COL:750/7250

Foundations of Automatic Verification

Instructor: Priyanka Golia

Course Webpage

https://priyanka-golia.github.io/teaching/COL-750-COL7250/index.html

From SAT & SMT to Temporal Logic

SAT: Checks whether a propositional formula is satisfiable.

SMT: Extends SAT with richer theories (e.g., arithmetic, arrays).

But what about time?

SAT/SMT/FOL verify properties in static systems.

Many real-world systems evolve over time (e.g., software, robots, protocols).

"A robot should always eventually return to its charging station."
"A user who enters a correct password will eventually get access."
"How can we verify that a system never reaches an error state?"

Can we express this in SAT or FOL?

From SAT & SMT to Temporal Logic

Classical logic (SAT/SMT) = Static Reasoning

Temporal logic = Reasoning over time

"Temporal" here refers to "ordered events"; no explicit notion of time.

Linear Temporal Logic (LTL) —

- Assumes a single timeline (one possible sequence of events).
- Each moment in time has a well-defined successor moment.
- Each moment in time has exactly one possible future.
- Introduced by Pneuli in the 1970.

From SAT & SMT to Temporal Logic

Linear Temporal Logic (LTL) —

- Assumes a single timeline (one possible sequence of events).
- Each moment in time has a well-defined successor moment.
- Introduced by Pneuli in the 1970.

Examples:

- Eventually, the system will reach a safe state.
- If a system encounters an error, it never recovers.
- If a red light is on, it must eventually turn green.
- At most one process is in the critical section at any time.

LTL Syntax

F = True

= p (atomic proposition)

$= F_1 \wedge F_2$

$= \neg F_1$

= $\mathbf{N}\ F_1$ **N** is "Next". F_1 is True at next step. Often represented as **O**, **X** .

= $F_1\ \mathbf{U}\ F_2$ **U** is "Until". F_2 is True at "some point", and until then F_1 is True.

LTL Syntax

F = $\mathbf{N}\, F_1$ **N** is "Next". F_1 is True at next step. Often represented as **O**, **X** .

If you press the accelerator, the car will move in the next step.

$$accelerate \rightarrow \mathbf{N}\ moving$$

If you shoot the ball, the result will be known in the next step.

$$shoot \rightarrow \mathbf{N}\ (goal \lor miss)$$

F = $F_1\ \mathbf{U}\ F_2$ **U** is "Until". F_2 is True at "some point", and until then F_1 is True.

Mario will keep jumping until he lands.

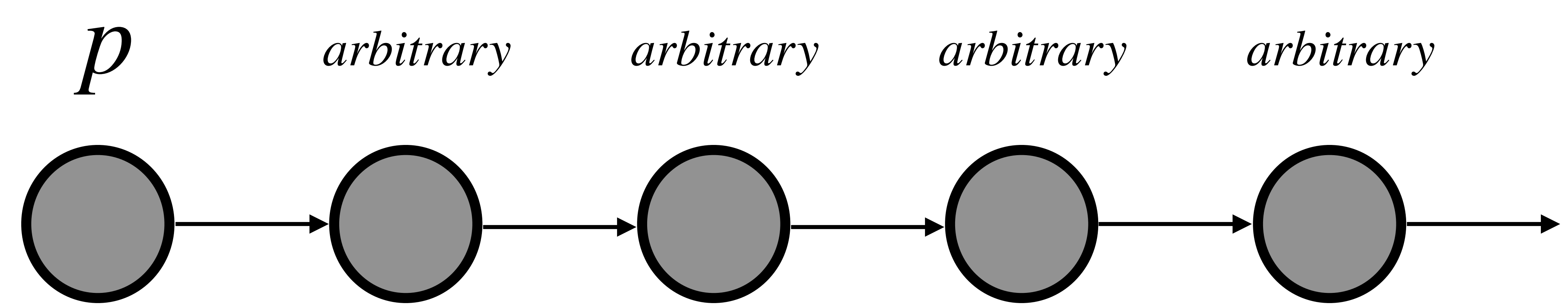
$$jumping\ \mathbf{U}\ landed$$

The emergency light will stay on until the power comes back.

