Discrete abstractions of continuous systems for control

George J. Pappas
Departments of ESE and CIS
University of Pennsylvania
pappasg@ee.upenn.edu
http://www.seas.upenn.edu/~pappasg

DISC Summer School on
Modeling and Control of Hybrid Systems
Veldhoven, The Netherlands
June 23-26, 2003
http://lcewww.et.tudelft.nl/~disc_hs/

Outline of this mini-course

Lecture 1 : Monday, June 23
Examples of hybrid systems, modeling formalisms

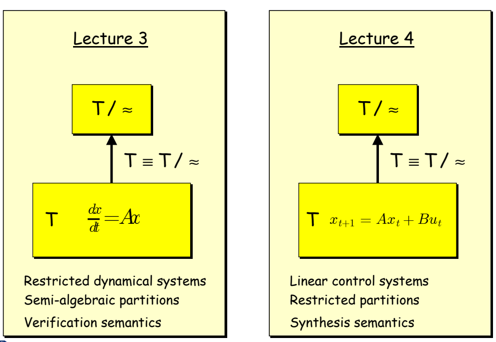
Lecture 2 : Monday, June 23
Transitions systems, temporal logic, refinement notions

Lecture 3 : Tuesday, June 24
Discrete abstractions of hybrid systems for verification

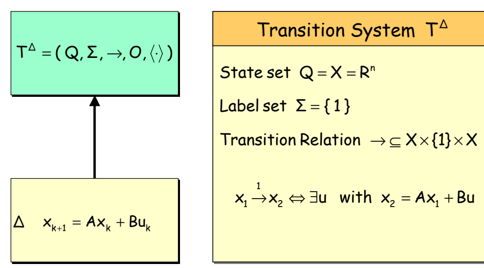
Lecture 4 : Tuesday, June 24
Discrete abstractions of continuous systems for control

Lecture 5 : Thursday, June 26
Bisimilar control systems

Continuous to discrete (Lectures 3 & 4)

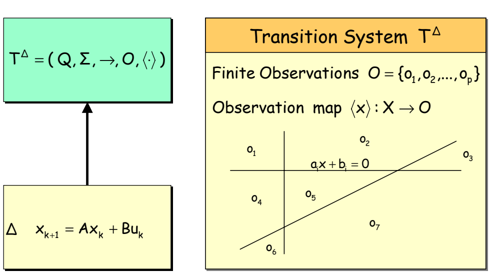


Control abstract transitions



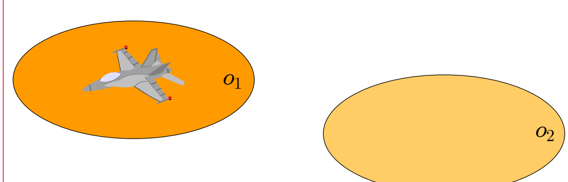
Transitions maintain time but abstract away control
We can safely remove the unique label from all transitions

Finite observations



Observation symbols are atomic propositions of LTL formulas

Temporal logic (LTL) examples



Specification for UAV:

- (Periodicity) Cycle between o_1 and o_2.
- (Coverage) Stay in o_1 for two time steps.
- (Temporal) No more than 4 steps from o_1 to o_2 and vice versa.

$$\Box(o_1 \Rightarrow \Diamond_4 o_2 \wedge o_2 \Rightarrow \Diamond_4(o_1 \wedge \bigcirc o_1))$$

LTL synthesis for control systems

Given linear control system Δ, and temporal formula φ

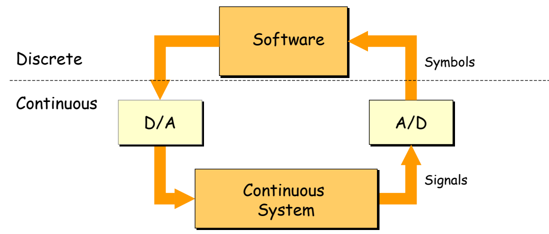
Basic synthesis problem

$$\Delta \;\|\; C \models \varphi$$

Solution for continuous systems can be lifted to hybrid systems.

