Lecture 3
Linear Temporal Logic (LTL)

Richard M. Murray
Caltech

Ufuk Topcu
UT Austin

Nok Wongpiromsarn
UT Austin/Iowa State

EECI-IGSC, 9 Mar 2020

Outline
• Syntax and semantics of LTL
• Specifying properties in LTL
• Equivalence of LTL formulas
• Fairness in LTL
• Other temporal logics (if time)

Principles of Model Checking,
C. Baier and J.-P. Katoen,
The MIT Press, 2008
Chapter 5
Formal Methods for System Verification

Specification using LTL
- Linear temporal logic (LTL) is a math’l language for describing linear-time prop’s
- Provides a particularly useful set of operators for constructing LT properties without specifying sets

Methods for verifying an LTL specification
- *Theorem proving*: use formal logical manipulations to show that a property is satisfied for a given system model
- *Model checking*: explicitly check all possible executions of a system model and verify that each of them satisfies the formal specification
  - Roughly like trying to prove stability by simulating every initial condition
  - Works because discrete transition systems have finite number of states
  - Very good tools now exist for doing this efficiently (SPIN, nuSMV, etc)
Temporal Logic Operators

Two key operators in temporal logic
• ◊ “eventually” – a property is satisfied at some point in the future
• □ “always” – a property is satisfied now and forever into the future

“Temporal” refers underlying nature of time
• Linear temporal logic ⇒ each moment in time has a well-defined successor moment
• Branching temporal logic ⇒ reason about multiple possible time courses
• “Temporal” here refers to “ordered events”; no explicit notion of time

LTL = linear temporal logic
• Specific class of operators for specifying linear time properties
• Introduced by Pnueli in the 1970s (recently passed away)
• Large collection of tools for specification, design, analysis

Other temporal logics
• CTL = computation tree logic (branching time; will see later, if time)
• TCTL = timed CTL - check to make sure certain events occur in a certain time
• TLA = temporal logic of actions (Lamport) [variant of LTL]
• μ calculus = for reactive systems; add “least fixed point” operator (more on Thu)
Syntax of LTL

LTL formulas:

\[ \varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbf{U} \varphi_2 \]

- \( a \) = atomic proposition
- \( \bigcirc = \text{"next"} \): \( \varphi \) is true at next step
- \( \mathbf{U} = \text{"until"} \): \( \varphi_2 \) is true at some point, \( \varphi_1 \) is true until that time

Operator precedence:
- Unary bind stronger than binary
- \( \mathbf{U} \) takes precedence over \( \land, \lor \) and \( \rightarrow \)

Formula evaluation: evaluate LTL propositions over a sequence of states (path):

- Same notation as linear time properties: \( \sigma \models \varphi \) (path satisfies specification)
Additional Operators and Formulas

“Primary” temporal logic operators

- Eventually $\Diamond \phi := true \cup \phi$  \quad $\phi$ will become true at some point in the future
- Always $\square \phi := \neg \neg \neg \phi$  \quad $\phi$ is always true; "(never (eventually (\neg \phi)))"

Some common composite operators

- $p \rightarrow \Diamond q$  \quad $p$ implies eventually $q$ (response)
- $p \rightarrow q \cup r$  \quad $p$ implies $q$ until $r$ (precedence)
- $\square \Diamond p$  \quad always eventually $p$ (progress)
- $\Diamond \square p$  \quad eventually always $p$ (stability)
- $\Diamond p \rightarrow \Diamond q$  \quad eventually $p$ implies eventually $q$ (correlation)

Operator precedence

- Unary binds stronger than binary
- Bind from right to left: $\square \Diamond p = (\square (\Diamond p))$
  $p \cup q \cup r = p \cup (q \cup r)$
- $U$ takes precedence over $\wedge$, $\vee$ and $\rightarrow$
Example: Traffic Light

System description
- Focus on lights in one particular direction
- Light can be any of three colors: green, yellow, read
- Atomic propositions = light color

Ordering specifications
- Liveness: “traffic light is green infinitely often”
  \[ \square \Diamond \text{green} \]
- Chronological ordering: “once red, the light cannot become green immediately”
  \[ \square (\text{red} \rightarrow \neg \Diamond \text{green}) \]
- More detailed: “once red, the light always becomes green eventually after being yellow for some time”
  \[ \square (\text{red} \rightarrow (\Diamond \text{green} \land (\neg \text{green} \lor \Diamond \text{yellow}))) \]
  \[ \square (\text{red} \rightarrow \Diamond (\text{red} \lor \Diamond (\text{yellow} \lor \Diamond (\text{yellow} \lor \Diamond \text{green})))) \]

Progress property
- Every request will eventually lead to a response
  \[ \square (\text{request} \rightarrow \Diamond \text{response}) \]
Semantics: when does a path satisfy an LTL spec?