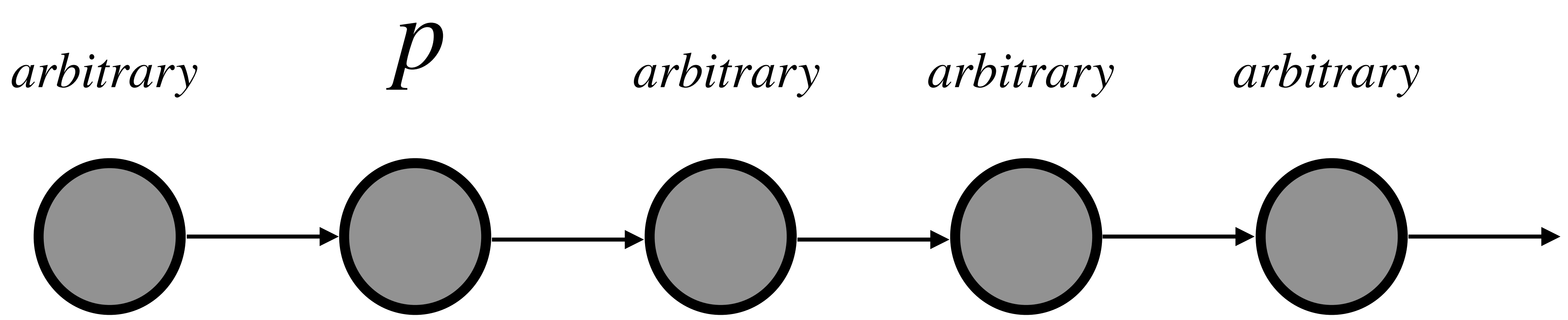
$$EmergencyLight\ \mathbf{U}\ PowerRestored$$

LTL Syntax Sequence of states (paths).

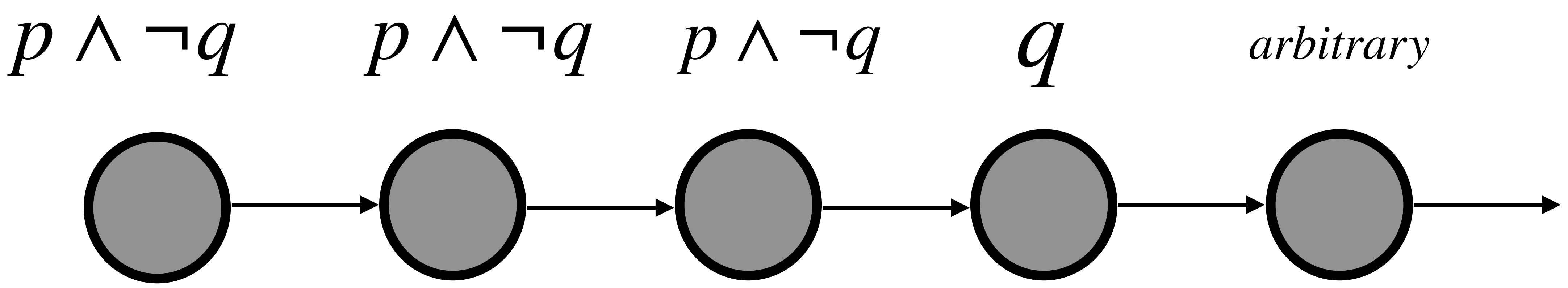
Atomic prop. P



$\mathbf{N}\ p$



$p\ \mathbf{U}\ q$



LTL Syntax

Primary temporal operators: **N U**

Additional operators

Eventually $\Diamond$ F F will become true at some point in the future.

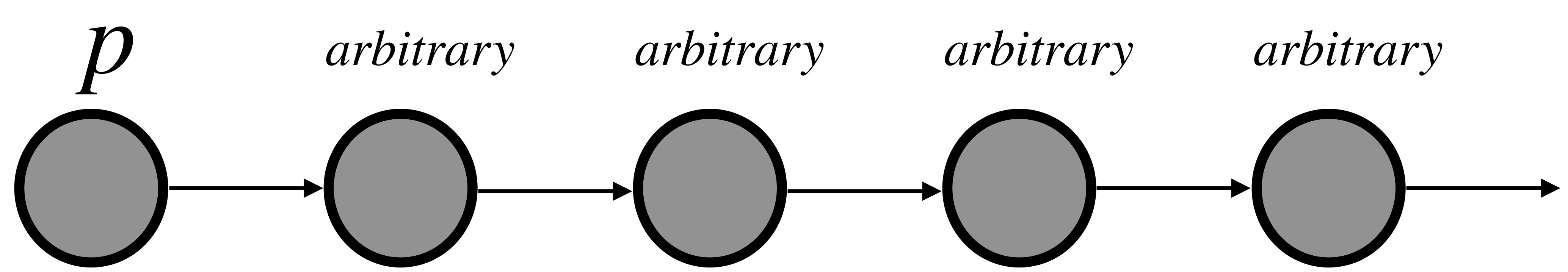
$$\Diamond F \equiv True\ \mathbf{U}\ F$$

Always (valid) $\Box F$ F is always True.

