Chapter 5

Linear Temporal Logic (LTL)

Presented by

Rehab Ashari Sahar Habib

CONTENT

Temporal Logic & Linear Temporal Logic

- Temporal logics (TL) is a convenient formalism for specifying and verifying properties of reactive systems. We can say that the modalities in Temporal Logic are Time abstract
- linear temporal logic (LTL) that is an infinite sequence of states where each point in time has a unique successor, based on a linear-time perspective.
- Linear temporal property is a temporal logic formula that describes a set of infinite sequences for which it is true
- Purpose→ Translate the properties which are written using the natural languages into LTL by using special syntax. By given the TS and LTL formula φ, we can check if φ hold in TS or not.
- Model checking tools → SPIN
- An important way to model check is to express desired properties (such as the ones described above) using LTL operators and actually check if the model satisfies this property. One technique is to obtain a Büchi automaton that is "equivalent" to the model and one that is "equivalent" to the negation of the property. The intersection of the two non-deterministic Büchi automata is empty if the model satisfies the property.

Syntax

LTL formula is built up from :

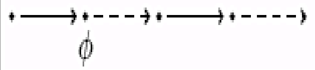
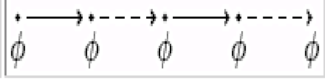
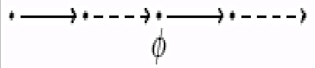
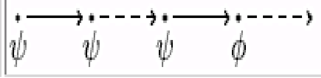
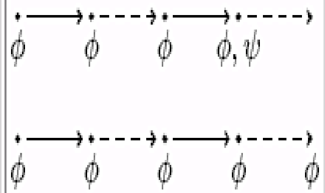
- A finite set of Atomic propositions → (State label "a" ϵ AP in the transition system)
- Basic Logical Operators → ¬ (negation) , ∧ (conjunction)
- Basic Temporal Operators → **O** (next) , **U** (until) , **true**
- There are additional logical operators are ∨ (disjunction), →(implication), ↔(equivalence)
- There are additional temporal operators are :

$\Diamond$	"eventually" (eventually in the future)
$\Box$	"always" (now and forever in the future)

- By combining the temporal modalities ◊ and □, new temporal modalities are obtained.

$\Box\Diamond\varphi$	"infinitely often φ"
$\Diamond\Box\varphi$	"eventually forever φ"

Syntax

Textual	Symbolic†	Explanation	Diagram
Unary operators:			
X φ	$\bigcirc\phi$	neXt: φ has to hold at the next state.	
G φ	$\Box\phi$	Globally: φ has to hold on the entire subsequent path.	
F φ	$\Diamond\phi$	Finally: φ eventually has to hold (somewhere on the subsequent path).	
Binary operators:			
ψ U φ	$\psi\mathcal{U}\phi$	Until: ψ has to hold *at least* until φ, which holds at the current or a future position.	
ψ R φ	$\psi\mathcal{R}\phi$	Release: φ has to be true until and including the point where ψ first becomes true; if ψ never becomes true, φ must remain true forever.	

SYNTAX

- ◊ → "F" Finally which means something in the future.
- □ → "G" Globally which means globally in the future.
- ○ → "X" NeXt time.
- LTL can be extended with past operators
  - $\square^{-1}$ → Always in the past.
  - $\Diamond^{-1}$ → sometimes in the past.
  - $\circ^{-1}$ → Previous state.

  $\square\,(\text{red} \rightarrow \circ^{-1}\,\text{yellow})$
- Weak until (a **W** b),

  requires that a remains true until b becomes true, but does not require that b ever does becomes true (i.e. a remains true forever). It follows the expansion law of until.
- Release (a **R** b),

  informally means that b is true until a becomes true, or b is true forever.

