Analysis and Control of Partially-Observed Discrete-Event Systems: Introduction and Recent Advances

Xiang Yin

EECS Department, University of Michigan

Department of Automation, Shanghai Jiao-Tong University
May 27, 2016, Shanghai, China
• **Name**: 殷翔  **Born**: Jan 1991, Hefei, Anhui

• **Education**
  - **Zhejiang University**, College of Electrical Engineering  
    Bachelor of Engineering, Major: Power Electronics  
    June 2012
  - **University of Michigan**, Ann Arbor, Department of EECS  
    * Master of Science, Major: Control & Math  
      Dec 2013  
    * PhD Candidate, Major: Control & Math  
      April 2017 (expected)
    * Advisor: Prof. Stephane Lafortune
    * Thesis Committee: D. Teneketzis, D. Tilbury & N. Ozay

• **Research**
  - Control of discrete-event/hybrid systems
  - Model-based fault diagnosis/prognosis
  - Privacy and security in cyber-physical systems
Outline

- Motivation: Why we study discrete-event system
- Partially-Observed Discrete-Event Systems
- Analysis of Partially-Observed DES
  - Verification of Security/Diagnosability/Prognosability
- Control of Partially-Observed DES
  - Synthesis of supervisory control strategies
  - Synthesis of sensor activation strategies
- Applications:
  - Location-Based Services (analysis, security issue)
  - Vehicular Electrical Power Systems (control, safety-critical systems)
- Conclusion and Future Directions
Cyber-Physical Control Systems

- Perception
- Discrete Decision Maker
- Continuous Controller
- Actuator
- Physical Plant
- Sensor

External Dynamic Environment
Cyber-Physical Control Systems

Cyber Layer
- Abstracted Model
- High Level Controller (Supervisor)

Physical Layer
- Continuous Dynamic
- Low Level Controller

- Perception
- Discrete Decision Maker
- Continuous Controller
- Actuator
- Physical Plant
- Sensor
- External Dynamic Environment
physical, continuous

\[ \dot{x}_p = f_p(x_p, u, \eta) \]
\[ s = g_p(x_p, u, \mu) \]
\[ \dot{x}_c = f_p(x_c, s) \]
\[ u = g_p(x_c, s) \]

Model: Differential Equation

Specification: Stability, reference tracking, optimality...
Continuous v.s. Discrete

**physical, continuous**

\[ \dot{x}_p = f_p(x_p, u, \eta) \]
\[ s = g_p(x_p, u, \mu) \]
\[ \dot{x}_c = f_p(x_c, s) \]
\[ u = g_p(x_c, s) \]

**Model:** Differential Equation

**Specification:** Stability, reference tracking, optimality...

**computational, discrete**

\[ S: \text{Obs}(L(G)) \rightarrow 2^E \]

**Model:** Discrete-event systems, automata, transition systems, formal languages

**Specification:** Safety, liveness, diagnosability, security
Current Control Design Process for Cyber-Physical Systems

- Given some spec (plain English) use art of design (engineering intuition, experience) and extensive testing to come up with a single solution
- Ad hoc approaches, Large lists of “if-then-else” rules
- Little or no formal guarantees on correctness
Current Control Design Process for Cyber-Physical Systems

• Given some spec (plain English) use art of design (engineering intuition, experience) and extensive testing to come up with a single solution
• Ad hoc approaches, Large lists of “if-then-else” rules
• Little or no formal guarantees on correctness

Better Alternative

• Formal Methods!
Formal Approach: Verification and Synthesis

Formal Methods
(Model-Based Approach)

- Requirements on the system behavior
- Assumptions on the environment
- System

formal specification
System model

verification
synthesis

Satisfied (+certificate)
Violated (+counterexample)
Controller (Correction Guaranteed)
No such solution

X.Yin (UMich)

SJTU 2016
May 2016
6/31
Formal Approach: Verification and Synthesis

Discrete-event systems

- Model: Automata
- Specification: Formal Languages

Formal Methods
(Model-Based Approach)

- Requirements on the system behavior
- Assumptions on the environment
- System

formal specification
System model

verification
synthesis

Satisfied (+certificate)
Violated (+counterexample)
Controller (Correction Guaranteed)
No such solution
Formal Approach: Verification and Synthesis

Discrete-event systems

- Model: Automata
- Specification: Formal Languages

Verification (Analysis)

- Formal guarantee for specification

Formal Methods
(Model-Based Approach)

- Requirements on the system behavior
- Assumptions on the environment
- System

Formal specification

- Satisfied (+certificate)
- Violated (+counterexample)
- Controller (Correction Guaranteed)
- No such solution

