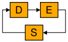
Monitoring and Control of Discrete Event Systems: Some Key Results and Recent Research

Stéphane Lafortune

EECS Department
University of Michigan, USA

Plenary Lecture at *Congresso Brasileiro de Automática CBA'2008*
Juiz de Fora, Brazil

17 September 2008



Outline of Talk

- Discrete Event Systems (DES): The Big Picture
- *Part 1*- Control Problem
- *Part 2*- Diagnosis Problem
- *Part 3*- Active Sensing Problem
- Conclusion

Acknowledgments

1- Control / 2- Diagnosis / 3- Active Sensing

- Collaborators
  - *Brazil:*
    1. Patrícia Nascimento Pena (UFMG)
       José Eduardo Ribeiro Cury (UFSC)
       Antonio Eduardo Carrilho da Cunha (IME-RJ)
       Max Hering de Queiroz (UFSC)
    2. João Carlos Basilio (UFRJ)
  - *Michigan:*
    1. Dawn Tilbury (UM)
    2. Demosthenis Teneketzis (UM)
    3. Feng Lin (Wayne State U.)
  - *Students:*
    1. Richard Hill (U. Detroit-Mercy)
    2. Many...
    3. Weilin Wang (UM Post-Doc)

Acknowledgments

- Financial Support:
  - National Science Foundation (USA)
  - Office of Naval Research (USA)
  - HP Labs.
  - Xerox Corp.

Discrete Event Systems: The Big Picture

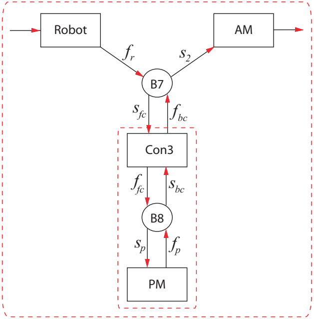
What are Discrete Event Systems?

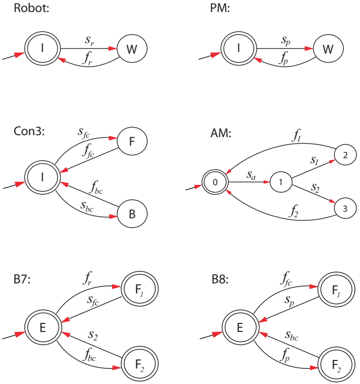
- Discrete State Spaces
- Event-driven Dynamics

Baggage Handling Systems - Beijing Airport *(Siemens)*

Discrete Event Systems: The Big Picture

How Do We Model DES? → Answer 1: **Automata**





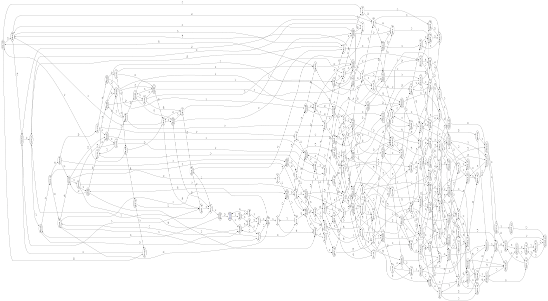
Discrete Event Systems: The Big Picture

How Do We Obtain the Complete System?

Parallel Composition of Automata: $||$ *Common Events*

184 reachable states (out of $2 \times 2 \times 3 \times 4 \times 3 \times 3 = 432$)

482 transitions



DESUMA Software Tool

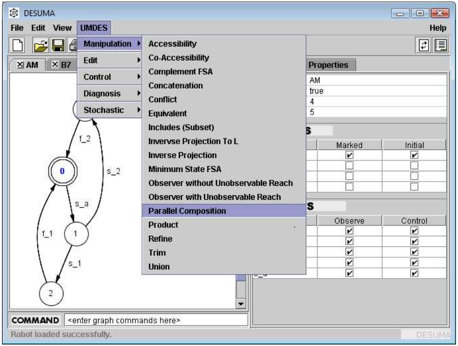


Figure: DESUMA menu for manipulation of automata

DESUMA Software Tool

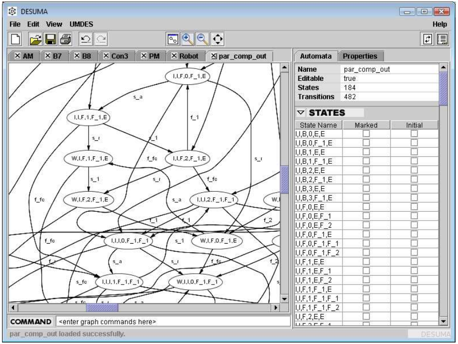
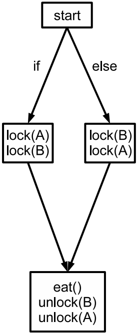
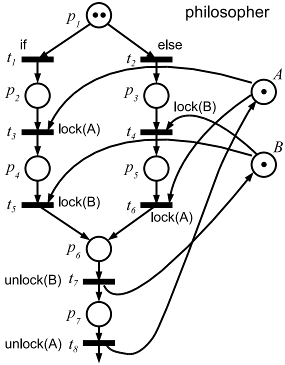


Figure: Small FMS automaton

Discrete Event Systems: The Big Picture

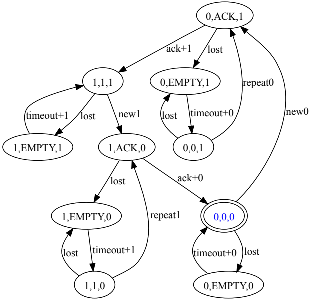
How Do We Model DES? → Answer 2: **Petri Nets**





Discrete Event Systems: Untimed or Logical Behavior

- Automaton: G
- Event Set of G: E
- Set of trajectories of G:
  - **Language** $\mathcal{L}(G)$
  - string/trace: $s \in \mathcal{L}(G)$
- Set of *marked* trajectories of G:
  - **Marked Language** $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$
  - completed operations/tasks



Discrete Event Systems: Logical Properties

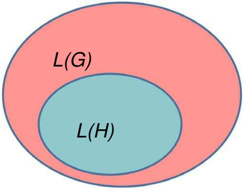
Safety:

- no illegal *states* reached
- no illegal *substrings* executed
- Formally: Specification automaton H

$$\mathcal{L}(H) \subseteq \mathcal{L}(G)$$

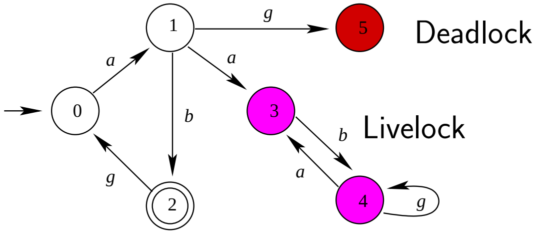
$$\mathcal{L}_m(H) = \mathcal{L}(H) \cap \mathcal{L}_m(G) \subseteq \mathcal{L}_m(G)$$

w.l.o.g.: think of H as a subautomaton of G



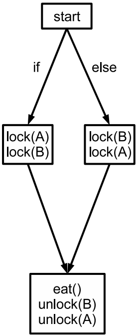
Discrete Event Systems: Logical Properties

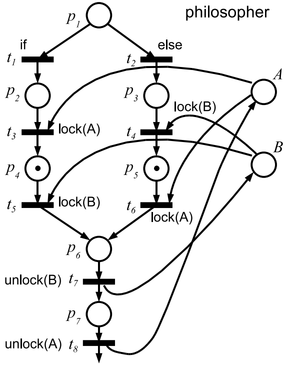
Nonblocking: no deadlocks or livelocks



Discrete Event Systems: Logical Properties

Deadlock in Petri Nets:

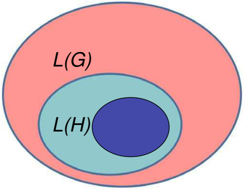




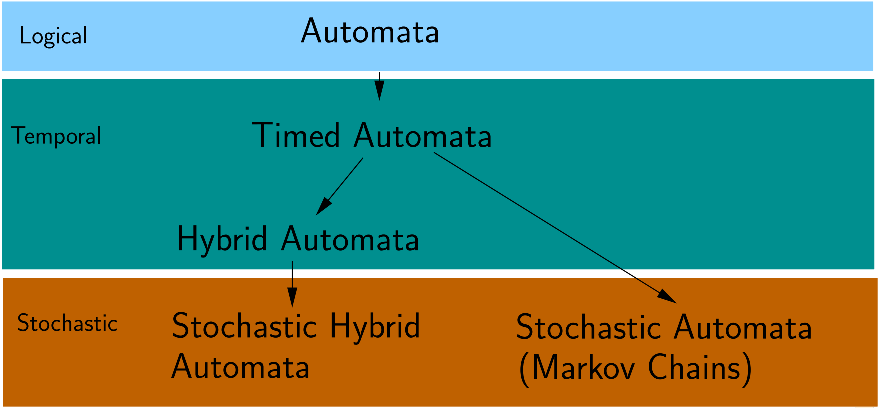
Discrete Event Systems: Logical Properties