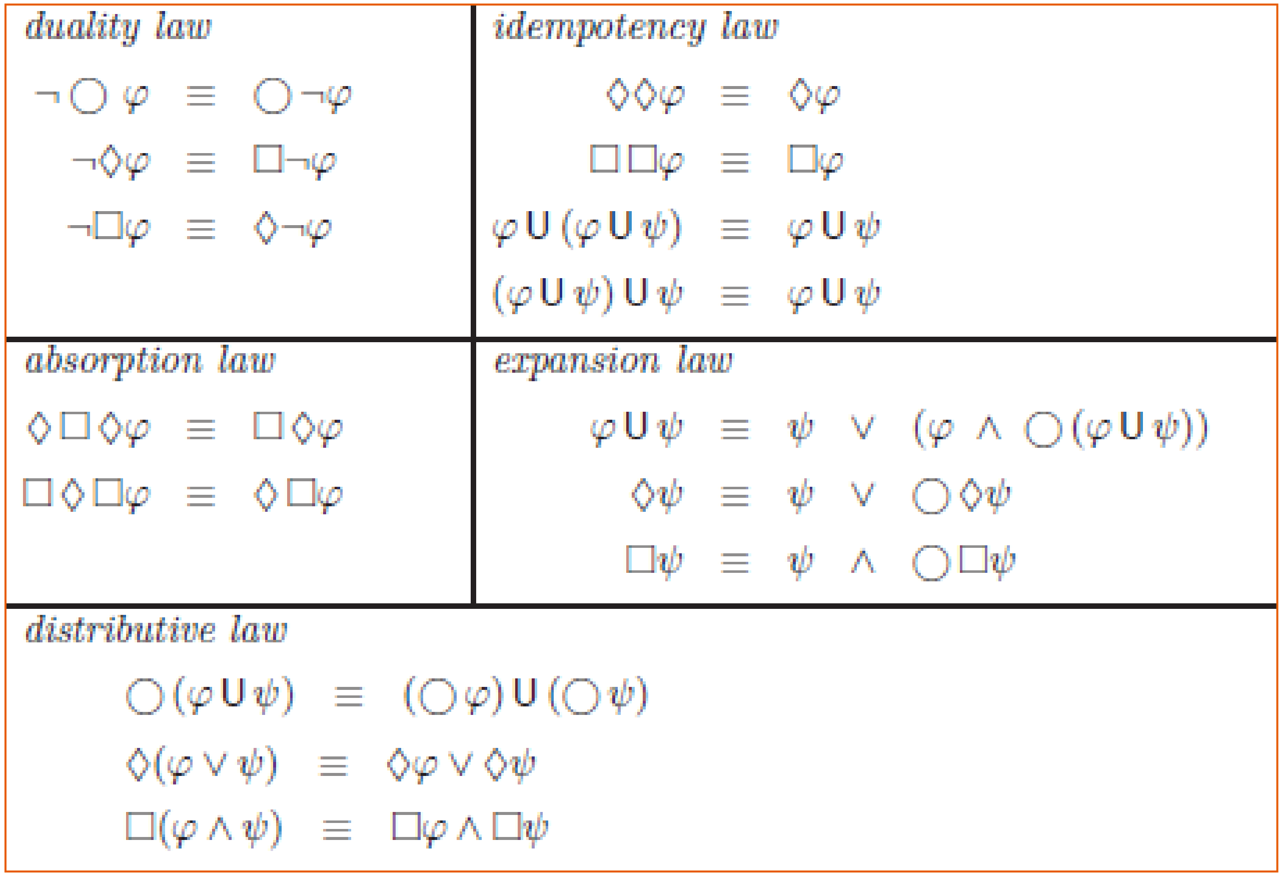
Semantics

- LTL formulae φ stands for properties of paths (Traces) and The path can be either fulfill the LTL formula or not.
- First, The semantics of φ is defined as a language Words(φ). Where Words(φ) contains all infinite words over the alphabet 2^{AP} that satisfy φ
- Then, the semantics of φ is extended to an interpretation over paths and states of a TS.
- Thus, a transition system TS satisfies the LT property P *if all its traces respect P, i.e., if all* its behaviors are admissible. A state satisfies *P whenever all traces starting in this state* fulfill *P*.
- The transition system TS satisfies ϕ if TS satisfies the LT property Words(ϕ).

  i.e., if all initial paths of TS paths starting in an initial state $s_0 \in I$ satisfy ϕ.
- Thus, it is possible that a TS (or s_i) satisfies neither ϕ *nor* $\neg\phi$
- Any LTL formula can be transformed into a canonical form, the so-called positive normal form (PNF). In order to transform any LTL formula into PNF, for each operator, a dual operator needs to be incorporated into the syntax of PNF

  formulae.

