A brief introduction to TLA: the temporal logic of actions

Uğur Yavuz
(Boston University)

Oct 15, 2025

Overview

- **Motivation for TLA:**
  Traditional formalisms based on pre- and post-conditions are inadequate for describing *reactive systems* that continuously interact with their environment and need not terminate.

  Temporal logic provides an elegant framework for expressing *safety* and *liveness* properties of such systems.

Overview

- TLA takes the **linear-time** view of temporal logic, where a system is represented by its set of executions, modeled as infinite sequences of states.
- TLA expresses both *system specifications* and *correctness properties* within the same logical language.
- TLA is well-suited to writing **state machine**-like specifications of systems, whose behaviors are described in terms of **action** relations between current and next states that define how variables change.
- TLA allows *stuttering steps* that leave high-level variables unchanged, making it possible to relate specifications written at different levels of abstraction.

Language of TLA

- TLA distinguishes:
  - **Transition formulas:** describe individual state transitions.
  - **Temporal formulas:** describe infinite sequences of states (behaviors).
- Transition formulas are built from ordinary **first-order logic (FOL)**.
- TLA$^+$ is TLA over a **first-order set-theoretical** language.

Language of TLA: transition formulas

- The underlying **first-order signature** consists of:
  - L_F: set of *function symbols*, each with fixed arity (e.g. $+$, $*$, f).
  - L_P: set of *predicate symbols*, each with fixed arity (e.g. $=$, $<$, p).
  - V: set of *variables*.
- Variables are partitioned into:
  - V_F: **flexible variables** (v): may change between states.
  - V_R: **rigid variables** (x): constant across all states.

Transition formulas

- Define:
$$V_{F'} = \{ v' \mid v \in V_F \}, \qquad V_E = V \cup V_{F'}$$
as the set of all variables (rigid, flexible, and primed).
- **Transition expressions** are first-order expressions built from the symbols in L_F and L_P, and from the variables in V_E.
- **Transition predicates (actions)** are first-order formulas built from the symbols in L_F and L_P, and from the variables in V_E, using usual logical connectives and quantifiers, where quantification is only over rigid variables.
- In TLA literature,

$$\text{transition formulas} = \text{transition expressions} + \text{actions}.$$

Example: action

$$C \triangleq \exists x : \exists y : p(f(v, x)) \land \neg(x = y) \land v' = y$$

- $v \in V_F$: flexible
- $v' \in V_{F'}$: primed flexible
- $x, y \in V_R$: rigid

There exists x, y such that $p(f(v, x))$ holds, x differs from y, and v' is y.

Semantics of transition formulas

- An **interpretation** I defines:
  - Universe $|I|$ of values
  - For each n-ary $f \in L_F$, a function $|I|^n \to |I|$
  - For each n-ary $p \in L_P$, a relation on $|I|^n$
- A **state** $s : V_F \to |I|$, and a **valuation** $\xi : V_R \to |I|$.
- For states s, t and valuation ξ, define the combined valuation $\alpha_{s,t,\xi}$ over all variables in V_E:

Rigid $x \in V_R$	**Flexible** $v \in V_F$	**Primed** $v' \in V_{F'}$
$\alpha_{s,t,\xi}(x) = \xi(x)$	$\alpha_{s,t,\xi}(v) = s(v)$	$\alpha_{s,t,\xi}(v') = t(v)$

Semantics of transition formulas

- The semantics of a transition expression or predicate E is written as

$$[\![E]\!]^{I,\xi}_{s,t}$$

  and defined as the standard first-order interpretation of E under the combined valuation $\alpha_{s,t,\xi}$.

  We will omit I when clear from context.

- An action (transition predicate) A is
  - **valid** for I if $[\![A]\!]^{I,\xi}_{s,t} = \top$ for all s, t, ξ;
  - **satisfiable** for I if $[\![A]\!]^{I,\xi}_{s,t} = \top$ for some s, t, ξ.

Constant and state formulas

- **Constant formulas:** only rigid variables.

$$x > 0 \text{ (constant predicate)}$$

  Their semantics depend only on ξ.
- **State formulas:** no primed flexible variables.

$$v_1 + v_2 \text{ (state expression)}$$

$$\exists x : \exists y : p(f(v, x)) \wedge \neg(x = y) \text{ (state predicate)}$$

  Their semantics (denoted $[\![P]\!]_s^\xi$) depends only on state s (and ξ).

