Linear Temporal Logic

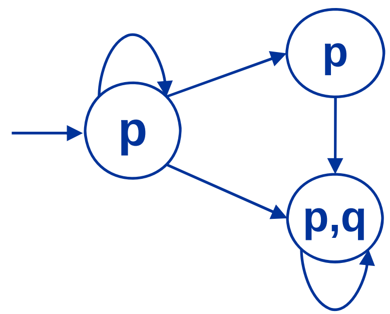
Moonzoo Kim
School of Computing
KAIST

Motivation for verification

- There is a great advantage in being able to verify the correctness of computer systems
  - This is most obvious in the case of safety-critical systems
    - ex. Cars, avionics, medical devices
  - Also applies to mass-produced embedded devices
    - ex. handphone, USB memory, MP3 players, etc
- Formal verification can be thought of as comprising three parts
  1. a system description language
  2. a requirement specification language
  3. a verification method to establish whether the description of a system satisfies the requirement specification.

Model checking

- Model checking
  - In a model-based approach, the system is represented by a model $\mathcal{M}$. The specification is again represented by a formula ϕ.
    - The verification consists of computing whether $\mathcal{M}$ satisfies ϕ $\mathcal{M} \vDash \phi$
      - Caution: $\mathcal{M} \vDash \phi$ represents satisfaction, not semantic entailment
- In model checking,
  - The model $\mathcal{M}$ is a transition systems and
  - the property ϕ is a formula in temporal logic
    - ex. □ p, □ q, ◇ q, □ ◇ q



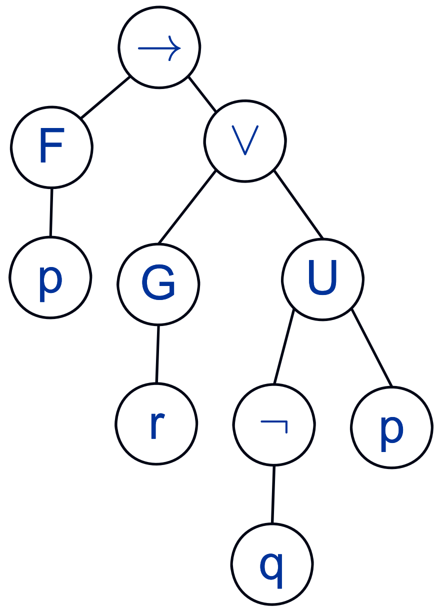
Linear time temporal logic (LTL)

- LTL models time as a sequence of states, extending infinitely into the future
  - sometimes a sequence of states is called a computation path or an execution path, or simply a path
- Def 3.1 LTL has the following syntax
  - ϕ ::= T | $\bot$ | p | $\neg\, \phi$ | $\phi \wedge \phi$ | $\phi \vee \phi$ | $\phi \rightarrow \phi$

    | X ϕ | F ϕ | G ϕ | ϕ U ϕ | ϕ W ϕ | ϕ R ϕ

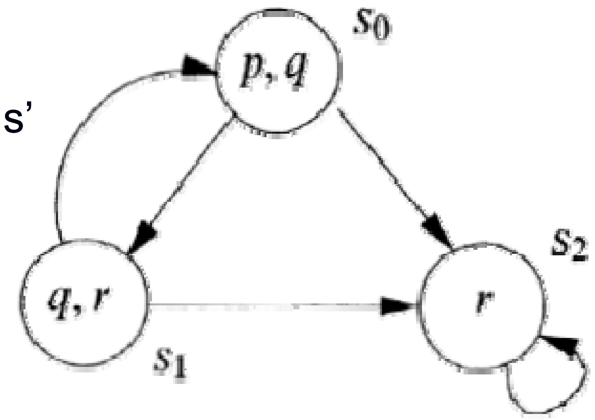
    where p is any propositional atom from some set Atoms
  - Operator precedence
    - the unary connectives bind most tightly. Next in the order come U, R, W, $\wedge$, $\vee$, and $\rightarrow$

F p $\rightarrow$ G r $\vee$ $\neg$ q U p



Semantics of LTL (1/3)

- Def 3.4 A transition system (called model) $\mathcal{M}$ = (S, $\rightarrow$, L)
  - a set of states S
  - a transition relation $\rightarrow$ (a binary relation on S)
    - such that every s $\in$ S has some s' $\in$ S with s $\rightarrow$ s'
  - a labeling function L: S $\rightarrow$ $\mathcal{P}$ (Atoms)



