ACCESS LAB
Autonomous Cooperative Control of Emergent Systems of Systems

TECHLAV

An Active Learning Approach For Inferring Discrete Event Automata

Mohammad Mahdi Karimi

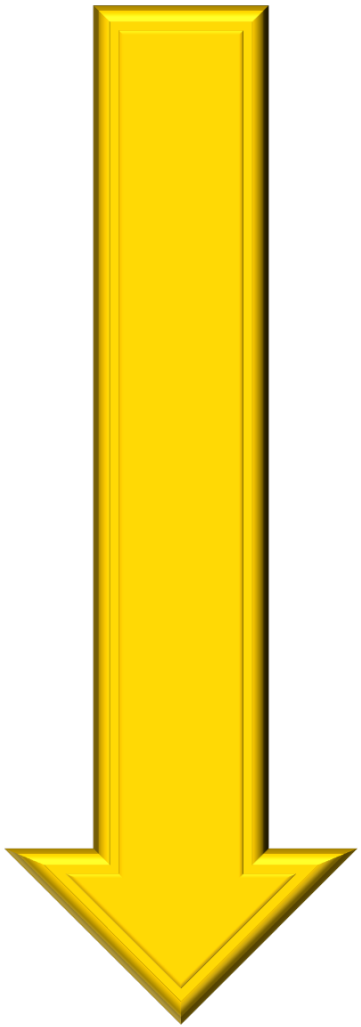
PhD. Candidate, ECE

Supervisor: Dr Ali Karimoddini

Summer 2015

Content

1. Discrete Event Systems
   - Definitions
   - Applications
2. Automata Theory
   - Language
   - Regular Language
   - Automata Representation
   - Modeling in Automata
3. L^{star} Learning
   - Definitions
   - Algorithm
   - UAV Example
4. L^{star} Features
5. L^{star} Matlab Toolbox
6. Conclusion



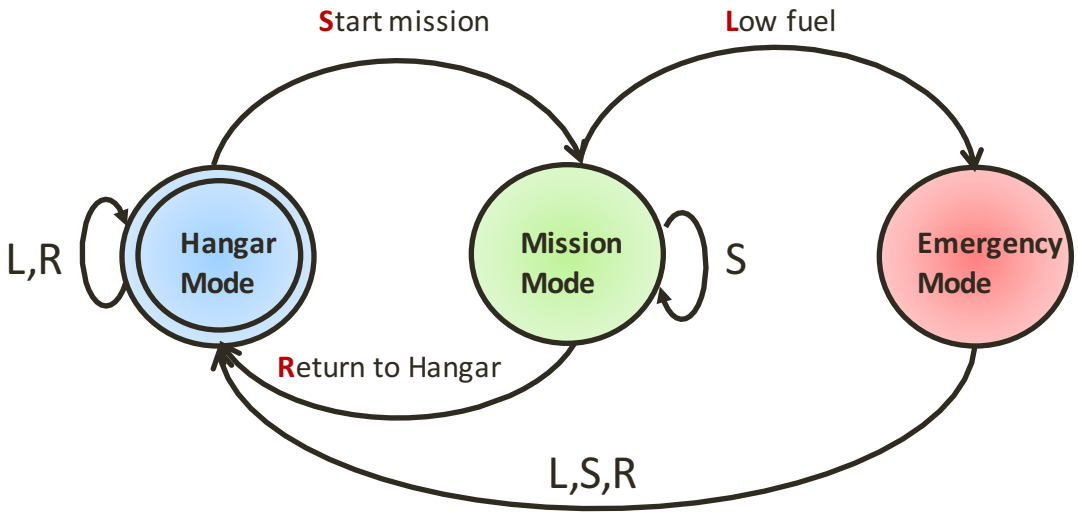
Definitions and Applications

SECTION 1: DISCRETE EVENT SYSTEMS

What is DES?

Discrete Event Systems:

Definition: A Discrete Event System (DES) is a discrete-state, event-driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time.

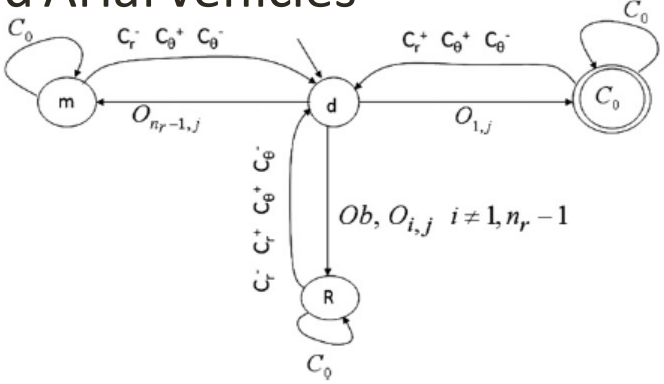


Motivation, Why DES?

