Chapter 5

Linear Temporal Logic (LTL)

Presented by

Rehab Ashari          Sahar Habib
CONTENT

- Temporal Logic & Linear Temporal Logic (LTL)
- Syntax
- Semantics
- Equivalence of LTL Formulae
- Fairness in LTL
- Automata-Based LTL Model Checking
- NBA & Generalized NBA (GNBA)
- GNBA and Closure of $\phi$
- LTL Satisfiability and Validity checking
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 \( \varphi \), we can check if \( \varphi \) 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.
LTL formula is built up from:

- A finite set of Atomic propositions \( \rightarrow \) (State label “a” \( \in \) AP in the transition system)
- Basic Logical Operators \( \rightarrow \) \(-\) (negation), \( \wedge\) (conjunction)
- Basic Temporal Operators \( \rightarrow \) \(O\) (next), \(U\) (until), \textbf{true}
- There are additional logical operators are \( \vee\) (disjunction), \(\rightarrow\) (implication), \(\leftrightarrow\) (equivalence)
- There are additional temporal operators are:

  - \(\Diamond\) “eventually” (eventually in the future)
  - \(\square\) “always” (now and forever in the future)

- By combining the temporal modalities \(\Diamond\) and \(\square\), new temporal modalities are obtained:

  - \(\square\Diamond \varphi\) “infinitely often \(\varphi\)”
  - \(\Diamond\square \varphi\) “eventually forever \(\varphi\)”
**Syntax**

<table>
<thead>
<tr>
<th>Textual</th>
<th>Symbolic</th>
<th>Explanation</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unary operators:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| \( X \phi \) | \( \Diamond \phi \) | \text{Next: } \phi \text{ has to hold at the next state.} | \[
\xrightarrow{*} \xrightarrow{\cdots} \xrightarrow{\cdots} \phi
\] |
| \( G \phi \) | \( \square \phi \) | \text{Globally: } \phi \text{ has to hold on the entire subsequent path.} | \[
\phi \quad \phi \quad \phi \quad \phi \quad \phi \quad \phi
\] |
| \( F \phi \) | \( \Diamond \phi \) | \text{Finally: } \phi \text{ eventually has to hold (somewhere on the subsequent path).} | \[
\xrightarrow{*} \xrightarrow{\cdots} \xrightarrow{\cdots} \phi
\] |
| **Binary operators:** | | | |
| \( \psi U \phi \) | \( \psi \ulcorner \phi \urcorner \) | \text{Until: } \psi \text{ has to hold at least until } \phi, \text{ which holds at the current or a future position.} | \[
\psi \quad \psi \quad \psi \quad \psi \quad \phi
\] |
| \( \psi R \phi \) | \( \psi \Re \phi \) | \text{Release: } \phi \text{ has to be true until and including the point where } \psi \text{ first becomes true; if } \psi \text{ never becomes true, } \phi \text{ must remain true forever.} | \[
\phi \quad \phi \quad \phi \quad \phi \quad \psi
\] |
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
  - □⁻¹ → Always in the past.
  - ♦⁻¹ → sometimes in the past.
  - ○⁻¹ → Previous state.

□ ( red → ○⁻¹ yellow)

- Weak until (a W b),
  requires that a remains true until b becomes true, but does not require that b ever 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** $\varphi$ stands for properties of paths (Traces) and The path can be either fulfill the **LTL formula** or not.

- First, The semantics of $\varphi$ is defined as a language $\text{Words}(\varphi)$. Where $\text{Words}(\varphi)$ contains all infinite words over the alphabet $2^\text{AP}$ that satisfy $\varphi$.

- Then, the semantics of $\varphi$ is extended to an interpretation over paths and states of a TS.

- Thus, a transition system $\text{TS}$ 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 $\text{TS}$ satisfies $\varphi$ if $\text{TS}$ satisfies the **LT property** $\text{Words}(\varphi)$. i.e., if all initial paths of $\text{TS}$ paths starting in an initial state $s_0 \in I$ satisfy $\varphi$.

- Thus, it is possible that a $\text{TS}$ (or $s_0$) satisfies neither $\varphi$ nor $\overline{\varphi}$.

