Lecture 14: Temporal Logic

Kenneth M. Anderson
Foundations of Software Engineering
CSCI 5828 - Spring Semester, 1999

Today's Lecture

- Discuss Temporal Logic in depth

Temporal Logic

- In classical logic, a predicate's truth value is static; for a given interpretation it is always true or always false
- In real-life situations, implications are causal
  - if P and P $\Rightarrow$ Q then Q
  - We need P to be true at one point, then if an event causes P $\Rightarrow$ Q to be true, then at the next point, we need P to be false and Q to be true
    - Think of it in terms of states. S1: P is true, S2: Q is true

Example

- P: "the train is approaching the gate"
- P $\Rightarrow$ Q: "if the train approaches the gate, the gate is lowered"
- Q: "the gate is lowered before the train is in the gate
- R: "the gate remains closed until the train crosses the gate"
- We cannot formalize these statements in propositional or predicate logic

Concurrent and Reactive Systems

- A reactive program is one that continuously maintains an ongoing interaction with the environment controlled by it.
  - Reactive systems may be concurrent and may have to obey strict timing constraints
  - For example, “train approaching gate” and “gate lowering” are concurrent activities

Relevant properties

- safety: “something bad will not happen”
- Example
  - “the gate will remain closed while a train crosses the gate”
- safety properties
  - partial correctness
  - mutual exclusion
  - deadlock-freedom

- liveness: “something good will eventually happen”
- Example
  - “whenever the gate is directed to raise, it will eventually do so”
- liveness properties
  - program termination
  - starvation-freedom

Notions of Time

- Underlying model for time must match the system’s requirements
- Temporal logic does not try to define time, only operators that denote change and ordering
- Types of Time
  - Discrete or Continuous
  - Linear and Branching

Discrete or Continuous

- When operations are continuously varying, a dense model of time is appropriate
  - The topology of time in this instance is typically a proper subset of real numbers
- If properties are only present at certain time instants, then a discrete model of time is chosen
    - In this case, the topology is mapped into a subset of the natural numbers
- These models can be bounded and can also be broken into distinct intervals

Linear and Branching Time Models

- for any given moment in time
  - one may postulate one future time (linear) or several possible future times (branching)
- branching
  - useful for modeling uncertainty (i.e. alternatives can be considered)
- We are going to study discrete linear temporal logic

The specification hierarchy

- Temporal logic can be used to specify requirements, design, and programs
  - Requirements
    - behavior model; time constraints between predicates
  - Design
    - state changes within objects can be specified
  - Programs
    - state changes for entire programs

Common techniques

- After an object's state is specified
  - Specify formulas for properties that hold over
    - (a) all sequences of all states
    - (b) some sequence of states
    - (c) some future state in some sequence of states
- With respect to program states
  - Temporal logic formulas can specify properties
    - that hold over (subsets of) executions of the program

Temporal Logic: Syntax

- Vocabulary
  - constants, functions, propositions, states, and predicates
- It also includes
  - constant values
    - boolean constants, natural numbers, ε (empty string or list), $\varnothing$ (empty set)
  - function symbols
    - $+, -, \cup, \cap$
  - predicate symbols
    - $>, \leq, \subset, \in$

Syntax and Semantics, continued

- = (assumed defined for all types)
- A well-formed formula of predicate logic is also a well-formed formula of temporal logic
  - Temporal Logic adds
    - a sequence of states: S_1, S_2, …, S_n
    - a function that assigns to each state, a set of predicates that are true for that state
  - Temporal Logic defines three new operators
    - ☐ (always), ✧ (eventually), and ○ (next)

Syntax and Semantics, continued

- Additional Well-Formed Formula Rules
  - If f is a well-formed formula, then so are
    - ¬(f), ☐(f), ✧(f), and ○(f)
  - If f and g are well-formed formulas, then so are
    - (f ∧ g), (f∨g), (f⇒g), and (f≡g)
- Examples
  - ☐(f ⇒ ○(g))
  - ∃q•(head(s) = q) ∧ ✧(head(s) = q + 1)
  - ☐(p) ∧ ✧(q) ⇒ ☐(p ⇒ ○(r))

Types of Temporal Logic

- Use of only propositions
  - propositional linear temporal logic
- Use of quantifiers and predicates
  - first-order linear temporal logic

Interpretation of Temporal Logic

- An atomic action causes a state change
- A state history is notated:
  - σ : S_1, S_2, …, S_n
  - It represents the behavior of an object.
    - These states can correspond to either abstract object states in a design or to concrete states in a program.
  - In order to verify whether the behavior has a property as represented by a formula f, we interpret the formula over the given state history.

