

CS6133 Fall 2011 Software Specification and Verification

Lecture 3 Temporal Logic

Temporal Logic: Overview

- ◆ Temporal Logic was designed for expressing the temporal ordering of events and states within a logical framework
- ◆ State is an assignment of values to the model's variables. Intuitively, the system state is a snapshot of the system's execution, in which every variable has some value
- ◆ Event is a trigger (e.g., signal) that can cause a system to change its state and won't persist

Trace

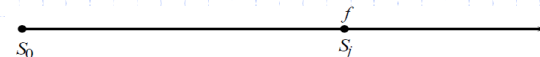
- ◆ In temporal logic, the notion of exact time is abstracted away
- ◆ In temporal logic, we keep track of changes to variable values and the order in which they occur
- ◆ A trace σ is an infinite sequence of states that represents a particular execution of the system starts from an initial state s_0 , which is determined by the initial values of all the variables

$$\sigma = s_0, s_1, s_2, \dots$$

Linear Temporal Logic Formula

- ◆ In linear temporal logic (LTL), a formula f is evaluated with respect to a trace σ and a particular state s_j in that trace

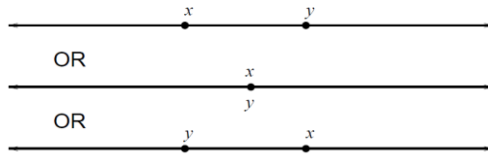
$(\sigma, j) \models f$ iff f is true in state s_j of σ



LTL Characteristics

- ◆ Time is totally ordered

$$\forall x, y : \text{Time}. (x < y \vee x = y \vee y < x)$$



CS6133

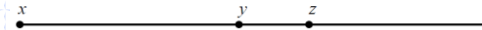
5

LTL Characteristics

- ◆ Time is bounded in the past and unbounded in the future

$$\exists x : \text{Time}. (\neg \exists z : \text{Time}. (z < x))$$

$$\forall y : \text{Time}. (\exists z : \text{Time}. (y < z))$$



CS6133

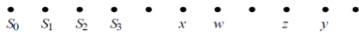
6

LTL Characteristics

- ◆ Time is discrete

$$\forall x, y : \text{Time}. \\ (x < y \rightarrow \exists w : \text{Time}. \\ (x < w \leq y \wedge \neg \exists u : \text{Time}. \\ (x < u < w))) \wedge$$

$$\forall x, y : \text{Time}. \\ (x < y \rightarrow \exists z : \text{Time}. \\ (x \leq z < y \wedge \neg \exists u : \text{Time}. \\ (z < u < y)))$$



CS6133

7

Future Temporal Operators

- ◆ Future temporal operators are shorthand notations that quantify over states

- henceforth: \square
- eventually: \diamond
- next state: \bigcirc
- until: \mathcal{U}
- unless: \mathcal{W}

CS6133

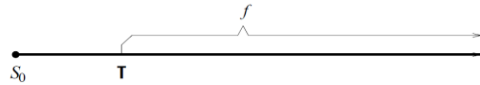
8

Henceforth

$\Box f =$

$\begin{cases} T & \text{if } f \text{ is true in the current and} \\ & \text{all future system states} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models \Box f$ iff
 $\forall i. (j \leq i \rightarrow (\sigma, i) \models f)$



CS6133

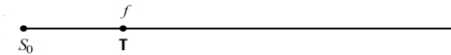
9

Eventually

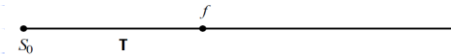
$\Diamond f =$

$\begin{cases} T & \text{if } f \text{ is true in the current or} \\ & \text{some future system state} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models \Diamond f$ iff
 $\exists i. (j \leq i \wedge (\sigma, i) \models f)$



OR



CS6133

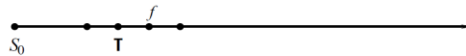
10

Next State

$\bigcirc f =$

$\begin{cases} T & \text{if } f \text{ is true in the next} \\ & \text{system state} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models \bigcirc f$ iff
 $(\sigma, j + 1) \models f$



CS6133

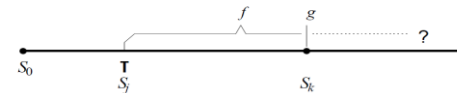
11

Until

$f \mathcal{U} g =$

$\begin{cases} T & \text{if } g \text{ is eventually true, and} \\ & f \text{ is true until then} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models f \mathcal{U} g$ iff
 $\exists k. (k \geq j \wedge ((\sigma, k) \models g) \wedge$
 $\forall i. (j \leq i < k \rightarrow (\sigma, i) \models f))$



