MEC302: Embedded Computer Systems

Theme III: Analysis and Verification

Lecture 10 – Invariants and Temporal Logic

Dr. Timur Saifutdinov

Assistant Professor at EEE, SAT

Email:
Timur.Saifutdinov@xjtlu.edu.cn

Outline

- Specifications and temporal logic;
- Invariants;
- Linear Temporal Logic (LTL);
- LTL formulas and operators;
- LTL formulas in practice.

Specifications

Every embedded system must be designed to meet certain requirements. Such system requirements are called **system properties** or **specifications**.

"*A design without specifications cannot be right or wrong, it can only be surprising!*"

Young *et al.*

A **mathematical specification** of system properties is also known as a **formal specification**:

- **Temporal logic** is a precise mathematical notation with associated rules for representing and reasoning about timing-related properties of systems, e.g.:
  - An **invariant** is one of the most common kinds of system property;
  - More elaborate system properties use **Linear Temporal Logic (LTL)** formulations.

Invariants

An **invariant** is a property that holds for a system if it is true in the initial state of the system, and it remains true as the system evolves, after every reaction, in every state.

Invariant properties are commonly used to specify the *key requirements* for implementations of embedded computer systems, e.g. using **natural language**:

- The program never dereferences a null pointer.
- If a thread *A* blocks while trying to acquire a mutex lock, then the thread *B* that holds that lock must not be blocked attempting to acquire a lock held by *A*.
- The traffic lights at an intersection cannot both be green at the same time.

To give precise specifications, **invariants** can be formulated with **temporal logic** using **propositional logic formulas**.

Propositional Logic Formulas

A **propositional logic formula** or **proposition** is a predicate that combines atomic propositions using logical connectives:

- *conjunction* (logical AND, denoted ∧)
- *disjunction* (logical OR, denoted ∨)
- *negation* (logical NOT, denoted ¬)
- *implies* (logical implication, denoted ⟹)

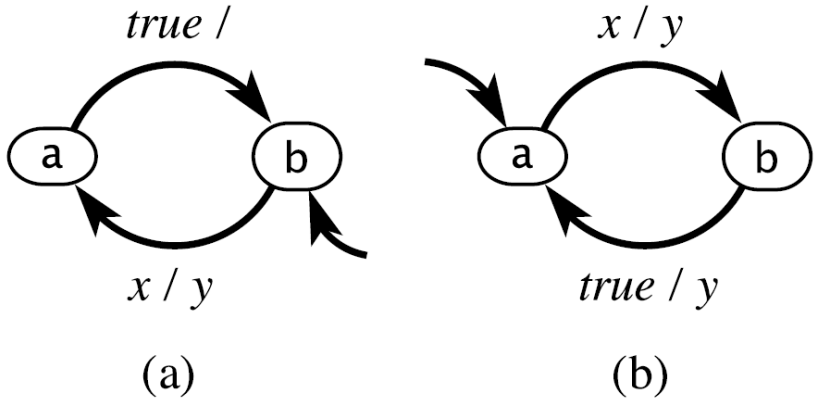
#Logical implication:

If p_1 and p_2 are propositions, then the following propositions are equivalent:

- $p_1 \Longrightarrow p_2$
- $\neg p_2 \Longrightarrow \neg p_1$
- $\neg p_1 \vee p_2$

#Consider two state machines and examples of propositions:

input: x: pure
output: y: pure



(a) (b)

$x \wedge y$	True if x and y are both present.
$x \vee y$	True if either x or y is present.
$\neg y$	True if y is absent.
$\mathrm{a} \Longrightarrow y$	True if whenever the FSM is in state a, the output y will be made present by the reaction
$true$	Always true

Linear Temporal Logic (LTL)

LTL expresses a property over a *single arbitrary execution* of a system, e.g., it can express the following kinds of properties:

- Occurrence of an event and its properties;
- Causal dependency between events;
- Ordering of events.

LTL is a formalism for describing linear-time properties of a system.

Let us consider **Finite-State Machines (FSMs)** as an object of reasoning, where the following notations are used:

$q_0, q_1, q_2, q_3, \ldots$ is the execution trace of FSM, where

$q_j = (x_j, s_j, y_j)$, s_j is the state, x_j is the input valuation, y_j is the output valuation at reaction j.

