SRC Technical Note
1994 - 001

December 16, 1994

Introduction to TLA

Leslie Lamport

digital

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

http://www.research.digital.com/SRC/

A Simple Example

We begin by specifying a system that starts with x equal to 0 and keeps incrementing x by 1 forever. In a conventional programming language, this might be written

initially $x = 0$;
loop forever $x := x + 1$ **end loop**

The TLA specification is a formula Π defined as follows, where the meaning of each conjunct is indicated by the comments.

$\Pi \triangleq (x = 0)$	Initially, x equals 0.
$\quad\quad \wedge\ \Box[x' = x + 1]_x$	Always ($\Box$), the value of x in the next state (x') equals its value in the current state (x) plus 1. Ignore the subscript x for now.
$\quad\quad \wedge\ \mathrm{WF}_x(x' = x + 1)$	Ignore this for now.

As specifications get more complicated, we need better methods of writing formulas. We use lists of formulas bulleted with $\wedge$ and $\vee$ to denote conjunctions and disjunctions, and we use indentation to eliminate parentheses. The definition of Π can then be written as

$$\begin{aligned} \Pi \triangleq\ & \wedge\ x = 0 \\ & \wedge\ \Box[x' = x + 1]_x \\ & \wedge\ \mathrm{WF}_x(x' = x + 1) \end{aligned}$$

What a Formula Means

A TLA formula is true or false on a *behavior*, which is a sequence of *states*, where a state is an assignment of values to variables. Formula Π is true on a behavior in which the i^{th} state assigns the value $i - 1$ to x, for $i = 1, 2, \ldots$.

Systems are real; behaviors are mathematical objects. To decide if a system S satisfies formula Π, we must first have a way of representing an execution of S as a behavior (a sequence of states). Given such a representation, we say that system S satisfies formula Π (or that S implements the specification Π) iff (if and only if) Π is true for every behavior corresponding to a possible execution of S.

Another Example

Next, we specify a system that starts with x and y both equal to 0 and repeatedly increments x and y by 1. A step increments either x or y (but not both). The variables are incremented in arbitrary order, but each is incremented infinitely often. This system might be represented in a conventional programming language as

> **initially** $x = 0, y = 0$;
> **cobegin**
>   **loop forever** $x := x + 1$ **end loop** $\|$
>   **loop forever** $y := y + 1$ **end loop**
> **coend**

The TLA specification is the formula Φ, defined as follows. For convenience, we first define two formulas $\mathcal{X}$ and $\mathcal{Y}$, and then define Φ in terms of $\mathcal{X}$ and $\mathcal{Y}$.

$$\begin{array}{lcll}
\mathcal{X} & \triangleq & \land\ x' = x + 1 & \text{An } \mathcal{X} \text{ step is one that increments } x \\
 & & \land\ y' = y & \text{and leaves } y \text{ unchanged.} \\[1ex]
\mathcal{Y} & \triangleq & \land\ y' = y + 1 & \text{A } \mathcal{Y} \text{ step is one that increments } y \\
 & & \land\ x' = x & \text{and leaves } x \text{ unchanged.} \\[1ex]
\Phi & \triangleq & \land\ (x = 0) \land (y = 0) & \text{Initially, } x \text{ and } y \text{ equal 0.} \\
 & & \land\ \Box[\mathcal{X} \lor \mathcal{Y}]_{\langle x,y \rangle} & \text{Every step is either an } \mathcal{X} \text{ step or a } \mathcal{Y} \text{ step.} \\
 & & \land\ \mathrm{WF}_{\langle x,y \rangle}(\mathcal{X}) \land \mathrm{WF}_{\langle x,y \rangle}(\mathcal{Y}) & \text{As explained later, this asserts that in-} \\
 & & & \text{finitely many } \mathcal{X} \text{ and } \mathcal{Y} \text{ steps occur.}
\end{array}$$

Formulas $\mathcal{X}$ and $\mathcal{Y}$ are called *actions*. An action is true or false on a *step*, which is a pair of states—an old state, described by unprimed variables, and a new state, described by primed variables.

