

Temporal Logic of Actions (TLA)

Leslie Lamport

John A. Akinyemi

Department of Computer Science
University of Manitoba, Winnipeg, Canada

Course: 74.757 - Formal Logic

Instructor: Dr. Christel Kemke

Outline



- Introduction
 - Overview
 - Background Information on the Logic of Actions, Temporal Logic, and Raw Temporal Logic of Actions (RTLA)
- Temporal Logic of Actions (TLA)
 - Concepts, Symbols, Syntax, Meaning, and Examples
- Conclusion

Introduction

- Temporal logic of actions (TLA) is a logic that combines temporal logic and logic of actions for specifying and reasoning about concurrent and reactive discrete systems.
- TLA is used for program verification and proving liveness properties of programs.
- In TLA, algorithms are represented with formulas.
- Semantics of TLA formulas are built on the semantics of RTLTL formulas based on sequences of states.

Introduction

- All TLA formulas can be expressed in terms of familiar mathematical operators (e.g. \wedge) plus three additional ones, namely: Prime ($'$), always (\square), and existential quantifier (\exists).
- TLA is simple and expressive, with a minimally complex expressive power.
- Elementary formulas in TLA are actions.

TLA Supporting Tools

- TLA+ is a **specification language** based on TLA.
- TLP is a system for **mechanically checking** TLA proofs (**program verification**).
- The three available **TLA+ tools** are:
 - TLATeX, a program for **typesetting** TLA+ specs.
 - The Syntactic Analyzer, a **parser** and **syntax checker** for TLA+ specifications.
 - TLC, a **model checker** and **simulator** for a **subclass of "executable"** TLA+ specifications.

TLA Application & Usage

- TLA has been used in the following systems:
 - DisCo (Distributed Co-operation) [4], a formal specification method for reactive systems has its logical foundation on TLA.
 - Isabelle [5], a theorem prover.
 - Algorithm verifications, and specification and analysis of aircraft systems and joint human-machine tasks in aviation [6]
 - Specification and Verification in TLA of RLP1, the data link layer protocol of TDMA mobile cellular phone systems [6]

TLA Building Blocks

- Actions: $\mathcal{A}, \mathcal{M}, \mathcal{M}_1, \mathcal{M}_2$
- Predicates: P, Init_ϕ
- Variables: x, y, hr
- Primed variables: x', y', hr'
- States: s, s'
- State function: f
- Behavior: $\langle s_0, s_1, \dots, s_n \rangle$
- Values: Data items, e.g. **Integers, constants**
- Semantics: $\llbracket f \rrbracket, \llbracket hr \rrbracket, s \llbracket f \rrbracket, s \llbracket hr \rrbracket$
- Formulas: **F, G, Φ**
- Operators: $\Box, \Diamond, \sim, \neg, \vee, \wedge$
- Symbols: $\triangleq, \langle, \rangle, ', [,], (,), =, \neq, \equiv, \in$
- Quantifiers: \forall, \exists

Basic Ideas

- Formulas in the Logic of Actions are built using:
 - Values, Variables, States, State functions, and Actions
- Temporal Logic (TL) is a class of logic that models reasoning about sequences of states (the logic of time).
- Raw Temporal Logic of Actions (RTL_A) is a logic of actions. Elementary formulas are actions.
- Temporal Logic of Actions (TL_A) is a TL-based specification language built on RTL_A.

Definitions

- A **state** is an **assignment of values to variables**, e.g. assigning **value 22** to **variable *hr*** is a **state** in the **clock system**. A state is **a mapping from variables to values**.
 - The meaning $\llbracket hr \rrbracket$ of the variable *hr* is **a mapping from states to values**. $\llbracket hr \rrbracket \triangleq (hr=22) \rightarrow 10p.m.$
- An **action** is a boolean-valued expression consisting of **variables**, **primed variables**, and **constant symbols**.
 - An action shows a relation between an old and a new state. **Example** of an action:
$$hr' = hr + 1 \quad \equiv \quad hr'(23) = hr(22) + 1$$