Interpretation, continued

- The value of a variable (expression, predicate, …)
  - for a given state is notated
    - s[x], s[e], s[p], s[f]
- To evaluate formulas without temporal operators
  - Step 1: evaluate expressions
    - Assign values to all free variables
  - Step 2: evaluate predicates
    - For a predicate P(t1, t2, …), define s[P] = P(s[t1], s[t2], …)
    - For formulas, $s[\neg p] = \neg s[p]$, $s[p \wedge q] = s[p] \wedge s[q]$, etc.

Interpretation, continued

- To evaluate formulas without temporal operators
  - Step 3: evaluate quantified formulas
    - $s[\forall x \bullet p] = \forall x \bullet s[p]$
    - $s[\exists x \bullet p] = \exists x \bullet s[p]$
- Example
  - state s = (x=-1, y=3, z=1)
  - formula $(x+y>z) \Rightarrow (y \leq 2 * z)$
  - $s[(x+y>z) \Rightarrow (y \leq 2 * z)]$
  - $= (s[x]+s[y]>s[z]) \Rightarrow (s[y] \leq 2 * s[z])$
  - $= (-1 + 3 > 1) \Rightarrow (3 \leq 2 * 1)$
  - $=$ (true $\Rightarrow$ false) (This expression thus evaluates to false for state s)

Semantics of Temporal Formulas

- $\Box P$
  - P always hold
    - $\Box P$ holds at S_j iff P holds at all states S_k, $k \geq j$
- $\diamond P$
  - P holds sometimes
    - $\diamond P$ holds at S_j iff P holds at some state S_k, $k \geq j$
- $\bigcirc P$
  - P holds at the next instant
    - $\bigcirc P$ holds at S_j iff P holds at state S_{j+1}

Examples

- $\Box(\text{lost}(x) \Rightarrow \neg\text{instacks}(x))$
  - A lost book is not on the stacks
- $\Box(\text{inc}(x) \Rightarrow \Box\text{inc}(x))$
  - Once x is incremented, then it is incremented in every state thereafter
- $\bigcirc \Box(x = 1) \Rightarrow (\diamond \Box(y=0) \wedge \diamond(z = 1))$
  - If at the next step, x becomes permanently 1, then eventually y becomes permanently zero and z eventually becomes 1

Returning to the Train example

- Propositions
  - G1: the gate is lowered
  - G2: the gate is closed
  - G3: the gate is raised
  - G4: the gate is open
  - T1: the train is approaching
  - T2: the train is crossing the gate
  - T3: the train has crossed the gate

Train example, continued

- Assume
  - σ: S0, S1, … where
    - S0: G1 $\wedge$ T1 is true (Gate is lowered)
    - S1: G2 $\wedge$ T1 is true (Gate is closed)
    - S2, S3: G2 $\wedge$ T2 is true (Train is crossing)
    - S4: G2 $\wedge$ T3 is true (Gate is closed)
    - S5: G3 $\wedge$ T3 is true (Gate is raised)
    - S6, …: G4 $\wedge$ T3 is true (Gate is open)

Train example, continued

- We can conclude the following
- $\Diamond$(G2) is true for states 0, 1, 2, 3, 4; otherwise not
- $\bigcirc$(G2) is true for states 0, 1, 2, 3; otherwise not
- $\Box$(G4) is false for states 0-5, and true thereafter
- $\Box$(T3) is true for all states ≥ 4
- $\Box$(T3 $\Rightarrow$ $\Diamond$G4)
  - The gate will eventually open, after the train has crossed
- $\Box\Diamond$(G4)
  - There are an infinite number of states where the gate is open
- $\Diamond\Box$($\neg$T1 $\Rightarrow$ G4)
  - There exists a state where the proposition holds for all later states

Additional Temporal Operators

- Until
  - P holds continuously at least until the first occurrence of Q
- Waiting-For
  - P holds forever or until the first occurrence of Q
- Since
  - Q has happened at sometime in the past and P has continuously held ever since
- Once
  - P has happened at sometime in the past
- I don’t have the correct font for the symbols of these operations, instead I will use their name in place of their symbol in formulas

Frequently used Formulas

- $f \Rightarrow \lozenge g$
  - If f at one state, then eventually g
- $\square(f \Rightarrow \lozenge g)$
  - Holds for all states
- $\square(f \Rightarrow \bigcirc g)$
  - If f is true at state n, then g is true in state n+1, f is true at state 0
- $\square(f \Rightarrow f \text{ Until } g)$
  - Where f is true, f continues to remain true until g becomes true
- $\square(f \Rightarrow \text{Once } g)$
  - In every state where f is true, it was preceded by a state where g is true

Examples

- We will now work through some examples on paper
  - Library Books
  - Communication Channels
  - A thread-safe queue