Title:	Learning Automata of Black Box Software and Hardware via Side Channel Analysis
Author:	Johnathan DiMatteo
Supervisor:	Sebastian Fischmeister

Motivation

The goal of this paper is to enforce security and safety in these devices through reverse engineering, or *Model Learning*: automatic algorithms that "determine in which global states the device can be and which state transitions and outputs occur in response to which inputs" [1]. Our proposed method is to use *Side Channel Analysis* to make learning more "black box". One application of this technology is to verify software and hardware systems conform to design specifications. Another is to perform this on a live system to see if it has been compromised (i.e. an attacker downloads new firmware onto the device), thus alerting users.

Background

State diagrams are a way to model a system concerning a finite number of states, inputs and outputs. **Inputs** are external stimuli that interact with the system. **Outputs** are a system's response to an input. A **state** determines the behavior of a system. For example, a coffee machine can have three states, "Off", "Standby", and "Ready". The coffee maker will only pour coffee after the press of Button B in the "Ready" state, will be turned off in the "Off" state, and will wait for the user to load a coffee pod in the "Standby" state. A state is bound to a set of rules and functions (i.e. it is impossible to pour coffee in the "Off" state). State diagrams are common in software validation. We will be focusing on a specific type of state diagram model known as a Mealy Machine.

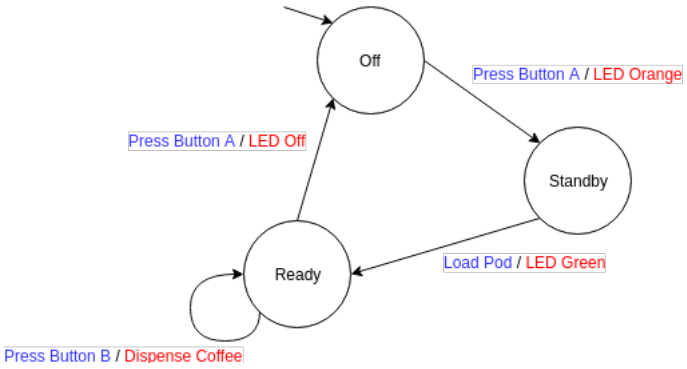
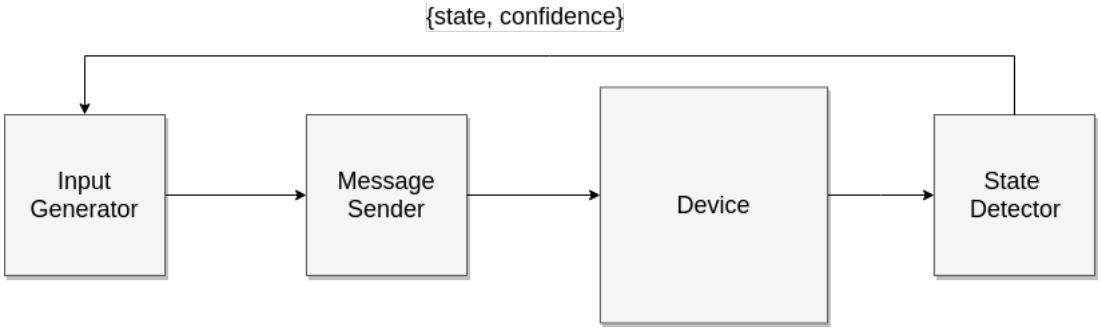


Figure 1: Example of a simple Mealy Machine for a coffee maker.

Problem Statement

1. **Input Generator**: Given the current predicted state (**output? - ask Sebastian**) and history of previous states (**outputs?**) and queries, determine the next query that is most likely to induce a state change in the system under observation.



2. **State Detector**: Given a response from the system under observation and history of previous responses, estimate the current state (**output? - ask Sebastian**).

3. **Message Sender**: Given a query, translate the query into a valid message accepted by the system.

References

[1] Frits Vaandrager. Model learning. *Communications of the ACM*, 60(2):86–95, 2017.