Definition 5.6. Semantics of LTL (Interpretation over Words)

Let $\varphi$ be an LTL formula over $AP$. The LT property induced by $\varphi$ is

$$\text{Words}(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

where the satisfaction relation $\models \subseteq (2^{AP})^\omega \times \text{LTL}$ is the smallest relation with the properties in Figure 5.2.

\[
\begin{align*}
\sigma &\models \text{true} \\
\sigma &\models a \quad \text{iff} \quad a \in A_0 \quad (\text{i.e., } A_0 \models a) \\
\sigma &\models \varphi_1 \land \varphi_2 \quad \text{iff} \quad \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
\sigma &\models \neg \varphi \quad \text{iff} \quad \sigma \not\models \varphi \\
\sigma &\models \Box \varphi \quad \text{iff} \quad \exists j \geq 0. \sigma[j...] \models \varphi \\
\sigma &\models \varphi_1 \lor \varphi_2 \quad \text{iff} \quad \exists j \geq 0. \sigma[j...] \models \varphi_2 \text{ and } \sigma[i...] \models \varphi_1, \text{ for all } 0 \leq i < j
\end{align*}
\]

Figure 5.2: LTL semantics (satisfaction relation $\models$) for infinite words over $2^{AP}$. 
Semantics of LTL

The semantics of the combinations of □ and ◇ can now be derived:

\[ \sigma \models □ ◇ \varphi \iff \exists j. \sigma[j\ldots] \models \varphi \]

\[ \sigma \models ◇ □ \varphi \iff \forall j. \sigma[j\ldots] \models \varphi. \]

Here, \( \exists j \) means \( \forall i \geq 0. \exists j \geq i \), “for infinitely many \( j \in \mathbb{N} \)”, while \( \forall j \) stands for \( \exists i \geq 0. \forall j \geq i, “for almost all j \in \mathbb{N}” \).

Definition 5.7. Semantics of LTL over Paths and States

Let \( TS = (S, Act, \rightarrow, I, AP, L) \) be a transition system without terminal states, and let \( \varphi \) be an LTL-formula over \( AP \).

- For infinite path fragment \( \pi \) of \( TS \), the satisfaction relation is defined by
  \[ \pi \models \varphi \iff \text{trace}(\pi) \models \varphi. \]

- For state \( s \in S \), the satisfaction relation \( \models \) is defined by
  \[ s \models \varphi \iff (\forall \pi \in \text{Paths}(s). \pi \models \varphi). \]

- \( TS \) satisfies \( \varphi \), denoted \( TS \models \varphi \), if \( \text{Traces}(TS) \subseteq \text{Words}(\varphi) \).
Semantics of LTL

From this definition, it immediately follows that

\[ TS \models \varphi \]

iff

\[ \text{Traces}(TS) \subseteq \text{Words}(\varphi) \]

iff

\[ TS \models \text{Words}(\varphi) \]

iff

\[ \pi \models \varphi \text{ for all } \pi \in \text{Paths}(TS) \]

iff

\[ s_0 \models \varphi \text{ for all } s_0 \in I. \]

(* Definition 5.7 *)

(* Definition of \( \models \) for LT properties *)

(* Definition of \( \text{Words}(\varphi) \) *)

(* Definition 5.7 of \( \models \) for states *)

Remarks

- Which condition you use depends on type of problem under consideration
- For reasoning about correctness, look for (lack of) intersection between sets:
Consider the following transition system

Consider the transition system $TS$ depicted in Figure 5.3 with the set of propositions $AP = \{a, b\}$. For example, we have that $TS \models \square a$, since all states are labeled with $a$, and hence, all traces of $TS$ are words of the form $A_0 A_1 A_2 \ldots$ with $a \in A_i$ for all $i \geq 0$. Thus, $s_i \models \square a$ for $i = 1, 2, 3$. Moreover:

$s_1 \models \Diamond (a \land b)$ since $s_2 \models a \land b$ and $s_2$ is the only successor of $s_1$

$s_2 \not\models \Diamond (a \land b)$ and $s_3 \not\models \Diamond (a \land b)$ as $s_3 \in \text{Post}(s_2)$, $s_3 \in \text{Post}(s_3)$ and $s_3 \not\models a \land b$.

