Partially observed discrete-event systems: from state estimation to intrusion detection

Carla Seatzu

Department of Electrical and Electronic Engineering,
University of Cagliari, Italy

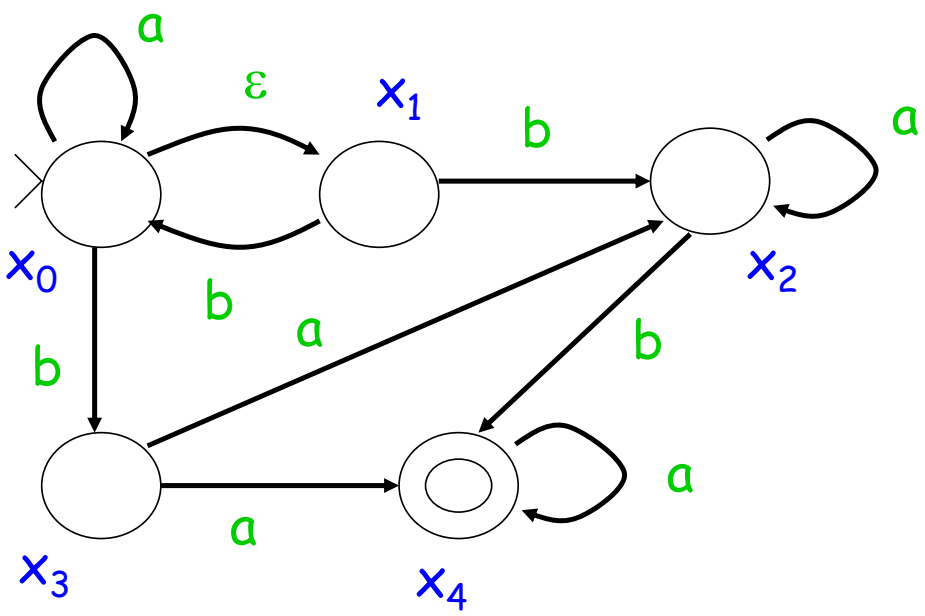
Sorrento, June 1th, 2018

Outline

- **Introduction**
- State estimation
- Fault diagnosis
- Opacity verification
- Intrusion detection
- Conclusions and open issues

Different forms of nondeterminism

- Partially observed discrete event systems are a general formalism dating back to the definition of nondeterministic automata.



- Some events are silent, other events are indistinguishable.
- There could be partial or no information on the initial state.

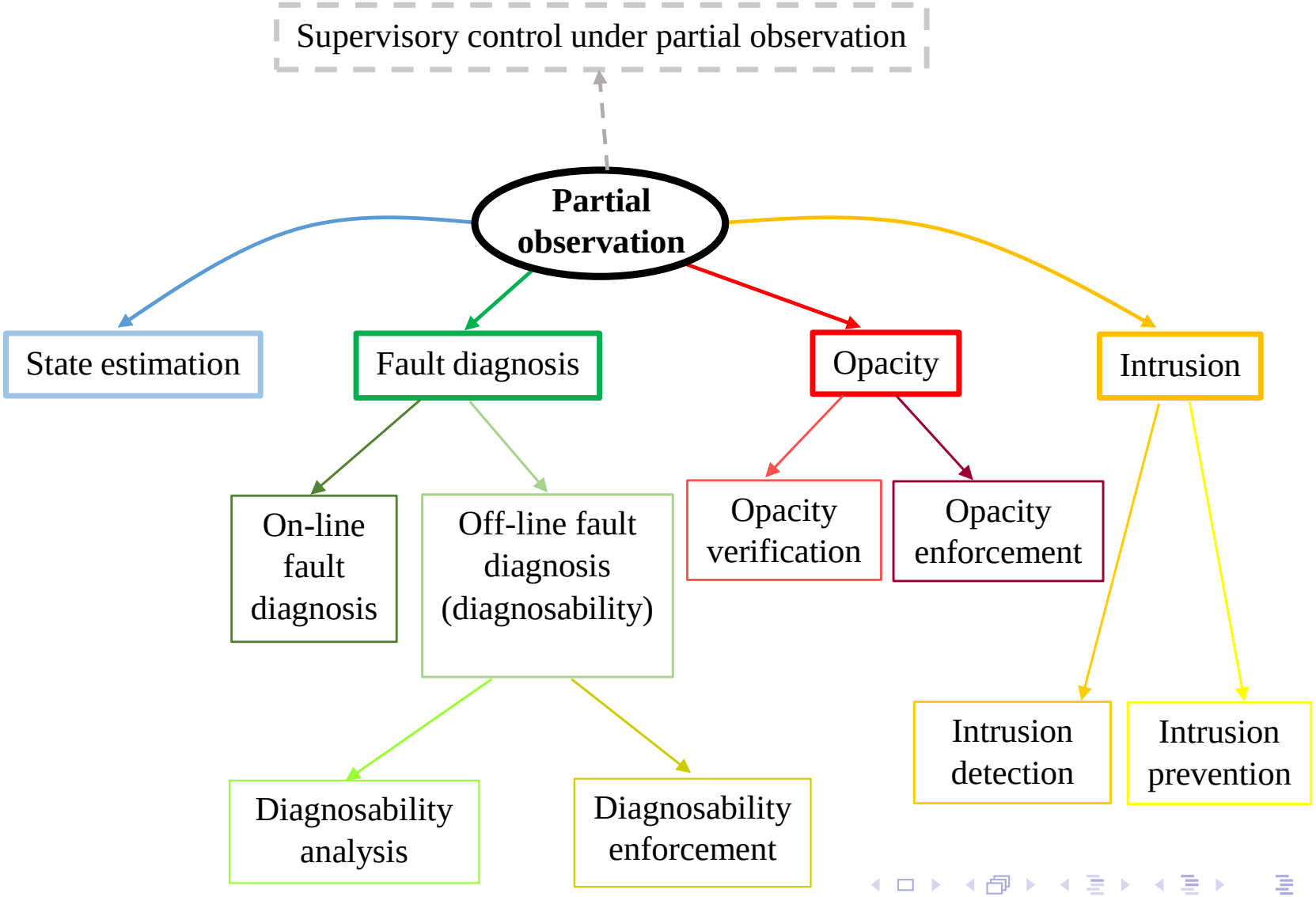
Different forms of nondeterminism

- The initial state is unknown $\Rightarrow$ This case is natural during error recovery. Consider a plant remotely controlled: if the communication fails the state will be at best partially known. $\Rightarrow$ In manufacturing, resources enter unobserved, or we know how many resources have entered the system but not their exact location.

- Silent events $\Rightarrow$ events that cannot be observed at all, e.g., faults.

- Indistinguishable events $\Rightarrow$ events that produce the same output symbol. $\Rightarrow$ In a manufacturing environment, the occurrence of a certain event may correspond to the entrance of a part in the plant, but parts of different kinds produce the same output signal.

Main problems originating from partial observation



Focus of the talk

- A very rich literature exists on supervisory control under partial observation, where different control specifiations and requirements, different sources of nondeterminism, and different formalisms have been considered (Cielsak et al., 1988; Lin and Wonham, 1988; Wonham, 1989; Giua et al., 1992; Li and Wonham, 1993; Takai et al., 1995; Moody et al., 1996; Moody and Antsaklis, 2000; Giua et al., 2004; Cabasino et al., 2017; ...).

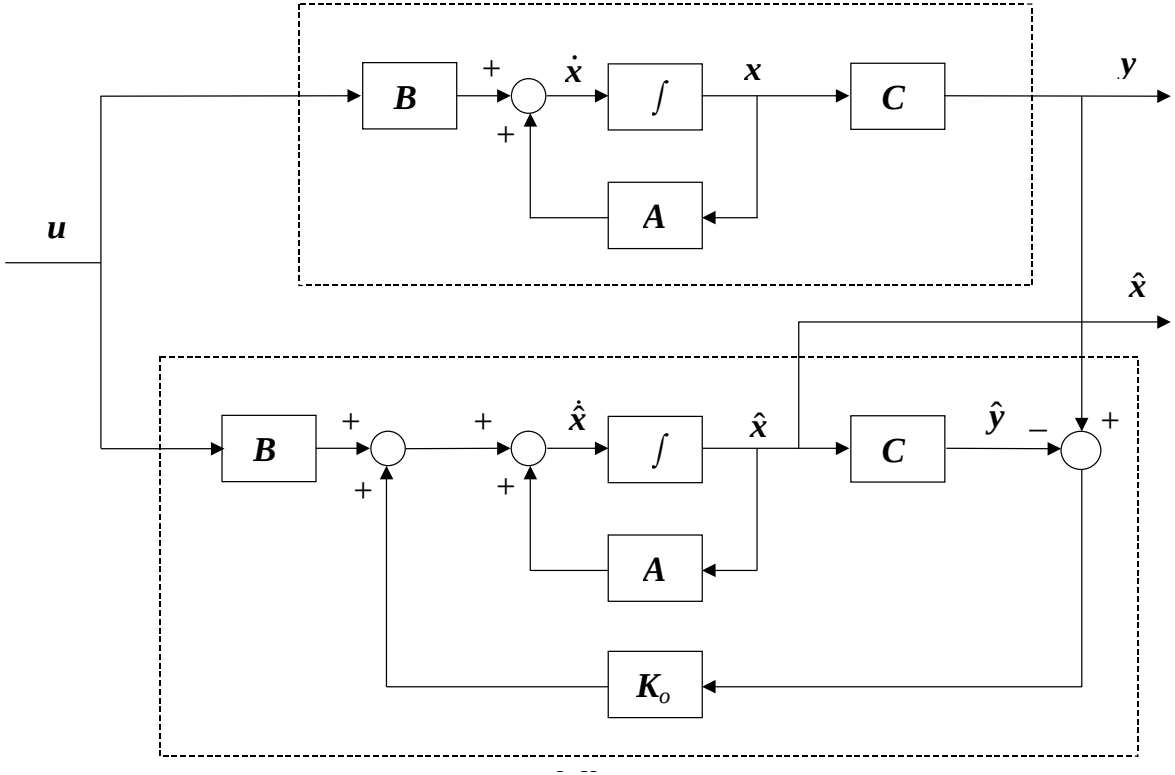
- In this talk the focus is on state estimation and on the other problems mentioned above. In all such cases a solution is typically provided taking advantage of tools inspired by the notion of observer (Ramadge, 1986).