Verification

Synthesis

Discrete-event systems

Model: Automata

Specification: Formal Languages

Requirements on the system behavior

Assumptions on the environment

System

formal specification

Satisfied (+certificate)

Violated (+counterexample)

Controller (Correction Guaranteed)

No such solution

Verification

Synthesis
Formal Approach: Verification and Synthesis

Discrete-event systems
- Model: Automata
- Specification: Formal Languages

Verification (Analysis)
- Formal guarantee for specification

Synthesis (Control Design)
- Reactive to environment, e.g., uncontrollability & unobservability
- Correct-by-construction! (No need to verify)

Formal Methods
(Model-Based Approach)

Requirements on the system behavior
Assumptions on the environment
System

formal specification
System model
verification
synthesis

Satisfied (+certificate) ( +counterexm)
Violated
Controller (Correction Guaranteed)
No such solution

Discrete-event systems

- Model: Automata
- Specification: Formal Languages
Why Discrete-Event Models

Why Discrete-Event Models

- Many systems are **Inherently Event-Driven** and have **Discrete State-Spaces**

Manufacturing Systems, Software Systems, PLCs, Protocols

Why Discrete-Event Models

Many systems are **Inherently Event-Driven** and have **Discrete State-Spaces**

Manufacturing Systems, Software Systems, PLCs, Protocols


DES Model comes from **Finite Abstraction** of the original continuous system

Linear Systems, Nonlinear Systems, Stochastic Systems, Networked Systems

**System Model**

\[ G = (X, E, f, x_0, X_m) \] is a **deterministic** FSA

- \( X \) is the finite set of states
- \( E \) is the finite set of events
- \( f : X \times E \rightarrow X \) is the partial transition function
- \( x_0 \) is the initial state;
- \( X_m \) is the set of marked states.

Plant \( G \)
Discrete-Event Systems

• **System Model**

\[ G = (X, E, f, x_0, X_m) \] is a *deterministic* FSA

- \( X \) is the finite set of states
- \( E \) is the finite set of events
- \( f : X \times E \rightarrow X \) is the partial transition function
- \( x_0 \) is the initial state;
- \( X_m \) is the set of marked states.

• **System’s Behaviors**

- String: a sequence of events, e.g., \( abccab \)....
**System Model**

\[ G = (X, E, f, x_0, X_m) \] is a deterministic FSA

- \( X \) is the finite set of states
- \( E \) is the finite set of events
- \( f : X \times E \rightarrow X \) is the partial transition function
- \( x_0 \) is the initial state;
- \( X_m \) is the set of marked states.

**System’s Behaviors**

- String: a sequence of events, e.g., \( abccab \)....
- Language: a set of strings
Discrete-Event Systems

• **System Model**

\[ G = (X, E, f, x_0, X_m) \] is a deterministic FSA
- \( X \) is the finite set of states
- \( E \) is the finite set of events
- \( f : X \times E \rightarrow X \) is the partial transition function
- \( x_0 \) is the initial state;
- \( X_m \) is the set of marked states.

• **System’s Behaviors**

- String: a sequence of events, e.g., \( abcabc \)...
- Language: a set of strings
- Generated language: \( \mathcal{L}(G) = \{ s \in E^* : f(x_0, s) \} \)
  \[ \mathcal{L}(G) = \{ \epsilon, a, aa, ab, abc, ... \} \]
• **System Model**

\[ G = (X, E, f, x_0, X_m) \] is a **deterministic** FSA
- \( X \) is the finite set of states
- \( E \) is the finite set of events
- \( f : X \times E \rightarrow X \) is the partial transition function
- \( x_0 \) is the initial state;
- \( X_m \) is the set of marked states.

• **System’s Behaviors**

  - String: a sequence of events, e.g., \( abccab \)....
  - Language: a set of strings
  - Generated language: \( \mathcal{L}(G) = \{ s \in E^*: f(x_0, s) \} \)
    \[ \mathcal{L}(G) = \{ \epsilon, a, aa, ab, abc, ... \} \]
Discrete-Event Systems

- **System Model**

  \[ G = (X, E, f, x_0, X_m) \] is a *deterministic* FSA
  - \( X \) is the finite set of states
  - \( E \) is the finite set of events
  - \( f: X \times E \to X \) is the partial transition function
  - \( x_0 \) is the initial state;
  - \( X_m \) is the set of marked states.

