Fault Diagnosis of Discrete-Event Systems

Alejandro White, Doctoral Candidate

Advisor: Dr. Karimoddini

ACCESS Laboratory

North Carolina Agricultural and Technical State University

TECHLAV

Motivation

- Faults are always
- Faults are unwanted
- Faults are arbitrary
- Faults are costly
- Faults are DEADLY

Our motivation for the provision of fault diagnostics is simple: we wish to minimize an everlasting, unpredictable, life destroying entity.

TECHLAV Project

- Testing, Evaluation and Control of Heterogeneous Large-scale Autonomous systems of Vehicles (TECHLAV)▫

- **Thrust 2:** Resilient Control and Communication of Large-scale Autonomous Vehicle

- **Task 2-1:** Develop fault detection and isolation mechanism

TECHLAV Project

Objective	Impact
To develop techniques for automatic diagnosis of failures in the system to timely diagnose (detect, identify and locate) occurred.	Upon a fault occurrence, a system will autonomously become aware of the fault's occurrence, and initiate a systematic procedure that isolates, identifies, and accommodates the fault in order to ensure proper utilization of the system's remaining resources, allowing a resilient post fault system operation that is both safe and stable.

Outline

- Definition of Fault
- Definition of Fault Diagnosis
- Survey of Methods of Fault Diagnosis
- Formulation of Fault Diagnosis within Discrete-Event System
- Constructing the Diagnoser
- Diagnosability Condition
- Future Work

What is fault?

- Fault - a malfunction in system component(s) (actuators, sensors,…etc.) that results in unacceptable system performance, and/or system instability

Fault Diagnosis

- Fault Diagnosis - the detection of a fault's occurrence conjoined with the identification of a fault's nature, through examination of a system's symptoms
  - **Fault detection:** If a fault has occurred?
  - **Fault identification:** What is the type and nature of failure?
  - **Fault isolation:** Where in the system has occurred?

Why do we need Fault Diagnosis?

- To better accommodate system behavior post fault occurrence
  - Ensures system stability
  - Increases system reliability
  - Reduce number of failed missions
  - Save lives

State of the Art

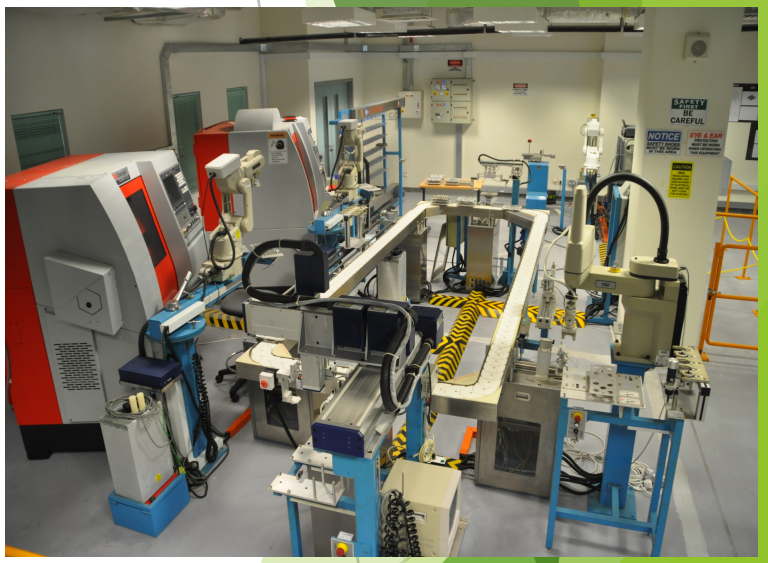
- Analytical Model Based: modelled system operation is compared to observed system operation
  - Residuals - comparison of observed signals from the system with predicted values; residuals are usually designed to be zero if not fault present (Frank & Ding, 1997; Roth et al., 2011)
  - Templates – specify the expected correct timing and sequencing of events (Holloway & Chand, 1994)
  - Fault free - observed system operation is compared to a nominal fault-free model (Pandalai & Holloway, 2000)

