MIDES:A Tool for Supervisor Synthesis via Active Learning

Ashfaq Farooqui | Fredrik Hagebring | Martin Fabian

Department of Electrical Engineering,
Chalmers University of Technology,
Göteborg, Sweden

17th IEEE International Conference on Automation Science and Engineering
August 2021

Outline

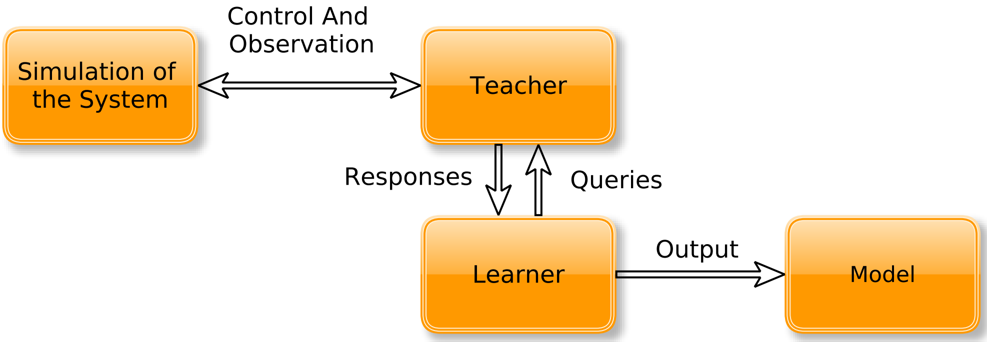
1. Problem Statement
2. Active Learning
3. MIDES
4. Learning from PLC
5. Modular Learning
6. Learning a model of LSM
7. Conclusion and Future Steps

Problem Statement

Supervisory Control Theory relies on access to models of plant and specifications to calculate supervisors.
Obtaining plant models is usually a bottleneck:

- Manually constructing them is hard, time-consuming, and error-prone.
- Manual construction of models is potentially intractable for large and complex systems.
- Once constructed, these models, need to be maintained to reflect the behavior of the real system.

Active Learning



MIDES: Model Inference for Discrete Event Systems[1]

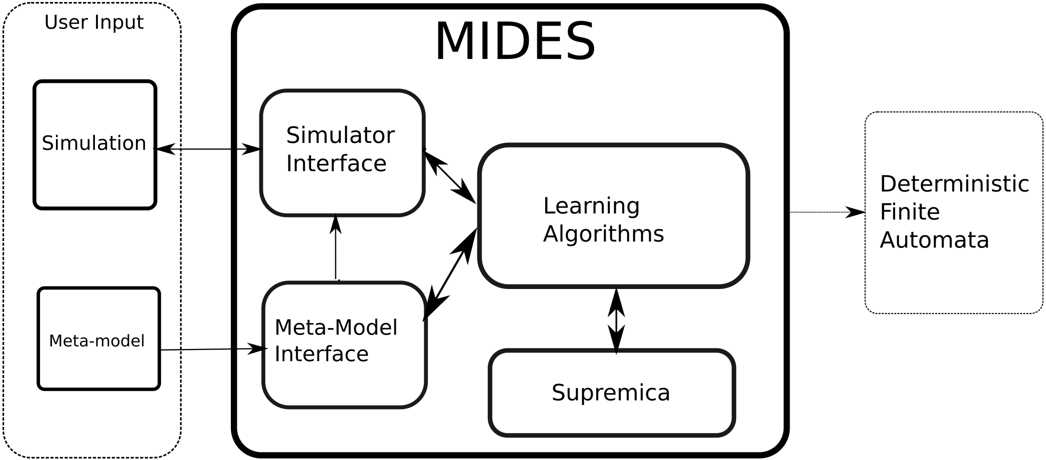
A opensource tool for automatic learning of models and supervisors for discrete event systems using active learning.

Aims

- Ability to prototype and test multiple learning algorithms quickly.
- Easy to interface a variety of external simulation tools.
- Generic output that can be used with existing tools within the supervisory context.

[1] `https://github.com/ashfaqfarooqui/MIDES`

MIDES: Model Inference for Discrete Event Systems



Algorithms

- L^* [a]
- SupL^* [b]
- Modular Plant Learner [c]
- Modular Supervisor Learner [d]

---

[a] Angluin, Dana. "Learning regular sets from queries and counterexamples." Information and computation 75.2 (1987): 87-106.

