Outline

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


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:

$$\varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

- $a =$ atomic proposition
- $\bigcirc =$ “next”: $\varphi$ is true at next step
- $\mathcal{U} =$ “until”: $\varphi_2$ is true at some point, $\varphi_1$ is true until that time

Operator precedence

- Unary bind stronger than binary
- $\mathcal{U}$ 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)
“Primary” temporal logic operators

- **Eventually** \( \Diamond \phi := \text{true} U \phi \) \( \phi \) will become true at some point in the future
- **Always** \( \Box \phi := \neg \Diamond \neg \phi \) \( \phi \) is always true; “(never (eventually (\neg \phi)))”

Some common composite operators

- \( p \rightarrow \Diamond q \) \( p \) implies eventually \( q \) (response)
- \( p \rightarrow q U r \) \( p \) implies \( q \) until \( r \) (precedence)
- \( \Box \Diamond p \) always eventually \( p \) (progress)
- \( \Diamond \Box p \) eventually always \( p \) (stability)
- \( \Diamond p \rightarrow \Diamond q \) eventually \( p \) implies eventually \( q \) (correlation)

Operator precedence

- Unary binds stronger than binary
- Bind from right to left:
  \( \Box \Diamond p = (\Box (\Diamond p)) \)
  \( p U q U r = p U (q U r) \)
- \( U \) takes precedence over \( \land, \lor \) and \( \rightarrow \)
Example: Traffic Light

System description
- Focus on lights in one particular direction
- Light can be any of three colors: green, yellow, red
- 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”
  \( \square (\text{red} \rightarrow (\Diamond \text{green} \land (\neg \text{green} \lor (\neg \text{green} \lor \Diamond \text{green})))) \)

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.

$\sigma \models \text{true}$
$\sigma \models a$ iff $a \in A_0$ (i.e., $A_0 \models a$)
$\sigma \models \varphi_1 \land \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$
$\sigma \models \Diamond \varphi$ iff $\exists j \geq 0. \sigma[j \ldots] \models \varphi$
$\sigma \models \neg \varphi$ iff $\sigma \not\models \varphi$
$\sigma \models \Box \varphi$ iff $\forall j \geq 0. \sigma[j \ldots] \models \varphi$
$\sigma \models \bigcirc \varphi$ iff $\sigma[1 \ldots] = A_1A_2A_3\ldots \models \varphi$
$\sigma \models \varphi_1 \cup \varphi_2$ iff $\exists j \geq 0. \sigma[j \ldots] \models \varphi_2$ and $\sigma[i \ldots] \models \varphi_1$, for all $0 \leq i < j$

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

The semantics of the combinations of $\square$ and $\Diamond$ can now be derived:

$$\sigma \models \square \Diamond \varphi \iff \exists j. \sigma[j \ldots] \models \varphi$$

$$\sigma \models \Diamond \square \varphi \iff \forall j. \sigma[j \ldots] \models \varphi.$$

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

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

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 $TS |= \square a$, 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, $s_i |= \square a$ for $i = 1, 2, 3$. Moreover:

$s_1 |= \Diamond (a \land b)$ since $s_2 \models a \land b$ and $s_2$ is the only successor of $s_1$
$s_2 \notmodels \Diamond (a \land b)$ and $s_3 \notmodels \Diamond (a \land b)$ as $s_3 \in \text{Post}(s_2)$, $s_3 \in \text{Post}(s_3)$ and $s_3 \notmodels a \land b$.

This yields $TS \notmodels \Diamond (a \land b)$ as $s_3$ is an initial state for which $s_3 \notmodels \Diamond (a \land b)$. 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,

$TS \notmodels b \lor (a \land \neg b)$,

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 $b \lor (a \land \neg b)$.

■
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.:

$$\bigcirc^k \varphi \overset{\text{def}}{=} \underbrace{\bigcirc \bigcirc \ldots \bigcirc}_{\text{k-times}} \varphi$$

“$\varphi$ holds after (exactly) $k$ time instants”.