Implementation and Stuttering

We say that a specification (TLA formula) F implements a specification G iff every system that satisfies F also satisfies G. This is true if every behavior that satisfies F also satisfies G, which means that all behaviors satisfy the formula $F \Rightarrow G$. A formula is said to be *valid* iff it is satisfied by all behaviors. ("All behaviors" means all sequences of states, not just ones that represent the execution of some particular system.) So, F implements G if the formula $F \Rightarrow G$ is valid. Implementation is implication.

A system that repeatedly increments x and y repeatedly increments x. Therefore, specification Φ should implement specification Π. This means that every behavior satisfying Φ should also satisfy Π. Behaviors that satisfy Φ allow steps that increment y and leave x unchanged. Therefore, Π must allow steps that leave x unchanged. That's where the subscript x comes in. For any action (Boolean formula containing constants, variables and primed variables) $\mathcal{A}$ and every state function (expression containing only constants and unprimed variables) f, we define

$$[\mathcal{A}]_f \;\triangleq\; \mathcal{A} \vee (f' = f)$$

where f' is the expression obtained by priming all the variables in f. Thus, a step satisfies $[\mathcal{A}]_f$ iff it satisfies $\mathcal{A}$ or it leaves f unchanged. The formula $\Box[\mathcal{A}]_f$ asserts that every step is an $\mathcal{A}$ step (one that satisfies $\mathcal{A}$) or leaves f unchanged. Hence, the conjunct $\Box[x' = x + 1]_x$ of Π does allow steps that leave x unchanged. Such steps are called *stuttering* steps.

In mathematics, the formula $x^2 = x + 1$ is not an assertion about a universe just containing x; it is an assertion about a universe containing all possible variables, including x, y, and z. The formula $x^2 = x + 1$ simply doesn't say anything about y and z. Similarly, formula Π is an assertion about sequences of states, where a state is an assignment of values to all variables, not just to x. Formula Π specifies a system whose execution is described by the changes to x. But a behavior represents a history of some entire universe containing that system. To be a sensible specification, Π must allow stuttering steps in which other parts of the universe change while x remains unchanged.

Similarly, Φ allows steps that leave the pair $\langle x, y\rangle$ unchanged, and therefore leave both x and y unchanged. If we are just observing x and y, then there is no way to tell that such a step has occurred.

Stuttering steps make it unnecessary to consider finite behaviors. An execution in which a system halts is represented by an infinite behavior in which the variables describing that system stop changing after a finite number of steps. When a system halts, it doesn't mean that the entire universe comes to an end. Thus, by a behavior, we mean an infinite sequence of states.

Fairness

Formula $\Box[x' = x+1]_x$ allows arbitrarily many steps that leave x unchanged. In fact, it is satisfied by a behavior in which x never changes. We want to require that x be incremented infinitely many times, so our specification must rule out behaviors in which x is incremented only a finite number of times. This is accomplished by the WF formula, as we now explain.

An action $\mathcal{A}$ is said to be *enabled* in a state s iff there exists some state t such that the pair of states $\langle$old-state s, new-state $t\rangle$ satisfies $\mathcal{A}$. The formula $\mathrm{WF}_f(\mathcal{A})$ asserts of a behavior that, if the action $\mathcal{A} \wedge (f' \neq f)$ ever becomes enabled and remains enabled forever, then infinitely many $\mathcal{A} \wedge (f' \neq f)$ steps occur. In other words, if it ever becomes possible and remains forever possible to execute an $\mathcal{A}$ step that changes f, then infinitely many such steps must occur.