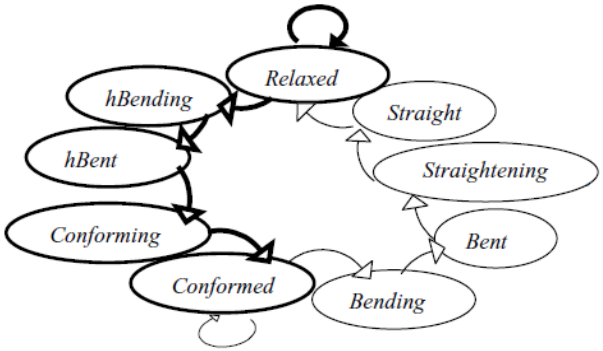
- To treat large-scale systems we need an abstract and simple tool:
  - DES can effectively model large scale systems in an abstract mathematical fashion.
- DES can capture system's logics and rules and used for constructing the decision making units of systems.
- Many systems are inherently DES (Ex. Queuing system)
- Many others can be abstracted to DES (Ex. Robotic)

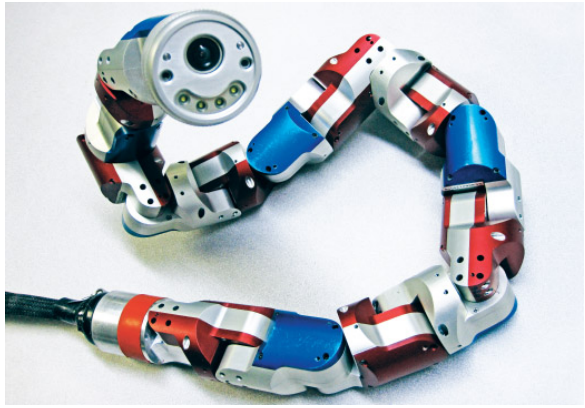
Some DES Examples

- Unmanned Arial Vehicles



- Robotic applications

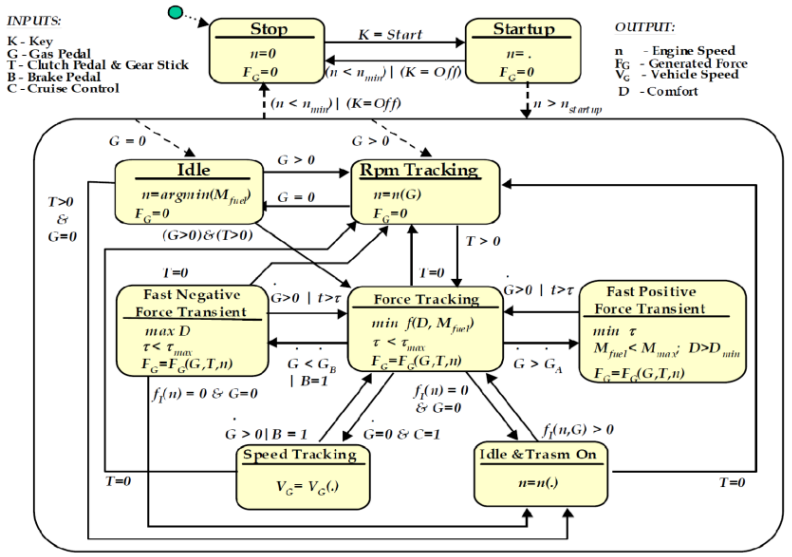




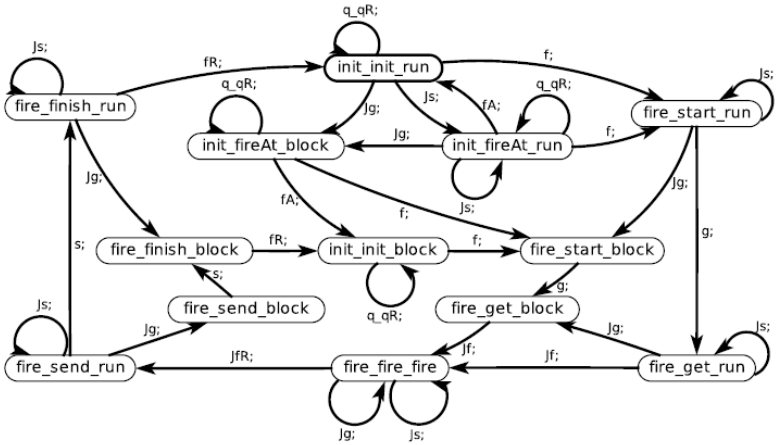
-Karimoddini, Ali, et al. "Hybrid formation control of the unmanned aerial vehicles." *Mechatronics* 21.5 (2011): 886-898.
-Tzionas, Panagiotis G., Adonios Thanailakis, and Philippos G. Tsalides. "Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata." Robotics and Automation, IEEE Transactions on 13.2 (1997): 237-250.

Some DES Applications

- Power-train Specification



- Internet Of Things(IOT):



Balluchi, Andrea, et al. "Automotive engine control and hybrid systems: Challenges and opportunities." *Proceedings of the IEEE* 88.7 (2000): 888-912.

Lohstroh, Marten, Lee, Edward A., "An Interface Theory for the Internet of Things." 13th International Conference on Software Engineering and Formal Methods (SEFM), 2015.