[b] Farooqui, Ashfaq, Ramon Claase, Martin Fabian, "On Plant-Free Active Learning of Supervisors" Submitted IEEE-TASE

[c] Farooqui, Ashfaq, Fredrik Hagebring, and Martin Fabian. "Active learning of modular plant models." IFAC-PapersOnLine 53.4 (2020): 296-302.

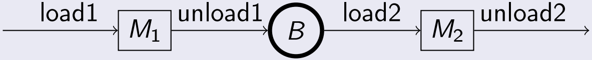
[d] Hagebring, Fredrik, Ashfaq Farooqui, and Martin Fabian. "Modular Supervisory Synthesis for Unknown Plant Models Using Active Learning." IFAC-PapersOnLine 53.4 (2020): 324-330.

Simulation Interfaces

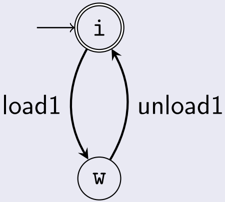
- Internal simulator using variables, predicates, and actions.
- Interface to MATAB engine to simulate MATLAB functions.
- OPC-UA based interface for PLC systems.

Machine Buffer Machine

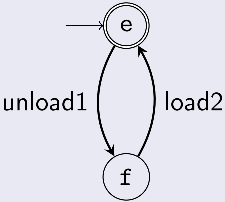
Product Flow



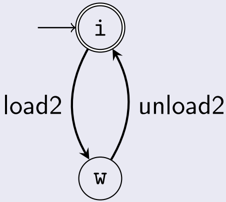
Behavior



(a) M_1

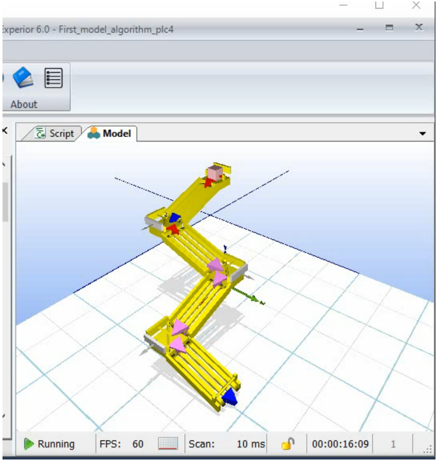


(b) B



(c) M_2

Simulation



Experior 6.0 - First_model_algorithm_plc4
About
Script
Model
Running
FPS: 60
Scan: 10 ms
00:00:16:09
1

Example learning from PLC

Learning Experior MBM

Learning outcome

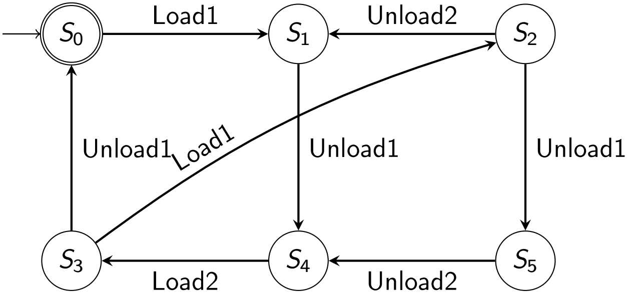
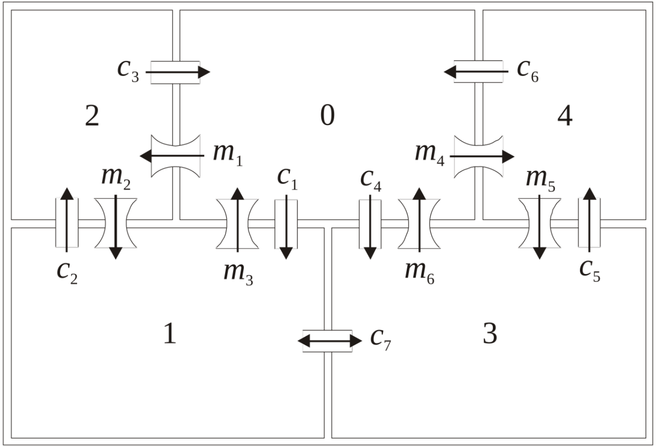


Figure: The maximally permissive controllable and non-blocking learnt supervisor

- Farooqui, Ashfaq, Ramon Claase, Martin Fabian, "On Plant-Free Active Learning of Supervisors" Submitted IEEE-TASE
- Farooqui, Ashfaq, and Martin Fabian. "Synthesis of supervisors for unknown plant models using active learning." 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). IEEE, 2019.