Outline

- Introduction
- **State estimation**
- Fault diagnosis
- Opacity verification
- Intrusion detection
- Conclusions and open issues

Introduction

- The problem of state estimation has been extensively studied in the automatic control community for time-driven systems [Kalman 1960].
- The uncertainty originates from partial information on the initial state.



Luenberger observer for linear time-invariant systems

Introduction

In the eighties, the issue of state estimation started attracting the attention of the DES community:

- automata: Ramadge, 1986; Ozveren and Willsky, 1990; Kumar et al., 1993; ...
- Petri nets: Giua and Seatzu, 2002; Corona et al., 2007; Jiroveanu et al., 2008; Ru and Hadjicostis, 2009a; Cabasino et al., 2010; ...

Two main approaches to state estimation in the DES framework:

1. Control theory approach: assumes systems deterministic $\Rightarrow$ all the events are observable but we have no information (or at most a partial information) on the initial system state;
2. Computer science approach: assumes systems non deterministic $\Rightarrow$ we know the initial state but some events are not observable (silent) or indistinguishable.
   - This is the most common approach.

Set of consistent states

In both cases, there exists a fundamental difference beetween DES and time-driven systems:

- While the state estimation problem in time-driven systems consists in producing an estimate $\hat{x}(t)$ of the actual state $x(t)$, in the DES framework what is computed is an enumerable set of consistent states.

Petri nets allow to characterize this set in linear algebraic terms, while automata based approaches require explicit enumeration.

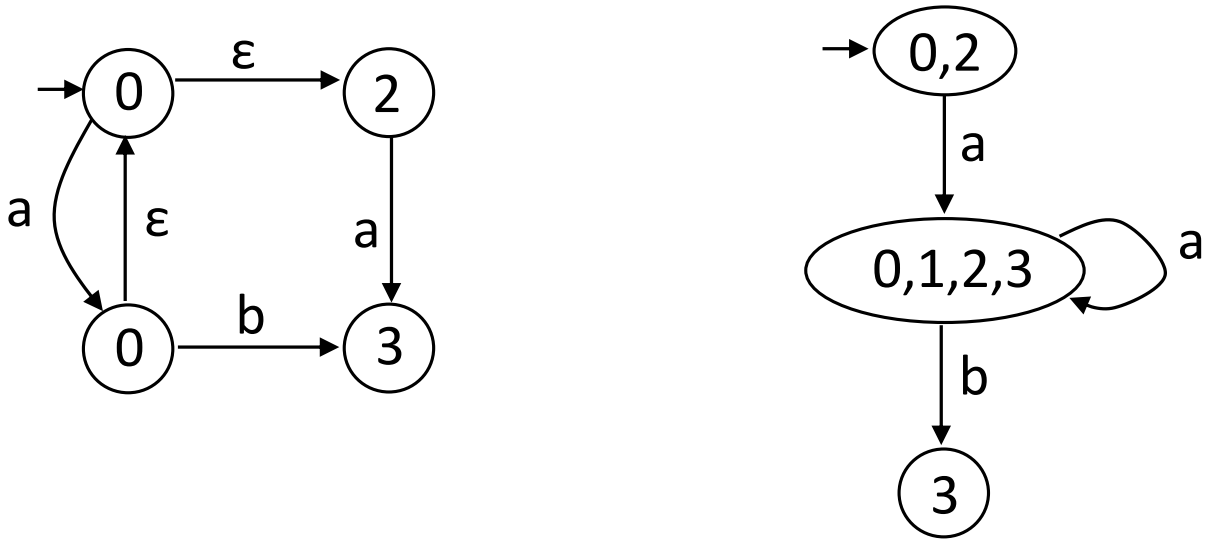
The notion of observer

- Most of the works dealing with the problem of computing the set of states consistent with an observation, are based on the key notion of observer, a useful tool denoted as a deterministic automaton, firstly formalized by Ramadge (1986).

- The observer contains all the information to reconstruct the set of current states consistent with an observation. It allows to move off-line most of the burdensome parts of the computations and can be efficiently used for on-line state estimation.

- The notion of observer proposed under the assumption that the initial state is known, could be generalized to the case where the initial state is not perfectly known.

State observer: a very simple example



An FSA G and its observer ($E_o = \{a, b\}$)

The state of the observer reached following the path labeled with the observation w corresponds to $\mathcal{C}(w)$, namely the set of states consistent with the observation w.

Advantage of using Petri nets

- The notion of observer could obviously be applied to bounded Petri nets: we may construct the observer of the reachability graph.
- However, this would require, as in the case of automata, an exhaustive enumeration of the states.
- This problem could be partially addressed using a more compact representation of the reachability graph, called basis reachability graph (BRG), which uses the notions of basis marking and justification (Cabasino et al., 2010, 2011).

Labeled Petri nets

Labeling function $\mathcal{L} : T \to L \cup \{\varepsilon\}$ assigns to each transition $t \in T$ either a symbol from a given alphabet L or the empty string ε.

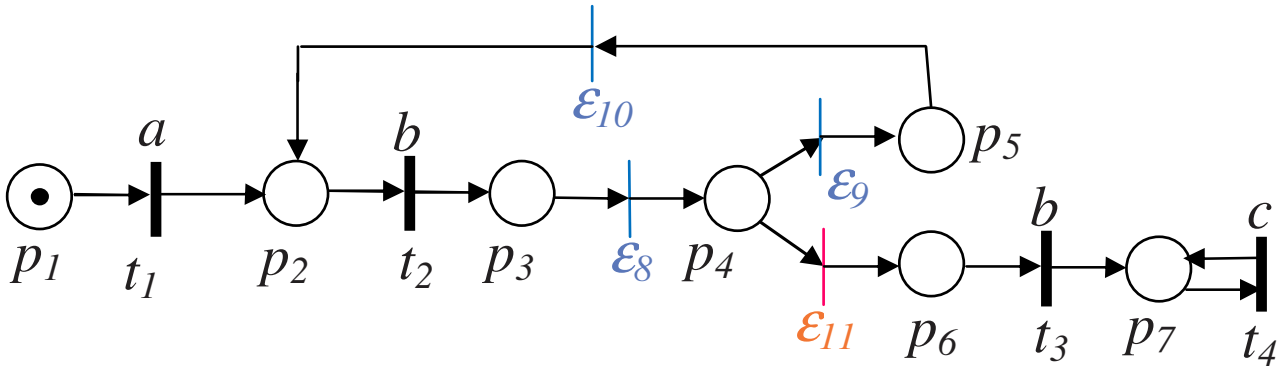
Unobservable (silent) transitions: $T_u = \{t \in T \mid L(t) = \varepsilon\}$.

Observable transitions: $T_o = \{t \in T \mid L(t) = l\}$.

Indistinguishable transitions: $\mathcal{L}(t_1) = \mathcal{L}(t_2) = l \in L$.

$\Rightarrow$ The label is the observed output $\Leftarrow$

Example



$T_o = \{t_1, t_2, t_3, t_4\}$ $\quad T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}\}$

$\mathcal{L}(t_1) = a \quad \mathcal{L}(t_2) = \mathcal{L}(t_3) = b \quad \mathcal{L}(t_4) = c$

$T_f = \{\varepsilon_{11}\}$

Basic notions: justifications and basis markings

ASSUMPTION:

- the T_u-induced subnet is acyclic.

Two key notions:

- Justification
- Basis marking

Justification

- Consider an observation $w \in L^*$. There exists one (or more than one) sequence of transitions $\sigma_o \in T_o^*$ that could have produced the observation.
- In general σ_o is enabled provided that a sequence of unobservable transitions interleaved with it fires.
- We call justification of σ_o the sequence of unobservable transitions σ_u interleaved with σ_o whose firing enables it.
- A justification is said to be minimal if its firing vector is minimal. The firing vector of a minimal justification is called j-vector.

Basis markings

The marking

$$M_b = M_0 + C_u \cdot \pi(\sigma_u) + C_o \cdot \pi(\sigma_o)$$

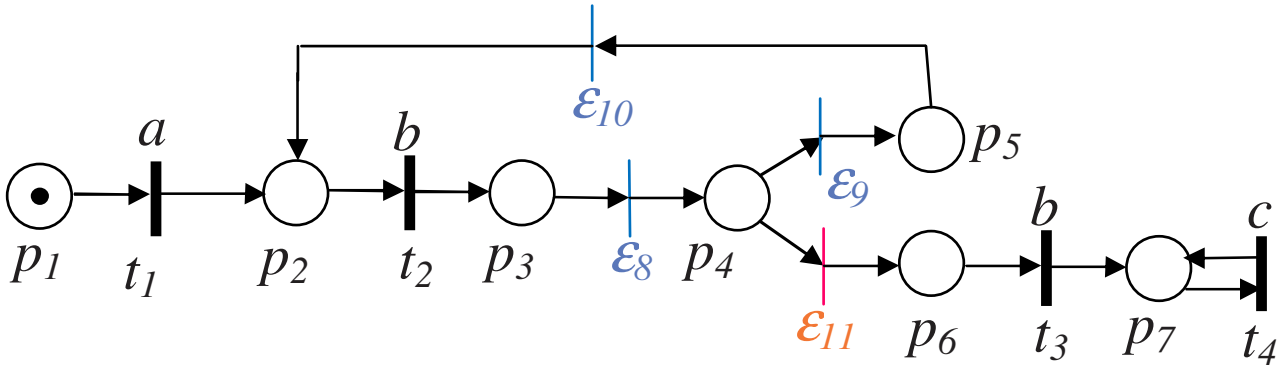
reached firing σ_o interleaved with the minimal justification σ_u, is called basis marking.

Remark: Given an observation w, the set of minimal justifications and the set of basis markings are in general not singleton.