Definitions

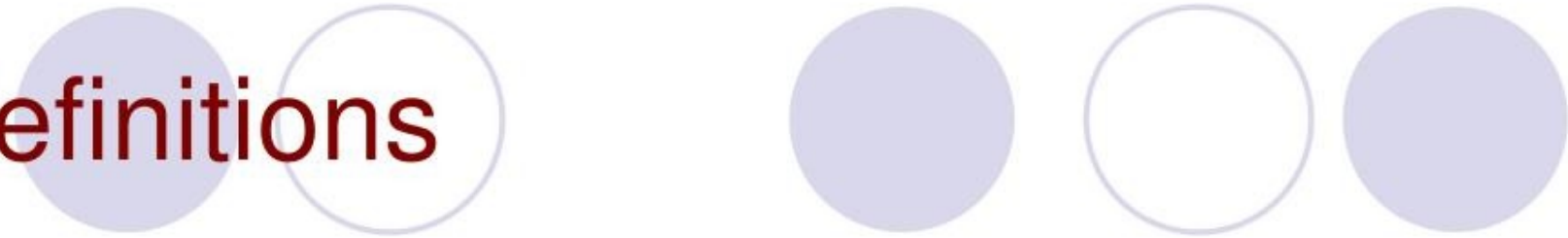
- A **state function** is a non-boolean expression built from **variables** and **constant** symbols.
 - The meaning $\llbracket f \rrbracket$ of a state function $f(hr) = hr + 1$ with **variable** hr and **constant** 1 (denoting 1 hour), is **a mapping from the collection of states (hr) to a collection of values**,
 - **Example**: $\llbracket hr+1 \rrbracket$ is the mapping that assigns to a state hr the value $\llbracket hr \rrbracket + 1$.
 - If $s \llbracket f \rrbracket$ is the value that $\llbracket f \rrbracket$ assigns to any state s , semantically,

$$s \llbracket f \rrbracket \triangleq f (\forall hr : s \llbracket hr \rrbracket / hr)$$

If $s \llbracket hr \rrbracket = 10p.m.$

then $s \llbracket f \rrbracket = s \llbracket hr \rrbracket + 1 = 10p.m.+1 = 11p.m.$

Definitions



<u>Variable</u>	<u>Value</u>	<u>State</u>	<u>Value</u>
hr	20	hr=20	8p.m.
	22	hr=22	10p.m.
	23	hr=23	11p.m.

Diagram illustrating the mapping between variables, values, and states. Arrows show that the variable 'hr' is associated with values 20, 22, and 23. These values are mapped to states 'hr=20', 'hr=22', and 'hr=23' respectively. The states are then mapped to values '8p.m.', '10p.m.', and '11p.m.'. A red oval highlights the state 'hr=22', and a red arrow points from it to the state 'hr=22' in the semantic evaluation below.

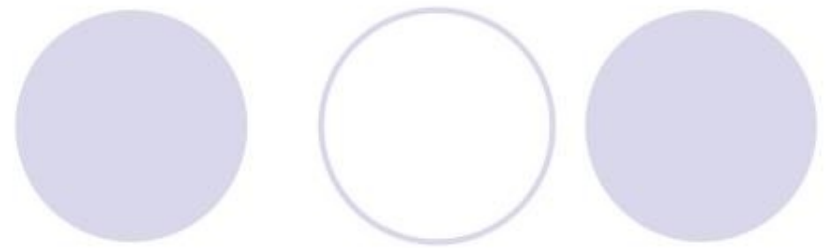
Semantics of state function (f), $s \llbracket f \rrbracket$ is:

$s \llbracket (hr=22) \rrbracket \longrightarrow 10p.m.$

Therefore,

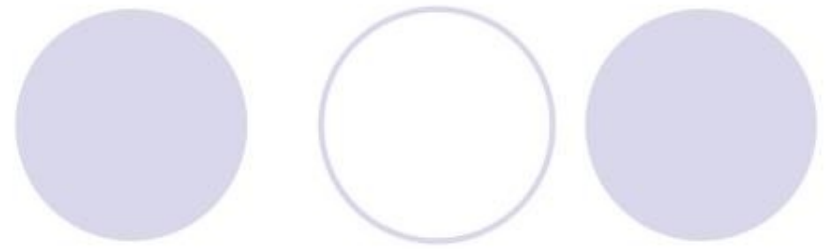
$$s \llbracket f \rrbracket = s \llbracket hr \rrbracket + 1 = 10p.m. + 1 = 11p.m.$$

