Temporal Logic (1/2)

Moonzoo Kim
CS Division of EECS Dept.
KAIST

moonzoo@cs.kaist.ac.kr
http://pswlab.kaist.ac.kr/courses/cs402-07

Introduction to temporal logic (1/3)

- Temporal logic is to reason about time
  - Temporal logic is applicable in many engineering fields
    - since the behavior of a target system can be described as a function of time
    - unlike mathematical expressions such as 1+1 = 2 whose behavior is static
- Consider the statement: "I am hungry."
  - Though its meaning is constant in time, the truth value of the statement can vary in time.
  - Sometimes the statement is true, and sometimes the statement is false, but the statement is never true and false simultaneously.
- In a temporal logic, statements can have a truth value which can vary in time.
  - Contrast this with an a predicate logic, which can only handle statements whose truth value is constant in time.

Introduction to temporal logic (2/3)

- Temporal logic refers modal-logic type of approach introduced around 1960 by Arthur Prior under the name of Tense Logic
  - subsequently developed further by logicians and computer scientists such as Amir Pnueli
  - Received great attention for its application on formal verification
- Example 11.1:
  - File server: If a request is made to print a file, eventually the file will be printed
  - Operating system: The system will always run. The system will never crash
- Timing properties can be expressed in predicate logic
  - ex. the file server property:
    - $\forall f\ \forall t_1\ (RequestPrint(f,t_1) \rightarrow \exists t_2\ ((t_2 \geq t_1) \wedge PrintedAt(f,t_2)))$
  - In temporal logic, new operators ($\Box$, $\Diamond$, U, etc) are introduced that enable the time variables and their relationships (e.g. $t_2 \geq t_1$) to be implicitly indicated

Introduction to temporal logic (3/3)

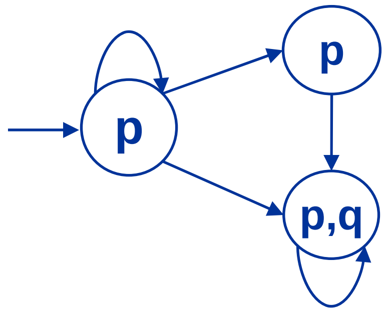
- Temporal logic has received great attention for its application in verification field since 1980
- Informal description
  - $\Box$ is the universal operator 'for any time t in the future' (always)
  - $\Diamond$ is the existential operator 'for some time t in the future' (eventually)
- The operators compose in the sense that $\Box\Diamond$ p means not just $\forall t_1 \exists t_2$ p, but $\forall t_1 \exists t_2 ((t_2 \geq t_1) \wedge p)$ and
- The file server property:
  - $\forall f\ \Box(RequestPrint(f) \rightarrow \Diamond PrintedAt(f))$
- Reasoning with a temporal formula is much easier than with its translation into the predicate calculus, because the relationships among the times are implicit
  - low-level details about dealing with time variables are hidden

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

- Proof-based vs. model-based (model checking)
  - In a proof-based approach, the system description is a set of formulas Γ and the specification is another formula ϕ.
    - The verification consists of trying to find a proof that $\Gamma \vdash \phi$
    - Requires guidance and expertise from the user
  - 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
  - The model-based approach is potentially simpler than the proof-based approach since
    - $\Gamma \vdash \phi$ means (under soundness and completeness)
      - for all models $\mathcal{M}$, if $\mathcal{M} \vDash \psi$ for all $\psi \in \Gamma$, then $\mathcal{M} \vDash \phi$
- 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



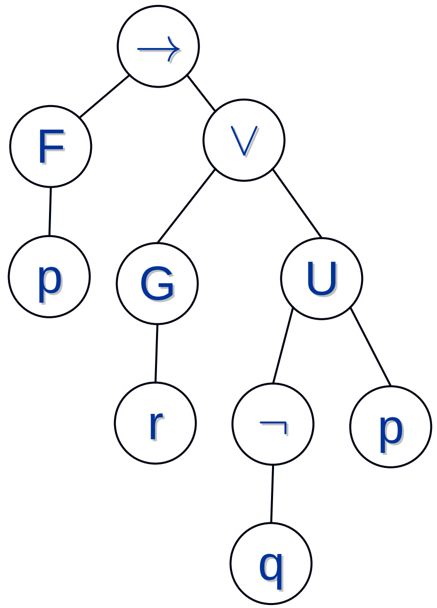
Linear time temporal logic (LTL)

- 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 | $\perp$ | 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

