North Carolina Agricultural and Technical State University

ACCESS Laboratory

DoD Center of Excellence in Autonomy

TECHLAV

Testing, Evaluation, and Control of Heterogeneous Large-scale Systems of Autonomous Vehicles

EVENT-BASED FAULT DIAGNOSIS FOR PARTIALLY-KNOWN DISCRETE EVENT SYSTEMS

Ira “Wendell” Bates, II

Ph.D. Candidate

Department: Electrical Engineering

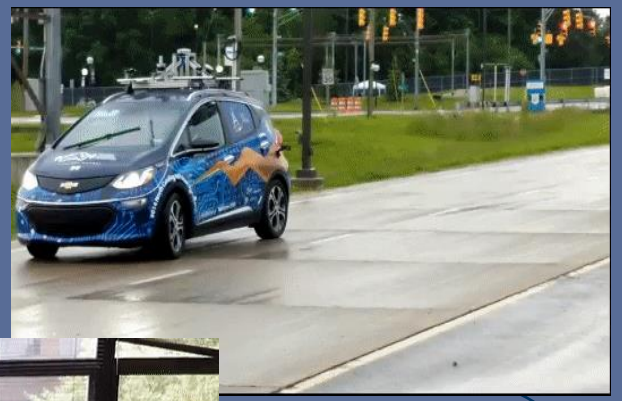
Advisor: Dr. Ali Karimoddini

AGGIES DO

TECHLAV Seminar - 11/13/2020

Systems

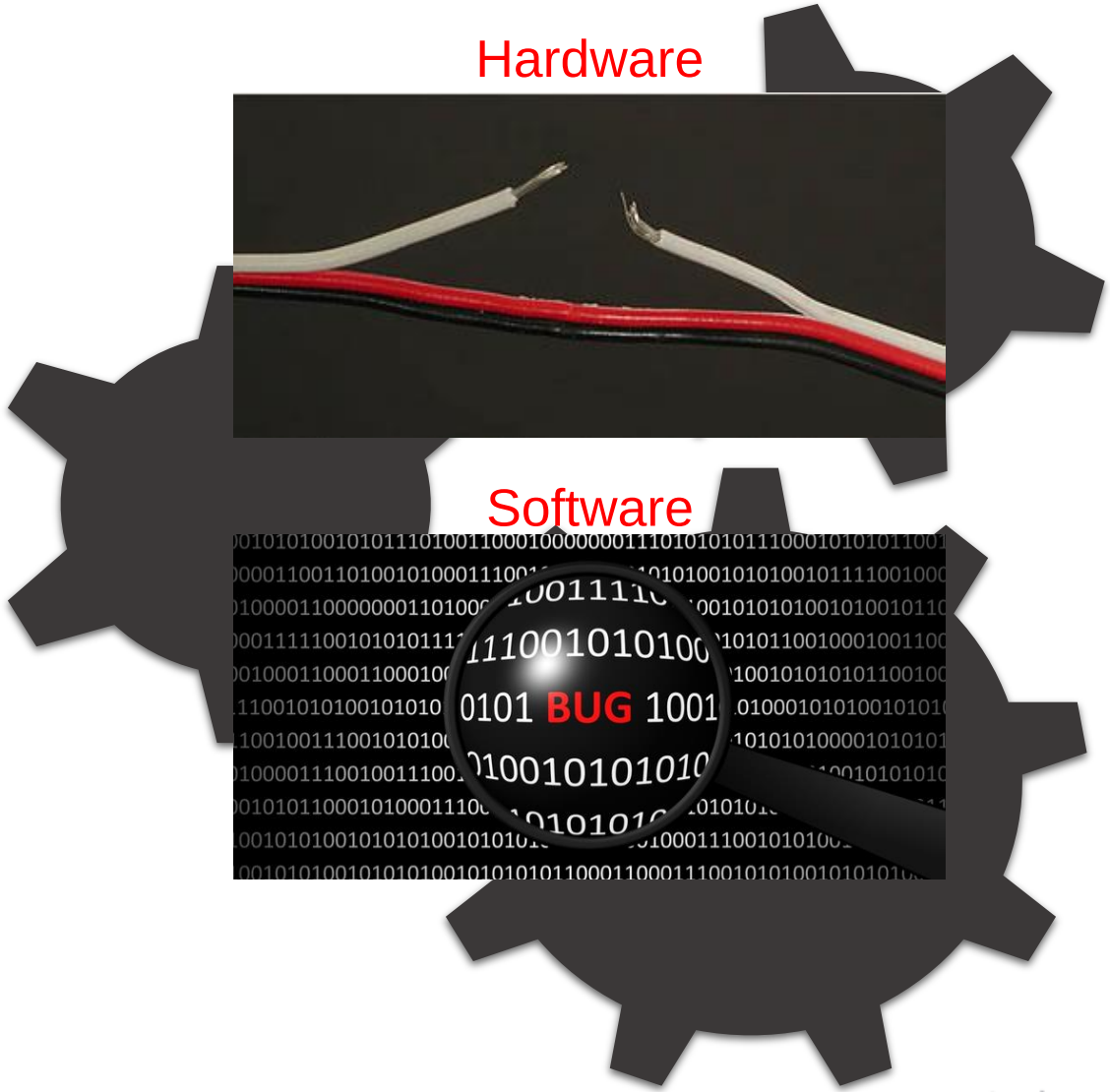
Different types of systems:



What is a Fault?

A ***fault*** can be defined as:

- A deviation of a system from its specified acceptable operation
- The termination of an item to perform a required function
- An abnormal condition that can cause an element of a system (or item) to fail



Why Do We Need Fault Diagnosis?

- Efforts have been increased to **improve reliability**
- Faults are **unpredictable** and often times things can go wrong
- Not **feasible** to place sensors on sensors

What We Have Developed for Fault Diagnosis

1. **I. Wendell Bates**, A. Karimoddini, and M. Karimadini, "*Learning a Partially-Known Discrete Event System*," IEEE Access, 8:61806–61816, 2020.
2. M. Karimi, A. Karimoddini, A. White, **W. Bates**, "*Event-Based Fault Diagnosis for an Unknown Plant*," Proc. of 55th IEEE Conference on Decision and Control.
3. **W. Bates**, A. Karimoddini, M. Karimadini, "*A Learning-based Approach for Diagnosis and Diagnosability of Initially Unknown Discrete Event Systems*", IEEE Systems, Man, and Cybernetics (under review)
4. White, Alejandro, and Ali Karimoddini. "Semi-asynchronous fault diagnosis of discrete event systems." *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2016.
5. Karimi, M. M., Karimoddini, A., White, A. P., & **Bates, I. W**. (2016, December). Event-based fault diagnosis for an unknown plant. In *2016 IEEE 55th Conference on Decision and Control (CDC)* (pp. 7216-7221). IEEE.
6. White, Alejandro, and Ali Karimoddini. "Asynchronous fault diagnosis of discrete event systems." *2017 American Control Conference (ACC)*. IEEE, 2017.
7. White, Alejandro, and Ali Karimoddini. "Event-based diagnosis of flight maneuvers of a fixed-wing aircraft." *Reliability Engineering & System Safety* 193 (2020): 106609.
8. White, Alejandro, Ali Karimoddini, and Rong Su. "Fault diagnosis of discrete event systems under unknown initial conditions." *IEEE Transactions on Automatic Control* 64.12 (2019): 5246-5252.

Boeing 737-MAX

Boeing 737 MAX

Lion Air Flight 610
October 2018
189 casualties

Ethiopian Airlines Flight 302
March 10, 2019
157 casualties

346 total casualties

Cruz, Bruno Silveira, and Murillo de Oliveira Dias. "CRASHED BOEING 737-MAX: FATALITIES OR MALPRACTICE?." *GSJ* 8.1 (2020): 2615-2624.

Outline

- Modeling of the Boeing 737 Takeoff Procedure
- Fault Diagnosis of Boeing Partially-Known System
  - » Capturing the Known Information
  - » Learning the Unknown Information
- Simulation and Properties
- Conclusion
- Questions

Modeling Boeing 737-MAX Takeoff Procedure

Boeing 737

History:

- First introduced to public in 1968
- Developed to supplement the Boeing 727 on short and thin routes
- Most ordered plane in commercial history in 1987
- Latest generation 737-MAX (v. 7/8/9/10) entered service in 2017
  - Powered by improved CFM LEAP high bypass turbofans
  - Able to accommodate 138-204 people

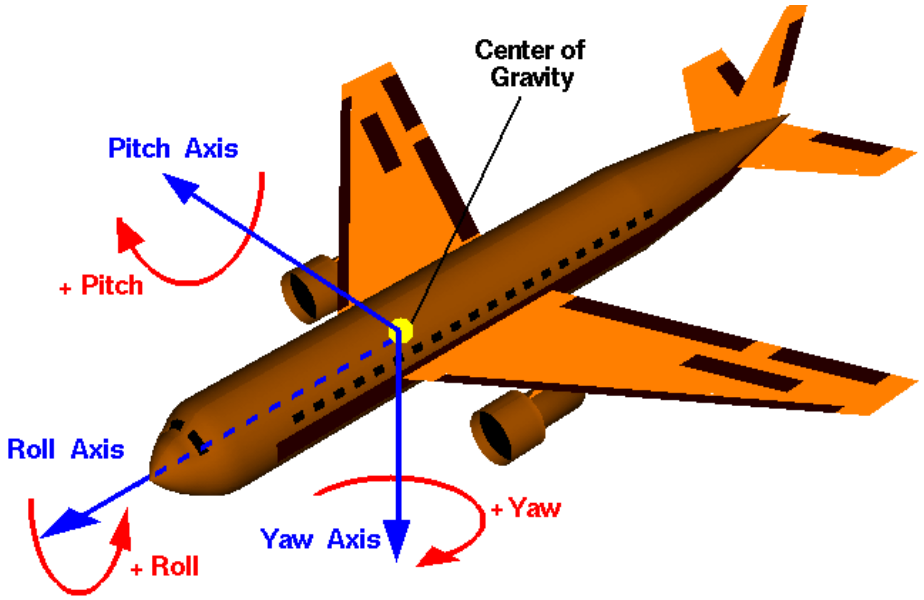
Boeing 737 MAX 8

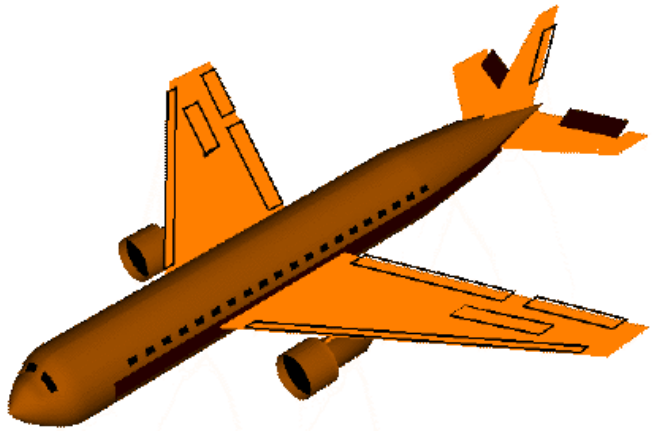
Specs:

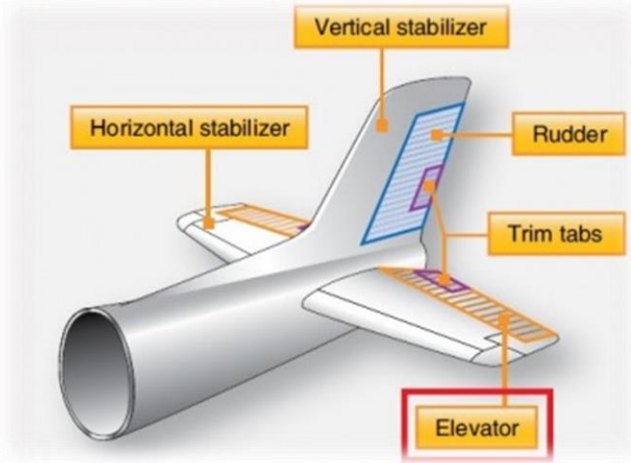
First put into service on May 22, 2017

Seats (2-class)	162-178
Maximum Seats	210
Range (km)	6570
Length	39.52m
Wingspan	35.9m
Engine	LEAP-1B from CFM International

Airplane Dynamics







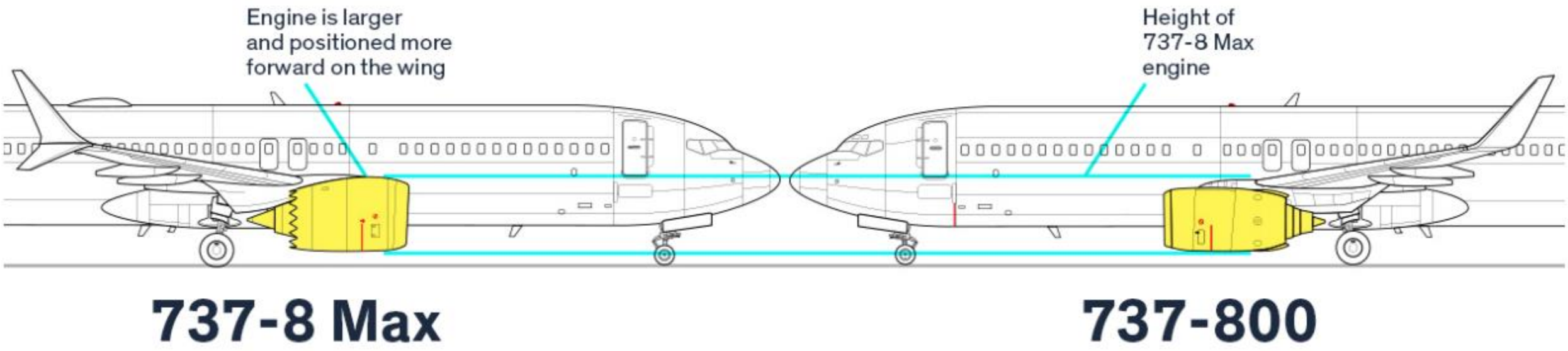
- There are three imaginary lines that pass through the center of gravity of an aircraft
  - Roll, Pitch, and Yaw

- Pitch is the motion of the aircraft along its lateral axis and is controlled by “elevators”

Competition

- Manufacturing companies continuously face constant pressure to design and build cheaper airplanes in order to remain competitive; with focus mainly on improving fuel efficiency.
- In 2010, Airbus surprised the aviation community with the announcement of plans to build a more fuel-efficient aircraft called the “A320neo” claiming to burn 16% less fuel
- Airbus’ plans were to have the A320neo available for full commercial operation by 2016.

Boeing 737



- In an aggressive effort to remain competitive 2014, the CEO of Boeing announced a new larger engine design.
- The new engine didn't properly clear the ground and was moved up on the wing.

1. *Carvalho, Stanley (November 5, 2014).* *"Boeing plans to develop new airplane to replace 737 MAX by 2030"*. *Reuters*.

Boeing 737 MAX

During initial flight tests pilots complained about the high pitch rate tendency during takeoff on the Boeing MAX .

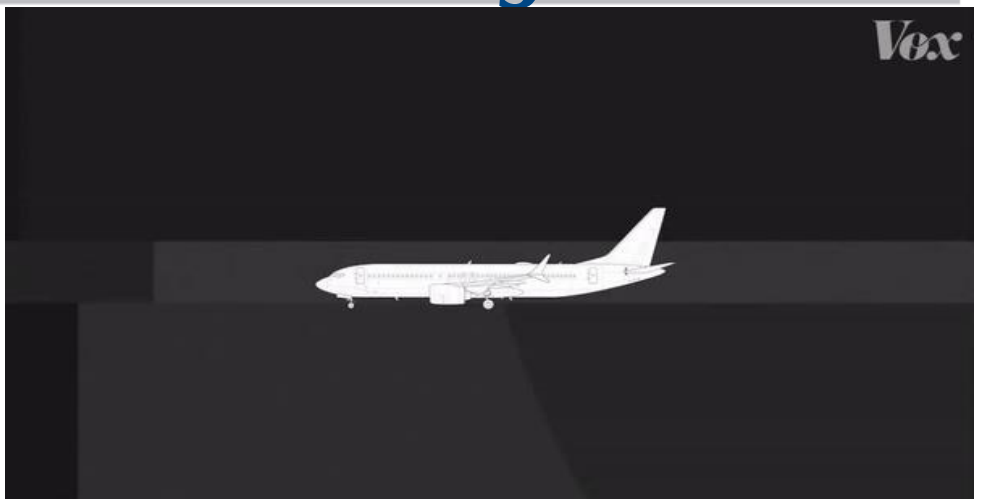
Around 20 percent of aviation fatalities occur during taxiing, takeoff, and initial climb.