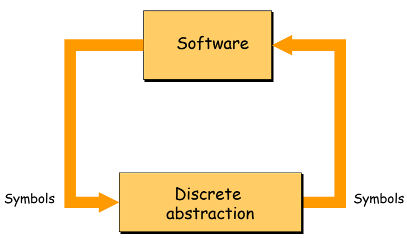
Composition semantics still undefined.

The modern feedback loop



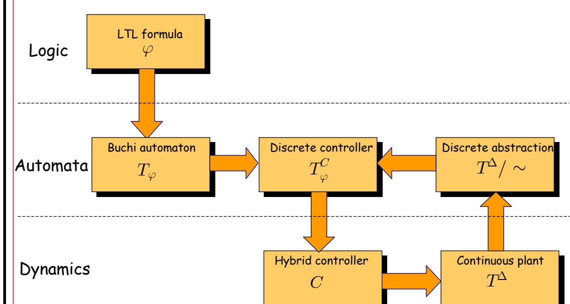
Control design problem : Given formula, design A/D, D/A, software

Software perspective

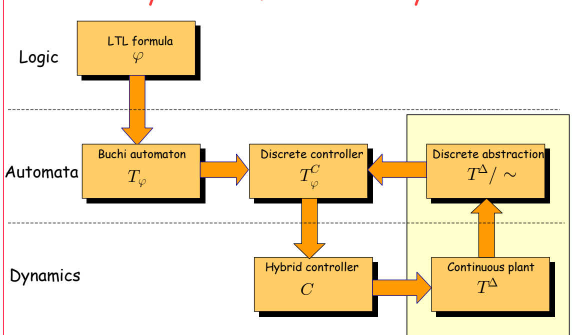


Problem : Extract equivalent (bisimilar) discrete abstraction

LTL synthesis for control systems



LTL synthesis for control systems



Once more

Bisimulation Algorithm

initialize $Q/_\sim = \{p \sim q \text{ iff } <q> = <p>\}$
while $\exists P, P' \in Q/_\sim$ **such that** $\emptyset \neq \subseteq P \cap Pre(P') \neq \subseteq P$
  $P_1 := P \cap Pre(P')$
  $P_2 := P \setminus Pre(P')$
  $Q/_\sim := (Q/_\sim \setminus \{P\}) \cup \{P_1, P_2\}$
end while

Given boolean algebra of sets, Pre is an endomorphism.

Boolean algebras

A Boolean algebra of subsets of $\Re^n$ is a collection $\mathcal{B}(\Re^n)$ of subsets where

$$A, B \in \mathcal{B} \Rightarrow A \cup B \in \mathcal{B}$$
$$A \in \mathcal{B} \Rightarrow \overline{A} \in \mathcal{B}$$

Trivial examples of boolean algebras include $2^{\Re^n}$ $\{\emptyset, \Re^n\}$

Nontrivial examples of boolean algebras include

Rectangular sets: Boolean algebra generated by $x_i \sim c_i \quad c_i \in Q, \quad \sim \in \{>, =, <\}$

Semi-linear sets: Boolean algebra generated by $a^T x \sim c \quad a \in Q^{1 \times n} \quad c \in Q,$

Semi-algebraic sets: Boolean algebra generated by $p(x) \sim 0 \quad \sim \in \{<, =, >\}$

Boolean algebra endomorphisms

A Boolean algebra endomorphism $F : \mathcal{B}(\Re^n) \rightarrow \mathcal{B}(\Re^n)$ is a map satisfying

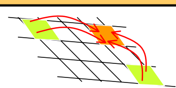
$$F(A \cup B) = F(A) \cup F(B)$$
$$F(\overline{A}) = \overline{F(A)}$$

A Boolean endomorphism is eventually idempotent if $F^{k+1} = F^k$ for some k.

Stable partitions

A partition $\Pi \subseteq \mathcal{B}(\Re^n)$ of $\mathbf{R}^n$ is stable under $F : \mathcal{B}(\Re^n) \rightarrow \mathcal{B}(\Re^n)$ if

$$\forall S_i \in \Pi \quad F(S_i) = \cup_{j \in J} S_j, \quad S_j \in \Pi$$



Bisimulation is a partition that refines observational equivalence and is stable under Pre.