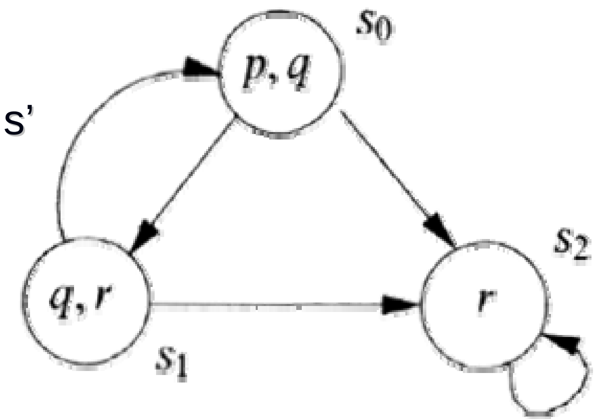
    where p is any propositional atom from some set Atoms
  - Operator precedence
    - the unary connectives bind most tightly. Next in the order come U, R, W, $\wedge$, $\vee$, and $\rightarrow$

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)

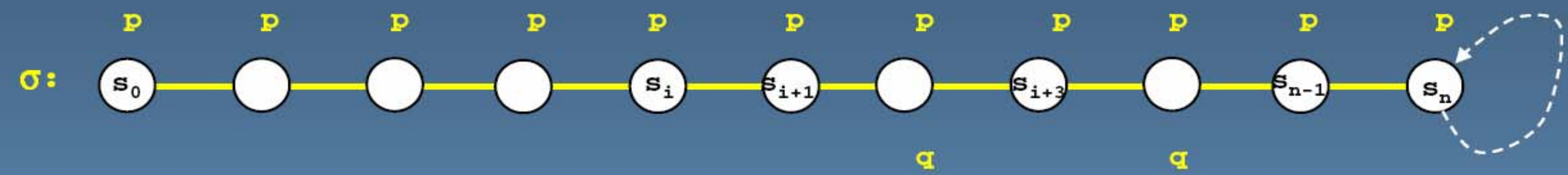


- Example
  - $S=\{s_0,s_1,s_2\}$
  - $\rightarrow=\{(s_0,s_1),(s_1,s_0),(s_1,s_2),(s_0,s_2),(s_2,s_2)\}$
  - $L=\{(s_0,\{p,q\}),(s_1,\{q,r\}), (s_2,\{r\})\}$
- Def. 3.5 A path in a model $\mathcal{M} = (S, \rightarrow, L)$ is an infinite sequence of states $s_{i_1}, s_{i_2}, s_{i_3}, \ldots$ in S s.t. 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 $\pi = s_1 \rightarrow \ldots$ 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: $\pi \vDash p \land q$ iff $\pi \vDash p$ and $\pi \vDash q$
    - similar for other boolean binary operators
  - $\pi \vDash X\ \phi$ iff $\pi^2 \vDash \phi$ (next $\circ$)
  - $\pi \vDash G\ \phi$ iff for all $i \geq 1$, $\pi^i \vDash \phi$ (always $\Box$)
  - $\pi \vDash F\ \phi$ iff there is some $i \geq 1$, $\pi^i \vDash \phi$ (eventually $\Diamond$)
  - $\pi \vDash \phi\ U\ \psi$ iff there is some $i \geq 1$s.t. $\pi^i \vDash \psi$ and for all $j=1,\ldots,i-1$ we have $\pi^j \vDash \phi$ (strong until)
  - $\pi \vDash \phi\ W\ \psi$ iff either (weak until)
    - either there is some $i \geq 1$ s.t. $\pi^i \vDash \psi$ and for all $j=1,\ldots,i-1$ we have $\pi^j \vDash \phi$
    - or for all $k \geq 1$ we have $\pi^k \vDash \phi$
  - $\pi \vDash \phi\ R\ \psi$ iff either (release)
    - either there is some $i \geq 1$ s.t. $\pi^i \vDash \phi$ and for all $j=1,\ldots,i$ we have $\pi^j \vDash \psi$
    - or for all $k \geq 1$ we have $\pi^k \vDash \psi$

