Notes for TLA$^+$ Class

Leslie Lamport

5–7 April 2016

Text colored **like this** is a link you can click on.

To Do Before the Class

On the laptop that you will bring to class:

- Download the TLA Toolbox. A description of the Toolbox and links to the download site are on **this Web page**. If you have a 64-bit computer, I recommend the 64-bit version; but the 32-bit version should also work.

  The toolbox needs a Java Runtime Environment (JRE) installed on your computer. Use either the **64-bit version** or the **32-bit version**, depending on the version of the Toolbox you download. (Both can be installed on the same computer.)

  To use the pretty-printer, you will need to install LaTeX. You can do this by installing **MikTeX**. If you don't already have it, you will also need to install the **Adobe Acrobat Reader**.

- Download the following two reference works.

  The TLA$^+$ Hyperbook A hypertext book.

  Specifying Systems A 360-page book available as a pdf file.

- Download **this zip file** and extract its files into a convenient directory.

- Check that the Toolbox is working by starting it (run the file `toolbox.exe` in the `toolbox` folder) and opening the specification file `DieHard.tla` in it as follows:

On the File menu, choose Open Spec and then Add New Spec. This pops up a dialog on which you should click Browse. This pops up a file browser, with which you should go to the folder in which you placed the unzipped files. Click on `DieHard.tla` and click Open. Then click Finish.

After you open the specification, you can check that LaTeX is properly installed by clicking on Produce PDF Version on the File menu.

The first example in the class will be a specification of the water jug problem from the movie *Die Hard 3*. In case you're curious about the source of the problem, you can find the relevant segment from the movie **here**.

To prepare for the next examples covered in the class, read the first three sections (pages 1–5) of the document *Transaction Commit Specifications* in the file `transaction-commit.pdf`. (The section on Paxos commit is optional reading. The TLA$^+$ specifications will be explained in the class.)

You might also want to read the first chapter of the Hyperbook before the class, but it's not required. Note that clicking on the **S** in the left margin of any page of the Hyperbook displays a useful summary of TLA$^+$.

This class is about *how* to use TLA$^+$; it will say little about *why* you should use it. That is explained in a 7-page document in the file `tla+.pdf`, which is in the zip file you downloaded.

Prerequisites

I will assume you are familiar with the following elementary math.

Propositional Logic

Elementary algebra is the mathematics of real numbers and the operators $+$, $-$, $*$ (multiplication), and / (division). Propositional logic is the mathematics of the two Boolean values TRUE and FALSE and the five operators whose names (and common pronunciations) are

$\land$ conjunction (and)	$\Rightarrow$ implication (implies)
$\lor$ disjunction (or)	$\equiv$ equivalence (is equivalent to)
$\neg$ negation (not)	

To learn how to compute with numbers, you had to memorize addition and multiplication tables and algorithms for calculating with multidigit numbers.

Propositional logic is much simpler, since there are only two values, TRUE and FALSE. To learn how to compute with these values, all you need to know are the following definitions of the five Boolean operators:

- $\wedge$ $F \wedge G$ equals TRUE iff both F and G equal TRUE.
- $\vee$ $F \vee G$ equals TRUE iff F or G equals TRUE (or both do).
- $\neg$ $\neg F$ equals TRUE iff F equals FALSE.
- $\Rightarrow$ $F \Rightarrow G$ equals TRUE iff F equals FALSE or G equals TRUE (or both).
- $\equiv$ $F \equiv G$ equals TRUE iff F and G both equal TRUE or both equal FALSE.

We can also describe these operators by *truth tables*. This truth table gives the value of $F \Rightarrow G$ for all four combinations of truth values of F and G:

F	G	$F \Rightarrow G$
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	TRUE
FALSE	FALSE	TRUE

The formula $F \Rightarrow G$ asserts that F implies G—that is, $F \Rightarrow G$ equals TRUE iff the statement "F implies G" is true. People often find the definition of $\Rightarrow$ confusing. They don't understand why FALSE $\Rightarrow$ TRUE should equal TRUE. The explanation is simple. We expect that if n is greater than 3, then it should be greater than 1, so $n > 3$ should imply $n > 1$. Therefore, the formula $(n > 3) \Rightarrow (n > 1)$ should equal TRUE for any number n. Substituting 4, 2, and 0 for n in this formula explains why $F \Rightarrow G$ means F *implies* G or, equivalently, *if* F *then* G.

Sets

Set theory is the foundation of ordinary mathematics. A set is often described as a collection of elements, but saying that a set is a collection doesn't explain very much. The concept of set is so fundamental that we don't try to define it. We take as undefined concepts the notion of a set and the relation $\in$, where $x \in S$ means that x is an element of S. We often say *is in* instead of *is an element of*.