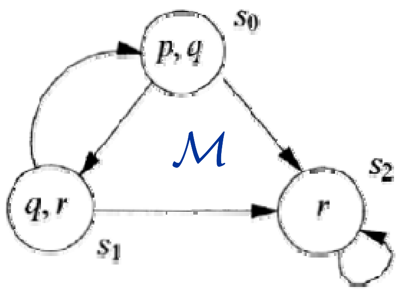
- Example
  - $S=\{s_0,s_1,s_2\}$
  - $\rightarrow=\{(s_0,s_1),(s_1,s_0),(s_1,s_2),(s_0,s_2),(s_2,s_2)\}$
  - $L=\{(s_0,\{p,q\}),(s_1,\{q,r\}), (s_2,\{r\})\}$
- Def. 3.5 A path in a model $\mathcal{M}$ = (S, $\rightarrow$, L) is an infinite sequence of states $s_{i_1}, s_{i_2}, s_{i_3}, \ldots$ in S s.t. for each $j \geq 1$, $s_{i_j} \rightarrow s_{i_{j+1}}$. We write the path as $s_{i_1} \rightarrow s_{i_2} \rightarrow \ldots$
  - From now on if there is no confusion, we drop the subscript index i for the sake of simple description
- We write π^i for the suffix of a path starting at s_i.
  - ex. π^3 is $s_3 \rightarrow s_4 \rightarrow \ldots$

Semantics of LTL (2/3)

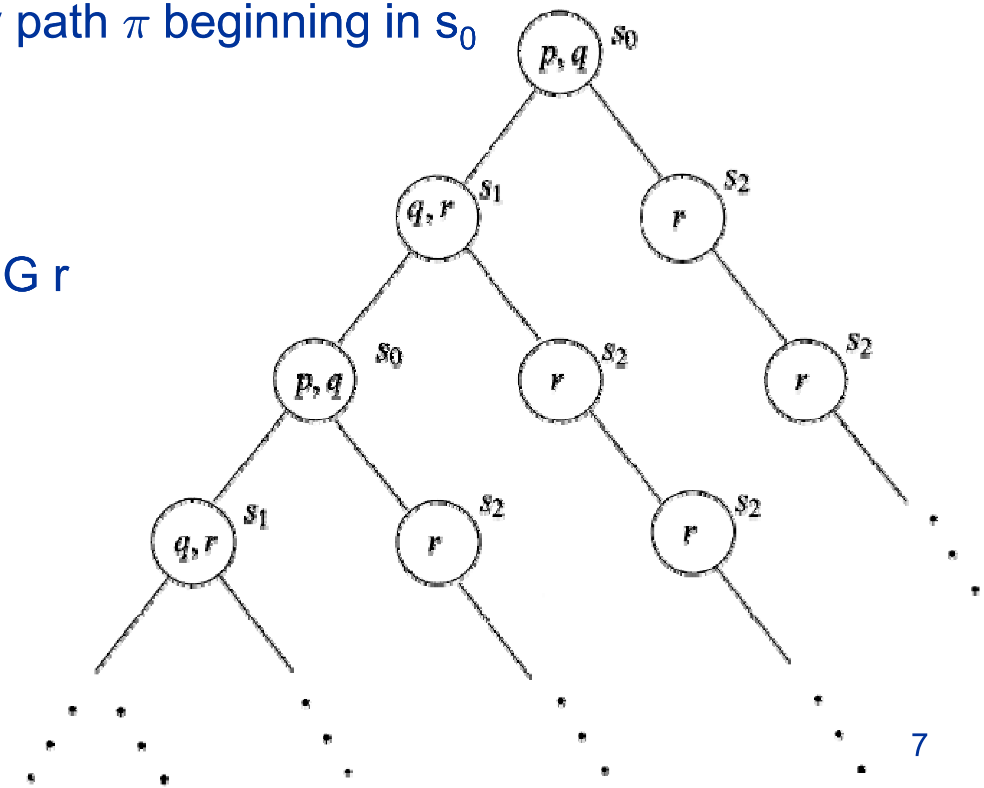
- Def 3.6 Let $\mathcal{M}$ = (S, $\rightarrow$, L) be a model and π = $s_1 \rightarrow \ldots$ be a path in $\mathcal{M}$. Whether π satisfies an LTL formula is defined by the satisfaction relation $\vDash$ as follows:
  - Basics: $\pi \vDash \top$, $\pi \nvDash \bot$, $\pi \vDash$p iff p $\in$ L(s_1) , $\pi \vDash \neg\phi$ iff $\pi \nvDash \phi$
  - Boolean operators: $\pi \vDash$ p $\wedge$ q iff $\pi \vDash$ p and $\pi \vDash$ q
    - similar for other boolean binary operators
  - $\pi \vDash$ X ϕ iff $\pi^2 \vDash \phi$ (next $\circ$)
  - $\pi \vDash$ G ϕ iff for all i $\geq$ 1, $\pi^i \vDash \phi$ (always $\Box$)
  - $\pi \vDash$ F ϕ iff there is some i $\geq$ 1, $\pi^i \vDash \phi$ (eventually $\Diamond$)
  - $\pi \vDash \phi$ U ψ iff there is some i $\geq$ 1s.t. $\pi^i \vDash \psi$ and for all j=1,…,i-1 we have $\pi^j \vDash \phi$ (strong until)
  - $\pi \vDash \phi$ W ψ iff either (weak until)
    - either there is some i $\geq$ 1 s.t. $\pi^i \vDash \psi$ and for all j=1,…,i-1 we have $\pi^j \vDash \phi$
    - or for all k $\geq$ 1 we have $\pi^k \vDash \phi$
  - $\pi \vDash \phi$ R ψ iff either (release)
    - either there is some i $\geq$ 1 s.t. $\pi^i \vDash \phi$ and for all j=1,…,i we have $\pi^j \vDash \psi$
    - or for all k $\geq$ 1 we have $\pi^k \vDash \psi$