Existence of bisimulations

Let $F : \mathcal{B}(\Re^n) \rightarrow \mathcal{B}(\Re^n)$ be a Boolean algebra endomorphism and $\Pi \subseteq \mathcal{B}(\Re^n)$ a finite partition of $\Re^n$. If F is eventually idempotent, then a finite and stable refinement of Π exists.

Therefore in order to obtain a finite bisimulation we must search for

1. A Boolean algebra of the reals
2. A Pre operator which is endomorphic for the boolean algebra
3. A Pre operator which is eventually idempotent

Controllability implies idempotency

Assume the linear system is completely controllable

$$x_{t+1} = Ax_t + Bu_t$$

Then by definition

$$Pre(Y) = Pre_1(Y) = \{x \in \Re^n \mid \exists y \in Y \, \exists u \quad y = Ax + Bu\}$$

and since the system is controllable

$$\exists k \leq n \qquad Pre^k(Y) = \Re^n$$

and therefore this Pre operator is **eventually idempotent**.

Controllability of linear systems can be decided using rank conditions

$$rank[B \;\; AB \;\; A^2B \; \ldots \; A^{n-1}B] = n$$

Searching for the right boolean algebra

First attempt : Semi-linear sets

Boolean algebra generated by sets of the form

$$a^T x \sim c \quad a \in Q^{1 \times n} \quad c \in Q,$$

Given semi-linear set Y, Pre(Y) is also a semi-linear set

$$F : \mathcal{B}(\Re^n) \rightarrow \mathcal{B}(\Re^n)$$

It is also true that $Pre(A \cup B) = Pre(A) \cup Pre(B)$

However it is NOT true that $Pre(\overline{A}) = \overline{Pre(A)}$

Therefore the Pre is NOT an endomorphism of this Boolean algebra

Searching for the right boolean algebra

Better attempt : Rectangular sets but in Brunovsky coordinates
Boolean algebra generated by sets of the form

$$y_i \sim c_i \quad c_i \in Q, \quad \sim \in \{>, =, <\}$$

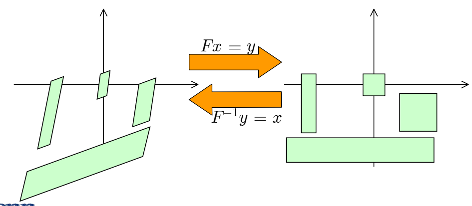
For any completely controllable linear system, there exist invertible linear transformations F and H, and a feedback G such that the resulting system is in Brunovsky normal form.

Original coordinates		Brunovsky coordinates
$x(t+1) = Ax(t) + Bu(t)$	$\begin{bmatrix} y \\ v \end{bmatrix} = U \begin{bmatrix} x \\ u \end{bmatrix}$ $U = \begin{bmatrix} F & 0_{n\times m} \\ G & H \end{bmatrix}$	$\left.\begin{array}{l} y_1(t+1) = y_2(t) \\ y_2(t+1) = y_3(t) \\ y_3(t+1) = v_1(t) \end{array}\right\} = k_1$ $\left.\begin{array}{l} y_4(t+1) = y_5(t) \\ y_5(t+1) = v_2(t) \end{array}\right\} = k_2$

Penn

Brunovsky boolean algebra

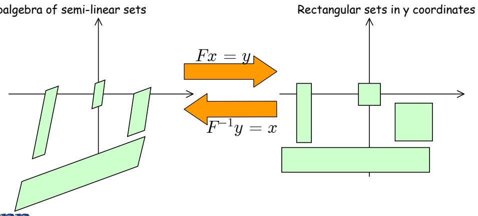
Original coordinates		Brunovsky coordinates
$x(t+1) = Ax(t) + Bu(t)$	$\begin{bmatrix} y \\ v \end{bmatrix} = U \begin{bmatrix} x \\ u \end{bmatrix}$ $U = \begin{bmatrix} F & 0_{n\times m} \\ G & H \end{bmatrix}$	$\left.\begin{array}{l} y_1(t+1) = y_2(t) \\ y_2(t+1) = y_3(t) \\ y_3(t+1) = v_1(t) \end{array}\right\} = k_1$ $\left.\begin{array}{l} y_4(t+1) = y_5(t) \\ y_5(t+1) = v_2(t) \end{array}\right\} = k_2$



