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 RTLA 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 ($\Box$), 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, $Init_{\Phi}$
- Variables: x, y, hr
- Primed variables: x', y', hr'
- States: s, s'
- State function: f
- Behavior: $\langle s_0, s_1, \ldots, s_n \rangle$
- Values: Data items, e.g. Integers, constants
- Semantics: $[\![f]\!]$, $[\![hr]\!]$, $s\, [\![f]\!]$, $s\, [\![hr]\!]$
- Formulas: F, G, Φ
- Operators: $\Box$, $\Diamond$, $\leadsto$, $\neg$, $\vee$, $\wedge$
- 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 (RTLA) is a logic of actions. Elementary formulas are actions.
- Temporal Logic of Actions (TLA) is a TL-based specification language built on RTLA.

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 $[\![hr]\!]$ of the variable *hr* is a mapping from states to values. $[\![hr]\!] \triangleq (hr{=}22) \rightarrow 10p.m.$
- 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 $[\![f]\!]$ 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: $[\![hr+1]\!]$ is the mapping that assigns to a state hr the value $[\![hr]\!] + 1$.
  - If $s\,[\![f]\!]$ is the value that $[\![f]\!]$ assigns to any state s, semantically,

$$s\,[\![f]\!] \triangleq f\,(\forall hr : s\,[\![hr]\!]\,/\,hr)$$

If $s\,[\![hr]\!] = 10p.m.$

then $s\,[\![f]\!] = s\,[\![hr]\!] + 1 = 10p.m. + 1 = 11p.m.$

Definitions

Variable	Value	State	Value
hr	20	hr=20	8p.m.
	22	hr=22	10p.m.
	23	hr=23	11p.m.

Semantics of state function (f), $s[\![f]\!]$ is:

$$s[\![(hr=22)]\!] \longrightarrow 10p.m.$$

Therefore,

$$s[\![f]\!] = s[\![hr]\!] + 1 = 10p.m.+1 = 11p.m.$$

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 hour = 60 minutes* at time *t* is true.
  - 2) for some time *t*, *hour = 3* at time *t* is true.
- TL eliminates a continuous dependence of a statement on time variables.
- TL uses □ (always), and ◊ (eventually) as primitive operators to implicitly describe timed statements.

Temporal Logic

- TL is used to describe dynamic bahavior 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, $hr \in \{1...12\}$; Minute, $min \in \{1...60\}$

  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

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

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

$\langle\text{predicate}\rangle \triangleq \langle\text{action}\rangle$ with no primed variables
$\quad \mid$ 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 $[\![\Box F]\!]$ in terms of $[\![F]\!]$ where $\langle s_0, s_1, s_2, \ldots \rangle$ represent the behavior is:

$$\langle s_0, s_1, s_2, \ldots \rangle [\![\Box F]\!] \triangleq \forall n \in \mathrm{Nat} : \langle s_n, s_{n+1}, s_{n+2}, \ldots \rangle [\![F]\!]$$

- The definition of $[\![\Diamond F]\!]$ in terms of $[\![F]\!]$ where $\langle s_0, s_1, s_2, \ldots \rangle$ represent the behavior is:

$$\langle s_0, s_1, s_2, \ldots \rangle [\![\Diamond F]\!] \equiv \exists n \in \mathrm{Nat} : \langle s_n, s_{n+1}, s_{n+2}, \ldots \rangle [\![F]\!]$$

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

$$\langle s_0, s_1, \ldots \rangle [\![\Box\Diamond F]\!] \equiv \forall n \in \mathrm{Nat} : \exists m \in \mathrm{Nat} : \langle s_{n+m}, s_{n+m+1}, \ldots \rangle [\![F]\!]$$

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 $\mathring{\mathcal{A}}$
  1. $\Box\mathring{\mathcal{A}}$ and
  2. $\Phi \triangleq Init_{\Phi} \wedge \Box\mathring{\mathcal{A}}$ are RTLA formulas.
- TLA derives meaning from the semantics of RTLA
- $[\![\mathring{\mathcal{A}}]\!]$ represent the meaning of an action $\mathring{\mathcal{A}}$, a boolean-valued function that assigns the value $s\ [\![\mathring{\mathcal{A}}]\!]\ s'$ to the pair of states s, s'.