1. Discrete Event Systems
   - Definitions
   - Applications
2. Automata Theory
   - Language
   - Regular Language
   - Automata Representation
   - Modeling in Automata
3. L^{star} Learning
   - Definitions
   - Algorithm
   - UAV Example
4. L^{star} Features
5. L^{star} Matlab Toolbox
6. Conclusion

Language and Automata

AUTOMATA THEORY

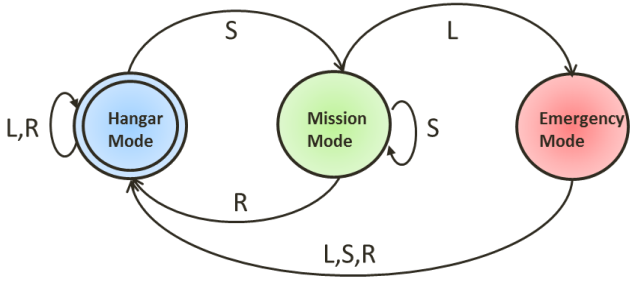
What is a Language?

- Behavior of a deterministic DES can be equivalently described by the set of all possible sequences of **events**:

$$\sigma_1\sigma_2\ldots$$

- and the **initial state** X_0. Each such event sequence is called a **trace** or **string** of the system, and a collection of traces starting at X_0 and ending at final states is called a (**marked**) **language**.

$L=\{L,R, SR, SLR,\ldots\}$



Problem:

To describe a system with its language, we may come up with a set of infinite strings.

Kumar, Ratnesh, and Vijay K. Garg. *Modeling and control of logical discrete event systems*. Vol. 300. Springer Science & Business Media, 2012.

Regular Languages

- Regular language is a class of language. In **regular languages**, a language is an infinite set, but it can be represented using finite state machines (Automata).

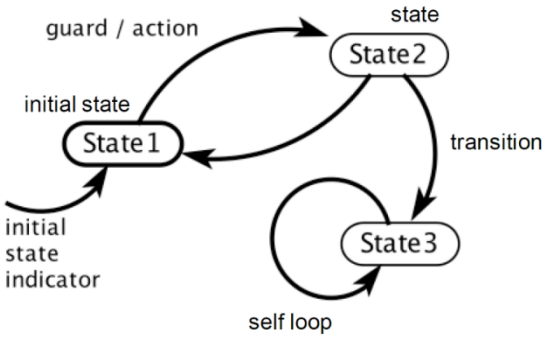
- **Theorem:**
  - Given a regular language model (K_m, K), there exists a DFSM, $G=(X,\sum,\alpha,x_0,X_m)$, such that $(L_m(G),L(G))=(K_m,K)$.

- It is proved that for any given regular language there is a DFSM,
- **Question** is now how to represent model in more graphical way?

Kumar, Ratnesh, and Vijay K. Garg. *Modeling and control of logical discrete event systems*. Vol. 300. Springer Science & Business Media, 2012.

Automata Representation

- A deterministic finite automaton is represented formally by a 5-tuple (Q,Σ,δ,q_0,F)
  - Q is a finite **set of *states***.
  - Σ is a finite set of symbols, called the **alphabet** (A) of the automaton.
  - δ is the **transition function**, that is, $\delta: Q \times \Sigma \rightarrow Q$.
    - $\mathbf{row}(sa) = \boldsymbol{\delta}\mathbf{(row}(s)\mathbf{,}a\mathbf{)}$
  - q_0 is the **start state**, that is, the state of the automaton before any input has been processed, where $q_0 \in Q$.
  - F is a set of states of Q (i.e. $F \subseteq Q$) called accepted states or **marked states**.
    - $\mathbf{F}=\{row(s) | s \in S \wedge T(s) = 1\}$



Problem:

How to model a system by an Automaton.

How to model system with Automata

- Analytical modeling
  - All information about the system is available in the form of a language and we try to build a DFSM out of given strings. (e.g. Myhill-Nerode construction method)
  - **Question:**
  - How about the case we do not have information about the system? We need to **learn** the system.
- Learning techniques

Passive Learning:	**Active Learning:**

Passive Learning Approach:

- Teacher provides abounding number of examples
- Learner fit a model for the provided examples.

- The learner passively learns the trained information and only can work on the training range.

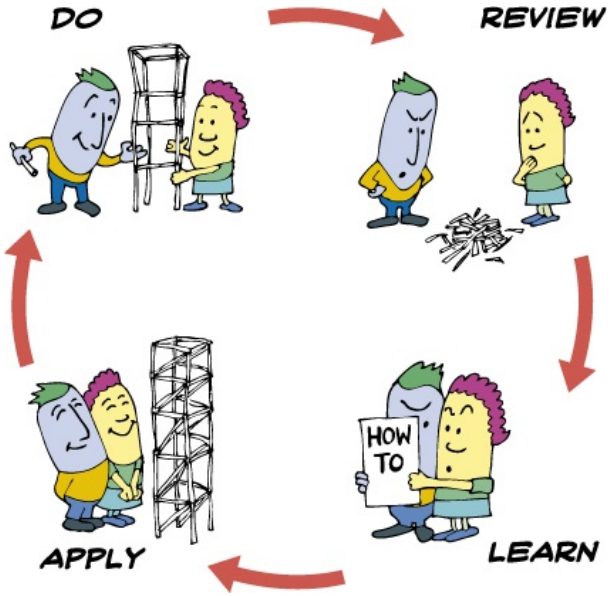
Question:

How about the case that a new situation happens and the learner is not trained for it?

