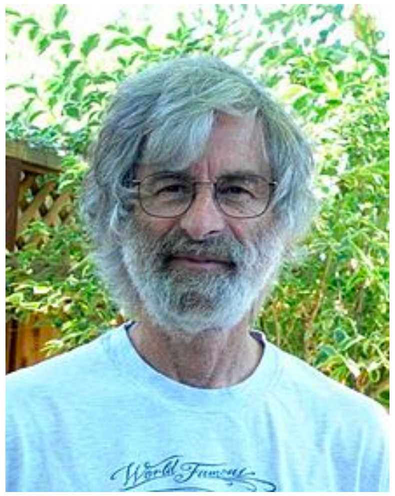
Introduction to Specification

Nebulas Research
Zaiyang Tang

NEBULAS

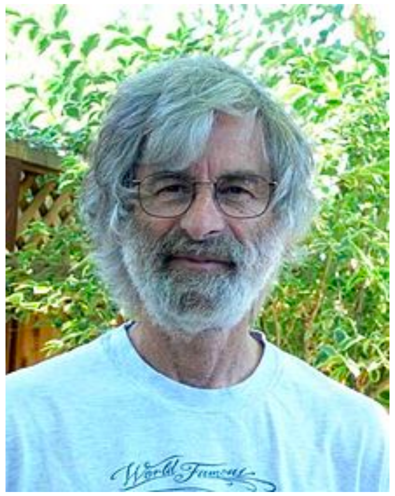
What is TLA and TLA+?

- In the late 1980's, Leslie Lamport invented TLA, the **Temporal Logic of Actions**—a simple variant of Pnueli's original temporal logic.



NEBULAS

What is TLA and TLA+?



- In the late 1980's, Leslie Lamport invented TLA, the **Temporal Logic of Actions**—a simple variant of Pnueli's original temporal logic.
- TLA provides a mathematical foundation for describing systems, and the complete language built atop that foundation is TLA+.
- TLA+ is a language for high-level modeling of digital systems.
- The tool most commonly used by engineers are the TLC model checker, and the TLA+ proof system TLAPS.

NEBULAS

The Industrial Use

- **Intel**: Pre-RTL formal verification;
- **Amazon**:
  1. Amazon has used TLA+ on more than 14 large complex systems;
  2. Amazon Web Services has been using TLA+ since 2011.
- **Microsoft**:
  1. TLA+ was used sporadically at Microsoft beginning around 2004 (Xbox360);
  2. Starting around 2015, use at Microsoft increased especially in Azure.
- ...

NEBULAS

What is Specification?

- A specification is a ***written description*** of what a system is supposed to do. Precise high-level models are called specifications.

- It’s a good idea to understand a system before building it, so it’s a good idea to write a specification of a system before implementing it.

- TLA+ can specify algorithms and high-level designs.

Learning Resources

- Learning TLA+: https://lamport.azurewebsites.net/tla/learning.html
  - TLA+ Video Course： https://lamport.azurewebsites.net/video/videos.html
  - The TLA+ BOOK & The TLA Hyperbook： https://lamport.azurewebsites.net/tla/learning.html
- Examples: https://github.com/tlaplus/Examples
  - Paxos: https://github.com/tlaplus/Examples/tree/master/specifications/Paxos
  - Raft: https://github.com/tlaplus/Examples/tree/master/specifications/raft
  - Byzantine Paxos Algorithm: https://lamport.azurewebsites.net/tla/byzpaxos.html
- *The Writings of Leslie Lamport： https://blog.bigchaindb.com/the-writings-of-leslie-lamport-abridged-a67df77f464
- Video of Byzantizing Paxos by Refinement： https://www.microsoft.com/en-us/research/video/dr-tla-series-byzantine-paxos/

NEBULAS

Basic Math: Sets

- Set theory is the foundation of ordinary mathematics and TLA.
- A set is often described as a collection of elements.
- A set can have a finite or infinite number of elements.

NEBULAS

Basic Math: Sets

- Set theory is the foundation of ordinary mathematics and TLA.
- A set is often described as a collection of elements.
- A set can have a finite or infinite number of elements.

