Graphical Animations of State Machines

Nguyen Thi Thanh Tam (tamnguyen@jaist.ac.jp)
School of Information Science,
Japan Advanced Institute of Science and Technology

2017

Keywords: DRAW-SVG, graphical animations, lemmas, Maude, state machines, state patterns.

Our society crucially depends on software. How much we rely on software will be surely getting larger and larger. The quality of software must affect the quality of our society. Therefore, we need to have reliable technologies to make software truly reliable. Systems verification with interactive theorem proving (ITP) is a promising technology that could make software reliable, although it is necessary to utilize many other technologies, such as testing, so as to make software really reliable. Various kinds of systems can be formalized as state machines. A state machine $M \triangleq \langle S, I, T \rangle$ consists of a set S of states including the set I of initial states and a binary relation $T \subseteq S \times S$ over states. An element $(s, s') \in T$ is called a state transition of M. The set R_M of the reachable states of M is inductively defined as follows: $I \subseteq R_M$ and if $s \in R_M$ and $(s, s') \in T$, then $s' \in R_M$. A state predicate p is called an invariant of M if and only if $(\forall s \in R_M)\, p(s)$. Many requirements of software can be formalized as invariants. Since verifications of other classes of properties often require invariants as lemmas, invariants are the most fundamental class of properties of state machines.

Lemma conjecture is one of the most intellectual activities in ITP. Accordingly, many researches have been conducted, trying to come up with how to conjecture lemmas. None of them, however, is good enough. Thus, we need to make further efforts to come up with a better way to do so. While we were performing systems verification with ITP, we happened to find out some state patterns in which the reachable states of a state machine are classified and conjectured several useful lemmas from the state patterns to complete the formal verification. It would be very useful to make it possible to obtain such state patterns of a given state machine with a reasonable amount of efforts. This research utilizes human beings' ability to recognize patterns in various kinds of data, such as graphical animations.

On the other hand, a counterexample generated by a model checker, such as Maude LTL model checker, consists of a sequence $s_0; \ldots; s_m$ of states and a loop $(s_{m+1}; \ldots; s_n)^\infty$

of states such that s_{m+1} is a successor state of s_m and s_n. A counterexample generated by the Maude LTL model checker is not necessarily the shortest one. The shorter a counterexample, the easier it is to comprehend the counterexample. Therefore, we have implemented a meta-program in Maude that takes a counterexample and generates a shorter one. We realized counterexamples generated by the Maude LTL model checker can be also animated graphically so that human users could comprehend them better. Since the loop is repeatedly played, the repetition could help human users realize what happen in the loop. It would be preferable to graphically animate a counterexample so that human users could comprehend it better. Thus, we can also use the state machine graphical animation tool help human users comprehend a counterexample. The tool is extended such that it takes a counterexample and plays its animation.

Thus, the research aims at designing and implementing a state machine graphical animation tool to support users recognize patterns to be used for conjecturing lemmas in ITP, and comprehend counterexamples better.