Active Learning Approach:

- Imagine we don't have information about the system and we do not have access to the system.
- In this method, the learner tries to actively build the model by asking minimal questions from a teacher.
- The teacher is an expert person who can answer the learner's questions about the system.

- L^{star} Learning is and example of active learning



1. Discrete Event Systems
   - Definitions
   - Applications
2. Automata Theory
   - Language
   - Regular Language
   - Automata Representation
   - Modeling in Automata
3. L^{star} Learning
   - Definitions
   - Algorithm
   - UAV Example
4. L^{star} Features
5. L^{star} Matlab Toolbox
6. Conclusion

Algorithm and UAV Example

L^{STAR} LEARNING

L^{star} Learning

- The L^{star} algorithm has introduces by Angluin in 1987.
- This technique actively identifies an unknown regular language, and generates a **minimum state** Deterministic Finite Automata (**DFA**).
- It is assumed that the regular set is available to a minimally adequate Teacher which can answer the learners questions and validate the model.
- The learner actively asks queries during the learning process to build an observation table and eventually comes up with conjecture set and DFSM.

L^{star} learning

- **Queries:**

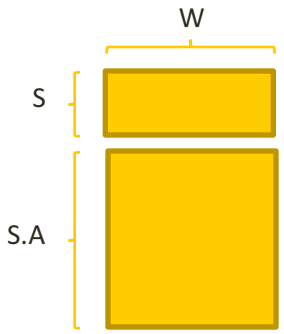
  Two types of queries are asked by learner:

  1. *Membership queries*
     - whether a certain input sequence is contained in *U*
  2. *Equivalence queries*
     - whether the constructed DFA *M* is correct, if not, learner asks for counter example.

- **Counter example:**
  - Those strings where are either are in system but not in marked language or vise versa, are provided to the learner as counter example of the Automata:

$$C = \{\forall t \in Q | t \epsilon U / L_M \wedge t \epsilon L_M / U\}$$

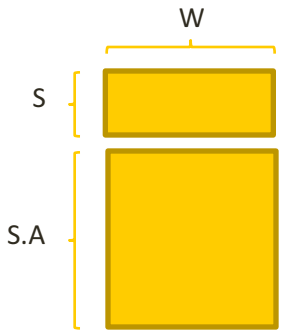
Cont.



- Observation table consist of three sections:
  - a nonempty finite prefix-closed set S of strings
  - a nonempty finite suffix-closed set E of strings
  - a finite function T mapping ((S∪S.A). E) to {0,1}

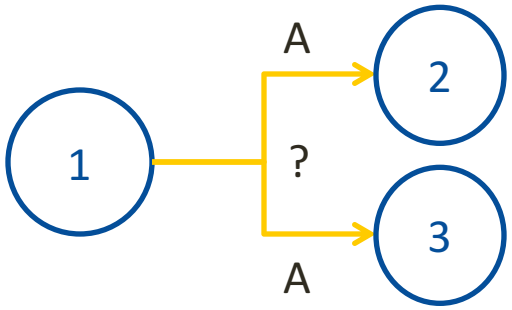
- The interpretation of T is that T(u) is 1 if and only if u is a member of the unknown regular set, U.

Consistency and Closeness



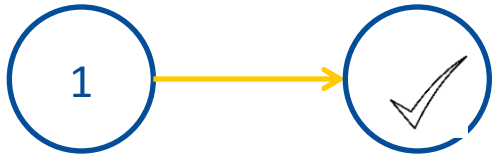
- Consistent
  - An observation table is called consistent provided that whenever s_1 and s_2 are elements of S such that row(s_1) = row(s_2), for all a in A .

$$\{\boldsymbol{s_1, s_2 \in S, a \in A, w \in W} | \\ \mathbf{row(s_1) = row(s_2)} \vee \ \boldsymbol{T(s_1 a w) = T(s_2 a w)}\}$$



- Closed
  - An observation table is considered closed if for each t in S.A there exists an s in S such that:

$$\{\boldsymbol{s \in S, a \in A, \forall t \in S} | \\ \mathbf{row(sa) = row(t)}\}$$



Solution

- **Not Consistent**
  - If the observation table is not consistent:

$$\boldsymbol{aw} = \{\boldsymbol{s_1}, \boldsymbol{s_2} \in \boldsymbol{S}, \boldsymbol{a} \in \boldsymbol{A}, \boldsymbol{w} \in \boldsymbol{W} | \mathbf{row}(\mathbf{s_1}) = \mathbf{row}(\mathbf{s_2}) \vee \ \boldsymbol{T}(\boldsymbol{s_1 aw}) \neq \boldsymbol{T}(\boldsymbol{s_2 aw})\}$$

- *aw* is added to *W* and update the observation table.