- Non-Model Based: a single abstract representation encompassing normal and faulty system operation is analyzed
  - State based - system condition (failure status) is determined by state or set of states the system belongs to (Lin, 1993; Zad et al., 2003)
  - Event based - system failure determined by observance of sequences of events (Sampath et al., 1995)
  - Fault tree - fault diagnosis method based upon deductive fault analysis (Vesely et al., 1981; Lee et al., 1985)

- Knowledge Based: heuristic
  - Expert system - past knowledge obtained by experts used to model unknown system aspects (Scherer & White, 1989; Handelman & Stengel 1989)
  - Artificial Neural Network - an abstract model of the brain's neural pathways designed to actively "learn" the normal and faulty behavior of a system (Elias Kosmatopoulos & Polycarpou, 1995; Diao & Passino, 2001)

Why Discrete Event System Framework?

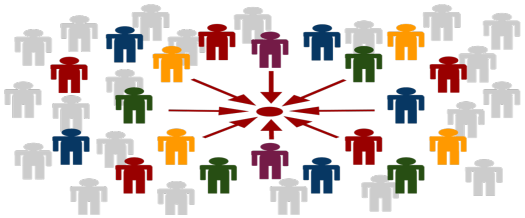
- DES is an Event-driven time abstract formalism suitable for large-scale complex systems



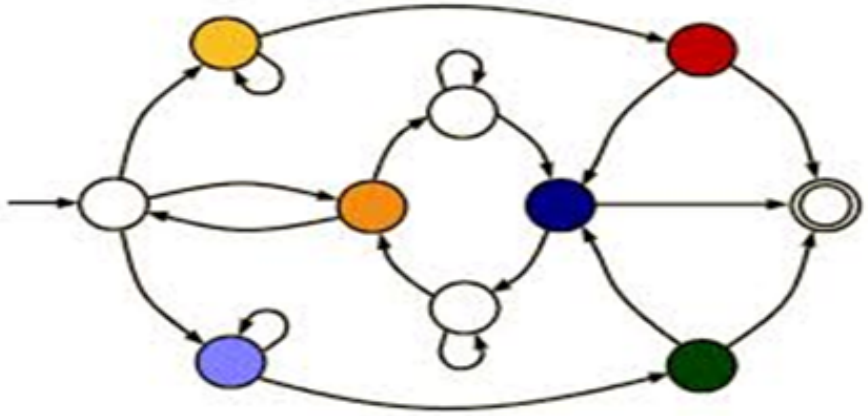
- For diagnostic purposes, several large and complex real-world systems are successfully modeled as Discrete-Event Systems (e.g., cyber networks, manufacturing systems, smart grids)

- Naturally captures faults as abrupt changes (e.g., sequence of events)

- Matches human thinking
  - coordination (e.g., interactions of systems) *group*
  - cause and effect (e.g., a fault causing event sequence)



Automaton



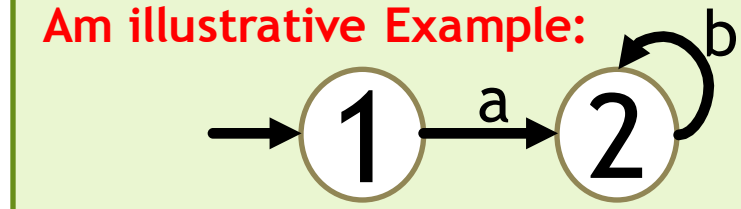
- Definition: a non-deterministic finite-state Discrete-Event System (DES) can represented by a four-tuple

$$G = (X, \Sigma, \delta, x_o)$$

  - State space: X
  - Event set: $\Sigma = \Sigma_o \cup \Sigma_u$
    - ❑ Events (∑) : Notable occurrence of asynchronous discrete changes in a system
    - ❑ Observable events ($\sum_o$): Events observed by a sensor (e.g., opening of valve)
    - ❑ Unobservable events ($\sum_u$)- Events that are unable to be detected by sensors; possibly due to sensor absence/damage (e.g., failure event)