The Cat and Mouse example



Using the internal simulator

```
override val guards: Map[Command, Predicate] = Map(
    c1 -> EQ(cat, R0),
    c2 -> EQ(cat, R1),
    c3 -> EQ(cat, R2),
    c4 -> EQ(cat, R0),
    c5 -> EQ(cat, R3),
    c6 -> EQ(cat, R4),
    c7 -> OR(EQ(cat, R1), EQ(cat, R3)),
    m1 -> EQ(mouse, R0),
    m2 -> EQ(mouse, R2),
    m3 -> EQ(mouse, R1),
    m4 -> EQ(mouse, R0),
    m5 -> EQ(mouse, R4),
    m6 -> EQ(mouse, R3)
  )
```

Using the internal simulator

```
override val actions: Map[Command, List[Action]] = Map(
  c1 -> List(Assign(cat, R1)),
  c2 -> List(Assign(cat, R2)),
  c3 -> List(Assign(cat, R0)),
  c4 -> List(Assign(cat, R3)),
  c5 -> List(Assign(cat, R4)),
  c6 -> List(Assign(cat, R0)),
  c7 -> List(ToggleWithValues(cat, (R1, R3))),
  m1 -> List(Assign(mouse, R2)),
  m2 -> List(Assign(mouse, R1)),
  m3 -> List(Assign(mouse, R0)),
  m4 -> List(Assign(mouse, R4)),
  m5 -> List(Assign(mouse, R3)),
  m6 -> List(Assign(mouse, R0))
)
```

Cat and Mouse example

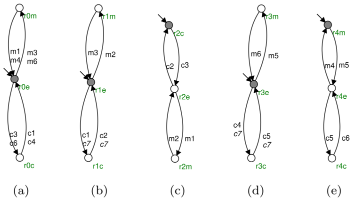
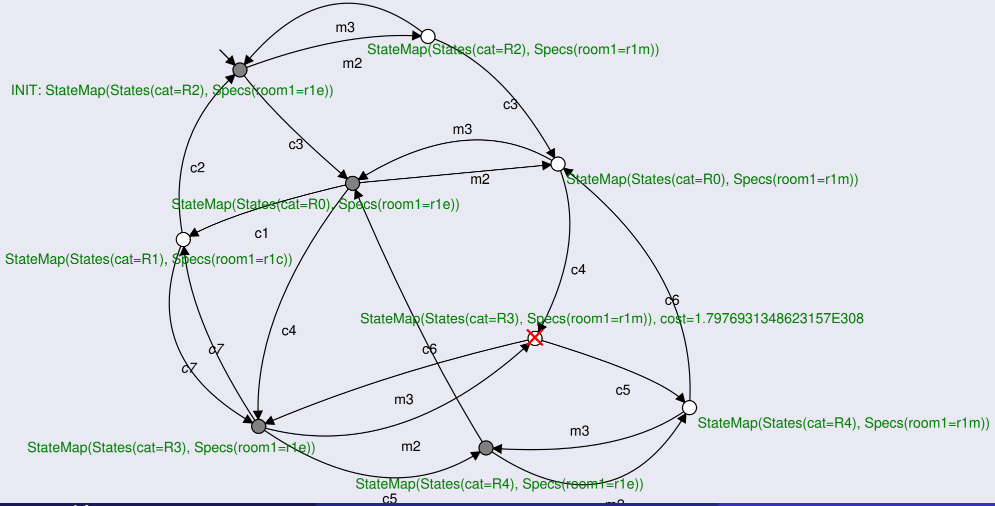


Fig. 2. Specifications for the different rooms ($Kr_0, Kr_1, Kr_2, Kr_3, Kr_4$, in that order) ensuring that only one of either the cat or the mouse can be present at a given time. Each state is identified using a unique name.