Maximal Permissiveness:

- Optimality criterion is *set inclusion*
- Only disable an event if absolutely necessary to guarantee safety and nonblocking



Discrete Event Systems: Levels of Abstraction



Discrete Event Systems: Timed Automata

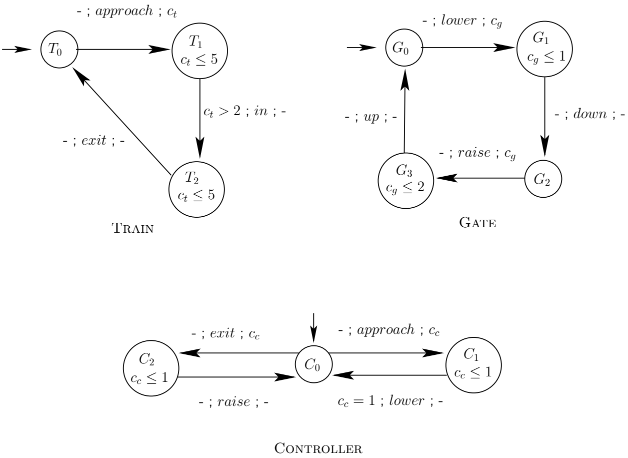


Figure: Three timed automata that jointly model a railroad crossing

Discrete Event Systems: Hybrid Automata

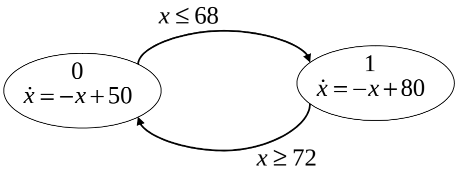


Figure: Thermostat with two discrete states

Discrete Event Systems: This Talk

- Logical (untimed) systems: Languages, Automata
- Reasoning on "simple, unstructured" models can help to elucidate fundamental system- and control-theoretic properties
- Formal approaches are needed in many applications: logic control, networked systems, software systems, transportation systems, etc.

Discrete Event Systems: This Talk

A few things to keep in mind:

- DES theoretical papers: too much notation!
- DES applications: too many states!
- This talk: too many slides!

Discrete Event Systems: This Talk

A few things to keep in mind:

- DES theoretical papers: too much notation!
- DES applications: too many states!
- This talk: too many slides!

Discrete Event Systems: This Talk

A few things to keep in mind:

- DES theoretical papers: too much notation!
- DES applications: too many states!
- This talk: too many slides!

Discrete Event Systems: This Talk

A few things to keep in mind:

- DES theoretical papers: too much notation!
- DES applications: too many states!
- This talk: too many slides!

Discrete Event Systems: This Talk

A few things to keep in mind:

- DES theoretical papers: too much notation!
- DES applications: too many states!
- This talk: too many slides!

Discrete Event Systems: This Talk

A few things to keep in mind:

- DES theoretical papers: too much notation!
- DES applications: too many states!
- This talk: too many slides!

Discrete Event Systems: This Talk

A few things to keep in mind:

- DES theoretical papers: too much notation!
- DES applications: too many states!
- This talk: too many slides!

Control DES

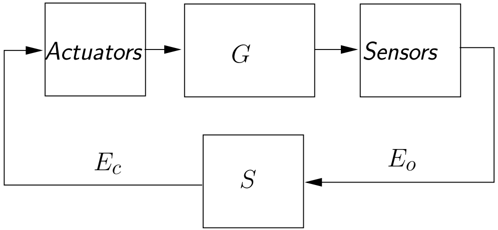
First Part of this Talk

How to ensure safety and nonblocking by feedback control...

Control of DES

- Languages/Automata: Supervisory Control Theory
  - Initiated by Ramadge & Wonham, 1980's
  - Mature body of theory: centralized, decentralized, *modular*
- Control of Petri Nets
  - Many approaches: supervision based on place-invariants, MILP, etc.

The Basic Control Problem: Statement



- Let: $E = E_c \cup E_{uc}$ and $E = E_o \cup E_{uo}$
- Given: System: G, E_c, E_o $\quad + \quad$ Spec: $\mathcal{L}(H) \subseteq \mathcal{L}(G)$
- *Synthesize:* Supervisor S such that S/G is:
  safe *and* **nonblocking** *and* **maximally permissive**

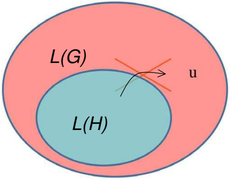
The Basic Control Problem: Solution

- Full Observation: $E_o = E$

$$\mathcal{L}_m(S/G) = [\mathcal{L}(H) \cap \mathcal{L}_m(G)]^{\uparrow C}$$

where $\uparrow C$ = *supremal controllable* operation

- safe, nonblocking, maximally permissive
- $\uparrow C$: quadratic complexity in $H||G$
- **controllability**:
  $\mathcal{L}(H)E_{uc} \cap \mathcal{L}(G) \subseteq \mathcal{L}(H)$

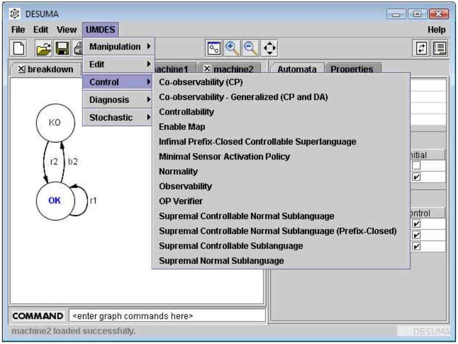


The Basic Control Problem: Solution

- Partial Observation: $E_o \subset E$

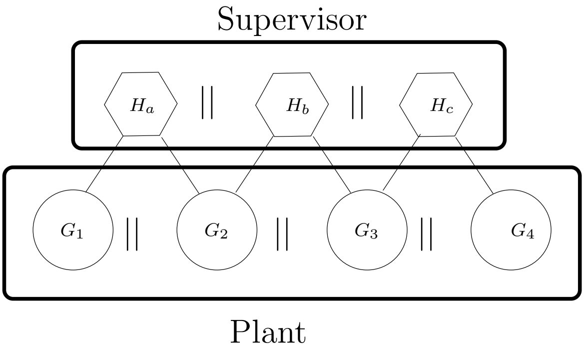
$\longrightarrow$ more difficult – control not discussed in this talk!

The Basic Control Problem: DESUMA Commands



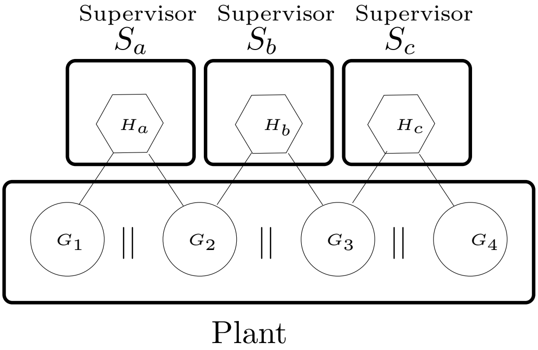
Towards a Modular Approach to Control

- Sets of subplants and specifications
- Monolithic Approach



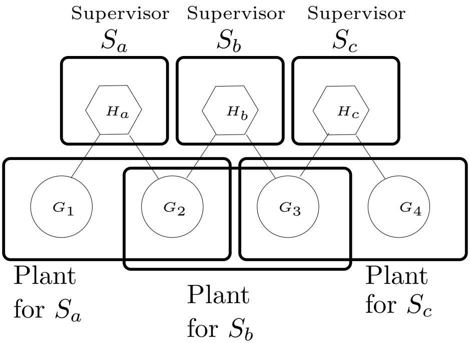
Towards a Modular Approach to Control

- Control with Modular Specifications (Ramadge & Wonham, 1988)



Towards a Modular Approach to Control

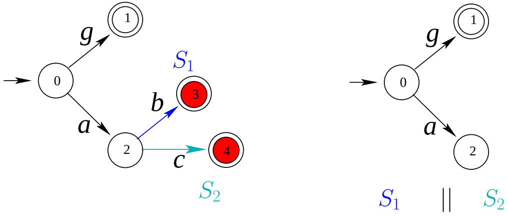
- Local Modular Supervisory Control (Queiroz and Cury, 2000)



- Several related approaches: Heymann et al., Marchand et al., Schmidt et al., van Schuppen et al.

Safety and Nonblocking under Composition

- Safety: composable!
- Nonblocking: not composable!