The most common operations on sets are

$S \cap T$	The set of elements in both S and T. $\{1, -1/2, 3\} \cap \{1, 2, 3, 5, 7\} = \{1, 3\}$
$S \cup T$	The set of elements in S or T (or both). $\{1, -1/2\} \cup \{1, 5, 7\} = \{1, -1/2, 5, 7\}$
$S \subseteq T$	True iff every element of S is an element of T. $\{1, 3\} \subseteq \{3, 2, 1\}$
$S \setminus T$	The set of elements in S that are not in T. $\{1, -1/2, 3\} \setminus \{1, 5, 7\} = \{-1/2, 3\}$

NEBULAS

Basic Math: Sets

Other powerful operators of set theory:

UNION S	The union of the elements of S. In other words, a value e is an element of UNION S iff it is an element of an element of S. For example: $$\text{UNION } \{\{1,2\},\{2,3\},\{3,4\}\} = \{1,2,3,4\}$$
SUBSET S	The set of all subsets of S. In other words, $T \in$ SUBSET S iff $T \subseteq S$. For example: $$\text{SUBSET } \{1,2\} = \{\{\},\{1\},\{2\},\{1,2\}\}$$

$Cardinality(S)$	The number of elements in set S, if S is a finite set.
$IsFiniteSet(S)$	True iff S is a finite set.

NEBULAS

Basic Math: Sets

Two important constructs of set:

$\{x \in S : p\}$	The subset of S consisting of all elements x satisfying property p. For example, the set of odd natural numbers can be written $\{n \in Nat : n \% 2 = 1\}$. The identifier x is bound in p; it may not occur in S.
$\{e : x \in S\}$	The set of elements of the form e, for all x in the set S. For example, $\{2 * n + 1 : n \in Nat\}$ is the set of all odd natural numbers. The identifier x is bound in e; it may not occur in S.

NEBULAS

Basic Math: Logic

1. Boolean Values
2. Propositional Logic
3. Predicate Logic
4. CHOOSE

NEBULAS

Basic Math: Propositional Logic

- Elementary algebra: real numbers and operators (+, -, * and /)
- Propositional logic: Boolean values (TRUE and FALSE) and five operators

$\wedge$ conjunction (and)	$\Rightarrow$ implication (implies)
$\vee$ disjunction (or)	$\equiv$ equivalence (is equivalent to)
$\neg$ negation (not)	

NEBULAS

Basic Math: Propositional Logic

- Elementary algebra: real numbers and operators (+, -, * and /)
- Propositional logic: Boolean values (TRUE and FALSE) and five operators

$\wedge$	conjunction (and)	$\Rightarrow$	implication (implies)
$\vee$	disjunction (or)	$\equiv$	equivalence (is equivalent to)
$\neg$	negation (not)		

F	G	$F \Rightarrow G$	$\neg F$	$\neg F \vee G$	$(F \Rightarrow G) \equiv \neg F \vee G$
TRUE	TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE

Basic Math: Predicate Logic

- How to describe “some formula is true for all the elements of a set, or for some of the elements of a set”?
- Predicate logic extends propositional logic with the quantifiers on sets.

$\forall$	universal quantification (for all)
$\exists$	existential quantification (there exists)

NEBULAS

Basic Math: Predicate Logic

- How to describe "some formula is true for all the elements of a set, or for some of the elements of a set"?

- Predicate logic extends propositional logic with the quantifiers on sets.

$\forall$	universal quantification (for all)
$\exists$	existential quantification (there exists)

e.g., $\exists n \in Nat : n^2 = 2$ asserts that there exists a natural number n whose square equals 2. This formula happens to be false.

Basic Math: CHOOSE

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

- The expression CHOOSE x : F equals an arbitrarily chose value x that satisfies the formula F.
- If no such x exists, the expression has a completely arbitrary value.

Basic Math: Functions

- An assignment of elements of x to elements of e.
- In a programming language, such an assignment is called an array of type x indexed by e.
- In mathematics, it's called a function from x to e.