Penn

Brunovsky boolean algebra

Brunovksy Boolean Algebra		Algebra in Brunovsky coordinates
$f^T x \sim c_i \quad c_i \in Q, \quad \sim \in \{>, =, <\}$ $f^T \in rows(F)$	$y = Fx$ → ← $x = F^{-1}y$	$w^T y_i \sim c_i \quad c_i \in Q, \quad \sim \in \{>, =, <\}$ $w \in \left\{ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \pm \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \pm \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right\}$
Subalgebra of semi-linear sets		Rectangular sets in y coordinates



Penn

Brunovsky boolean algebra

Brunovsky Boolean Algebra		Algebra in Brunovsky coordinates
$f^T x \sim c_i \quad c_i \in Q, \quad \sim \in \{>, =, <\}$ $f^T \in rows(F)$	$y = Fx$ → ← $x = F^{-1}y$	$w^T y_i \sim c_i \quad c_i \in Q, \quad \sim \in \{>, =, <\}$ $w \in \left\{ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \pm \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \pm \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right\}$
Subalgebra of semi-linear sets		Rectangular sets in y coordinates

Every controllable linear system comes equipped with a Brunovsky boolean algebra

Consider a discrete time controllable linear system in Brunovsky normal form. Then, the Pre operator is a Boolean algebra endomorphism for the Boolean algebra generated by rectangular sets in Bruvonsky coordinates.

Penn

Finite bisimulations of control systems

Consider any discrete time controllable linear system. Then, Pre is a Boolean algebra endomorphism for the Boolean algebra of Brunovsky sets.

\+

Consider any discrete time controllable system. Then the Pre operator is eventually idempotent.

=

Let T^Δ be a transition system associated with a discrete time, controllable linear system. For any initial finite partition Π contained in the Boolean algebra of Brunovsky sets, there exits a finite bisimulation of T^Δ refining partition Π.

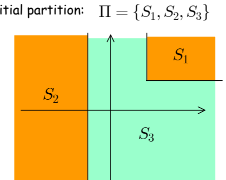
Penn

Example

Control system: $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Change of coordinates: $F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad f_1 = [1 \quad 0] \quad f_2 = [0 \quad 1]$

Initial partition: $\Pi = \{S_1, S_2, S_3\}$



$S_1 = \{x \in \mathbf{R}^2 \ : \ f_1 x - 3 > 0 \wedge f_2 x - 3 > 0\}$

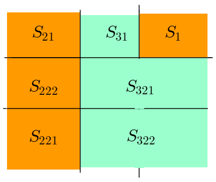
$S_2 = \{x \in \mathbf{R}^2 \ : \ f_1 x + 2 < 0\}$

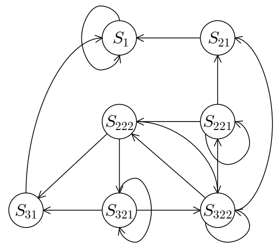
$S_3 = \overline{S_1 \cup S_2}$

Penn

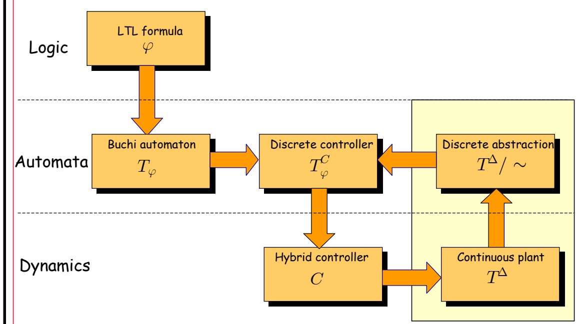
Example

Refined (bisimulation) partition: $\Pi = \{S_1, S_{21}, S_{221}, S_{222}, S_{31}, S_{321}, S_{322}\}$