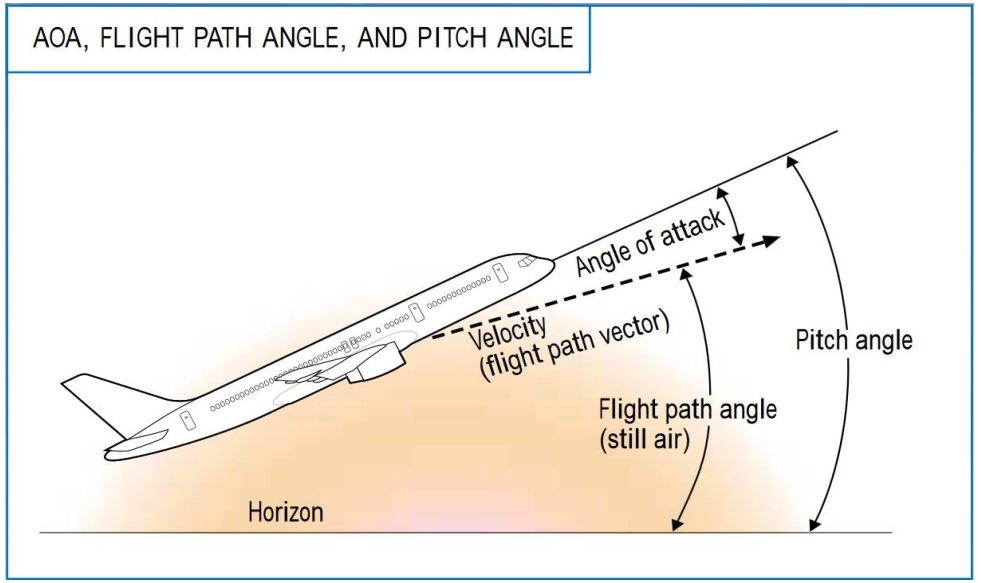
Johnston, Phillip, and Rozi Harris. "The Boeing 737 MAX saga: lessons for software organizations." *Software Quality Professional* 21.3 (2019): 4-12.

Airplanes, Boeing Commercial. "Statistical summary of commercial jet airplane accidents: Worldwide operations 1959–2015." *Aviation Safety, Boeing Commercial Airplanes, Seattle, WA. http://www. boeing. com/resources/boeingdotcom/company/about_bca/pdf/statsum. pdf* (2016).

Boeing 737

- High pitch rate causes an Angle of Attack (AoA) to be too high
- AoA is the angle between the oncoming air and a reference line on the airplane wing
- Too high of an AoA causes the airflow across the upper surface of the wing to becoming detached resulting in loss of lift (aka “stall”)





https://www.faa.gov/other_visit/aviation_industry/airline_operators/training/media/takeoff_safety.pdf

Boeing 737 MAX

In order to combat this occurrence, Boeing developed the **M**aneuvering **C**haracteristics **A**ugmentation **S**ystem (**MCAS**)

The responsibility of the MCAS was to automatically push the nose of the plane down to aid the pilot in case an irregularly high AoA was detected.

https://www.boeing.com/commercial/737max/737-max-software-updates.page

Boeing 737 MAX

Reports from investigations:

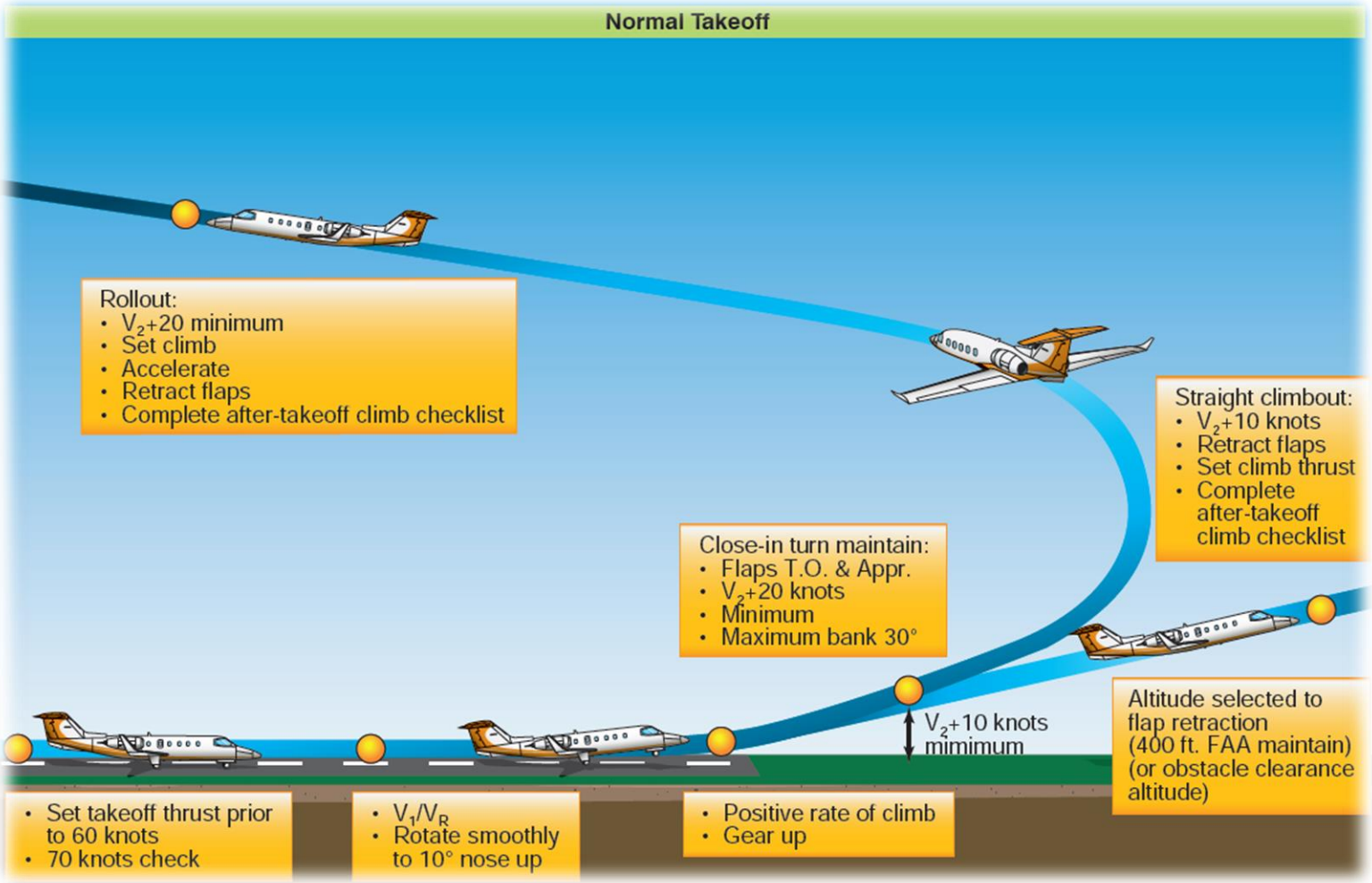
- The MCAS depended on a single AoA sensor instead of the installed two, creating a single point failure
  - Readings from black box data shows AoA sensor readings differing by 20 degrees

- The existence of the MCAS was “hidden”
  - The software and its functions were not disclosed to the pilots to avoid increased pilot training requirements
  - AoA disagreement alerts were inoperable despite having been certified as a standard cockpit feature
  - MCAS was removed from the Flight Crew Operations Manual

- Due to a significant increase in maximum movement for the plane’s horizontal stabilizer, the MCAS required more manual power from the pilots to overtake control in case of errant sensor reading.

1. Johnston, Phillip, and Rozi Harris. "The Boeing 737 MAX saga: lessons for software organizations." *Software Quality Professional* 21.3 (2019): 4-12.
2. https://www.faa.gov/news/updates/?newsId=93206

The Different Modes for Takeoff of an Aircraft

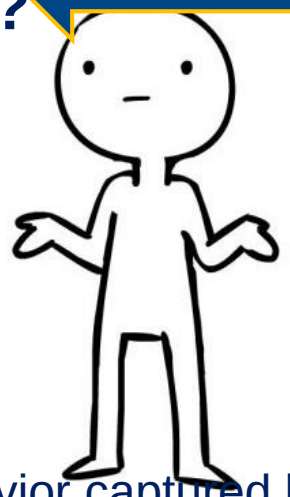


Why Discrete Event Systems?

Real world system

Modeling Relation

hy?



Model

Structure for generating behavior representative of the real world

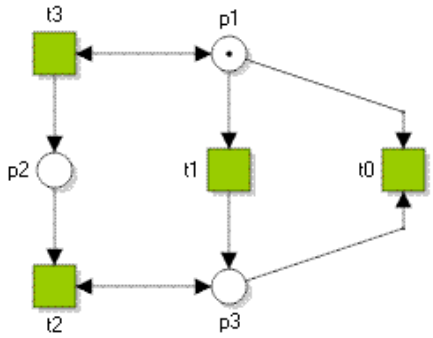
- DESs
  - Model high-level behavior captured by a sequence of events
  - Represent logical system behavior in a more useful way
- Faults can be considered as abrupt changes (events) in the system

What is an event?

Events may be identified as:

- A specific action taken (e.g. a button press)
- Any observed spontaneous behavior (e.g. an assembly line shutting down for no apparent reason)
- The result of several conditions being suddenly met (e.g. data overflow to a sensor).
- A system exceeding the maximum or minimum threshold of a continuous signal

Types of DES Models



Petri-Nets

A. Giua and M. Silva, "*Petri nets and automatic control: A historical perspective*," Annual Reviews in Control, vol. 45, pp. 223–239, 2018.

Ya Shi, Cong Tian, Zhenhua Duan, and Mengchu Zhou. Model checking petri nets with msvl.
Information Sciences, 363:274–291, 2016.

$$b \xrightarrow{b} \surd \qquad \left(\frac{}{v \xrightarrow{v} \surd},\ v := b\right)$$

$$a + b \xrightarrow{b} \surd \qquad \left(\frac{y \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd},\ v := b,\ x := a,\ y := b\right)$$

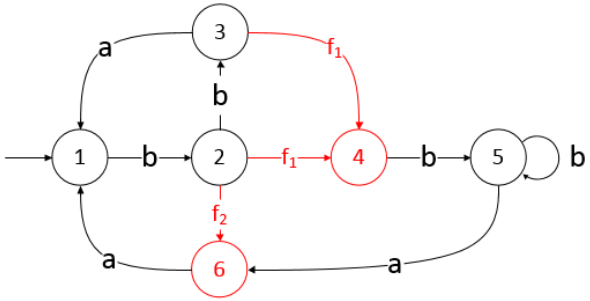
$$(a + b)\cdot c \xrightarrow{b} c \qquad \left(\frac{x \xrightarrow{v} \surd}{x\cdot y \xrightarrow{v} y},\ v := b,\ x := a + b,\ y := c\right)$$

$$((a + b)\cdot c)\cdot d \xrightarrow{b} c\cdot d \qquad \left(\frac{x \xrightarrow{v} x'}{x\cdot y \xrightarrow{v} x'\cdot y},\ v := b,\ x := (a + b)\cdot c,\ x' := c,\ y := d\right)$$

Process Algebra

Fokkink, Wan. *Introduction to process algebra*. Springer Science & Business Media, 2013.

Jos CM Baeten. A brief history of process algebra. Theoretical Computer Science, 335(2-3):131–146, 2005.



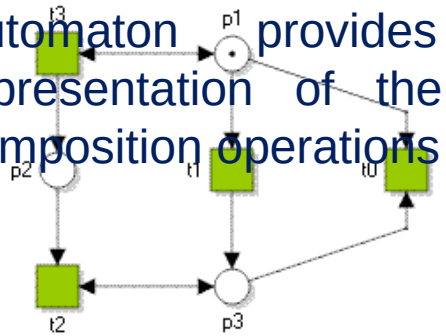
Automata

C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Science & Business Media, 2009.

Jacques Sakarovitch, Reuben Thomas, and Reuben Thomas. Elements of automata theory, volume 6. Cambridge University Press Cambridge, 2009.

Types of DES Models

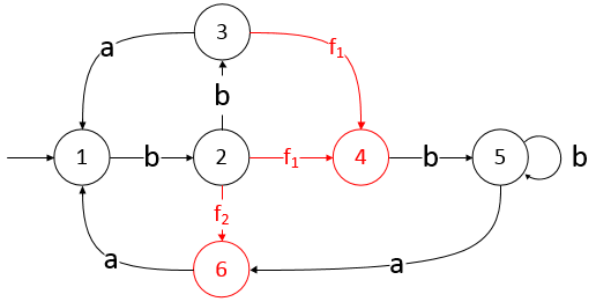
Automaton provides an intuitive and visual representation of the system that is amenable to composition operations as well as analysis.



$$\frac{}{b \xrightarrow{b} \surd} \qquad \frac{}{(\frac{}{v \xrightarrow{v} \surd},\ v := b)}$$

$$\frac{}{a + b \xrightarrow{b} \surd} \qquad \frac{}{(\frac{}{x + y \xrightarrow{v} \surd},\ v := b,\ x := a,\ y := b)}$$

$$\frac{(a + b)\cdot c \xrightarrow{b} c}{((a + b)\cdot c)\cdot d \xrightarrow{b} c\cdot d} \qquad \frac{(\frac{x \xrightarrow{v} \surd}{x\cdot y \xrightarrow{v} y},\ v := b,\ x := a + b,\ y := c)}{(\frac{x \xrightarrow{v} x'}{x\cdot y \xrightarrow{v} x'\cdot y},\ v := b,\ x := (a + b)\cdot c,\ x' := c,\ y := d)}$$



Petri-Nets

A. Giua and M. Silva, "*Petri nets and automatic control: A historical perspective*," Annual Reviews in Control, vol. 45, pp. 223–239, 2018.

Ya Shi, Cong Tian, Zhenhua Duan, and Mengchu Zhou. Model checking petri nets with msvl.
Information Sciences, 363:274–291, 2016.

Process Algebra

Fokkink, Wan. *Introduction to process algebra*. Springer Science & Business Media, 2013.

Jos CM Baeten. A brief history of process algebra. Theoretical Computer Science, 335(2-3):131–146, 2005.

Automata

C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Science & Business Media, 2009.

Jacques Sakarovitch, Reuben Thomas, and Reuben Thomas. Elements of automata theory, volume 6. Cambridge University Press Cambridge, 2009.