Temporal Logic



- Example: In a clock system:
 - Statement 1: 1 hour = 60 minutes
 - Statement 2: The current hour is 3
- In FOPL, the 2 statements are of the form:
 - 1) for all time t , $1 \text{ hour} = 60 \text{ minutes}$ at time t is true.
 - 2) for some time t , $\text{hour} = 3$ at time t is true.
- TL eliminates a continuous dependence of a statement on time variables.
- TL uses \Box (always), and \Diamond (eventually) as primitive operators to implicitly describe timed statements.

Temporal Logic



- TL is used to describe **dynamic** behavior of programs.
- TL is used to **formulate properties** of **reactive programs** which do not compute an 'answer', but are intended to **run indefinitely** and still **correctly** exhibit dynamic behavior in response to **external stimuli**.
- A good example is the **clock system**.
Hour, $hr \in \{1 \dots 12\}$; Minute, $min \in \{1 \dots 60\}$
if $hr < 12$ then $hr' = hr + 1$ else $hr' = 1$
*where hr' is the **next hour**. Property minute can change while the hour remains the same.*

Temporal Logic

- TL allows reasoning about sequences of states.
- A temporal formula is built from elementary formulas using boolean operators and unary operators \Box and \Diamond .

○ For example, if F and G are temporal formulas, then

$$\Box F, \Diamond F, \Box G, \Diamond G, \neg F, \neg G, F \wedge G, F \vee G$$

are temporal formulas

- Eventually: $\Diamond F \triangleq \neg \Box \neg F$.
- Eventually always: $\Diamond \Box F$.
 - an assertion that eventually F is always true.
- Leads to: $F \leadsto G \equiv \Box(F \Rightarrow \Diamond G)$

Syntax of TLA

$\langle \text{formula} \rangle \triangleq \langle \text{predicate} \rangle \mid \Box[\langle \text{action} \rangle]_{\langle \text{state function} \rangle}$
 $\mid \neg \langle \text{formula} \rangle \mid \langle \text{formula} \rangle \wedge \langle \text{formula} \rangle$
 $\mid \Box \langle \text{formula} \rangle$

$\langle \text{action} \rangle \triangleq$ boolean-valued expression containing
constant symbols, variables, and primed
variables

$\langle \text{predicate} \rangle \triangleq \langle \text{action} \rangle$ with no primed variables
 $\mid \text{Enabled } \langle \text{action} \rangle$

$\langle \text{state function} \rangle \triangleq$ nonboolean expression containing
constant symbols and variables

Semantics of Temporal Logic

- The definition of $\llbracket \Box F \rrbracket$ in terms of $\llbracket F \rrbracket$ where $\langle s_0, s_1, s_2, \dots \rangle$ represent the behavior is:

$$\langle s_0, s_1, s_2, \dots \rangle \llbracket \Box F \rrbracket \triangleq \forall n \in \text{Nat} : \langle s_n, s_{n+1}, s_{n+2}, \dots \rangle \llbracket F \rrbracket$$

- The definition of $\llbracket \Diamond F \rrbracket$ in terms of $\llbracket F \rrbracket$ where $\langle s_0, s_1, s_2, \dots \rangle$ represent the behavior is:

$$\langle s_0, s_1, s_2, \dots \rangle \llbracket \Diamond F \rrbracket \equiv \exists n \in \text{Nat} : \langle s_n, s_{n+1}, s_{n+2}, \dots \rangle \llbracket F \rrbracket$$

- Infinitely often (always eventually): $\Box \Diamond F$.

