Lecture 19
Linear Temporal Logic (LTL)

David Garlan

Carnegie Mellon University

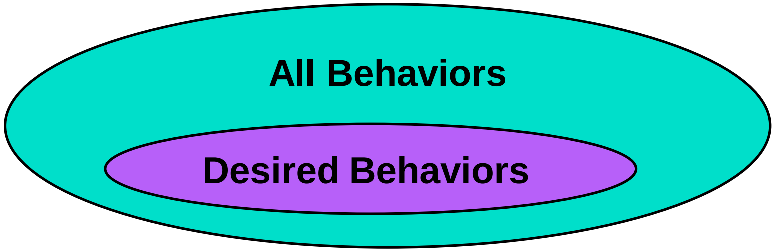
Models of Software Systems
Fall 2016

This Lecture

- **Linear Temporal Logic**
  - > **safety & liveness review**
  - > **basic temporal logic (always, eventually, next)**
  - > **examples and common patterns**

Recall: Safety versus Liveness

- As we have noted, sometimes a distinction is made between **safety** and **liveness** properties of a model
  - > safety: a property that guarantees that no system invariant is violated
  - > liveness: a property that guarantees that something useful actually happens
- Thus far we have mostly concentrated on safety properties



Safety

Safety: "nothing bad happens"

> This is usually phrased as an invariant of the system representing "good" states
  - » Then we show that each initial state is a good state, and that every transition takes us from a good state to another good state.

> Notations like Z are particularly well-suited to this kind of property

> Note: a system that does nothing usually satisfies its safety properties

> Examples
  - » the # of candies is never more than the # coins
  - » dom birthday = names
  - » no two threads can execute the same critical code at the same time

Liveness

Liveness: "something good happens"

- Usually phrased as promise of an eventual outcome
  - Then we must show that every/some trace of the system leads to desired state
- Note: these are statements about what a system will do over time
- A stopped system will, in general, not satisfy these conditions
- Examples
  - the system will eventually halt
  - whenever a happens b will eventually follow
  - a system is in the ready state infinitely often
  - if a certain transition is enabled, it will eventually occur

Temporal Logics

- A "Temporal Logic" characterizes the behavior of systems by specifying some property about system traces
- We have already seen such specifications in FSP
- Examples

```
property PROP = (req -> reply -> PROP).
```

Guarantees that requests and replies strictly alternate

```
progress {heads}
```

Guarantees that the “heads” event happens infinitely often

Temporal Logics

But now a couple of things will be different

1. **We focus on state-based traces**

   <s_1, s_2, ... , $s_{n,...}$>, where s_i is a state of the system, and traces start with index 1

2. **We consider all traces to be infinite**

   for finite traces, the final state is simply repeated

3. **We then introduce new notation to express common kinds of properties about these traces such as**

   » **Some property *always* holds**

   » **Some property *will eventually* hold**

   » **Some property *always follows* after another property becomes true**

   » **Some property *is true infinitely often***

(Linear) Temporal Logic: Syntax

tempformula =

predicate

| "~", tempformula

| "(", tempformula, "∨", tempformula, ")"

| "(", tempformula, "∧", tempformula, ")"

| "(", tempformula, "⇒", tempformula, ")"

| "(", tempformula, "⇔", tempformula, ")"

| "□", tempformula

| "◊", tempformula

| tempformula, "U", tempformula

| tempformula, "Uw", tempformula

| " ○", tempformula;

Note: No Quantification over temporal formulae

© Garlan 2016

(Linear) Temporal Logic: Semantics

- **Temporal formulae are interpreted over traces.**
- **Let $\sigma = \langle s_1, s_2, \ldots, s_i, \ldots \rangle$ be a (state-based) trace, and P be a temporal logic formula**
- **We write**

  $(\sigma, i) \models P$ and say "P holds in the i^{th} state of trace σ".
- **When we write just P, we will mean that P holds in the <u>first</u> state of the sequence σ**

  -- that is, P is the same as $(\sigma,1) \models P$

Temporal Operators: Semantic Intuition

- $\Box$ P – “always”: P holds in *every* state of a trace.
- $\Diamond$ P – “eventually”: P holds in *at least one* state of a trace.
- $\circ$ P – “next”: P holds in the *next* state in a trace.
- P **U** Q– “until”: P holds *until* Q becomes true

Temporal Operators (always)

- $\Box$
  - > Pronounced "box", "henceforth", "always", "from now on", "forever"

$$(\sigma , i) \vDash \Box P$$

means

$$\forall j: i.. \bullet (\sigma , j) \vDash P$$

- When we write just $\Box$ P, we will mean

  P holds for all states in the trace starting at i = 1

  That is, $\Box$ P is the same as $(\sigma , 1) \vDash \Box P$

Temporal Operators (eventually)