Automaton Structure

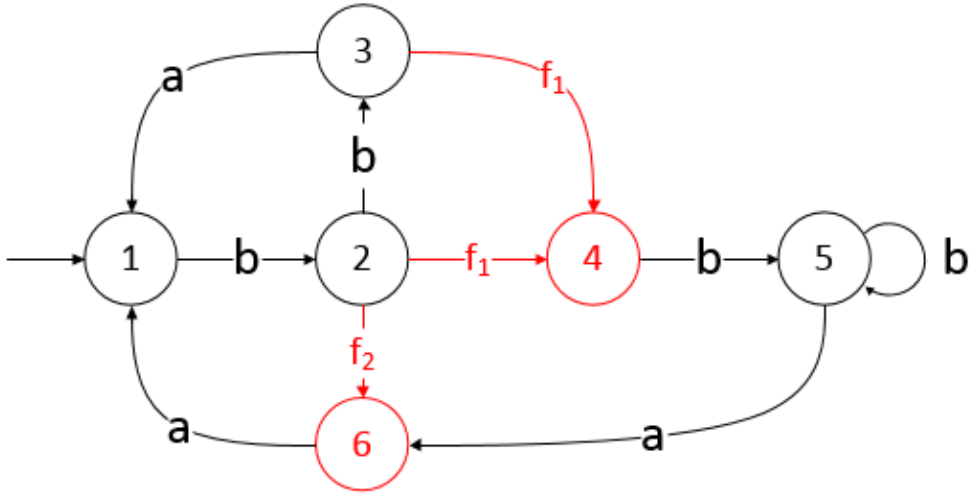
$A\ non-deterministice\ DES\ can\ be\ represented\ by\ a\ tuple$

$\boldsymbol{G} = (X, \Sigma, \delta, x_0)\ \ where$

- $\boldsymbol{X}$ – the state space: $\{1, 2, 3, 4, 5, 6\}$
- Σ – $(\Sigma_o \cup \Sigma_{uo})$ the event set: $\{a, b, f_1, f_2\}$

  Σ_o – $observable\ events\ (e.g.\ sensor\ data\ transfer)$: $\{a, b\}$
  Σ_{uo} – $unobservable\ events\ (e.g.\ sensor\ damage)$: $\{f_1, f_2\}$

- $\boldsymbol{\delta}$ – $the\ transition\ relation$
  - relation ($\delta: X \times \Sigma \rightarrow 2^X$): a partial relation that determines all feasible system state transitions caused by system events (2^X is the set of all possible combinations of states)
- $\boldsymbol{x_0}$ – $the\ initial\ state$: $\{1\}$



Language

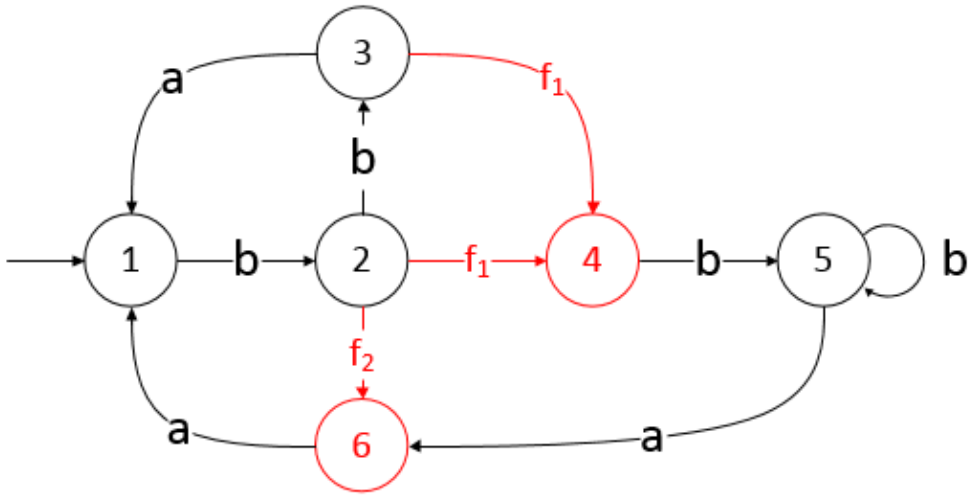
Strings - a sequence of one or more events, allowable by the system's behavior

e.g., $s = \sigma_1\sigma_2 \ldots \sigma_n$ where $\sigma_i \in \Sigma$

Language ($\mathcal{L}(G)$) - the set of all system ***strings*** which originate at the system's initial state x_0

$\mathcal{L}(G) = \{s \in \Sigma^* | \delta(x_0, s) \text{ is defined}\}$

(Σ^* is the Kleene Closure of Σ)

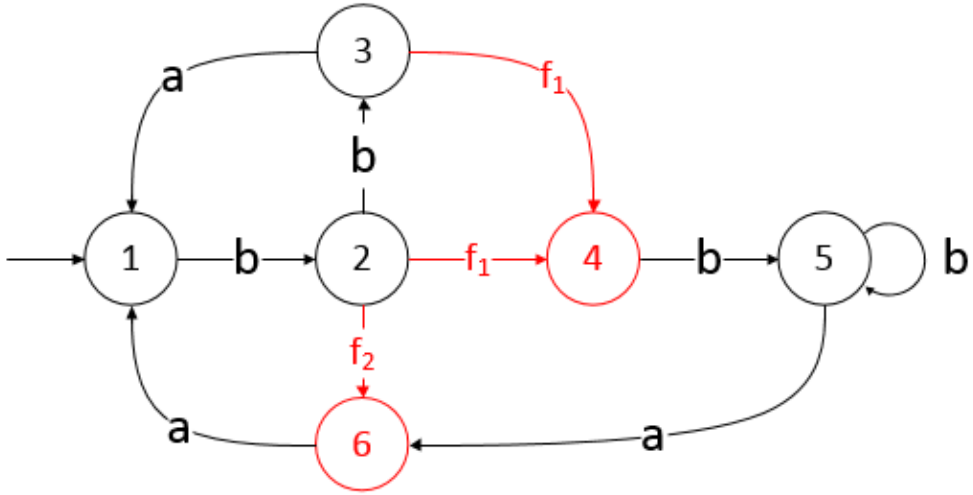


Natural Projection

In order to represent the visibility of the behavior of the system, the natural projection is used

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

$$P(\varepsilon) = \varepsilon$$
$$P(\sigma) = \sigma \qquad \text{if } \sigma \in \Sigma_0$$
$$P(\sigma) = \varepsilon \qquad \text{if } \sigma \notin \Sigma_o$$
$$P(s\sigma) = P(s)P(\sigma) \qquad \text{for } s \in \Sigma^* \text{ and } \sigma \in \Sigma$$

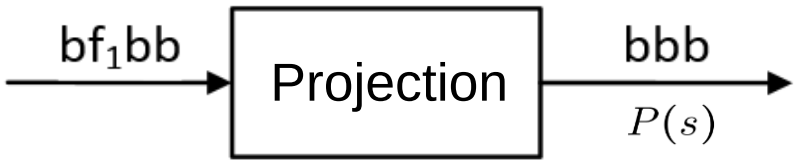


- Extension of the natural projection to the languages:

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

- Inverse of Natural Projection

$$P^{-1}_{\mathcal{L}(G)}(w) = \{s \in \mathcal{L}(G) \mid P(s) = w\}$$



Model Construction

Assumptions

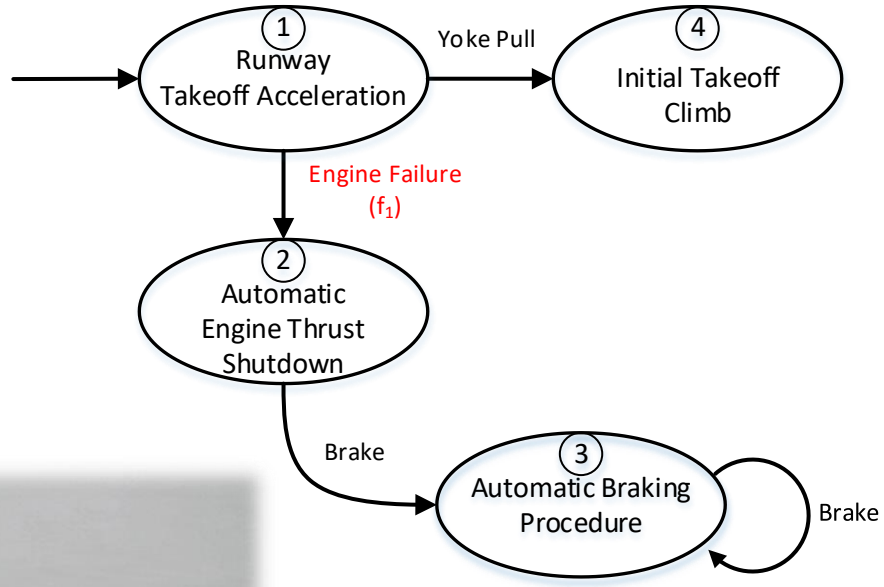
- Starting from the beginning of the runway.
- All standard pre-flight checks have been approved (i.e. no error has occurred before the start)
- Air Traffic Controller (ATC) has approved takeoff clearance.

https://www.faa.gov/regulations_policies/handbooks_manuals/

Model Construction

Runway Takeoff

- V_1 – the speed where take-off should no longer be aborted
  - if an engine fails before V_1 is reached, the pilot will abort the takeoff and the aircraft's system will immediately cancel thrust and apply the brakes
- V_R – the speed at which the pilot begins to apply control inputs (control stick "yoke" pull) to cause the aircraft nose to pitch up; after which the aircraft will leave the ground
- V_2 – the speed at which the aircraft may safely climb with one engine inoperative

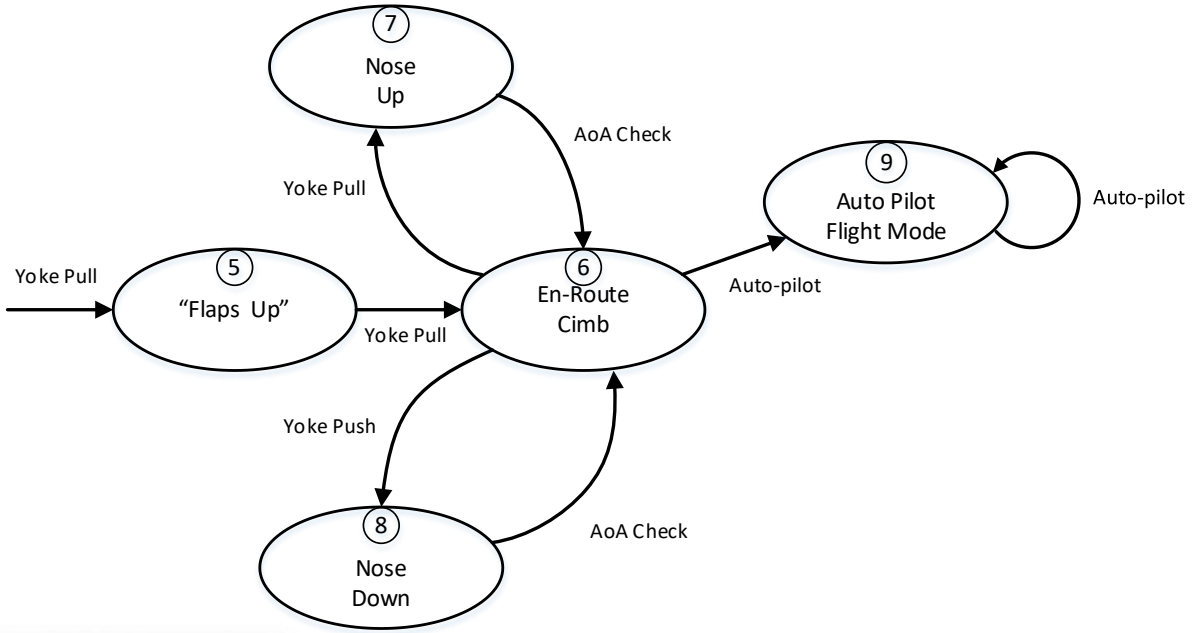


https://www.faa.gov/regulations_policies/handbooks_manuals/

Model Construction

En-Route Climb

- The pilot must maintain proper pitch and AoA for a successful climb through, making pitch adjustments as needed in order to maintain the desired climb velocity
- Upon stabilized climb, landing gear is retracted and shortly thereafter, wing flaps are retracted to achieve optimum lift and drag configuration
- The pilot should be aware of and anticipate the sudden changes in AoA upon gear and flap retraction to avoid stall
- Continuous pitch adjustments should be made until cruising altitude is reached and the auto-pilot mode can be set



https://www.faa.gov/regulations_policies/handbooks_manuals/

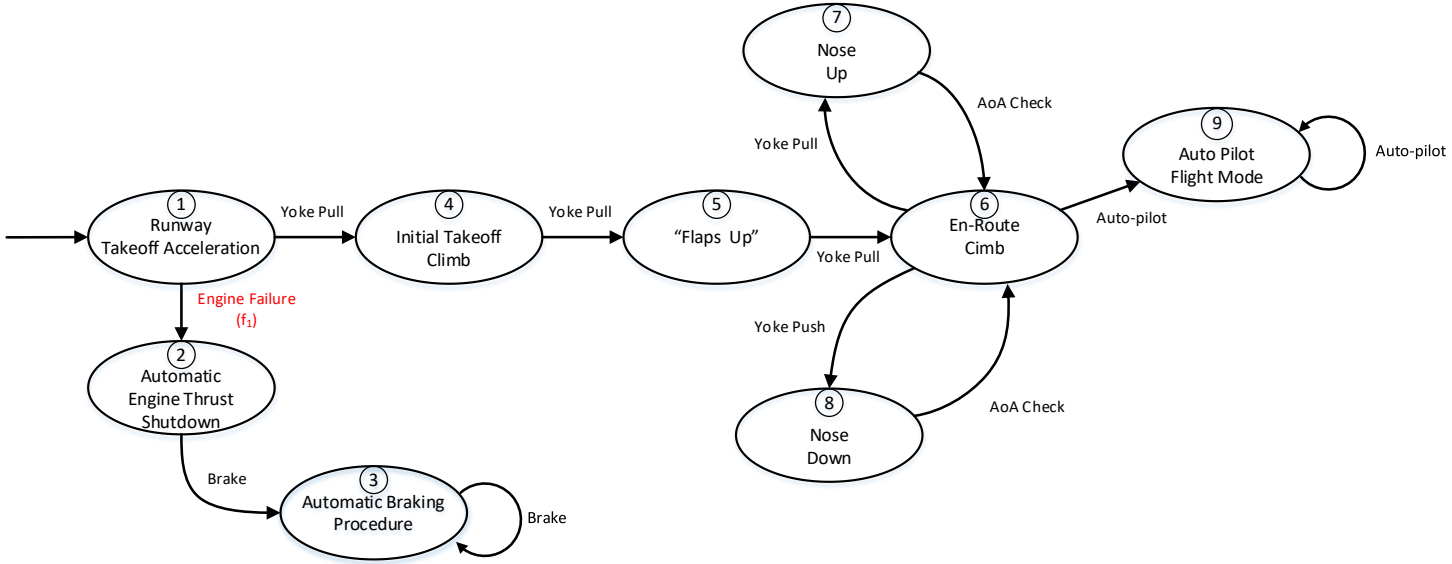
Model Construction

States

State	Representation	State	Representation
1	Runway Takeoff Acceleration	6	En-Route Climb
2	Automatic Engine Thrust Down	7	Nose Up
3	Automatic Braking Procedure	8	Nose Down
4	Initial Takeoff Climb	9	Auto-pilot Mode
5	"Flaps Up"		

Events

Event	Representation	Event	Representation
a	Yoke Pull	d	AoA Check
b	Yoke Push	e	Auto-Pilot
c	Brake	f_1	Engine Failure

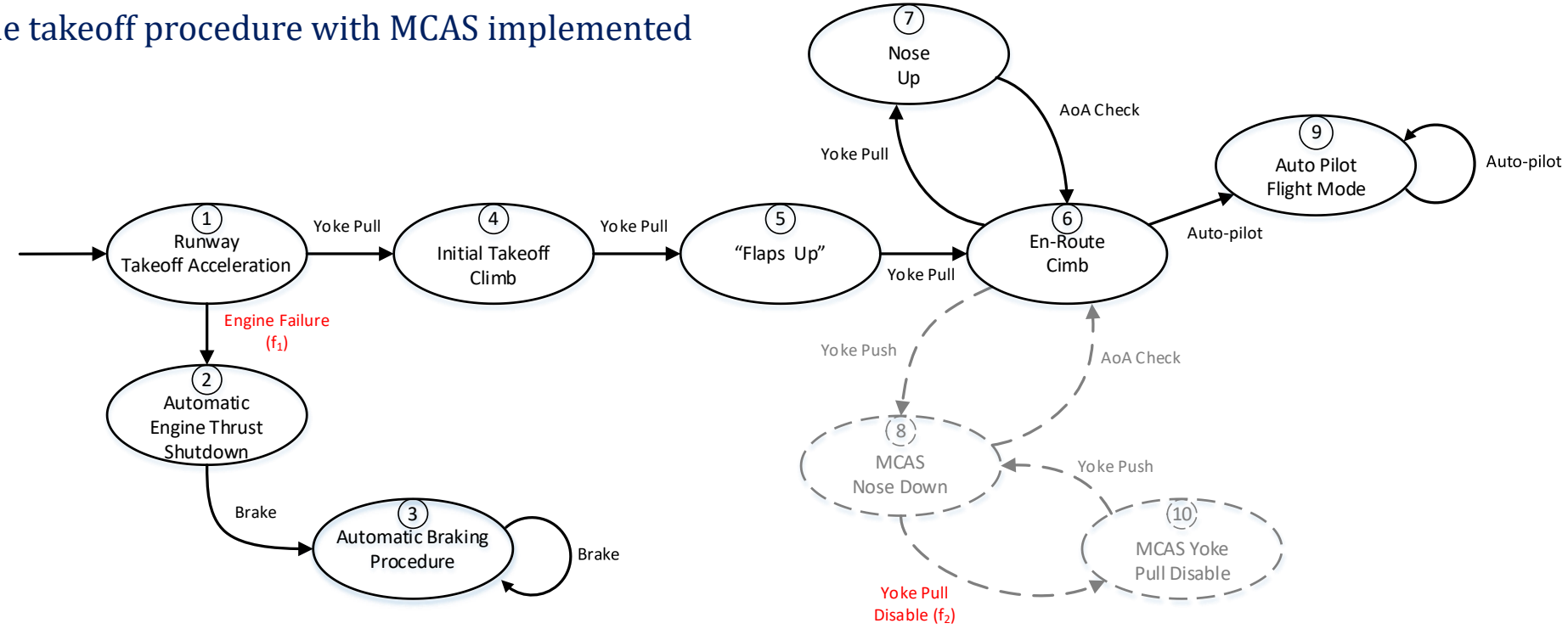


A model of the takeoff procedure under normal conditions

https://www.faa.gov/regulations_policies/handbooks_manuals/

Model Construction

A model of the takeoff procedure with MCAS implemented



MCAS

- MCAS was set to activate in the background after the airplane's flap had been retracted
- Upon activation, the "yoke pull" would be deactivated in order to avoid manual interference with the MCAS pitch correction
- Due to faulty software implementation and incorrect AoA sensor data, the nose of the plane was immediately pushed down

