DOI:10.1145/2967606

Model learning emerges as an effective method for black-box state machine models of hardware and software components.

BY FRITS VAANDRAGER

Model Learning

WE ROUTINELY MANAGE to learn the behavior of a device or computer program by just pressing buttons and observing the resulting behavior. Especially children are very good at this and know exactly how to use a smartphone or microwave oven without ever consulting a manual. In such situations, we construct a mental model or state diagram of the device: through experiments we determine in which global states the device can be and which state transitions and outputs occur in response to which inputs. This article is about the design and application of algorithms that perform this task automatically.

There are numerous approaches where models of software components are inferred through analysis of the code, mining of system logs, or by performing tests. Many different types of models are inferred, for example, hidden Markov models, relations between variables, and class diagrams. In this article, we focus on one specific type of models, namely *state diagrams*, which are crucial for understanding the behavior of many software systems, such as (security and network) protocols and embedded control software. Model inference techniques can be either white box or black box, depending on whether they need access to the code. In this article, we discuss *black box* techniques. Advantages of these techniques are that they are relatively easy to use and can also be applied in situations where we do not have access to the code or to adequate white box tools. As a final restriction, we only consider techniques for *active learning*, that is, techniques that accomplish their task by actively doing experiments (tests) on the software. There is also an extensive body of work on passive learning, where models are constructed from (sets of) runs of the software. An advantage of active learning is that it provides models of the full behavior of a software component, and not just of the specific runs that have occurred during actual operation.

The fundamental problem of active, black-box learning of state diagrams (or automata) has been studied for decades. In 1956, Moore[31] first

» key insights

- **Model learning aims to construct black-box state diagram models of software and hardware systems by providing inputs and observing outputs. The design of algorithms for model learning constitutes a fundamental research problem.**
- **Recently, much progress has been made in the design of new algorithms, both in a setting of finite state diagrams (Mealy machines) and in richer settings with data (register automata). Through the use of abstraction techniques, these algorithms can be applied to complex systems.**
- **Model learning is emerging as a highly effective bug-finding technique, with applications in areas such as banking cards, network protocols, and legacy software.**