- The conjunction of nonblocking supervisors may be blocking
  $\Longrightarrow$ S_1 AND S_2 ARE *conflicting*

Conflict Test

- After designing the supervisors $\Rightarrow$ TEST FOR CONFLICT
  Test shows if the composed system is nonblocking, i.e., if the supervisors are nonconflicting (overbar notation means *prefix-closure*):

$$\overline{S_1} \| \overline{S_2} \| \ldots \| \overline{S_m} \stackrel{?}{=} \overline{S_1 \| S_2 \| \ldots \| S_m}$$

$$\Downarrow$$

HIGH COMPLEXITY

Recent Work: P. Pena, J. Cury, S. Lafortune [2006-08]

Objective

Present a new test for conflict based on abstractions of the original supervisors, with reduced complexity.

Instead of calculating

$$\overline{S_1}\|\overline{S_2}\|\ldots\|\overline{S_m} \stackrel{?}{=} \overline{S_1\|S_2\|\ldots\|S_m}$$

we calculate

$$\overline{\theta_1(S_1)}\|\overline{\theta_2(S_2)}\|\ldots\|\overline{\theta_m(S_m)} \stackrel{?}{=} \overline{\theta_1(S_1)\|\theta_2(S_2)\|\ldots\|\theta_m(S_m)}.$$

Recent Work: P. Pena, J. Cury, S. Lafortune [2006-08]

Objective

Present a new test for conflict based on abstractions of the original supervisors, with reduced complexity.

Instead of calculating

$$\overline{S_1} \| \overline{S_2} \| \ldots \| \overline{S_m} \stackrel{?}{=} \overline{S_1 \| S_2 \| \ldots \| S_m}$$

we calculate

$$\overline{\theta_1(S_1)} \| \overline{\theta_2(S_2)} \| \ldots \| \overline{\theta_m(S_m)} \stackrel{?}{=} \overline{\theta_1(S_1) \| \theta_2(S_2) \| \ldots \| \theta_m(S_m)}.$$

Abstractions

Abstractions: "simplify" the model by "erasing" some of the events and building a *projected version* of the original automaton

- Roughly: merge states that are connected by erased events
- Determinize the automaton if necessary
- *OP-abstractions:* have the property that (determinized) result has no more states than the original automaton

Reduced-Complexity Conflict Test

Theorem

If the natural projections $\theta_j(S_j)$ are **OP-abstractions** *and if certain conditions over events not erased by these projections*[a] *are fulfilled, then*

$$\mathop{||}_{j=1}^{m} \overline{\theta_j(S_j)} = \overline{\mathop{||}_{j=1}^{m} \theta_j(S_j)} \iff \mathop{||}_{j=1}^{m} \overline{S_j} = \overline{\mathop{||}_{j=1}^{m} S_j}.$$

[a] Two sets of conditions were developed

$\Downarrow$

THE CONFLICT TEST CAN BE PERFORMED OVER THE ABSTRACTIONS!

Approach of Pena et al.

1. Solve according to the local modular approach (Queiroz & Cury)
2. Pick “good” θ_j, that are *OP-abstractions*, for the local supervisors
    - Specific strategies are proposed in Ph.D. dissertation of P. Pena [2007]
3. Perform the conflict test over the abstractions.

Throughout the process the entire system is never built

Local Modular Synthesis

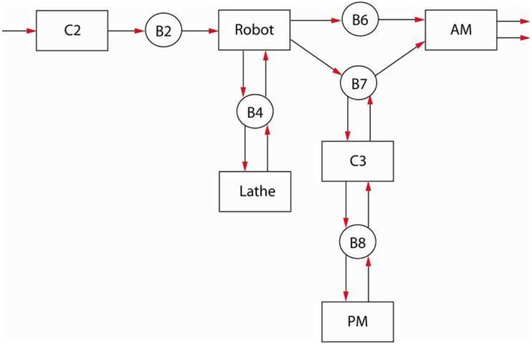


Figure: The FMS Example: 13,428 reachable states; 46,424 transitions

Local Modular Synthesis

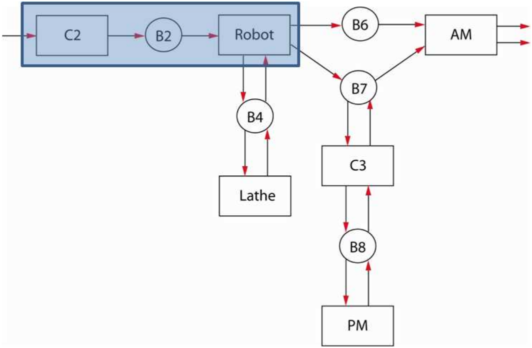


Figure: Synthesize supervisor for $B2$ using $C2$ and $Robot$

Local Modular Synthesis

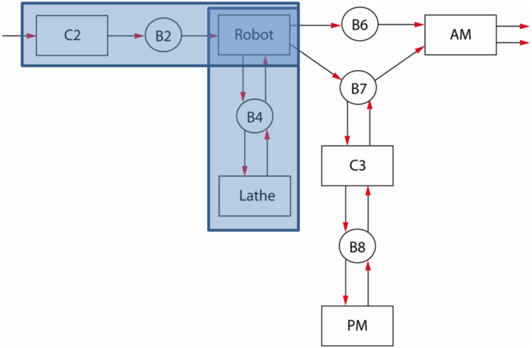


Figure: Synthesize supervisor for $B4$ using $Robot$ and $Lathe$

Local Modular Synthesis

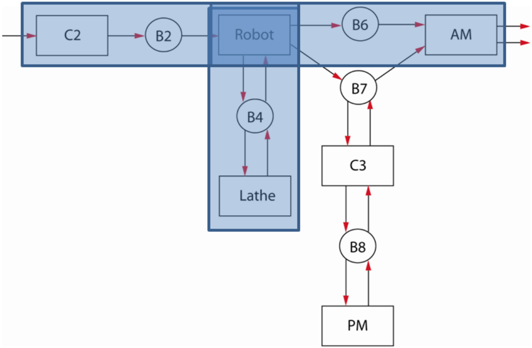


Figure: Synthesize supervisor for $B6$ using $Robot$ and AM

Local Modular Synthesis

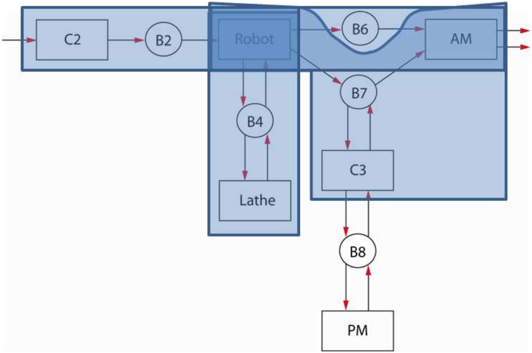


Figure: Synthesize supervisor for $B7$ using $Robot$, AM, and $C3$

Local Modular Synthesis

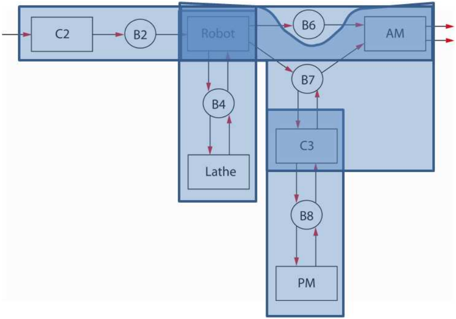


Figure: Synthesize supervisor for $B8$ using $C3$ and PM

- Overall: *Safe* but *blocking*... What do we do?

Recent Research: R. Hill, D. Tilbury, S. Lafortune [2006-08]

- Refine the local modular approach in order to resolve conflict and obtain a safe and nonblocking system
- Three approaches proposed in Ph.D. dissertation of R. Hill [2008]
  - One of the approaches developed in collaboration with J. Cury and M. de Queiroz
- No "free lunch": may not be maximally permissive

Modular Synthesis Using Conflict Resolution and Abstraction

Exploit a notion of *equivalence* for states defined by R. Malik, H. Flordal et al.

