A CONCEPTUAL FRAMEWORK FOR TESTING COMMUNICATIONS PROTOCOLS

This chapter mainly introduces a fundamental architecture called OSI (Open System Interconnection) Reference Model, which standardizes the communication protocols. The chapter emphasizes on fourth section in which formal tools to specify the protocol are presented.

The fundamental assumption made in all conformance testing of communication protocol is it must be tested as a black box. And formal methods have been primarily based on FSM (Finite state machine) model, which used to recognize strings in language at first. To be specific, FSM has finite stable conditions (state), which change one state to another by an external stimulus giving by tester such as a certain input. The set of permissible input and output called the permissible input set $I$ and the permissible output set $O$. If every member of $I$ exists a permissible output for every state, this kind of system is called fully specified FSM. On the contrary, if some input are not allowed in some states, it is called partially specified FSM. But, partially specified system is not wrong because it is a design choice and sometime it is impossible to generate every input at every state. A minimal FSM stands for a system does not have two or more identical states. So, current input and state will determine its output and next state if the FSM is deterministic. Apart from that, this section also gives a formal definition and a mathematic model for FSM. Compared with FSM, an Extended FSM (EFSM)’s next state and output is determined by input as well as variable, boolean expression referring to those variables.

When it comes to the representing FSMs, two methods are presented. The most common way to do it is state transition tables. The intersection of a row and a column corresponding to a certain state and a certain input. The following Figure.1 gives the state $s_0$ with input $\text{U.ConReq}$ will transfer to state $s_i$ with output $\text{N.Setup}$. Also, the inputs that are not allowed are represented as ‘-’ in the figure.
Figure 1. An example of transition table

According to this chart, we could use another tool for representing FSMs called graph representation. The syntax of this method is easy to follow. The chapter represents the edges between states by \((v_i, v_j; \text{input}_k/\text{output}_l)\) where FSM change from \(s_i\) (corresponds to \(v_i\)) to \(s_j\) with an output \(l\) when the input is \(k\). Figure 2 shows the simple graph corresponding to Figure 1.

Figure 2. An example of transition graph

In addition, a tour in graph is a non-null sequence of consecutive edges that starts and ends at the same vertex. And current input and state will determine its output and next state if the FSM is deterministic. Plus, a stronger directed graph is free of livelock as well as deadlock. This is very important, because deadlock situation will starve all other states. The following paragraphs of this section mainly sums up the Bochmann’s paper.

Actually, ISO also has defined formal language to specify a communication protocol called FDT (formal description techniques). In this section, the writer gives examples using Estelle, SDL (Specification and Description Language), and LOTOS. After reading Babich’s paper in detail, I could figure out that Estelle is comprised of several fields such as declaration of data type, a hierarchy of module instances. SDL has a great graphical syntax and is a high-level description tool for communicating system. Last but not least, LOTOS adopts a peculiar process algebra to eliminate ambiguities in the specification.

In a nutshell, formal methods provide us with a completed approach to the reliability problem by enforcing the system requirements on a mathematical model or tools. So the correctness can be automatically proved. In addition, FSMs and EFSMs provide a state table and directed graphs to specify a certain protocol. Plus, ISO also defined formal description languages like SDL, Estelle and LOTOS.