We denote as:

- $\mathcal{J}(w)$ the set of pairs (sequence $\sigma_o \in T_o^*$ that could have produced the observation w - corresponding minimal justification),
- $\mathcal{M}(w)$ the set of pairs (basis marking relative to the observation w - corresponding minimal j-vector),

Example

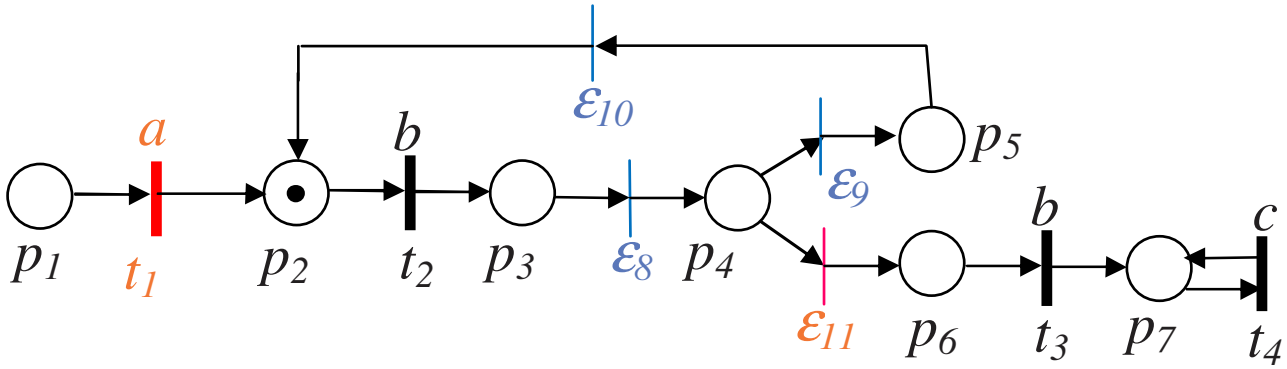


$w = a \quad M_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0]^T$

$\mathcal{J}(w) = \{(t_1, \varepsilon)\} \quad M_b^1 = [0\ 1\ 0\ 0\ 0\ 0\ 0]^T$

$\mathcal{M}(w) = \{(M_b^1, \pi(\varepsilon))\}$

Example

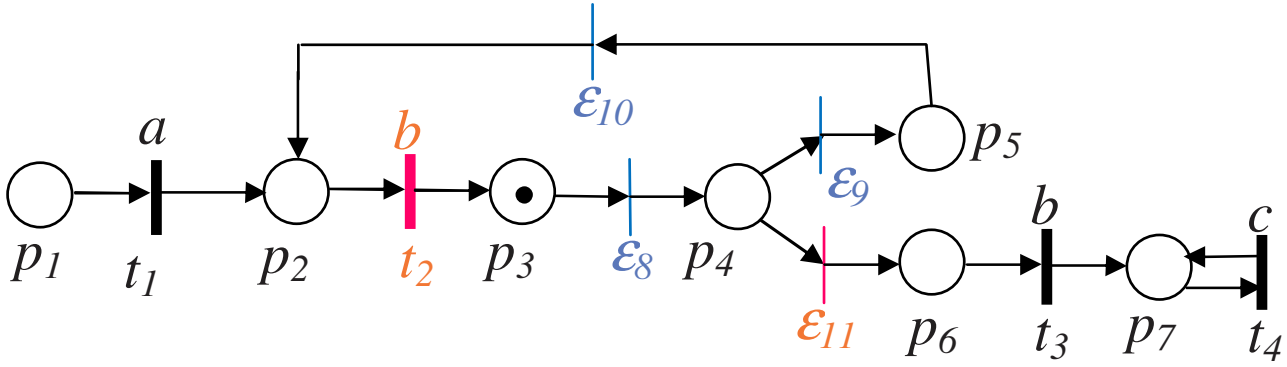


$w = ab \quad M_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0]^T$

$\mathcal{J}(w) = \{(t_1 t_2, \varepsilon)\} \quad M_b^2 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T$

$\mathcal{M}(w) = \{(M_b^2, \pi(\varepsilon))\}$

Example



$w = abb \quad M_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0]^T$

$\mathcal{J}(w) = \{(t_1 t_2 t_2, \varepsilon_8 \varepsilon_9 \varepsilon_{10}), (t_1 t_2 t_3, \varepsilon_8 \varepsilon_{11})\}$

$M_b^2 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T, \qquad M_b^3 = [0\ 0\ 0\ 0\ 0\ 0\ 1]^T$

$\mathcal{M}(w) = \{(M_b^2, \pi(\varepsilon_8 \varepsilon_9 \varepsilon_{10})), (M_b^3, \pi(\varepsilon_8 \varepsilon_{11}))\}$

The set of consistent markings

Definition: The set of markings consistent with $w \in T_o^*$ is

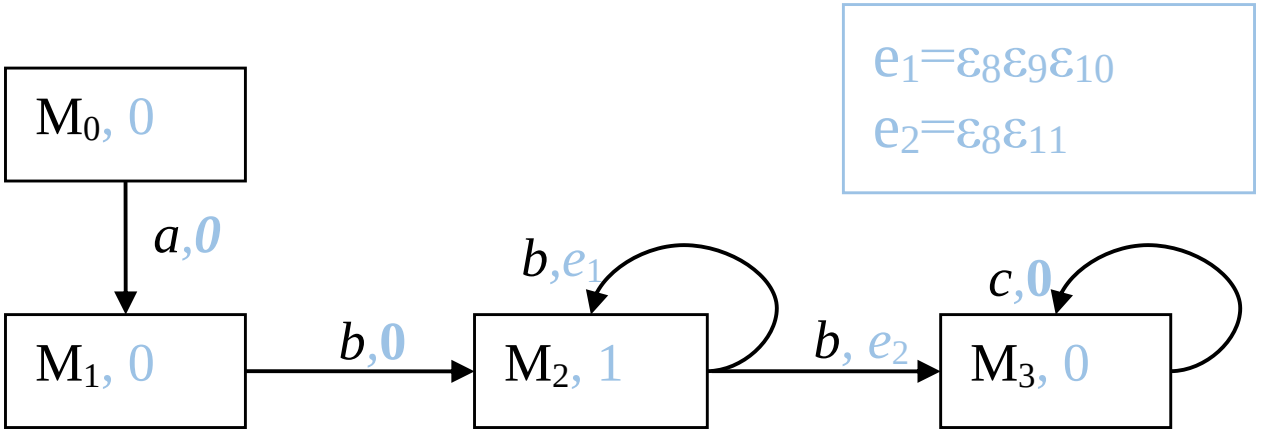
$$\mathcal{C}(w) = \{M \in R(N, M_0) \mid \exists \sigma \in L(N, M_0) : M_0[\sigma\rangle M,\ P_o(\sigma) = w\}.$$

Theorem: Let us consider a net system $\langle N, M_0\rangle$ whose unobservable subnet is acyclic. For any $w \in L^*$ it holds that

$$\mathcal{C}(w) = \{M \in \mathbb{N}^m \mid M = M_b + C_u \cdot j \ : j \geq \vec{0} \text{ and } M_b \in \mathcal{M}_{basis}(w)\},$$

where $\mathcal{M}_{basis}(w)$ is the set of basis markings corresponding to $w \in L^*$.

Bounded net systems $\Longrightarrow$ Basis Reachability graph



$$M_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0]^T$$
$$M_1 = M_b^1 = [0\ 1\ 0\ 0\ 0\ 0\ 0]^T$$
$$M_2 = M_b^2 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$
$$M_3 = M_b^3 = [0\ 0\ 0\ 0\ 0\ 0\ 1]^T$$

We may construct the observer of the BRG

$\Rightarrow$ The Reachability graph has 7 markings $\Leftarrow$

Other contributions by the University of Cagliari

- The initial marking is not known or is known to belong to a given convex set, while all the transitions are observable. The problem of state estimation is studied, several observability properties are introduced, and their decidability properties investigated (Giua and Seatzu, 2002).

- Under the above framework, we studied the problem of state feedback control (Giua, Seatzu, and Basile, 2004).

- The initial marking is known to belong to a finite set of initial markings, each with some a priori probability, and the goal is to obtain the conditional probabilities of possible markings, conditioned on an observed sequence of labels (Cabasino, Hadjicostis, and Seatzu, 2015) .

Other contributions by the University of Cagliari

- The problem of state estimation of Time Petri nets is studied and the Modified State Class Graph is defined (Basile, Cabasino, and Seatzu, 2015).
- The initial marking is known to belong to a given convex set, and some transitions are silent and/or indistinguishable (Cabasino, Hadjicostis, and Seatzu, 2017).
- The BRG has been extended to unbounded nets (Lefaucheux, Giua, and Seatzu, 2018) and the basis coverabilty graph has been introduced.

Outline

- Introduction
- State estimation
- **Fault diagnosis**
- Opacity verification
- Intrusion detection
- Conclusions and open issues

Introduction

- Fault diagnosis consists in the detection of specific events (faults) that lead to an abnormal (faulty) behavior which deviates the plant from its nominal operating conditions.
- Typically addressed under the assumption that:
  - both a model of the nominal and the faulty behaviour are available,
  - faults are unobservable events, whose occurrence should be reconstructed based on the observation of a certain number of events or, in the most general case, on the output signals produced by them.

Introduction

Two different problems need to be investigated.

1. On-line fault diagnosis: consists in deriving a procedure to detect the occurrence of a fault.
2. Off-line fault diagnosis or diagnosability analysis: consists in establishing if the occurrence of a given fault may be detected after a finite number of event occurrences.

Typically, fault events are partitioned into classes when it is not important to distinguish among faults in the same class.