LTL formulas

In contrast to propositions, **LTL formulas** apply to an entire trace of a system:

$$q_0, q_1, q_2, q_3, \ldots$$

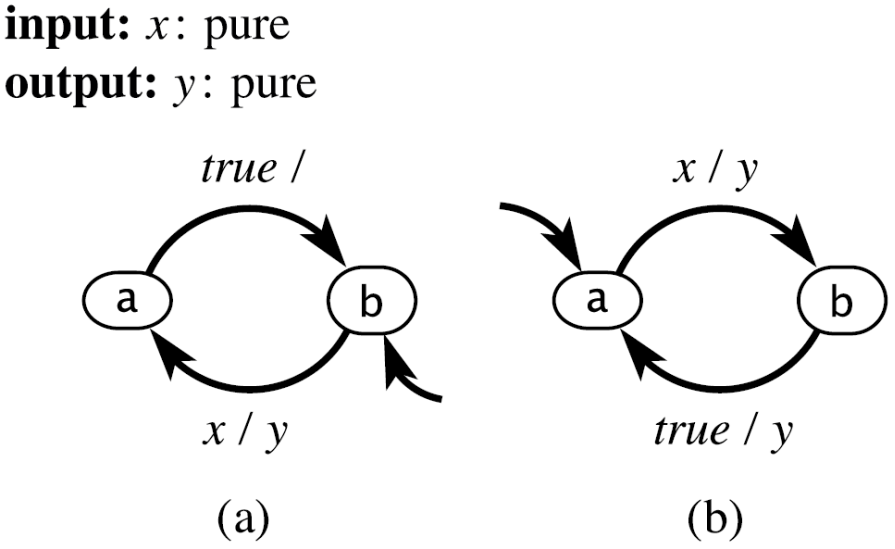
If p is a proposition, then by definition, we say that **LTL formula** $\phi = p$ holds for the trace $q_0, q_1, q_2, q_3, \ldots$ if and only if p is true for q_0.

Reasoning about **LTL formulas**:

- To demonstrate that an LTL formula **does not hold**, it is sufficient to give one trace for which it is false (i.e., **counterexample**);
- To show that an LTL formula **holds**, it is necessary to demonstrate/prove that it is true for all traces.

#Consider two **FSMs** and several **LTL formulas**:

- $\phi_1 = \mathrm{a}$
- $\phi_2 = x \Longrightarrow y$
- $\phi_3 = y$



LTL operators

LTL provides a number of operators to support more elaborate system properties:

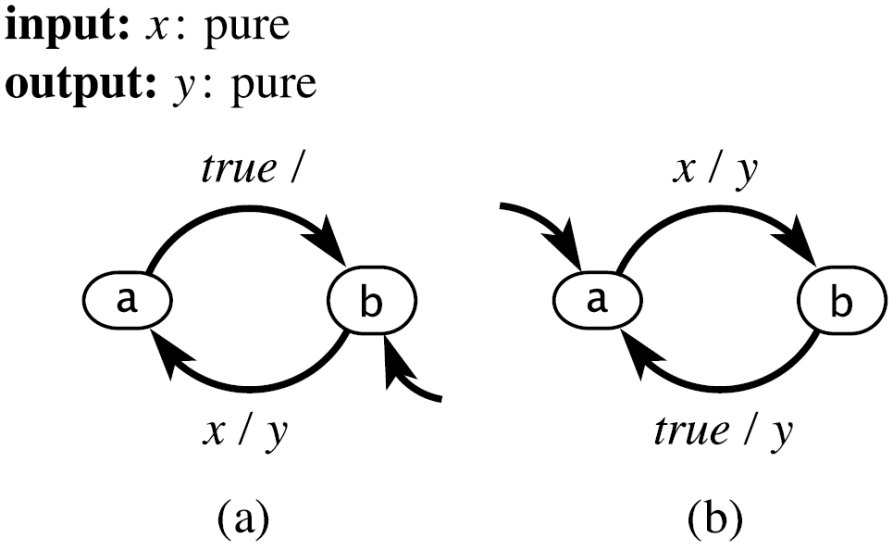
- **G Operator** applies an LTL formula to the whole trace:

  $\mathbf{G}\phi$ ("**globally** ϕ") holds for the trace if and only if, for <u>*all*</u> $j \geq 0$, formula ϕ holds in the trace $q_j, q_{j+1}, q_{j+2}, \ldots$

If ϕ is a propositional logic formula, then $\mathbf{G}\phi$ means that ϕ holds in every reaction:

#Consider two **FSMs** and several **LTL formulas**:

- $\phi_1 = \mathbf{G}(x \Longrightarrow y)$
- $\phi_2 = \mathbf{G}(x \wedge y)$



LTL operators

- **F Operator** applies an LTL formula to a suffix* of the trace:

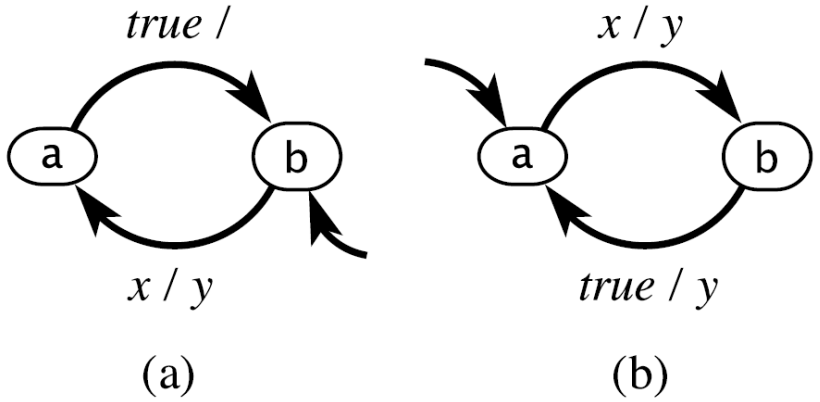
  $\mathbf{F}\phi$ ("**finally** ϕ") holds for the trace if and only if, for *some* $j \geq 0$, formula ϕ holds in the trace suffix* $q_j, q_{j+1}, q_{j+2}, \ldots$

If ϕ is a propositional logic formula, then $\mathbf{F}\phi$ means that ϕ will eventually hold (at least once):

#Consider two **FSMs** and several **LTL formulas**:

- $\phi_1 = \mathbf{F}\mathrm{b}$
- $\phi_2 = \mathbf{G}(x \Longrightarrow \mathbf{F}\mathrm{b})$
- $\phi_3 = (\mathbf{G}x) \Longrightarrow (\mathbf{F}\mathrm{b})$

input: x: pure
output: y: pure



* – A suffix is a tail of the trace beginning with some reaction and including all subsequent reactions.

LTL operators

- **X Operator** applies an LTL formula to the next state in the trace:

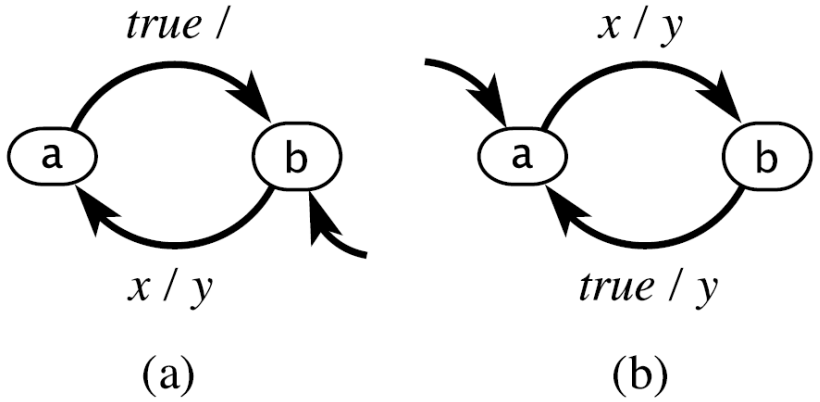
  $\mathbf{X}\phi$ ("**next state** ϕ") holds for a trace $q_0, q_1, q_2, \ldots$ if and only if ϕ holds for the trace $q_1, q_2, q_3, \ldots$.