- **Not Closed**
  - If the observation table is not closed:
  - $\boldsymbol{sa} = \{\boldsymbol{s} \ \in \boldsymbol{S}, \boldsymbol{a} \in \boldsymbol{A}, \forall \boldsymbol{t} \in \boldsymbol{S} | \mathbf{row}(\boldsymbol{sa}) \neq \mathbf{row}(\mathbf{t})\}$

- Add sa to *S* and add all 'sa.w', $\forall$ w$\in$*W* to *S.A*.

Counter Example

- If an observation table is consistent and closed, corresponding DFA is defined.

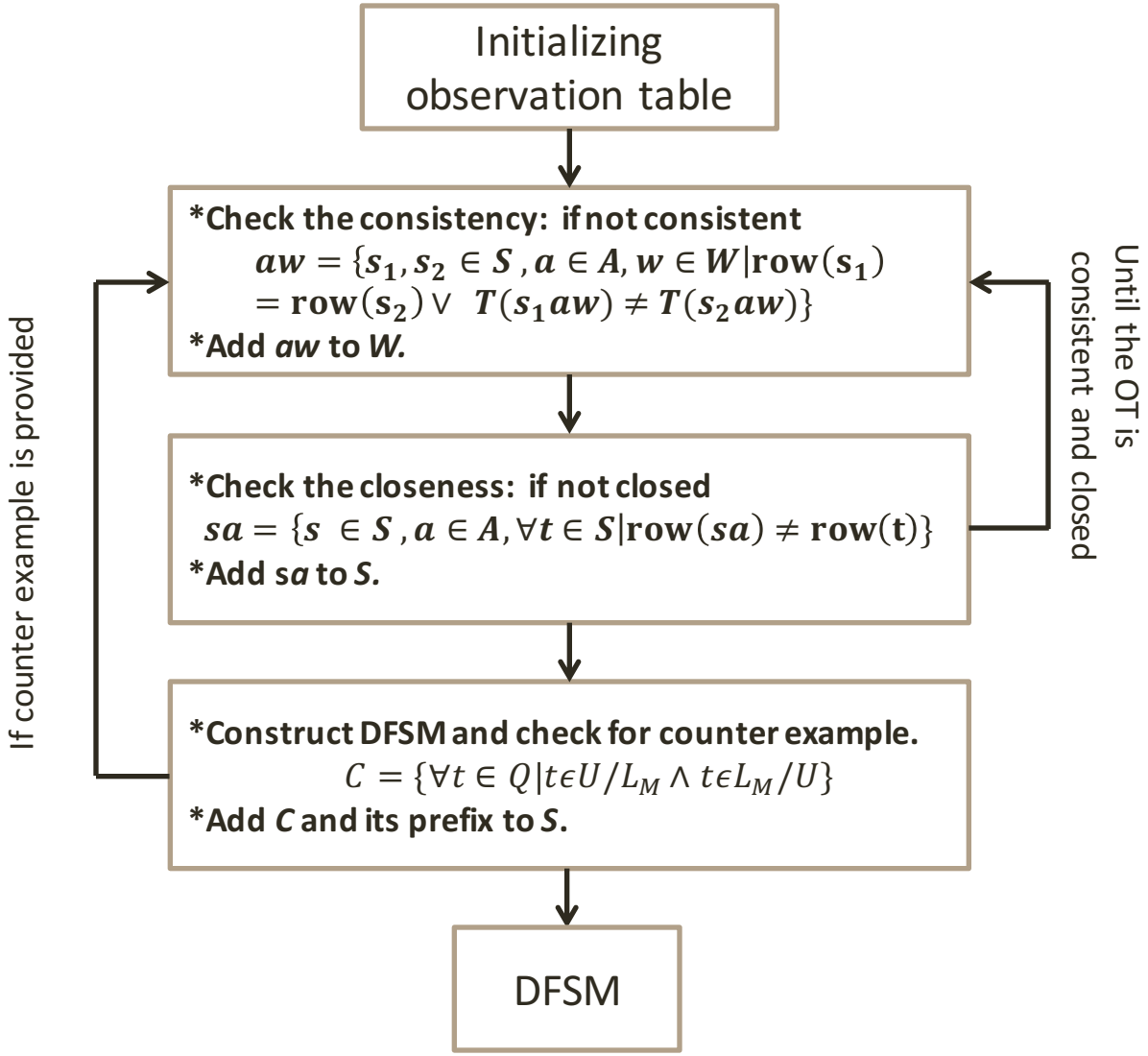
- $M(S,E,T)=(Q,\Sigma,\delta,q_0,F)$
  - $Q = \{row(s):s \in S\}$,
  - $q_0 = row(\mathcal{E})$,
  - $F = \{row(s) : s \in S \text{ and } T(s)=1\}$,
  - $\delta(row(s), a) = row(s.a)$.