$$\langle s_0, s_1, \dots \rangle \llbracket \Box \Diamond F \rrbracket \equiv \forall n \in \text{Nat} : \exists m \in \text{Nat} : \langle s_{n+m}, s_{n+m+1}, \dots \rangle \llbracket F \rrbracket$$

Semantics of TLA & RTLA Formulas

- RTLA formulas are built from actions that use **logical operators** (e.g. \wedge) and the **temporal operator** \Box . Thus, for predicate $Init_\phi$ asserting the initial condition in the formula Φ , and action \mathcal{A}
 1. $\Box \mathcal{A}$ and
 2. $\Phi \triangleq Init_\phi \wedge \Box \mathcal{A}$ are RTLA formulas.
- TLA derives meaning from the semantics of RTLA
- $\llbracket \mathcal{A} \rrbracket$ represent the meaning of an action \mathcal{A} , a boolean-valued function that assigns the value $s \llbracket \mathcal{A} \rrbracket s'$ to the pair of states s, s' .

Semantics of TLA & RTLA Formulas

- Step of an action \mathcal{A} (“ \mathcal{A} step”). A pair of states s, s' is an “ \mathcal{A} step” iff $s \llbracket \mathcal{A} \rrbracket s'$ is True.
 - s' is the value of s in the final state of a step.
 - A behavior satisfies $\Box[\mathcal{A}]_f$ iff every step of the behavior is an \mathcal{A} step.
 - a behavior satisfies a predicate P iff the first state of the behavior satisfies P .
 - A behavior satisfies $\Box P$ iff all states in the behavior satisfy P .
- $\llbracket \mathcal{A} \rrbracket$ is true for a behavior iff the first pair of states in the behavior is an \mathcal{A} step.

$$\langle s_0, s_1, s_n, \dots \rangle \llbracket \mathcal{A} \rrbracket \triangleq s_0 \llbracket \mathcal{A} \rrbracket s_1$$

Semantics of TLA & RTLA Formulas

- If \mathcal{A} is an action, $\Box \mathcal{A}$ is an RTLA formula
- A formal description of the meaning of $\Box \mathcal{A}$ is as follows:
$$\langle s_0, s_1, s_2, \dots \rangle \llbracket \Box \mathcal{A} \rrbracket \equiv \forall n \in \text{Nat} : \langle s_n, s_{n+1}, s_{n+2}, \dots \rangle \llbracket \mathcal{A} \rrbracket$$
$$\equiv \forall n \in \text{Nat} : s_n \llbracket \mathcal{A} \rrbracket s_{n+1}$$
- if P is a predicate, then $s \llbracket P \rrbracket t$ equals $s \llbracket P \rrbracket$.

Therefore,

$$\langle s_0, s_1, \dots \rangle \llbracket P \rrbracket \equiv s_0 \llbracket P \rrbracket$$
$$\langle s_0, s_1, \dots \rangle \llbracket \Box P \rrbracket \equiv \forall n \in \text{Nat} : s_n \llbracket P \rrbracket$$

- A behavior satisfies a predicate P iff the **first state** of the behavior satisfies P .
- A behavior satisfies $\Box P$ iff **all states** in the behavior satisfy P .

Problem Description

This program initially sets x and y to 0, and repeatedly increments x or y (in a single operation), choosing nondeterministically between them.

```
var natural  $x, y = 0$  ,  
do { true  $\rightarrow x := x + 1$  }  
  □  
  { true  $\rightarrow y := y + 1$  } od
```

Figure 1: A program written in a conventional language

Raw Temporal Logic of Actions (RTL_A)

- RTL_A formulas Φ of the program in Figure 1.