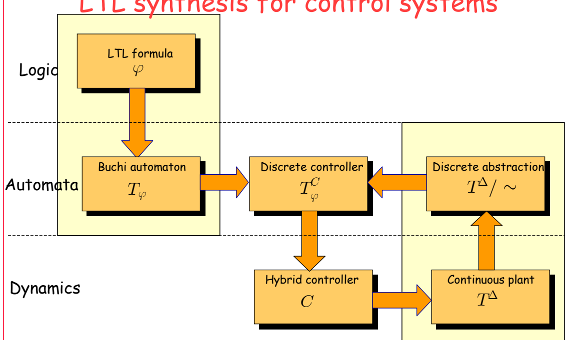




LTL synthesis for control systems



LTL synthesis for control systems



Buchi automata

A Buchi automaton is a finite transition system recognizing infinite strings:

$$T = (Q, Q^0, \longrightarrow, O, <\cdot>, F)$$

The final states $F \subseteq Q$ have to be visited infinitely often by any recognized string.

A string $s = o_1, o_2, o_3, \ldots \in O^\omega$ is recognized by the Buchi automaton if there exits a string $\rho = q_1, q_2, q_3, \ldots \in Q^\omega$ such that:

$$q_1 \in Q^0$$

$$q_i \longrightarrow q_{i+1}, \quad \forall i \in \mathbf{N}$$

$$< q_i >= o_i \quad \forall i \in \mathbf{N}$$

$$\inf(\rho) \cap F \neq \emptyset$$

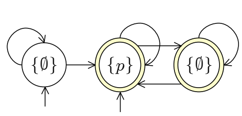
where $\inf(\rho)$ is the set of states visited infinitely often by string .

From LTL to Buchi

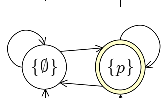
Buchi automata recognize every infinite string satisfying LTL formula φ .

Examples:

$\diamond p$



$\square \diamond p$



Translation from LTL formulas to Buchi automata is automatic

Safety versus liveness

LTL allows to express two different kinds of properties: Safety and Liveness.

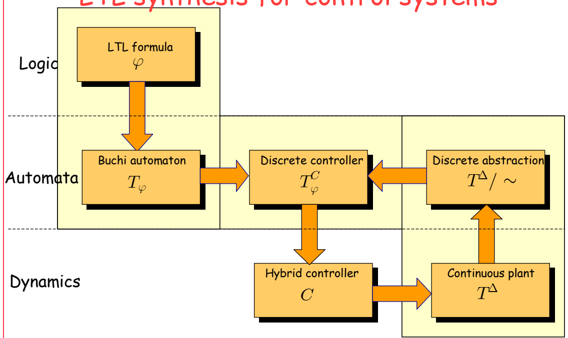
Safety properties require that **bad** things will never happen. We can therefore check if a safety property is violated in finite time. Safety properties can also be modeled by transition systems on finite strings.

Liveness properties require that **good** things will eventually happen. We cannot determine if a liveness property is violated in finite time, since it may hold at some later time in the future. Liveness properties require infinite strings.

Safety properties are closed under union which ensures that maximally permissive controllers for safety properties exist.

Liveness properties are not closed under union which precludes the existence of maximally permissive controllers. However, arbitrarily close approximations can always be found.

LTL synthesis for control systems



Supervisor synthesis

Given a LTL specification formula φ and the finite bisimulation T^Δ of a discrete time controllable linear system, we can use standard supervisory control techniques to obtain a controller T_φ^C for T^Δ ensuring that

$$T_\varphi^C \parallel T^\Delta \models \varphi$$

Recall that every transition in T^Δ is controllable.

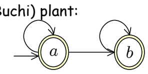
We can, therefore, (naively) define a controller as the Buchi automaton T_φ^C satisfying:

$$L_\omega(T_\varphi^C) = L_\omega(T_\varphi) \cap L_\omega(T^\Delta)$$

If T_φ^C is blocking, search for a non-blocking subsystem of T_φ^C.

