An Investigation of Machine Learning and a consideration on its application to Theorem Proving

Ho, Dung Tuan (1310065)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 10, 2016

Keywords: Machine Learning, Interactive Theorem Proving, CafeOBJ, Maude, Lemma Conjecturing.

Systems Verification logically checks if systems satisfy desired properties to make them reliable. The techniques used are largely classified into *Model Checking* and *(Interactive) Theorem Proving*. This project focuses on *Interactive Theorem Proving (ITP)* that often requires *eureka* steps. One typical eureka step is to discover a non-trivial lemma, called *Lemma Conjecturing*. Some techniques have been proposed such that lemmas can be systematically or automatically discovered, and successfully applied to some specific applications. None of those techniques, however, can discover all lemmas for all possible proof problems (not only mathematical but also engineering ones, i.e., systems verification. In general, it is necessary to understand proof targets profoundly to some extent to discovery non-trivial lemmas. Proof targets are systems and/or system behaviours in Systems Verification. Human users often rely on some information to conjecture such lemmas. This information characterises some important aspects of reachable states of systems. But, it is time-consuming to extract the information from a large amount of reachable states. We predict that *Machine Learning* can help to do so since the technique can be applied to big data. In ITP, Machine Learning may extract some patterns from a large number of reachable states. The patterns expresses some characteristics of the reachable states such that they can help human users to get better understanding of not only the reachable states but also the system's state machine. Then, the understanding can hopefully leads the human users to conjecture some non-trivial lemmas for some specific proof problems. The aim of the project is to learn some advanced techniques of machine learning, confirming that the technique can be used to extract useful information from reachable states of systems such that the information helps human users to conjecture non-trivial lemmas.

A *state machine* M consists of a set S of states that includes the initial states I and a binary relation T over states. $(s, s') \in T$ is called a *transition*. *Reachable states* R_M of M are 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 distributed system DS can be formalized as M and many desired properties of DS can be expressed as invariants of M. An *invariant* of M is a state predicate p of M such that p holds for all $s \in R_M$. To prove that p is an invariant of M, it suffices to find an

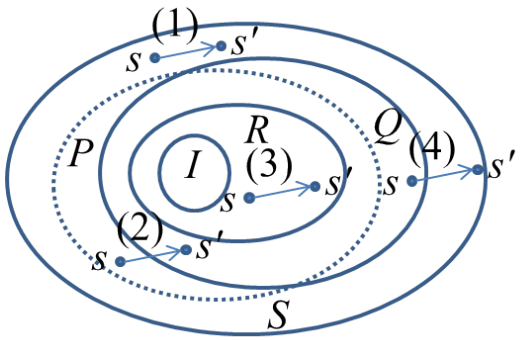

Figure 1: Some possible situations when proving that p is an invariant of M

inductive invariant q of M such that $q(s) \Rightarrow p(s)$ for each $s \in S$. An *inductive invariant* q of M is a state predicate of M such that $(\forall s_0 \in I)\, q(s_0)$ and $(\forall (s, s') \in T)\, (q(s) \Rightarrow q(s'))$. Note that an inductive invariant of M is an invariant of M but not vice versa.

Finding an inductive invariant q (or conjecturing a lemma q) is one of the most intellectual activities in ITP[2]. This activity requires human users to profoundly understand the system under verification or M formalizing the system to some extent. The users must rely on some reliable sources that let them get better understandings of the system and/or M to conduct the non-trivial task, namely *lemma conjecture*. For this end, our experiences on ITP tell us that it is useful to get better understandings of R_M. Some characteristics of R_M can be used to systematically construct a state predicate q_i that is a part of q.