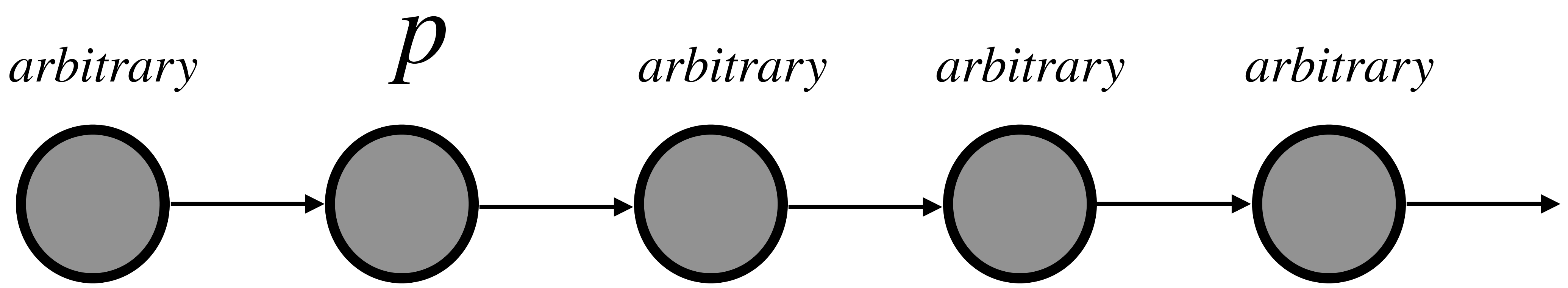
$$\Box F \equiv \neg \Diamond \neg F$$ (Never (Eventually ($\neg F$))).

LTL Syntax Sequence of states (paths).