Semantics of LTL (3/3)

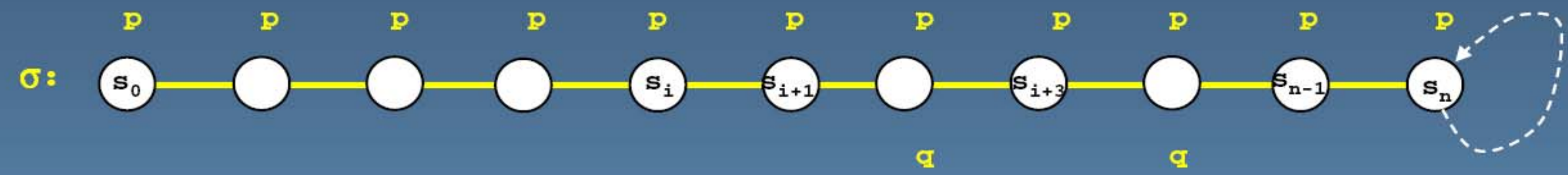
- Def 3.8 Suppose $\mathcal{M}$ = (S, $\rightarrow$, L) is a model, s $\in$ S, and ϕ an LTL formula. We write $\mathcal{M}$,s $\vDash \phi$ if for every execution path π of $\mathcal{M}$ starting at s, we have $\pi \vDash \phi$
  - If $\mathcal{M}$ is clear from the context, we write s $\vDash \phi$



- Example
  - $s_0 \vDash p \wedge q$ since $\pi \vDash p \wedge q$ for every path π beginning in s_0
  - $s_0 \vDash \neg r$, $s_0 \vDash \top$
  - $s_0 \vDash X\ r$, $s_0 \nvDash X\ (q \wedge r)$
  - $s_0 \vDash G\ \neg(p \wedge r)$, $s_2 \vDash G\ r$
  - For any s of $\mathcal{M}$, $s \vDash F(\neg q \wedge r) \rightarrow F\ G\ r$
    - Note that s_2 satisfies $\neg q \wedge r$
  - $s_0 \nvDash G\ F\ p$
    - $s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \ldots \vDash G\ F\ p$
    - $s_0 \rightarrow s_2 \rightarrow s_2 \rightarrow s_2 \ldots \nvDash G\ F\ p$
  - $s_0 \vDash G\ F\ p \rightarrow G\ F\ r$
  - $s_0 \nvDash G\ F\ r \rightarrow G\ F\ p$