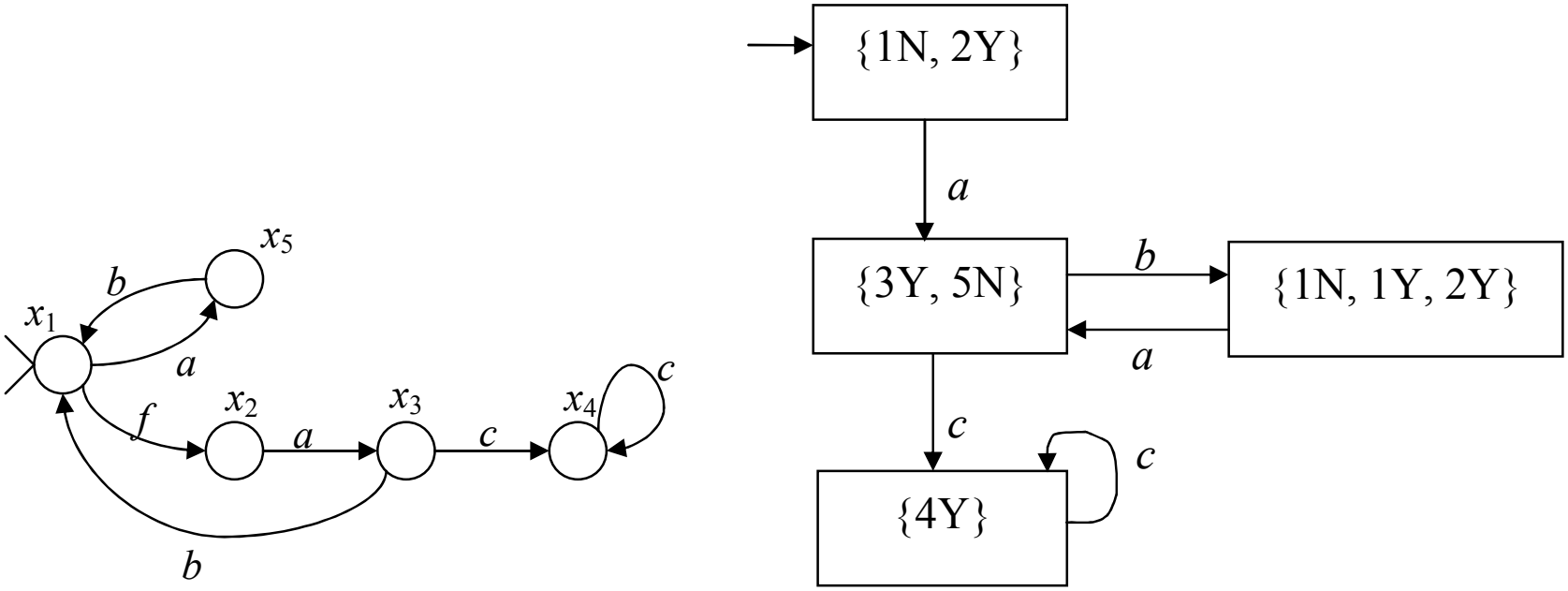
In general, the faults are not the only unobservable events: there may also be events that are unobservable and model a regular behavior.

Literature

- A special attention should be devoted to the seminal papers by Sampath et. (1995, 1996) who introduced the fundamental notion of diagnoser.

- Several other relevant contributions using automata: (Prock, 1991; Zad et al., 2003; Yoo and Lafortune, 2002; Wu and Hadjicostis, 2005),...
  and
- Petri nets: (Lefebvre and Delherm, 2007; Basile et al., 2009; Cabasino et al., 2010; Ru and Hadjicostis, 2009b; Dotoli et al., 2009; Boel and Jiroveanu, 2010; Mahuela et al., 2010; Cabasino et al., 2011; Mahuela et al., 2011; Cabasino et al., 2012; Mahuela et al., 2012; Cabasino et al., 2014), ...

Diagnoser by Sampath et. (1995, 1996)



A FSA G and its diagnoser $Diag(G)$

- States of $Diag(G)$: positive states (a fault has occurred), negative states (no fault), and uncertain states.
- We may study on-line and off-line diagnosis.

Fault diagnosis using labeled Petri nets

The set of unobservable transitions is partitioned as

$$T_u = T_f \cup T_{reg}$$

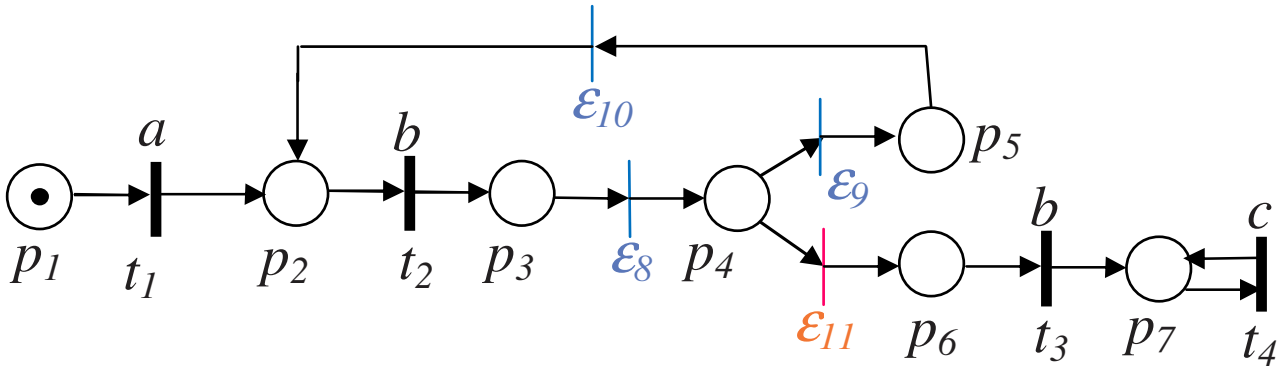
- T_f includes all fault transitions,
- T_{reg} includes all transitions relative to unobservable but regular events.

The set T_f is further partitioned in r fault classes:

$$T_f = T_f^1 \cup T_f^2 \cup \ldots \cup T_f^r.$$

We say that the i-th fault has occurred when a transition in T_f^i has fired.

Example



$T_o = \{t_1, t_2, t_3, t_4\}$ $\quad T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}\}$

$\mathcal{L}(t_1) = a \quad \mathcal{L}(t_2) = \mathcal{L}(t_3) = b \quad \mathcal{L}(t_4) = c$

$T_f = \{\varepsilon_{11}\}$

On-line diagnosis

ASSUMPTIONS:

- The structure of the net and its initial marking M_0 is known;
- the T_u-induced subnet is acyclic.

AIM of on-line diagnosis: Consider a PN system $\langle N, M_0 \rangle$ with labeling function $\mathcal{L}$ and set of transitions T_u partitioned in r fault classes. Given an observed word $w \in L^*$ we want to establish if a fault in the generic class i $(i = 1, \ldots, r)$ has occurred or not.

Diagnoser

A *diagnoser* is a function $\Delta : L^* \times \{T_f^1, T_f^2, \ldots, T_f^r\} \to \{0, 1, 2, 3\}$:

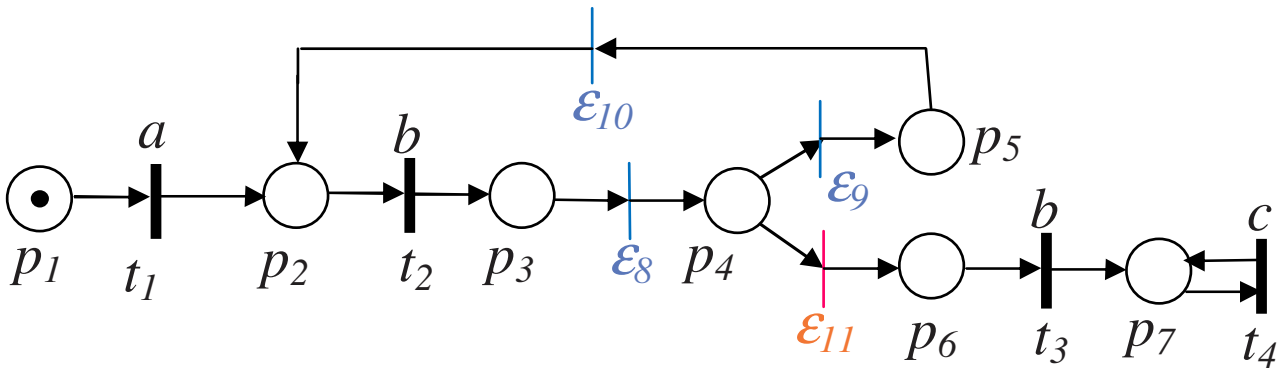
$\Delta(w, T_f^i) = 0 \Rightarrow$ The ith fault cannot have occurred because none of the firing sequences consistent with the observation contains transitions in T_f^i.

$\Delta(w, T_f^i) = 1 \Rightarrow$ The ith fault may have occurred but never while firing a minimal justification of w.

$\Delta(w, T_f^i) = 2 \Rightarrow$ The firing of at least one of the transitions in T_f^i could have occurred while firing a minimal justification of w.

$\Delta(w, T_f^i) = 3 \Rightarrow$ The ith fault must have occurred because all firable sequences consistent with the observation contain at least one transition in T_f^i.

Example

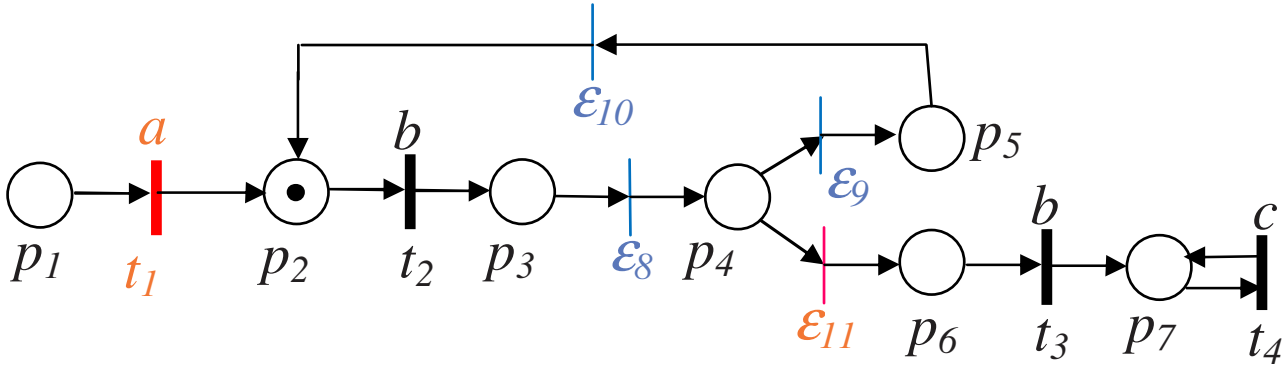


$w = a \quad M_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0]^T$

$\mathcal{J}(w) = \{t_1, \varepsilon\} \quad M_b^1 = [0\ 1\ 0\ 0\ 0\ 0\ 0]^T \Longrightarrow \Delta(T_f, a) = 0$

For sure the fault has not occurred.