Equivalence of LTL Formulae



duality law

$$\begin{aligned} \neg \bigcirc \varphi &\equiv \bigcirc \neg \varphi \\ \neg \Diamond \varphi &\equiv \Box \neg \varphi \\ \neg \Box \varphi &\equiv \Diamond \neg \varphi \end{aligned}$$

idempotency law

$$\begin{aligned} \Diamond \Diamond \varphi &\equiv \Diamond \varphi \\ \Box \Box \varphi &\equiv \Box \varphi \\ \varphi \,\mathsf{U}\, (\varphi \,\mathsf{U}\, \psi) &\equiv \varphi \,\mathsf{U}\, \psi \\ (\varphi \,\mathsf{U}\, \psi) \,\mathsf{U}\, \psi &\equiv \varphi \,\mathsf{U}\, \psi \end{aligned}$$

absorption law

$$\begin{aligned} \Diamond \Box \Diamond \varphi &\equiv \Box \Diamond \varphi \\ \Box \Diamond \Box \varphi &\equiv \Diamond \Box \varphi \end{aligned}$$

expansion law

$$\begin{aligned} \varphi \,\mathsf{U}\, \psi &\equiv \psi \vee (\varphi \wedge \bigcirc (\varphi \,\mathsf{U}\, \psi)) \\ \Diamond \psi &\equiv \psi \vee \bigcirc \Diamond \psi \\ \Box \psi &\equiv \psi \wedge \bigcirc \Box \psi \end{aligned}$$

distributive law

$$\begin{aligned} \bigcirc (\varphi \,\mathsf{U}\, \psi) &\equiv (\bigcirc \varphi) \,\mathsf{U}\, (\bigcirc \psi) \\ \Diamond (\varphi \vee \psi) &\equiv \Diamond \varphi \vee \Diamond \psi \\ \Box (\varphi \wedge \psi) &\equiv \Box \varphi \wedge \Box \psi \end{aligned}$$

FAIRNESS IN LTL

- LTL Fairness Constrains and Assumptions

Let Φ and Ψ be propositional logic formulae over AP.

1. An *unconditional LTL fairness constraint* is an LTL formula of the form

$$ufair = \Box\Diamond\Psi.$$

2. A *strong LTL fairness condition* is an LTL formula of the form

$$sfair = \Box\Diamond\Phi \longrightarrow \Box\Diamond\Psi.$$

3. A *weak LTL fairness constraint* is an LTL formula of the form

$$wfair = \Diamond\Box\Phi \longrightarrow \Box\Diamond\Psi.$$

Φ stands for "something is enabled"; Ψ for "something is taken"

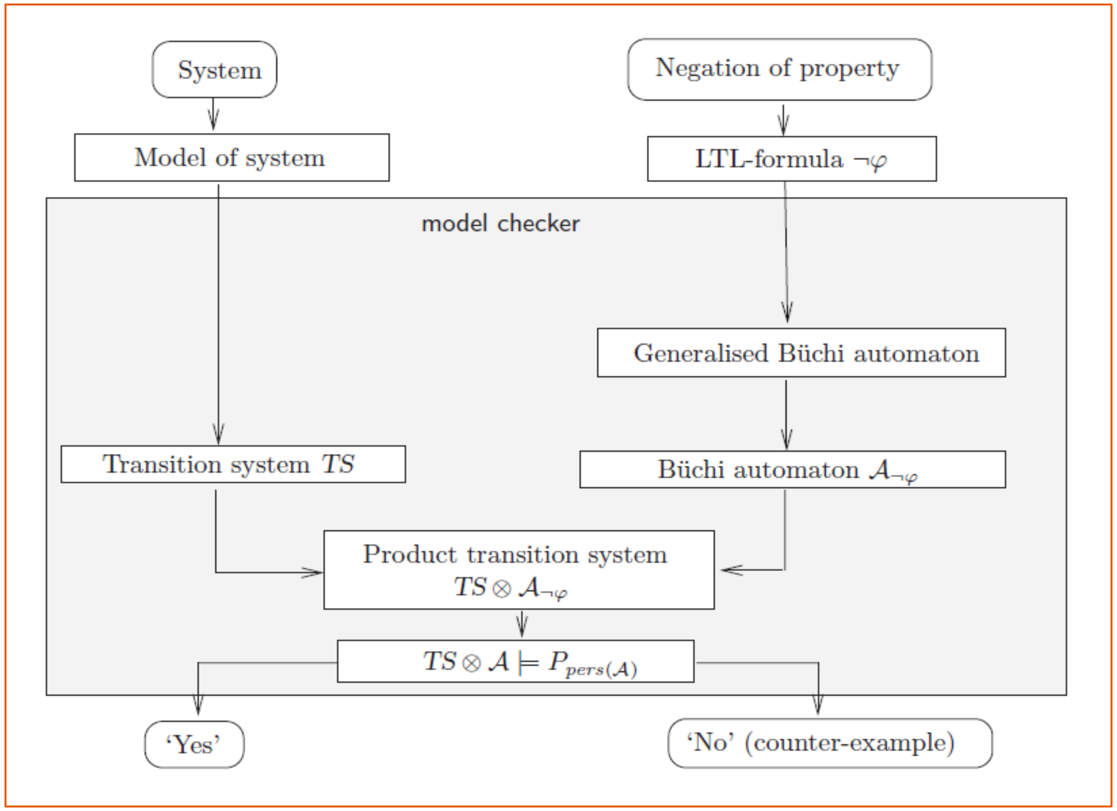
- That is to say , rather than determining for transition system TS and LTL formula ϕ whether TS$|=\phi$, we focus on the fair executions of TS.
- An LTL fairness assumption is a conjunction of LTL fairness constraints.