- $\Diamond$
  > Pronounced "diamond", "eventually", "sometime"

$$(\sigma, i) \models \Diamond P$$

means

$$\exists j: i.. \bullet (\sigma, j) \models P$$

- **When we write just $\Diamond P$, we will mean**

  starting at i = 1, there is *some* state in the trace for which P holds

  That is, $\Diamond P$ is the same as $(\sigma, 1) \models \Diamond P$

Note on Operators

- **Note that we have just introduced a new way to define predicates**
  - > **The difference is that the predicates are applied to traces: i.e., to sequences of states (*not to individual states*)**
- **Operators such as □ and ◊ are sometimes called "modalities"**
  - > **they are similar to existential and universal "quantifiers" in "quantified expressions" for FOPL**
- **A predicate of the form ◊P is called an "eventuality"**

Precedence Rules

- $\Box$ and $\Diamond$ bind tighter than other logical operators
- So if we want to say that the predicate $P \Rightarrow Q$ is true in every state, we would write $\Box (P \Rightarrow Q)$
- In contrast, $\Box P \Rightarrow Q$ means $(\Box P) \Rightarrow Q$, which as a very different meaning
  - > We'll see what that is in a few slides

Formally Defining $\models$

$(\sigma, i) \models P \iff P(\sigma(i))$ is true

provided P contains no modalities
(i.e., P is a FOPL predicate, may include quantifiers)

$(\sigma, i) \models P \vee Q \iff (\sigma, i) \models P \vee (\sigma, i) \models Q$

$(\sigma, i) \models P \Rightarrow Q \iff (\sigma, i) \models P \Rightarrow (\sigma, i) \models Q$

other operators (~, $\wedge$, etc.) are similar

$(\sigma, i) \models \Box P \iff \forall j: i .. \bullet (\sigma, j) \models P$

$(\sigma, i) \models \Diamond P \iff \exists j: i .. \bullet (\sigma, j) \models P$

Example

- Suppose we our state consists of two values: x, y: $\mathbb{Z}$
- Consider the following two predicates: P(x,y) == x > 0 and Q(x,y) == x > y
- Let σ == <(-1,0), (0,1), (1,2), (2,3), … >
  (increments both x and y)

- Which of the following are true of σ?

1. P
2. Q
3. $\Box P$
4. $\Diamond P$
5. $\Diamond P$
6. $P \wedge Q$
7. $\Box (P \wedge Q)$
8. $\Box P \wedge Q$
9. $\Box P \vee Q$
10. $\sim \Box P$
10. $P \Rightarrow Q$
11. $\Box (P \Rightarrow Q)$

Examples using Box

- $\Box\ (x > 0)$

  an invariant of the model
- $\Box\ (\sim (\ in\_crit(p_1) \wedge in\_crit(p_2)\ )\ )$

  mutual exclusion
- $(\sigma\ ,\ 2) \models \sim at(init)$
- $(\sigma\ ,\ 2) \models \Box\ (\sim at(init)\ )$
- $(\sigma\ ,\ 2) \models \sim \Box\ (at(init)\ )$
- $at(halt) \Rightarrow \Box\ (\ at(halt)\ )$

  Is this really what we want?
- $\Box\ (P \Rightarrow \Box\ P)$

  once P, always P

© Garlan 2016

Nested Modalities

$\text{at(halt)} \Rightarrow \Box\ \text{at(halt)}$

$= (\sigma, 1) \models \text{at(halt)} \Rightarrow \Box\ \text{at(halt)}$

$= (\sigma, 1) \models \text{at(halt)} \Rightarrow (\sigma, 1) \models \Box\ \text{at(halt)}$

$= (\sigma, 1) \models \text{at(halt)} \Rightarrow \forall\ j: 1..\ \bullet\ (\sigma, j) \models \text{at(halt)}$

Compare this to

$\Box\ (\text{at(halt)} \Rightarrow \Box\ \text{at(halt))}$

$= (\sigma, 1) \models \Box\ (\text{at(halt)} \Rightarrow \Box\ \text{at(halt)}$

$= \forall\ i: 1..\ \bullet\ (\sigma, i) \models (\text{at(halt)} \Rightarrow \Box\ \text{at(halt))}$

$= \forall\ i: 1..\ \bullet\ (\sigma, i) \models \text{at(halt))} \Rightarrow (\sigma, i) \models \Box\ \text{at(halt)}$

$= \forall\ i: 1..\ \bullet\ (\sigma, i) \models \text{at(halt)} \Rightarrow$

$\forall\ j: i..\ \bullet\ (\sigma, j) \models \text{at(halt)}$

Partial Correctness

- **For systems characterized in terms of**

  pre: P

  post: Q

- **$P \Rightarrow \Box\, (\, at(halt) \Rightarrow Q\,)$**

  Note operator precedence requires ()

  > **specifies partial correctness, assuming that at(halt) means the program has terminated**

  > **i.e., if P is true initially, and the program terminates, then Q will be true**

- **$P \wedge \Box\, (\, at(halt) \Rightarrow Q\,)$**

  alternative partial correctness definition

Qn: What is the difference?