$f[e]$	[Function application]
$DOMAIN\ f$	[Domain of function]
$[x \in S \mapsto e]$	[Function f with such that $f[x] = e$ for $x \in S$]
$[S \to T]$	[Set of functions f with $f[x] \in T$ for $x \in S$]
$[f\ EXCEPT\ ![e_1] = e_2]$	[Function $\hat{f}$ equal to f except $\hat{f}[e_1] = e_2$]

The Syntax of TLA+: Operators

Constant Operators

Logic

$\land \quad \lor \quad \neg \quad \Rightarrow \quad \equiv$	
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 $\widehat{f}[e_1] = e_2$]

NEBULAS

The Syntax of TLA+: Operators

Constant Operators

Records

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

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)]

NEBULAS

The Syntax of TLA+: Operators

Constant Operators

Conditional Constructs

Let/In Construct

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]

$$\begin{array}{l} \land \; p_1 \\ \quad \vdots \\ \land \; p_n \end{array} \quad \text{[the conjunction } p_1 \land \ldots \land p_n\text{]}$$

$$\begin{array}{l} \lor \; p_1 \\ \quad \vdots \\ \lor \; p_n \end{array} \quad \text{[the disjunction } p_1 \lor \ldots \lor p_n\text{]}$$

NEBULAS

What is Temporal Logic?

In logic, temporal logic is any system of rules and symbolism for representing, and reasoning about, propositions qualified in terms of time.

- “I am hungry”
- ”I am ***always*** hungry”
- “I will ***eventually*** be hungry”
- “I will be hungry ***until*** I eat something"

NEBULAS

State Machine

An execution of a system is represented as a **sequence of discrete steps**.

State Machine

An execution of a system is represented as a **sequence of discrete steps**.
How to describe (or specify) all possible executions (or behaviors) of a system?

- Programming languages
- Turing machines
- Many different kinds of automata
- Hardware description languages

State Machine

An execution of a system is represented as a **sequence of discrete steps**.

How to describe (or specify) all possible executions (or behaviors) of a system?

- Programming languages
- Turing machines
- Many different kinds of automata
- Hardware description languages

Use state machine

A state machine is described by:

1. All possible initial states.
2. What next states can follow any given state.

State Machine in TLA+

Terminology

- State: a state is an assignment of values to variables.
- Behavior (execution): a behavior is any infinite sequence of states.
- State function: a state function is an expression that is built from declared variables, declared constants, and constant operators.
- State predicate: a state predicate is a Boolean-valued state function.

NEBULAS

State Machine in TLA+

Terminology

- State: a state is an assignment of values to variables.
- Behavior (execution): a behavior is any infinite sequence of states.
- State function: a state function is an expression that is built from declared variables, declared constants, and constant operators.
- State predicate: a state predicate is a Boolean-valued state function.

$$\underbrace{[x = 1] \to [x = 30] \to \overset{\textbf{state } s}{\overset{\downarrow}{[x = 42]}} \to [x = -5] \to \dots}_{\textbf{behavior}}$$

NEBULAS

State Machine in TLA+

Terminology

- State: a state is an assignment of values to variables.
- Behavior (execution): a behavior is any infinite sequence of states.
- State function: a state function is an expression that is built from declared variables, declared constants, and constant operators.
- State predicate: a state predicate is a Boolean-valued state function.

state *s*

$$\underbrace{[x = 1] \rightarrow [x = 30] \rightarrow [x = 42] \rightarrow [x = -5] \rightarrow \dots}_{\textbf{behavior}}$$

state function *e* $x \in T$

NEBULAS

State Machine in TLA+

Terminology

- State: a state is an assignment of values to variables.
- Behavior (execution): a behavior is any infinite sequence of states.
- State function: a state function is an expression that is built from declared variables, declared constants, and constant operators.
- State predicate: a state predicate is a Boolean-valued state function.

state *s*

$$\underbrace{[x = 1] \rightarrow [x = 30] \rightarrow \overset{\downarrow}{[x = 42]} \rightarrow [x = -5] \rightarrow \dots}_{\textbf{behavior}}$$

declared variable   declared constant

state function *e* $x \in T$

constant operator

The value of function e in state s is the constant expression $42 \in T$

State Machine in TLA+

Terminology

- Step: a pair of successive states.
- Transition function: a transition function is an expression built from state functions using the **priming operator and the other action operators** of TLA+.
- Action: an action is a **Boolean-valued** transition function.

