Specifying Systems Notes

BY ARNAV KUMAR

Web: https://arnavcs.github.io

Specifying Systems is a publication written by Leslie Lamport on the TLA$^+$ language. I choose purposely to ommit leaving details in these notes about the grammar of the language, since this can be easily found. Additionally, this is not a summary or recreation of the next in any manner. As such, please read the text to gain a better understading of the contents.

1 System Specifications

Color Scheme Key
Definition
Note

System Specification	A system specification is a description of what a system should do or is intended to do. The behavioural properties of a system are also called the functional or logical properties of a system and is our focus. We do not consider performance properties.
State of a System and a Step	A state is an assignment of values to variables. A pair of successive states is called a step. We can mathematically write a pair as shown below. $$\begin{bmatrix} a=1 \\ b=0 \end{bmatrix} \rightarrow \begin{bmatrix} a=1 \\ b=1 \end{bmatrix}$$
Behaviour of a System	Formally, a behaviour is a sequence of states.
Temporal Logic	A temporal logic formula is a formula that describes a system's behaviour by relating the next state of a system with the current state.
TLA$^+$	TLA$^+$ stands for the Temporal Logic of Actions and supports both assertional resoning and temporal logic. This system is quite good with describing asynchronous systems, but can be used for nearly any purpose: APIs and distributed systems included.
Propositional Logic	The two basic boolean values, TRUE and FALSE can be used in propositional logic with the operators $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and $\equiv$ (from highest to lowest precendence).
Tautology	A tautology is a proposition that is true for all possible truth values of its identifiers. For example, the following logic-proposition is a tautology: $$F \Rightarrow F \vee G$$
Sets	A set is a collection of elements that is determined by its elements. We denote sets with curly brackets, so the set of the first three natural numbers is $\{1,2,3\}$. The empty set will be denoted as $\{\}$, and operations on sets are $\cap$, $\cup$, $\setminus$, and $\subseteq$ (highest to lowest precedence). Membership is denoted with $\in$.

Predicate Logic

The two quantifiers, $\forall$ and $\exists$, are followed with a colon and the variable in the quantifier is called "bound" as opposed to a "free" variable. See the example below where x and y are both bound.

$$\forall x \in S : (\forall y \in T : F)$$

Formulas vs. Statements

Note that by default, something like $2 * x > x$ is a noun; it is true or false depending on the value of x. On the other hand, if we would like to assert if the formula is true, then we should instead write the statement $2 * x > x$ is true.

Action

An action is true or false of a step, meaning that it it contains primed variables (from the second state) and unprimed variables (from the first state).

Anatomy of a TLA$^+$ Specification

A TLA$^+$ specification usually consists of an initial predicate and a next-state relation.

Initial Predicate	Specifies all the possible initial values of the initial state. This is a predicate that is true if the variables are possible initial values and false otherwise.
Next-State Relation	This is an action that specifies how the state can change in any step. The relation is true if the step is valid, and false otherwise.

We can use the temporal-logic ($\Box$) unary operatior to ensure that the ensuing formula is always true. Thus, given that I and N are an initial predicate and a next-state relation respectively, we see the specification of our system is described as $I \wedge \Box N$.

The specification is broken into Modules, each having their own defintions, variables, and theorems.

Uniqueness of Specifications

Since there are multiple ways to model the same thing, two specifications of the same thing are not neccessarily unique. The only thing that matters is that if F_1 and F_2 are formulas for the same behaviour, then $F_1 \equiv F_2$ is a theorem.