Semantics of TLA & RTLA Formulas

- Step of an action $\mathring{\mathcal{A}}$ ("$\mathring{\mathcal{A}}$ step"). A pair of states s, s' is an "$\mathring{\mathcal{A}}$ step" iff $s \ [\![\mathring{\mathcal{A}}]\!] \ s'$ is True.
  - s' is the value of s in the final state of a step.
  - A behavior satisfies $\Box[\mathring{\mathcal{A}}]_f$ iff every step of the behavior is an $\mathring{\mathcal{A}}$ 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.
- $[\![\mathring{\mathcal{A}}]\!]$ is true for a behavior iff the first pair of states in the behavior is an $\mathring{\mathcal{A}}$ step.

$$\langle s_0, s_1, s_{n, \ldots} \rangle \ [\![\mathring{\mathcal{A}}]\!] \triangleq s_0 \ [\![\mathring{\mathcal{A}}]\!] \ s_1$$

Semantics of TLA & RTLA Formulas

- If $\mathring{\mathcal{A}}$ is an action, $\Box\mathring{\mathcal{A}}$ is an RTLA formula
- A formal description of the meaning of $\Box\mathring{\mathcal{A}}$ is as follows:

$$\langle s_0, s_1, s_2, \ldots \rangle \; [\![\Box\mathring{\mathcal{A}}]\!] \equiv \forall n \in Nat : \langle s_n, s_{n+1}, s_{n+2}, \ldots \rangle \; [\![\mathring{\mathcal{A}}]\!]$$

$$\equiv \forall n \in Nat : s_n \; [\![\mathring{\mathcal{A}}]\!] \; s_{n+1}$$

- if P is a predicate, then $s \; [\![P]\!] \; t$ equals $s \; [\![P]\!]$.

  Therefore,

$$\langle s_0, s_1, \ldots \rangle \; [\![P]\!] \equiv s_0 \; [\![P]\!]$$

$$\langle s_0, s_1, \ldots \rangle \; [\![\Box P]\!] \equiv \forall n \in Nat : s_n \; [\![P]\!]$$

- A behavior satisfies a predicate P iff the first state of the behavior satises 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.

$$
\begin{array}{l}
\mathbf{var\ natural}\ \ x, y = 0\ , \\
\mathbf{do}\ \ \langle\ \mathsf{true} \rightarrow x := x + 1\ \rangle \\
\qquad \Box \\
\qquad\ \ \langle\ \mathsf{true} \rightarrow y := y + 1\ \rangle \qquad \mathbf{od}
\end{array}
$$

Figure 1: A program written in a conventional language

Raw Temporal Logic of Actions (RTLA)

- RTLA formulas Φ of the program in Figure 1.

$$Init_\Phi \triangleq (x = 0) \wedge (y = 0)$$
$$\mathcal{M}_1 \triangleq (x' = x + 1) \wedge (y' = y) \qquad \mathcal{M}_2 \triangleq (y' = y + 1) \wedge (x' = x)$$
$$\mathcal{M} \triangleq \mathcal{M}_1 \vee \mathcal{M}_2$$
$$\Phi \triangleq Init_\Phi \wedge \Box\mathcal{M}$$

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

- TLA formulas are subsets of RTLA formulas.
- Elementary formulas in TLA are predicates and formulas of the form $Init_\Phi \wedge \Box[\mathring{\mathcal{A}}]_f$

  where $[\mathring{\mathcal{A}}]_f \equiv [\mathring{\mathcal{A}} \vee (f' = f)]$ and

  Predicate, $Init_\Phi \triangleq (x = 0) \wedge (y = 0)$

TLA Concepts

- Stuttering step. A stuttering step on an action $\mathring{\mathcal{A}}$ under the vector variables f occurs when either the action $\mathring{\mathcal{A}}$ occurs or the variables in f is unchanged.
  - *Example*: In the Clock system, hour can stutter while its seconds are executed.
- The stuttering operator

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

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.

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

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 $\mathcal{\mathring{A}}$, $WF_f(\mathcal{\mathring{A}})$ – asserts that an operation must be executed if it remains possible to do so for a long enough time.
- Strong fairness of action $\mathcal{\mathring{A}}$, $SF_f(\mathcal{\mathring{A}})$ – asserts that an operation must be executed if it is often enough (eventually always) possible to do so.

TLA Concepts

- $\mathrm{WF}_f(\mathring{\mathcal{A}})$ is satisfied by a behavior iff

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

  is infinitely often not enabled, or infinitely many

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

- $\mathrm{SF}_f(\mathring{\mathcal{A}})$ is satisfied by a behavior iff

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

  is only finitely often enabled, or infinitely many

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

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

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

$$\mathrm{WF}_f(\mathcal{A}) \triangleq (\Box\Diamond\langle\mathcal{A}\rangle_f) \vee (\Box\Diamond\neg \mathit{Enabled}\ \langle\mathcal{A}\rangle_f)$$

$$\mathrm{SF}_f(\mathcal{A}) \triangleq (\Box\Diamond\langle\mathcal{A}\rangle_f) \vee (\Diamond\Box\neg \mathit{Enabled}\ \langle\mathcal{A}\rangle_f)$$

TLA Concepts

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

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

- $F \stackrel{+}{\rhd} G$. 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

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

- Adding Liveness to the TLA Formula

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

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

- Adding Fairness to the TLA Formula

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

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