- The teacher replies either with *match* or replies with a counter example *C*.

$$C = \{\forall t \in Q | t \epsilon U/L_M \wedge t \epsilon L_M/U\}$$

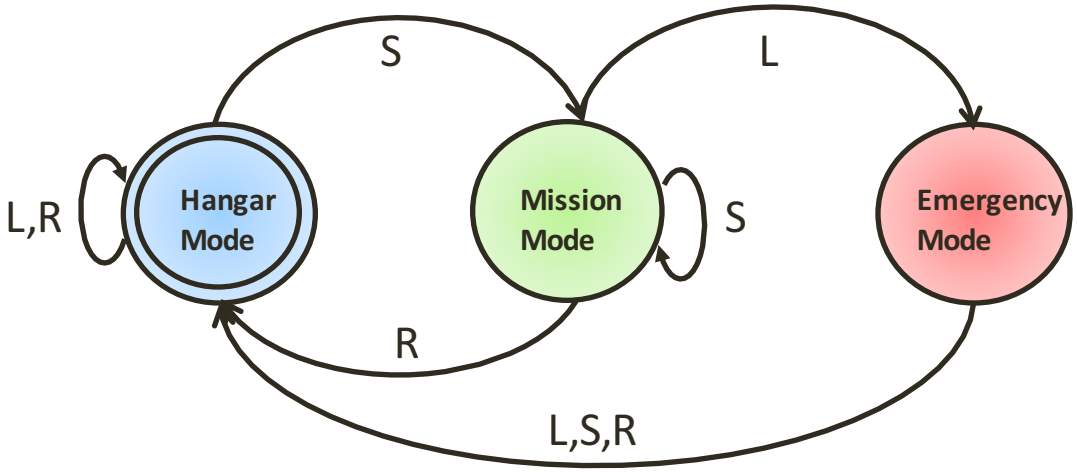
- Then *C* and all its prefixes are added to S.

L^star Algorithm



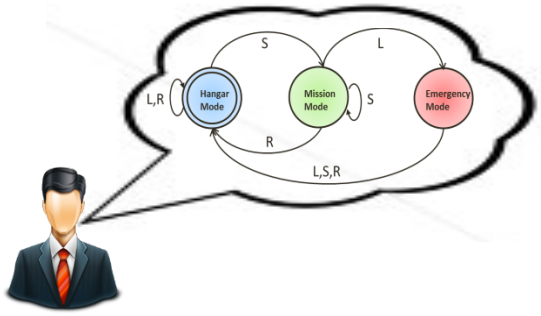
UAV- Example

- Consider the following example of UAV, Quadrator mission operation model. UAV receives commands to start its mission and return to hangar. But during the mission it may face with emergency condition and need to get back to hangar.
- Assuming we don't know the system, we will try to use Lstar learning to actively generate corresponding Automata.



S: Start mission
R: Return to hangar
L: Low fuel alarm

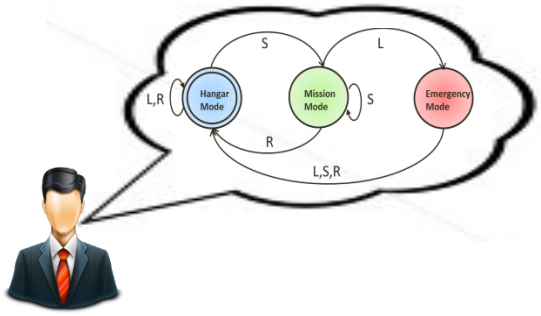
UAV, Example:



- Step 1:
  - *S*:
    - Starting with the initial observation table with empty string.
    - *S*={λ}
  - *S.A*:
    - Considering A={S, L, R} the *S.A* is :
    - *S.A*={S, L, R}
  - Analyze:
    - Table is not closed due to the row(S) from *S.A* which does not have similar row in *S*.
  - **Action:**
    - Adding row(S) to *S*

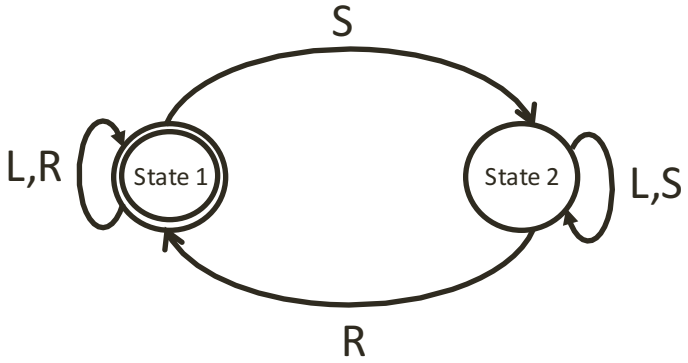
L* DFSM	λ
λ	1
S	0
L	1
R	1

UAV, Example:

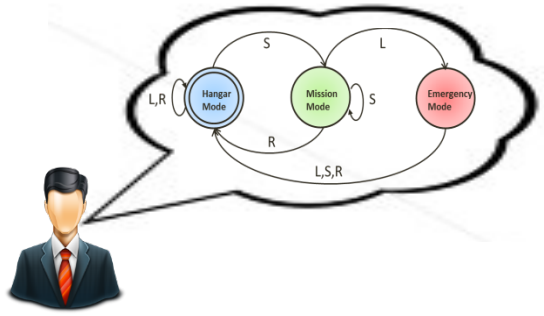


- Step 2:
  - *S*:
    - String S is added
    - *S*={λ, S}
  - *S.A*:
    - *S.A*={L, R, SL, SS, SR}
  - Analyze:
    - Table is now closed and consistent.
  - **Action:**
    - Generating the DFSM and asking for any existing counter example.
  - Counter example:
    - C={SLL}

