Synthesis of Supervisors for Unknown Plant Models Using Active Learning

Ashfaq Farooqui and Martin Fabian

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

15th IEEE International Conference on Automation Science and Engineering, Vancouver, Canada
August 2019

Outline

Topic

Problem Statement

Given a simulation of the plant, its corresponding specifications and the set of events, synthesize a maximally permissive controllable and non-blocking supervisor that can be used to supervise the plant while satisfying the given specification.

Motivation

- Lack of accurate plant models.
- The plant and specifications constantly updated.
- Easier access and availability of simulations and computing power.

Topic

Automata Learning

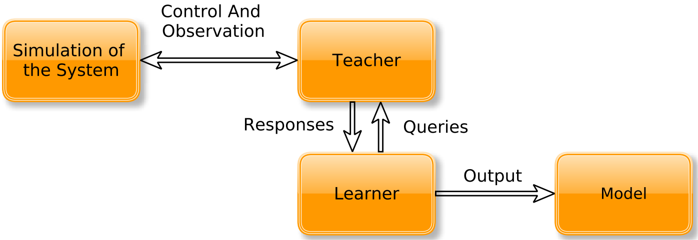
- Passive Learning
- Active Learning

Active Learning

Learning regular sets from queries and counterexamples. Dana Angluin. Information and Computation, 1987

- Famously called L^*
- L^* makes it possible to learn deterministic automata

Active Learning



L*

Learner Queries

- Membership queries $w \in \mathcal{L}_m$?
- Equivalence queries $\mathcal{L}(H) = L$?

Observation Table

Representation of the current knowledge about the target system.

- Contains two sets $S, E \subseteq \Sigma^*$
- $row : S \cup S.\Sigma \rightarrow E \rightarrow \{1, 0\}$
- $cell(se) = 1 \leftrightarrow se \in \mathcal{L}, | \, s \in S \text{ and } e \in E$

		E		
		τ	a	aa
S	τ	0	0	1
$S.\Sigma$	a	0	1	0
	b	0	0	0

Topic

Membership Query

Plant query

$$\lambda_G(s) = \begin{cases} 2, & s \in L_m(G) \\ 1, & s \in L(G) - L_m(G) \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Specification query

$$\lambda_K(s) = \begin{cases} 2, & s \in L_m(K) \\ 1, & s \in L(K) - L_m(K) \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Membership Query

Controllability query

$$\lambda_C(s) = \begin{cases} 1, & (\forall s' \in \overline{s}, \sigma \in \Sigma_u^*)\ \lambda_K(s') \neq 0 \text{ and} \\ & \lambda_G(s'.\sigma) \neq 0 \Rightarrow \lambda_K(s'.\sigma) \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

This is the standard controllability discussed by Ramadge and Wonham.

Membership Query

Putting them together

$$T(s) = \begin{cases} \min(\lambda_G(s), \lambda_K(s)), & \lambda_C(s) = 1 \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Observation table

The observation table can now contain three values: {0,1,2}.

Handling the Controllability Problem

- Convert the controllability problem into a potential blocking problem.
- In doing so, we will check controllability over Σ_u instead of ${\Sigma_u}^*$

Σ_u-Saturated Specifications[1]

- Add a transition from every state, for every uncontrollable transition not enabled in that state, to a "dump" state – $\perp$.

$$\delta^{\perp} = \delta \cup \{(q, u, \perp) \mid q \in Q, u \in \Sigma_u, \delta(q, u) \text{ is undefined}\}$$

1. Flordal et. al, "Compositional synthesis of maximally permissive supervisors using supervision equivalence," Discrete Event Dynamic Systems.

Equivalence Query

W-Method[2]

Generate test strings according to $P \cdot U \cdot W$ where

- $U = (\Sigma^0 \cup \Sigma \cup \Sigma^2 \cup \ldots . \Sigma^{n-m})$.
- P = Set of all strings that lead to all states
- W = Set of strings that distinguish each pair of states
- m = size of the hypothesis

Size estimate

An estimate of the maximum size of the supervisor – n

1. T. Chow, "Testing software design modeled by finite-state machines," IEEE Trans. on Software Engineering, 1978.

Equivalence Query

Equivalence test

$$R(s) = \begin{cases} 2, & s \in L_m(H) \\ 1, & s \in L(H) - L_m(H) \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Counter example

CE= $\{s | T(s) \neq \text{R(s)}, \forall \text{ s} \in \text{P.U.W}\}$

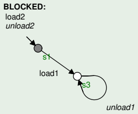
Obtaining the Supervisor

Automaton

$$
\begin{aligned}
Q &= \{row(s) \mid s \in S \text{ and } \; cell(s\tau) > 0\} \\
q_0 &= row(\tau) \text{ if } cell(\tau) > 0 \\
\delta(row(s), \sigma) &= \begin{cases} row(s\sigma), & cell(s\sigma) > 0 \\ \text{undefined}, & \text{otherwise.} \end{cases} \\
Q_m &= \{row(s) \mid s \in S \text{ and } cell(s\tau) = 2\}
\end{aligned} \tag{6}
$$

Example (Automaton)

T	τ
τ	2
τ.load1	1
τ.unload1	0
$\tau.\tau$	2
τ.load2	0
τ.unload2	0
τ.load1.τ	1
τ.load1.load2	0
τ.load1.load1	0
τ.unload1.τ	0
τ.unload1.load1	0
τ.load1.unload1	1
τ.load1.unload2	0
τ.unload1.load2	0
τ.unload1.unload1	0
τ.unload1.unload2	0



Obtaining the Supervisor

Supervisor

- If the automaton obtained is non-blocking then this automaton is the maximally permissive controllable and non-blocking supervisor.
- If the automaton is blocking, the maximally permissive controllable and non-blocking supervisor can be obtained by running existing synthesis algorithms.

