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:
  - □: always
  - ◯: sometime or eventually
  - Q: 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, k] sequence starting from jth state
- S[j, k] sequence from j to k

More Formal Semantics

- Classical Logic:
  - S ⊨ f: sequence S satisfies formula f
  - S ⊨ true: for any S
  - S ⊨ (f and g): S ⊨ f and S ⊨ g

- Temporal Logic:
  - S ⊨ □f: if for any j, S[j, k] ⊨ f
  - S ⊨ ◯f: if for some j, S[j, k] ⊨ f
  - S ⊨ Qf: if S[1, k] ⊨ f
  - S ⊨ f U g: if for some k, S[k, k] ⊨ g, and for any j < k, S[j, k] ⊨ f

Example

- Automation machine produces state sequence:
  - abcabcabc...

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

- **Next**: $S[j] \models O f \iff S[j+1] \models f$
- **Always**: $S[j] \models \Box f \iff (\forall k: k \geq j \rightarrow S[k] \models f)$
- **Sometime**: $S[j] \models \Diamond f \iff (\exists k: k \geq j \land S[k] \models f)$

Sample Specifications

- **Mutual Exclusion**: $\Box (\neg inCs A \lor \neg inCs B)$
- **Response**: $\Box (\text{wantsInA} \Rightarrow \Diamond inCs A)$
- 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
**CTL Syntax**

- Basic logic AND
- Temporal expressions:
  - Temporal operators are defined in pairs
  - Path part:
    - $A$: means "all paths" (necessarily)
    - $E$: means "on some path" (possibly)
  - Property part:
    - $F$: same as $\Diamond$ % For some
    - $G$: same as $\Box$ % Globally holds
    - $X$: same as $\lnot $ % Next
    - $U$: same as $U$ % Until
- 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:
  - $AG \neg \text{bad-thing}$
- Liveness: good thing eventually happens
  - $AF \text{good-thing}$
- Bad thing could happen:
  - $EF \text{bad-thing}$

**System satisfying EFp**

![System diagram]

---

*FMSDOverview*
System satisfying $\text{EG}_p$

System satisfying $\text{AG}_p$

System satisfying $\text{AF}_p$
Sample Specification Patterns

- Possible to get to \texttt{started} state, but \texttt{ready} does not hold
  - $\text{EF}([\texttt{started}] \wedge \neg \text{\texttt{ready}})$

- If a request occurs, then it will eventually be acknowledged
  - $\text{AG}([\texttt{request}] \rightarrow \text{\texttt{acknowledged}})$

- Process is enabled infinitely often on every path
  - $\text{AG}([\texttt{AF enabled}]$)

- Whatever happens, a certain process will eventually be permanently deadlocked
  - $\text{AF}([\texttt{deadlocked}]$)

- From any state, it is possible to get to a \texttt{restart} state
  - $\text{AG}([\texttt{AF restart}]$)

Example Elevator

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

- $\text{AG}([\texttt{floor} = 2 \wedge \texttt{direction} = \texttt{up} \wedge$
  *ButtonPressed5 $\Rightarrow$
  *$\text{A}[\texttt{direction} = \texttt{up} \cup \texttt{floor} = 5]$
  *
  *$\Rightarrow U$

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 (Kurehan, mostly for HW)
  - Versoft
  - FDR
  - Etc.