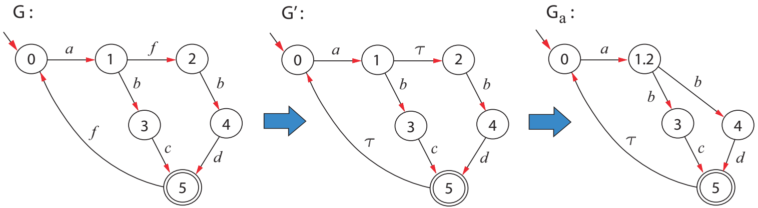


Figure: Abstraction based on *conflict equivalence*; event f is not "relevant"

Modular Synthesis Using Conflict Resolution and Abstraction

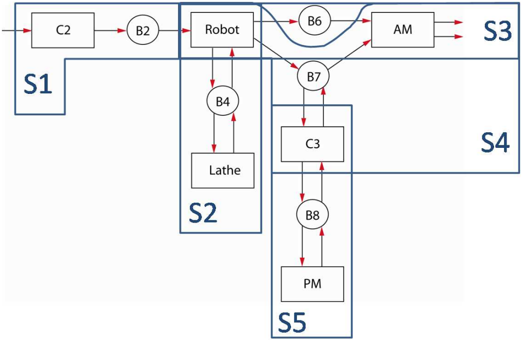


Figure: Synthesize $S1$ $(B2)$, $S2$ $(B4)$, $S3$ $(B6)$, $S4$ $(B7)$, $S5$ $(B8)$

Modular Synthesis Using Conflict Resolution and Abstraction

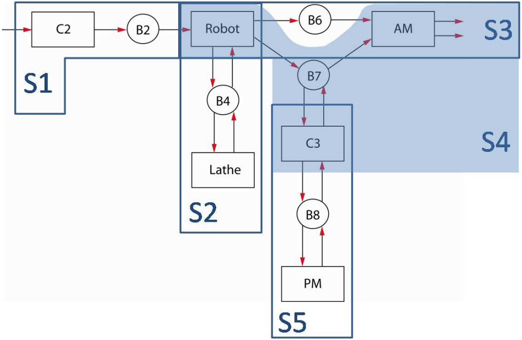


Figure: Abstract Controlled System 4: $S4$ for $Robot||C3||AM$

Modular Synthesis Using Conflict Resolution and Abstraction

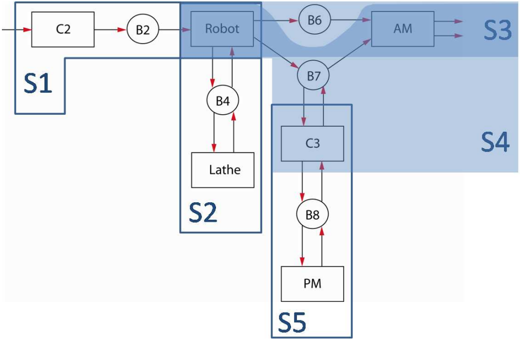


Figure: Abstract Controlled System 3: $S3$ for $Robot||AM$

Modular Synthesis Using Conflict Resolution and Abstraction

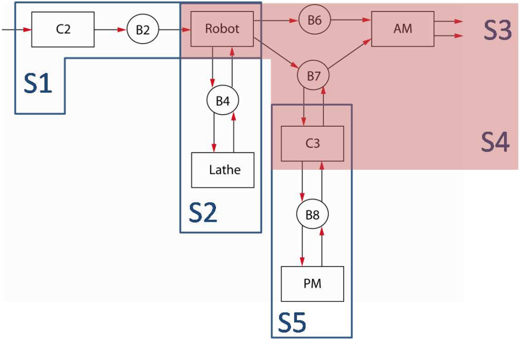


Figure: Nonconflict Test of Abstracted Controlled Systems 3 and 4: OK

Modular Synthesis Using Conflict Resolution and Abstraction

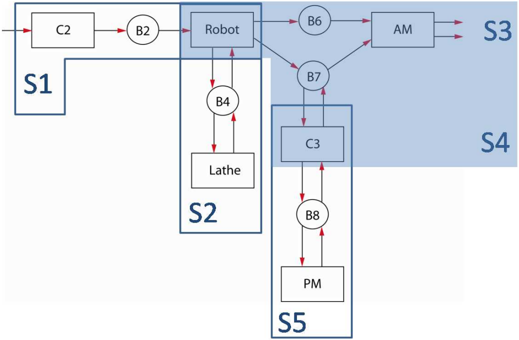


Figure: Abstract Composed 3&4 of previous step

Modular Synthesis Using Conflict Resolution and Abstraction

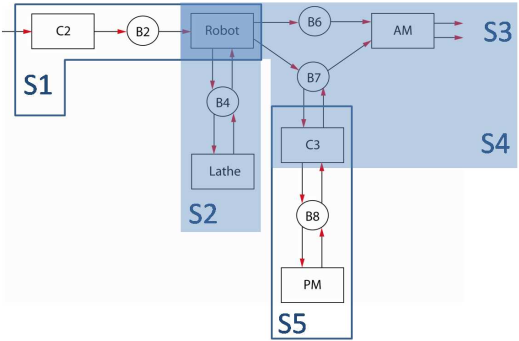


Figure: Abstract Controlled System 2: $S2$ for $Lathe||Robot$

Modular Synthesis Using Conflict Resolution and Abstraction

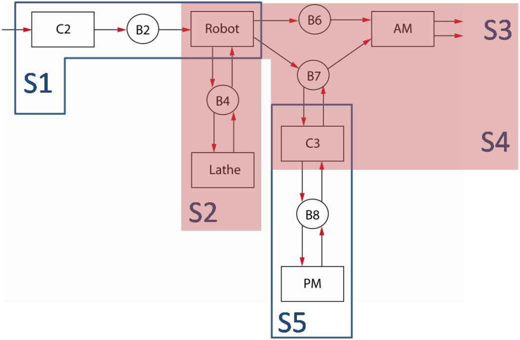


Figure: Nonconflict Test of Abstracted Controlled Systems 2 and 3&4: OK

Modular Synthesis Using Conflict Resolution and Abstraction

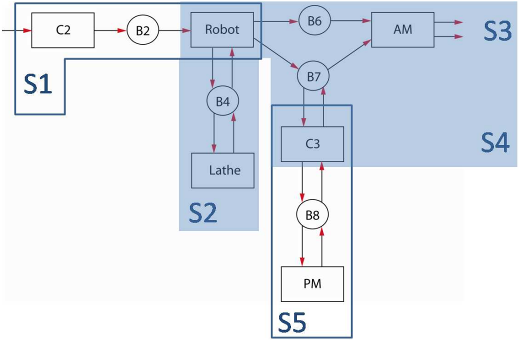


Figure: Abstract Composed 2&3&4 of previous step

Modular Synthesis Using Conflict Resolution and Abstraction

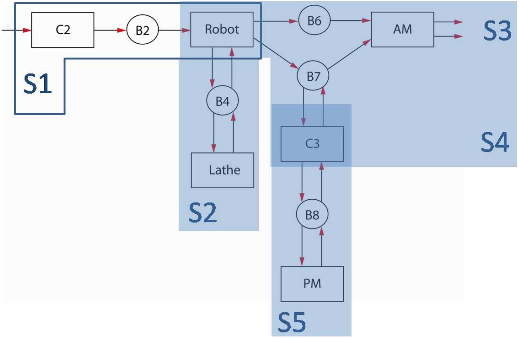


Figure: Abstract Controlled System 5: $S5$ for $PM||C3$

Modular Synthesis Using Conflict Resolution and Abstraction

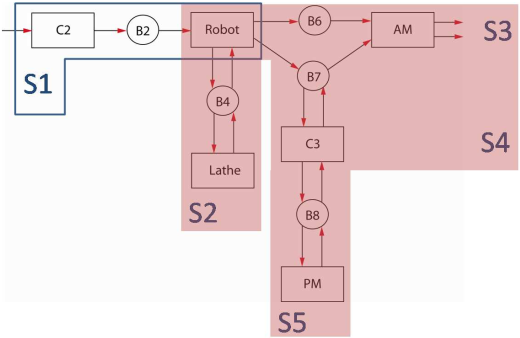


Figure: Nonconflict Test of Abstracted Controlled Systems 5 and 2&3&4: *Conflict!*

