Temporal Logic

- Classical logic:
  - Good for describing static conditions

- Temporal logic:
  - Adds temporal operators
  - Describe how static conditions change over time

- Two main ways to represent temporal logic:
  - Linear time: describes single possible time line
    - LTL (Linear Temporal Logic): Spin
  - Branching time: describes all possible time lines
    - CTL (Computation Tree Logic): SMV

What can you do with TL?

- Use Automaton to describe system behavior
  - Describes actions of system
  - In terms of state sequences

- Use temporal logic to describe property of state sequences

- Use model checker to verify TL property
  - Checks TL property against automaton
  - Exhaustively checks the automaton
  - Automatic checking

Basic Idea of Temporal Logic

- Truth changes over time
- Must say when things are true,
  - Not just what is true

- Most model checking tools do not allow quantification
- Use propositional TLs (PTLs)
Semantics of Temporal Logic

- 4 basic operators:
  - \( \square \): always
  - \( \Diamond \): sometime or eventually
  - \( \Delta \): next time or next step
  - \( U \): until

- Conceptual representation:

Let \( S \) be a sequence of states
- \( S[0] \): first state; \( S[j] \): jth state
- \( S[j..] \): sequence starting from jth state
- \( S[j..k] \): sequence from j to k

More Formal Semantics

- **Classical Logic:**
  - \( S \models f \): sequence \( S \) satisfies formula \( f \)
  - \( S \models \text{true} \): for any \( S \)
  - \( S \models (f \text{ and } g) \): \( S \models f \) and \( S \models g \)

- **Temporal Logic:**
  - \( S \models \square f \): if for any \( j \), \( S[j..] \models f \)
  - \( S \models \Diamond f \): if for some \( j \), \( S[j..] \models f \)
  - \( S \models \diamond f \): if \( S[1..] \models f \)
  - \( S \models (\text{if for some } k, S[k..] \models g, \text{ and for any } j < k, S[j..] \models f) \)

Example

- Automaton/machine produces state sequence:
  - \( abcabcabcabc \ldots \)

- Sequence satisfies property:
  - \( \square (a \Rightarrow b) \) and
  - It's always the case that \( a \) implies that the next step will be \( b \)
  - \( \square (a \Rightarrow \Diamond b) \)
  - It's always the case that \( a \) implies that the next step will eventually be an \( a \).
More Semantics of PTL

- Next $f$:
  $S[j] \models \text{Next } f$ iff $S[j+1] \models f$

- Always $\Box$:
  $S[j] \models \text{Always } f$ iff ($\exists k : k \geq j \land S[k] \models f$)

- Sometime $\Diamond$:
  $S[j] \models \text{Sometime } f$ iff ($\exists k : k \geq j \land S[k] \models f$)

- $f \text{ U } g$:
  $S[j] \models f \text{ U } g$ if
  ($\exists k : k \geq j \land S[k] \models g$)
  ($\exists l : j \leq l < k \land S[l] \models f$)

Sample Specifications

- Mutual Exclusion:
  $\square (\neg \text{InCsA} \land \neg \text{InCsB})$

- Response:
  $\square (\text{wantsInA} \land \neg \text{InCsA})$

- See CSE814 handouts for Promela codification.

Computation Tree Logic

- Most distributed, reactive systems are **nondeterministic**
- Cannot be represented by sequence of possible states or transitions
- Has a **tree** of possible computations
- Can use CTL to represent these cases

Computation Tree Logic
Temporal Logic-Overview

### CTL Syntax

- **Basic logic AND**
- **Temporal expressions:**
  - Temporal operators are defined in pairs
  - **Path part:**
    - $A$: means "all paths" (inevitably)
    - $E$: means "on some path" (possibly)
  - **Property part:**
    - $F$: same as $\Diamond$ % For some
    - $G$: same as $\square$ % Globally holds
    - $X$: same as $\nu$ % next
    - $U$: same as $\nu$
- **Sample expressions:**
  - $AGp$: On all paths, property $p$ always holds
  - $EGp$: On some paths, property $p$ always holds

### Specification Patterns

- **Safety:** bad thing never happens:
  - $AG (? bad-thing)$
- **Liveness:** good thing eventually happens
  - $AF good-thing$
- **Bad thing could happen:**
  - $EF bad-thing$

### System satisfying EFp

![Diagram of system satisfying EFp]
Sample Specification Patterns

- Possible to get to started state, but ready does not hold
  - EF(started) & !ready

- If a request occurs, then it will eventually be acknowledged
  - AG(requested) & AF(acknowledged)

- Process is enabled infinitely often on every path
  - AG(AF enabled)

- Whatever happens, a certain process will eventually be permanently deadlocked
  - AF(AG(deadlocked))

- From any state, it is possible to get to a restart state
  - AG(AF restart)

Example Elevator

- An elevator at the 2nd floor traveling upward towards 5th floor destination does not change its direction

  - AG(floor = 2 U direction = up)

  - ButtonPressed5

  - A(direction = up U floor)

Tool Support

- Model checkers: check that system satisfies property
- Reachability analysis: describe paths of behavior of system
- Each tool uses different algorithms for optimization purposes
  - SPIN (Holzmann et al)
  - SMV (Clarke et al)
  - Nitpick (D. Jackson)
  - COSPAN (Kurshan, mostly for HW)
  - Verisoft
  - FDR
  - Etc.