State Machine in TLA+

Terminology

- Step: a pair of successive states.
- Transition function: a transition function is an expression built from state functions using the **priming operator and the other action operators** of TLA+.
- Action: an action is a **Boolean-valued** transition function.

$$[x = 1] \rightarrow [x = 2] \rightarrow \underbrace{\boxed{\overset{\textbf{state } \boldsymbol{s}}{\overset{\downarrow}{[x = 3]}} \rightarrow \overset{\textbf{state } \boldsymbol{t}}{\overset{\downarrow}{[x = 4]}} \rightarrow}}_{\textbf{step}} \ldots$$

State Machine in TLA+

Terminology

- Step: a pair of successive states.
- Transition function: a transition function is an expression built from state functions using the **priming operator and the other action operators** of TLA+.
- Action: an action is a **Boolean-valued** transition function.

state s **state** t

$$[x = 1] \rightarrow [x = 2] \rightarrow \underbrace{\boxed{[x = 3] \rightarrow [x = 4] \rightarrow}}_{\textbf{step}} \ldots$$

step $s \rightarrow t$

transition function e $x' - x$

The value of transition function e on step $s \rightarrow t$ is 4-3

NEBULAS

State Machine in TLA+

Terminology

- Step: a pair of successive states.
- Transition function: a transition function is an expression built from state functions using the **priming operator and the other action operators** of TLA+.
- Action: an action is a **Boolean-valued** transition function.

state ***s*** **state** ***t***

$$[x = 1] \rightarrow [x = 2] \rightarrow \boxed{[x = 3] \rightarrow [x = 4] \rightarrow} \ldots$$

step

step $s \rightarrow t$

transition function *e* $x' - x$

The value of transition function e on step $s \rightarrow t$ is 4-3

action *A* $x' = x + 1$

The action A is true on step $s \rightarrow t$

The Syntax of TLA+(cont'd)

Nonconstant Operators

- The nonconstant operators are what distinguish TLA+ from ordinary mathematics.
- There are two classes of nonconstant operators: **action operators** and **temporal operators**.

The Syntax of TLA+(cont'd)

Nonconstant Operators

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]

NEBULAS

The Syntax of TLA+(cont'd)

Some more

Recursion

$$fact[n] \;=\; \text{IF } n = 0 \text{ THEN } 1 \text{ ELSE } n * fact[n-1], \text{ for all } n \in Nat$$

The Syntax of TLA+(cont'd)

Some more

Recursion

$$fact[n] \;=\; \text{IF } n = 0 \text{ THEN } 1 \text{ ELSE } n * fact[n-1], \text{ for all } n \in Nat$$

Functions vs. Operators

operator $\quad Tail(s) \triangleq [i \in 1 \,..\, (Len(s) - 1) \mapsto s[i+1]]$

function $\quad fact[n \in Nat] \triangleq \text{IF } n = 0 \text{ THEN } 1 \text{ ELSE } n * fact[n-1]$

NEBULAS

The Syntax of TLA+(cont'd)

Some more

Recursion

$$fact[n] \quad = \quad \text{IF} \;\; n = 0 \;\; \text{THEN} \;\; 1 \;\; \text{ELSE} \;\; n * fact[n-1], \;\; \text{for all } n \in Nat$$

Functions vs. Operators

operator $\qquad Tail(s) \;\triangleq\; [i \in 1 \,..\, (Len(s) - 1) \mapsto s[i+1]]$

function $\qquad fact[n \in Nat] \;\triangleq\; \text{IF} \;\; n = 0 \;\; \text{THEN} \;\; 1 \;\; \text{ELSE} \;\; n * fact[n-1]$

1. A function by itself is a complete expression that denotes a value, but an operator is not.
2. Unlike an operator, a function must have a domain, which is a set.
3. Unlike a function, an operator cannot be defined recursively.

NEBULAS

Writing a Specification

What is specification?

Writing a Specification

What is specification?

A specification is a **mathematical model** of a particular view of **some part of a system**.

Writing a Specification

What is specification?

A specification is a **mathematical model** of a particular view of **some part of a system**.

How to write a specification?

NEBULAS

Writing a Specification

What is specification?

A specification is a **mathematical model** of a particular view of **some part of a system**.

How to write a specification?

1. Pick the **variables** and define the type **invariant** and **initial predicate**;
2. Write the **next-state action**, which can be the disjunction of actions describing the different kinds of system operations;
3. Define those actions;
4. Combine the initial predicate, next-state action, and any fairness conditions chosen into the definition of a **single temporal formula** that is the specification;
5. Assert **theorems** about the specification.