- **System’s Behaviors**

  - String: a sequence of events, e.g., \( abccab \), ...
  - Language: a set of strings
  - Generated language: \( \mathcal{L}(G) = \{ s \in E^*: f(x_0, s)! \} \)

\[ \mathcal{L}(G) = \{ e, a, aa, ab, abc, ... \} \]
Discrete-Event Systems

**System Model**

\[ G = (X, E, f, x_0, X_m) \] is a deterministic FSA
- \( X \) is the finite set of states
- \( E \) is the finite set of events
- \( f : X \times E \rightarrow X \) is the partial transition function
- \( x_0 \) is the initial state;
- \( X_m \) is the set of marked states.

**System’s Behaviors**

- String: a sequence of events, e.g., \( abccab \)....
- Language: a set of strings
- Generated language: \( \mathcal{L}(G) = \{ s \in E^*: f(x_0, s) \} \)
  \[ \mathcal{L}(G) = \{ \epsilon, a, aa, ab, abc, ... \} \]
• **Formal Specifications**

  • Safety: Regular language $L_{am}$

  • Non-blockingness: no deadlocks or livelocks

  ![Diagram](image)

  • Other properties: Observation properties, Temporal logics
• Not all behaviors can be observed
  - Internal behavior
  - Limited sensor capability: energy, communication constraint
Partially-Observed Discrete-Event Systems

Not all behaviors can be observed
- Internal behavior
- Limited sensor capability: energy, communication constraint

Observation Model
\[ E = E_o \cup E_{uo} \]

Natural Projection \( P : E^* \rightarrow E_o^* \) erase events in \( E_{uo} \)
- \( E = \{a, b, c\}, E_o = \{a, b\}, P(abcca) = aba \)
- \( P(L(G)) \) is the behavior we can observe
Property Verification of Partially-Observed DES

Does the system satisfy some property?

- **Opacity**: Security and privacy issue in information-flow
- **Diagnosability**: Fault detection and isolation
- **Prognosability**: Fault prediction and alarm
• **Opacity**

The system’s *secret* cannot be revealed based on the intruder’s observation.
Opacity

The system’s secret cannot be revealed based on the intruder’s observation.

Current State Opacity

- A set of secret states $X_s \subseteq X$
- The intruder never know the system is at secret state
- Ex: I know that you are visiting hospital
**K-Step Opacity**

The intruder cannot infer that the system was at a secret state for some specific instant **K-step ahead** in the past.

**Infinite-Step Opacity**

The intruder cannot infer that the system was at a secret state for any specific instant in the past.
**K-Step Opacity**
The intruder cannot infer that the system was at a secret state for some specific instant *K-step ahead* in the past.

**Infinite-Step Opacity**
The intruder cannot infer that the system was at a secret state for any specific instant in the past.

\[ E_o = \{o, a, b\} \]
K-Step Opacity and Infinite-Step Opacity

• **K-Step Opacity**
The intruder cannot infer that the system was at a secret state for some specific instant *K-step ahead* in the past.

• **Infinite-Step Opacity**
The intruder cannot infer that the system was at a secret state for any specific instant in the past.

\[
\hat{X}_{\mid s \mid -0}(o)
\]

\[
\begin{align*}
0 & \rightarrow 1 \rightarrow 2 \rightarrow 3 \\
4 & \rightarrow 5 \rightarrow 6 \rightarrow 7 \\
8 & \rightarrow 9 \rightarrow 10 \rightarrow 11
\end{align*}
\]

\[E_o = \{o, a, b\}\]
K-Step Opacity and Infinite-Step Opacity

- **K-Step Opacity**
  The intruder cannot infer that the system was at a secret state for some specific instant **K-step ahead** in the past.

- **Infinite-Step Opacity**
  The intruder cannot infer that the system was at a secret state for any specific instant in the past.

\[ \hat{X}_{s|-1}(o) \]

\[ E_o = \{o, a, b\} \]
K-Step Opacity and Infinite-Step Opacity

- **K-Step Opacity**
The intruder cannot infer that the system was at a secret state for some specific instant *K-step ahead* in the past.

- **Infinite-Step Opacity**
The intruder cannot infer that the system was at a secret state for any specific instant in the past.

\[
\hat{X}_{|s|-2}(o)
\]

\[
E_o = \{o, a, b\}
\]
**K-Step Opacity**
The intruder cannot infer that the system was at a secret state for some specific instant *K-step ahead* in the past.

**Infinite-Step Opacity**
The intruder cannot infer that the system was at a secret state for any specific instant in the past.