$$\begin{aligned} FairPaths(s) &= \{\, \pi \in Paths(s) \mid \pi \models fair \,\}, \\ FairTraces(s) &= \{\, trace(\pi) \mid \pi \in FairPaths(s) \,\}. \end{aligned}$$

Automata-Based LTL Model Checking

To check whether ϕ holds for TS

Constructs an NBA for the negation of the input formula ϕ (representing the ”bad behaviors”)


System
Negation of property
Model of system
LTL-formula $\neg\varphi$
model checker
Generalised Büchi automaton
Transition system TS
Büchi automaton $\mathcal{A}_{\neg\varphi}$
Product transition system $TS \otimes \mathcal{A}_{\neg\varphi}$
$TS \otimes \mathcal{A} \models P_{pers(\mathcal{A})}$
‘Yes’
‘No’ (counter-example)

Generalized Büchi Automata

- **Generalized Büchi automaton (GBA)** is a variant of **Büchi automaton**
- The difference with the Büchi automaton is its accepting condition, i.e., a set of sets of states.
- A run is accepted by the automaton if it visits at least one state of every set of the accepting condition infinitely often.
- Generalized Büchi automata (GBA) is equivalent in expressive power with Büchi automata
- A generalized Buchi automaton (GBA) over Σ is

$$A = (S, \Sigma, T, I, F)$$

- S is a finite set of states
- $\Sigma = \{a, b, \ldots\}$ is a finite alphabet set of A
- $T \subseteq S \times \Sigma \times S$ is a transition relation
- $I \subseteq S$ is a set of initial states
- $F = \{F1, \ldots, F_k\} \subseteq 2^S$ is a set of sets of final states.
- *A* accepts exactly those runs in which the set of infinitely often occurring states contains at least a state from each $\mathbf{F_1,\ldots,F_n}$.
- A run π of a GBA is said to be accepting iff,

$$\text{for all } 1 \leq i \leq k, \text{ we have } \inf(\pi) \cap F_i \neq \emptyset$$

NBA & Generalized NBA (GNBA)

Nondeterministic Büchi Automaton

$A = (Q, \Sigma, \delta, I, F)$, where $F \subseteq Q$ is the set of accepting states.

