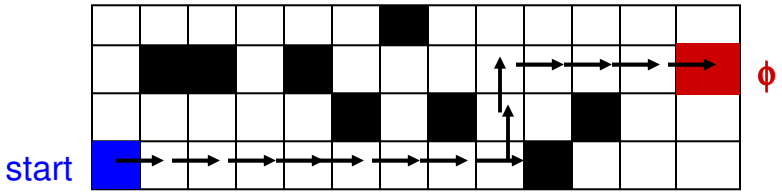
Checking that p is reached infinitely often for an adversarial environment:

Two steps:

1. Check if p is reachable from the initial state,
   no matter what the adversary does
2. Check if p is reachable from itself,
   no matter what the adversary does

For each of these steps, use the algorithm we used on slide 13 of the previous lecture

Synthesizing a Continuous Trajectory



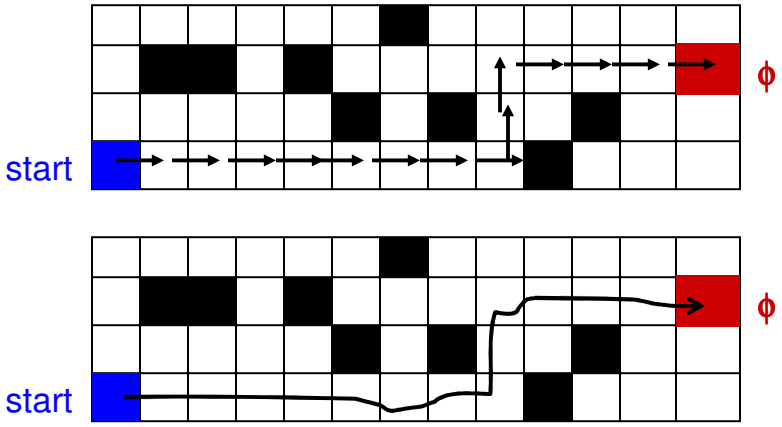
Suppose we have a discrete trajectory (path) to ϕ

How do we transform that into the desired continuous trajectory?

(assume static obstacles)

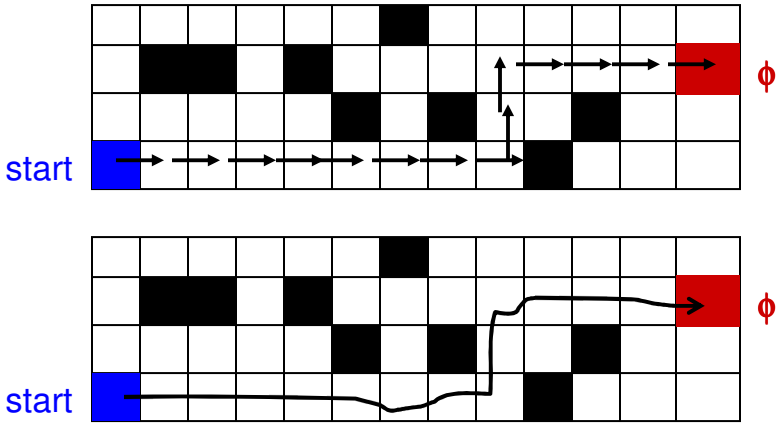
Necessary Condition

If there exists a discrete trajectory, then there must also exist a continuous trajectory



The Other Condition

If there isn't a discrete trajectory, it is possible/OK for a continuous trajectory to exist?



Bisimulation (revisited)

The property we need is bisimulation.

Given:
System = Robot + Environment
The original system H, which is a hybrid system
The discretized version D of H, which is an FSM

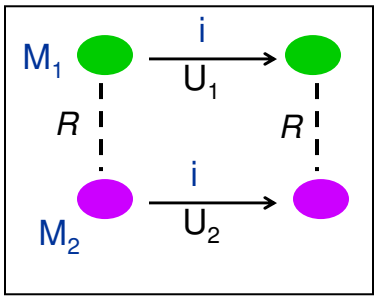
Claim:
If there is a bisimulation between D and H, that suffices to map a trajectory of D to one of H, and vice versa

Bisimulation for FSMs

Let $M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$
where $I = I_1 = I_2$ and $O = O_1 = O_2$

We say M_1 bisimulates M_2 iff
there exists a set $R \subseteq S_1 \times S_2$ such that

1. $R(s_{10}, s_{20})$
2. For all $(s_1, s_2) \in R$, the following conditions hold:
   For all $i \in I$, and $(t_2, o_2) = U_2(s_2, i)$,
   there exists a $(t_1, o_1) = U_1(s_1, i)$ s.t.
   $(t_1, o_1) \in R$ and $o_2 = o_1$
   For all $i \in I$, and $(t_1, o_1) = U_1(s_1, i)$,
   there exists a $(t_2, o_2) = U_2(s_2, i)$ s.t.
   $(t_2, o_2) \in R$ and $o_2 = o_1$



Bisimulation between FSM and Hybrid System (HS)

Given:

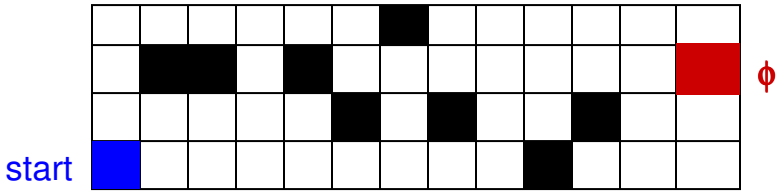
Suppose the FSM M is obtained by partitioning up the continuous state space of the HS H into regions (e.g. rectangles)

- Let the partition be $P : \mathbb{R}^2 \rightarrow Q$

Then M bisimulates H if:

1. If P(x) = P(y), then points x and y are observationally equivalent
2. If P(x) = P(y), then
   for every x' reachable from x, there is a y' reachable from y s.t. P(x') = P(y')
   and vice-versa

Our Example

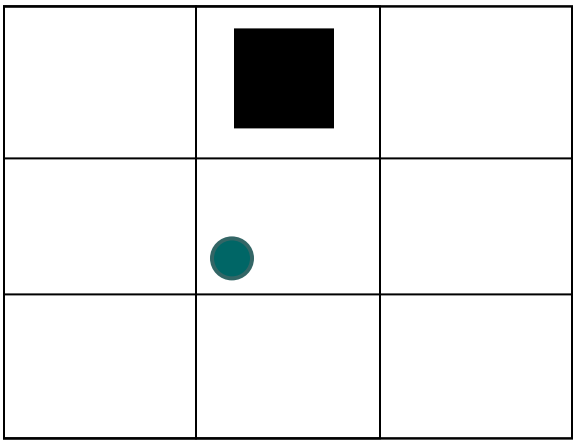


The grid above is a partition P of the 2-D space in the room

When is P a bisimulation?

Zooming In: Bisimulation Condition 1

Sensors should work the same anywhere in a square



Zooming In: Bisimulation Condition 2

Synthesize local control laws that mimic a discrete step from square to adjacent square