It is not 2-step opaque!
Verification of $K$-Step Opacity and Infinite-Step Opacity

- **Previous Result**
  - K-step opacity can be verified in $O(|E_o| \times 2^{|X|} \times (|E_o| + 1)^K)$  
    [Saboori & Hadjicostis, 2011]
  - Infinite-step opacity can be verified in $O(|E_o| \times 2^{|X|} \times 2^{|X|^2})$  
    [Saboori & Hadjicostis, 2013]
  - Different approaches for different properties
Verification of $K$-Step Opacity and Infinite-Step Opacity

- **Previous Result**
  - K-step opacity can be verified in $O(|E_o| \times 2^{|X|} \times (|E_o| + 1)^K)$ \cite{SabooriHadjicostis2011}.
  - Infinite-step opacity can be verified in $O(|E_o| \times 2^{|X|} \times 2^{|X|^2})$ \cite{SabooriHadjicostis2013}.
  - Different approaches for different properties.

- **Recent Advances**
  - New approach for the verification of K-step and infinite-step opacity.
  - A unified approach based on a separation principle.
  - K-Step: $O(|E_o| \times 2^{|X|} \times \min\{|E_o|^K, 2^{|X|}\})$ vs $O(|E_o| \times 2^{|X|} \times (|E_o| + 1)^K)$.
  - Infinite-Step: $O(|E_o| \times 2^{|X|} \times 2^{|X|})$ vs $O(|E_o| \times 2^{|X|} \times 2^{|X|^2})$.


Location-Based Services

- Provide services to mobile users by exploiting their location information
- Finding nearby restaurants, tracking users’ running routes, etc.
- May not be secure!
Application of Opacity: Location-Based Services

Location-Based Services

• Provide services to mobile users by exploiting their location information
• Finding nearby restaurants, tracking users’ running routes, etc.
• May not be secure!

Attack Model for the Intruder

• Is located at the LBS server
• Has mobility patterns of users
• Receives location information in LBS queries
Application of Opacity: Location-Based Services

Application of Opacity: Location-Based Services

Is state 6 (cancer center) opaque?
• No! Consider string $cdd$

**Diagnosability** [Sampath, et al, 1995]

The occurrence of any fault event can be *detected* unambiguously within a finite delay.

**Prognosability** [Genc & Lafortune, 2009, Kumar & Takai, 2011]

The occurrence of any fault event can be *predicted* with no miss-alarm and no false-alarm.
**Diagnosability** [Sampath, et al, 1995]

The occurrence of any fault event can be *detected* unambiguously within a finite delay.

**Prognosability** [Genc & Lafortune, 2009, Kumar & Takai, 2011]

The occurrence of any fault event can be *predicted* with no miss-alarm and no false-alarm.
**Diagnosability** [Sampath, et al, 1995]

The occurrence of any fault event can be *detected* unambiguously within a finite delay.

**Prognosability** [Genc & LaFortune, 2009, Kumar & Takai, 2011]

The occurrence of any fault event can be *predicted* with no miss-alarm and no false-alarm.
**Diagnosability** [Sampath, et al, 1995]

The occurrence of any fault event can be *detected* unambiguously within a finite delay.

**Prognosability** [Genc & Lafortune, 2009, Kumar & Takai, 2011]

The occurrence of any fault event can be *predicted* with no miss-alarm and no false-alarm.

Not diagnosable if we cannot see event a
Diagnosability [Sampath, et al, 1995]
The occurrence of any fault event can be \textit{detected} unambiguously within a finite delay.

Prognosability [Genc & Lafortune, 2009, Kumar & Takai, 2011]
The occurrence of any fault event can be \textit{predicted} with no miss-alarm and no false-alarm.

Recent Advances

- Diagnosability and observability are equivalent
  - \textbf{X. Yin} and S. Lafortune, “Codiagnosability and coobservability under dynamic observations: transformation and verification.” \textit{Automatica}, vol.61, pp. 241-252, 2015. (Regular Paper)

- Performance and reliability issue in decentralized fault prognosis
What if Verification Fails?
- For example: LBS example
• What if Verification Fails?
  - For example: LBS example

• Synthesis!
  - Synthesis of *supervisory control strategies*
  - Synthesis of *sensor activation strategies*
Property Enforcement via Supervisory Control

Observation:
\[ E = E_o \cup E_{uo} \]

Supervisor:
\[ E = E_c \cup E_{uc}, E_{uc} \text{ uncontrollable events (environment)} \]

Disable events in \( E_c \) based on its observations

System Property
\[ S(s) \]

Observation Property
\[ P(s) \]
• **System Property**
  
  - Safety: never visited illegal states
  - Non-blockingness: no deadlocks or livelocks
Formal Specifications

