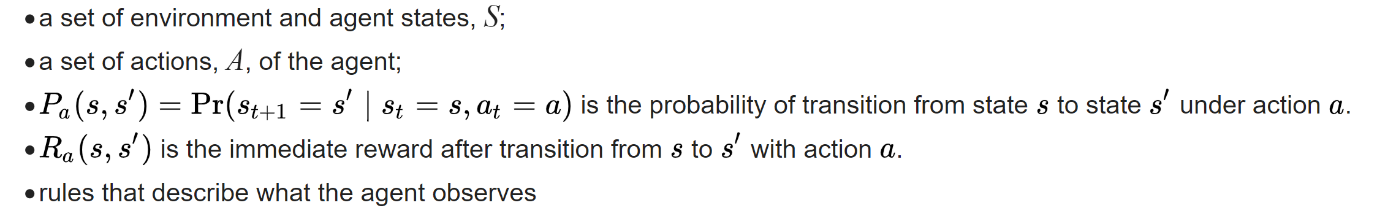
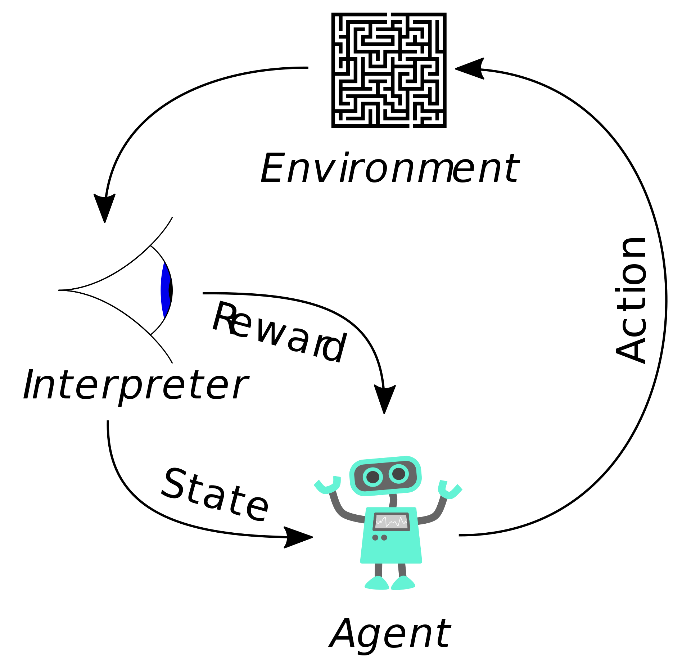
Achieving Adaptive Partition Testing based on Reinforcement Learning

# Background

Adaptive Partition Testing (APT) was recently proposed to integrate random testing and partition testing. In APT, test cases are randomly selected from some partition whose probability of being selected is adaptively adjusted along the testing process. Two algorithms of APT have been proposed: One is based on Markov Chain (MAPT), while the other based on the award and punishment mechanism (RAPT).

# Motivation

Reinforcement learning is one of the three classical machine learning paradigms (the other two are supervised and unsupervised learning). Unlike the other two, reinforcement learning does not require data for training. Basically speaking, reinforcement learning can be modeled as a Markov decision process:



A reinforcement learning agent interacts with its environment in discrete time steps. At each time *t*, the agent receives an observation *o­t*, which typically includes the reward *rt*. It then chooses an action *at* from the set of available actions, which is subsequently sent to the environment. The environment moves to a new state *st*+1 and the reward *rt*+1 associated with the transition (*st*, *at*, *st*+1) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible. The agent can (possibly randomly) choose any action as a function of the history.

From the above concepts of reinforcement learning, we can find many common keywords with APT, such as reward, Markov decision. This motivates some investigations to use reinforcement learning techniques to achieve the “adaptiveness” in APT. Hopefully, from the investigations, we can develop some advanced testing techniques that are more “intelligent” and more effective.

# Roadmap

Since it is still in the phase of “initial idea”, I hereby only suggest some steps for us to take before we can formally define the concrete research methodology.

1. Investigate the basic concepts of reinforcement learning and software testing, and work out the matching map between them. For example, which part of software testing environment can be defined as agent, action, state? What can reward mean for software testing? etc. There might already exist some research in this aspect. A literature review may thus be required for this investigation.
2. Based on the result in Step 1, try to interpret the existing two algorithms of APT in the context of reinforcement learning. For example, which part of APT has actually implemented some elements in reinforcement learning? e.g., can we say MAPT already implemented the state transition in reinforcement learning? can we say the reward in RAPT has the same meaning as the reward in reinforcement learning? Hopefully, we can find some ways to improve the current algorithms or even develop new algorithms based on reinforcement learning.
3. There are various algorithms for reinforcement learning, such as Monte Carlo, Q-Learning, SARSA, and even deep reinforcement learning (which is the basis for the famous Alpha Go). Some research can be further conducted to apply/translate these algorithms into the context of APT.
4. In the long run, if we can use reinforcement learning to achieve “adaptiveness” in APT, it is natural to extend the research to other testing techniques, for example, adaptive testing (software cybernetics) and adaptive random testing.