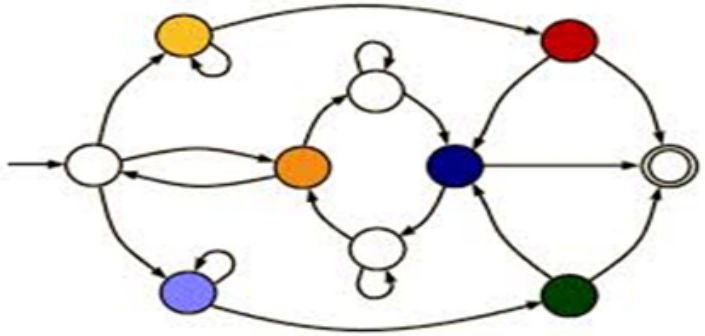
  - State-transition relation: a partial relation that determines all feasible system state transitions caused by system events $\delta : X \times \Sigma \rightarrow 2^X$
  - Initial state: $x_o \in X$

Am illustrative Example:



X = {1,2}
Σ = {a,b}
$\delta(1, a) = 2; \delta(2, b) = 2$
x_0 = 1

Language



Definition: the system language is a discrete representation of the system's behaviors (normal and faulty) in the form of sequences of events

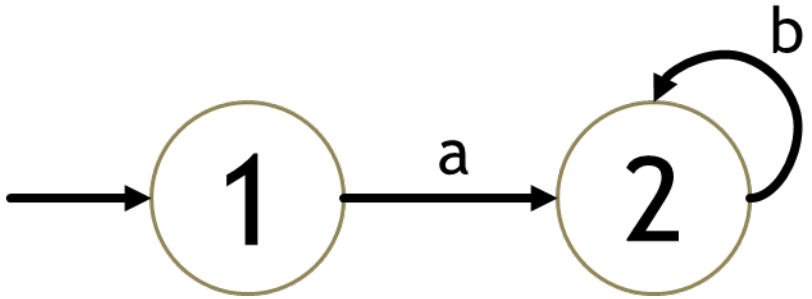
- Trace (string) - a sequence of events allowable by the system's behavior

$$s = e_1 e_2 \dots e_n \ \ where\ e_i \in \Sigma$$

- language – the set of all system traces which originate at the system's initial state

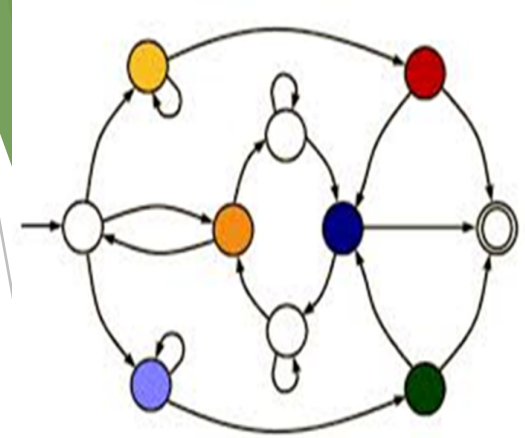
$$L(G) = \{s \in a^* | \delta(x_0, s)\}$$

Example:



Σ= {a,b}
L ={a, ab*}

Natural Projection



- Our purpose is to diagnose unobservable faults from the observable behavior of the system.
- The system's observable behavior can be described by the natural projection of the system's language to the observable event set of the system. $P : \Sigma^* \longrightarrow \Sigma_0^*$

$$P(\varepsilon) = \varepsilon$$

$$\mathrm{P(e)} = \mathrm{e} \quad \text{if} \quad \mathrm{e} \in \Sigma_o$$

$$\mathrm{P(e)} = \varepsilon \quad \text{if} \quad \mathrm{e} \notin \Sigma_o$$

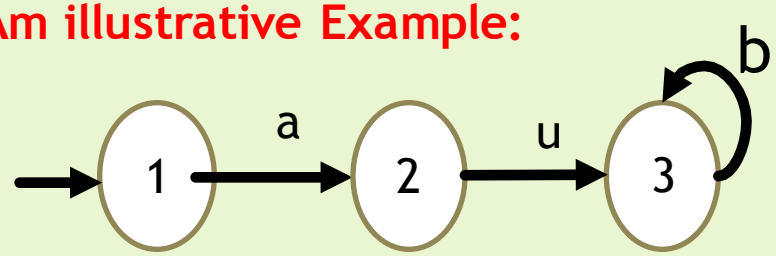
$$P(se) = P(s)P(e) \quad \text{for} \quad s \in \Sigma^* \quad \text{and} \quad e \in \Sigma$$