examples



[]p is satified at all locations in σ

<>p is satisfied at all locations in σ

[]<>p is satisfied at all locations in σ

<>q is satisfied at all locations except s_{n-1} and s_n

Xq is satisfied at s_{i+1} and at s_{i+3}

pUq (strong until) is satisfied at all locations except s_{n-1} and s_n

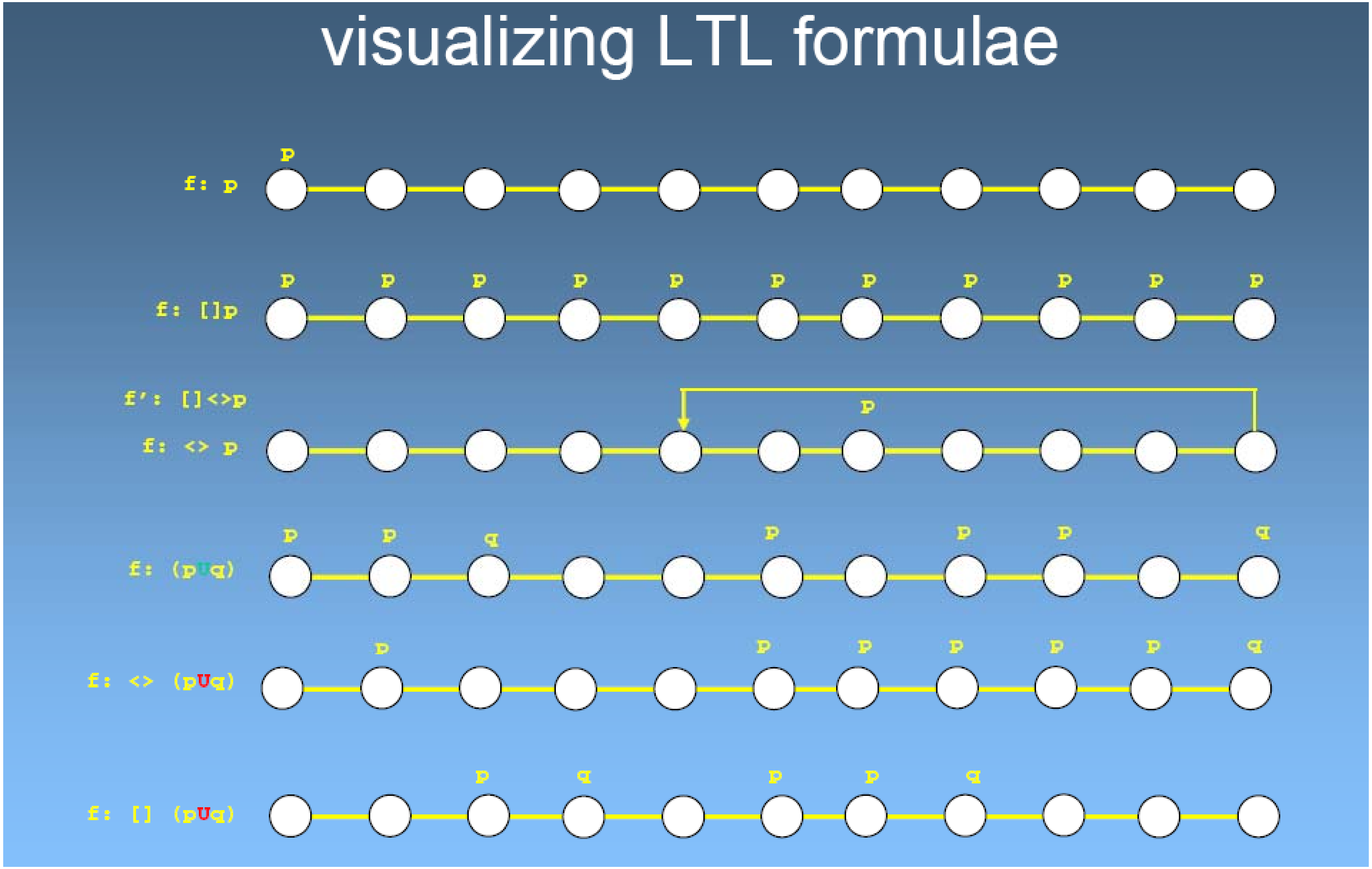
<>(pUq) (strong until) is satisfied at all locations except s_{n-1} and s_n