NEBULAS

Writing a Specification

MODULE *HourClock*

EXTENDS *Naturals*

VARIABLE hr

$HCini \triangleq hr \in (1 \,..\, 12)$

$HCnxt \triangleq hr' = \text{IF } hr \neq 12 \text{ THEN } hr + 1 \text{ ELSE } 1$

$HC \triangleq HCini \land \Box[HCnxt]_{hr}$

THEOREM $HC \Rightarrow \Box HCini$

NEBULAS

Writing a Specification

MODULE $HourClock$

EXTENDS $Naturals$

Imports other modules

VARIABLE hr

$HCini \triangleq hr \in (1 \,..\, 12)$

$HCnxt \triangleq hr' = \textsc{if}\ hr \neq 12\ \textsc{then}\ hr + 1\ \textsc{else}\ 1$

$HC \triangleq HCini \land \Box[HCnxt]_{hr}$

THEOREM $HC \Rightarrow \Box HCini$

NEBULAS

Writing a Specification

MODULE *HourClock*

EXTENDS *Naturals* — Imports other modules

VARIABLE *hr* — Declare the variables

$HCini \triangleq hr \in (1 \,..\, 12)$ — Initial state predicate

$HCnxt \triangleq hr' = \text{IF } hr \neq 12 \text{ THEN } hr + 1 \text{ ELSE } 1$ — Next-state relation(action)

$HC \triangleq HCini \land \Box[HCnxt]_{hr}$

THEOREM $HC \Rightarrow \Box HCini$

Writing a Specification

$$
\begin{array}{ll}
\multicolumn{2}{c}{\textsc{module}\ \mathit{HourClock}} \\
\textsc{extends}\ \mathit{Naturals} & \text{Imports other modules} \\
\textsc{variable}\ hr & \text{Declare the variables} \\
\mathit{HCini} \triangleq hr \in (1 \,..\, 12) & \text{Initial state predicate} \\
\mathit{HCnxt} \triangleq hr' = \textsc{if}\ hr \neq 12\ \textsc{then}\ hr + 1\ \textsc{else}\ 1 & \text{Next-state relation(action)} \\
\mathit{HC} \triangleq \mathit{HCini} \wedge \Box[\mathit{HCnxt}]_{hr} & \text{Specification} \\
\hline
\textsc{theorem}\ \mathit{HC} \Rightarrow \Box \mathit{HCini} & \text{Theorem}
\end{array}
$$

NEBULAS

Writing a Specification

$$\text{MODULE } \mathit{HourClock}$$

$\text{EXTENDS } \mathit{Naturals}$	Imports other modules
$\text{VARIABLE } hr$	Declare the variables
$\mathit{HCini} \triangleq hr \in (1 \,..\, 12)$	Initial state predicate
$\mathit{HCnxt} \triangleq hr' = \text{IF } hr \neq 12 \text{ THEN } hr + 1 \text{ ELSE } 1$	Next-state relation(action)
$\mathit{HC} \triangleq \mathit{HCini} \land \Box[\mathit{HCnxt}]_{hr}$	Specification
$\text{THEOREM } \mathit{HC} \Rightarrow \Box \mathit{HCini}$	Theorem

Behavior 1 $[hr = 11] \rightarrow [hr = 12] \rightarrow [hr = 1] \rightarrow [hr = 2] \rightarrow \ldots$

Behavior 2 $[hr = 11] \rightarrow [hr = 77.2] \rightarrow [hr = 78.2] \rightarrow [hr = \sqrt{-2}] \rightarrow \ldots$

NEBULAS

Writing a Specification

$$\textsf{MODULE } \mathit{HourClock}$$

$\textsc{extends } \mathit{Naturals}$	Imports other modules
$\textsc{variable } hr$	Declare the variables
$HCini \triangleq hr \in (1 \,..\, 12)$	Initial state predicate
$HCnxt \triangleq hr' = \textsc{if } hr \neq 12 \textsc{ then } hr + 1 \textsc{ else } 1$	Next-state relation(action)
$HC \triangleq HCini \land \Box[HCnxt]_{hr}$	Specification
$\textsc{theorem } HC \Rightarrow \Box HCini$	Theorem

Behavior 1 $[hr = 11] \rightarrow [hr = 12] \rightarrow [hr = 1] \rightarrow [hr = 2] \rightarrow \ldots$

Behavior 2 $[hr = 11] \rightarrow [hr = 77.2] \rightarrow [hr = 78.2] \rightarrow [hr = \sqrt{-2}] \rightarrow \ldots$

Behavior 1 satisfies formula *HC* and behavior 2 does not.