Any integer can be incremented by 1 to produce a different integer. Hence, the action $(x' = x+1) \wedge (x' \neq x)$ is enabled in any state where x is an integer. The formula $(x = 0) \wedge \Box[x' = x + 1]_x$, which asserts that x is initially 0 and in every step is either incremented by 1 or left unchanged, implies that x is always an integer. Hence, this formula implies that $(x' = x + 1) \wedge (x' \neq x)$ is always enabled. Hence, the conjunct $\mathrm{WF}_x(x' = x + 1)$ of Π asserts that infinitely many $(x' = x + 1) \wedge (x' \neq x)$ steps occur. Hence, Π asserts that x is incremented infinitely often, as desired.

Similarly, $(x = 0) \wedge \Box[\mathcal{X} \vee \mathcal{Y}]_{\langle x,y\rangle}$ implies that x is always an integer, so $\mathcal{X} \wedge (\langle x, y\rangle' \neq \langle x, y\rangle)$ is always enabled. Hence, Φ implies that x is incremented infinitely often. Every behavior satisfying Φ does satisfy Π, so $\Phi \Rightarrow \Pi$ is valid.

WF stands for *Weak Fairness*. TLA specifications also use *Strong Fairness* formulas of the form $\mathrm{SF}_f(\mathcal{A})$, where f is a state function and $\mathcal{A}$ an action. This formula asserts that if $\mathcal{A} \wedge (f' \neq f)$ is enabled infinitely often (in infinitely many states of the behavior), then infinitely many $\mathcal{A} \wedge (f' \neq f)$ steps must occur. If an action ever becomes enabled forever, then it is enabled infinitely often. Hence, $\mathrm{SF}_f(\mathcal{A})$ implies $\mathrm{WF}_f(\mathcal{A})$; strong fairness implies weak fairness.

The subscripts in WF and SF formulas (and in the formula $\Box[\mathcal{N}]_f$) make it syntactically impossible to write a formula that can distinguish whether or not stuttering steps have occurred. In practice, whenever we write a formula of the form $\mathrm{WF}_f(\mathcal{A})$ or $\mathrm{SF}_f(\mathcal{A})$, action $\mathcal{A}$ will imply $f' \neq f$, so any $\mathcal{A}$ step changes f.

Hiding

The formula $\exists\!\!\exists\, y : \Phi$ is satisfied by a behavior iff there is some sequence of values that can be assigned to y which would produce a behavior satisfying Φ. (This definition is only approximately correct; see [2] for the precise definition.) The temporal existential quantifier $\exists\!\!\exists\, y$ is the formal expression of what it means to "hide" the variable y in a specification. If we hide y in a specification asserting that x and y are repeatedly incremented, we get a specification asserting that x is repeatedly incremented. Thus, the specification obtained by hiding y in Φ should be equivalent to Π. Indeed, the formula $\exists\!\!\exists\, y : \Phi$ is equivalent to Π. In other words, the formula $(\exists\!\!\exists\, y : \Phi) \equiv \Pi$ is valid.

Composition

Let $\mathcal{X}$ and $\mathcal{Y}$ be the actions defined above, and let

$$\begin{aligned}
\Pi_x &\triangleq (x = 0) \wedge \Box[\mathcal{X}]_x \wedge \mathrm{WF}_{\langle x,y \rangle}(\mathcal{X}) \\
\Pi_y &\triangleq (y = 0) \wedge \Box[\mathcal{Y}]_y \wedge \mathrm{WF}_{\langle x,y \rangle}(\mathcal{Y})
\end{aligned}$$

A simple calculation shows that, if x and y are integers, then $[\mathcal{X}]_x \wedge [\mathcal{Y}]_y$ is equivalent to $[\mathcal{X} \vee \mathcal{Y}]_{\langle x,y \rangle}$. It follows from this and the laws of temporal logic that $\Pi_x \wedge \Pi_y$ is equivalent to Φ. We can interpret Π_x and Π_y as the specifications of two processes, one repeatedly incrementing x and the other repeatedly incrementing y, in a program whose variables are x and y. Composing two such processes yields a program, with variables x and y, that repeatedly increments both x and y—the program specified by Φ.