Primed formulas

For a state expression or predicate E, we introduce the notation E' for the formula obtained by replacing each flexible v with v'.

Example:

$$D \triangleq \exists x : \exists y : p(f(v, x)) \wedge \neg(x = y)$$

$$D' = \exists x : \exists y : p(f(v', x)) \wedge \neg(x = y)$$

Action enabledness

- For an action A, define the state predicate

$$\textsc{Enabled}\, A \triangleq \exists x_1, \ldots, x_n : A\big[\, x_1/v'_1, \ldots, x_n/v'_n \,\big]$$

  where
  - $v'_1, \ldots, v'_n$ are all the (free) *primed variables* in A,
  - $x_1, \ldots, x_n$ are fresh rigid variables (i.e. not in A).

Action enabledness — observation

Observation

$[\![\textsc{Enabled}\, A]\!]^{\xi}_{s} = \top$ iff there exists a state t such that $[\![A]\!]^{\xi}_{s,t} = \top$.

Formally, $[\![\textsc{Enabled}\, A]\!]^{\xi}_{s} = \top$ iff $[\![\exists \vec{x} : A[\vec{x}/\vec{v}']]\!]^{\xi}_{s} = \top$, which holds iff there exist values $\vec{a} \in |I|^n$ such that $[\![A[\vec{x}/\vec{v}']]\!]^{\xi[\vec{x}\mapsto\vec{a}]}_{s} = \top$.

Let $t = s[\vec{v} \mapsto \vec{a}]$, leaving all other flexible variables unchanged. By induction on the structure of formulas, we can show that

$$[\![A]\!]^{\xi}_{s,t} = [\![A[\vec{x}/\vec{v}']]\!]^{\xi[\vec{x}\mapsto\vec{a}]}_{s} = \top.$$

Intuition: $\textsc{Enabled}\, A$ holds in state s iff there is a next state t, obtained by assigning suitable values to the primed variables of A, such that A is true between s and t. In other words, the action A *may occur* in s.

Stuttering steps

- For an action A and state expression sf:

$$[A]_{sf} \triangleq A \lor (sf' = sf)$$

- For a pair of states s, t,

$$[\![[A]_{sf}]\!]^{\xi}_{s,t} = \top \text{ iff } [\![A \lor (sf' = sf)]\!]^{\xi}_{s,t} = \top$$

$$\text{iff } ([\![A]\!]^{\xi}_{s,t} = \top) \text{ or } \underbrace{[\![sf']\!]^{\xi}_{s,t}}_{[\![sf]\!]^{\xi}_{t}} = \underbrace{[\![sf]\!]^{\xi}_{s,t}}_{[\![sf]\!]^{\xi}_{s}}.$$

Intuition: $[A]_{sf}$ holds for a pair of states s, t iff A holds between s and t, or the value of sf is the same in both states. In other words, $[A]_{sf}$ is true for all transitions that either satisfy A or leave sf unchanged (a stuttering step on sf).

Temporal formulas

- Temporal formulas are inductively built predicates:

Form	**Meaning / Condition**
P	any *state predicate* P
$\neg F,\ F_1 \wedge F_2,\ F_1 \Rightarrow F_2,\ \ldots$	Boolean combinations of formulas
$\Box F$	always F
$\Box[A]_{sf}$	always (action A or stuttering on sf)
$\exists x : F$	quantification over rigid $x \in V_R$
$\exists\!\!\exists\, v : F$	quantification over flexible $v \in V_F$

Temporal formulas – observation

Observation

Every *state predicate* P is a temporal formula. However, an *action* A is not, nor is $[A]_{sf}$ itself. Actions appear in temporal formulas only in the form $\Box[A]_{sf}$.

Semantics of temporal formulas

- Temporal formulas are evaluated under an interpretation I, a valuation $\xi : V_R \to |I|$, and a **behavior** $\sigma = \sigma_0\sigma_1\sigma_2\ldots$: an ω-sequence of states $\sigma_i : V_F \to |I|$.
- The semantics assigns a truth value to $[\![F]\!]^{\xi}_{\sigma}$.