Topic

Reflections

- It was possible to learn a maximally permissive supervisor for a given system when plant models are absent.
- The resulting supervisor is monolithic and hence, the current approach cannot be scaled to larger systems.
- The number of states in the learning process is increased due to the use of Σ_u-saturated specifications.
- The W-method requires as input the number of states in the target supervisor, this is not always known in advance.

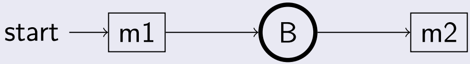
Future Work

- Learn controllable supervisors from the start.
- Move towards a modular approach of learning.
- Optimize learning by getting rid of unwanted queries and more efficient data structures to represent the observations.

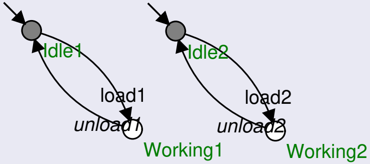
Thank You!

The Plant and its Simulation

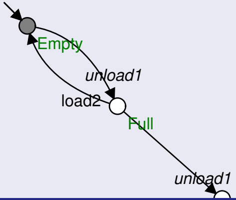
Two Machines and a Buffer



Plant Components

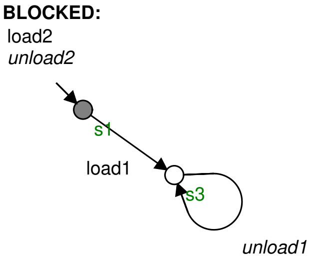


Σ_u-saturated Specification



First Iteration

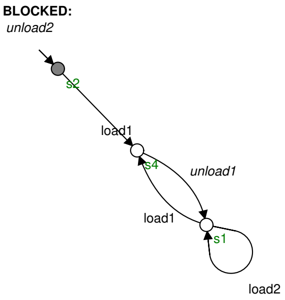
T	τ
τ	2
τ.load1	1
τ.unload1	0
$\tau.\tau$	2
τ.load2	0
τ.unload2	0
τ.load1.τ	1
τ.load1.load2	0
τ.load1.load1	0
τ.unload1.τ	0
τ.unload1.load1	0
τ.load1.unload1	1
τ.load1.unload2	0
τ.unload1.load2	0
τ.unload1.unload1	0
τ.unload1.unload2	0



Counter Example: load1.unload1.unload1

Second Iteration

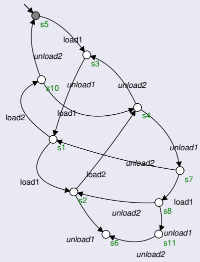
T	τ	load1.τ
τ	2	1
τ.load1	1	0
τ.unload1	0	0
τ.load1.unload1	1	1
τ.load1.unload1.unload1	0	0
τ.load1.unload1.unload1.τ	0	0
$\tau.\tau$	2	1
τ.load2	0	0
τ.unload2	0	0
τ.load1.τ	1	0
τ.load1.load2	0	0
τ.load1.load1	0	0
τ.unload1.τ	0	0
τ.unload1.load1	0	0
τ.load1.unload2	0	0
τ.unload1.load2	0	0
τ.unload1.unload1	0	0
τ.unload1.unload2	0	0
τ.load1.unload1.τ	1	1
τ.load1.unload1.load1	1	0
τ.load1.unload1.load2	1	1
τ.load1.unload1.unload2	0	0
τ.load1.unload1.unload1.load2	0	0
τ.load1.unload1.unload1.load1	0	0
τ.load1.unload1.unload1.unload1	0	0
τ.load1.unload1.unload1.$\tau.\tau$	0	0
τ.load1.unload1.unload1.unload2	0	0
τ.load1.unload1.unload1.τ.load1	0	0
τ.load1.unload1.unload1.τ.load2	0	0
τ.load1.unload1.unload1.τ.unload2	0	0
τ.load1.unload1.unload1.τ.unload1	0	0



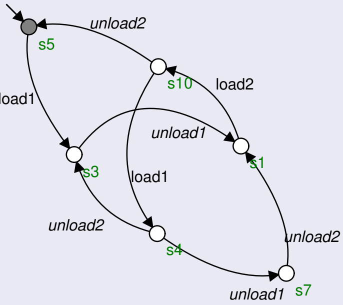
Counter Example: load1.unload1.load1.load2

Resulting Supervisor

Blocking Supervisor



Non-blocking Supervisor



Closed

An observation table is said to be closed if for all $t \in S, a \in \Sigma$ there is an $s \in S$ such that the $row(s) = row(t.a)$.

Consistent

A table is consistent if for $s_1 \in S$ and $s_2 \in S$ and $row(s_1) = row(s_2)$ then for all $a \in \Sigma$, $row(s_1.a) = row(s_2.a)$.