Examples using Diamond

- $\Diamond (x > 0)$
- $\Diamond (\sim in\_crit(p_1) \wedge \sim in\_crit(p_2))$
- $(\sigma , 2) \models \Diamond (\sim at(init))$
- $at(init) \Rightarrow \Diamond (at(halt))$
- $in\_crit(p_1) \Rightarrow \Diamond (\sim in\_crit(p_1))$
- $(P \Rightarrow (\Box (at(halt) \Rightarrow Q)) \wedge (\Diamond at(halt)))$
  - total correctness

Combining Box and Diamond

- **◊ □ P**

  "Eventually P will be true forever“

  > **Example: ◊ □ at(halt)**

- **□ ◊P**

  "Henceforth P will eventually be true"

  > **In every state from now on P will eventually become true**

  P is true infinitely often

  > **Note that if traces are finite, then P must hold in the final state**

Example: Expanding $\Diamond \Box P$

$\Diamond \Box P$

=

$(\sigma , 1) \models \Diamond \Box P$

=

$\exists j: 1 .. \bullet (\sigma , j) \models \Box P$

=

$\exists j: 1 .. \bullet (\forall k: j .. \bullet (\sigma , k) \models P)$

Example: Expanding $\Box \Diamond P$

$\Box \Diamond P$

$=$

$(\sigma, 1) \models \Box \Diamond P$

$=$

$\forall j: 1 .. \bullet (\sigma, j) \models \Diamond P$

$=$

$\forall j: 1 .. \bullet (\exists k: j .. \bullet (\sigma, k) \models P)$

Leads To

- $\Box (P \Rightarrow \Diamond Q)$
- If P is true then eventually Q will be true
- This is sometimes read: P leads to Q
- Examples:
  > P might be "makes a request" and Q might be "receives a reply"

  > P might be "enters a critical region" and Q might be "exits that region"

Qn: how is this different from $P \Rightarrow \Diamond Q$?

Qn: how is this different from $\Diamond P \Rightarrow \Diamond Q$?

© Garlan 2016

Box and Diamond are "Dual"

- Temporal logic has a number of important algebraic properties
- Two of the most useful are

$\Box P \equiv \sim \Diamond \sim P$

and

$\Diamond P \equiv \sim \Box \sim P$

Why are these true?

- Sometimes you see G for $\Box$

  F for $\Diamond$

Summary of Some Common Forms

- $P \Rightarrow \Diamond Q$
  - if initially P then eventually Q
- $\Box (P \Rightarrow \Diamond Q)$
  - every P-position is followed by a Q-position
- $\Box \Diamond Q$
  - Q happens infinitely often
- $\Diamond \Box Q$
  - eventually permanently Q
  - alt: the sequence contains only finitely-many ~Q-positions
- $\Box (P \Rightarrow \Box P)$
  - once P, always P

Until

- We can express “P is true until Q holds”

  $(\sigma, i) \models P \,\mathbf{U}\, Q \iff$

  $\exists\, k: i.. \bullet\ (\sigma, k) \models Q \wedge (\forall\, j: i\ ..\ k\text{-}1 \bullet (\sigma, j) \models P)$

- Example

  $\Box$ (request_made $\Rightarrow$
  (request_registered U request_answered))

- Until requires that Q becomes true eventually

- We also have weak version U_w, defined as
  $P\ U_w\ Q \iff \Box P \vee (P\ U\ Q)$
  sometimes written **P W Q**

What's Next?

- **Circle: the Next state operator**

  $(\sigma, i) \models \bigcirc P \Leftrightarrow (\sigma, i+1) \models P$

- **That is, $\bigcirc$ P is true at position i $\Leftrightarrow$ P is true at position i+1**
- **As usual, $\bigcirc$ P, by itself, means $(\sigma, 1) \models \bigcirc P$**
- **Sometimes $\bigcirc$ is represented by the letter "X"**
- **Some argue people that $\bigcirc$ is not a good operator for specification**
  - > **specifications should be insensitive to orderings, particularly when concurrency is represented by interleaving**

Other Temporal Logics

- We have presented a particular temporal logic called “Linear Temporal Logic” (LTL)
- But there are other varieties
- One that is often used with model checkers is “Computation Tree Logic” (CTL)
  - It views the behavior of a state machine as a tree in which each node is a state and branches represent possible next states
  - One can then quantify over paths in the tree
  - Neither LTL nor CTL is strictly more expressive

© Garlan 2016