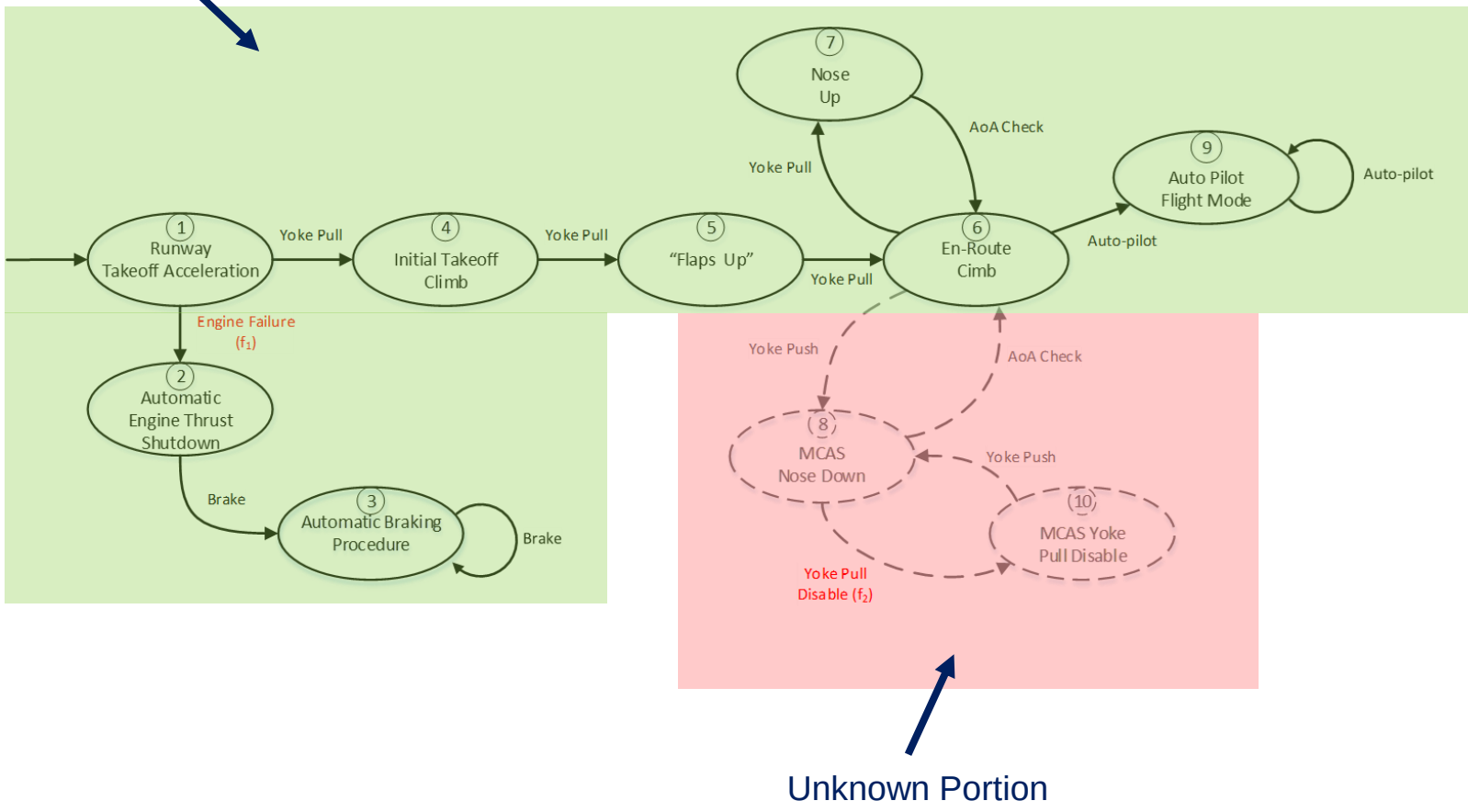
https://www.faa.gov/regulations_policies/handbooks_manuals/

Fault Diagnosis for Boeing 737-MAX

Model Construction

A model of the takeoff procedure with MCAS implemented

- There are different types of faults in the system
- Fault events are unobservable, otherwise their diagnosis is trivial
- Part of the system is known
- Part of the system is unknown as it is added later to the system with the intention making the system less complex

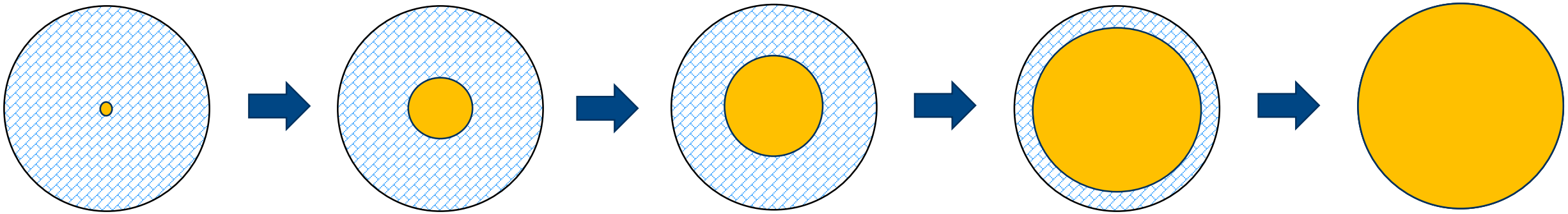


Challenge

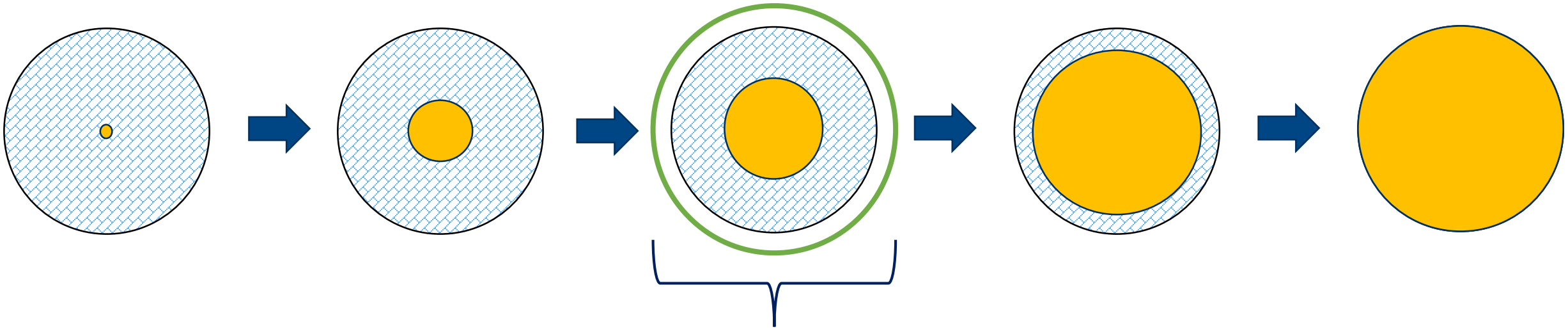
Strategy 1 – Ignoring the available information and treating the system as a completely unknown system

Start from the beginning and learn the system with a finite number of iterations while assuming no prior knowledge of the system.

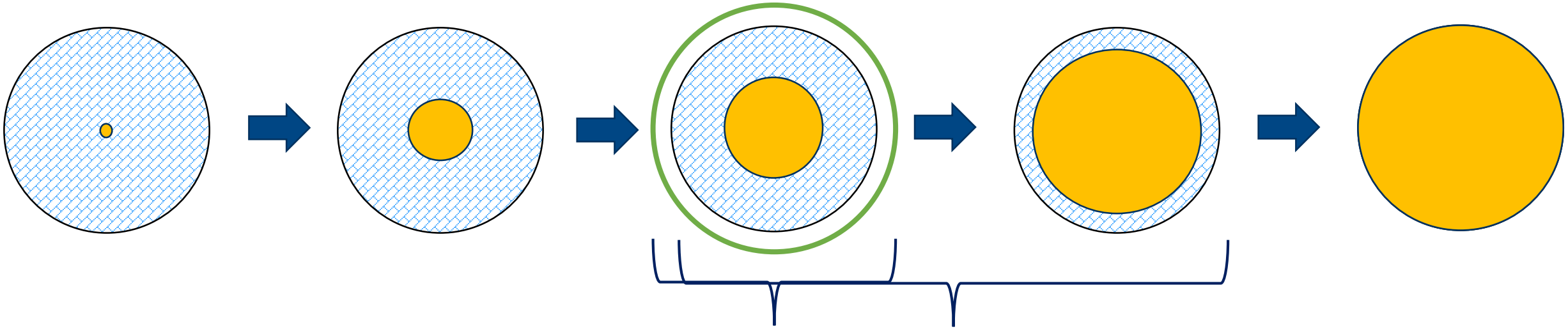
Starting point



Strategy 2 – Taking advantage of the available information



Strategy 2 – Taking advantage of the available information



1. **Wendell Bates**, A. Karimoddini, and M. Karimadini, "*Learning a Partially-Known Discrete Event System*," IEEE Access, 8:61806–61816, 2020.
2. **W. Bates**, A. Karimoddini, M. Karimadini, "*A Learning-based Approach for Diagnosis and Diagnosability of Initially Unknown Discrete Event Systems*", IEEE Systems, Man, and Cybernetics (under review)

Learning

Passive

In passive methods, a set of training data is supplied to the algorithm for model construction.

- Cannot adapt to new situations which would require new information
- Susceptible to missing information

Active

Active-learning techniques continuously engage the system and have the ability to inquire about any missing information.

Contribution

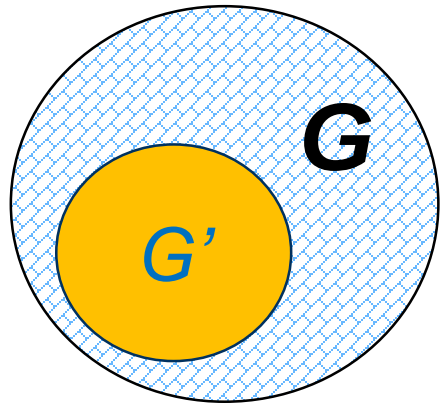
Problem Statement:

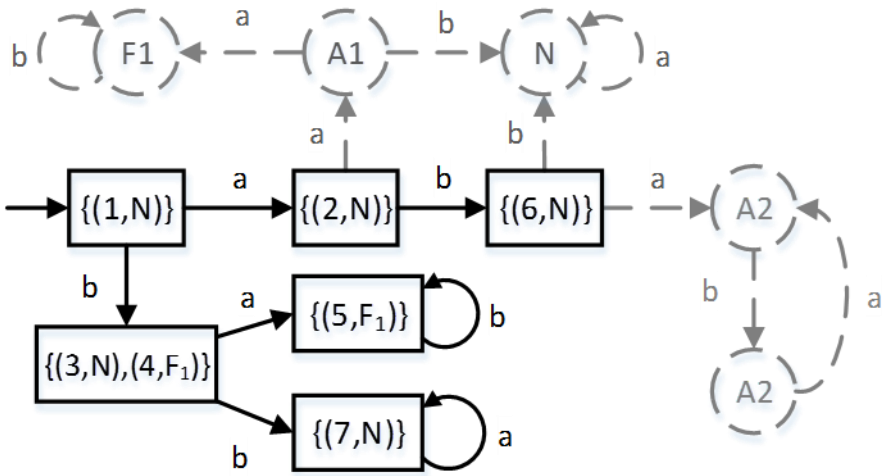
Consider a deterministic discrete event system G, whose sub-automaton G' is known. Given the known part, G', and the observable part of string s generated by G, P(s), determine fault occurrence in the system G and diagnose the type of occurred faults.

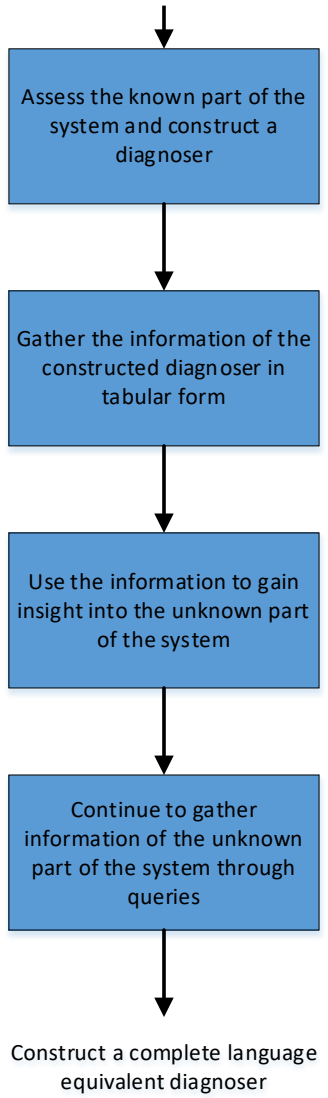
Proposed a novel approach for constructing a diagnosis tool pertaining to a partially-known DES system.

Definition:

- $\boldsymbol{G'} = (X', \Sigma, \delta', x'_0)$ *is a sub − automaton of* $\boldsymbol{G} = (X, \Sigma, \delta, x_0)$ *if and only if* $x'_0 = x_0, X' \subseteq X$, *and for any* $x_1, x_2 \in X' \subseteq X$ *and for any* $\sigma \in \sum$ *with any* $x_2 \in \delta(x_1, \sigma)$, *then* $x_2 \in \delta'(x_1, \sigma)$.







Diagnoser Functionality

System
$G = (X, \Sigma, \delta, x_0)$

$s \in \mathcal{L}(G)$

Natural Projection
$P: \Sigma^* \rightarrow \Sigma_o^*$

$P(s)$

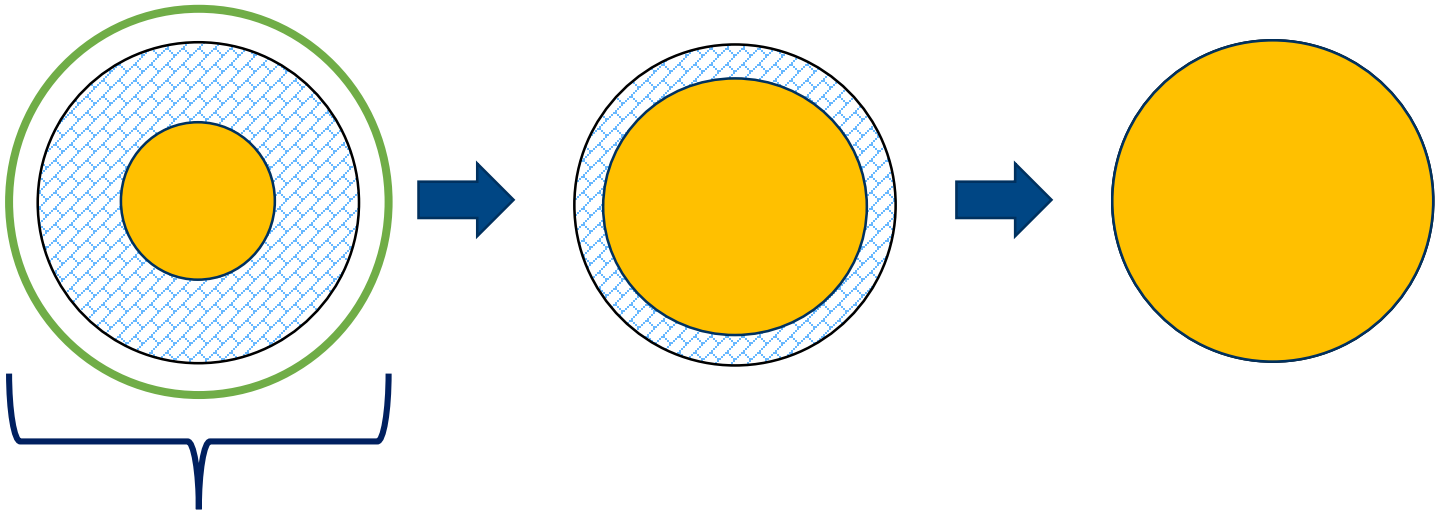
Diagnoser

f_i

- Fault Isolation
- Fault Detection
- Fault Location

- Diagnostics of the system under analysis are provided by the diagnoser.
- Using the observable behavior of the system, the diagnoser reveals information of the system's current state (faulty or normal).
- Upon the occurrence of an observable event in the system under analysis, the diagnoser updates its information on the system's condition.