Extension of the natural projection to the languages:

$$P(L) = \{P(s) \mid s \in L\}$$

Inverse of natural projection $P_L^{-1}(\mathrm{w}) = \{\mathrm{s} \in \mathrm{L} \mid \mathrm{P(s)} = \mathrm{w}\}$

Am illustrative Example:



Σ= {a,b,u}

Σ_o= {a,b}, Σ_u={u}

L ={a, au,aub*}

P(L)= {a,ab*}

P_L^{-1}(a)= {a,au}

Diagnosis within DES Framework

Plant

$G = (X, \Sigma, \delta, x_o)$

$L(G)$

Natural Projection

$P : \Sigma^* \rightarrow \Sigma_0^*$

$P(L(G))$

Diagnoser

- Detected failures
- Type of detected failures
- Location of the system at detection time

How the diagnoser works?

- The diagnoser provides fault diagnostics by extracting information from the original system's observable behaviors, in order to estimate the original system's current state and current condition (faulty or non-faulty).
- The diagnoser's state transition rule is only defined over the original system's observable events.
- Upon observance of the original system's behavior, the diagnoser updates its estimation of the original system's state and condition.

Assumptions

- **Faults are unobservable** $\Sigma_f \subseteq \Sigma_u \subseteq \Sigma$

Otherwise their detection would be trivial.

- **Understudied Faults do not bring the system to the halt mode.**

This gives us enough time to diagnose the fault.

- **No arbitrarily long strings of unobservable events.**

$$\forall suv \in L, s, v \in \sum_o^*, u \in \sum_u^*, \exists n \in N \; such \; that \; \|u\| \leq n$$

This ensures that following the occurrence of an unobservable, sooner or later the system will produce an observable event. This is needed for detection of an unobservable event

- **Live Language: state transition relation is defined for at least one event at all system states**

$$\forall x \in X, \exists e \in \sum such \; that \; \delta(x,e) is \; defined.$$

This is to ensure that in the future the system will always produce a string of observable event to be used for diagnosis.

Capturing different types of faults

Different faults may result in the same failure results.
Example: An open circuit and a stuck closed valve may result in equivalent sensor reading.

We can partition the failure event set into m disjoint subset, each representing a failure type

$$\Sigma_f = \Sigma_{f_1} \dot{\cup} \Sigma_{f_2} \dot{\cup} \cdots \dot{\cup} \Sigma_{f_m} \quad m := \text{failure type}$$

Diagnoser

$$G = (Q, \Sigma, \delta, x_o)$$

$$G_d = (Q_d, \Sigma_d, \delta_d, q_o)$$

$\Sigma_d = \Sigma_o$ Event set soley consisting of observable events

$q_o = 2^{x_0 \times \Delta}$ Initial diagnoser state

$Q_d = \{(x_1, l_1), (x_2, l_2), \ldots, (x_n, l_n)\}, x_i \in X_o, l_i \in \Delta,$ Diagnoser state space

$$\delta_d(q, e) = \bigcup_{\substack{(x,l) \in q \\ t \in P_L^{-1}(e)}} \{(\delta(x, e), \mathrm{LP}((x, l), t))\}$$