1. State predicates are evaluated at the first state of the behavior:

$$[\![P]\!]^{\xi}_{\sigma} = [\![P]\!]^{\xi}_{\sigma_0}.$$

Semantics of temporal formulas

2. Boolean connectives are interpreted as in first-order logic. For example, for implication:

$$[\![F_1 \Rightarrow F_2]\!]^{\xi}_{\sigma} = \top \text{ iff } [\![F_1]\!]^{\xi}_{\sigma} = \top \text{ implies } [\![F_2]\!]^{\xi}_{\sigma} = \top.$$

This means that (under fixed ξ), every behavior σ that satisfies F_1 also satisfies F_2.

Semantics of temporal formulas

Notation: $\sigma|_i$ denotes the suffix $\sigma_i\sigma_{i+1}\ldots$

3. For $\Box F$, we have that:

$$[\![\Box F]\!]^{\xi}_{\sigma} = \top \text{ iff } [\![F]\!]^{\xi}_{\sigma|_i} = \top \text{ for all } i \in \mathbb{N}.$$

This is the standard "always" operator of LTL.

If F is a state predicate, note this is equivalent to $[\![F]\!]^{\xi}_{\sigma_i} = \top$ for all i.

Semantics of temporal formulas

4. For $\Box[A]_{sf}$, we have that:

$$[\![\Box[A]_{sf}]\!]^{\xi}_{\sigma} = \top \text{ iff for all } i \in \mathbb{N},$$

$$\left([\![sf]\!]^{\xi}_{\sigma_i} = [\![sf]\!]^{\xi}_{\sigma_{i+1}}\right) \text{ or } \left([\![A]\!]^{\xi}_{\sigma_i,\sigma_{i+1}} = \top\right).$$

This means for every successive pair of states in σ, either sf is unchanged (a stuttering step on sf), or the action A holds between them.

Semantics of temporal formulas

5. For quantification over rigid $x \in V_R$:

$$[\![\exists x : F]\!]^{\xi}_{\sigma} = \top \text{ iff there exists a value } a \in |I|$$

$$\text{such that } [\![F]\!]^{\xi[x \mapsto a]}_{\sigma} = \top.$$

   This is standard first-order quantification over rigid variables.

6. We omit the semantics of quantification over flexible $v \in V_F$. It needs careful definition to respect *stuttering invariance*.

- A formula F is **valid** for I if $[\![F]\!]^{\xi}_{\sigma} = \top$ for all σ, ξ. We denote $\models_I F$, or simply $\models F$ when I is clear.

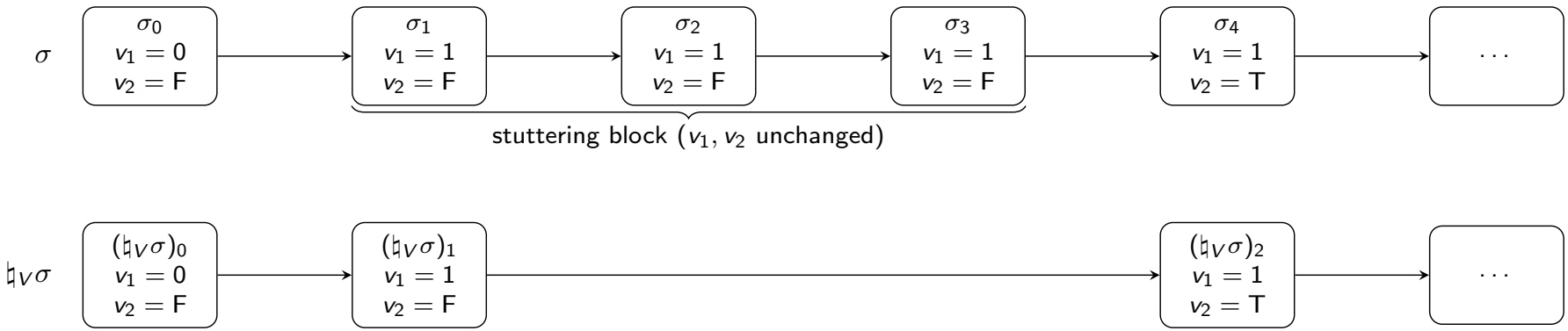
Stuttering equivalence

- For states s, t, and a set V of flexible variables, define

$$s =_V t \;\text{ iff }\; \text{for all } v \in V,\; s(v) = t(v)$$

(s *and* t *are* V*-equivalent*.)

- For a behavior $\sigma = \sigma_0\sigma_1\ldots$, its **$V$-unstuttered variant** $\natural_V\,\sigma$ replaces each maximal finite block of V-equivalent states by its first state.