<>(pUq) (weak until) is satisfied at all locations

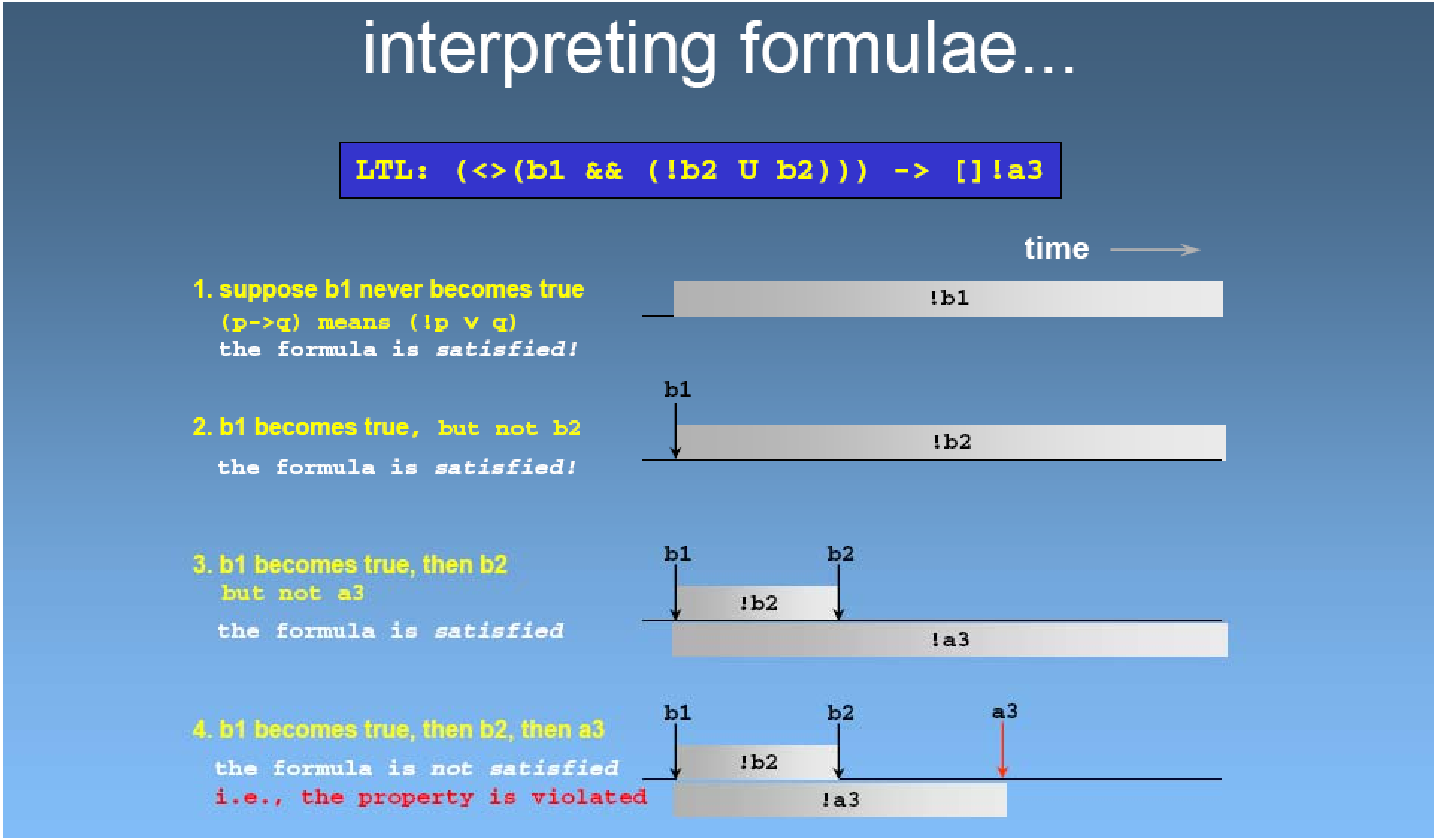
[]<>(pUq) (weak until) is satisfied at all locations

in model checking we are typically only interested in whether a temporal logic formula is satisfied for all runs of the system, starting in the initial system state (that is: at s_0)

slide quoted from Caltech 101b.2 "Logic Model Checking" by Dr.G.Holzmann



slide quoted from Caltech 101b.2 "Logic Model Checking" by Dr.G.Holzmann



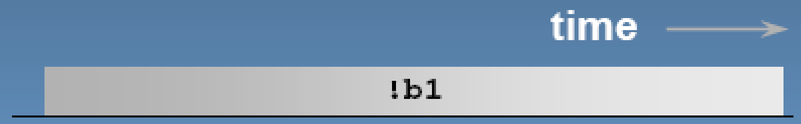
slide quoted from Caltech 101b.2 "Logic Model Checking" by Dr.G.Holzmann

