Good Morning everyone, today I wanna introduce you a paper <Learning contract invariants using reinforcement learning>, it uses the framework of reinforcement learning to infer the contract invariants in the Solidity Programme.

Background：

With the popularity of smart contracts in the modern financial ecosystem grows, it’s important for us to verify the correctness and the security properties. Thus the vulnerabilities such as arithmetic overflow in Program written by the smart contract specific language like Solidity should be solved..

Problems:

There exist several problems in this area such as arithmetic overflow and reentrancy. Although existing research has verified the smart contracts from different angles, the all rely on the contract invariant. The contract invariant is a logical formula over storage variables which holds the transaction boundaries. However, current methods that verify the arithmetic overflow are time-consuming and in lack of accuracy, which is not efficient to achieve the fast detect of the so-called contract invariants.

Examples:

Let’s exemplify to illustrate the importance of the contract invariants for the overflow checking.

Look at the program, there is no overflow checking at line 20, so the balances may cause the lethal safety problems. But at line 18 and line 12, the \_value is limited by the balances and total Supply to ensure the program’s security. We can get a so-called contract invariant to prove the overflow safety. So in conclusion, the balances is limited by the totalSupply in the program, but when doing the test on the mint without any information about the totalSupply, the arithmetic overflow will occur and cause lethal results.

Solution

The currently popular methods handling such issues is based on the refinement type system for the Solidity, it is likely the compile principle. CHC introduces second-order variables V that represent unknown refinement type annotations and then add hard and soft constraints over the variables. So the author proposed using reinforcement learning to infer the contract variants called CIDER. In the train phase, the author train an RL agent using arithmetic dependency graph representation, this representation is encoded by the Graph neural network using the Graph abstraction of the connections between the invariants, then the agent gives the candidates to the verifier for the performance evaluation feedback as reward to encourage it. And in the inference phase, CIDER generates the invariants for unseen contracts and proposes a sequence of invariants with decreasing likelihood based on the ADG representation.

Interestingly, the author proposes a new way to model the MDP for contract invariants based on the grammar symbols and Program, to generate from the non-terminal signal as the MDP’s action and the current program and the partial contract invariant as the state. The reinforcement learning algorithms can be implemented accordingly.

Experiments:

The author compares its algorithm to two baselines called HOUDINI and SOLIDCHC. And the graphs tell us the efficiency oh the CIDER, it costs less time and obtain better results simultaneously. The author also dose some ablation studies on the influence of ADG representation for the CIDER, and the graphs tell that the ADG is important to achieve better performance.

Conclusion:

The author proposes a new way implemented on the framework of reinforcement learning called CIDER, and it can achieve better performance and less time-consuming results than other methods. I appreciate his innovation in the creation of the MDP.