Example

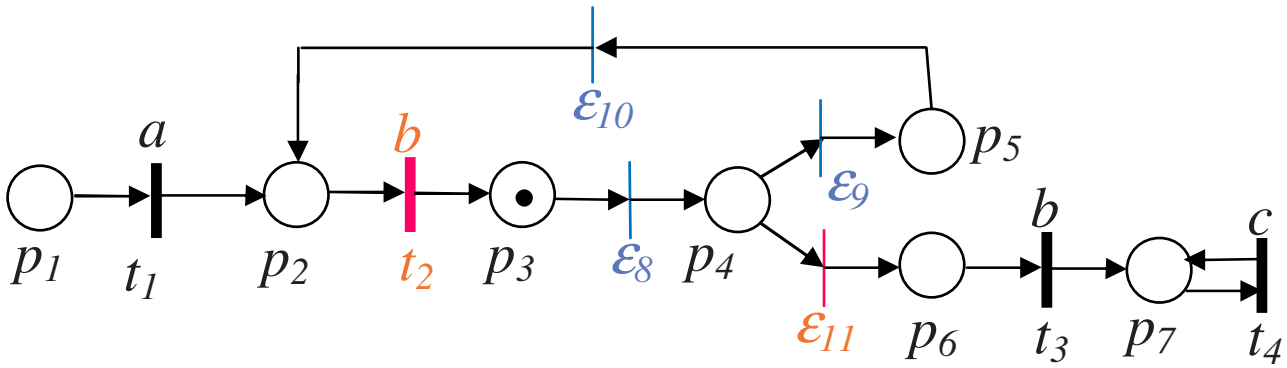


$w = ab \quad M_b^1 = [0\ 1\ 0\ 0\ 0\ 0\ 0]^T$

$\mathcal{J}(w) = \{t_1 t_2 \varepsilon\} \quad M_b^2 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T \Longrightarrow \Delta(T_f, ab) = 1$

The fault could have occurred, but not while reaching a basis marking.

Example



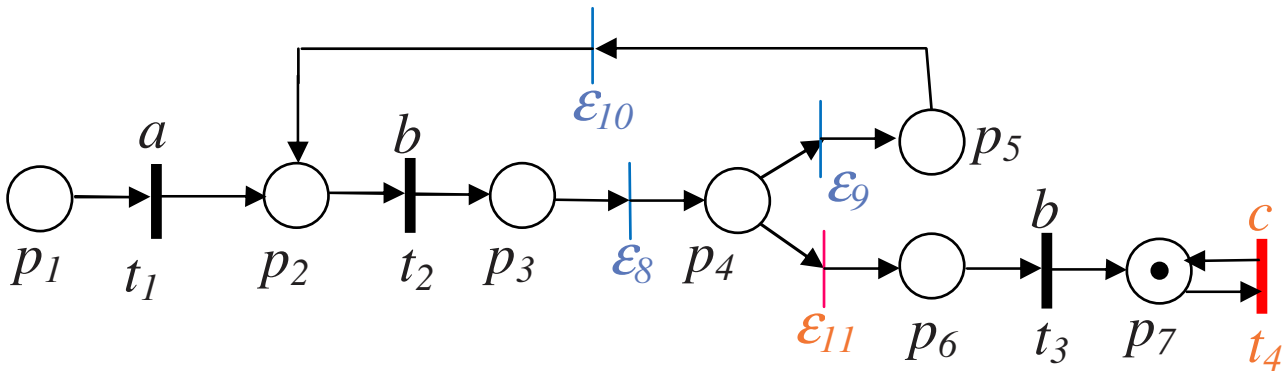
$w = abb \quad M_b^2 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T$

$j_1 = \varepsilon_8\varepsilon_9\varepsilon_{10} \quad M_b^2 = [0\ 0\ 0\ 0\ 0\ 0\ 1]^T$

$j_2 = \varepsilon_8\varepsilon_{11} \quad M_b^3 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T \Longrightarrow \Delta(T_f, abb) = 2$

The fault could have occurred while reaching one of the two basis markings.

Example

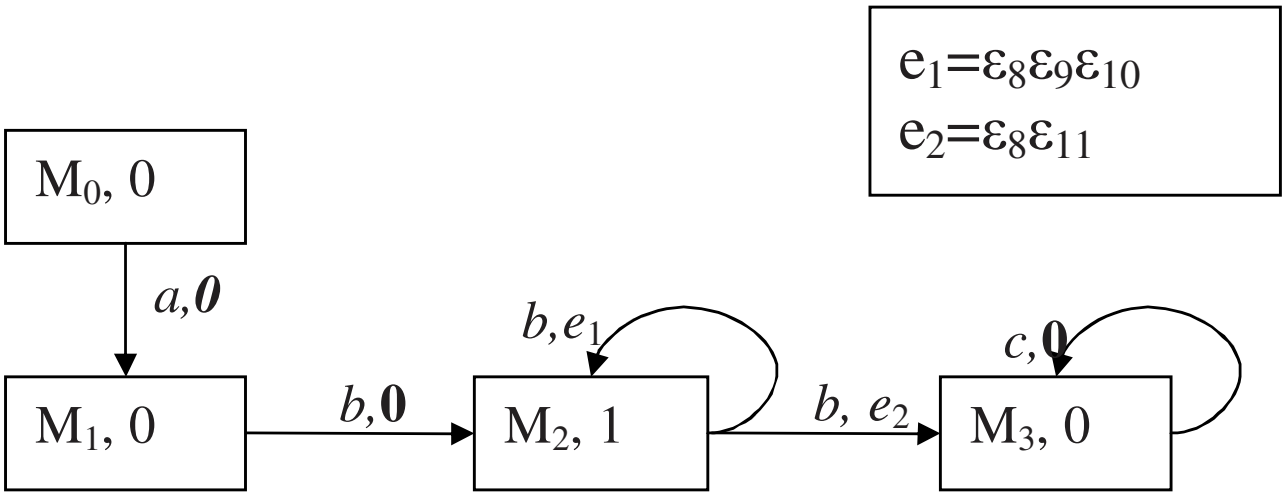


$w = abbc \quad M_b^3 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T$

$j_2 = \varepsilon_8\varepsilon_{11} \quad M_b^3 = [0\ 0\ 1\ 0\ 0\ 0\ 0]^T \Longrightarrow \Delta(T_f, abbc) = 3$

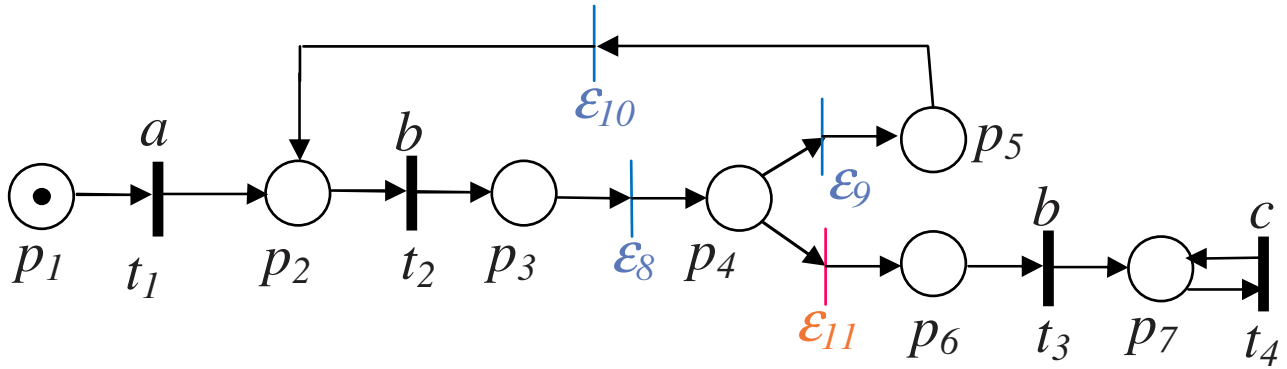
For sure the fault has occurred.

Bounded net systems $\Longrightarrow$ Basis Reachability graph

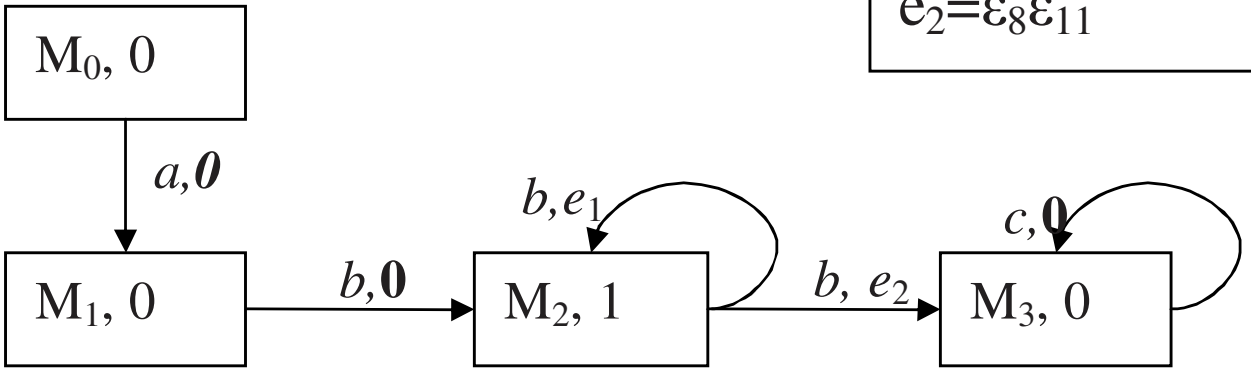


- The second entry in a node is a vector with as many entries as the number of fault classes: the ith entry is 1 if a fault in the ith class could fire at the basis marking in the node, 0 otherwise.
- The second entry in an arc is a minimal j-vector.

Basis Reachability graph and on-line diagnosis



$e_1=\varepsilon_8\varepsilon_9\varepsilon_{10}$
$e_2=\varepsilon_8\varepsilon_{11}$



Diagnosability analysis

- We can build the BRG assuming that fault transitions are observable: Modified BRG (MBRG).
- Starting from the MBRG, and following the same ideas of the Diagnoser of Sampath et al., we can construct the Basis Reachability Diagnoser (BRD).
- Diagnosability analysis could be performed based on the analysis of indeterminate cycles in the BRD.
- Again, the notions of basis marking and justification lead to advantages in terms of computational complexity because we do not need to exhaustively enumerate all the reachable states.