Modular Synthesis Using Conflict Resolution and Abstraction

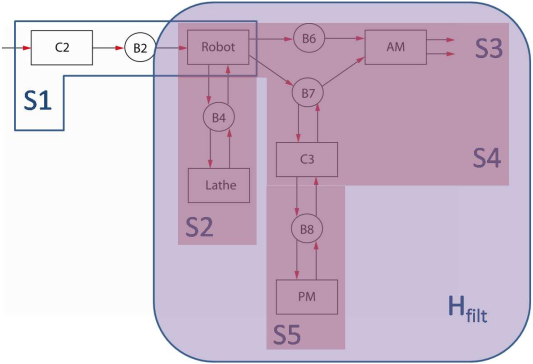


Figure: Synthesize *filter* H_{filt} to make 5 with 2&3&4 nonblocking

Modular Synthesis Using Conflict Resolution and Abstraction

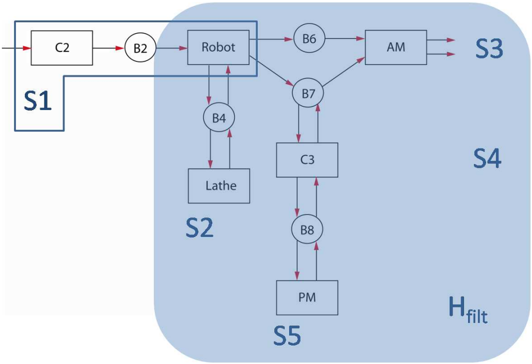


Figure: Abstract Composed 2&3&4&5&H_{filt} of previous step

Modular Synthesis Using Conflict Resolution and Abstraction

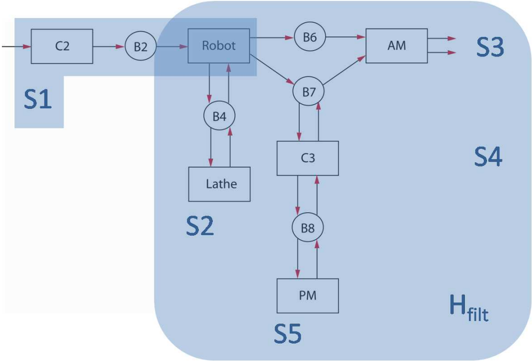


Figure: Abstract Controlled System 1: $S1$ for $C2||Robot$

Modular Synthesis Using Conflict Resolution and Abstraction

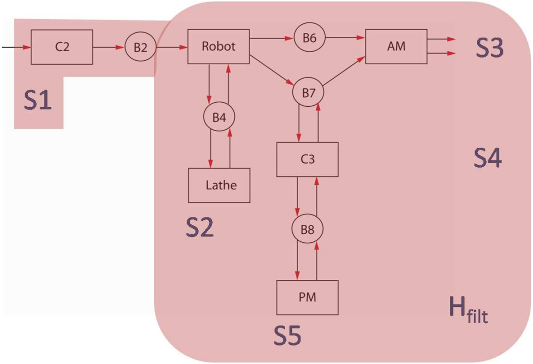


Figure: Test for Nonconflict of Abstracted Controlled Systems 1 and 2&3&4&5&H_{filt}: OK

Modular Synthesis Using Conflict Resolution and Abstraction

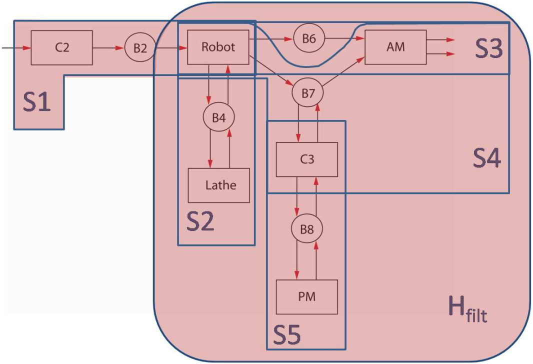


Figure: Overall, 6 modular controllers: $S1$, $S2$, $S3$, $S4$, $S5$, and H_{filt}

Modular Synthesis Using Conflict Resolution and Abstraction

Computational Gains:

Case	Largest Supervisor #states(#trans)	Largest intermediate automaton #states(#trans)	Number of pieces active
monolithic	2256 (7216)	13,248	6
EBCR approach	80 (259)	128 (429)	5

Modular Synthesis Using Conflict Resolution and Abstraction

What do we gain/lose?

- Safety guaranteed
- Nonblocking guaranteed
- Not maximally permissive in general
- Computations reduced

Research Trends

- Modular control: use of abstraction, hierarchical methods, structured models with *interfaces*
- Decentralized control architectures for partially-observed systems
- Distributed control with communication (networked systems)
- Fault tolerant control: need for *fault diagnosis*!

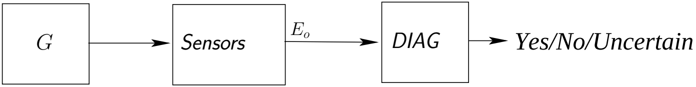
Diagnosis of Partially Observed DES

Second Part of this Talk

How to detect unobservable events...

Diagnosis of Partially Observed DES

- Model-based inferencing about past occurrence of *significant* (aka *fault*) events

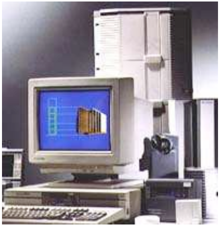


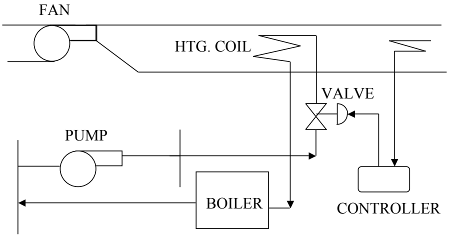
- Initiated by F. Lin (WSU, 1994) and M. Sampath, R. Sengupta, K. Sinnamohideen, S. Lafortune and D. Teneketzis (1995)
- Numerous extensions: timed, intermittent faults, decentralized and distributed architectures, etc.

Heating, Ventilation, and Air Conditioning Systems

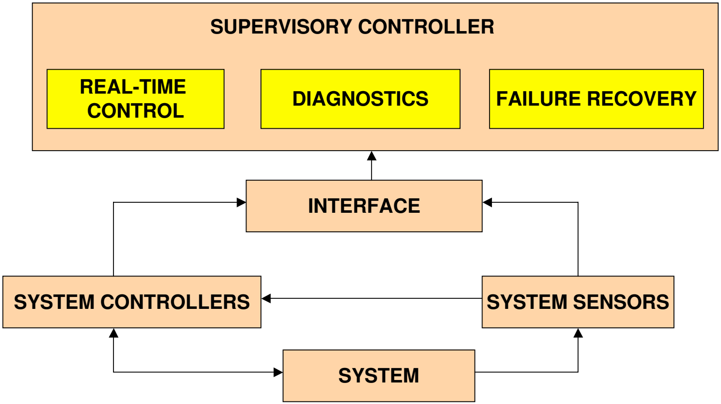
K. Sinnamohideen, M. Sampath (Johnson Controls, Inc.)

- Components hard to access, few sensors
- Valve, pump, controller faults, etc.
- Objective: *Automate* detection and isolation of faults

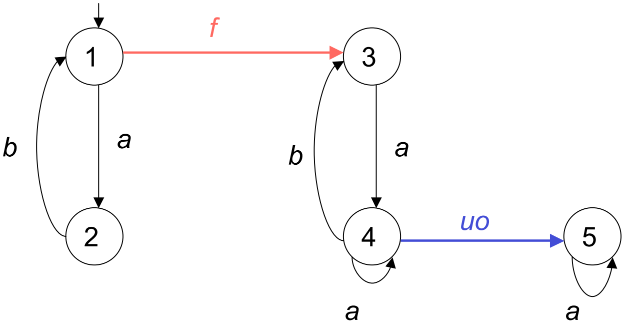




Conceptual System Architecture



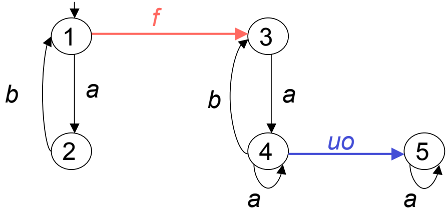
The Essence of the Problem

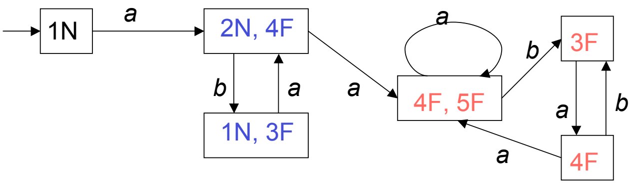


abab … ab : ???

abab … abaa : fault!

The Essence of the Problem - Diagnosers

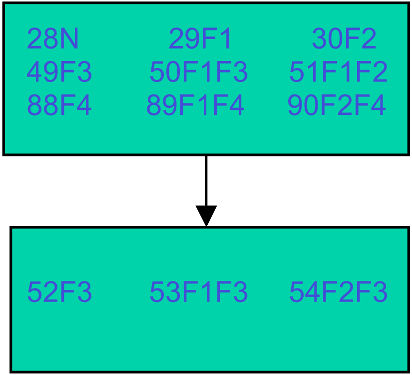




Information in Diagnoser States

From (simplified) HVAC example - ≈ 150 states

Uncertain for:
F1, F2, F3, F4



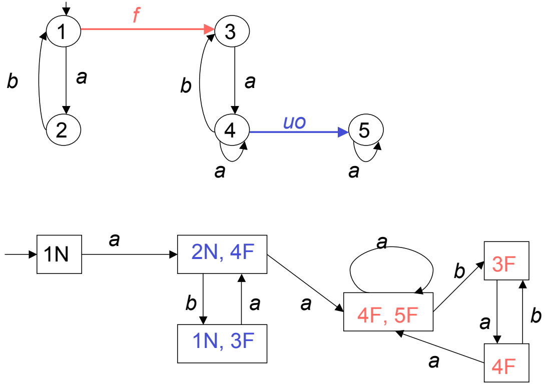
Uncertain for: F1, F2
Certain for F3

Steps in the Diagnoser Approach

- Model complete system with sensors, including faulty behavior
- Observable vs. unobservable events
- **Analysis:** Can the faults always be diagnosed?
  - Notion of *diagnosability*
  - Tests using diagnoser / verifier automata
- **Online Diagnosis:** How to detect faults online?
  - Diagnoser Automata / Petri Nets

Diagnosability Analysis

What Should We Worry About? *Indeterminate Cycles* in Diagnoser:



Diagnosability Analysis

Diagnosability

An unobservable (fault) event f is diagnosable in language $\mathcal{L}(G)$ if every occurrence of f can be detected with certainty in a bounded number of events after it occurs.

