A Summary of TLA$^+$

Leslie Lamport

25 Aug 2000

Module-Level Constructs

—— MODULE M ——

Begins the module or submodule named M.

EXTENDS $M_1, \ldots, M_n$

Incorporates the declarations, definitions, assumptions, and theorems from the modules named $M_1, \ldots, M_n$ into the current module.

CONSTANTS $C_1, \ldots, C_n$ (1)

Declares the C_j to be constant parameters (rigid variables). Each C_j is either an identifier or has the form $C(\_, \ldots, \_)$, the latter form indicating that C is an operator with the indicated number of arguments.

VARIABLES $x_1, \ldots, x_n$ (1)

Declares the x_j to be variables (parameters that are flexible variables).

ASSUME P

Asserts P as an assumption.

$F(x_1, \ldots, x_n) \triangleq exp$

Defines F to be the operator such that $F(e_1, \ldots, e_n)$ equals *exp* with each identifier x_k replaced by e_k. (For $n = 0$, it is written $F \triangleq exp$.)

$f[x \in S] \triangleq exp$ (2)

Defines f to be the function with domain S such that $f[x] = exp$ for all x in S. (The symbol f may occur in *exp*, allowing a recursive definition.)

(1) The terminal S in the keyword is optional.

(2) $x \in S$ may be replaced by a comma-separated list of items $v \in S$, where v is either a comma-separated list or a tuple of identifiers.

INSTANCE M WITH $p_1 \leftarrow e_1, \ldots, p_m \leftarrow e_m$

For each defined operator F of module M, this defines F to be the operator whose definition is obtained from the definition of F in M by replacing each declared constant or variable p_j of M with e_j. (If $m = 0$, the WITH is omitted.)

$N(x_1, \ldots, x_n) \triangleq$ INSTANCE M WITH $p_1 \leftarrow e_1, \ldots, p_m \leftarrow e_m$

For each defined operator F of module M, this defines $N(d_1, \ldots, d_n)!F$ to be the operator whose definition is obtained from the definition of F by replacing each declared constant or variable p_j of M with e_j, and then replacing each identifier x_k with d_k. (If $m = 0$, the WITH is omitted.)

THEOREM P

Asserts that P can be proved from the definitions and assumptions of the current module.

LOCAL *def*

Makes the definition(s) of *def* (which may be a definition or an INSTANCE statement) local to the current module, thereby not obtained when extending or instantiating the module.

└──────────────────────────┘

Ends the current module or submodule.

The Constant Operators

Logic

$\land \quad \lor \quad \lnot \quad \Rightarrow \quad \equiv$

TRUE FALSE BOOLEAN [the set {TRUE, FALSE}]

$\forall x : p \quad \exists x : p \quad \forall x \in S : p$ (1) $\quad \exists x \in S : p$ (1)

CHOOSE $x : p$ [An x satisfying p] CHOOSE $x \in S : p$ [An x in S satisfying p]

Sets

$= \quad \neq \quad \in \quad \notin \quad \cup \quad \cap \quad \subseteq \quad \setminus$ [set difference]

$\{e_1, \ldots, e_n\}$	[Set consisting of elements e_i]
$\{x \in S : p\}$ (2)	[Set of elements x in S satisfying p]
$\{e : x \in S\}$ (1)	[Set of elements e such that x in S]
SUBSET S	[Set of subsets of S]
UNION S	[Union of all elements of S]

Functions

$f[e]$	[Function application]
DOMAIN f	[Domain of function f]
$[x \in S \mapsto e]$ (1)	[Function f such that $f[x] = e$ for $x \in S$]
$[S \rightarrow T]$	[Set of functions f with $f[x] \in T$ for $x \in S$]
$[f$ EXCEPT $![e_1] = e_2]$ (3)	[Function $\widehat{f}$ equal to f except $\widehat{f}[e_1] = e_2$. An @ in e_2 equals $f[e_1]$.]

Records

$e.h$	[The h-component of record e]
$[h_1 \mapsto e_1, \ldots, h_n \mapsto e_n]$	[The record whose h_i component is e_i]
$[h_1 : S_1, \ldots, h_n : S_n]$	[Set of all records with h_i component in S_i]
$[r$ EXCEPT $!.h = e]$ (3)	[Record $\widehat{r}$ equal to r except $\widehat{r}.h = e$. An @ in e equals $r.h$.]