• **System Property**
  - Safety: never visited illegal states
  - Non-blockingness: no deadlocks or livelocks

• **Observation Property**
  - Opacity, Diagnosability, Prognosability, Observability
• **System Property**
  - Safety: never visited illegal states
  - Non-blockingness: no deadlocks or livelocks

• **Observation Property**
  - Opacity, Diagnosability, Prognosability, Observability

• **Maximal Permissiveness**
  - Optimality criterion is set inclusion.
    Only disable an event if absolutely necessary
Formal Specifications

- **System Property**
  - Safety: never visited illegal states
  - Non-blockingness: no deadlocks or livelocks

- **Observation Property**
  - Opacity, Diagnosability, Prognosability, Observability

- **Maximal Permissiveness**
  - Optimality criterion is set inclusion. Only disable an event if absolutely necessary

Standard Supervisory Control [Ramadge & Wonham, 1980s]
## Property Enforcing Supervisory Control Problem

<table>
<thead>
<tr>
<th>Property</th>
<th>Safety</th>
<th>Opacity</th>
<th>Diagnosability</th>
<th>Detectability</th>
<th>Anonymity</th>
<th>Attractability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous Work</td>
<td>[1]-[3]</td>
<td>[4],[5]</td>
<td>[6]</td>
<td>[7]</td>
<td>None</td>
<td>[8]</td>
</tr>
<tr>
<td>Previous Assumptions</td>
<td>None</td>
<td>$E_a \subseteq E_o$</td>
<td>$E_c \subseteq E_o$</td>
<td>$E_c \subseteq E_o$</td>
<td>N/A</td>
<td>$E_c \subseteq E_o$</td>
</tr>
</tbody>
</table>

[1] [Lin and Wonham, 1988]  
[2] [Cieslak et al., 1988]  
[3] [Ben Hadj-Alouane et al., 1996]  
[4] [Dubreil et al., 2010]  
[5] [Saboori and Hadjicostis, 2011]  
[6] [Sampath et al., 1998]  
[7] [Shu and Lin, 2013]  
[8] [Schmidt and Breindl, 2014]
## Property Enforcing Supervisory Control Problem

### Previous Work

<table>
<thead>
<tr>
<th>Property</th>
<th>Safety</th>
<th>Opacity</th>
<th>Diagnosability</th>
<th>Detectability</th>
<th>Anonymity</th>
<th>Attractability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous Work</td>
<td>[1]-[3]</td>
<td>[4],[5]</td>
<td>[6]</td>
<td>[7]</td>
<td>None</td>
<td>[8]</td>
</tr>
</tbody>
</table>

### Previous Assumptions

<table>
<thead>
<tr>
<th>Property</th>
<th>Safety</th>
<th>Opacity</th>
<th>Diagnosability</th>
<th>Detectability</th>
<th>Anonymity</th>
<th>Attractability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous Assumptions</td>
<td>None</td>
<td>$E_a \subseteq E_o$</td>
<td>$E_c \subseteq E_o$</td>
<td>$E_c \subseteq E_o$</td>
<td>N/A</td>
<td>$E_c \subseteq E_o$</td>
</tr>
</tbody>
</table>

### Our Assumption

<table>
<thead>
<tr>
<th>Property</th>
<th>Safety</th>
<th>Opacity</th>
<th>Diagnosability</th>
<th>Detectability</th>
<th>Anonymity</th>
<th>Attractability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Assumption</td>
<td>None</td>
<td>$E_a = E_o$</td>
<td>None</td>
<td>None</td>
<td>$E_a = E_o$</td>
<td>None</td>
</tr>
</tbody>
</table>

[1] [Lin and Wonham, 1988]
[2] [Cieslak et al., 1988]
[3] [Ben Hadj-Alouane et al., 1996]
[4] [Dubreil et al., 2010]
[5] [Saboori and Hadjicostis, 2011]
[6] [Sampath et al., 1998]
[7] [Shu and Lin, 2013]
[8] [Schmidt and Breindl, 2014]

## A Uniform Approach

**X. Yin** and S. Lafortune, “A uniform approach for synthesizing property-enforcing supervisors for partially-observed DES.” *IEEE Transactions Automatic Control*, vol.61, no.8, 2016. *(Regular Paper)*
• Information State: a set of states; \( I = 2^X \).
• State Estimate: all possible states consistent with observation

- Supervisor \( S \) disables nothing
- \( I(o) = \{3, 4\}, I(oo) = \{5, 6\} \)

\[
E_c = \{c_1, c_2\}, \ E_o = \{o\}
\]
A Uniform Approach for Property Enforcement

- Information State: a set of states; $I = 2^X$.
- State Estimate: all possible states consistent with observation
- Information-State Based Property: $\varphi: 2^X \rightarrow \{0,1\}$
- It contains: safety, opacity, diagnosability, detectability, attractability, anonymity, etc.

