Linear Temporal Logic (LTL) Tutorial

Introduction

Linear Temporal Logic (LTL) is a way to describe how system properties change over time along a linear path of states. It's mainly used in hardware and software verification when you want to be precise about safety ("nothing bad ever happens") or liveness ("something good eventually happens").

Basic Building Blocks

- ➔ Atomic Propositions (p, q, r, …)
  - ◆ These variables will consist of your boolean formulas. For example, p could mean "the door is open," and q could mean "the light is on." It will hold a binary value.

- ➔ Boolean Connectors
  - ◆ ¬p means "not p."
  - ◆ p $\wedge$ q means "p and q."
  - ◆ p $\vee$ q means "p or q."
  - ◆ p → q means "if p then q."

- ➔ Temporal Operators
  - ◆ X p (Next): "p holds in the next state." If you're at time t, then at time t + 1, p must be true
  - ◆ G p (Always): "p holds in all future states." If p ever fails, G p is false.
  - ◆ F p (Eventually): "p will hold at some future state." If p never becomes true, F p is false.
  - ◆ p U q (Until): "p remains true until q becomes true, and q must eventually become true." Once q happens, p doesn't need to hold afterward.

You can combine these however you like. For example, X G p means "one step from now, p holds forever."

Reading Common LTL Patterns

- G F p: "At every moment, p will eventually be true."
- F G p: "There is a point after which p is always true." Basically, "eventually, p stays true forever."
- G (p → F q): "Every time p happens, q will happen at some point afterward."
- F (p → X X q): "At least once, p occurs and exactly two steps later q holds."

How to Write LTL Specifications

- ➔ Identify Your Atomic Propositions
  - ◆ List the key Boolean signals in your system (e.g., request, grant, error, ready).
- ➔ Decide on Safety vs. Liveness
  - ◆ Safety ("nothing bad ever happens"): use G and ¬.
  - ◆ Liveness ("something good eventually happens"): use F.
- ➔ Combine Them
  - ◆ If you want "if request then eventually grant," write G (request → F grant)