Stuttering equivalence

- Two behaviors σ and τ are V-**stuttering equivalent** $(\sigma \approx_V \tau)$ if

$$\natural_V \sigma = \natural_V \tau.$$

- Intuitively, $\sigma \approx_V \tau$ means one can be obtained from the other by inserting or deleting finite repetitions of V-equivalent states.

Stuttering invariance

- **TLA is insensitive to stuttering equivalence.**

Theorem (Stuttering invariance)

Let F be a TLA formula whose free flexible variables are in V.

If $\sigma \approx_V \tau$ and ξ is a valuation, then

$$[\![F]\!]^{\xi}_{\sigma} = [\![F]\!]^{\xi}_{\tau}.$$

- Intuitively, TLA formulas cannot distinguish behaviors that are stuttering equivalent with respect to their free flexible variables.

Derived operators

- TLA defines several **derived operators** on temporal formulas:
  - **Eventually** ($\Diamond$):

    $$\Diamond F \triangleq \neg\Box\neg F$$

    holds if F is true for some suffix of the behavior.
  - **Eventually always** ($\Diamond\Box$) and **infinitely often** ($\Box\Diamond$).

Observation

Only the combinations $\Box$, $\Diamond$, $\Box\Diamond$, and $\Diamond\Box$ yield distinct temporal operators:

$$\Box\Box F \equiv \Box F, \qquad \Diamond\Diamond F \equiv \Diamond F \qquad \Diamond\Box\Diamond F \equiv \Box\Diamond F, \qquad \Box\Diamond\Box F \equiv \Diamond\Box F$$

Derived operator: Leads-to

- The **leads-to** operator expresses progress:

$$F \rightsquigarrow G \triangleq \Box(F \Rightarrow \Diamond G)$$

  It means that whenever F holds, G will eventually hold.

- Typical use: specifying that a system *makes progress* from one condition to another, e.g., a request eventually receives a response.

Fairness conditions

- Fairness ensures that actions which remain or become enabled will eventually occur.

  Notation: $\langle A \rangle_{sf} \triangleq A \land (sf' \neq sf)$.

- **Weak fairness** ($WF_{sf}(A)$):

$$WF_{sf}(A) \triangleq \Diamond\Box \,\textsc{Enabled}\, \langle A \rangle_{sf} \Rightarrow \Box\Diamond \langle A \rangle_{sf}$$

  The action A must eventually occur if it remains continuously enabled.

- **Strong fairness** ($SF_{sf}(A)$):

$$SF_{sf}(A) \triangleq \Box\big(\Box\Diamond \,\textsc{Enabled}\, \langle A \rangle_{sf} \Rightarrow \Diamond \langle A \rangle_{sf}\big)$$

  The action A must eventually occur if it is enabled infinitely often.

System specifications and properties

- In TLA, there is **no formal distinction** between system specifications and their properties: both are **temporal formulas**.
- A typical system specification has the form:

$$Init \ \wedge \ \Box[Next]_{vars} \ \wedge \ L$$

$Init$	state predicate describing initial conditions
$Next$	action describing the next-state relation; typically a disjunction of individual actions
$vars$	tuple of all flexible state variables
L	optional fairness assumptions

$Init \ \wedge \ \Box[Next]_{vars}$ will be our standard form of specification.

Safety properties

- A property F is **satisfied** by a specification S if $\models S \Rightarrow F$. That is, every behavior satisfying S also satisfies F.
- Notation: $\sigma_i|$ denotes the ω-sequence $\sigma_0\sigma_1 \ldots \sigma_i\sigma_i\sigma_i \ldots$
- A property F is a **safety property** if for all ξ and behaviors σ,

$$[\![F]\!]^{\xi}_{\sigma} \text{ holds iff for all } i \in \mathbb{N},\ [\![F]\!]^{\xi}_{\sigma_i|} \text{ holds.}$$

Intuitively: if F is violated by σ, then it is already violated by some finite prefix of σ (completed by infinite stuttering).

Invariants

- An **invariant** is expressed as $\Box P$ where P is a state predicate.
- A specification S **satisfies** the invariant P if

$$\models S \Rightarrow \Box P$$

meaning that in every behavior satisfying S, each state satisfies P.

Invariants are the most common kind of **safety property**.