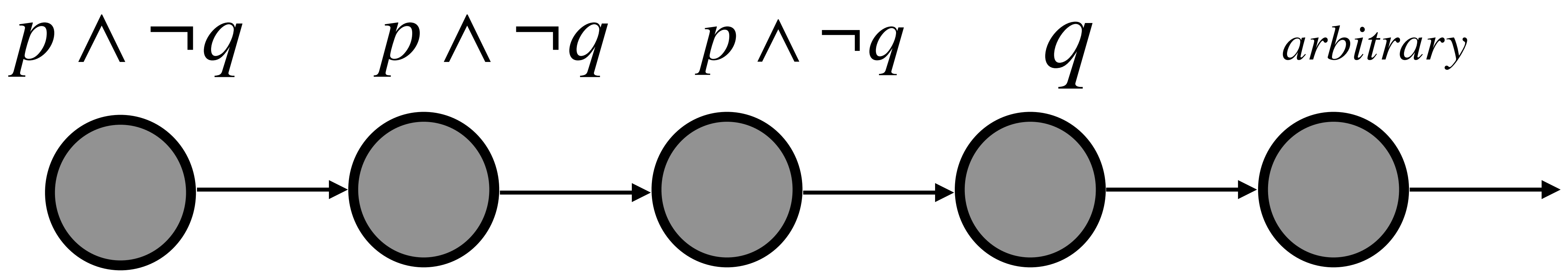
Atomic prop. P



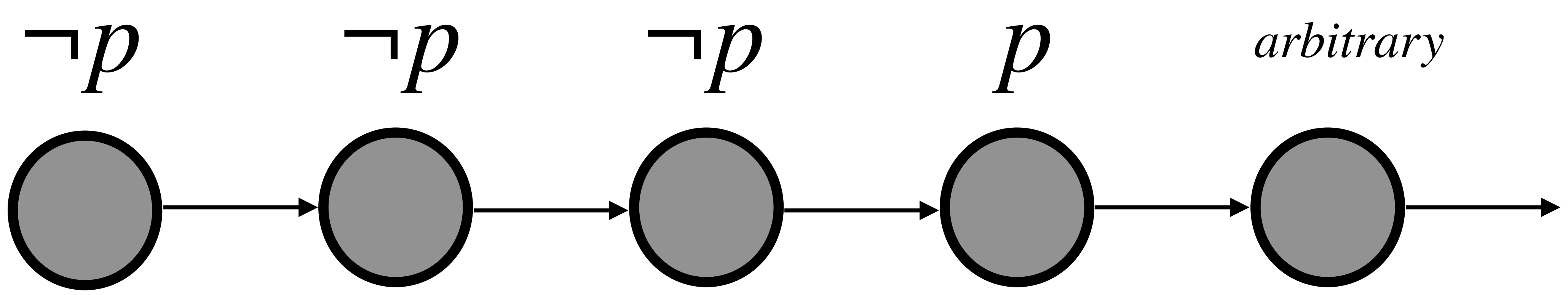
$\mathbf{N}\ p$



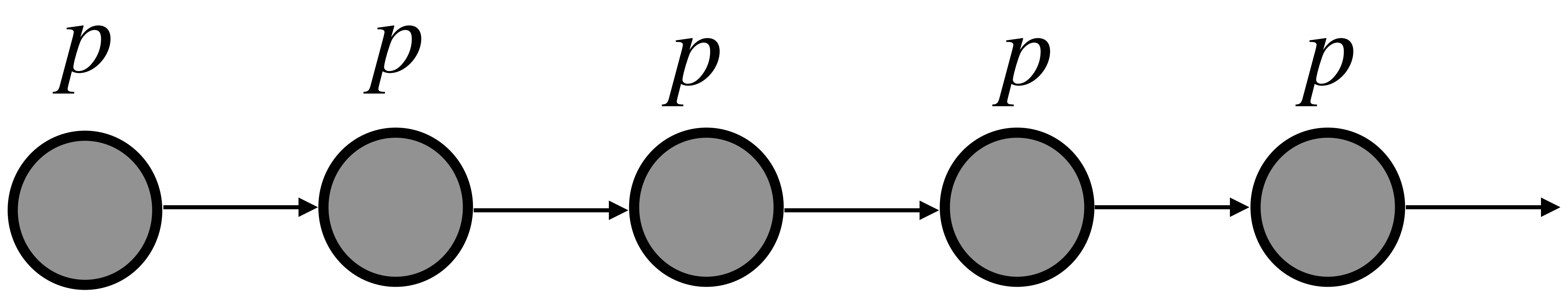
$p\ \mathbf{U}\ q$



$\Diamond\ p$



$\Box\ p$



LTL: Operator Precedence

How to read $\mathbf{N}\ p\ \mathbf{U}\ q$?

Temporal operators before negation $\neg \mathbf{N}p \equiv \neg(\mathbf{N}p)$

Next before Until $\mathbf{N}\ p\ \mathbf{U}\ q \equiv ((\mathbf{N}\ p)\ \mathbf{U}\ q)$

The next state must satisfy p, and p must hold until q happens

Always/Eventually before Until $\Box\ p\ \mathbf{U}\ q \equiv ((\Box\ p)\ \mathbf{U}\ q)$

Always p holds until q happens

Always/Eventually before logical operators $\Box\ p \vee q \equiv ((\Box\ p) \vee q)$

Either p always holds or q must hold in the current state.

LTL: Common Cases

Response — If p then eventually q. $p \rightarrow \Diamond q$

Precedence — If p then q until r. $p \rightarrow (q\ \mathbf{U}\ r)$

Stability — Once we reach the stable state, we will always be in stable state.

$\Box(p \rightarrow \mathbf{N}\ p)$ Or $\Box(p \rightarrow \Box\ p)$

— We will definitely reach stable state, and once we reach the stable state, we will always be in stable state. $\Diamond \Box p$

Progress — We will always reach the stable state or desired state. $\Box \Diamond p$

Correlation — Eventually p implies eventually q. $(\Diamond p) \rightarrow (\Diamond q)$

LTL: Examples

Traffic light is green infinitely often.

$$\Box \Diamond \mathit{green}$$

Once red, the light can't become green immediately.

$$\Box (\mathit{red} \rightarrow \neg \mathbf{N}\, \mathit{green})$$

Once red, the light always becomes green eventually after being yellow for some time.

$$\square(red \rightarrow (\Diamond\ green\ \wedge (\neg green\ \mathbf{U}\ yellow)))$$

$$\square(red \rightarrow \mathbf{N}\ (red\ \mathbf{U}\ (yellow\ \wedge\ \mathbf{N}\ (yellow\ \mathbf{U}\ green))))$$

$$\square(red \rightarrow (\mathbf{N}\ (\neg green \wedge yellow) \wedge ((\neg green \wedge yellow)\mathbf{U}\ green))))$$

Suggestions from today's class:

$$\square(red \rightarrow (\neg\mathbf{N}\ green \wedge (yellow\mathbf{U}\ green)))$$

$$\square(red \rightarrow (\neg\mathbf{N}\ green \wedge ((\mathbf{N}\ yellow)\mathbf{U}\ green)))$$

$$\square(red \rightarrow \mathbf{N}\ red \vee (\ \mathbf{N}\ yellow \wedge (yellow\ \mathbf{U}\ green)))$$

$$\square(red \rightarrow ((\neg\mathbf{N}\ green) \wedge \Diamond yellow \wedge (yellow\ \mathbf{U}\ green)))$$

Course Webpage

Thanks!