Taking into account the known part



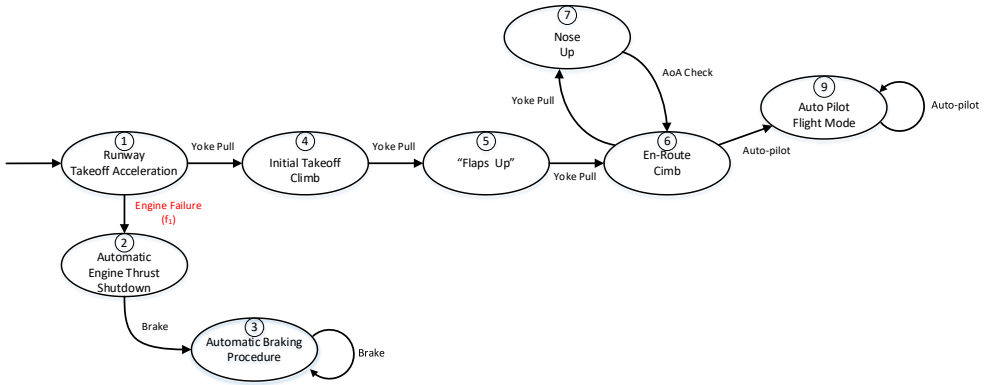
Building the Diagnoser for the Known Part

$$G_{d_k} = \left(Q_{d_k}, \Sigma_o, \delta_{d_k}, q_{0_{d_k}}\right)$$

- $q_{d_k} = \{(x'_1, \ell'_1), \ldots, (x'_k, \ell'_k)\}$, where
  - $x'_j \in X', \ell'_j \in L$
- Label Updating Function $\nabla : L \times \Sigma^* \rightarrow L$
  - $\ell' = \nabla(\ell, t) =$

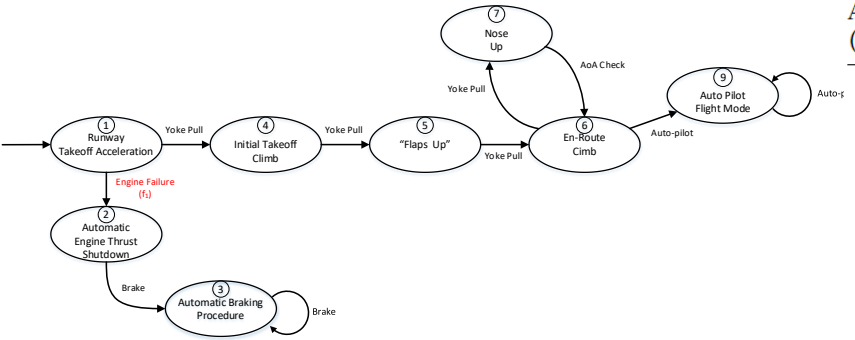
$$\begin{cases} \{N\}, & if\ \ell = \{N\}\ and\ \forall f, f \notin t \\ \{F_i \in \ell\ or\ \exists f \in \Sigma_{f_i}\}, & otherwise \end{cases}$$

- Under the assumption that G' is initially normal at x_0
  - $\bar{q}_{0_{d_k}} = \left\{\left(x, \nabla(\{N\}, s)\right) \mid s \in \mathcal{L}(G'), x \in \delta'(x'_0, s), s \in \Sigma_u^*\right\}$
  - $\delta_{d_k}(\bar{q}, e) = \left\{\left(y, \nabla(\ell, t)\right) \mid (x, \ell) \in \bar{q}, t \in \Sigma_u^*, y \in\right.$



The known part of system, G'.

Building the Diagnoser for the Known Part



Algorithm 1 Construction of the Diagnoser $G_{d_k} = (Q_{d_k}, \Sigma_o, \delta_{d_k}, q_{0_{d_k}})$ for $G' = (X', \Sigma, \delta', x'_o)$

Step 1: Construct $\bar{q}_{0_{d_k}}$

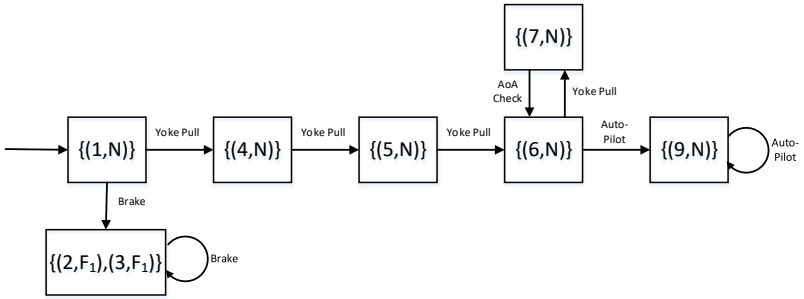
$\bar{q}_{0_{d_k}} = \{(x, \nabla(\{N\}, s)) \mid s \in \mathcal{L}(G'), x \in \delta'(x'_0, s), s \in \Sigma_u^*\}$

Step 2: Construct $\bar{Q}_d$ **and** $\bar{\delta}_d$

$\bar{Q}_{d_k} := \{\bar{q}_{0_{d_k}}\}$;
repeat
  for $\bar{q} \in \bar{Q}_{d_k}$ and $e \in \Sigma_o$ **do**
    if $\exists (x, \ell) \in \bar{q}$ s.t. $\delta'(x, e)$ is defined but $\bar{\delta}_{d_k}(\bar{q}, e)$ is not **then**
      $\bar{\delta}_{d_k}(\bar{q}, e) = \{(y, \nabla(\ell, t)) | (x, \ell) \in \bar{q}, t \in \Sigma_u^*, y \in \delta'(x, e.t)\}$;
      add $\bar{\delta}_{d_k}(\bar{q}, e)$ to $\bar{Q}_d$;
    end if
  end for
until there is no new state $\delta_d(q, e)$ for all $e \in \Sigma_o$.

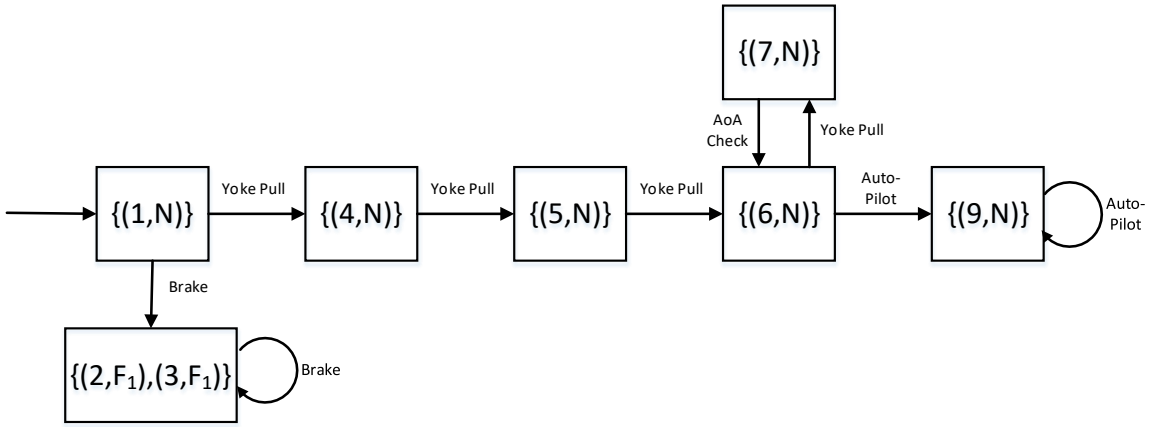
Step 3: Refine $\bar{Q}_d$ **and** $\bar{\delta}_d$

$q_{0_{d_k}} = Ref(\bar{q}_{0_{d_k}})$;
$Q_{d_k} = q_{0_{d_k}}$;
for $\bar{q} \in \bar{Q}_d$ **do**
  $Q_{d_k} = Q_{d_k} \cup Ref(\bar{q})$
  for $e \in \Sigma_o$ **do**
    if $\bar{q}' = \bar{\delta}_{d_k}(\bar{q}, e)$ **then**
      $Ref(\bar{q}') = \delta_{d_k}(Ref(\bar{q}), e)$;
    end if
  end for
end for



1. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," IEEE Transactions on Automatic Control, vol. 40, no. 9, pp. 1555–1575, 1995.

Capturing the Information From the Known Part in Tabular Form



V'		E'
		ε
S'	ε	{(1,N)}
	a	{(4,N)}
	b	0
	c	{(2,F_1),(3,F_1)}
	d	0
	e	0
	aa	{(5,N)}
	cc	{(2,F_1),(3,F_1)}
	⋮	0
	aaa	{(6,N)}
	aab	0
	aae	0
	ccc	{(2,F_1),(3,F_1)}
	⋮	0
	aaaa	{(7,N)}
	⋮	0
	aaae	{(9,N)}
	cccc	{(2,F_1),(3,F_1)}
	⋮	0
	eeee	0
$S'.\Sigma_o - S'$	aaaaa	0
	aaaab	0
	aaaac	0
	aaaad	{(6,N)}
	⋮	⋮
	cccccc	{(2,F_1),(3,F_1)}
	⋮	⋮
	eeeee	

Capturing the Information From the Known Part in Tabular Form

Observation Table

Observation Table: (S', E', T')

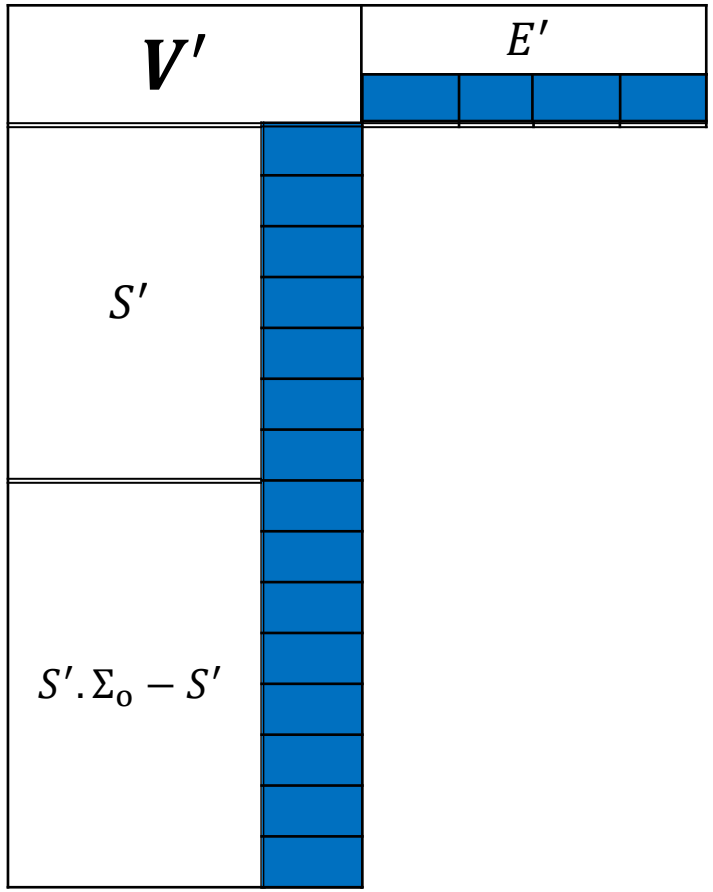
V'		E'			
S'					
$S'.\Sigma_o - S'$					

Capturing the Information From the Known Part in Tabular Form

Observation Table

Observation Table: (S', E', T')

- Rows and columns contain different strings

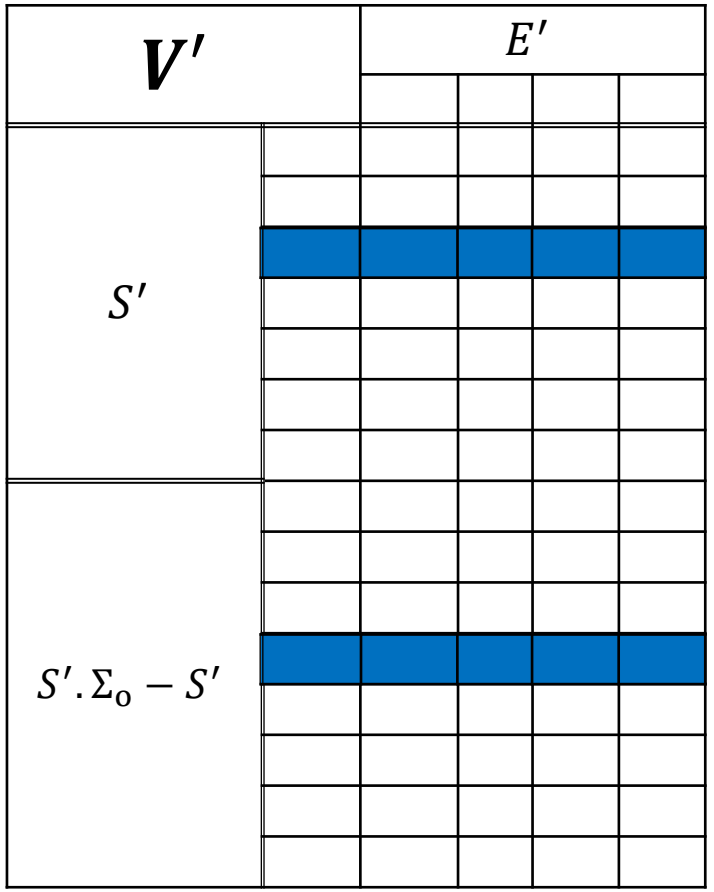


Capturing the Information From the Known Part in Tabular Form

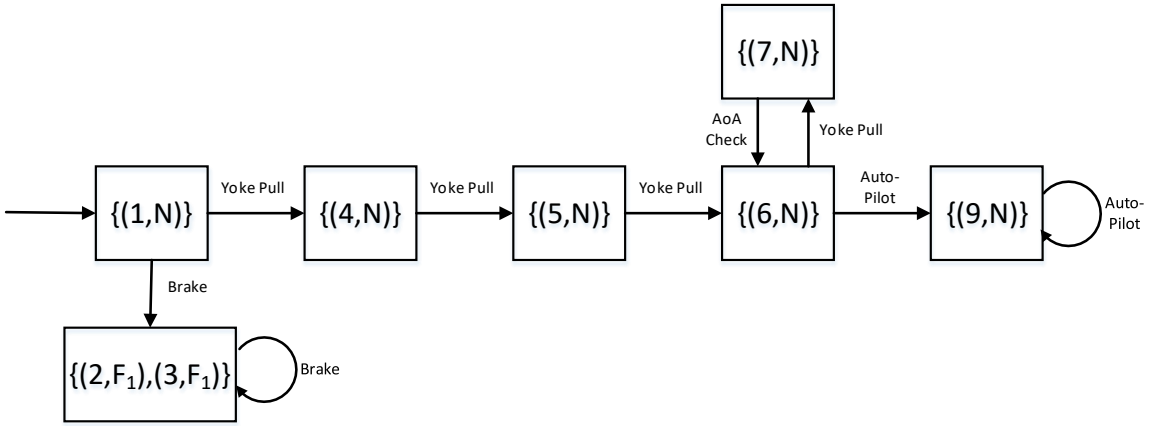
Observation Table

Observation Table: (S', E', T')

- Rows and columns contain different strings
- Rows in S' represent distinct states in the automata
  - *S' is a non-empty, prefix-closed set of finite strings*
- Rows in $(S'.\Sigma_o - S')$ represent states that might be reachable by Σ_o from a row/state in S'.
  - *$(S'.\Sigma_o - S')$ is a non-empty, prefix-closed set of finite strings*