Other contributions by the University of Cagliari

- Diagnosability analysis of unbounded Petri nets (Cabasino, Giua, Lafortune, and Seatzu, 2012).
- Fault diagnosis of continuous Petri nets (Mahuela, Seatzu, Cabasino, and Silva, 2012).
- Fault diagnosis and diagnosability analysis of Time Petri nets (Basile, Cabasino, Seatzu, 2015, 2017).
- Optimal sensor selection for ensuring diagnosability in labeled Petri nets (Cabasino, Lafortune, and Seatzu, 2013).
- Codiagnosability analysis of bounded Petri Nets (Ran, Su, Giua, Seatzu, 2018).

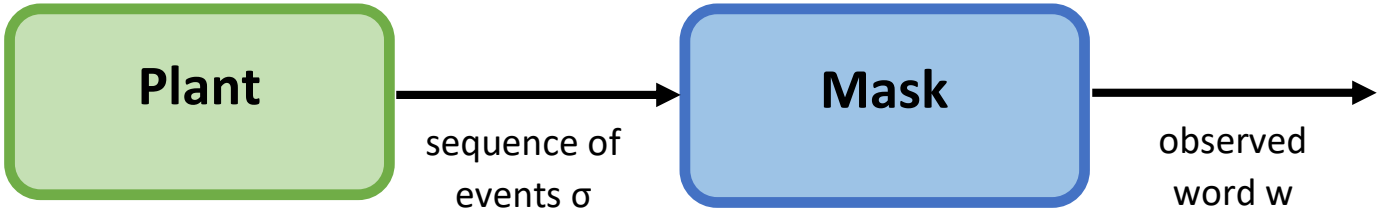
Outline

- Introduction
- State estimation
- Fault diagnosis
- **Opacity verification**
- Intrusion detection
- Conclusions and open issues

Introduction: opacity is related to privacy and security

Consider a malicious observer, called intruder, who

- knows the structure of the system,
- partially observes the system.

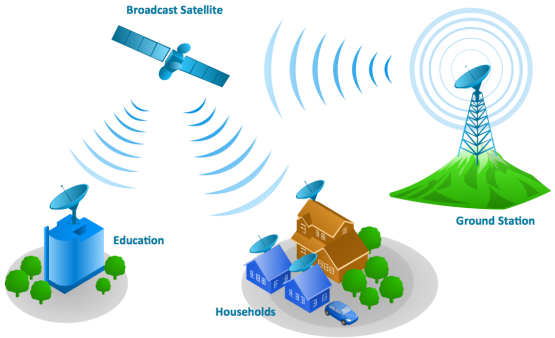


Opacity characterizes whether a given "secret" about the system behavior is hidden or not from the intruder.

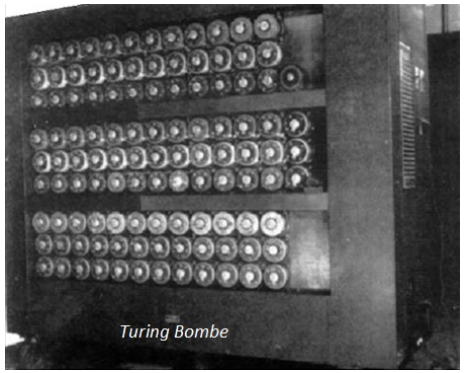
Based on the info on the system's structure and its observation w, the intruder constructs an estimate of the system's behavior. The secret is said to be opaque if the intruder's estimate never reveals the system's secret.

Motivations

Internet Safety



Communication Network



Military Defence

MANY OTHERS...

Opacity properties of DESs

Opacity was first introduced in 2004 in computer science by Mazarè (2004) to analyze cryptographic protocols, and then formulated in terms of Petri nets by Bryans et al. (2008).

Opacity formalizes the impossibility for an intruder to infer the truth of a predicate representing the secret information.

In DESs, the predicate can be a language or a subset of the state space, therefore, opacity properties can be categorized into two main classes: language-based opacity and state-based opacity.

Different formulations in the two classes have been presented in the literature. The most significant ones are recalled in the following.
For simplicity we introduce them using FSA as the reference model.

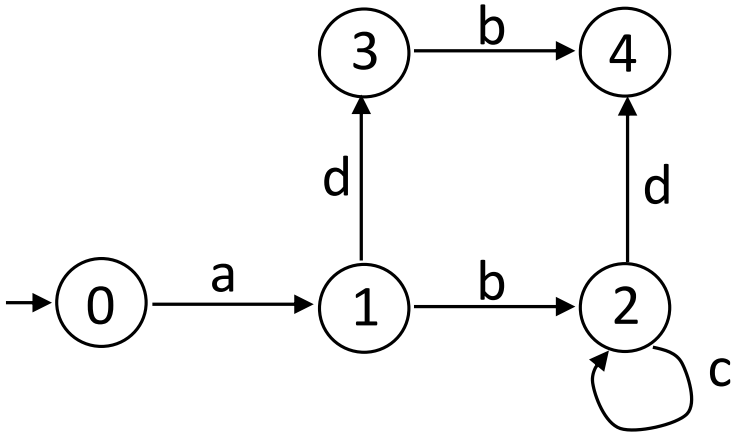
Language-based opacity - LBO

[Def. first introduced by Badouel et al. (2007) and Dubreil et al. (2008)]

Given a system $G = (X, E, f, X_0)$, where $E = E_o \cup E_{uo}$, a secret language $L_S \subseteq \mathcal{L}(G, X_0)$, and a non-secret language $L_{NS} \subseteq \mathcal{L}(G, X_0)$, G is language-based opaque if $\forall t \in L_S$, $\exists t' \in L_{NS}$ such that $P(t) = P(t')$.

In words, the system is LBO if for any string $t \in L_S$, there exists, at least, one string $t' \in L_{NS}$ with the same projection. Therefore the intruder, looking at $P(t) = P(t')$, cannot conclude whether the secret string t or the non-secret string t' has occurred.

LBO: examples [from Wu & Lafortune (2013)]



Let $E_o = \{a, b, c\}$.

- Let $L_S = \{abd\}$ and $L_{NS} = \{abcc^*d, adb\}$: G is LBO.
  Whenever the intruder sees $P(L_S) = \{ab\}$, it is not sure whether abd or adb has occurred.
- Let $L_S = \{abcd\}$ and $L_{NS} = \{adb, abd, abccc^*d\}$: G is not LBO.
  No string in L_{NS} has the same projection as the secret string $abcd$.

Current-state opacity - CSO

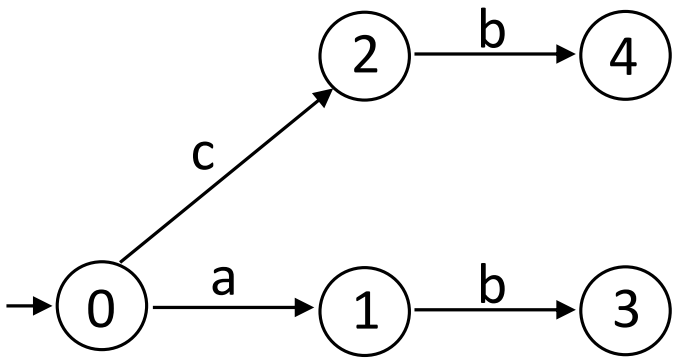
[Def. first introduced by Bryans et al. (2005) in the context of PNs, then extended to FSA by Saboori and Hadjicostis (2007)]

Given a system $G = (X, E, f, X_0)$, where $E = E_o \cup E_{uo}$, a set of secret states $X_S \subseteq X$, and a set of non secret states $X_{NS} \subseteq X$, G is current-state opaque if

$$\forall i \in X_0 \text{ and } \forall t \in \mathcal{L}(G, i) \text{ such that } f(i, t) \in X_S,$$
$$\exists j \in X_0 \text{ and } \exists t' \in \mathcal{L}(G, j) \text{ such that } f(i, t') \in X_{NS} \text{ and } P(t) = P(t').$$

In words, the system is CSO if for every string t that leads to a secret state, there exists another string t' with the same projection, leading to a non-secret state. Therefore the intruder, looking at $P(t) = P(t')$, cannot conclude whether the system state is in a secret or in a non-secret state.

CSO: examples [from Wu & Lafortune (2013)]



Let $X_S = \{3\}$ and $X_{NS} = X \setminus X_S$.

- Let $E_o = \{b\}$: G is CSO.
  The intruder cannot distinguish between ab (leading to the secret state 3) and cb (leading to the non-secret state 4).
- Let $E_o = \{a, b\}$: G is not CSO.
  When observing ab the intruder knows for sure that the system is in the secret state 3.

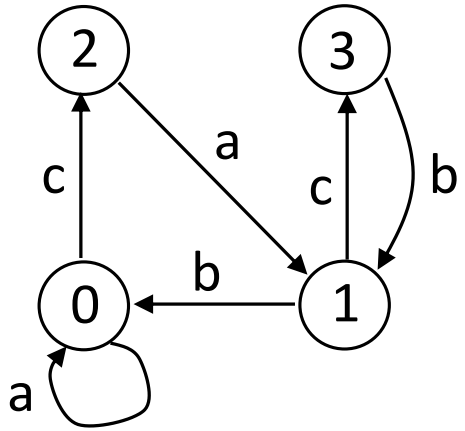
Initial-state opacity - ISO

Given a system $G = (X, E, f, X_0)$, where $E = E_o \cup E_{uo}$, a set of secret initial states $X_S \subseteq X_0$, and a set of non secret initial states $X_{NS} \subseteq X_0$, G is initial-state opaque if

$$\forall i \in X_S \text{ and } \forall t \in \mathcal{L}(G, i),$$
$$\exists j \in X_{NS} \text{ and } \exists t' \in \mathcal{L}(G, j) \text{ such that } P(t) = P(t').$$

In words, the system is ISO if, for every string t that originates from a secret state, there exists another string t' originating from a non-secret state such that $P(t) = P(t')$. Therefore the intruder, looking at $P(t) = P(t')$, cannot establish whether the system started from a secret or a non-secret state.