$E_c = \{c_1, c_2\}, E_o = \{o\}$

- Supervisor $S$ disables nothing
- $I(o) = \{3,4\}, I(oo) = \{5,6\}$
A Uniform Approach for Property Enforcement

- Information State: a set of states; \( I = 2^X \).
- State Estimate: all possible states consistent with observation
- Information-State Based Property: \( \varphi : 2^X \rightarrow \{0,1\} \)
- It contains: safety, opacity, diagnosability, detectability, attractability, anonymity, etc.

\[ \varphi(i) = 0 \iff i \cap BAD \neq \emptyset \]

- Supervisor \( S \) disables nothing
- \( I(o) = \{3,4\}, I(oo) = \{5,6\} \)

\[ E_c = \{c_1, c_2\}, E_o = \{o\} \]
A Uniform Approach for Property Enforcement

- **Information State**: a set of states; \( I = 2^X \).
- **State Estimate**: all possible states consistent with observation.
- **Information-State Based Property**: \( \varphi: 2^X \rightarrow \{0,1\} \).
- It contains: safety, opacity, diagnosability, detectability, attractability, anonymity, etc.

**Key Result:**

Any IS-based property can be enforced by an IS-based supervisor.

- \( E_c = \{c_1, c_2\}, E_o = \{ o \} \)
- **Supervisor** \( S \) disables nothing.
- \( I(o) = \{3,4\}, I(oo) = \{5,6\} \)
A Uniform Approach for Property Enforcement

**Basic Idea:** Construct an information structure that captures all possible controlled behaviors of the system

**All Inclusive Controller:**
- A “Game” between environment and controller
- Two kinds of states: Y-states and Z-states
- It embeds (infinite many) solutions in its finite structure

\[ E_c = \{ c_1, c_2 \}, E_o = \{ 0 \} \]
A Uniform Approach for Property Enforcement

- **Basic Idea:** Construct an information structure that captures all possible behaviors

- **All Inclusive Controller:**
  - A “Game” between environment and controller
  - Two kinds of states: Y-states and Z-states
  - It embeds (infinite many) solutions in its finite structure

- **Synthesis:** Pick a *locally maximal* control decision at each Y-state

\[ E_c = \{c_1, c_2\}, E_o = \{0\} \]
A Uniform Approach for Property Enforcement

- **Basic Idea:** Construct an information structure that captures all possible behaviors

- **All Inclusive Controller:**
  - A “Game” between environment and controller
  - Two kinds of states: Y-states and Z-states
  - It embeds (infinite many) solutions in its finite structure

- **Synthesis:** Pick a *locally maximal* control decision at each Y-state

\[
E_c = \{c_1, c_2\}, \quad E_o = \{0\}
\]
A Uniform Approach for Property Enforcement

**Basic Idea:** Construct an information structure that captures all possible behaviors

**All Inclusive Controller:**
- A “Game” between environment and controller
- Two kinds of states: Y-states and Z-states
- It embeds (infinite many) solutions in its finite structure

**Synthesis:** Pick a *locally maximal* control decision at each Y-state

\[
E_c = \{c_1, c_2\}, E_o = \{0\}
\]
## Standard Supervisory Control Problem

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Safe+Max</th>
<th>Safe+NB</th>
<th>Safe+NB+Max</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centralized Upper Bound</strong></td>
<td>[1],[2],[3]</td>
<td>[4]</td>
<td>[5]</td>
<td>OPEN</td>
</tr>
<tr>
<td><strong>Centralized Range</strong></td>
<td>[1],[2],[3]</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td><strong>Decentralized Upper Bound</strong></td>
<td>[2],[6]</td>
<td>OPEN</td>
<td>Undecidable [7],[8]</td>
<td>Undecidable</td>
</tr>
<tr>
<td><strong>Decentralized Range</strong></td>
<td>[2],[6]</td>
<td>OPEN</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>

[1] Lin and Wonham, 1988
[2] Cieslak et al., 1988
[6] Rudie and Wonham, 1992
[8] Thistle, 2005
# Standard Supervisory Control Problem