Capturing the Information From the Known Part in Tabular Form



$$\hat{\ell} = \max_{q \in Q_{d_k}} \min_{s \in S'^{q}_{q_{0_{d_k}}}} |s|$$

$$where \ S'^{q}_{q_{0_{d_k}}} = \left\{ u \mid q \in \delta_d(q_{0_{d_k}}, u) \right\}$$

$$\hat{\ell} = 4, \qquad \Sigma = \{a, b, c, d, e\},$$

$$S' = Pre(\Sigma_o^{\hat{\ell}})$$

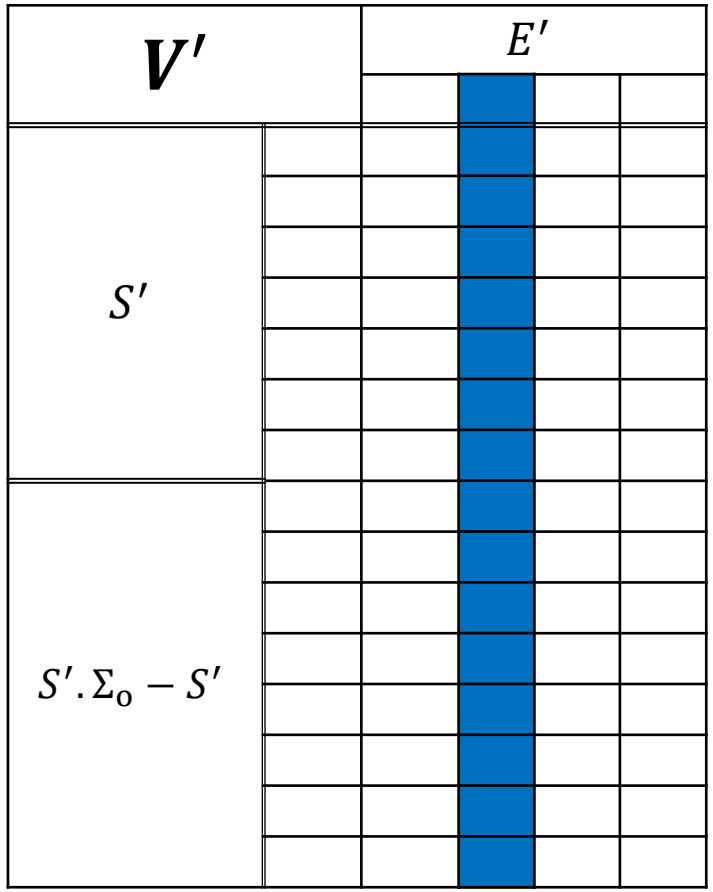
V'		E'
		ε
S'	ε	
	a	
	b	
	c	
	d	
	e	
	aa	
	cc	
	⋮	
	aaa	
	aab	
	aae	
	ccc	
	⋮	
	aaaa	
	⋮	
	aaae	
	cccc	
	⋮	
	eeee	
$S'.\Sigma_o - S'$	aaaaa	
	aaaab	
	aaaac	
	aaaad	
	⋮	
	⋮	
	⋮	
	⋮	
	ccccc	
	⋮ ⋮	
	eeeee	

Capturing the Information From the Known Part in Tabular Form

Observation Table

Observation Table: (S', E', T')

- Rows and columns contain different strings
- Rows in S' represent distinct states in the automata
  - *S' is a non-empty, prefix-closed set of finite strings*
- Rows in $(S'.\Sigma_o - S')$ represent states that might be reachable by Σ_o from a row/state in S'.
  - *$(S'.\Sigma_o - S')$ is a non-empty, prefix-closed set of finite strings*
- Columns represent experiments testing for new states
  - *E' is a non-empty, suffix-closed set of finite strings*



Diagnoser Construction

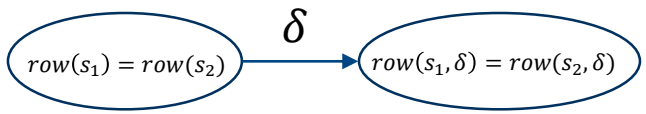
Completeness: (consistent and closed)

- Consistent – a table is consistent if and only if:

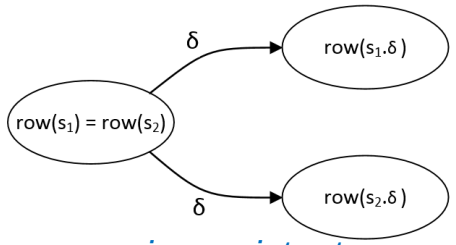
$$\forall s_1, s_2 \in \boldsymbol{S} \; with \; row(s_1) = row(s_2) \Rightarrow row(s_1.\sigma) = row(s_2), \forall \sigma \in \Sigma_o$$

– $\text{row}(\varepsilon) = \text{row}(ab) = \{N\}$ but $\text{row}(\varepsilon.a) = \{A_1\} \neq \text{row}(aba) = 0$

**consistency ensures that the automaton we are representing is deterministic*



consistent



inconsistent

V'		E'
		ε
S'	ε	N
	a	A_1
	b	0
	aa	F_1
	ab	N
	abb	N
$S'.\Sigma_o - S'$	ba	0
	bb	0
	aaa	F_1
	aab	0
	aba	0
	abba	N
	abbb	0

V'		E'	
		ε	a
S'	ε	N	A_1
	a	A_1	F_1
	b	0	0
	aa	F_1	F_1
	ab	N	0
	abb	N	N
$S'.\Sigma_o - S'$	ba	0	0
	bb	0	0
	aaa	F_1	F_1
	aab	0	0
	aba	0	0
	abba	N	N
	abbb	0	0

There exist $s_1, s_2 \in S'$ with $row(s_1) = row(s_2)$ and $\sigma \in \Sigma_o$ and $e \in E'$ such that $\text{row}(s_1\sigma.e) \neq \text{row}(s_2\sigma.e)$

Add σe to E'

Diagnoser Construction

Completeness: (consistent and closed)

- Consistent – a table is consistent if and only if:

$$\forall s_1, s_2 \in \boldsymbol{S'} \; with \; row(s_1) = row(s_2) \Rightarrow row(s_1.\sigma) = row(s_2), \forall \sigma \in \Sigma_o$$

– $\text{row}(\varepsilon) = \text{row}(\text{ab}) = \{\text{N}\}$ but $\text{row}(\varepsilon.\text{a}) = \{\text{A}_1\} \neq \text{row}(\text{aba}) = 0$

**consistency ensures that the automaton we are representing is deterministic*

- Closed – a table is closed if and only if:

$$\forall t \in (\boldsymbol{S'.\Sigma_o - S'}) \; with \; row(t) \neq (0, \cdots, 0), \exists s \in \boldsymbol{S'} \; such \; that \; row(s) = row(t)$$

– $row(s = aa)$ *exists in* $(\boldsymbol{S'.\Sigma_o - S'})$ *but does not exist in* $\boldsymbol{S'}$ → Add s to S'

**closedness ensures that all possible transitions to all existing states exists*

V'		E'
		ε
S'	ε	N
	a	A_1
	b	0
$S'.\Sigma_o - S'$	aa	F_1
	ab	N
	ba	0
	bb	0

→

V'		E'
		ε
S'	ε	N
	a	A_1
	b	0
	aa	F_1
$S'.\Sigma_o - S'$	ab	N
	ba	0
	bb	0
	aaa	F_1
	aab	0

Capturing the Information from the Known Diagnoser

Algorithm 2 Capturing the Known Diagnoser Information

1: **input:** $G_{d_k} = (Q_{d_k}, \Sigma_o, \delta_{d_k}, q_{0_{d_k}})$, the diagnoser for the known part of the automaton
2: **output:** The complete table $V' = (S', E', T')$ which captures the information of the diagnoser G_{d_k}
3: **initialization:** $S' = Pre(\Sigma_o^{\hat{\ell}})$ and $E' = \varepsilon$
4: Fill the the table $V' = (S', E', T')$ with proper labels using $T'((S' \cup S'.\Sigma_o).E')$
5: **return:** $V' = (S', E', T')$

780 Rows

V'		E'
		ε
S'	ε	{(1,N)}
	a	{(4,N)}
	b	0
	c	{(2,F_1),(3,F_1)}
	d	0
	e	0
	aa	{(5,N)}
	cc	{(2,F_1),(3,F_1)}
	⋮	0
	aaa	{(6,N)}
	aab	0
	aae	0
	ccc	{(2,F_1),(3,F_1)}
	⋮	0
	aaaa	{(7,N)}
	⋮	0
	aaae	{(9,N)}
	cccc	{(2,F_1),(3,F_1)}
	⋮	0
	eeee	0
$S'.\Sigma_o - S'$	aaaaa	0
	aaaab	0
	aaaac	0
	aaaad	{(6,N)}
	⋮	⋮
	cccccc	{(2,F_1),(3,F_1)}
	⋮	⋮
	eeeee	

Construction of the Diagnoser for the Known Part

Algorithm 1 Construction of the Diagnoser $G_{d_k} = (Q_{d_k}, \Sigma_o, \delta_{d_k}, q_{0_{d_k}})$ for $G' = (X', \Sigma, \delta', x'_o)$

Step 1: Construct $\bar{q}_{0_{d_k}}$

$\bar{q}_{0_{d_k}} = \{(x, \nabla(\{N\}, s)) \mid s \in \mathcal{L}(G'), x \in \delta'(x'_0, s), s \in \Sigma_u^*\}$

Step 2: Construct $\bar{Q}_d$ **and** $\bar{\delta}_d$

$\bar{Q}_{d_k} := \{\bar{q}_{0_{d_k}}\}$;

repeat

  for $\bar{q} \in \bar{Q}_{d_k}$ and $e \in \Sigma_o$ **do**

    if $\exists (x, \ell) \in \bar{q}$ s.t. $\delta'(x, e)$ is defined but $\bar{\delta}_{d_k}(\bar{q}, e)$ is not **then**

      $\bar{\delta}_{d_k}(\bar{q}, e) = \{(y, \nabla(\ell, t)) | (x, \ell) \in \bar{q}, t \in \Sigma_u^*, y \in \delta'(x, e.t)\}$;

      add $\bar{\delta}_{d_k}(\bar{q}, e)$ to $\bar{Q}_d$;

    end if

  end for

until there is no new state $\delta_d(q, e)$ for all $e \in \Sigma_o$.

Step 3: Refine $\bar{Q}_d$ **and** $\bar{\delta}_d$

$q_{0_{d_k}} = Ref(\bar{q}_{0_{d_k}})$;

$Q_{d_k} = q_{0_{d_k}}$;

for $\bar{q} \in \bar{Q}_d$ **do**

  $Q_{d_k} = Q_{d_k} \cup Ref(\bar{q})$

  for $e \in \Sigma_o$ **do**

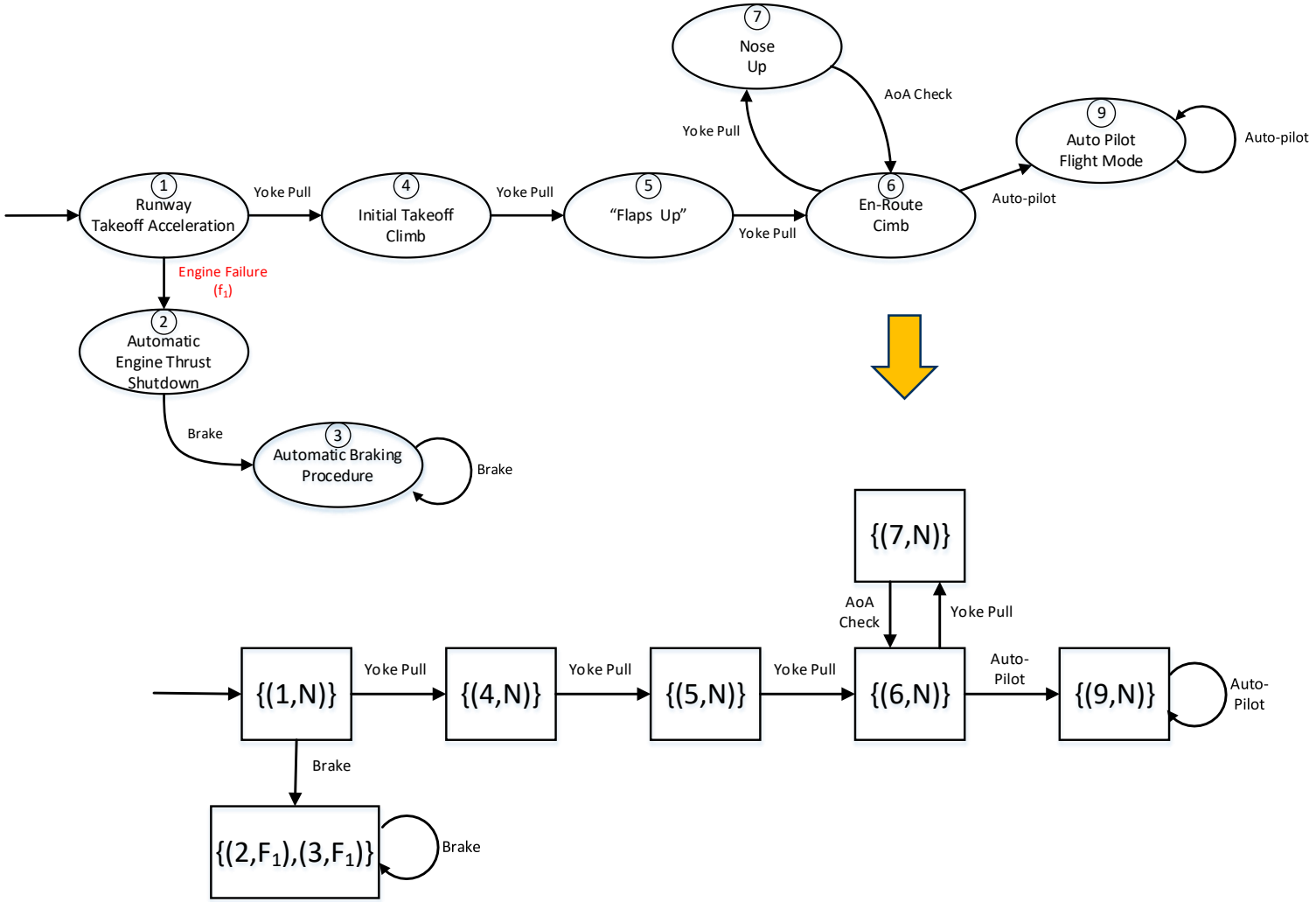
    if $\bar{q}' = \bar{\delta}_{d_k}(\bar{q}, e)$ **then**

      $Ref(\bar{q}') = \delta_{d_k}(Ref(\bar{q}), e)$;

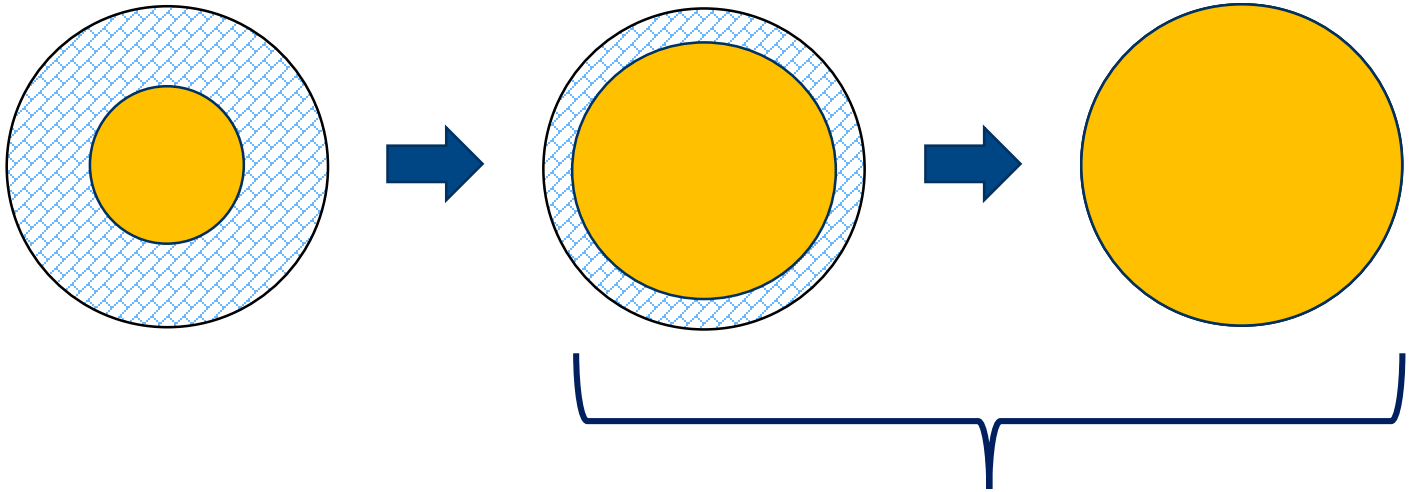
    end if

  end for

end for



Learning the Unknown Part and Building the Diagnoser



Proposed Diagnoser Structure

Modeled System
$G = (X, \Sigma, \delta, x_0)$

$s \in \mathcal{L}(G)$

Natural Projection
$P: \Sigma^* \rightarrow \Sigma_o^*$

$P(s)$

Diagnoser

f_i

- Fault Isolation
- Fault Detection
- Fault Location

$G_d = (Q, \Sigma_o, \delta_d, h, q_0)$ *where*

- Q – $(Q_{d_k} \cup Q_{d_u})$ the diagnoser states
  - » Q_{d_k} - the known diagnoser states
  - » Q_{d_u} - the unknown diagnoser states
- Σ_o – the event set (Σ)
- δ_d – the diagnoser's transition rule
- h – the output function
- q_0 – the initial state

Active Learning of the Unknown Information

Queries to the Oracle:

- ***Membership queries*** – does a particular string exist in $P(\mathcal{L}(G))$ of the unknown plant and is the string faulty?