Theorem

A system modeled by automaton G is diagnosable iff its Diagnoser G_d does not contain indeterminate cycles.

Diagnosability Analysis

Diagnosability

An unobservable (fault) event f is diagnosable in language $\mathcal{L}(G)$ if every occurrence of f can be detected with certainty in a bounded number of events after it occurs.

Theorem

A system modeled by automaton G is diagnosable iff its Diagnoser G_d does not contain indeterminate cycles.

Diagnosability Analysis in DESUMA

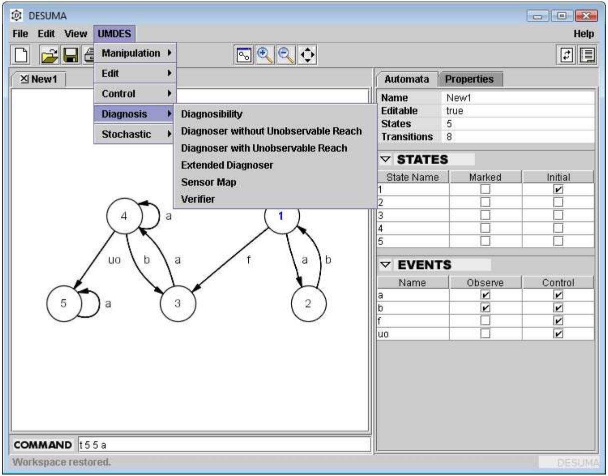


Figure: Diagnosis commands in DESUMA

Diagnosability Analysis in DESUMA

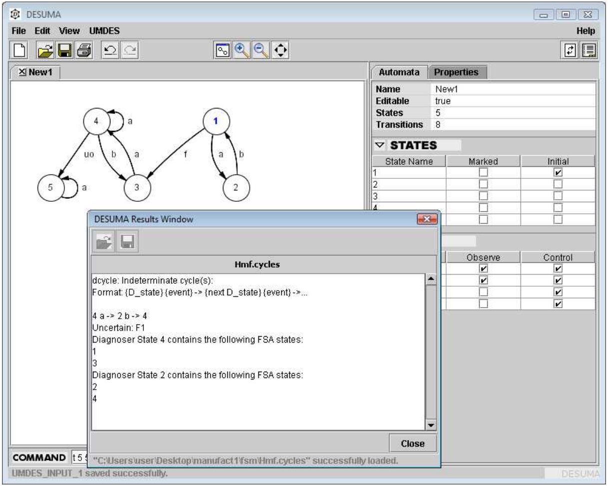


Figure: Indeterminate cycle analysis for diagnosability

Diagnosability Analysis in DESUMA

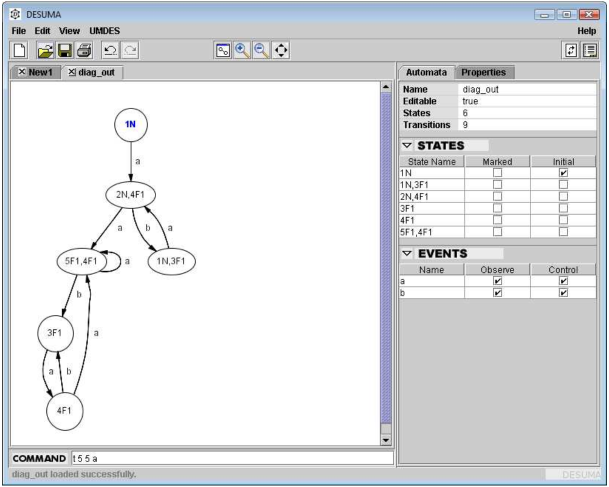


Figure: Diagnoser automaton

Document Processing Systems

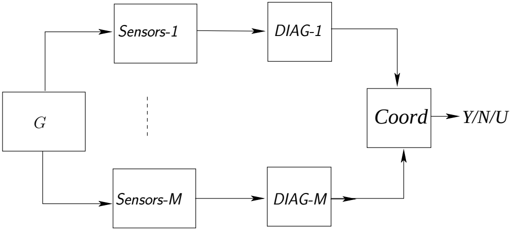
Meera Sampath et al. (Xerox Corp.)

- Complex processes, few sensors
- Electro-mechanical faults (paper path)
- Image quality faults (virtual sensor approach)

Automated Highway Systems

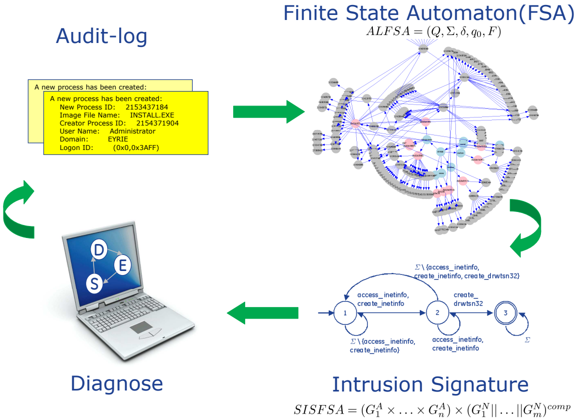
Raja Sengupta et al. (U. California at Berkeley)

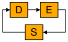
- Platoons of vehicles
- In-vehicle faults
- Transmitter and receiver faults
- Decentralized diagnosis with coordinator



Intrusion Detection in Computer Systems

Diagnosis of *Patterns*: Sahika Genc (GE)
(Annual Symposium on Information Assurance, Albany, NY, 2008)
Related work: H. Marchand et al. (IRISA, France)





Recent Research: J.C. Basilio [2007-08]

Robustness properties of architecture of R. Debouk et al. (2000):

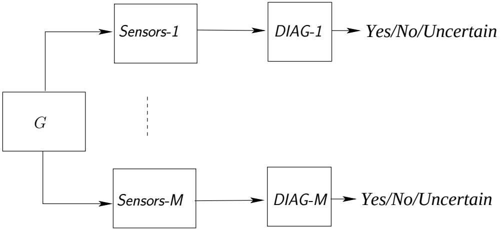


Figure: No coordinator: At least one site should detect each fault

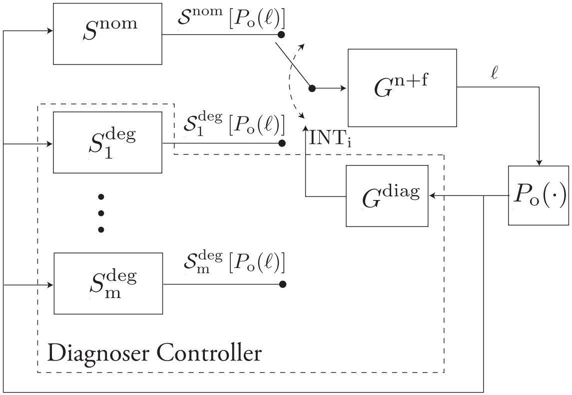
One of more sites may fail →Robust Decentralized Diagnosability

- Definition, test, online robust diagnosis

Research Trends

- Various decentralized / distributed architectures
- Methodologies based on Petri net models
- Inverse problem: security (opacity)
- Merge diagnosis and control: Fault tolerant control
- Sensor networks: *use sensors efficiently*!

Fault Tolerant Control (A. Paoli, Bologna)



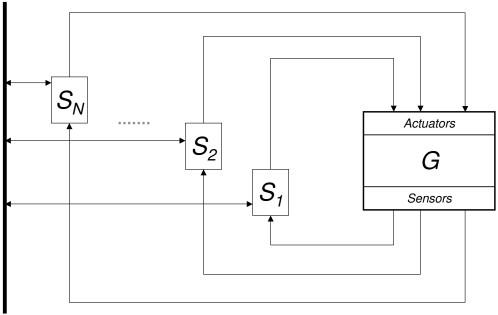
Active Sensing of Partially Observed DES

Third Part of this Talk

How to use sensors efficiently...