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

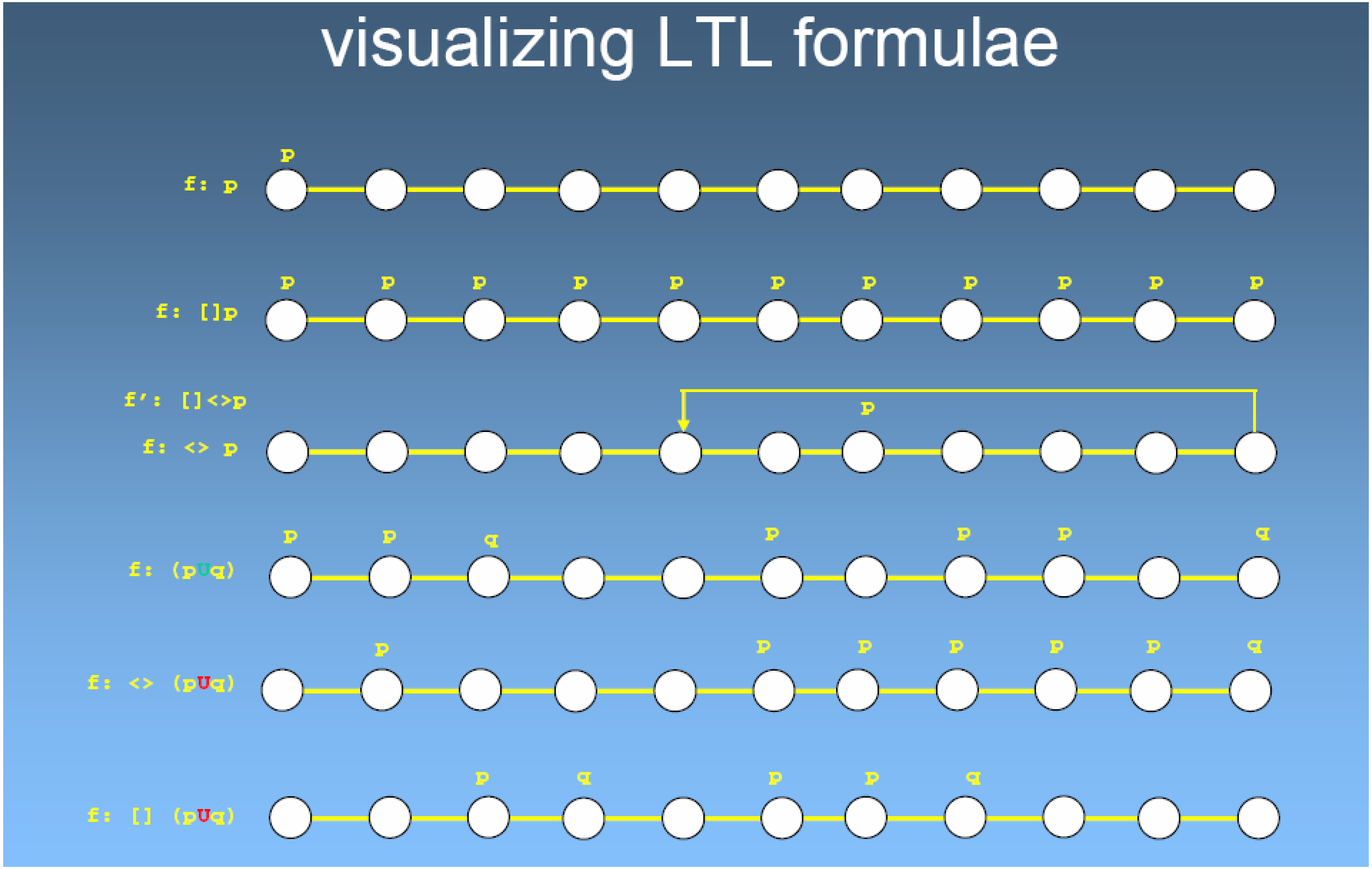
<>(pUq) (strong until) is satisfied at all locations except s_{n-1} and s_n

<>(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

interpreting formulae...

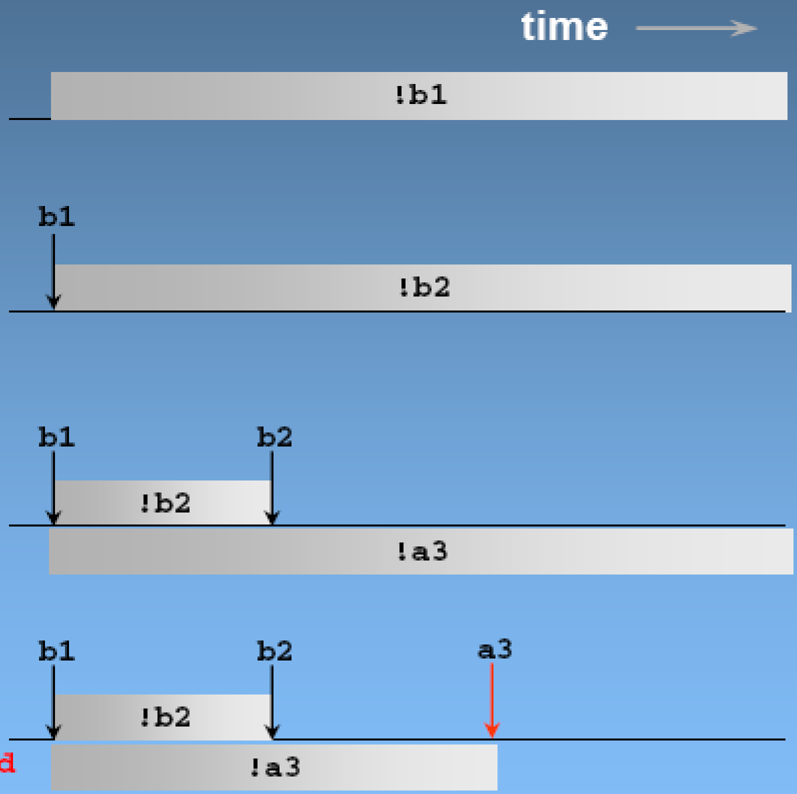
`LTL: (<>(b1 && (!b2 U b2))) -> []!a3`

1. suppose b1 never becomes true
   `(p->q) means (!p v q)`
   `the formula is` *`satisfied!`*