L* DFSM	λ
λ	1
S	0
L	1
R	1
SL	0
SS	0
SR	1



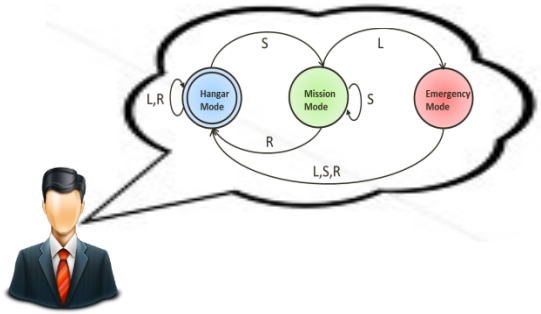
Ex. Cont.



- Step 3:
  - *S*:
    - String SLL and its prefix strings are added
    - *S*={λ, S, SL, SLL}
  - *S.A*:
    - *S.A*={L, R, SL, SS, SR, SLS, SLR, SLLL, SLLS, SLLR}
  - Analyze:
    - Table is not consistent :
      - row(S) = row(SL) but T(SL) ≠ T(SLL).
  - **Action:**
    - Adding L to event column and update the table.

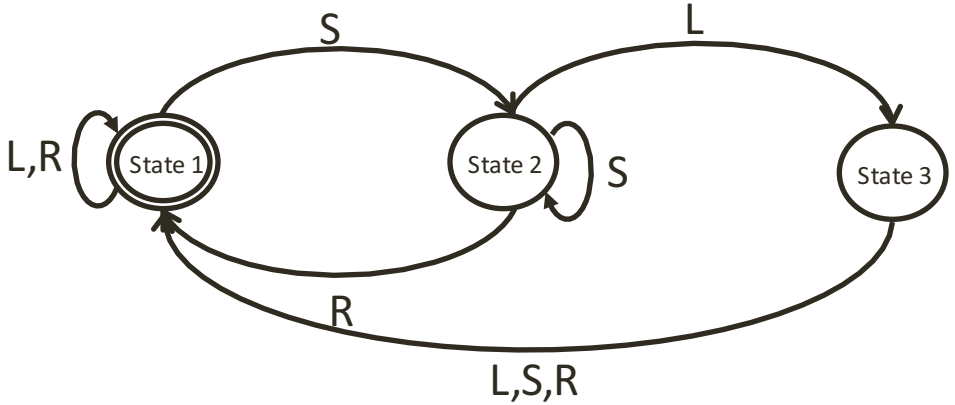
L* DFSM	λ
λ	1
S	0
SL	0
SLL	1
L	1
R	1
SS	0
SR	1
SLS	1
SLR	1
SLLL	1
SLLS	0
SLLR	1

Ex. Cont.



- Step 2:
  - *S*:
    - *No change.*
  - *S.A*:
    - *No change.*
  - Analyze:
    - Table is now closed and consistent.
  - **Action:**
    - Generating the DFSM and teacher approves the model, replies match.

L* DFSM	λ	L
λ	1	1
S	0	0
SL	0	1
SLL	1	1
L	1	1
R	1	1
SS	0	0
SR	1	1
SLS	1	1
SLR	1	1
SLLL	1	1
SLLS	0	0
SLLR	1	1



Theorems and Proofs

SECTION 4: L^{STAR} LEARNING FEATURES

L^{star} Learning Features

- Constructed DFA with L^{star} learning is **minimal**. Any other DFA consistent with T, would have more than or equal number of states.
- Any DFA consistent with T, with the same number of states is **isomorphic** with M.
- Total number of **operation** is at most *n-1*
  - since there is initially at least one value of row(s) and there cannot be more than *n*. (*n* is number of states)
- L^* **terminates** after making at most *n* conjectures and executing its main loop a total of at most *n - 1* times.

Angluin, Dana. "Learning regular sets from queries and counter examples. "Information *and computation* 75.2 (1987): 87-106.

Any other DFA consistent with T, would have more than or equal number of states.

- **Proof:**

- Proof is by contradiction, we assume that there exists a closed and consistent acceptor M' has less than or equal number of states

- We will show in order for M' to be consistent with T it must have at least ***n*** states

Proof Cont.

- As M' is consistent with T,

$$\{\forall s \in S.A \wedge e \in E |\, \delta'(q_0', S.e) \in F' \iff T(s.e) = 1\}$$

Where :

$$\delta'(q_0', s.e) = \delta'(\delta'(q_0', s), e)$$

- Also from M, it is consistent with T:

$$\{\forall s \in S.A \; and \; e \in E | T(s.e) = 1 \iff \delta(q_0, S.e) \in F\}$$

  - We can also say:

$$\delta(q_0, s.e) = \delta(\delta(q_0, s), e)$$

Proof Cont.