Tuples

$e[i]$	[The i^{th} component of tuple e]
$\langle e_1, \ldots, e_n \rangle$	[The n-tuple whose i^{th} component is e_i]
$S_1 \times \ldots \times S_n$	[The set of all n-tuples with i^{th} component in S_i]

Strings and Numbers

"$\mathsf{c}_1 \ldots \mathsf{c}_n$"	[A literal string of n characters]
STRING	[The set of all strings]
$d_1 \ldots d_n \quad d_1 \ldots d_n \, . \, d_{n+1} \ldots d_m$	[Numbers (where the d_i are digits)]

---

(1) $x \in S$ may be replaced by a comma-separated list of items $v \in S$, where v is either a comma-separated list or a tuple of identifiers.

(2) x may be an identifier or tuple of identifiers.

(3) $![e_1]$ or $!.h$ may be replaced by a comma separated list of items $!a_1 \cdots a_n$, where each a_i is $[e_i]$ or $.h_i$.

Miscellaneous Constructs

IF p THEN e_1 ELSE e_2	[e_1 if p true, else e_2]
CASE $p_1 \to e_1 \; \Box \; \ldots \; \Box \; p_n \to e_n$	[Some e_i such that p_i true]
CASE $p_1 \to e_1 \; \Box \; \ldots \; \Box \; p_n \to e_n \; \Box$ OTHER $\to e$	[Some e_i such that p_i true, or e if all p_i are false]

LET $d_1 \triangleq e_1 \;\; \ldots \;\; d_n \triangleq e_n$ IN e [e in the context of the definitions]

$$\begin{array}{ll} \land \; p_1 & \text{[the conjunction } p_1 \land \ldots \land p_n\text{]} \\ \quad \vdots & \\ \land \; p_n & \end{array} \qquad \begin{array}{ll} \lor \; p_1 & \text{[the disjunction } p_1 \lor \ldots \lor p_n\text{]} \\ \quad \vdots & \\ \lor \; p_n & \end{array}$$

The Action Operators

e'	[The value of e in the final state of a step]
$[A]_e$	[$A \lor (e' = e)$]
$\langle A \rangle_e$	[$A \land (e' \neq e)$]
ENABLED A	[An A step is possible]
UNCHANGED e	[$e' = e$]
$A \cdot B$	[Composition of actions]

The Temporal Operators

$\Box F$	[F is always true]
$\Diamond F$	[F is eventually true]
$\mathrm{WF}_e(A)$	[Weak fairness for action A]
$\mathrm{SF}_e(A)$	[Strong fairness for action A]
$F \leadsto G$	[F leads to G]
$F \stackrel{+}{\rhd} G$	[F guarantees G (an assumption/guarantee specification)]
$\boldsymbol{\exists}\, x : F$	[Temporal existential quantification (hiding).]
$\boldsymbol{\forall}\, x : F$	[Temporal universal quantification.]

User-Definable Operator Symbols

Infix Operators

$+$ [1]	$-$ [1]	$*$ [1]	$/$ [2]	$\circ$ [3]	$++$
$\div$ [1]	$\%$ [1]	$\hat{}$ [1,4]	$..$ [1]	$...$	$--$
$\oplus$ [5]	$\ominus$ [5]	$\otimes$	$\oslash$	$\odot$	$**$
$<$ [1]	$>$ [1]	$\leq$ [1]	$\geq$ [1]	$\sqcap$	$//$
$\prec$	$\succ$	$\preceq$	$\succeq$	$\sqcup$	$\hat{}\hat{}$
$\ll$	$\gg$	$<:$	$:>$ [6]	$\&$	$\&\&$
$\sqsubset$	$\sqsupset$	$\sqsubseteq$ [5]	$\sqsupseteq$	$\mid$	$\|\|$
$\subset$	$\supset$		$\supseteq$	$\star$	$\%\%$
$\vdash$	$\dashv$	$\models$	$=\!\mid$	$\bullet$	$\#\#$
$\sim$	$\simeq$	$\approx$	$\cong$	$\$$	$\$\$$
$:=$	$::=$	$\asymp$	$\doteq$	$?$	$??$
$\propto$	$\wr$	$\uplus$	$\bigcirc$	$!!$	$@@$ [6]

Postfix Operators [7]

$\hat{}+$ $\quad\hat{}*$ $\quad\hat{}\#$

Prefix Operator

$-$ [8]

---

(1) Defined by the *Naturals*, *Integers*, and *Reals* modules.
(2) Defined by the *Reals* module.
(3) Defined by the *Sequences* module.
(4) $x\hat{}y$ is printed as x^y.
(5) Defined by the *Bags* module.
(6) Defined by the *TLC* module.
(7) $e\hat{}+$ is printed as e^+, and similarly for $\hat{}*$ and $\hat{}\#$.
(8) Defined by the *Integers* and *Reals* modules.