This yields $TS \not\models \Diamond (a \land b)$ as $s_3$ is an initial state for which $s_3 \not\models \Diamond (a \land b)$. As another example:

$TS \models \square(\neg b \rightarrow \square(a \land \neg b))$,

since $s_3$ is the only $\neg b$ state, $s_3$ cannot be left anymore, and $a \land \neg b$ in $s_3$ is true. However,

$TS \not\models b \cup (a \land \neg b)$,

since the initial path $(s_1 s_2)\omega$ does not visit a state for which $a \land \neg b$ holds. Note that the initial path $(s_1 s_2)^* s_3^\omega$ satisfies $b \cup (a \land \neg b)$.
Specifying Timed Properties for Synchronous Systems

For synchronous systems, LTL can be used as a formalism to specify “real-time” properties that refer to a discrete time scale. Recall that in synchronous systems, the involved processes proceed in a lock step fashion, i.e., at each discrete time instance each process performs a (sometimes idle) step. In this kind of system, the next-step operator $\bigcirc$ has a “timed” interpretation: $\bigcirc \varphi$ states that “at the next time instant $\varphi$ holds”. By putting applications of $\bigcirc$ in sequence, we obtain, e.g.:

$$
\bigcirc^k \varphi \overset{\text{def}}{=} \bigcirc \bigcirc \ldots \bigcirc \varphi \quad \text{“$\varphi$ holds after (exactly) $k$ time instants”}.
$$

Assertions like “$\varphi$ will hold within at most $k$ time instants” are obtained by

$$
\lozenge \leq^k \varphi = \bigvee_{0 \leq i \leq k} \bigcirc^i \varphi.
$$

Statements like “$\varphi$ holds now and will hold during the next $k$ instants” can be represented as follows:

$$
\square \leq^k \varphi = \neg \lozenge \leq^k \neg \varphi = \neg \bigvee_{0 \leq i \leq k} \bigcirc^i \neg \varphi.
$$

Remark

- Idea can be extended to non-synchronous case (e.g., Timed CTL [later])
### Equivalence of LTL Formulas

**Definition 5.17. Equivalence of LTL Formulae**

LTL formulae $\varphi_1, \varphi_2$ are *equivalent*, denoted $\varphi_1 \equiv \varphi_2$, if $\text{Words}(\varphi_1) = \text{Words}(\varphi_2)$.

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

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

<table>
<thead>
<tr>
<th>Distributive Law</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Box (\varphi U \psi) \equiv (\Box \varphi) U (\Box \psi)$</td>
</tr>
<tr>
<td>$\Diamond (\varphi \lor \psi) \equiv \Diamond \varphi \lor \Diamond \psi$</td>
</tr>
<tr>
<td>$\Box (\varphi \land \psi) \equiv \Box \varphi \land \Box \psi$</td>
</tr>
</tbody>
</table>

**Non-identities**

- $\Diamond (a \land b) \neq \Diamond a \land \Diamond b$
- $\Box (a \lor b) \neq \Box a \lor \Box b$
Distributed Systems

Distributed systems
- A distributed system consists of a set of agents (also called processes) and a set of directed channels.
- A channel is directed from one agent to one agent. The system can be represented by a directed graph (separate from the program graph within each agent).

Definition of the “state” of a distributed system
- Minimum amount of information such that the future behavior can be predicted without any other information about the past
- Typically consists of the value of all variables that are part of any processes as well as messages that might be in transit

Execution for a distributed system
- Any agent can execute an action within its transition system at any time
- Synchrony models
  - Handshake (see L2)
  - Interleaving: async
  - Synchronized actions
- “Fairness”: assumption to ensure everyone gets to execute
Fairness

Weak Fairness
• Every action is guaranteed to be selected infinitely often
• Implication: between any two selections of a particular action, there are a finite (but unbounded) number of selections of other actions.

Strong Fairness
• Each action is selected infinitely often and if an action is enabled infinitely often then it is selected infinitely often
• Avoids situations where we get “unlucky” and never select an action at a time when it is enabled (mainly applies to guarded actions)

Door opening example
• Human can walk forward or backward
• Door can open or close (asynchronously)
• Treat as two separate processes
  - Human: actions = forward, backward
  - Door: actions = open, close
• Q: is it always possible for the human to get from one side of the door to the other?
Fairness Properties in LTL

**Definition 5.25  LTL Fairness Constraints and Assumptions**

Let $\Phi$ and $\Psi$ be propositional logical formulas over a set of atomic propositions

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

An LTL fairness assumption is a conjunction of LTL fairness constraints (of any arbitrary type).

$$fair = ufair \land sfair \land wfair.$$ 

**Rules of thumb**

- strong (or unconditional) fairness: useful for solving contentions
- weak fairness: sufficient for resolving the non-determinism due to interleaving.
Fairness Properties in LTL

Fair paths and traces

\[ \text{FairPaths}(s) = \{ \pi \in \text{Paths}(s) \mid \pi \models \text{fair} \}, \]
\[ \text{FairTraces}(s) = \{ \text{trace}(\pi) \mid \pi \in \text{FairPaths}(s) \}. \]