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Safe+Max</th>
<th>Safe+NB</th>
<th>Safe+NB+Max</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centralized Upper Bound</strong></td>
<td>[1],[2],[3]</td>
<td>[4]</td>
<td>[5]</td>
<td>Solved</td>
</tr>
<tr>
<td><strong>Centralized Range</strong></td>
<td>[1],[2],[3]</td>
<td><strong>Solved</strong></td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td><strong>Decentralized Upper Bound</strong></td>
<td>[2],[6]</td>
<td>OPEN</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td><strong>Decentralized Range</strong></td>
<td>[2],[6]</td>
<td>OPEN</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>

[1] [Lin and Wonham, 1988]  
[2] [Cieslak et al., 1988]  
[3] [Rudie and Wonham, 1990]  
[4] [Ben Hadj-Alouane et al., 1996]  
[5] [Yoo and Lafortune, 2006]  
[6] [Rudie and Wonham, 1992]  
[7] [Tripakis, 2004]  
[8] [Thistle, 2005]

## Recent Advances

**X. Yin** and S. Lafortune, “Synthesis of maximally permissive supervisors for partially observed DES."  
*IEEE Transactions Automatic Control*, vol.61, no.5, 2016. *(Regular Paper)*

**X. Yin** and S. Lafortune. "Synthesis of maximally-permissive supervisors for the range control problem,"  
### Standard Supervisory Control Problem

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Safe+Max</th>
<th>Safe+NB</th>
<th>Safe+NB+Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>[1],[2],[3]</td>
<td>[4]</td>
<td>[5]</td>
<td><strong>Solved</strong></td>
</tr>
<tr>
<td>Upper Bound</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralized</td>
<td>[1],[2],[3]</td>
<td><strong>Solved</strong></td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decentralized</td>
<td>[2],[6]</td>
<td>OPEN</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Upper Bound</td>
<td></td>
<td>OPEN</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Decentralized</td>
<td>[2],[6]</td>
<td>OPEN</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- [1] [Lin and Wonham, 1988]
- [2] [Cieslak et al., 1988]
- [3] [Rudie and Wonham, 1990]
- [4] [Ben Hadj-Alouane et al., 1996]
- [5] [Yoo and Lafortune, 2006]
- [6] [Rudie and Wonham, 1992]
- [7] [Tripakis, 2004]
- [8] [Thistle, 2005]

### Recent Advances

**X. Yin** and S. Lafortune, "Synthesis of maximally permissive supervisors for partially observed DES." *IEEE Transactions Automatic Control*, vol.61, no.5, 2016. (Regular Paper)

Non-blocking Control Problem

- **Observation:** $2^X$ is not sufficient to make a decision

$$E_c = \{c_1, c_2\}, E_o = \{0\}$$
Non-blocking Control Problem

- **Observation:** $2^X$ is not sufficient to make a decision

$$E_c = \{c_1, c_2\},\ E_0 = \{0\}$$
Non-blocking Control Problem

- Observation: $2^X$ is not sufficient to make a decision

$$E_c = \{c_1, c_2\}, E_o = \{0\}$$
Non-blocking Control Problem

- **Observation:** $2^X$ is not sufficient to make a decision

\[ E_c = \{c_1, c_2\}, E_o = \{0\} \]
Non-blocking Control Problem

- **Observation:** $2^X$ is not sufficient to make a decision
- **Basic Idea:** unfold the solution space until it converges
- **Key Result:** We need additional, but finite, information

$$E_c = \{c_1, c_2\}, E_o = \{o\}$$
Non-blocking Control Problem

- **Observation**: $2^X$ is not sufficient to make a decision
- **Basic Idea**: unfold the solution space until it converges
- **Key Result**: We need additional, but finite, information

$$E_c = \{c_1, c_2\}, E_o = \{o\}$$
Non-blocking Control Problem

- Observation: $2^X$ is not sufficient to make a decision
- Basic Idea: unfold the solution space until it converges
- Key Result: We need additional, but finite, information

$$E_c = \{c_1, c_2\}, E_o = \{0\}$$
Non-blocking Control Problem

- **Observation:** $2^X$ is not sufficient to make a decision
- **Basic Idea:** unfold the solution space until it converges
- **Key Result:** We need additional, but finite, information

$$E_c = \{c_1, c_2\}, E_o = \{0\}$$
Non-blocking Control Problem

- **Observation:** $2^X$ is not sufficient to make a decision
- **Basic Idea:** unfold the solution space until it converges
- **Key Result:** We need additional, but finite, information
Centralized Sensor Activation Problem