- A run ρ of A on omega word α is an infinite sequence $\rho = q_o, q_1, q_2, \ldots$ s.t. $q_0 \in I$ and $q_i \xrightarrow{a_i} q_{i+1}$ for $0 \leq i$.
- The run ρ is accepting if

  $$Inf(\rho) \cap F \neq \emptyset.$$
- The language accepted by A

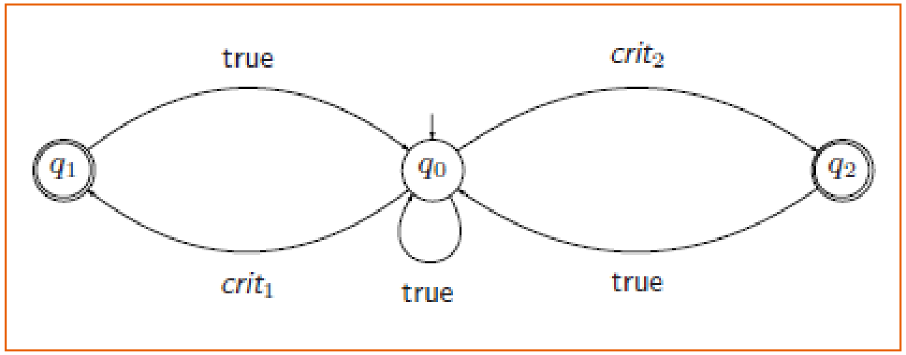
  $\mathcal{L}(A) = \{\alpha \in \Sigma^\omega \mid A \text{ has an accepting run on } \alpha\}$

A Generalized Büchi Automaton is $A := (Q, \Sigma, \delta, I, FT)$ where $FT = < F_1, F_2, \ldots, F_k >$ with $F_i \subseteq Q$.

A run ρ of A is accepting if $Inf(\rho) \cap F_i \neq \emptyset$ for each $1 \leq i \leq k$.

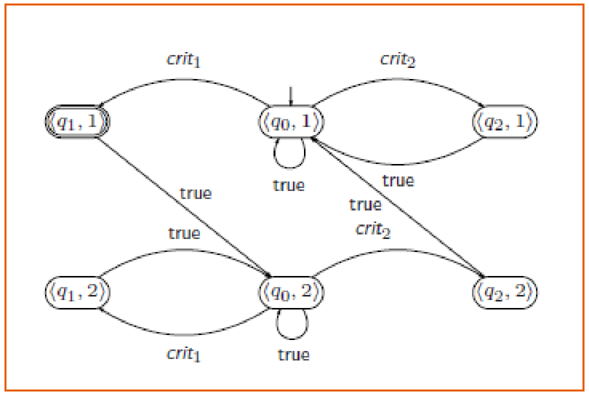
NBA & Generalized NBA (GNBA)

A GNBA for the property "both processes are infinitely often in their critical section"
F = { {q1 }, { q2 }}



Sketch of transformation GNBA (with $|\mathcal{F}| = k$) into equivalent NBA:

- make k copies of the GNBA
- initial states of NBA := the initial states in the first copy
- final states of NBA := accept set F_1 in the first copy
- on visiting in i-th copy a state in F_i, then move to the $(i+1)$-st copy



NBA & Generalized & Closure ϕ

- GNBA are like NBA, but have a distinct *acceptance criterion*
- a GNBA requires to visit several sets *F1, . . . , Fk* (k ≥ 0) *infinitely often*
  - for *k=0, all runs are accepting*
  - for *k=1 this boils down to an NBA*
- *GNBA are useful to relate temporal logic and automata,* but they are equally expressive as NBA
- Closure ϕ → Consisting of all subformulae ψ of ϕ and their negation $\neg\psi$

The Satisfiability Problem:

- for a given LTL formula Ø, **there exists** a model for which Ø holds. That is, we have Words(Ø) = Ø.

The Validity problem:

- Formula Ø is valid whenever Ø holds **under all** interpretations, i.e., $\phi \equiv$ *true*.

LTL Satisfiability and Validity checking

PSPACE Complexity:

- In computer science, the **space complexity** of an algorithm quantifies the amount of memory space that an algorithm needs to run as a function of the size of the input to solve the problem.
- The space complexity of an algorithm is commonly expressed using big O notation.
- In complexity theory, **PSPACE** is the set of all decision problems which can be solved by an algorithm using a polynomial amount of memory space.
- In complexity theory, a decision problem is **PSPACE-complete** if it is in the complexity class PSPACE, and every problem in PSPACE can be reduced to it in polynomial space
- A problem can be PSPACE-hard but not PSPACE-complete because it may not be in PSPACE.
- *More efficient technique cannot be* achieved as both the validity and satisfiability problems are PSPACE-hard. In fact, both problems are even PSPACE-complete.