ISO: examples [from Wu & Lafortune (2013)]



Let $E_o = \{a, b\}$.

- Let $X_S = \{2\}$ and $X_{NS} = X \setminus X_S$: G is ISO.
  For every string t starting from state 2, there is another string ct starting from state 0 that produces the same observation.
- Let $X_S = \{0\}$ and $X_{NS} = X \setminus X_S$: G is not ISO.
  Whenever the intruder observes aa, it knows for sure that the system started from state 0.

Other notions of opacity

- Initial-and-Final-State Opacity (IFSO): $X_{SP} \subseteq X_0 \times X$ and $X_{NSP} \subseteq X_0 \times X$. $\Rightarrow$ A system is IFSO if for any string t that starts from x_0 and ends in x_f, with $(x_0, x_f) \in X_{SP}$, there exists another string t' starting from x'_0 and ending in x'_f, where $(x'_0, x'_f) \in X_{NSP}$, which has the same projection.
- K-step Opacity, Infinite-step Opacity,
- Probabilistic notions of Opacity,
- .....
- Distributed Opacity: more than one intruder.
  - Each intruder has its own observation mask and secret. The system is said concurrently opaque if all secrets are safe.
  - Several intruders collaborate through a coordinator in order to discover the same secret (joint opacity).
  - .....

CSO Verification

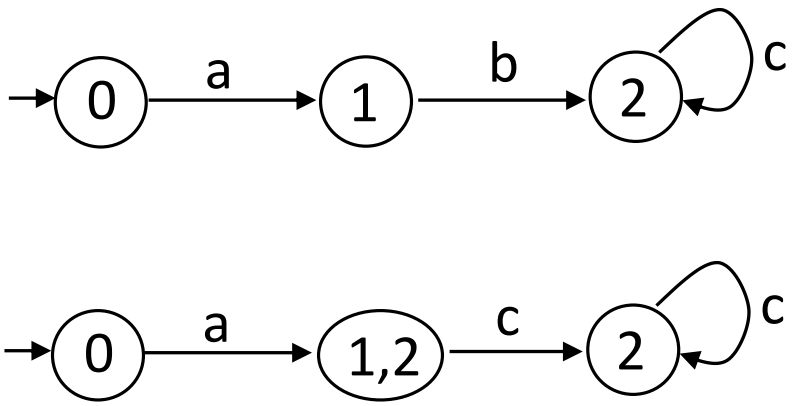
- Given a system G and a secret $X_S \subseteq X$ ($X_{NS} = X \setminus X_S$), G is current-state opaque wrt X_S if $\forall w$,

$$\mathcal{C}(w) \not\subseteq X_S.$$

Thus, verification could be done using the notion of state observer (Ramadge, 1986).

CSO Verification: main result

G is CSO wrt $X_S \iff$ in the observer of G there does not exist a state that is a subset of X_S.



$E_o=\{a,c\}$

- If $X_S = \{1, 2\}$, G is not CSO wrt X_S.
- If $X_S = \{1\}$, G is CSO wrt X_S.

The complexity is $O(2^n)$ where $n = |X|$.

Advantages and disadvantages of automata

Advantages of automata

- Easy to formalize the problem.
- When the secret changes, there is no need to reconstruct the observer.

Disadvantages of automata

- Each set $\mathcal{C}(w)$ must be exhaustively enumerated.
- To compute $\mathcal{C}(w)$ we need to compute $\mathcal{C}(w')$ for all prefixes $w' \preceq w$.
- If the NFA has n states, the observer can have up to 2^n states.
- Not able to reconstruct the event sequences consistent with the observation.

CSO Verification using Petri nets

- Opacity problems have also been studied using Petri nets.
- In particular, the CSO problem in bounded Petri nets can be efficiently solved using the basis reachability graph (Tong, Li, Seatzu, and Giua, 2017) .
- This approach is practically efficient since the exhaustive enumeration of the reachability space can be avoided.

Verification of CSO using Petri nets

Weakly Exposable Marking: A marking M is said to be weakly exposable if $\mathcal{U}(M) \not\subseteq X_S$, where $\mathcal{U}(M) = \{M' \in \mathbb{N}^m | M[\sigma_u\rangle M', \sigma_u \in T_u^*\}$.

We denote $\mathcal{M}_b(w) = \mathcal{C}(w) \cap \mathcal{M}_b$ the set of basis markings consistent with w.

Theorem: Let G be an LPN whose T_u-induced subnet is acyclic and X_S be a secret. G is CSO wrt X_S iff $\forall w \in E^*$, $\exists M_b \in \mathcal{M}_b(w)$ that is weakly exposable.

BRG for current-state opacity

Each state of the BRG is associated with a pair $(M_b, \alpha(M_b))$, where $\alpha(M)$ is a binary scalar that is defined as follows.

$$\alpha(M_b) = \begin{cases} 1 & \text{if } M_b \text{ is weakly exposable;} \\ 0 & \text{otherwise.} \end{cases}$$

CSO Verification Algorithm

Input: A bounded LPN whose T_u-induced subnet is acyclic;
A secret X_S.

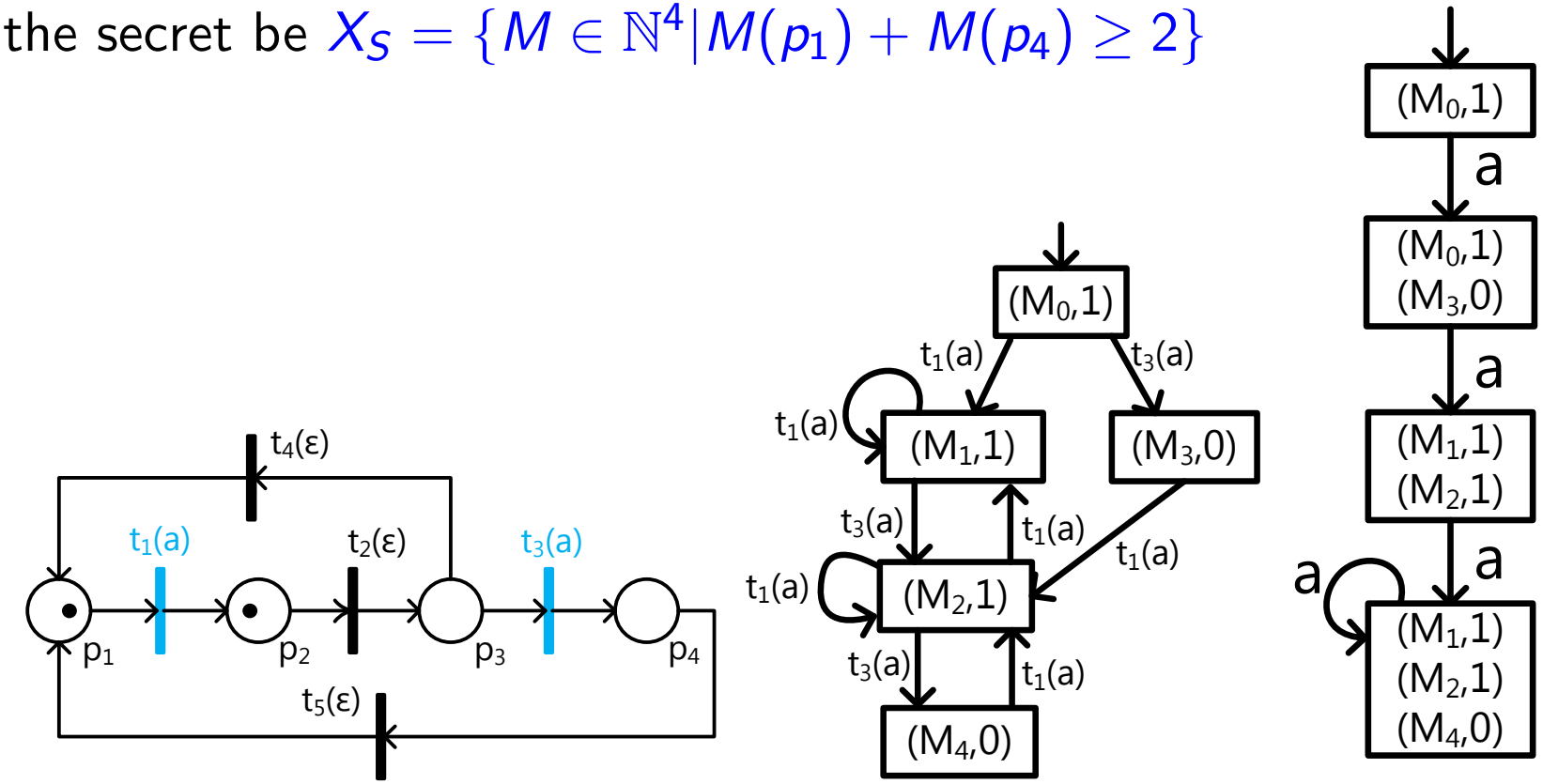
Output: $Opac = 1$, the LPN is current-state opaque wrt X_S;
otherwise, $Opac = 0$

1. Construct the BRG for current-state opacity.
2. Construct the observer for the BRG.
3. **If** all states of the observer have a pair $(M, 1)$
   $Opac = 1$;
   else
   $Opac = 0$.
   end
4. Output $Opac$.