ASCII Representations of Symbols

$\land$	`/\` or `\land`	$\lor$	`\/` or `\lor`	$\Rightarrow$	`=>`
$\lnot$	`~` or `\lnot` or `\neg`	$\equiv$	`<=>` or `\equiv`	$\triangleq$	`==`
$\in$	`\in`	$\notin$	`\notin`	$\neq$	`#` or `/=`
$\langle$	`<<`	$\rangle$	`>>`	$\Box$	`[]`
$<$	`<`	$>$	`>`	$\Diamond$	`<>`
$\leq$	`\leq` or `=<`	$\geq$	`\geq` or `>=`	$\leadsto$	`~>`
$\ll$	`\ll`	$\gg$	`\gg`	$\stackrel{+}{\triangleright}$	`-+->`
$\prec$	`\prec`	$\succ$	`\succ`	$\mapsto$	`|->`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\div$	`\div`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\cdot$	`\cdot`
$\subset$	`\subset`	$\supset$	`\supset`	$\circ$	`\o` or `\circ`
$\sqsubset$	`\sqsubset`	$\sqsupset$	`\sqsupset`	$\bullet$	`\bullet`
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\star$	`\star`
$\vdash$	`|-`	$\dashv$	`-|`	$\bigcirc$	`\bigcirc`
$\models$	`|=`	$=\!\mid$	`=|`	$\sim$	`\sim`
$\rightarrow$	`->`	$\leftarrow$	`<-`	$\simeq$	`\simeq`
$\cap$	`\cap` or `\intersect`	$\cup$	`\cup` or `\union`	$\asymp$	`\asymp`
$\sqcap$	`\sqcap`	$\sqcup$	`\sqcup`	$\approx$	`\approx`
$\oplus$	`(+)` or `\oplus`	$\uplus$	`\uplus`	$\cong$	`\cong`
$\ominus$	`(-)` or `\ominus`	$\times$	`\X` or `\times`	$\doteq$	`\doteq`
$\odot$	`(.)` or `\odot`	$\wr$	`\wr`	x^y	`x^y` [2]
$\otimes$	`(\X)` or `\otimes`	$\propto$	`\propto`	x^+	`x^+` [2]
$\oslash$	`(/)` or `\oslash`	“s”	`"s"` [1]	x^*	`x^*` [2]
$\exists$	`\E`	$\forall$	`\A`	$x^\#$	`x^#` [2]
$\boldsymbol{\exists}$	`\EE`	$\boldsymbol{\forall}$	`\AA`	$'$	`'`
	`--------` [3]		`--------` [3]		
	`--------` [3]		`========` [3]		

(1) s is a sequence of characters.
(2) x and y are any expressions.
(3) a sequence of four or more `-` or `=` characters.

The Most Common Standard Modules

Modules *Naturals*, *Integers*, *Reals*

Define $+ \quad - \quad * \quad / \quad \hat{} \quad .. \quad Nat \quad Real$
$\div \quad \% \quad \leq \quad \geq \quad < \quad > \quad Int \quad Infinity$

Prefix $-$ is not defined in *Naturals*.

$a\hat{}b$ denotes a^b.

Nat, *Int*, and *Real* are the sets of naturals, integers, and real numbers.

$a \,..\, b$ equals $\{n \in Int : a \leq n \leq b\}$.

$a \% b$ equals a mod b, defined so $0 \leq a \% b < b$, if b is a positive integer.

$\div$ is defined so $a = b * (a \div b) + (a \% b)$ for a and b integers with $b > 0$.

$/$ (division) is defined only in *Reals*.

Infinity is defined in *Reals* so $-Infinity < r < Infinity$ for all $r \in Real$.

Module *Sequences*

Defines $\circ \quad Head \quad SelectSeq \quad SubSeq$
$Append \quad Len \quad Seq \quad Tail$

The tuple/sequence $\langle e_1, \ldots, e_n \rangle$ equals the function $[i \in 1 \,..\, n \mapsto e_i]$.

$s \circ t$ is the concatenation of sequences s and t.

$Append(\langle e_1, \ldots, e_n \rangle, e_{n+1}) = \langle e_1, \ldots, e_{n+1} \rangle$

$Head(\langle e_1, \ldots, e_n \rangle) = e_1$

$Tail(\langle e_1, \ldots, e_n \rangle) = \langle e_2, \ldots, e_n \rangle$

$Len(\langle e_1, \ldots, e_n \rangle) = n$

$Seq(S)$ is the set of all finite sequences of elements of S.

$SubSeq(\langle e_1, \ldots, e_n \rangle, j, k) = \langle e_j, \ldots, e_k \rangle$

$SelectSeq(s, Test)$ is the subsequence of elements e of s satisfying $Test(e)$.

Module *FiniteSets*

Defines *IsFiniteSet* *Cardinality*

$IsFiniteSet(S)$ is true iff S is a finite set.

$Cardinality(S)$ is the number of elements in S, if S is a finite set.