another example

`LTL: (<>b1) -> (<>b2)`

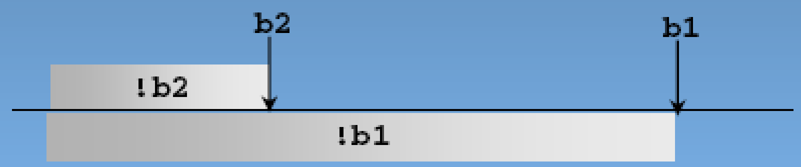
1. b1 never becomes true

   `formula satisfied`



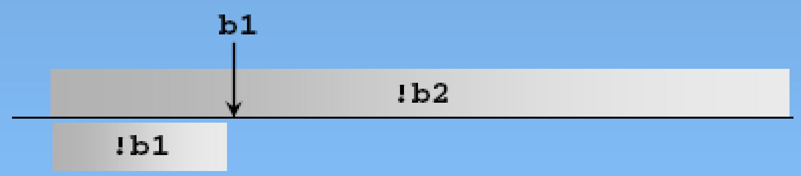
2. b1 and b2 both become true

   `formula satisfied`



3. b1 becomes true but not b2

   formula *not* satisfied
   the property is *violated*



slide quoted from Caltech 101b.2 “Logic Model Checking” by Dr.G.Holzmann

Equivalences between LTL formulas

- Def 3.9 $\phi \equiv \psi$ if for all models $\mathcal{M}$ and all paths π in $\mathcal{M}$: $\pi \vDash \phi$ iff $\pi \vDash \psi$
- $\neg G\ \phi \equiv F\ \neg\phi$, $\neg F\ \phi \equiv G\ \neg\phi$, $\neg X\ \phi \equiv X\ \neg\phi$
- $\neg\ (\phi\ U\ \psi) \equiv \neg\phi\ R\ \neg\psi$, $\neg(\phi\ R\ \psi) \equiv \neg\phi\ U\ \neg\psi$
- $F\ (\phi \vee \psi) \equiv F\ \phi \vee F\ \psi$
- $G\ (\phi \wedge \psi) \equiv G\ \phi \wedge G\ \psi$
- $F\ \phi \equiv T\ U\ \phi$, $G\ \phi \equiv \perp R\ \phi$
- $\phi\ U\ \psi \equiv \phi\ W\ \psi \wedge F\ \psi$
- $\phi\ W\ \psi \equiv \phi\ U\ \psi \vee G\ \phi$
- $\phi\ W\ \psi \equiv \psi\ R\ (\phi \vee \psi)$
- $\phi\ R\ \psi \equiv \psi\ W\ (\phi \wedge \psi)$

Practical patterns of specification

- For any state, if a request occurs, then it will eventually be acknowledge
  - G(requested $\rightarrow$ F acknowledged)
- A certain process is enabled infinitely often on every computation path
  - G F enabled
- Whatever happens, a certain process will eventually be permanently deadlocked
  - F G deadlock
- If the process is enabled infinitely often, then it runs infinitely often
  - G F enabled $\rightarrow$ G F running
- An upwards traveling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor
  - G (fllor2 $\wedge$ directionup $\wedge$ ButtonPressed5 $\rightarrow$ (directionup U floor5)
- It is impossible to get to a state where a system has started but is not ready
  - ϕ = G $\neg$(started $\wedge$ $\neg$ready)
  - What is the meaning of (intuitive) negation of ϕ ?
    - For every path, it is possible to get to such a state (started $\wedge\neg$ready).
    - There exists a such path that gets to such a state.
      - we cannot express this meaning directly
- LTL has limited expressive power
  - For example, LTL cannot express statements which assert the existence of a path
    - From any state s, there exists a path π starting from s to get to a restart state
    - The lift can remain idle on the third floor with its doors closed
  - Computation Tree Logic (CTL) has operators for quantifying over paths and can express these properties

Summary of practical patterns

G p	always p	invariance
F p	eventually p	guarantee
$p \rightarrow (F\ q)$	p implies eventually q	response
$p \rightarrow (q\ U\ r)$	p implies q until r	precedence
G F p	always, eventually p	recurrence (progress)
F G p	eventually, always p	stability (non-progress)
$F\ p \rightarrow F\ q$	eventually p implies eventually q	correlation