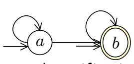
Safety versus liveness

Consider the following (Buchi) plant:



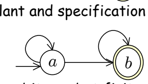
$a^\omega + a^+ b^\omega$

and the LTL specification $\Diamond \Box b$ which can be translated to the automaton:



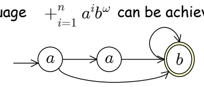
$a^* b^\omega$

The intersection of the plant and specification behaviors is captured by:



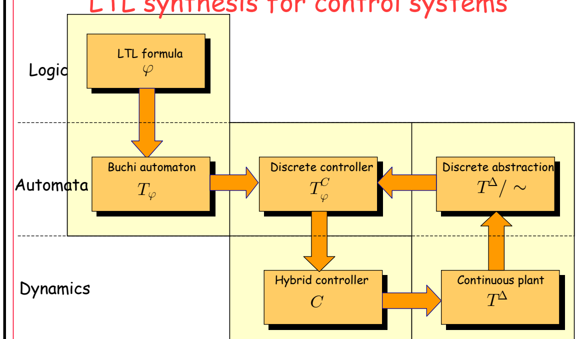
$a^+ b^\omega$

Any supervisor allowing an arbitrary but finite number of occurrences of a also allows an infinite number of such occurrences. However, for any fixed $n > 1$ the language $+_{i=1}^{n} a^i b^\omega$ can be achieved by control:



$ab^\omega + aab^\omega$

LTL synthesis for control systems



From discrete to continuous

Recall that: $T_1 \equiv T_2 \Rightarrow T_1 \models \varphi$ iff $T_2 \models \varphi$

$T_1 \equiv T_2 \Rightarrow T_1 \parallel T \equiv T_2 \parallel T$ for any T

We can thus show that a controller for $T^\Delta/\sim$ is also a controller for T^Δ:

$$T_\varphi^C || (T^\Delta/\sim) \models \varphi$$

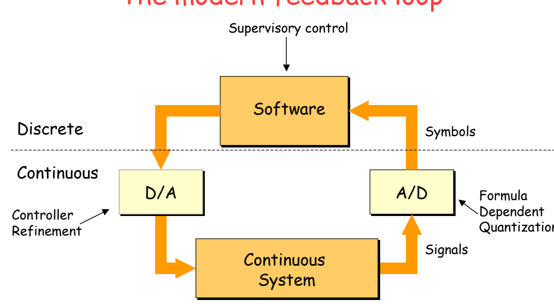
$$T^\Delta \equiv (T^\Delta/\sim) \Rightarrow T_\varphi^C || T^\Delta \equiv T_\varphi^C || (T^\Delta/\sim)$$

$$T_\varphi^C || (T^\Delta/\sim) \models \varphi \Rightarrow T_\varphi^C || T^\Delta \models \varphi$$

Since we can decide the existence of T_C we immediately conclude the following

Consider a discrete time, controllable linear system and an LTL specification formula whose atomic propositions denote sets in the Brunovsky Boolean algebra. Then, the existence of an LTL controller can be decided.

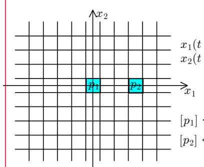
The modern feedback loop



Control design problem : Given formula, design A/D, D/A, software

A Simple Example

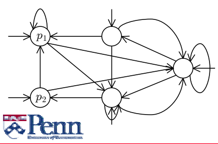
Mobile sensor model:



$$x_1(t+1) = x_2(t)$$
$$x_2(t+1) = u(t)$$

$[p_1] \leftrightarrow (0,0)$

$[p_2] \leftrightarrow (3,0)$

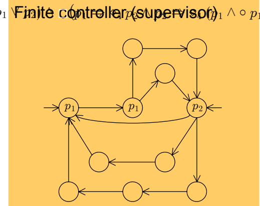


Specifications for mobile sensor:

- Cycle between p_1 and p_2.
- Stay in p_1 for two time steps.
- Take no more than 4 steps to move from p_1 to p_2 and vice versa.