Proving invariants

- The basic rule for proving an invariant P:

$$\frac{P \ \wedge \ [Next]_{vars} \ \Rightarrow \ P'}{P \ \wedge \ \Box[Next]_{vars} \ \Rightarrow \ \Box P} \ \text{(INV1)}$$

The hypothesis $P \wedge [Next]_{vars} \Rightarrow P'$ is a **transition formula**:

- Every (possibly stuttering) transition preserves P.

The conclusion $P \wedge \Box[Next]_{vars} \Rightarrow \Box P$ is a **temporal formula**:

- Every behavior which starts in a P-state and follows $Next$ or stutter transitions, is a behavior where P always holds.

Observation

$P \wedge [Next]_{vars} \Rightarrow P'$ can be proven using first-order reasoning.

Inductive invariants

- The rule (INV1) presupposes that the invariant P is **strong enough** to imply P' under every possible system transition:

$$P \ \wedge \ [\text{Next}]_{vars} \ \Rightarrow \ P'$$

  However, this need not hold for every desired invariant Q.

Inductive invariants

- In such cases, we find a strengthened **inductive invariant** P that implies the desired property Q.

$$\frac{\text{Init} \Rightarrow P \qquad P \ \wedge \ [\text{Next}]_{vars} \Rightarrow P' \qquad P \Rightarrow Q}{\text{Init} \ \wedge \ \Box[\text{Next}]_{vars} \Rightarrow \Box Q} \ (\text{INV})$$

Inductive invariants usually contain interesting design information about the model and capture the overall correctness idea.

Proof rules (simple TLA)

STL1. $\dfrac{\varphi}{\Box\varphi}$

STL2. $\Box\varphi \Rightarrow \varphi$

STL3. $\Box\Box\varphi \equiv \Box\varphi$

STL4. $\dfrac{\varphi \Rightarrow \psi}{\Box\varphi \Rightarrow \Box\psi}$

STL5. $\Box(\varphi \wedge \psi) \equiv \Box\varphi \wedge \Box\psi$

STL6. $\Diamond\Box(\varphi \wedge \psi) \equiv \Diamond\Box\varphi \wedge \Diamond\Box\psi$

TLA1. $\dfrac{P \wedge e' = e \Rightarrow P'}{\Box P \equiv P \wedge \Box[P \Rightarrow P']_e}$

TLA2. $\dfrac{P \wedge [A]_e \Rightarrow Q \wedge [B]_f}{\Box P \wedge \Box[A]_e \Rightarrow \Box Q \wedge \Box[B]_f}$

Lattice. $\dfrac{(S, \prec) \text{ is a well-founded ordering} \quad \forall x \in S : \varphi(x) \rightsquigarrow (\psi \vee \exists y \in S : y \prec x \wedge \varphi(y))}{(\exists x \in S : \varphi(x)) \rightsquigarrow \psi}$

Figure 1 Proof rules for simple TLA.

Proof rules (quantifier rules)

$$F[c/x] \Rightarrow \exists\, x \,:\, F \quad (\exists\,\mathrm{I}) \qquad \frac{F \Rightarrow G}{(\exists\, x \,:\, F) \Rightarrow G}\ (\exists\,\mathrm{E})$$

$$F[t/v] \Rightarrow \boldsymbol{\exists}\, v \,:\, F \quad (\boldsymbol{\exists}\,\mathrm{I}) \qquad \frac{F \Rightarrow G}{(\boldsymbol{\exists}\, v \,:\, F) \Rightarrow G}\ (\boldsymbol{\exists}\,\mathrm{E})$$

Sources

L. Lamport, "The Temporal Logic of Actions," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 3, pp. 872–923, May 1994. doi:10.1145/177492.177726

S. Merz, "The Specification Language TLA$^+$," in *Logics of Specification Languages*, D. Bjørner and M. C. Henson, Eds., *Monographs in Theoretical Computer Science: An EATCS Series*. Berlin, Heidelberg: Springer, 2008, pp. 401–451. doi:10.1007/978-3-540-74107-7_8

L. Lamport, *Specifying Systems: The TLA$^+$ Language and Tools for Hardware and Software Engineers*. Boston, MA: Addison–Wesley, 2003.

L. Lamport, "Safety, Liveness, and Fairness," May 26, 2019. lamport.azurewebsites.net/tla/safety-liveness.pdf.