- $\Sigma_{K_1} = \{c_1, c_3, c_4, c_6, m_1, m_3, m_4, m_6\}$,
- $\Sigma_{K_1} = \{c_1, c_2, c_7, m_2, m_3\}$,
- $\Sigma_{K_2} = \{c_2, c_3, m_1, m_2\}$,
- $\Sigma_{K_3} = \{c_4, c_5, c_7, m_5, m_6\}$,
- $\Sigma_{K_4} = \{c_5, c_6, m_4, m_5\}$,

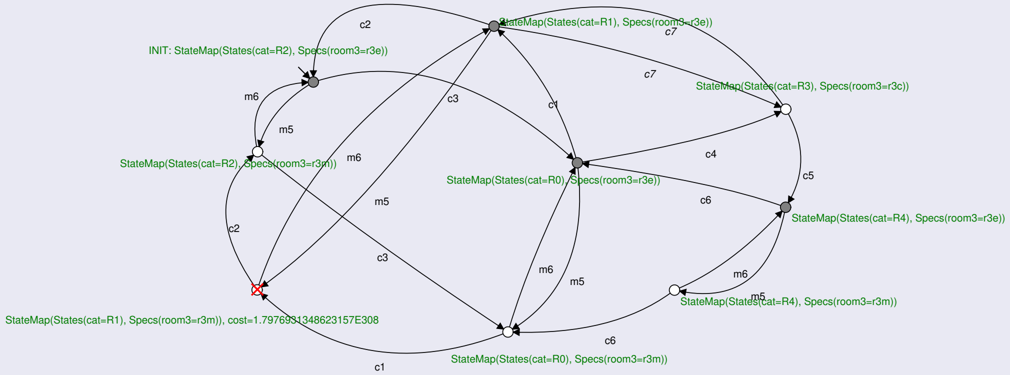
Cat and Mouse example

room 1 (M_{K1})



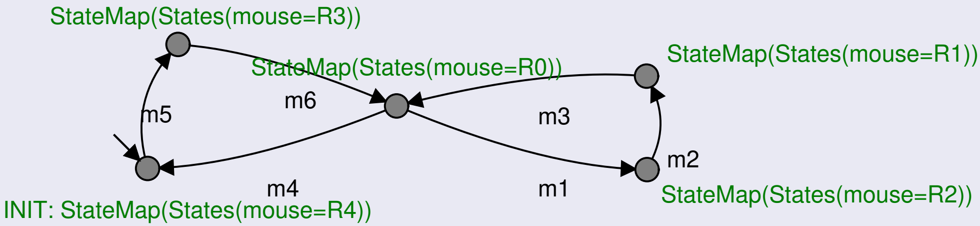
Cat and Mouse example

room 3 (M_{K3})



Cat and Mouse example

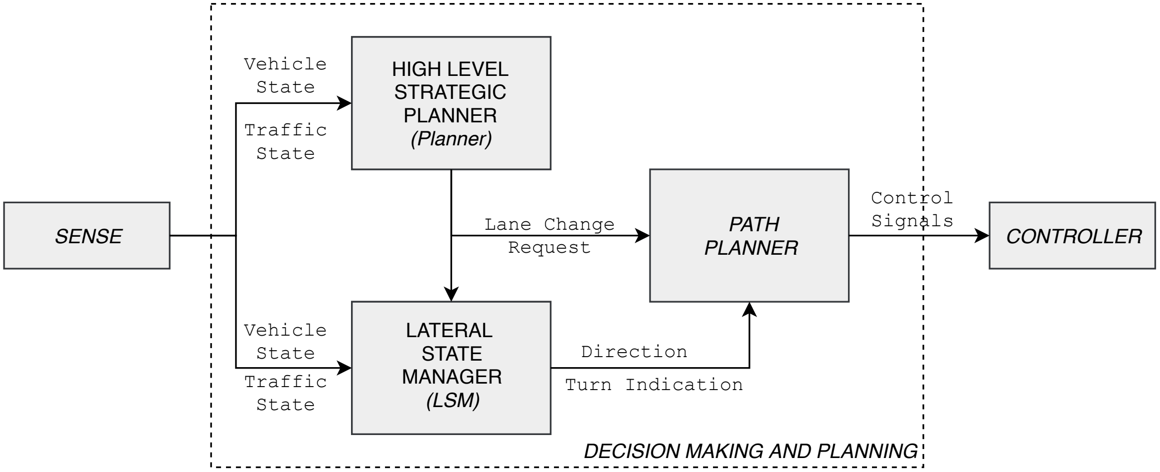
Mouse



The resulting supervisors (and plants) can then be used in existing tools to generate a maximally permissive controllable and non-blocking supervisor.