Sensor activation policy
A function that determines which events to monitor next

Dynamic Sensor Activation Problem
Find a sensor activation policy $\omega$ such that
- some property can be guaranteed
- It is optimal: numerical (average cost) or logical (set inclusion)
Centralized Sensor Activation Problem

Sensor activation policy
A function that determines which events to monitor next

Dynamic Sensor Activation Problem
Find a sensor activation policy $\omega$ such that
- some property can be guaranteed
- It is optimal: numerical (average cost) or logical (set inclusion)

A Fault Diagnosis Problem

• Static Sensors: always observe $a$ and $b$
Centralized Sensor Activation Problem

Sensor activation policy
A function that determines which events to monitor next

Dynamic Sensor Activation Problem
Find a sensor activation policy $\omega$ such that
- some property can be guaranteed
- It is optimal: numerical (average cost) or logical (set inclusion)

A Fault Diagnosis Problem
- **Static Sensors**: always observe $a$ and $b$
- **Dynamic Sensors**: observe both $a$ and $b$ initially
  - turn off all sensors after seeing $a$ or $b
Centralized Sensor Activation Problem

Sensor activation policy
A function that determines which events to monitor next

Dynamic Sensor Activation Problem
Find a sensor activation policy $\omega$ such that
- some property can be guaranteed
- It is optimal: numerical (average cost) or logical (set inclusion)

A Fault Diagnosis Problem
- **Static Sensors**: always observe $a$ and $b$
- **Dynamic Sensors**: observe both $a$ and $b$ initially
  - turn off all sensors after seeing $a$ or $b$

Recent Advances
- A general approach for solving sensor activation problem
- A new structure called the Most Permissive Observer
- A minimal sensor activation policy can be synthesized from the MPO

Decentralized Sensor Activation Problem

Decentralized Diagnosis Problem
- Large-scale systems
- Plant is monitored by multiple agents

Synthesis Problem
- Synthesis of local sensor activation strategies for each agent such that they are diagnose the fault as a group

Solution Approach
- Person-by-person approach
- Iteration converge finitely
- It is an optimal solution

Apply Synthesis Techniques to Vehicular Electrical Power Systems

Assumption

- Generators cannot fail at the same time
- Only one failure/recovery occurs within $T_{max}$
- A control action takes time $t_{trf}$
Apply Synthesis Techniques to Vehicular Electrical Power Systems

Assumption
- Generators cannot fail at the same time
- Only one failure/recovery occurs within $T_{max}$
- A control action takes time $t_{trf}$

Specification
- Generators paralleling is not allowed, i.e., no bus should be powered by more than one generators at the same time
- Buses should not be unpowered for more than $T_{max}$
Apply Synthesis Techniques to Vehicular Electrical Power Systems

Assumption
• Generators cannot fail at the same time
• Only one failure/recovery occurs within $T_{max}$
• A control action takes time $t_{trf}$

Specification
• Generators paralleling is not allowed, i.e., no bus should be powered by more than one generators at the same time
• Buses should not be unpowered for more that $T_{max}$

Large-Scale System is Decentralized!
Apply Synthesis Techniques to Vehicular Electrical Power Systems

When the system is huge
• Safety-critical system
• Intuition is hard to handle
• Need formal synthesis techniques!

An aircraft EPS: Honeywell Inc. patent
Apply Synthesis Techniques to Vehicular Electrical Power Systems

When the system is huge
- Safety-critical system
- Intuition is hard to handle
- Need formal synthesis techniques!

Our Results
- Build DES Model: the state-space is already discrete; discretize time
- Apply supervisor synthesis technique developed
- Algorithm implemented by Alloy*, an efficient model finder embedding SAT solver (On going)
Summary

• Recent Advances on the verification and synthesis of partially-observed DES
• Verification: Opacity, Diagnosability, Prognosability
• Synthesis
  - Supervisory Control Strategies: a uniform approach & non-blockingness
  - Sensor Activation Strategies: centralized/decentralized solutions
• Two Applications: LBS and EPS
Summary

• Recent Advances on the verification and synthesis of partially-observed DES
• Verification: Opacity, Diagnosability, Prognosability
• Synthesis
  - Supervisory Control Strategies: a uniform approach & non-blockingness
  - Sensor Activation Strategies: centralized/decentralized solutions
• Two Applications: LBS and EPS

Future Directions

• More Properties: Temporal Logic, LTL, CTL*..., (Bi)Simulation
• More Models: Petri nets, Stochastic DES (Markov chains)
• More Applications to Cyber-Physical Systems:
  SCADA systems (PLC), Intelligent transportation systems, Cyber-security
References