$$\Delta = \{N\} \cup 2^{\Delta_f}, \Delta_f = \{F_1, F_2, \ldots F_m\}$$

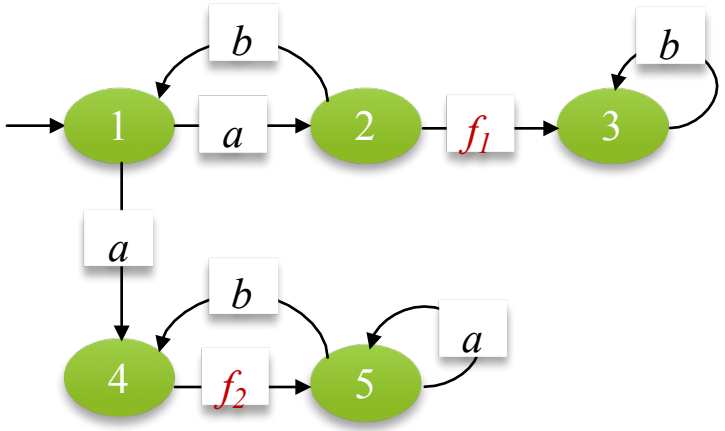
$$l_i = \begin{cases} N & \text{if } x_i \text{ is normal} \\ \{F_{i_1}, F_{i_2}, \ldots, F_{i_k}\} & \text{if } x_i \text{ has reached by failures of type } F_{i_1}, F_{i_2}, \ldots, F_{i_k} \end{cases}$$

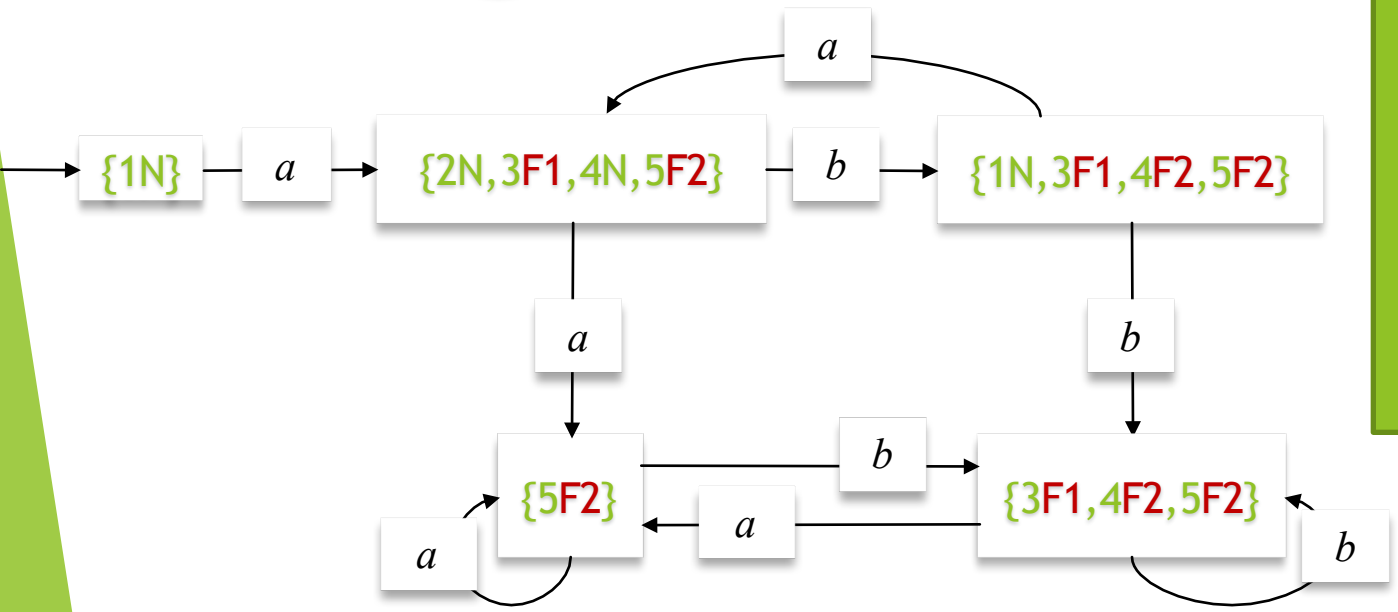
Label propagation mechanism:

$$\mathrm{LP}((x, l), t) = \begin{cases} N & \text{if } l = \{N\} \text{ and } \sum_{F_i} \notin t \text{ for all } i = 1, \ldots, m \\ l \cup \{F_i\} & \text{if } F \notin l \text{ and } \sum_{F_i} \in t \end{cases}$$

Question: How to construct Q_d *and* δ_d?