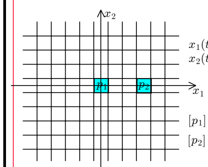
$(p_1 \vee \ldots \wedge \circ p_1))$

Finite controller (supervisor)



A Simple Example

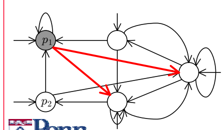
Mobile sensor model:



$$x_1(t+1) = x_2(t)$$
$$x_2(t+1) = u(t)$$

$[p_1] \leftrightarrow (0,0)$

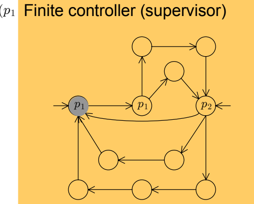
$[p_2] \leftrightarrow (3,0)$



Specifications for mobile sensor:

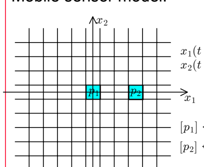
- Cycle between p_1 and p_2.
- Stay in p_1 for two time steps.
- Take no more than 4 steps to move from p_1 to p_2 and vice versa.

$(p_1$ Finite controller (supervisor) $)$



A Simple Example

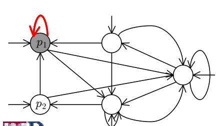
Mobile sensor model:



$$x_1(t+1) = x_2(t)$$
$$x_2(t+1) = u(t)$$

$[p_1] \leftrightarrow (0,0)$

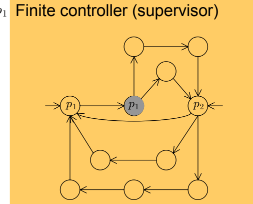
$[p_2] \leftrightarrow (3,0)$



Specifications for mobile sensor:

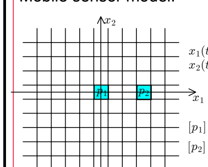
- Cycle between p_1 and p_2.
- Stay in p_1 for two time steps.
- Take no more than 4 steps to move from p_1 to p_2 and vice versa.

$(p_1$ Finite controller (supervisor) $)$



A Simple Example

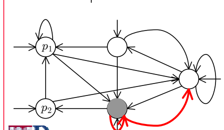
Mobile sensor model:



$$x_1(t+1) = x_2(t)$$
$$x_2(t+1) = u(t)$$

$[p_1] \leftrightarrow (0,0)$

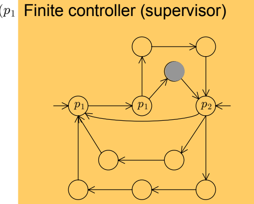
$[p_2] \leftrightarrow (3,0)$



Specifications for mobile sensor:

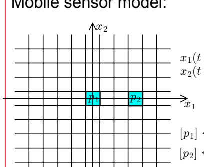
- Cycle between p_1 and p_2.
- Stay in p_1 for two time steps.
- Take no more than 4 steps to move from p_1 to p_2 and vice versa.

$(p_1$ Finite controller (supervisor) $)$



A Simple Example

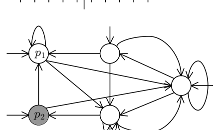
Mobile sensor model:



$$x_1(t+1) = x_2(t)$$
$$x_2(t+1) = u(t)$$

$[p_1] \leftrightarrow (0,0)$

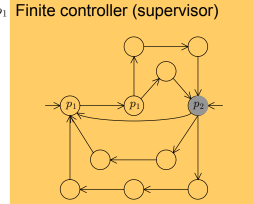
$[p_2] \leftrightarrow (3,0)$



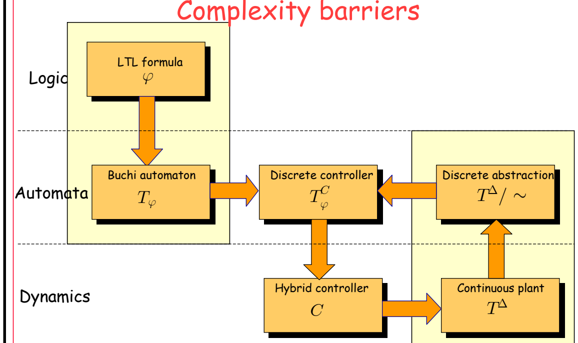
Specifications for mobile sensor:

- Cycle between p_1 and p_2.
- Stay in p_1 for two time steps.
- Take no more than 4 steps to move from p_1 to p_2 and vice versa.

$(p_1$ Finite controller (supervisor) $)$



Complexity barriers



Continuous to continuous (Lecture 5)