Formula *HC* is regarded to be the specification of an hour clock because it is satisfied by exactly those behaviors that represent histories of the universe in which the clock functions properly.

NEBULAS

Temporal Formula: Type-Correctness

Terminology

- Temporal formula: a temporal formula is an assertion about behaviors.
- Theorem: a temporal formula satisfied by every behavior is called a theorem.
- Invariant: an invariant *Inv* of a specification *Spec* is a state predicate such that *Spec*=>[]*Inv* is a theorem.

Temporal Formula: Type-Correctness

Terminology

- Temporal formula: a temporal formula is an assertion about behaviors.
- Theorem: a temporal formula satisfied by every behavior is called a theorem.
- Invariant: an invariant *Inv* of a specification *Spec* is a state predicate such that *Spec*=>[]*Inv* is a theorem.

A invariant is a Boolean expression that's checked at the end of every "step" of the model.

NEBULAS

Temporal Formula: Type-Correctness

Terminology

- Temporal formula: a temporal formula is an assertion about behaviors.
- Theorem: a temporal formula satisfied by every behavior is called a theorem.
- Invariant: an invariant *Inv* of a specification *Spec* is a state predicate such that *Spec*=>[]*Inv* is a theorem.

A invariant is a Boolean expression that’s checked at the end of every “step” of the model.

$\Box F$ Formula F is always true.

Temporal Formula: Type-Correctness

Terminology

- Temporal formula: a temporal formula is an assertion about behaviors.
- Theorem: a temporal formula satisfied by every behavior is called a theorem.
- Invariant: an invariant *Inv* of a specification *Spec* is a state predicate such that *Spec*=>[]*Inv* is a theorem.

A invariant is a Boolean expression that's checked at the end of every "step" of the model.

$\Box F$ Formula *F* is always true.

$\text{THEOREM } HC \Rightarrow \Box HCini$

$\Box HCini$ should be true for any behavior satisfying HC. Thus the formula $HC \Rightarrow \Box HCini$ should be satisfied by every behavior, which is a theorem.

HCini is a state predicate as well as invariant.

NEBULAS

Temporal Formula: Liveness

So far, the specifications is about what a system ***must not*** do.

Temporal Formula: Liveness

So far, the specifications is about what a system ***must not*** do.

Safety property

NEBULAS

Temporal Formula: Liveness

So far, the specifications is about what a system ***must not*** do.

Safety property

How to specify that something ***does*** happen?

NEBULAS

Temporal Formula: Liveness

So far, the specifications is about what a system ***must not*** do.

Safety property

How to specify that something ***does*** happen?

Liveness property

Temporal Formula: Liveness

So far, the specifications is about what a system ***must not*** do.

Safety property

How to specify that something ***does*** happen?

Liveness property

Express liveness properties as **temporal formulas**.

NEBULAS

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

$\sigma \models F$ Temporal formula F assigns a Boolean value to a behavior σ.

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

$\sigma \models F$ Temporal formula F assigns a Boolean value to a behavior σ.

$\sigma^{+n} \triangleq \sigma_n \to \sigma_{n+1} \to \sigma_{n+2} \to \cdots$

NEBULAS

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

$\sigma \models F$ Temporal formula F assigns a Boolean value to a behavior σ.

$\sigma^{+n} \triangleq \sigma_n \rightarrow \sigma_{n+1} \rightarrow \sigma_{n+2} \rightarrow \cdots$

$\Box F$ Formula F is always true. $\sigma \models \Box F \triangleq \forall n \in Nat : \sigma^{+n} \models F$

NEBULAS

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

$\sigma \models F$ Temporal formula F assigns a Boolean value to a behavior σ.

$\sigma^{+n} \triangleq \sigma_n \to \sigma_{n+1} \to \sigma_{n+2} \to \cdots$

$\Box F$ Formula F is always true. $\sigma \models \Box F \triangleq \forall n \in Nat : \sigma^{+n} \models F$

$\Diamond F$ Formula F is eventually true. $\sigma \models \Diamond F \equiv \sigma \models \neg \Box \neg F$

NEBULAS

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

$\sigma \models F$ Temporal formula F assigns a Boolean value to a behavior σ.