Constructing the diagnoser





Algorithm

Let $q_0 = \{(x_0, N)\}$

Let $Q_d = q_0$

Repeat

  For $q \in Q_d$ *and* $e \in \sum_o$ *do*

    if $\delta_d(q,e) \neq \varnothing$ and $\delta_d(q,e) \notin Q_d$ then

      $Q_d = Q_d \cup \delta_d(q,e)$

    end if

  end for

Until there is no new state $\delta_d(q,e)$ for all $q \in Q_d$ *and* $e \in \sum_o$

Diagnosability

Question: An important question is that whether the diagnoser can detect and locate the failure?

Diagnosability - a system fault is considered diagnosable if upon its occurrence, all possible consequential system behaviors allow for the definitive diagnosis (detection, isolation, identification) of its occurrence.

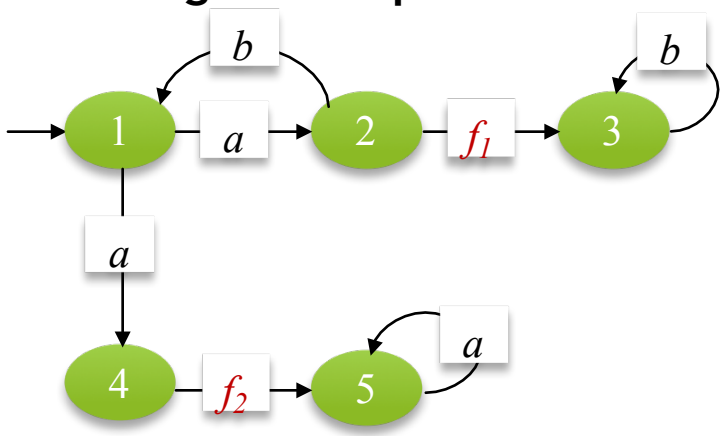
Definition – A system with the live language L is said to be is said to be diagnmosable w.r.t. the natural projection if the following holds:

$$(\forall f_i \in \sum_f)(\exists n_i \in \bullet)[\forall s \in \psi(\sum_{f_i})](\forall t \in L/s))[\|t\| \geq n_i \Rightarrow D]$$

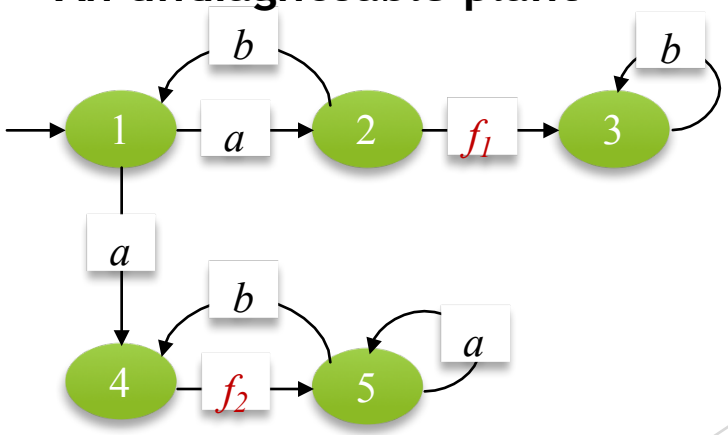
where the dignosability condition D is $D: \forall \omega \in P_L^{-1}(P(s)) \Rightarrow \sum_{f_i} \in s$

$\Psi(\sum_{fi}) = \{s\sigma_f \in L : \sigma_f \in \sum_{fi}\}$, where $\Psi(\sum_{fi})$ denotes the set of all traces of L that end in a failure event σ_f.

A Diagnosable plant



An undiagnosable plant



Question: How to check it based on the structure of the diagnoser?