In general, a specification F of a system S describes the behaviors (representing histories) of a universe in which S operates correctly. A specification G of a system T describes behaviors of the same universe in which T operates correctly. Composing S and T means ensuring that both S and T operate correctly in that universe. The behaviors of a universe in which both systems operate correctly are described by the formula $F \wedge G$. Composition is conjunction.

Assumption/Guarantee Specifications

An assumption/guarantee specification asserts that a system operates correctly if the environment does. Let M be a formula asserting that the system does what we want it to, and let E be a formula asserting that the environment does what it is supposed to. We would expect the assumption/guarantee specification to be $E \Rightarrow M$, the formula asserting that either M is satisfied (the system behaved as desired) or E is not satisfied (the environment did not behave correctly). However, we instead write the stronger specification $E \stackrel{+}{-\!\!\rhd} M$, which asserts both that E implies M, and that no step can make M false unless E has already been made false. The precise meaning of the formula $E \stackrel{+}{-\!\!\rhd} M$ is given in [1].

All of TLA

TLA is built on a logic of actions, which is a language for writing predicates, state functions, and actions, and a logic for reasoning about them. A predicate is a Boolean expression containing constants and variables; a state function is a nonBoolean expression containing constants and variables; and an action is a Boolean expression containing constants, variables, and primed variables. The complete specification language TLA$^+$, described elsewhere, includes such a language.

Syntactically, a TLA formula has one of the following forms:

$$\begin{array}{lllll}
P & \Box[\mathcal{A}]_f & \Box F & \boldsymbol{\exists}\, x : F & \\
\neg F & F \wedge G & F \vee G & F \Rightarrow G & F \equiv G \\
\mathrm{WF}_f(\mathcal{A}) & \mathrm{SF}_f(\mathcal{A}) & F \stackrel{+}{-\!\!\rhd} G & \Diamond F & F \leadsto G
\end{array}$$

where P is a predicate, f is a state function, $\mathcal{A}$ is an action, x is a variable, and F and G are TLA formulas. The last row of formulas can be expressed in terms of the others (and of course, all the Boolean operators can be defined from $\neg$ and $\wedge$). The Boolean operators have their usual meanings; the meanings of the other operators are described below.

P — Satisfied by a behavior iff P is true for (the values assigned to variables by) the initial state.

$\Box[\mathcal{A}]_f$ — Satisfied by a behavior iff every step satisfies $\mathcal{A}$ or leaves f unchanged.

$\Box F$ — (F is always true.) Satisfied by a behavior iff F is true for all suffixes of the behavior.

$\exists\!\!\exists\, x : F$ Satisfied by a behavior iff there are some values that can be assigned to x to produce a behavior satisfying F. (See [2] for the precise definition.)

$\mathrm{WF}_f(\mathcal{A})$ (Weak fairness of $\mathcal{A}$) Satisfied by a behavior iff $\mathcal{A} \wedge (f' \neq f)$ is infinitely often not enabled, or infinitely many $\mathcal{A} \wedge (f' \neq f)$ steps occur.

$\mathrm{SF}_f(\mathcal{A})$ (Strong fairness of $\mathcal{A}$) Satisfied by a behavior iff $\mathcal{A} \wedge (f' \neq f)$ is only finitely often enabled, or infinitely many $\mathcal{A} \wedge (f' \neq f)$ steps occur.

$F \stackrel{+}{\rhd} G$ Is true for a behavior iff G is true for at least as long as F is. (See [1] for the precise definition.)

$\Diamond F$ (F is eventually true) Defined to be $\neg \Box \neg F$.

$F \leadsto G$ (Whenever F is true, G will eventually become true) Defined to be $\Box(F \Rightarrow \Diamond G)$.

References

[1] Martín Abadi and Leslie Lamport. Conjoining specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–534, May 1995.

[2] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.