$\sigma^{+n} \triangleq \sigma_n \rightarrow \sigma_{n+1} \rightarrow \sigma_{n+2} \rightarrow \cdots$

$\Box F$ Formula F is always true. $\sigma \models \Box F \triangleq \forall n \in Nat : \sigma^{+n} \models F$

$\Diamond F$ Formula F is eventually true. $\sigma \models \Diamond F \equiv \sigma \models \neg \Box \neg F \equiv \neg(\sigma \models \Box \neg F) \equiv \neg(\sigma \models \Box \neg F)$

NEBULAS

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

$\sigma \models F$ Temporal formula F assigns a Boolean value to a behavior σ.

$$\sigma^{+n} \triangleq \sigma_n \rightarrow \sigma_{n+1} \rightarrow \sigma_{n+2} \rightarrow \cdots$$

$\Box F$ Formula F is always true. $\sigma \models \Box F \triangleq \forall n \in Nat : \sigma^{+n} \models F$

$\Diamond F$ Formula F is eventually true. $\sigma \models \Diamond F \equiv \sigma \models \neg \Box \neg F \equiv \neg(\sigma \models \Box \neg F) \equiv \neg(\sigma \models \Box \neg F)$

$$\equiv \neg(\forall n \in Nat : \sigma^{+n} \models \neg F)$$

NEBULAS

Temporal Formula: Liveness

Recall

- State: a state is an assignment of values to variables.
- Behavior: a behavior is any infinite sequence of states.
- Temporal formula: a temporal formula is an assertion about behaviors.
- A temporal formula is true or false of a behavior.
- A temporal formula satisfied by every behavior is called a theorem

$\sigma \models F$ Temporal formula F assigns a Boolean value to a behavior σ.

$\sigma^{+n} \triangleq \sigma_n \rightarrow \sigma_{n+1} \rightarrow \sigma_{n+2} \rightarrow \cdots$

$\Box F$ Formula F is always true. $\sigma \models \Box F \triangleq \forall n \in Nat : \sigma^{+n} \models F$

$\Diamond F$ Formula F is eventually true. $\sigma \models \Diamond F \equiv \sigma \models \neg \Box \neg F \equiv \neg(\sigma \models \Box \neg F) \equiv \neg(\sigma \models \Box \neg F)$

$$\equiv \neg(\forall n \in Nat : \sigma^{+n} \models \neg F)$$

$$\equiv \exists n \in Nat : \sigma^{+n} \models F$$

NEBULAS

The Syntax of TLA+(cont'd)

Nonconstant Operators

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]
$F \stackrel{+}{\triangleright} G$	[F guarantees G]
$\boldsymbol{\exists}\, x : F$	[Temporal existential quantification (hiding)]
$\boldsymbol{\forall}\, x : F$	[Temporal universal quantification]

Temporal Formula: Summary

Action

$$[A]_e \triangleq A \vee (e' = e)$$

$$\langle A \rangle_e \triangleq A \wedge (e' \neq e)$$

Temporal Formula: Summary

Action

$[A]_e \triangleq A \vee (e' = e)$ Next or stutter

$\langle A \rangle_e \triangleq A \wedge (e' \neq e)$ Next and change

Temporal Formula: Summary

Action

$[A]_e \triangleq A \vee (e' = e)$ Next or stutter

$\langle A \rangle_e \triangleq A \wedge (e' \neq e)$ Next and change

Specification

$Init \wedge \Box [Next]_v \wedge \Box \Diamond \langle Next \rangle_v$

NEBULAS

Temporal Formula: Summary

Action

$[A]_e \triangleq A \lor (e' = e)$ Next or stutter

$\langle A \rangle_e \triangleq A \land (e' \neq e)$ Next and change

Specification

$$Init \land \underbrace{\Box [Next]_v}_{\text{Safety}} \land \underbrace{\Box \Diamond \langle Next \rangle_v}_{\text{Liveness}}$$

NEBULAS

TBC…

- Weak & Strong Fairness
- TLC Model Checker
- Example: Paxos & Byzantine Paxos with TLA+

The hard part of learning to write TLA+ specs
is learning to think **abstractly** about the system.

Thanks