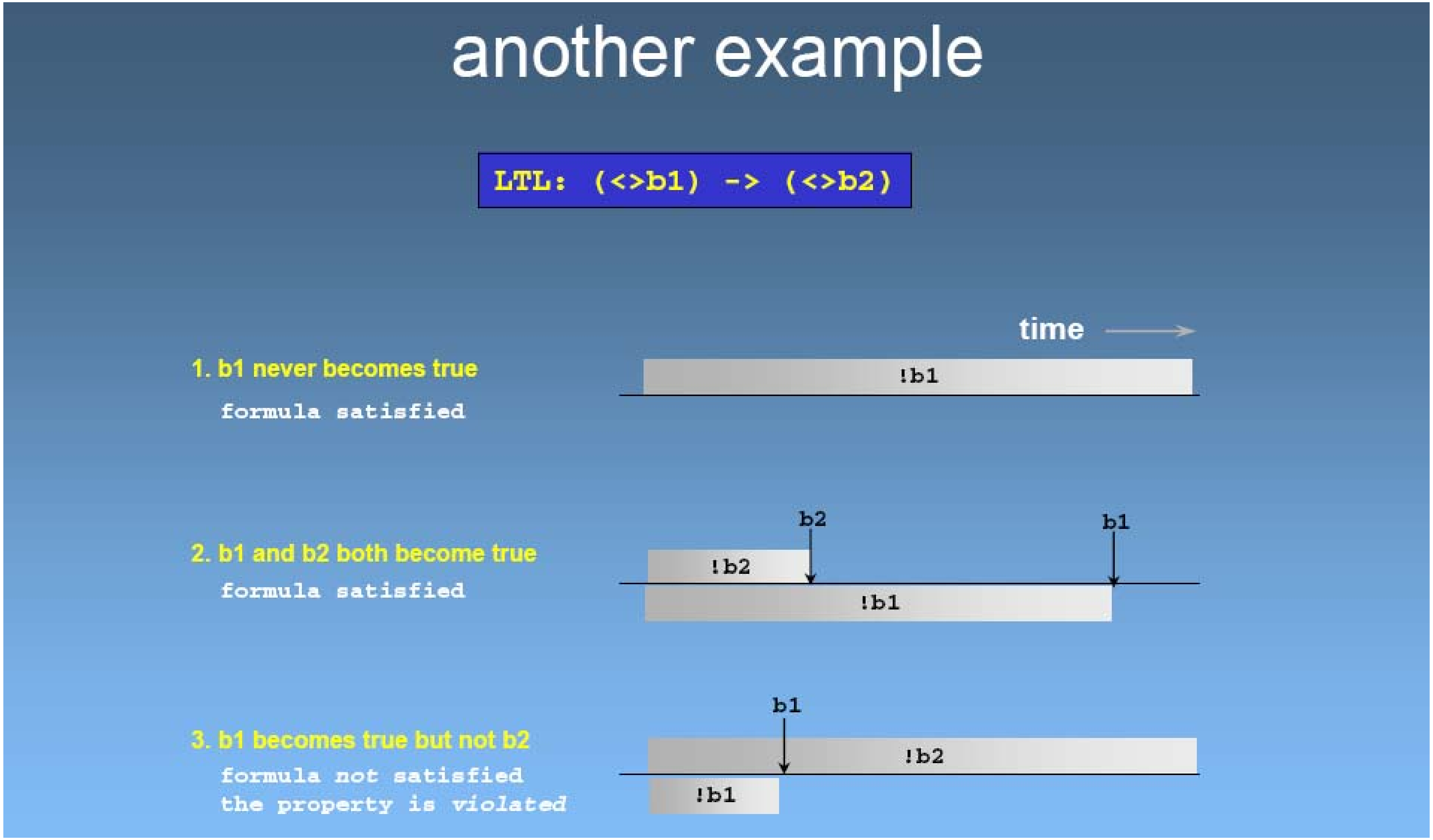
2. b1 becomes true, `but not b2`
   `the formula is` *`satisfied!`*

3. b1 becomes true, then b2
   `but not a3`
   `the formula is` *`satisfied`*

4. b1 becomes true, then b2, then a3
   `the formula is` *`not satisfied`*
   `i.e., the property is violated`



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