Decentralized Control or Diagnosis With Communication

Communication is **costly**: energy, bandwidth, security,...



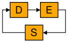
Who should communicate with whom and when?

Decentralized Control or Diagnosis With Communication

- Estimation, control, and communication are interdependent!
  - what you estimate depends on what others tell you and on your/their control actions
  - what you do for control affects what you/others observe and thus what you estimate
  - what you communicate affects the observations of others and thus their communications to you
  - what others communicate to you affects your estimation (and thus your control and your communications)
  - and so on and so forth
- Lack of *separation* in general ⇒ Computationally challenging

Decentralized Control or Diagnosis With Communication

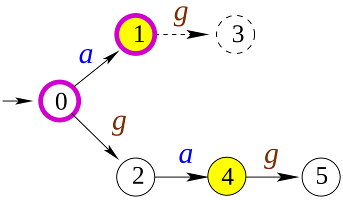
- Our Approach:
  - Fix control (diagnostic) and only solve the communication problem
    - Problem is still hard: all communication policies are interdependent
  - Solve only for *communication with sensors*
    - Called the *Active Sensing* Problem
  - Present solution for a *single* agent only (!)
    - [Wang et al. CDC'08]
    - Related work: [Thorsley-Teneketzis, 2007], [Cassez-Tripakis, 2008]



Active Sensing of Partially Observed DES

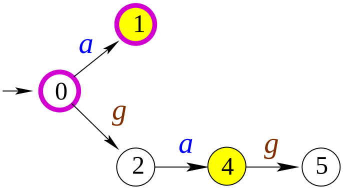
Formulation:

- Automaton: G
- Potentially Observable Event Set of G: $E_o = \{a, g\}$
- Set of state pairs of G that must be distinguished: *safety* specification $(0,1),(1,4)$
- When to activate a and g sensors?
  - Activate only if necessary, but enough to be safe
- Decide on the basis of the transitions in G



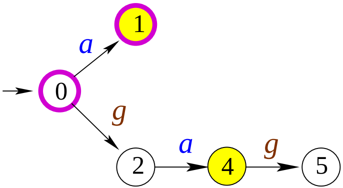
Active Sensing of Partially Observed DES

- Monotonicity:
  - Do not observe g at 0: (1,4) confused
  - Do not observe g at 0 **and** a at 2: (1,4) **not** confused!
  - But cannot do the above if you activate your own sensors!



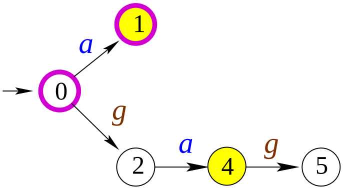
Active Sensing of Partially Observed DES

- Monotonicity:
  - Do not observe g at 0: (1,4) confused
  - Do not observe g at 0 **and** a at 2: (1,4) **not** confused!
  - But cannot do the above if you activate your own sensors!



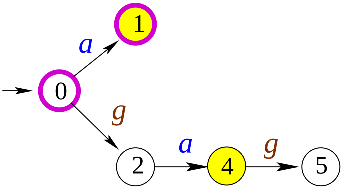
Active Sensing of Partially Observed DES

- Monotonicity:
  - Do not observe g at 0: (1,4) confused
  - Do not observe g at 0 **and** a at 2: (1,4) **not** confused!
  - But cannot do the above if you activate your own sensors!



Active Sensing of Partially Observed DES

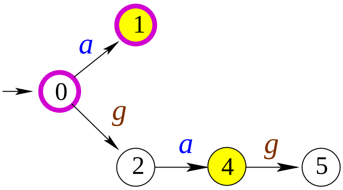
- Monotonicity:
  - Do not observe g at 0: (1,4) confused
  - Do not observe g at 0 **and** a at 2: (1,4) **not** confused!
  - But cannot do the above if you activate your own sensors!



Active Sensing: Feasibility

Sensor activation policy (SAP):
$\Omega \subseteq Transitions(G)$

- If two strings "look the same," then must have same activation decision on a *common possible event*
- Not activating g at 0 **and** a at 2 is not *feasible*: if g is not activated at 0, then 0 and 2 must have the same activation decision for a
- This is called the *feasibility* requirement of SAP



Active Sensing: Problem Statement

Given G, E_o, and a set of state pairs that must be distinguished, find $\Omega^* \subseteq Transitions(G)$ such that

- Ω^* satisfies the safety specification
- Ω^* satisfied the feasibility requirement
- Ω^* is a minimal set

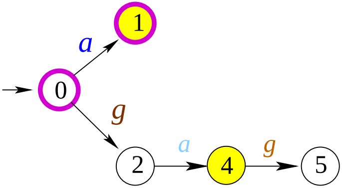


Figure: A Minimal Solution

Active Sensing: Main Theorems

Theorem

[Monotonicity]
Let Ω_1 and Ω_2 be two feasible SAP, such that $\Omega_1 \subset \Omega_2$. Then

$$\Omega_1 \text{ safe} \quad \Rightarrow \quad \Omega_2 \text{ safe}$$

Theorem

[Existence of Maximum Element]
Let Ω be an SAP. Then there exists a **maximum feasible subpolicy** $\Omega^{\uparrow F}$ *that contains all $\Omega_F \subseteq \Omega$ that are feasible.*
The complexity of performing $\uparrow F$ is **polynomial** *in the state space of G.*

Active Sensing: Main Theorems

Theorem

[Monotonicity]
Let Ω_1 and Ω_2 be two feasible SAP, such that $\Omega_1 \subset \Omega_2$. Then

$$\Omega_1 \text{ safe} \quad \Rightarrow \quad \Omega_2 \text{ safe}$$

Theorem

[Existence of Maximum Element]
Let Ω be an SAP. Then there exists a **maximum feasible subpolicy** $\Omega^{\uparrow F}$ *that contains all $\Omega_F \subseteq \Omega$ that are feasible.*
The complexity of performing $\uparrow F$ is **polynomial** *in the state space of G.*

Active Sensing: A Polynomial-Complexity Algorithm

- Let Ω be safe and feasible
- Let $\Omega_{test} = \Omega \setminus \{(x, e)\}$
  - If $\Omega_{test}^{\uparrow F}$ is not safe, then no subset of Ω_{test} that does not activate e at x will be safe
    $\Rightarrow$ Keep e activated at x and try to deactivate some other event at some other state
  - If $\Omega_{test}^{\uparrow F}$ is safe, then e need not be activated at x
    $\Rightarrow$ Reinitialize Ω to $\Omega_{test}^{\uparrow F}$
- Proceed until each (observable) event e at each state x has been considered for de-activation
  - Only one such consideration per transition (x, e) in G
  - A minimal (safe and feasible) solution Ω^* is found
- Ω^* depends on the order in which transitions are considered...

Active Sensing: A Polynomial-Complexity Algorithm

- Let Ω be safe and feasible
- Let $\Omega_{test} = \Omega \setminus \{(x,e)\}$
  - If $\Omega_{test}^{\uparrow F}$ is not safe, then no subset of Ω_{test} that does not activate e at x will be safe
    $\Rightarrow$ Keep e activated at x and try to deactivate some other event at some other state
  - If $\Omega_{test}^{\uparrow F}$ is safe, then e need not be activated at x
    $\Rightarrow$ Reinitialize Ω to $\Omega_{test}^{\uparrow F}$
- Proceed until each (observable) event e at each state x has been considered for de-activation
  - Only one such consideration per transition (x,e) in G
  - A minimal (safe and feasible) solution Ω^* is found
- Ω^* depends on the order in which transitions are considered...

Active Sensing: A Polynomial-Complexity Algorithm

- Let Ω be safe and feasible
- Let $\Omega_{test} = \Omega \setminus \{(x, e)\}$
  - If $\Omega_{test}^{\uparrow F}$ is not safe, then no subset of Ω_{test} that does not activate e at x will be safe
    $\Rightarrow$ Keep e activated at x and try to deactivate some other event at some other state
  - If $\Omega_{test}^{\uparrow F}$ is safe, then e need not be activated at x
    $\Rightarrow$ Reinitialize Ω to $\Omega_{test}^{\uparrow F}$
- Proceed until each (observable) event e at each state x has been considered for de-activation
  - Only one such consideration per transition (x, e) in G
  - A minimal (safe and feasible) solution Ω^* is found
- Ω^* depends on the order in which transitions are considered...