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

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

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

$$\Box \leq k \varphi = \neg \Diamond \leq k \neg \varphi = \neg \bigvee_{0 \leq i \leq k} \bigcirc^i \neg \varphi.$$

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>$\neg \Diamond \varphi \equiv \Box \neg \varphi$</td>
<td>$\Diamond \Diamond \varphi \equiv \Diamond \varphi$</td>
</tr>
<tr>
<td>$\neg \Box \varphi \equiv \Diamond \neg \varphi$</td>
<td>$\Box \Box \varphi \equiv \Box \varphi$</td>
</tr>
<tr>
<td>$\neg \Box \varphi \equiv \Diamond \neg \varphi$</td>
<td>$\varphi \mathcal{U} (\varphi \mathcal{U} \psi) \equiv \varphi \mathcal{U} \psi$</td>
</tr>
<tr>
<td>$(\varphi \mathcal{U} \psi) \mathcal{U} \psi \equiv \varphi \mathcal{U} \psi$</td>
<td></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>$\varphi \mathcal{U} \psi \equiv \psi \lor (\varphi \land (\Box (\varphi \mathcal{U} \psi)))$</td>
</tr>
<tr>
<td>$\Box \Diamond \varphi \equiv \Diamond \Box \varphi$</td>
<td>$\Diamond \psi \equiv \psi \lor \Diamond \Diamond \psi$</td>
</tr>
<tr>
<td>$\Box \psi \equiv \psi \land \Box \Box \psi$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distributive law</th>
<th>Non-identities</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Box (\varphi \mathcal{U} \psi) \equiv (\Box \varphi) \mathcal{U} (\Box \psi)$</td>
<td>$\Diamond (a \land b) \neq \Diamond a \land \Diamond b$</td>
</tr>
<tr>
<td>$\Diamond (\varphi \lor \psi) \equiv \Diamond \varphi \lor \Diamond \psi$</td>
<td>$\Box (a \lor b) \neq \Box a \lor \Box b$</td>
</tr>
<tr>
<td>$\Box (\varphi \land \psi) \equiv \Box \varphi \land \Box \psi$</td>
<td></td>
</tr>
</tbody>
</table>

EECI, Mar 2013 Richard M. Murray, Caltech CDS 12
LTL Specs for Control Protocols: RoboFlag Drill

Task description
- Incoming robots should be blocked by defending robots
- Incoming robots are assigned randomly to whoever is free
- Defending robots must move to block, but cannot run into or cross over others
- Allow robots to communicate with left and right neighbors and switch assignments

Goals
- Would like a provably correct, distributed protocol for solving this problem
- Should (eventually) allow for lost data, incomplete information

Questions
- How do we describe task in terms of LTL?
- Given a protocol, how do we prove specs?
- How do we design the protocol given specs?
Properties for RoboFlag program

CCL formulas (will cover in more detail later)
• \( q' \)  
  • \( q \)  
  evaluate \( q \) at the next action in path

• \( p \rightarrow q \)  
  \( \Box (p \rightarrow \Diamond q) \)  
  “\( p \) leads to \( q \)”: if \( p \) is true, \( q \) will eventually be true

• \( p \text{ co } q \)  
  “\( \Box (p \rightarrow \Diamond q) \)”  
  if \( p \) is true, then next time state changes, \( q \) will be true

Safety (Defenders do not collide)
\[
\forall i . \ y_i > 2\delta \land z_i + 2\delta < z_{i+1} \land \neg switch_{i,i+1} \text{ co } \neg switch_{i,i+1}
\]
True if robots \( i \) and \( i+1 \) have targets that cause crossed paths

Stability (switch predicate stays false)
\[
\forall i . \ 2\delta \land z_i + 2\delta < z_{i+1} \land \neg switch_{i,i+1} \text{ co } \neg switch_{i,i+1}
\]
Robots are "far enough" apart.