$$Init_{\Phi} \triangleq (x = 0) \wedge (y = 0)$$

$$\mathcal{M}_1 \triangleq (x' = x + 1) \wedge (y' = y) \quad \mathcal{M}_2 \triangleq (y' = y + 1) \wedge (x' = x)$$

$$\mathcal{M} \triangleq \mathcal{M}_1 \vee \mathcal{M}_2$$

$$\Phi \triangleq Init_{\Phi} \wedge \Box \mathcal{M}$$

Figure 2: An RTL_A Description of the Program in Figure 1

- TLA formulas are subsets of RTL_A formulas.
- Elementary formulas in TLA are predicates and formulas of the form $Init_{\Phi} \wedge \Box[\mathcal{A}]_f$
where $[\mathcal{A}]_f \triangleq [\mathcal{A} \vee (f' = f)]$ and
Predicate, $Init_{\Phi} \triangleq (x = 0) \wedge (y = 0)$

TLA Concepts

- **Stuttering step.** A stuttering step on an action \mathcal{A} under the vector variables f occurs when either the action \mathcal{A} occurs or the variables in f is unchanged.

○ *Example:* In the Clock system, **hour** can **stutter** while its **seconds** are executed.

- **The stuttering operator**

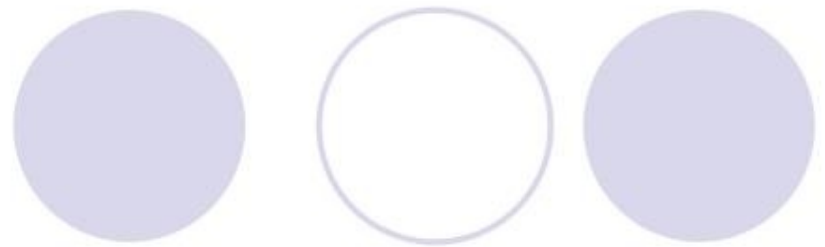
$$[\mathcal{A}]_f \triangleq \mathcal{A} \vee (f' = f)$$

TLA Concepts

- **Safety Property:** Assertion of things that must not happen. This specifies constraints in the program. Stuttering operator describes safety property.
- **Liveness:** Assertion that something must eventually happen. It prevents a program from satisfying the initial condition only, and not implementing any other action.

$$\langle \mathcal{A} \rangle_f \triangleq [\mathcal{A} \wedge (f' \neq f)]$$

TLA Concepts



- Fairness describes a cautious specification of liveness, by avoiding a liveness that results in a safety property.
 - Assertion that if a certain operation is possible, then the program must eventually execute it.
- Weak fairness of action \mathcal{A} , $WF_f(\mathcal{A})$ – asserts that an operation must be executed if it remains possible to do so for a long enough time.
- Strong fairness of action \mathcal{A} , $SF_f(\mathcal{A})$ – asserts that an operation must be executed if it is often enough (eventually always) possible to do so.

TLA Concepts

- $WF_f(\mathcal{A})$ is satisfied by a behavior iff

$$\mathcal{A} \wedge (f' \neq f)$$

is infinitely often not enabled, or infinitely many

$$\mathcal{A} \wedge (f' \neq f) \text{ steps occur.}$$

- $SF_f(\mathcal{A})$ is satisfied by a behavior iff

$$\mathcal{A} \wedge (f' \neq f)$$

is only finitely often enabled, or infinitely many

$$\mathcal{A} \wedge (f' \neq f) \text{ steps occur.}$$

weak fairness: $(\Box\Diamond \text{ executed}) \vee (\Box\Diamond \text{ impossible})$

strong fairness: $(\Box\Diamond \text{ executed}) \vee (\Diamond\Box \text{ impossible})$

$$WF_f(\mathcal{A}) \triangleq (\Box\Diamond \langle \mathcal{A} \rangle_f) \vee (\Box\Diamond \neg \text{Enabled } \langle \mathcal{A} \rangle_f)$$

$$SF_f(\mathcal{A}) \triangleq (\Box\Diamond \langle \mathcal{A} \rangle_f) \vee (\Diamond\Box \neg \text{Enabled } \langle \mathcal{A} \rangle_f)$$

TLA Concepts

- $F \leadsto G$. **Leads to**: (whenever F is true, G will eventually become true).