Checking the diagnosability based the structure of the diagnoser

Definition: Fi – Indeterminate Cycle:

A set of F_i – uncertain states $q_1, q_2, \cdots q_n \in Q_d$ is said to form an F_i – indeterminate cycle if

1) $q_1, q_2, \cdots q_n$ form a cycle in G_d with $\delta_d(q_j, \sigma_j) = x_{j+1}$, $j = 1, \ldots, n-1$, $\delta_d(q_n, \sigma_n) = q_1$, where $\sigma_j \in \sum_o$,

2) $\exists \left(x_j, \ell_j\right), \left(y_j, \tilde{\ell}_j\right) \in q_j, j = 1, \ldots, n$, s.t.

a) $F_i \in \ell_j, F_i \notin \tilde{\ell}_j \quad \forall j$

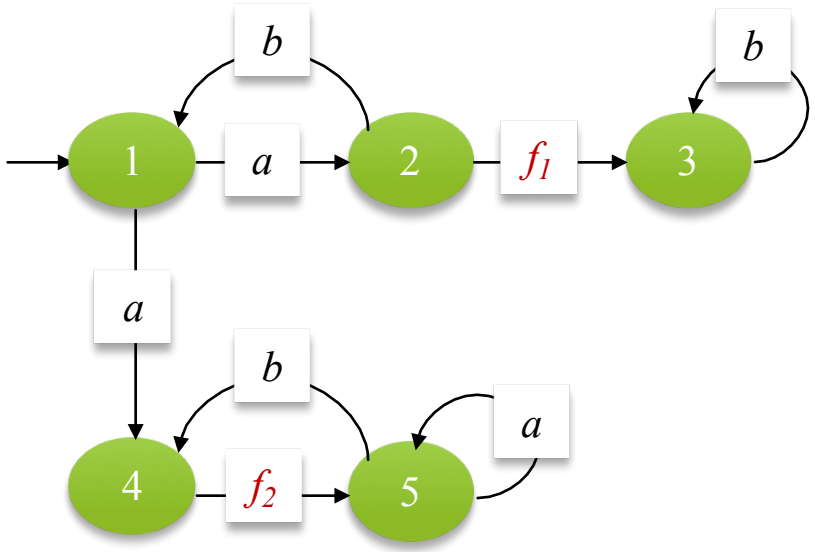
b) The sequences of states $\{x_j\}, j = 1, \ldots, n$, and $\{y_j\}, j = 1, \ldots, n$ form cycles in G' with

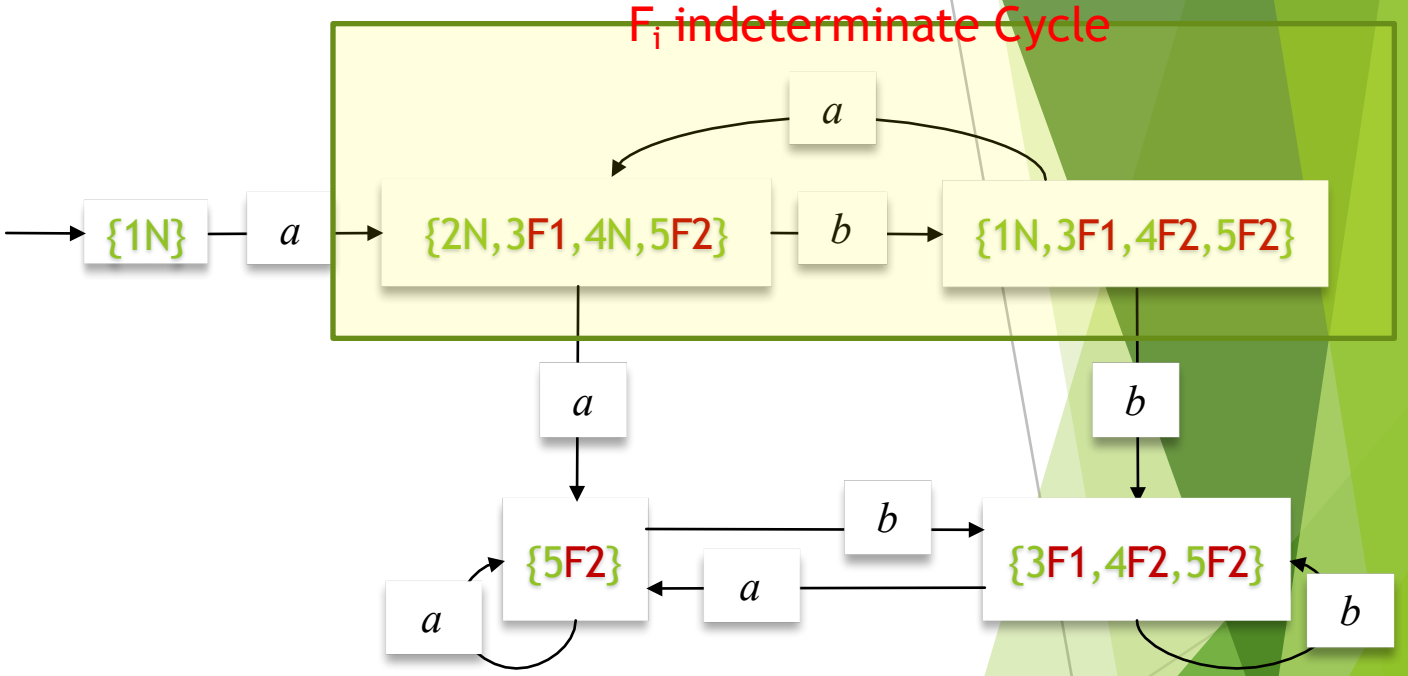
$\left(x_j, \sigma_j, x_{j+1}\right) \in \delta_{G'}, j = 1, \ldots, n-1$ and $\left(x_n, \sigma_n, x_1\right) \in \delta_{G'}$ and