“Lyapunov” stability
• Remains to show that we actually approach the goal (robots line up with targets)
• Will see later we can do this using a Lyapunov function
Fairness

Mainly an issue with concurrent processes

- To make sure that the proper interaction occurs, often need to know that each process gets executed reasonably often
- Multi-threaded version: each thread should receive some fraction of processes time

Two issues: implementation and specification

- Q1: How do we implement our algorithms to insure that we get “fairness” in execution
- Q2: how do we model fairness in a formal way to reason about program correctness

Example: Fairness in RoboFlag Drill

- To show that algorithm behaves properly, need to know that each agent communicates with neighbors regularly (infinitely often), in each direction

Difficulty in describing fairness depends on the logical formalism

- Turns out to be pretty easy to describe fairness in linear temporal logic
- Much more difficult to describe fairness for other temporal logics (eg, CTL & variants)
**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 $u_{\text{fair}} = \Box \Diamond \Psi$.

2. A *strong LTL fairness condition* is an LTL formula of the form $s_{\text{fair}} = \Box \Diamond \Phi \rightarrow \Box \Diamond \Psi$.

3. A *weak LTL fairness constraint* is an LTL formula of the form $w_{\text{fair}} = \Diamond \Box \Phi \rightarrow \Box \Diamond \Psi$.

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

$$
\text{fair} = u_{\text{fair}} \land s_{\text{fair}} \land w_{\text{fair}}
$$

**Rules of thumb**

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

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

Theorem 5.30. Reduction of \( \models_{\text{fair}} \) to \( \models \)
For transition system \( TS \) without terminal states, LTL formula \( \varphi \), and LTL fairness assumption \( \text{fair} \):

\[
TS \models_{\text{fair}} \varphi \quad \text{if and only if} \quad TS \models (\text{fair} \rightarrow \varphi).
\]
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 do we check?

Computational tree logic: allow evaluation over some or all paths

\[
\begin{align*}
  s \models \exists \varphi & \iff \pi \models \varphi \text{ for some } \pi \in \text{Paths}(s) \\
  s \models \forall \varphi & \iff \pi \models \varphi \text{ for all } \pi \in \text{Paths}(s)
\end{align*}
\]
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>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possibly the system never goes down</td>
<td>( \exists \Box \neg \text{down} )</td>
<td>Holds</td>
</tr>
<tr>
<td>Invariantly the system never goes down</td>
<td>( \forall \Box \neg \text{down} )</td>
<td>Doesn’t hold</td>
</tr>
<tr>
<td>It is always possible to start as new</td>
<td>( \forall \Box \exists \Diamond \text{up}_3 )</td>
<td>Holds</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) \cup \text{down}\right) )</td>
<td>Doesn’t hold</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Linear time</th>
<th>Branching time</th>
</tr>
</thead>
<tbody>
<tr>
<td>“behavior” in a state $s$</td>
<td>path-based: $\text{trace}(s)$</td>
<td>state-based: computation tree of $s$</td>
</tr>
</tbody>
</table>
| temporal logic          | LTL: path formulae $\varphi$ $s \models \varphi$ iff $\forall \pi \in \text{Paths}(s). \pi \models \varphi$ | CTL: state formulae existential path quantification $\exists \varphi$ universal path quantification: $\forall \varphi$

CTL* captures both

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

$\varphi ::= \Phi \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 \lor \varphi_2$

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

$\forall \Box (\text{far} \rightarrow \forall \Diamond \leq 1 \land \forall \Box \leq 1 \text{ up})$
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, etc)

- $\Box p \equiv \text{always } p$ (invariance)
- $\Diamond p \equiv \text{eventually } p$ (guarantee)
- $p \rightarrow \Diamond q \equiv p \text{ implies eventually } q$ (response)
- $p \rightarrow q \cup r \equiv q \text{ until } r$ (precedence)
- $\Box \Diamond p \equiv \text{always eventually } p$ (progress)
- $\Diamond \Box p \equiv \text{eventually always } p$ (stability)
- $\Diamond p \rightarrow \Diamond q \equiv \text{eventually } p \text{ implies eventually } q$ (correlation)