CS6133

12

Unless

$f \mathcal{W} g =$

$$\begin{cases} T & \text{if } f \text{ holds indefinitely or} \\ & \text{until } g \text{ holds} \\ F & \text{otherwise} \end{cases}$$

$f \mathcal{W} g$ iff $f \mathcal{U} g \vee \Box f$

Unless is like Until, without the guarantee that g might happen



OR

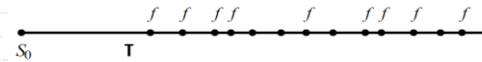


CS6133

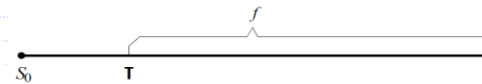
13

Examples

$\Box(\Diamond f)$ means that f happens infinitely often



$\Diamond(\Box f)$ means that eventually f is permanently true



OR



CS6133

14

LTL Properties

- ◆ Safety property can be expressed by a temporal formula of the form

$\Box p$

- ◆ Response property can be expressed by a temporal formula of the form

$\Box(p \rightarrow \Diamond \text{response})$

- ◆ Precedence (a happens before b happens)

$\neg b \mathcal{U} a \quad \neg b \mathcal{W} a$

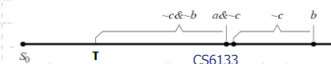
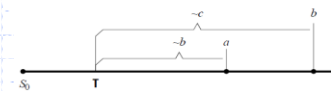
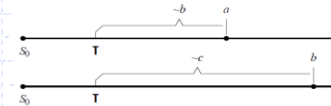
CS6133

15

LTL Properties

- ◆ Precedence Chain (a before b before c)

$\neg b \wedge \neg c \mathcal{W} (\neg c \wedge a \wedge \bigcirc(\neg c \mathcal{W} b))$



CS6133

16

LTL Properties

- ◆ P between Q and R

$$\Box((Q \wedge \neg R \wedge \Diamond R) \rightarrow (p \mathcal{U} R))$$

or

$$\Box((Q \wedge \neg R) \rightarrow (p \mathcal{W} R))$$

CS6133

17

Example: A Telephone System

- ◆ Given the predicates

onhook(user)	user is onhook
offhook(user)	user is offhook
dialing(user)	user is dialing a number
dial(user1, user2)	user1 has dialed user2 (user1 \neq user2)
busytone(user)	user hears a busytone
idletone(user)	user hears a idletone
ringtone(user)	user hears a ringtone
dialtone(user)	user hears a dialtone
connection(user1, user2)	there is a connection between users 1 and 2

CS6133

18

Examples Using Future Operators

- ◆ Formalize the following sentences in LTL
 - A user always needs to pick up the phone before dialing
 - After picking up the phone, the user eventually either goes back on hook or dials
 - Whenever a user dialed a number and heard the ring tone, a connection will only result after the other user picks up the phone
 - Immediately after the callee hangs up on a connection, the caller will hear an idle tone, then, the caller will hear a dial tone

CS6133

19

Past Temporal Operators

- ◆ Past temporal operators are shorthand notations that quantify over states
- ◆ Past temporal operators are a symmetric counterpart to each of the future temporal operators
 - Has-always-been \Box
 - Once \Diamond
 - Previous \ominus
 - Since S
 - Back-to \mathcal{B}

CS6133

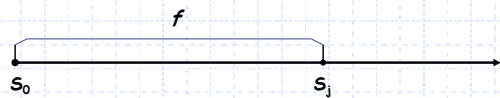
20

Has-always-been

$\Box f =$

$\begin{cases} T & \text{if } f \text{ is true in the current and} \\ & \text{all past system states} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models \Box f \text{ iff } \forall i. 0 \leq i \leq j \rightarrow (\sigma, i) \models f$



CS6133

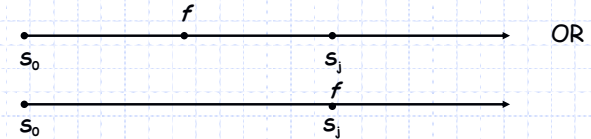
21

Once

$\Diamond f =$

$\begin{cases} T & \text{if } f \text{ is true in the current or} \\ & \text{some past system state} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models \Diamond f \text{ iff } \exists i. 0 \leq i \leq j \rightarrow (\sigma, i) \models f$