- 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

<table>
<thead>
<tr>
<th>Duality Law</th>
<th>Idempotency Law</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neg \bigcirc \varphi \equiv \bigdiamond \neg \varphi )</td>
<td>( \bigdiamond \bigdiamond \varphi \equiv \bigdiamond \varphi )</td>
</tr>
<tr>
<td>( \neg \bigdiamond \varphi \equiv \Box \neg \varphi )</td>
<td>( \Box \Box \varphi \equiv \Box \varphi )</td>
</tr>
<tr>
<td>( \neg \Box \varphi \equiv \bigdiamond \neg \varphi )</td>
<td>( \varphi \bigcirc (\varphi \bigcirc \psi) \equiv \varphi \bigcirc \psi )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Absorption Law</th>
<th>Expansion Law</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bigdiamond \Box \bigdiamond \varphi \equiv \Box \bigdiamond \varphi )</td>
<td>( \varphi \bigcirc \psi \equiv \psi \lor (\varphi \land \bigcirc (\varphi \bigcirc \psi)) )</td>
</tr>
<tr>
<td>( \Box \bigdiamond \Box \varphi \equiv \Box \bigdiamond \varphi )</td>
<td>( \bigdiamond \psi \equiv \psi \lor \bigcirc \bigdiamond \psi )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distributive Law</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bigcirc (\varphi \bigcirc \psi) \equiv (\bigcirc \varphi) \bigcirc (\bigcirc \psi) )</td>
</tr>
<tr>
<td>( \bigdiamond (\varphi \lor \psi) \equiv \bigdiamond \varphi \lor \bigdiamond \psi )</td>
</tr>
<tr>
<td>( \Box (\varphi \land \psi) \equiv \Box \varphi \land \Box \psi )</td>
</tr>
</tbody>
</table>
FAIRNESS IN LTL

- LTL Fairness Constrains and Assumptions

Let $\Phi$ and $\Psi$ 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
   \[ sfair = \Box \Diamond \Phi \rightarrow \Box \Diamond \Psi. \]

3. A *weak LTL fairness constraint* is an LTL formula of the form
   \[ ufair = \Diamond \Box \Phi \rightarrow \Box \Diamond \Psi. \]

- That is to say, rather than determining for transition system $TS$ and LTL formula $\phi$ whether $TS \models \phi$, we focus on the fair executions of $TS$.

- An *LTL fairness assumption* is a conjunction of LTL fairness constraints.

\[
\begin{align*}
\text{FairPaths}(s) & = \{ \pi \in \text{Paths}(s) \mid \pi \models fair \}, \\
\text{FairTraces}(s) & = \{ \text{trace}(\pi) \mid \pi \in \text{FairPaths}(s) \}.
\end{align*}
\]
AUTOMATA-BASED LTL MODEL CHECKING

To check whether $\phi$ holds for $TS$
Constructs an NBA for the negation of the input formula $\phi$ (representing the "bad behaviors")
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 Büchi automaton (GBA) over $\Sigma$ 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.
- $F = \{F_1, \ldots, F_k\} \subseteq 2^S$ 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 $F_1, \ldots, F_n$.
- A run $\pi$ of a GBA is said to be accepting iff, for all $1 \leq i \leq k$, we have $\inf(\pi) \cap F_i \neq \emptyset$. 


NBA & Generalized NBA (GNBA)

Nondeterministic Büchi Automaton
\[ A = (Q, \Sigma, \delta, I, F), \text{ where } F \subseteq Q \text{ is the set of accepting states.} \]

- A run \( \rho \) of \( A \) on omega word \( \alpha \) is an infinite sequence
  \[ \rho = q_0, q_1, q_2, \ldots \text{ s.t. } q_0 \in I \text{ and } q_i \xrightarrow{a_i} q_{i+1} \text{ for } 0 \leq i. \]
- The run \( \rho \) is accepting if
  \[ \mathsf{Inf}(\rho) \cap F \neq \emptyset. \]
- The language accepted by \( A \)
  \[ L(A) = \{ \alpha \in \Sigma^\omega \mid A \text{ has an accepting run on } \alpha \} \]

A Generalized Büchi Automaton is \( A := (Q, \Sigma, \delta, I, F_T) \) where
\[ FT = < F_1, F_2, \ldots, F_k > \text{ with } F_i \subseteq Q. \]
A run \( \rho \) of \( A \) is accepting if \( \mathsf{Inf}(\rho) \cap F_i \neq \emptyset \) for each \( 1 \leq i \leq k. \)
A GNBA for the property 
"both processes are
infinitely often in their
critical section"
$F = \{ \{q_1\}, \{q_2\}\}$

Sketch of transformation GNBA (with $|F| = k$) 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 $\phi$

- GNBA are like NBA, but have a distinct \textit{acceptance criterion}
- a GNBA requires to visit several sets $F_1, \ldots, F_k$ ($k \geq 0$) \textit{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 $\phi \Rightarrow$ Consisting of all subformulae $\psi$ of $\phi$ and their negation $\neg \psi$

\textbf{The Satisfiability Problem:}
- for a given LTL formula $\emptyset$, there exists a model for which $\emptyset$ holds. That is, we have $\text{Words}(\emptyset) = \emptyset$.

\textbf{The Validity problem:}
- Formula $\emptyset$ is valid whenever $\emptyset$ holds under all interpretations, i.e., $\phi \equiv \text{true}$. 
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.