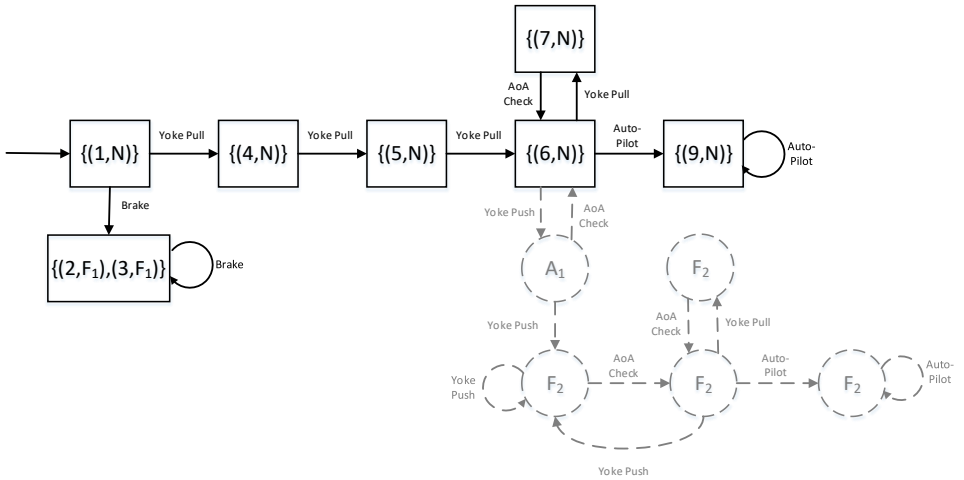
- ***Equivalence queries*** – does $\mathcal{L}(G_d) = P(\mathcal{L}(G))$? If not, the oracle provides a counterexample that exists within the symmetrical difference between $\mathcal{L}(G_d)$ and $P(\mathcal{L}(G))$.

Active Learning of the Unknown Information

Mapping function $T(s.t): (S_i.\Sigma_o \cup S_i).E_i \longrightarrow 2^{\left((X' \cup \star) \times L\right) \cup \{0\}}$
for $s \in (S_i.\Sigma_o \cup S_i)$ and $t \in E_i$

$\underline{T(s.t)}$:

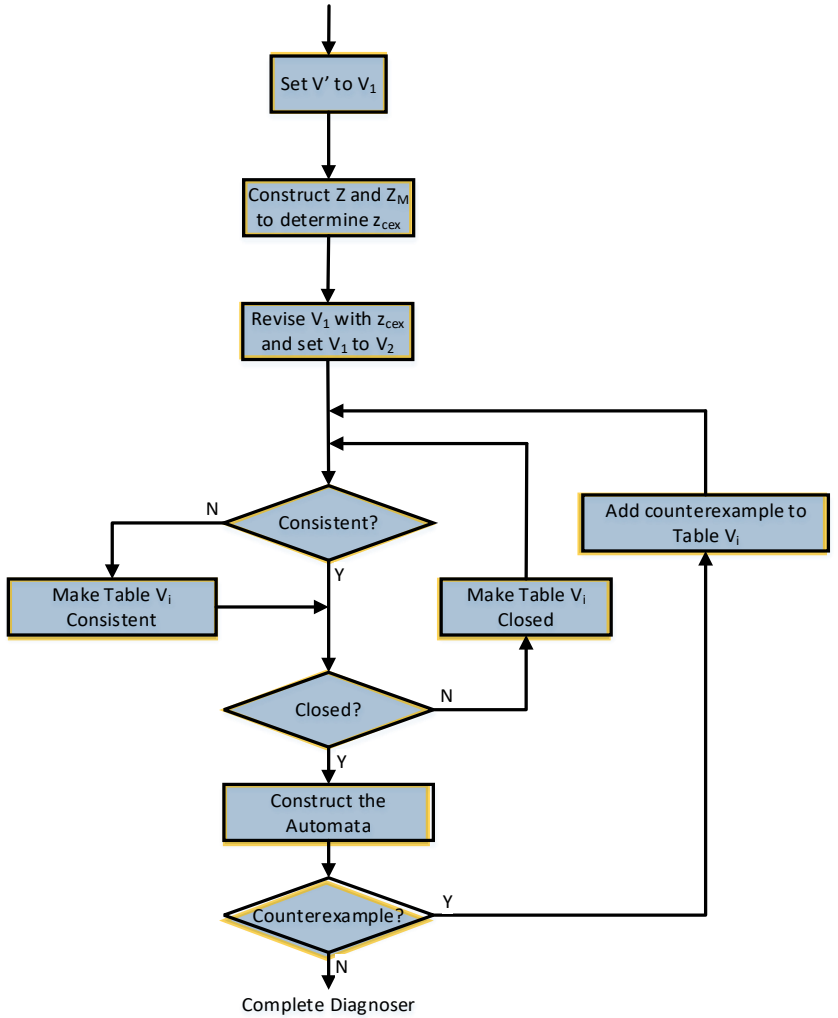
- $T(s.t) = \{0\}$ if $s.t$ is not in the $\mathcal{L}(G)$
- $T(s.t) = T'(s.t)$ if $s.t \in \mathrm{P}(\mathcal{L}(G'))$ is in the observable part of the language of the known part of the system
- $T(s.t) = \{(\star, N)\}$ if $s.t$ is a normal (non-faulty) observation existing within the observable language of the unknown part of the system, $\mathcal{L}(G) - \mathcal{L}(G')$
- $T(s.t) = \{(\star, \ell)\}, \ell \subseteq \{L_1, L_2, \ldots, L_n\}, L_i \in \{F_i, A_i\}$ where
  - $F_i \in \ell$ if $s.t \in \mathrm{P}(\mathcal{L}(G)) - \mathrm{P}(\mathcal{L}(G'))$ is the observation of a faulty string containing f_i
  - $A_i \in \ell$ if $s.t \in \mathrm{P}(\mathcal{L}(G)) - \mathrm{P}(\mathcal{L}(G'))$ creates ambiguity in the occurrence of fault f_i, meaning that $\exists u, u' \in P^{-1}(s.t)$ such that $f_i \in u$ and $f_i \notin u'$



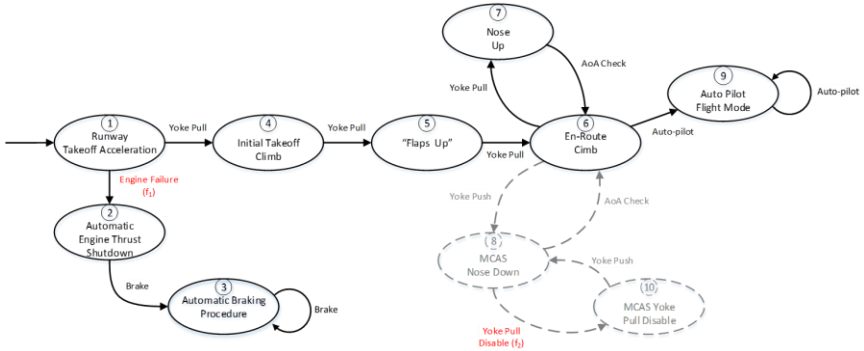
Active Learning of the Unknown Information

Algorithm 3 Active-learning of the Unknown Information

1: **input:** The complete table V' for the known part, the event set Σ_o
2: **output:** The complete diagnoser G_d with $\mathcal{L}(G_d) = P(\mathcal{L}(G))$
3: Set $V_1 = (S_1, E_1, T) = V'$
4: Find $Z = \{s.t. \mid s \in (S_1 \cup S_1.\Sigma_o),\ t \in E_1,\ T'(s.t) = 0\}$
5: Form $Z_M = \{z \mid T(z) \neq 0\}$ using membership queries and update the table with $T(z)$ for elements $z \in Z_M$.
6: Find $z_{cex} = \arg\min_{z \in Z_M} |z| \mid z \notin S_1$
7: Set $S_2 = S_1 \cup Pre(z_{cex})$ and $E_2 = E_1$
8: Fill the table V_2 over $(S_2 \cup S_2.\Sigma_o).E_2$ using the label propagation mechanism and membership queries
9: **while** $V_i(S_i, E_i, T)$ is not complete **do**
10: **if** V_i is not consistent **then**
11: Find $s_1, s_2 \in S_i$, $\sigma \in \Sigma_o$ and $e \in E_i$ such that $row(s_1) = row(s_2)$ but $T(s_1.\sigma.e) \neq T(s_2.\sigma.e)$;
12: Set $E_{i+1} = E_i \cup \sigma.e$ and $S_{i+1} = S_i$;
13: Set $i = i + 1$;
14: Form the new table V_i and fill it up over $(S_i \cup S_i.\Sigma_o).E_i$ using the label propagation mechanism and membership queries;
15: **end if**
16: **if** V_i is not closed **then**
17: Find $s_1 \in S$ and $\sigma \in \Sigma_o$ such that $row(s_1.\sigma)$
18: is different from $row(s)$ for all $s \in S_i$;
19: Set $S_{i+1} = S_i \cup \{s_1.\sigma\}$, $E_{i+1} = E_i$;
20: Set $i = i + 1$;
21: Form the new table V_i and fill it up
22: over $(S_i \cup S_i.\Sigma_o).E_i$ using the label propagation mechanism and membership queries;
23: **end if**
24: **end while**
25: Construct the automaton $G_d(V_i)$
26: Ask equivalence query if $\mathcal{L}(G_d(V_i)) = \mathcal{L}(G)$.
27: **if** The oracle replies with the counterexample cex **then**
28: Set $S_{i+1} = S_i \cup Pre(cex)$, $E_{i+1} = E_i$;
29: Set $i = i + 1$;
30: Form the new table V_i and fill it up over
31: $(S_i \cup S_i.\Sigma_o).E_i$ using the label propagation mechanism and membership queries;
32: Go to *line* 8.
33: **end if**
34: **return:** $G_d = G_d(V_i)$



Algorithm Flow Chart



System being assessed

Active Learning of the Unknown Information

Zero-Element sets Z, Z_M, and z_{cex}:

For any row $s \in (S_i.\Sigma_o \cup S_i)$ and any column $s \in E_i$

- Z is the set of all zero elements within table V_i

$$Z = \{s.t \mid s \epsilon (S_1 \cup S_1.\Sigma_o), t \epsilon E_1, T'(s.t) = 0\}$$

- Z_M is the set of all string that exist after asking membership queries of Z to the oracle

$$Z_M = \{z \epsilon Z \mid T(z) \neq 0\}$$

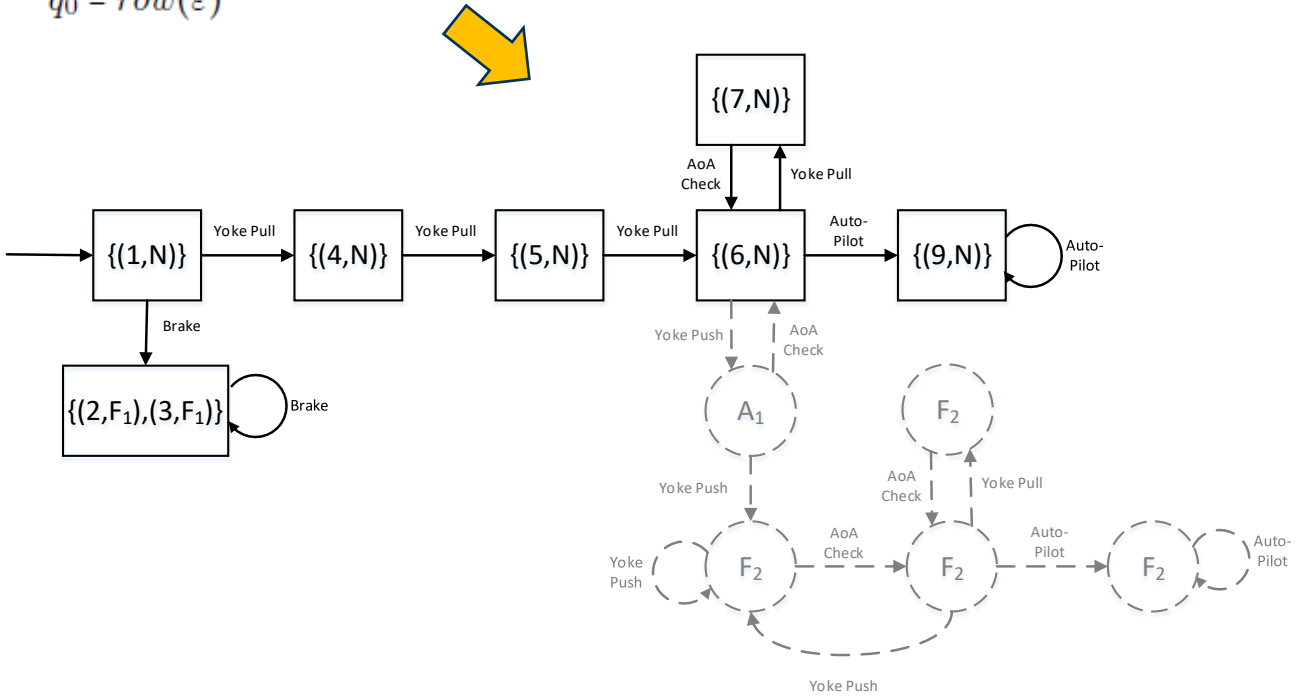
- z_{cex} is the shortest string in Z_M →

V'		E'
		ε
S'	ε	{(1,N)}
	a	{(4,N)}
	b	0
	c	{(2,F_1),(3,F_1)}
	d	0
	e	0
	aa	{(5,N)}
	cc	{(2,F_1),(3,F_1)}
	⋮	0
	aaa	{(6,N)}
	aab	0
	aae	0
	ccc	{(2,F_1),(3,F_1)}
	⋮	0
	aaaa	{(7,N)}
	⋮	0
	aaae	{(9,N)}
	cccc	{(2,F_1),(3,F_1)}
	⋮	0
	eeee	0
$S'.\Sigma_o - S'$	aaaaa	0
	aaaab	0
	aaaac	0
	aaaad	{(6,N)}
	⋮	⋮
	cccccc	{(2,F_1),(3,F_1)}
	⋮	⋮
	eeeee	