A set can have a finite or infinite number of elements. The set of all natural numbers (0, 1, 2, etc.) is an infinite set. The set of all natural

numbers less than 3 is finite, and contains the three elements 0, 1, and 2. We can write this set $\{0, 1, 2\}$.

A set is completely determined by its elements. Two sets are equal iff they have the same elements. Thus, $\{0, 1, 2\}$ and $\{2, 1, 0\}$ and $\{0, 0, 1, 2, 2\}$ are all the same set—the unique set containing the three elements 0, 1, and 2. The empty set, which we write $\{\}$, is the unique set that has no elements.

The most common operations on sets are

$\cap$ intersection $\quad \cup$ union $\quad \subseteq$ subset $\quad \setminus$ set difference

Here are their definitions and examples of their use.

$S \cap T$ The set of elements in both S and T.
$\{1, -1/2, 3\} \cap \{1, 2, 3, 5, 7\} = \{1, 3\}$

$S \cup T$ The set of elements in S or T (or both).
$\{1, -1/2\} \cup \{1, 5, 7\} = \{1, -1/2, 5, 7\}$

$S \subseteq T$ True iff every element of S is an element of T.
$\{1, 3\} \subseteq \{3, 2, 1\}$

$S \setminus T$ The set of elements in S that are not in T.
$\{1, -1/2, 3\} \setminus \{1, 5, 7\} = \{-1/2, 3\}$

Predicate Logic

Once we have sets, it's natural to say that some formula is true for all the elements of a set, or for some of the elements of a set. Predicate logic extends propositional logic with the two quantifiers

$\forall$ universal quantification (for all)
$\exists$ existential quantification (there exists)

The formula $\forall\, x \in S : F$ asserts that formula F is true for every element x in the set S. For example, $\forall\, n \in \mathit{Nat} : n + 1 > n$ asserts that the formula $n + 1 > n$ is true for all elements n of the set Nat of natural numbers. This formula happens to be true.

The formula $\exists\, x \in S : F$ asserts that formula F is true for at least one element x in S. For example, $\exists\, n \in \mathit{Nat} : n^2 = 2$ asserts that there exists a natural number n whose square equals 2. This formula happens to be false. Formula F is true for some x in S iff F is not false for all x in S—that is, iff it's not the case that $\neg F$ is true for all x in S. Hence, the formula

$$(1) \quad (\exists\, x \in S \,:\, F) \;\equiv\; \neg(\forall\, x \in S \,:\, \neg F)$$

is a tautology of predicate logic, meaning that it is true for all values of the identifiers S and F.

Since there exists no element in the empty set, the formula $\exists\, x \in \{\} : F$ is false for every formula F. By (1), this implies that $\forall\, x \in \{\} : F$ must be true for every F.

Universal quantification generalizes conjunction. If S is a finite set, then $\forall x \in S : F$ is the conjunction of the formulas obtained by substituting the different elements of S for x in F. For example,

$$(\forall\, x \in \{2, 3, 7\} \,:\, x < y^x) \;\equiv\; (2 < y^2) \land (3 < y^3) \land (7 < y^7)$$

We sometimes informally talk about the conjunction of an infinite number of formulas when we formally mean a universally quantified formula. For example, the conjunction of the formulas $x \leq y^x$ for all natural numbers x is the formula $\forall\, x \in \mathit{Nat} \,:\, x \leq y^x$. Similarly, existential quantification generalizes disjunction.

Mathematicians use some obvious abbreviations for nested quantifiers. For example,

$$\forall\, x \in S, y \in T \,:\, F \;\text{ means }\; \forall\, x \in S \,:\, (\forall\, y \in T \,:\, F)$$

$$\exists\, w, x, y, z \in S \,:\, F \;\text{ means }\; \exists\, w \in S \,:\, (\exists\, x \in S \,:\, (\exists\, y \in S \,:\, (\exists\, z \in S \,:\, F)))$$

You may have seen $\forall\, x$ and $\exists\, x$ used without the following $\in S$. For example, $\forall\, x : P$ means that S is true for all x, not just for all x in some set. Mathematicians often use this notation informally, where "all x" actually means all x in some set that's implied by the context. For example, in a book on real analysis, it may be the set of all real numbers. TLA$^+$ also has a construct like $\forall\, x : P$, which means that P is true for all possible values of x. However, it's hardly ever used in real specifications—for one thing, the TLC model checker can't evaluate such an expression. The formula $\forall\, x \in S : P$ is equivalent to $\forall\, x : (x \in S) \Rightarrow P$.