If ϕ is a propositional logic formula, then $\mathbf{X}\phi$ means that ϕ will hold for the next state:

#Consider two **FSMs** and several **LTL formulas**:

- $\phi_1 = x \Longrightarrow \mathbf{X}\mathrm{a}$
- $\phi_2 = \mathbf{G}(x \Longrightarrow \mathbf{X}\mathrm{a})$
- $\phi_3 = \mathbf{G}(\mathrm{b} \Longrightarrow \mathbf{X}\mathrm{a})$

input: x: pure
output: y: pure

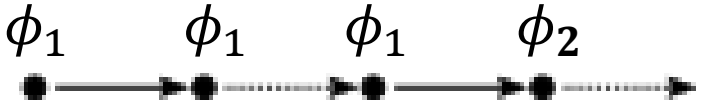


LTL operators

- **U Operator** applies two LTL formulas to the trace:

  $\phi_1 \mathbf{U} \phi_2$ ("ϕ_1 **until** ϕ_2") holds for the trace if and only if there exists $j \geq 0$ such that ϕ_2 holds in the suffix $q_j, q_{j+1}, q_{j+2}, \ldots$ and ϕ_1 holds in suffixes $q_i, q_{i+1}, q_{i+2}, \ldots$, for all $i < j$.

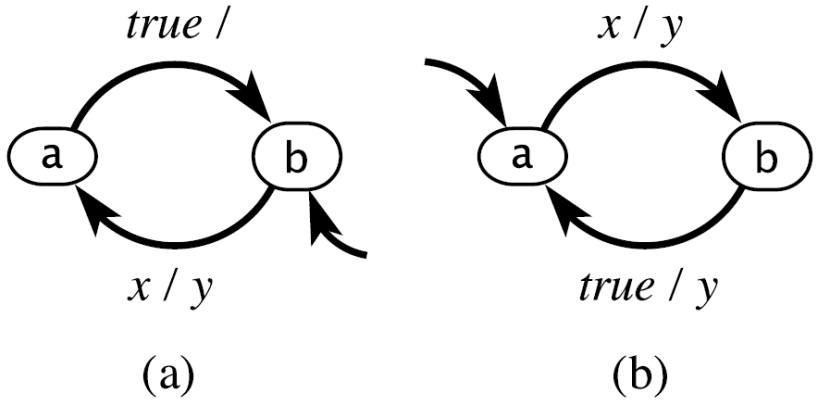
ϕ_1 may or may not hold for $q_j, q_{j+1}, q_{j+2}, \ldots$:



#Consider two **FSMs** and several **LTL formulas**:

- $\phi_1 = \mathrm{a}\mathbf{U}x$
- $\phi_2 = (\mathbf{G}\neg x) \vee (\mathrm{a}\mathbf{U}x)$

input: x: pure
output: y: pure



LTL formulas in practice

#Consider the following examples of specification requirements:

- *"Whenever the robot is facing an obstacle, it is required to move at least 5 cm away from the obstacle"*:

$$\mathbf{G}(p \Longrightarrow \mathbf{F}q),$$

where p is the condition that the robot is facing an obstacle,

q is the condition where the robot is at least 5 cm away from the obstacle.

- *"Whenever the reset signal is asserted the state machine shall move immediately to the* ErrorReset *state and remain there until the reset signal is de-asserted"*:

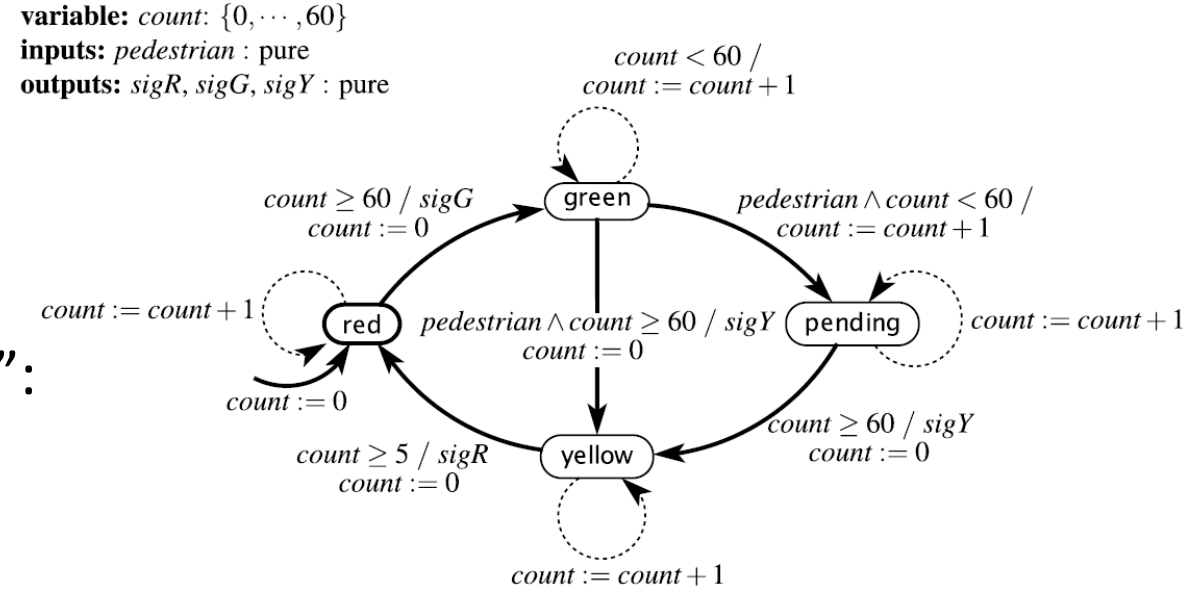
$$\mathbf{G}\big(reset \Longrightarrow \mathbf{X}(\text{ErrorReset}\mathbf{U}\neg reset)\big).$$

LTL formulas in practice

#Consider the ESM model of the traffic light controller:

- *"The traffic light controller should always provide signals according to the sequence:*

  Red → Green→ Yellow →Red →Green →…":



The **LTL** formula for the property:

$$\mathbf{G}\Big\{\Big(sigR \implies \mathbf{X}\big((\neg sigY \wedge \neg sigR)\ \mathbf{U}\ sigG\big)\Big)$$

$$\wedge\ \Big(sigG \implies \mathbf{X}\big((\neg sigR \wedge \neg sigG)\ \mathbf{U}\ sigY\big)\Big)$$

$$\wedge\Big(sigY \implies \mathbf{X}\big((\neg sigG \wedge \neg sigY)\ \mathbf{U}\ sigR\big)\Big)\Big\}$$

LTL formulas in practice

Other useful (common) **LTL formulas**:

- Infinitely many occurrences: $\mathbf{G}\,\mathbf{F}p$

  "p is *true* eventually" or "p is *true* infinitely often";

- Steady-state property: $\mathbf{F}\,\mathbf{G}p$

  "eventually p holds globally" or "from some point in the future, p holds at all times";

- Request-response property: $\mathbf{G}(p \implies \mathbf{F}q)$

  "a request p will eventually produce a response q".

To sum up

- **System properties** or **specifications** are used to specify requirements for a system to avoid “surprises”;
- **Temporal logic** is a formalism with associated rules for representing and reasoning about timing-related properties of systems:
  - An **invariant** is a property that holds for a system if it remains true at all times during operation of the system;
  - **Linear Temporal Logic (LTL)** formulations allow specifying more elaborate timing-related properties;
- To prove that an **LTL formula** is:
  - *False* – it is sufficient to give one trace for which it is false (i.e., **counterexample**);
  - *True* – it is necessary to demonstrate that it is true for all traces.
- In practice, **LTL** formulations are effective in translating system specifications from natural language (e.g., English) to precise mathematical notations.

The end!

See you next time for the concluding lecture on May 15;

Student Module Feedback Questionnaire (MQ) is open on LM:

- Please read the announcement (launch) email carefully before start your questionnaires, and you will have an opportunity to **win a big prize** (see below picture) if you complete all of your questionnaires before the deadline.

First Prize: iPad Air
1 Available

Second Prize: Apple Watch
2 Available

Third Prize: AirPods
3 Available

Participation Prize: XJTLU Bear/Bird
100 Available