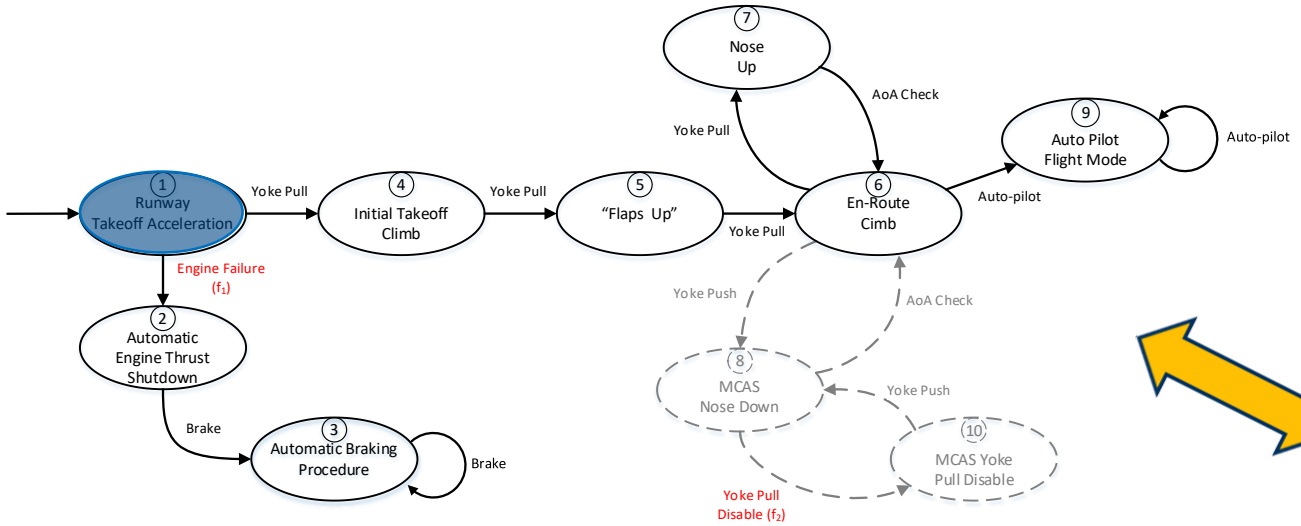
Active Learning of the Unknown Part

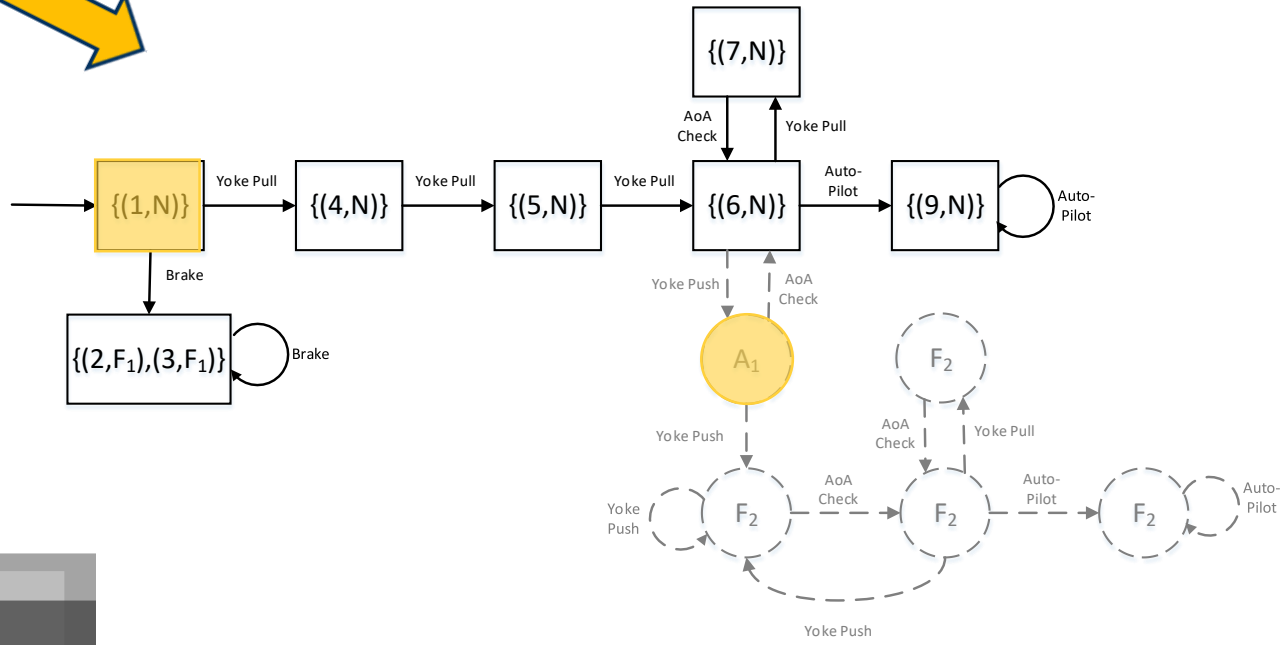
V		E		
		ε	a	⋯
S	ε	{(1,N)}	{(4,N)}	⋯
	a	{(4,N)}	{(5,N)}	⋯
	b	0	⋯	⋯
	c	{(2,F_1),(3,F_1)}	0	0
	d	0	0	0
	e	0	0	0
	aa	{(5,N)}	{(6,N)}	⋯
	cc	{(2,F_1),(3,F_1)}	0	⋯
	⋮	0	0	0
	aaa	{(6,N)}	{(7,N)}	⋯
	aab	0	0	0
	aae	0	0	0
	ccc	{(2,F_1),(3,F_1)}	0	⋯
	⋮	0	0	0
	aaaa	{(7,N)}	0	⋯
	aaab	{(⋆,A_2)}	0	
	aaae	{(9,N)}	0	⋯
	cccc	{(2,F_1),(3,F_1)}	0	⋯
	⋮	0	0	
	eeee	0	0	0
	aaabbd	{(⋆,F_2)}	0	⋯
	aaabbd	{(⋆,F_2)}	{(⋆,F_2)}	⋯
	⋮	⋮		⋯
	aaabbde	{(⋆,F_2)}	0	⋯
$S.\Sigma_o - S$	aaaaa	0	0	⋯
	aaaab	0	0	⋯
	aaaac	0	0	⋯
	aaaad	{(6,N)}	{(7,N)}	⋯
	⋮	⋮	⋯	⋯
	⋮		⋯	⋯
	⋮		⋯	⋯
	⋮			⋯
	ccccc	{(2,F_1),(3,F_1)}	⋯	⋯
	⋮ ⋮	⋮	⋯	⋯
	eeeee		0	⋯

$$Q = \{row(s) \mid s \in S_i,\ T(s.\varepsilon) \neq \{0\}\}$$
$$\delta_d(row(s), \sigma) = row(s.\sigma), for\ all\ row(s) \in Q$$
$$h(row(s)) = T(s.\varepsilon), for\ all\ row(s) \in Q$$
$$q_0 = row(\varepsilon)$$



Complete Diagnoser





Diagnosability

(F_i-*Diagnosability*) The plant G with the live language $\mathcal{L}(G)$ is said to be F_i-*diagnosable* with respect to the failure f_i and the natural projection P, if for all strings $s \in \mathcal{L}(G)$ with $f_i \in s$, there exists an upper bound $n_i \in \mathbb{N}$ such that for any string $t \in \mathcal{L}(G)/s$ with $|t| \geq n_i$, the following condition holds:

$$\{\forall w \in P^{-1}(P(s.t))\} \Rightarrow f_i \in w$$

The plant G is diagnosable, if it is F_i-*diagnosable* with respect to all $f_i \in \Sigma_F$

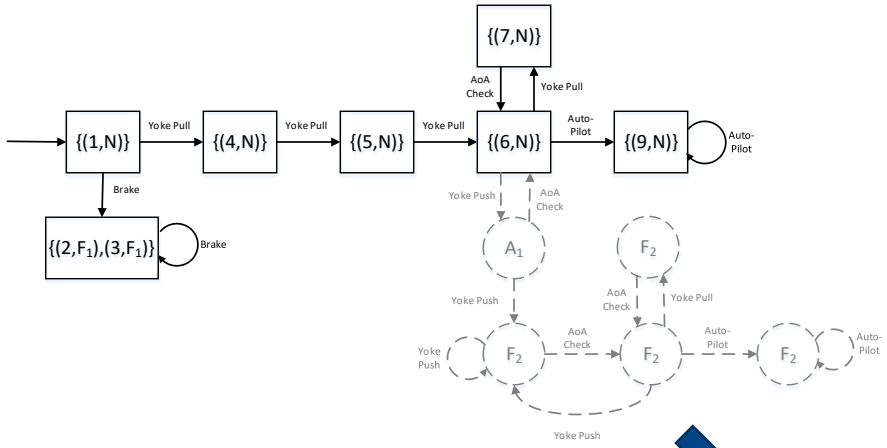
- A state $q \in Q$ in G_d is considered to be F_i-*certain* if it is labeled faulty (i.e., $F_i \in h(q)$), and it is called F_i-*ambiguous* if it is labeled ambiguous (i.e., $\{\star, A_i\} \in h(q)$).
- States $q_1, q_2, \ldots, q_n$ form a cycle in G_d if $q_{k+1} = \delta(q_k, e_k)$, $k = 1,\ldots, n-1$, $q_1 = \delta(q_n, e_n)$, $e_k > \Sigma_o$, $k = 1,\ldots,n$.
- If a partially-known system G has no cycle of F_i-*ambiguous* states, then the constructed diagnoser G_d can diagnose the fault f_i in at most n_i transitions,

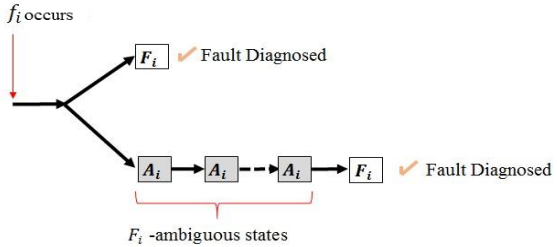
$$n_i = (n_o + 1)(I_i + 1)$$

where n_o is the maximum number of consecutive unobservable events that can occur before an observable event is reached, I_i is the total number of consecutive F_i-*ambiguous* states in the diagnoser.

Diagnosability

In the developed diagnoser G_d, if there is no cycle of F_i-*ambiguous* states, then the plant G is F_i-*diagnosable*.





- ✓ F_1 – diagnosable
- ✓ F_2 – diagnosable

W. Bates, A. Karimoddini, M. Karimadini, "*A Learning-based Approach for Diagnosis and Diagnosability of Initially Unknown Discrete Event Systems*", IEEE Transaction on Industrial Electronics (under preparation for submission).

Diagnosability

(F_i-Diagnosability) The plant G with the live language $\mathcal{L}(G)$ is said to be F_i-*diagnosable* with respect to the failure f_i and the natural projection P, if for all strings $s \in \mathcal{L}(G)$ with $f_i \in s$, there exists an upper bound $n_i \in \mathbb{N}$ such that for any string $t \in \mathcal{L}(G)/s$ with $|t| \geq n_i$, the following condition holds:

$$\{\forall w \in P^{-1}(P(s.t))\} \Rightarrow f_i \in w$$

The plant G is diagnosable, if it is F_i-*diagnosable* with respect to all $f_i \in \Sigma_F$

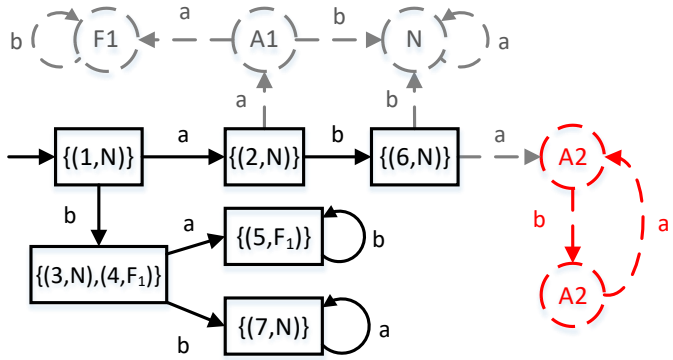
- A state $q \in Q$ in G_d is considered to be F_i-*certain* if it is labeled faulty (i.e., $F_i \in h(q)$), and it is called F_i-*ambiguous* if it is labeled ambiguous (i.e., $\{\star, A_i\} \in h(q)$).
- States $q_1, q_2, \ldots, q_n$ form a cycle in G_d if $q_{k+1} = \delta(q_k, e_k)$, $k = 1,\ldots, n-1$, $q_1 = \delta(q_n, e_n)$, $e_k > \Sigma_o$, $k = 1,\ldots,n$.
- If a partially-known system G has no cycle of F_i-*ambiguous* states, then the constructed diagnoser G_d can diagnose the fault f_i in at most n_i transitions,

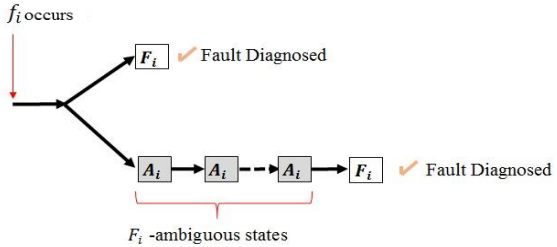
$$n_i = (n_o + 1)(I_i + 1)$$

where n_o is the maximum number of consecutive unobservable events that can occur before an observable event is reached, I_i is the total number of consecutive F_i-*ambiguous* states in the diagnoser.

Diagnosability

In the developed diagnoser G_d, if there is no cycle of F_i-*ambiguous* states, then the plant G is F_i-*diagnosable*.





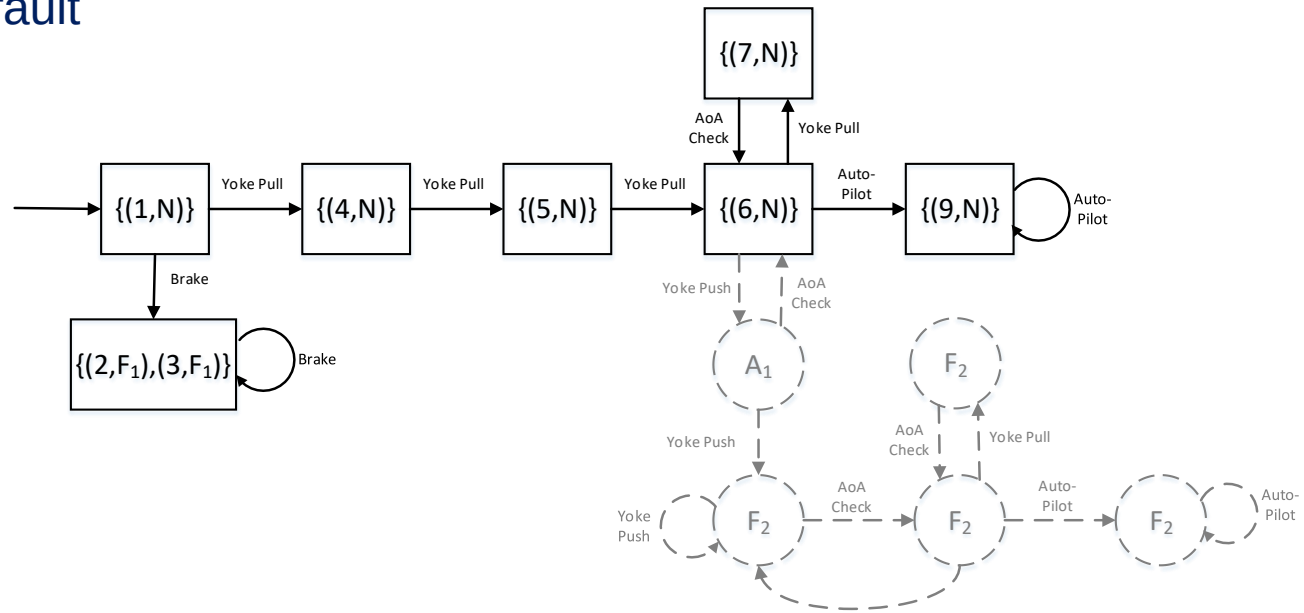
✓ F_1 – diagnosable

× F_2 – undiagnosable

W. Bates, A. Karimoddini, M. Karimadini, "*A Learning-based Approach for Diagnosis and Diagnosability of Initially Unknown Discrete Event Systems*", IEEE Transaction on Industrial Electronics (under preparation for submission).

Properties of our Developed Algorithms

- Developed a novel technique to construct a diagnoser for a partially-known system
  - Captures the information from the diagnoser corresponding to the known part in tabular form
  - Utilizes active-learning to discover the unknown part
  - Terminates in a finite number of iterations
  - Derived a condition to verify diagnosability
  - Obtained an upper bound for diagnosing a fault



Conclusion

- Developed a model for the takeoff procedure of an aircraft (Boeing-737 MAX)
- Developed an algorithm for the construction of a diagnoser when applied to a partially-known system
- Derived a condition to verify the diagnosability of a constructed diagnoser for a partially-known system.
- Applied the technique to construct a diagnoser for a Boeing-737 MAX case study regarding a partially-known system during the takeoff procedure.

Publications & Conference Presentations

Publications:

1. **I. Wendell Bates**, A. Karimoddini, and M. Karimadini, "*Learning a Partially-Known Discrete Event System*," IEEE Access, 8:61806–61816, 2020.
2. M. Karimi, A. Karimoddini, A. White, **W. Bates**, "*Event-Based Fault Diagnosis for an Unknown Plant*," Proc. of 55th IEEE Conference on Decision and Control.
3. **W. Bates**, A. Karimoddini, M. Karimadini, "*A Learning-based Approach for Diagnosis and Diagnosability of Initially Unknown Discrete Event Systems*", IEEE Systems, Man, and Cybernetics (under review)
4. **W. Bates**, A. Karimoddini, M. Karimadini, "*Learning-based Diagnosis for a Partially-Known Discrete Event System*", (under preparation for submission)
5. **W. Bates**, A. Karimoddini, M. Karimadini, "*Event-based Learning Technique for Fault Diagnosis of the Boeing 737 Maneuvering Characteristics Augmentation System (MCAS)",* Journal Paper, (under preparation for submission)
6. **W. Bates**, A. Karimoddini, M. Karimadini, "*MATLAB Automata Active-Learning Package*", Conference Paper (under development)
7. **W. Bates**, A. Karimoddini, M. Karimadini, "*MATLAB Automata Diagnoser Active-Learning Package*", Conference Paper (under development)

Special Thanks and Acknowledgments

Thank You

- Dr. Karimoddini and Dr. Homaifar
- Friends and Colleagues at ACCESS/TECHLAV Labs
- Air Force Research Laboratory and Department of Defense

ACCESS Laboratory

DoD Center of Excellence in Autonomy

TECHLAV

Testing, Evaluation, and Control of Heterogeneous Large-scale Systems of Autonomous Vehicles

QUESTIONS?