- Therefore:

$$Row\ (s) = row(\delta'(q_0', s))$$

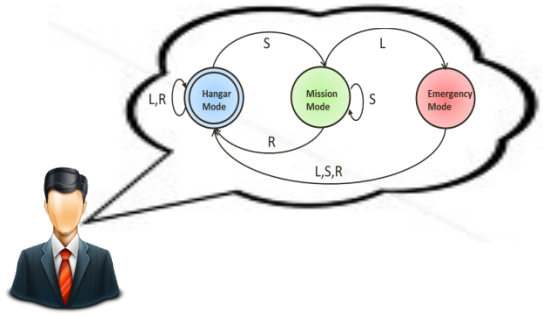
- As ***s*** **ranges over** **n** different states, there are ***n*** different row(s) and as result ***n*** different row($\delta'(q_0', s)$) in Q'. Which **contradicts** with the earlier assumption that M' has less than ***n*** states.

- **Conclusion:**
  - any M' consistent with T has at least ***n*** states.

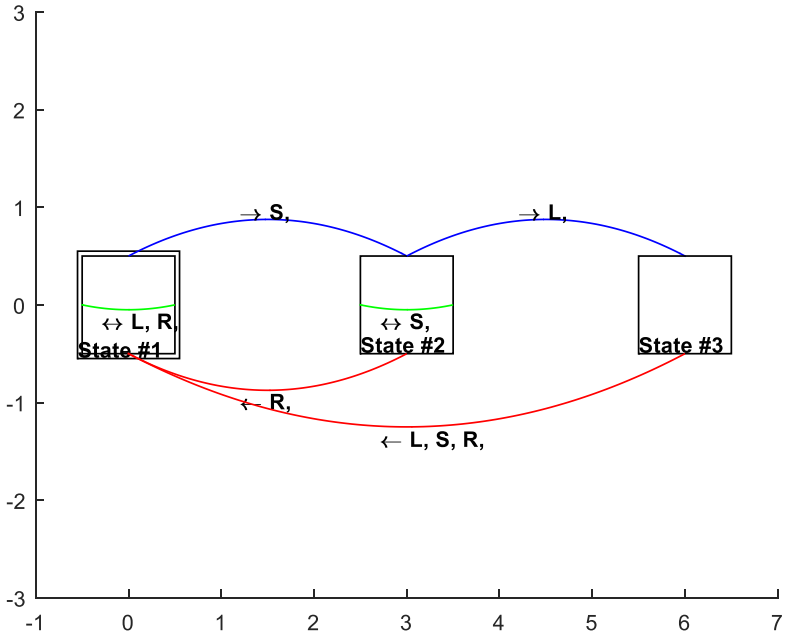
Theorems and Proofs

SECTION 4: L^{STAR} LEARNING TOOLBOX

Matlab Toolbox



- We developed a L^{star} learning Matlab toolbox.
- **UAV example**
  - We have used the developed toolbox and came up with DFSM of the UAV system.



Summary and Future Work

SECTION 5: CONCLUSION

Summary

- ✓ We introduced discrete event systems and their vast area of applications.
- ✓ We provided different aspect of modeling techniques used for identifying DES systems.
- ✓ We provided information about Automata representation
- ✓ We discussed an application example of Automata for an UAV system.
- ✓ We introduced Lstar Learning as an effective active learning approach for DES systems
- ✓ Lstar Matlab toolbox is provided and tested with UAV example.

Future Work

- **Developing a learning technique for diagnosis of DES.**
  - All of the existing techniques for fault diagnosis of DES assume perfect knowledge about the system.
  - We will try to develop a learning-based technique for diagnosis of unknown DES.

- **Developing learning technique for NFSM and corresponding fault diagnosing.**

References

- Angluin, Dana. "Learning regular sets from queries and counter examples."*Information and computation* 75.2 (1987): 87-106.
- Kumar, Ratnesh, and Vijay K. Garg. *Modeling and control of logical discrete event systems*. Vol. 300. Springer Science & Business Media, 2012.
- Karimoddini, Ali, et al. "Hybrid formation control of the unmanned aerial vehicles." *Mechatronics* 21.5 (2011): 886-898.
- Lohstroh, Marten, Lee, Edward A., “An Interface Theory for the Internet of Things.” 13th International Conference onSoftware Engineering and Formal Methods (SEFM), 2015.

Other References

- Cassandras, Christos G. *Introduction to discrete event systems*. Springer Science & Business Media, 2008.

- Lee, Edward Ashford, and Sanjit Arunkumar Seshia. *"Introduction to embedded systems: A cyber-physical systems approach*." Lee & Seshia, 2011.

- Nam, Wonhong, P. Madhusudan, and Rajeev Alur. "Automatic symbolic compositional verification by learning assumptions." *Formal Methods in System Design* 32.3 (2008): 207-234.

- Ipate, Florentin. "Learning finite cover automata from queries." *Journal of Computer and System Sciences* 78.1 (2012): 221-244.

ACCESS LAB

Autonomous Cooperative Control of Emergent Systems of Systems

TECHLAV

Thank you

Mohammad Mahdi Karimi

PhD. Candidate, ECE

Supervisor: Dr Ali Karimoddini

mmkarimi@aggies.ncat.edu