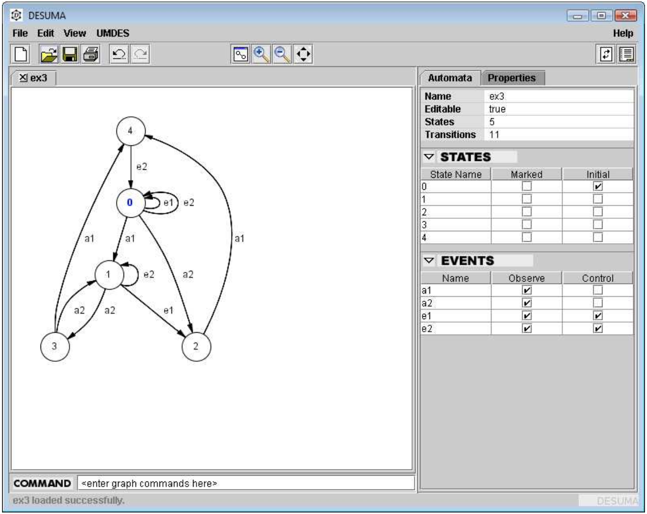
Active Sensing: A Polynomial-Complexity Algorithm

- Let Ω be safe and feasible
- Let $\Omega_{test} = \Omega \setminus \{(x, e)\}$
  - If $\Omega_{test}^{\uparrow F}$ is not safe, then no subset of Ω_{test} that does not activate e at x will be safe
    $\Rightarrow$ Keep e activated at x and try to deactivate some other event at some other state
  - If $\Omega_{test}^{\uparrow F}$ is safe, then e need not be activated at x
    $\Rightarrow$ Reinitialize Ω to $\Omega_{test}^{\uparrow F}$
- Proceed until each (observable) event e at each state x has been considered for de-activation
  - Only one such consideration per transition (x, e) in G
  - A minimal (safe and feasible) solution Ω^* is found
- Ω^* depends on the order in which transitions are considered...

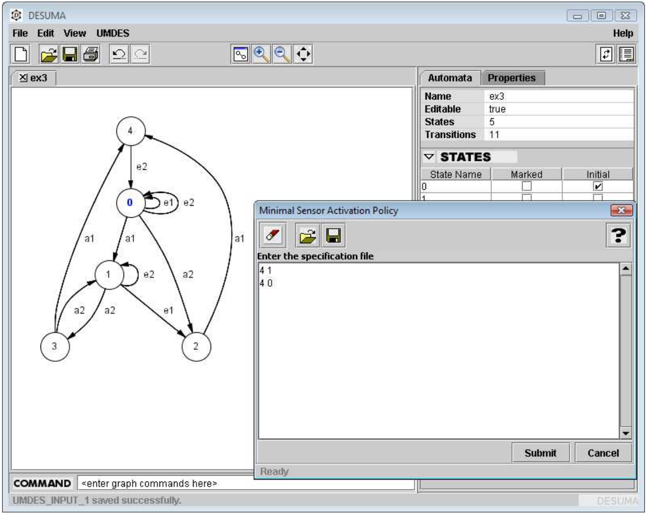
Active Sensing: A Polynomial-Complexity Algorithm

- Let Ω be safe and feasible
- Let $\Omega_{test} = \Omega \setminus \{(x, e)\}$
  - If $\Omega_{test}^{\uparrow F}$ is not safe, then no subset of Ω_{test} that does not activate e at x will be safe
    $\Rightarrow$ Keep e activated at x and try to deactivate some other event at some other state
  - If $\Omega_{test}^{\uparrow F}$ is safe, then e need not be activated at x
    $\Rightarrow$ Reinitialize Ω to $\Omega_{test}^{\uparrow F}$
- Proceed until each (observable) event e at each state x has been considered for de-activation
  - Only one such consideration per transition (x, e) in G
  - A minimal (safe and feasible) solution Ω^* is found
- Ω^* depends on the order in which transitions are considered...

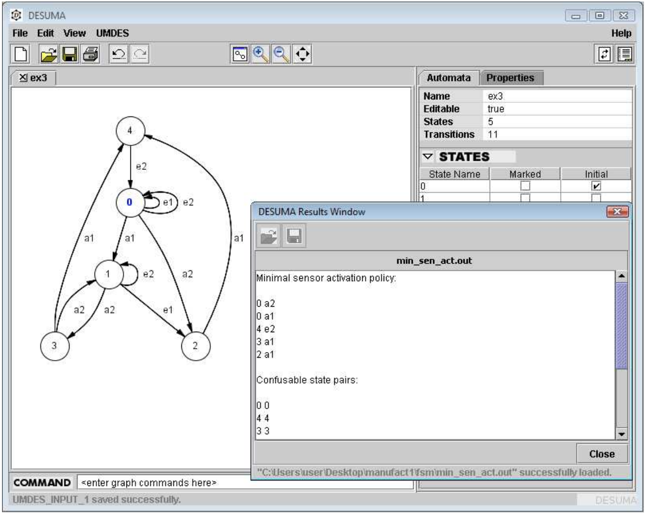
Active Sensing: Example



Active Sensing: Example



Active Sensing: Example



Research Trends

- Decentralized systems
- Quantitative approaches
- From active sensing to multi-agent communication...

Conclusion

What should you remember?

- Modeling formalisms:
  - Languages
  - Automata
  - Petri nets
- Concepts:
  - Safety
  - Nonblocking
  - Maximal Permissiveness
- Operations:
  - Parallel composition
  - Abstractions (Projections)

Conclusion

What should you remember?

- Modeling formalisms:
  - Languages
  - Automata
  - Petri nets
- Concepts:
  - Safety
  - Nonblocking
  - Maximal Permissiveness
- Operations:
  - Parallel composition
  - Abstractions (Projections)

Conclusion

What should you remember?

- Modeling formalisms:
  - Languages
  - Automata
  - Petri nets
- Concepts:
  - Safety
  - Nonblocking
  - Maximal Permissiveness
- Operations:
  - Parallel composition
  - Abstractions (Projections)

Conclusion

What should you remember?

- Modeling formalisms:
  - Languages
  - Automata
  - Petri nets
- Concepts:
  - Safety
  - Nonblocking
  - Maximal Permissiveness
- Operations:
  - Parallel composition
  - Abstractions (Projections)

Conclusion

What should you remember?

- Modeling formalisms:
  - Languages
  - Automata
  - Petri nets
- Concepts:
  - Safety
  - Nonblocking
  - Maximal Permissiveness
- Operations:
  - Parallel composition
  - Abstractions (Projections)

Conclusion

What should you remember?

- Modeling formalisms:
    - Languages
    - Automata
    - Petri nets
- Concepts:
    - Safety
    - Nonblocking
    - Maximal Permissiveness
- Operations:
    - Parallel composition
    - Abstractions (Projections)

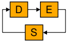
Conclusion

What should you remember?

- Modeling formalisms:
  - Languages
  - Automata
  - Petri nets
- Concepts:
  - Safety
  - Nonblocking
  - Maximal Permissiveness
- Operations:
  - Parallel composition
  - Abstractions (Projections)

Conclusion: Concepts to Remember

- Properties:
  - Controllability (observability)
  - Nonconflicting
  - Diagnosability
  - Feasibility
- Algorithmic Techniques:
  - $\uparrow C$
  - cycle analysis in Diagnosers
  - $\uparrow F$



Conclusion: Concepts to Remember

- Properties:
  - Controllability (observability)
  - Nonconflicting
  - Diagnosability
  - Feasibility
- Algorithmic Techniques:
  - $\uparrow C$
  - cycle analysis in Diagnosers
  - $\uparrow F$

Conclusion: Concepts to Remember

- Properties:
  - Controllability (observability)
  - Nonconflicting
  - Diagnosability
  - Feasibility
- Algorithmic Techniques:
  - $\uparrow C$
  - cycle analysis in Diagnosers
  - $\uparrow F$

Conclusion: Concepts to Remember

- Properties:
  - Controllability (observability)
  - Nonconflicting
  - Diagnosability
  - Feasibility
- Algorithmic Techniques:
  - $\uparrow C$
  - cycle analysis in Diagnosers
  - $\uparrow F$

Conclusion: Concepts to Remember

- Properties:
  - Controllability (observability)
  - Nonconflicting
  - Diagnosability
  - Feasibility
- Algorithmic Techniques:
  - $\uparrow C$
  - cycle analysis in Diagnosers
  - $\uparrow F$

Conclusion: What Lies Ahead

- Modular *reconfigurable* control
- Diagnosis + Control: *Fault-tolerant control*
- Computer security: *Opacity*, *Nontransitive interference*
- Communication in distributed control architectures

- Applications, Applications, Applications...
  (Modeling...)

Conclusion: What Lies Ahead

- Modular *reconfigurable* control
- Diagnosis + Control: *Fault-tolerant control*
- Computer security: *Opacity*, *Nontransitive interference*
- Communication in distributed control architectures

- Applications, Applications, Applications...
  (Modeling...)

Conclusion: Education

- Educating Control Engineers in the 21st Century

OBRIGADO!

Conclusion: Education

- Educating Control Engineers in the 21st Century

OBRIGADO!