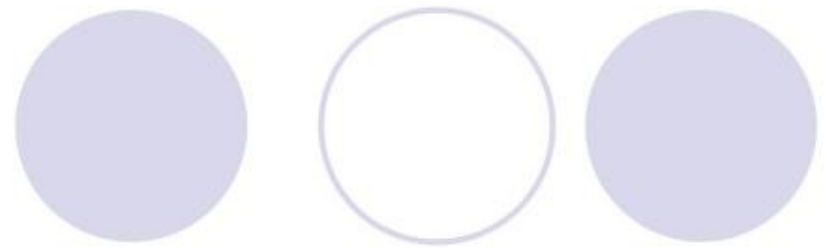
$$F \leadsto G \equiv \Box(F \Rightarrow \Diamond G)$$

- $F \Vdash G$. F guarantees G iff G is true for at least as long as (whenever) F is true.

○ It asserts that a system operates correctly if the environment does. It means:

- (i) F implies G , and
- (ii) no step can make G false unless F is made false.

TLA Concepts



- $\exists x : F$. (Hiding): satisfied by a behavior iff there are some values that can be assigned to x to produce a behavior satisfying F .
 - F asserts that irrespective of the values of x , x can have some values that make F hold.
- $F \Rightarrow G$. (F implements G) iff every behavior of a system that satisfies F also satisfies G .

An Example of TLA Programs

- A TLA Description of the Program in Figure 1

$$\Phi \triangleq \text{Init}_\Phi \wedge \Box[\mathcal{M}]_{\langle x, y \rangle}$$

- Adding **Liveness** to the TLA Formula

$$\Phi \triangleq \text{Init}_\Phi \wedge \Box[\mathcal{M}]_{\langle x, y \rangle} \wedge \Box\Diamond\langle \mathcal{M}_1 \rangle_{\langle x, y \rangle} \wedge \Box\Diamond\langle \mathcal{M}_2 \rangle_{\langle x, y \rangle}$$

Note: $\Box\Diamond\langle \mathcal{M} \rangle_{\langle x, y \rangle}$ equals $\text{WF}_{\langle x, y \rangle}(\mathcal{M})$

- Adding **Fairness** to the TLA Formula

$$\Phi \triangleq \text{Init}_\Phi \wedge \Box[\mathcal{M}]_{\langle x, y \rangle} \wedge \text{WF}_{\langle x, y \rangle}(\mathcal{M})$$

Limitation of TLA

- TLA properties are true or false for an individual behavior.
- It cannot express statistical properties of sets of behaviors, for example, that the program has probability greater than .99 of terminating.

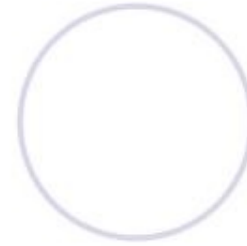
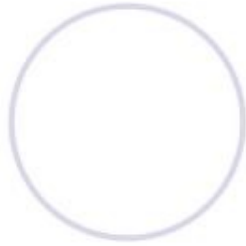
Conclusion



- TLA formulas semantically follows the semantics of RTLA - a logic of actions.
- TLA is a language for writing predicates, state functions, and actions, and a logic for reasoning about them.
- TLA is useful for specifying and verifying safety and liveness properties of discrete systems.
- TLA has tools that aid program specifications and verifications.

Conclusion

- A **safety property** asserts all constraints that ensure the **system does not enter an undesired state**, and a **liveness property** asserts that the system performs all specified actions.
- TLA makes it practical to describe a system by a single formula.
- TLA can be used to **formalize the transitions and evolution of states** in a dynamic system, e.g. I intend to use TLA to formalize the UML State diagrams in my thesis.



Thank you

References

1. Leslie Lamport. Introduction to TLA. *Technical Report # 1994-001*, Digital Systems Research Center, 1994. Available at <http://www.research.digital.com/SRC/>
2. Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*, Addison-Wesley, 2003.
3. Leslie Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872-923, May 1994.
4. DisCo. <http://disco.cs.tut.fi/index.html>
5. TLA. <http://research.microsoft.com/users/lamport/tla/tla.html>
6. Work With and On Lamport's TLA. <http://www.rvs.uni-bielefeld.de/publications/abstracts.html#TLA>