CS6133

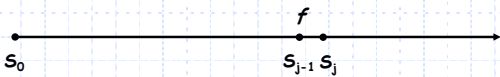
22

Previous

$\ominus f =$

$\begin{cases} T & \text{if } f \text{ is true in the previous system state} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models \ominus f \text{ iff } \exists i. i = j-1 \rightarrow (\sigma, i) \models f$



CS6133

23

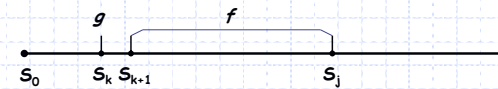
Since

$f S g =$

$\begin{cases} T & \text{if once } g \text{ was true and} \\ & f \text{ has been true since the last } g \text{ to the present} \\ F & \text{otherwise} \end{cases}$

$(\sigma, j) \models f S g \text{ iff } \exists k. 0 \leq k \leq j \wedge (\sigma, k) \models g$

$\wedge \forall i. k < i \leq j \rightarrow (\sigma, i) \models f$



CS6133

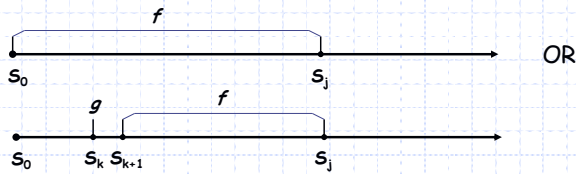
24

Back-to

$fB_g =$

$$\begin{cases} T & \text{if } f \text{ has-always-been true or} \\ & f \text{ since } g \\ F & \text{otherwise} \end{cases}$$

$fB_g \text{ iff } fSg \vee \Box f$



CS6133

25

Examples Using Past Operators

- ◆ Formalize the following sentences in LTL
 - When a caller hears the dial tone, the caller must have picked up the phone
 - When the callee hears the ring, a caller must dial the callee's number and hasn't hanged up
 - Whenever a user dialed a number and heard the ring tone, a connection is established if the other user picks up the phone

CS6133

26

Linear vs. Branching Views

- ◆ Two ways to think about the computations of reactive system
 - Linear time: LTL
 - Branching time: computation tree logic (CTL)
- ◆ A CTL formula is true/false relative to a state where as an LTL formula is true/false relative to a path

CS6133

27

CTL

- ◆ There are future temporal operators of LTL
- ◆ There are also path quantifiers to describe the branching structure of a computation tree:
 - A and E
 - A means for all computation paths
 - E means for some computation paths

CS6133

28

CTL Syntax

- ◆ If p is an atomic proposition, and f_1 and f_2 are CTL formulae, then the set of CTL formulae consists of
 1. p
 2. $\neg f_1, f_1 \wedge f_2, f_1 \vee f_2, f_1 \Rightarrow f_2$
 3. $AX f_1, EX f_1$
 4. $AG f_1, EG f_1$
 5. $AF f_1, EF f_1$
 6. $A [f_1 U f_2], E [f_1 U f_2]$
- ◆ Note that the path quantifiers and temporal operators are always paired together

CS6133

29

CTL Semantics

- ◆ $AX f$
if on all paths starting at state s , f holds in the next state
- ◆ $EX f$
if there exists a path starting at state s on which f holds at the next state.

CS6133

30

CTL Semantics

- ◆ $EF f$
if f is reachable (i.e., if there exists a path starting at state s , on which f holds in some future state).
- ◆ $AF f$
if f is inevitable (i.e., if on all paths that start at state s , f holds in some future state).

CS6133

31

CTL Semantics

- ◆ $EG f$
if there exists a path starting at state s , on which f holds globally.
- ◆ $AG f$
if f is invariant (i.e., if on all paths that start at state s , f holds globally).

CS6133

32

CTL Semantics

- ◆ $E[g \text{ U } f]$
if there exists a path starting at state s , on which g holds until f eventually holds.
- ◆ $A[g \text{ U } f]$
if on all paths that start at state s , g holds until f eventually holds.

Example of CTL Formulas

- ◆ "It is possible to get to a state where started holds, but ready does not hold."
- ◆ "For any state, if a request occurs, then it will eventually be acknowledged."
- ◆ "It is always the case that a certain process is enabled infinitely often on every computation path."

LTL vs. CTL

- ◆ In LTL, we could write: $FG p$

There is no equivalent of this formula in CTL.

LTL vs. CTL

- ◆ In CTL, we could write: $AG EF p$

There is no equivalent of this formula in LTL.