Definition 5.26. Satisfaction Relation for LTL with Fairness

For state \( s \) in transition system \( TS \) (over \( AP \)) without terminal states, LTL formula \( \varphi \), and LTL fairness assumption \( \text{fair} \) let

\[ s \models \text{fair} \varphi \text{ iff } \forall \pi \in \text{FairPaths}(s). \pi \models \varphi \text{ and } \]
\[ TS \models \text{fair} \varphi \text{ iff } \forall s_0 \in I. s_0 \models \text{fair} \varphi. \]

Theorem 5.30. Reduction of \( \models \text{fair} \) to \( \models \)

For transition system \( TS \) without terminal states, LTL formula \( \varphi \), and LTL fairness assumption \( \text{fair} \):

\[ TS \models \text{fair} \varphi \text{ if and only if } TS \models (\text{fair} \rightarrow \varphi). \]
Branching Time and Computational Tree Logic

Consider transition systems with multiple branches

- Eg, nondeterministic finite automata (NFA), nondeterministic Bucchi automata (NBA)
- In this case, there might be *multiple* paths from a given state
- Q: in evaluating a temporal logic property, which execution branch to we check?

![Diagram showing transition system with multiple branches](image)

**Computational tree logic:** allow evaluation over some or all paths

\[
\begin{align*}
s & \models \exists \varphi \quad \text{iff} \quad \pi & \models \varphi \text{ for some } \pi \in \text{Paths}(s) \\
s & \models \forall \varphi \quad \text{iff} \quad \pi & \models \varphi \text{ for all } \pi \in \text{Paths}(s)
\end{align*}
\]
Example: Triply Redundant Control Systems

Systems consists of three processors and a single voter

- $s_{i,j} = i$ processors up, $j$ voters up
- Assume processors fail one at a time; voter can fail at any time
- If voter fails, reset to fully functioning state (all three processors up)
- System is operation if at least 2 processors remain operational

Properties we might like to prove

<table>
<thead>
<tr>
<th>Property</th>
<th>Formalization in CTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possibly the system never goes down</td>
<td>$\exists \Box \neg \text{down}$</td>
</tr>
<tr>
<td>Invariantly the system never goes down</td>
<td>$\forall \Box \neg \text{down}$</td>
</tr>
<tr>
<td>It is always possible to start as new</td>
<td>$\forall \Box \exists \Diamond \text{up}_3$</td>
</tr>
<tr>
<td>The system always eventually goes down and is operational until going down</td>
<td>$\forall (\left(\text{up}_3 \lor \text{up}_2\right) \cup \text{down})$</td>
</tr>
</tbody>
</table>

Holds

Doesn’t hold

Holds

Doesn’t hold
Other Types of Temporal Logic

CTL ≠ LTL

- Can show that LTL and CTL are not proper subsets of each other
- LTL reasons over a complete path; CTL from a given state

CTL* captures both

\[ \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \varphi \]

\[ \varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \varnothing \varphi \mid \varphi_1 \lor \varphi_2 \]

Timed Computational Tree Logic

- Extend notions of transition systems and CTL to include “clocks” (multiple clocks OK)
- Transitions can depend on the value of clocks
- Can require that certain properties happen within a given time window

\[ \forall \square (far \rightarrow \forall \Diamond \leq 1 \forall \square \leq 1 \text{ up}) \]
Summary: Specifying Behavior with LTL

Description
- State of the system is a snapshot of values of all variables
- Reason about paths $\sigma$: sequence of states of the system
- No strict notion of time, just ordering of events
- **Actions** are relations between states: state $s$ is related to state $t$ by action $a$ if $a$ takes $s$ to $t$ (via prime notation: $x' = x + 1$)
- **Formulas** (specifications) describe the set of allowable behaviors
- Safety specification: what actions are allowed
- Fairness specification: when can a component take an action (eg, infinitely often)

Example
- Action: $a \equiv x' = x + 1$
- Behavior: $\sigma \equiv x := 1, x := 2, x := 3, ...$
- Safety: $\square x > 0$ (true for this behavior)
- Fairness: $\square(x' = x + 1 \lor x' = x) \land \square\diamond (x' \neq x)$

Properties
- Can reason about time by adding “time variables” ($t' = t + 1$)
- Specifications and proofs can be difficult to interpret by hand, but computer tools existing (eg, TLC, Isabelle, PVS, SPIN, Storm, etc)

$\square p \equiv$ always $p$ (invariance)
$\diamond p \equiv$ eventually $p$ (guarantee)
$p \rightarrow \diamond q \equiv$ implies eventually $q$ (response)
$p \rightarrow q \ U r \equiv$ implies $q$ until $r$ (precedence)
$\square\diamond p \equiv$ always eventually $p$ (progress)
$\diamond\square p \equiv$ eventually always $p$ (stability)
$\diamond p \rightarrow \diamond q \equiv$ eventually $p$ implies eventually $q$ (correlation)