- Hagebring, Fredrik, Ashfaq Farooqui, and Martin Fabian. "Modular Supervisory Synthesis for Unknown Plant Models Using Active Learning." IFAC-PapersOnLine 53.4 (2020): 324-330.
- Farooqui, Ashfaq, Fredrik Hagebring, and Martin Fabian. "Active learning of modular plant models." IFAC-PapersOnLine 53.4 (2020): 296-302.

The LSM



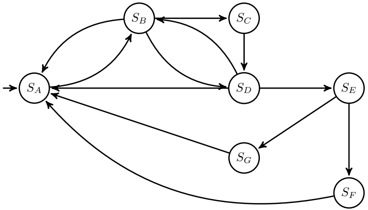
SENSE
Vehicle
State
Traffic
State
HIGH LEVEL
STRATEGIC
PLANNER
(Planner)
Vehicle
State
Traffic
State
LATERAL
STATE
MANAGER
(LSM)
Lane Change
Request
PATH
PLANNER
Direction
Turn Indication
Control
Signals
CONTROLLER
DECISION MAKING AND PLANNING

Learning models of MATLAB code[2]

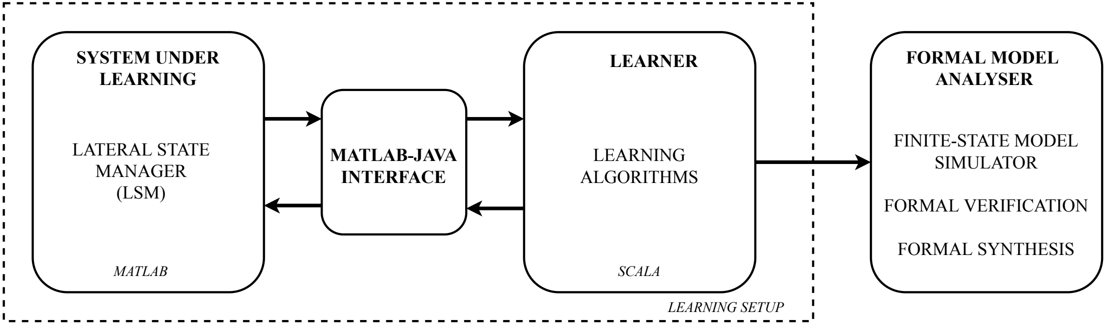
```
function duringStateA(self,
                      laneChangeRequest)
    var1 = function1();
    var2 = function2(laneChangeRequest);
    if var1 && var2
        self.state = stateB;
    end
end
```

[2] Selvaraj, Yuvaraj, et al. "Automatically learning formal models: an industrial case from autonomous driving development." Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. 2020.

- LSM is programmed in MATLAB
- Inputs from other sub-components are abstracted
- Functionality to run and observe LSM by the learner is introduced



Learning Setup



- The interface provides a standard API to run and observe the SUL
- Provide information to the learner to interpret the observed information

Outcome

- MPL manages to learn a model that was validated with good confidence
- The obtained model was simulated alongside the original code to check correctness

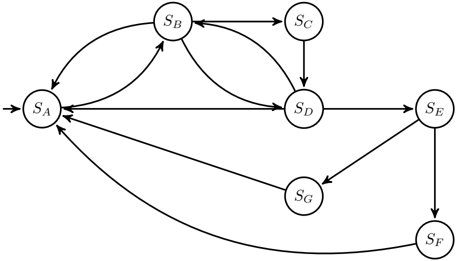


Figure: A meta-level finite-state abstraction of the LSM

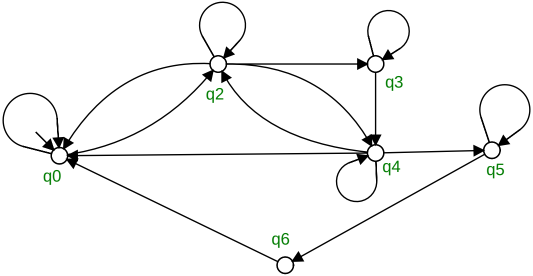


Figure: Language Minimised model

Conclusions and Future Steps

- MIDES, a tool for automatically learning supervisors in the absence of traditional plant models.
- Support for interfacing external simulation environment, eg, MATLAB, OPC-UA.

Future steps

- Improving algorithmic performance using better data-structures.
- Support for richer formalisms – Extended Finite Automata, Register Automata, Etc
- Automata simulators

Thank You!