The complexity of verifying current-state opacity is $O(2^{|\mathcal{M}_b|})$.
(Using the automaton-based approach, the complexity is $O(2^{|R(N,M_0)|})$)

Verification of CSO: Example

Let the secret be $X_S = \{M \in \mathbb{N}^4 | M(p_1) + M(p_4) \geq 2\}$

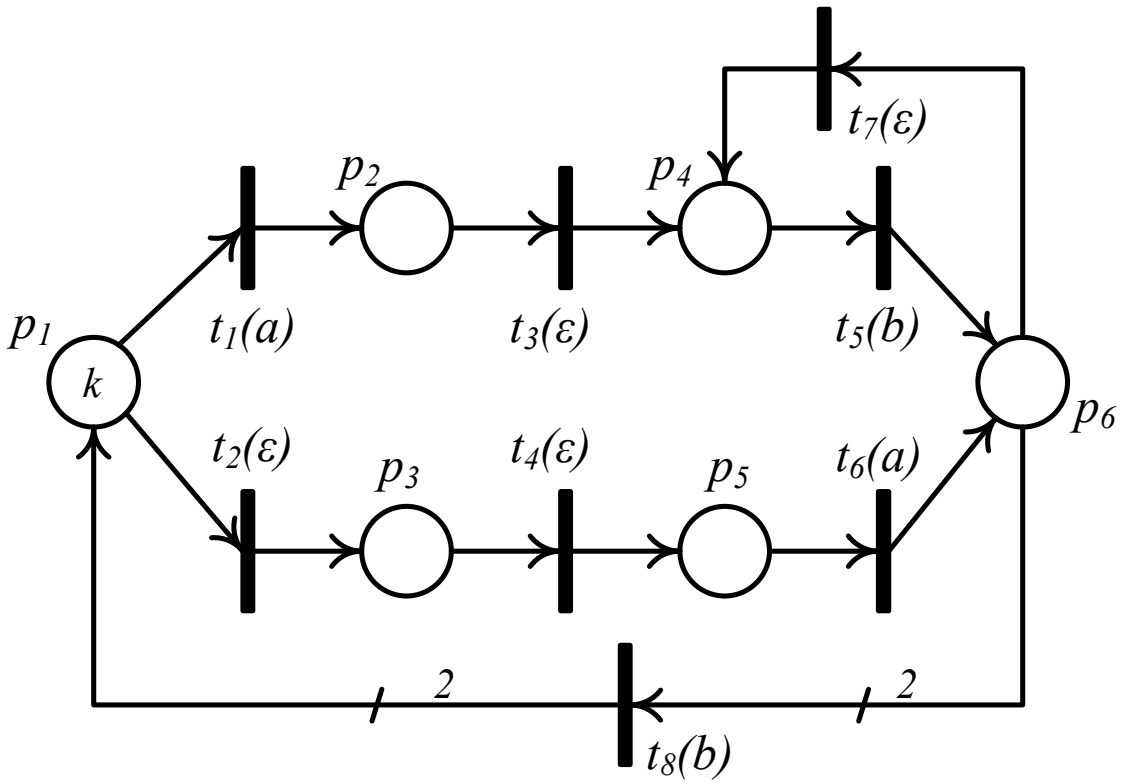


An LPN, its BRG, and the observer of its BRG

The net system is CSO wrt X_S.

Verification of CSO: Test

Consider the LPN with initial marking $M_0 = [k\ 0\ 0\ 0\ 0]^T$, $k \in \{2, 3, \cdots\}$, and $X_S = \{M \in \mathbb{N}^4 | M(p_3) + M(p_5) - M(p_2) - M(p_4) \geq 1\}$.



Verification of CSO: Results

T_{RG} (resp., T_{BRG}: time [s] to compute RG (resp. BRG);
T_{ObRG} (resp., T_{ObBRG}): time [s] to compute the observer of RG (resp. BRG).

k	$\|R(N, M_0)\|$	T_{RG}	$\|\mathcal{M}_b\|$	T_{BRG}
9	2002	57	55	0.16
10	3003	130	66	0.20
11	4368	280	78	0.24
12	6188	600	91	0.29
16	o.t.	o.t.	153	0.58
17	o.t.	o.t.	171	0.67

k	$\|\mathcal{X}_{ObRG}\|$	T_{ObRG}	$\|\mathcal{X}_{ObBRG}\|$	T_{ObBRG}	CSO
9	45	60	45	0.16	Y
10	54	150	54	0.24	Y
11	61	310	61	0.30	Y
12	71	720	71	0.41	Y
16	o.t.	o.t.	111	1.00	Y
17	o.t.	o.t.	121	1.20	Y

Another interesting problem

Opacity enforcement: this is another problem that has received a lot of attention in the DES community (Badouel et al., 2007; Dubreil et al., 2010; Sa- boori and Hadjicostis, 2012; Yin and Lafortune, 2016a,b; Tong et al., 2018).

$\Rightarrow$ Given a system that is not opaque, the opacity enforcement problem consists in turning the system into an opaque one.

Approaches to opacity enforcement may rely on:

- supervisory control,
- dynamically restraining the observability of events,
- inserting additional events in the output behavior of the system,
- ...

Remark

- The problems of state estimation and fault diagnosis are similar $\Longrightarrow$ both require the guaranty that a certain information on the system evolution may be detected on the basis of available information.
- Therefore, the more information an observer or a diagnoser has on the system evolution, the more it is likely to reconstruct the system state or to detect a fault occurrence.
- On the contrary, opacity requires that certain information related to the system evolution are hidden to the intruder. Indeed, the more information an intruder has on the system evolution, the more it is likely it will discover the secret.

Outline

- Introduction
- State estimation
- Fault diagnosis
- Opacity verification
- **Intrusion detection**
- Conclusions and open issues

Problem statement

- An external agent may affect the observation produced by the system during its evolution $\Longrightarrow$ adding or erasing information produced by sensors.

- From the point of view of the external malicious agent:
  - $\Longrightarrow$ the goal may be that of synthesizing an attack strategy that affects the state estimation of a system observer, preventing it from detecting that an unsafe critical state has been reached.
  - $\Longrightarrow$ If the system is under control, the attack could be at the supervisory control layer and may lead the controlled system to reach unsafe critical states (Goes et al., 2017).

Intrusion prevention

- From an internal point of view:

  $\Longrightarrow$ defence strategies should be defined to realize when an attack has occurred (intrusion detection) and/or to protect a system from reaching unsafe states after an attack (intrusion prevention).

  - This can be done for example, designing supervisors that enforce safety and liveness specifications and at the same time are robust to deception attacks (Carvalho et al., 2016). In such a case, we talk about intrusion prevention.

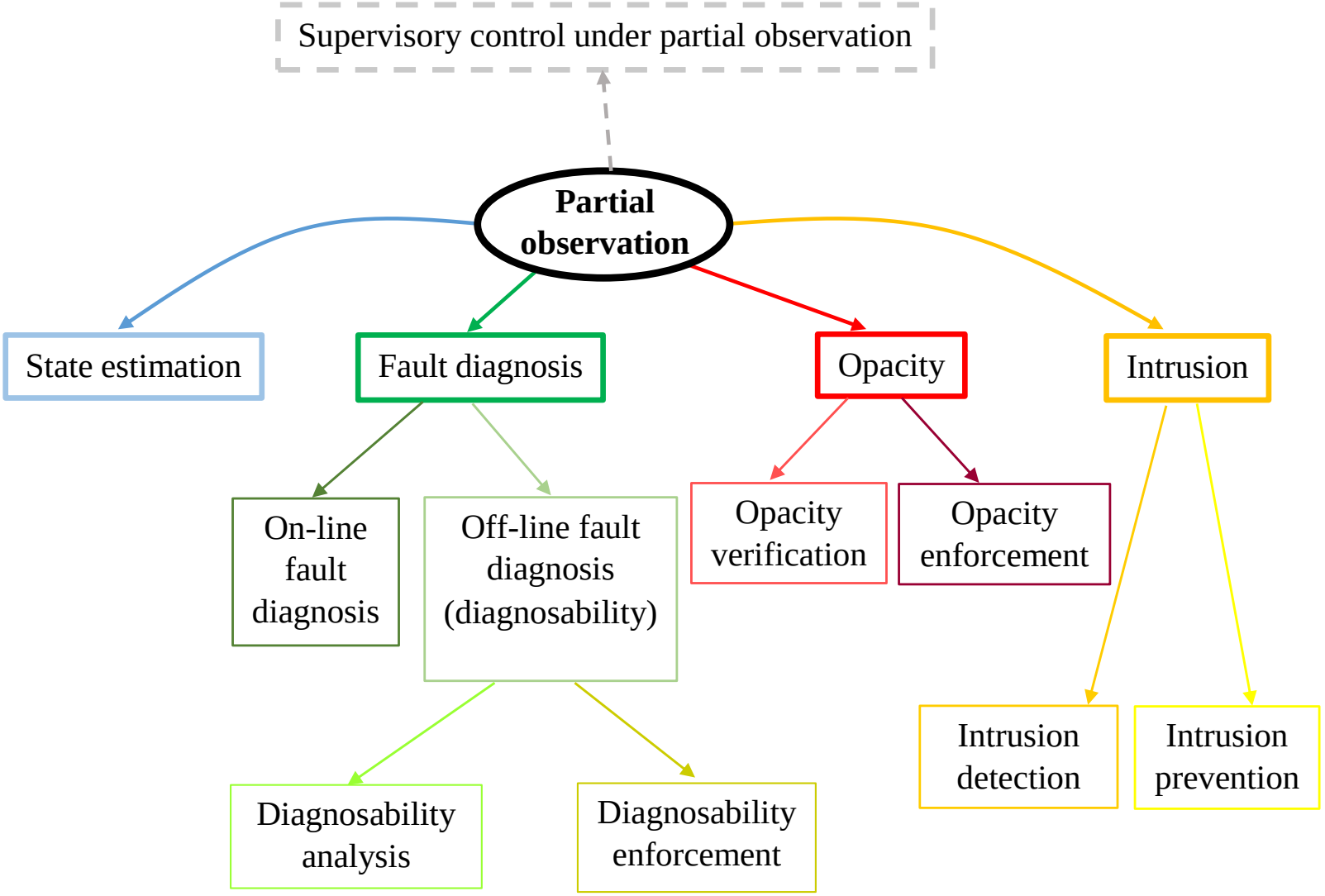
All such problems can obviously be formalized in the framework of

$\Rightarrow$ partial observation. $\Leftarrow$

Outline

- Introduction
- State estimation
- Fault diagnosis
- Opacity verification
- Intrusion detection
- **Conclusions and open issues**

Conclusions



Open issues

- Generalize some of the results proposed in a centralized setting, to a distributed or a decentralized setting;
- reformulate and solve some of the problems mentioned above in a probabilistic setting;
- more extensively address the problems of intrusion detection and prevention, particularly using Petri nets;
- advance the state of the art on timed discrete event systems;
- extend some of the above approaches to the case of systems with an infinite state space.

THANKS FOR YOUR ATTENTION

- ANY QUESTIONS?