Formal Method on Hardware Security

September 2022

1 Compare the Property in Rosette and in Coq

1.1 Property in Rosette.

Property to verify. Last week, we have a state machine and want to verify a property hold for each state of the machine. Formally,

$$\begin{aligned} \forall \ & \text{input} \ \ in_0, in_1, \ldots, in_n, \\ \text{Let} \ & \ S_0 \xrightarrow{in_0} S_1 \xrightarrow{in_1} \cdots \xrightarrow{in_n} S_{n+1} \\ \text{If} \ & \ \text{No smash in the } in_0, in_1, \ldots, in_n \\ \text{We have} \ & \ PropertyHold(S_0), PropertyHold(S_1), \ldots, PropertyHold(S_n) \end{aligned}$$

Property for induction step. The property I verify for the induction step is:

$$\begin{aligned} \forall \ & \text{state } S_i, \text{ input } in_i, \ in_{i+1}, \\ \text{Let} \ & \ S_i \xrightarrow{in_i} S_{i+1} \xrightarrow{in_{i+1}} S_{i+2} \\ \text{If} \ & \ \text{No smash in the } in_i, in_{i+1} \\ \text{We have} \ & \ PropertyHold(S_{i+1}) \Rightarrow PropertyHold(S_{i+2}) \end{aligned}$$

It is a little weird since why not just use:

$$\begin{aligned} \forall \ & \text{state } S_i, \text{ input } in_i, \\ \text{Let} \ & \ S_i \xrightarrow{in_i} S_{i+1} \\ \text{If} \ & \ \text{No smash in the } in_i \\ \text{We have} \ & \ PropertyHold(S_i) \Rightarrow PropertyHold(S_{i+1}) \end{aligned}$$

The answer is that, yes, this bottom version definitely looks more natural for our toy example. However, the above version used by mistake because I was looking at the property from another perspective. I want to explain this other perspective because firstly, that is how I will define it in coq and in more other scenarios for convenience. And secondly this will help you understand that induction itself is more powerful than you might think.

1.2 Property in Coq.

Property to verify. Let's translate the property in following way:

$$\begin{aligned} \forall \ & \text{input} \ \ in_0, in_1, \ldots, in_n, \\ \text{We have} \ & \ PropertyHold'(in_0, in_1, \ldots, in_n) \end{aligned}$$

Here, stead of considering the property is a function on the state, we consider it as a function on the whole input sequence. And it says that with this input sequence, if there's no smash in it, then we run it on the state machine we defined from initial state, it should not violate property we defined above.

Property for induction step. Now, let's consider how to use the induction on this property. It is actually:

$$\begin{aligned} \forall \ & \text{input} \ \ in_0, in_1, \ldots, in_i, in_{i+1}, \\ \text{We have} \ & \ PropertyHold'(in_0, in_1, \ldots, in_i) \Rightarrow PropertyHold'(in_0, in_1, \ldots, in_i, in_{i+1}) \end{aligned}$$

Note that this is exactly the version that we will prove in coq.

Relate the coq property with Rosette property. However how does this connect to our older induction step in rosette? We can try to expand this property a little bit.

$$\begin{aligned} \forall \ \text{input} \quad & in_0, in_1, \ldots, in_i, in_{i+1}, \\ \text{Let} \quad & S_0 \xrightarrow{in_0} S_1 \xrightarrow{in_1} \cdots \xrightarrow{in_n} S_i \xrightarrow{in_{i+1}} S_{i+2} \xrightarrow{in_{i+1}} S_{i+2} \\ \text{We have} \quad & PropertyHold(S_1) \wedge \cdots \wedge PropertyHold(S_{i+1}) \Rightarrow PropertyHold(S_{i+2}) \end{aligned}$$

To further become identical to the Rosette property we are verifying, we actually conservatively think the S_n here can be an arbitrary state.