$\left(y_j, \sigma_j, y_{j+1}\right) \in \delta_{G'}, j = 1, \ldots, n-1$ and $\left(y_n, \sigma_n, y_1\right) \in \delta_{G'}$

Theorem: A language L without multiple failures of the same type is diagnosable, if and only if its diagnoser G_d has no F_i indeterminate cycle, for all failure types F_i, i=1,…,m

Example





Faulty loop: $1 \xrightarrow{a} 4 \xrightarrow{f_2} 5 \xrightarrow{b} 4 \xrightarrow{a} 5$ $\quad s_1 = af_2ba$

Normal loop: $1 \xrightarrow{a} 2 \xrightarrow{b} 1 \xrightarrow{a} 2$ $\quad s_2 = aba$

$$\Rightarrow P(s_1) = P(s_2) = aba$$

Gaps

In the existing methods (including the method presented here), it is required to initialize and run the diagnoser synchronously with the plant. This allows the diagnoser to diagnose failures based on a rich set of information including both pre- and post-failure behaviours in the system.

Challenges:

1- In many practical situations, only after a fault occurs, the diagnosis tool (e.g. a portable industrial computer) has to be brought and connected to the faulty plant to diagnose the occurred fault. In many cases, it is not possible, or it is time-consuming and costly, to restart the plant to be synchronized with the diagnoser.

2- Even if the plant and the diagnoser are initialized simultaneously, it is possible that for any particular reason, the diagnoser misses an observation, and thereafter, it cannot track the plant, in turn requiring both the plant and the diagnoser to be restarted again.

Future work

- Asynchronous Fault Diagnosis: To develop a systematic and analytical approach to construc diagnoser that can be asynchronously turned on at anytime, even after the occurrence of a fault.

Problem formulation:

Problem : In a discrete event system G with the generated string s, for any successive string $t \in \mathcal{L}_{G/s}$, where t occurs upon diagnoser activation, from the observation $P(t)$, determine if $\exists f \in \Sigma_f$ such that $f \in s.t$. If yes, identify the type of fault, Σ_{f_i}, where $f \in \Sigma_{f_i}$, and locate the fault by finding the system state $x \in X$ subsequently reached upon fault occurrence.

Challenges:

1- Unlike conventional diagnosis techniques, the past history of the system before the activation of the diagnoser is not available, leaving the diagnoser with the challenge of diagnosing faults using only the future behaviors of the plant, observed after the activation of the diagnoser.

2-In contrast to existing methods, where the initial state of the system and correspondingly the initial state of the diagnoser are generally assumed to be non-faulty, upon its initialization, the asynchronous diagnoser is no longer able to assume that the current state of the system is normal.

Acknowledgement

- My colleagues at ACCESS LAB and TECHLAV Center
- Financial support from TECHLAV project


QUESTIONS?
COMMENTS?
CONCERNS?