Let P and Q be the sets of states that correspond to predicates p and q, respectively. S, I, R_M, P and Q can be depicted as shown in Fig. **??**. Proving that p is an invariant of M is the same as proving $R \subseteq P$. Let $(s, s') \in T$ be an arbitrary transition. In each induction case or a subcase of each induction case, all needed is basically to show $p(s) \Rightarrow p(s')$ so as to prove that p is an invariant of M. There are four possible situations: (1) s, $s' \notin P$, (2) $s \notin P$ and $s' \in P$, (3) $s, s' \in P$, and (4) $s \in P$ and $s' \notin P$. $p(s) \Rightarrow p(s')$ holds for (1), (2) and (3), but does not for (4). To complete the proof that p is an invariant of M, we need to know $s' \notin R_M$ for (4), namely that s' is not reachable for (4). To this end, we need to conjecture a lemma q such that q does not hold for s'. The systematic way to conjecture lemmas may not work for a complex system because case analysis may have to be repeated too many times until what to show reduces either true or false. Even if we reach the case in which what to show reduces false, a lemma conjectured could be so long that we may find it trouble to prove that the lemma is an invariant of M. Our experiences on ITP tell us that better understandings of M and/or how M behaves let us conjecture useful lemmas to complete the proof concerned. Moreover, the properties we are interested in are invariants. Therefore, it suffices to get better understandings of R_M. In general, R_M contains an infinite number of states in which each system state $s \in S$ is characterized by some values that are called *observable values*. In practical, the characteristics of R_M are correlations among observable values of the elements of R_M. Generally, the number

[2] q may be in the form $q_1 \wedge \ldots \wedge q_n$. Each q_i may be called a lemma and is an invariant of M if q is an inductive invariant of M, although q_i may not be an inductive invariant of M.

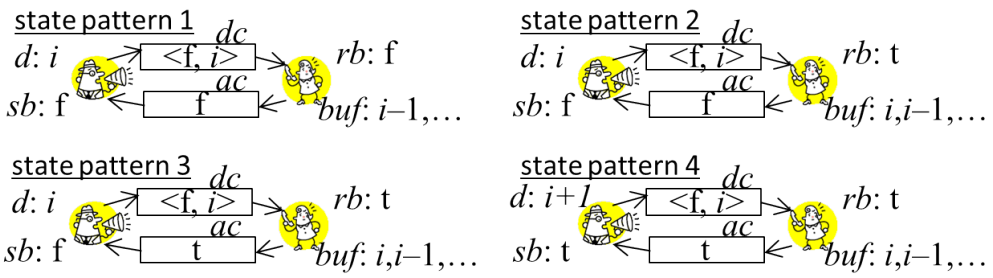

Figure 2: Four state patterns of M_{SCP}

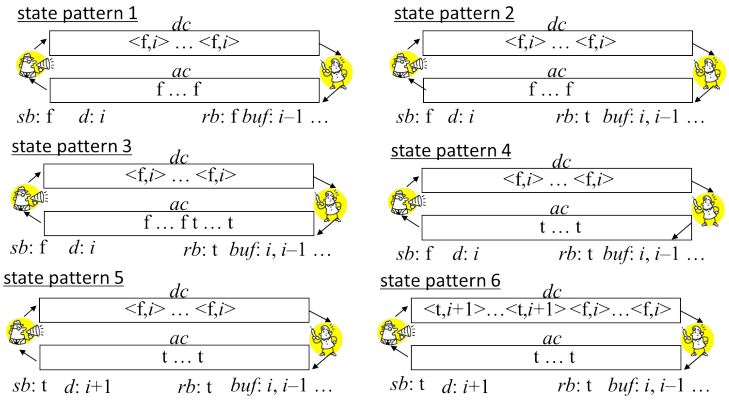

Figure 3: Six state patterns of M_ABP

of the elements of R_M is unbounded and then a huge number of reachable states are generated from M. The task of extracting correlations among a huge number of data (reachable states in our case) is the role of Machine Learning (ML). However, classical machine-learning techniques only work for a database whose elements are expressed in propositional form, while our database consists of system states expressed in first-order form. There is the ML technique that can deal with first-order forms: Inductive Logic Programming (ILP). This is why we use ILP.

We have conducted two case studies on Alternating Bit Protocol (ABP, a simplified version of Sliding Window Protocol used in TCP) and Simple Communication Protocol (SCP, a simplified version of ABP) using a framework in which Progol, an ILP system, has been mainly used to extract some characteristics of the reachable states of their state machines formalizing the protocols. We had conducted verification case studies in which it is proved that both protocols enjoy what is called the reliable communication property that whenever the current data to be delivered is i, the data upto i or $i-1$ have been successfully delivered to the receiver from the sender without any duplications nor drops. Through those verification case studies, we drew four possible reachable state patterns (see Fig. **??**) for SCP and six possible reachable state patterns (see Fig. **??**) for ABP. Those state patterns can be used as oracles for judging if learned hypotheses are reasonably good.