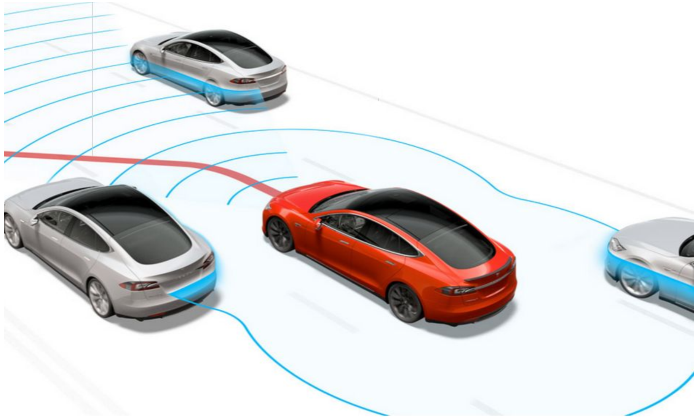
Linear Temporal Logic (LTL)

An Introduction

What is LTL?

When designing software or automated systems, we need a precise way to describe how things should behave over time. LTL allows us to specify rules about what must happen now, in the next step, or at any point in the future.



Autopilot System:

- If collision warning is detected, trigger emergency braking.

$\boxed{G\ (collision\_warning) \rightarrow X\ (emergency\_brake)}$

atomic prepositions

logical operator

$$G\ (\text{collision\_warning}) \Rightarrow X\ (\text{emergency\_brake})$$

temporal operators

Logical Operators

Operator	Meaning
AND (&)	Both conditions must be true. a & b *means* ‘Both “a” AND “b” must be true.’
OR (|)	At least one condition must be true" a | b *means* ‘Either “a” OR “b” (or both) must be true’
NOT (!)	The condition must be false !a *means* ‘“a” must be false’ or ‘NOT “a”’
IMPLIES (⇒)	If the first condition is true, then the second must also be true a ⇒b *means* ‘IF “a” is true, THEN “b” must be true’

Temporal Operators

Operator	Meaning
X (Next)	A statement must be true in the next time step. X d *means* ‘In the next moment, “d” must be true.’
G (Globally)	A statement must always be true in every future step. G b *means* ‘“b” must always hold.’
F (Eventually)	A statement must be true at some future point. F a *means* ‘At some point, “a” must hold.’
U (Until)	One statement must hold until another becomes true. b U a *means* ‘“b” must hold until “a” eventually holds.’

Semantics

- G (start)

  start, start, start, …

- X (cook)

  [any], cook, [any], …

- G (X cook)

  ε (the empty word)

- X (G cook)

  [any], cook, cook, cook, …

- F (cook)

  [any], [any], cook, [any], …

  (cook can be in any position)

Semantics … more formally

- G (start)

start, start, start, … is equivalent to say

start: true, true, true, …

- X (cook)

[any], cook, [any], …… is equivalent to say

cook: [any], true, [any], …

A sequence that satisfies the LTL formula:

$$G\ (\text{start} \Rightarrow X\ (\text{cook}\ U\ \text{stop})\ )$$

- It must be true that throughout the whole sequence that:
  - IF start is True at time t for ANY time t)
  - Then at time t+1: cook must be true until stop is true (t+1 or later).

time:	1	2	3	4	5	**6**	7	8	9	…
start:	false	**true**	false	**true**	false	**true**	false	false	false	
cook:	[any]	[any]	true	true	[any]	false	[any]	[any]	[any]	
stop:	[any]	[any]	false	false	**true**	false	**true**	[any]	[any]	

A sequence that satisfies the LTL formula:

$$G\,(\,start \Rightarrow X\,(cook\ U\ stop)\,)$$

- It must be true that throughout the whole sequence that:
  - IF start is True at time t for ANY time t)
  - Then at time t+1: cook must be true until stop is true (t+1 or later).

time:	1	2	3	4	5	**6**	7	8	9	…
start:	false	**true**	false	**true**	false	**true**	false	false	false	
cook:	[any]	[any]	true	true	[any]	false	[any]	[any]	[any]	
stop:	[any]	[any]	false	false	**true**	false	**true**	[any]	[any]	

The "promises" started when start = True are never broken

A sequence that satisfies the LTL formula:

$$G \,(\, \text{start} \Rightarrow X\, (\text{cook}\ U\ \text{stop})\,)$$

- It must be true that throughout the whole sequence that:
  - IF start is True at time t for ANY time t)
  - Then at time t+1: cook must be true until stop is true (t+1 or later).

BUT:

- Nothing in the formula indicates that hitting stop makes cook False (the microwave might keep running)
- Also, you should not activate start while it “is running” and the formula does not prevent this.

A sequence that satisfies the LTL formula:

G (start ⇒ X (cook U stop))

- It must be true that throughout the whole sequence that:
  - IF start is True at time t for ANY time t)
  - Then at time t+1: cook must be true until stop is true (t+1 or later).

A slightly better formula is:

G (start ⇒ X (cook U (stop & !cook)))

This helps specify that the microwave must stop cooking when stop is true.

A sequence that DOES NOT satisfy the LTL formula:

$$G\ (\ start \Rightarrow X\ (cook\ U\ stop)\ )$$

- It must be true that During the whole sequence that:
  - IF start is True at time t for ANY time t)
  - Then at time t+1: cook must be true until stop is true

time:	1	2	3	4	5	**6**	7	8	9	…
start:	false	**true**	false	**true**	false	**true**	false	false	false	
cook:	[any]	[any]	true	false	[any]	false	[any]	[any]	[any]	
stop:	[any]	[any]	false	false	**true**	false	**true**	[any]	[any]	

Where is the problem?

A sequence that DOES NOT satisfy the LTL formula:

$$G\ (\ start \Rightarrow X\ (cook\ U\ stop)\ )$$

- It must be true that During the whole sequence that:
  - IF start is True at time t for ANY time t)
  - Then at time t+1: cook must be true **until stop is true**

time:	1	2	3	4	5	**6**	7	8	9	…
start:	false	**true**	false	**true**	false	**true**	false	false	false	
cook:	[any]	[any]	true	**false**	[any]	false	[any]	[any]	[any]	
stop:	[any]	[any]	false	false	**true**	false	**true**	[any]	[any]	

G (start ⇒ X(cook U stop))

start:	false	false	false	false	false	…	Since start is always false, the formula is satisfied by this sequence, independently from the value of cook and stop.
cook:	[any]	[any]	[any]	[any]	[any]	…	
stop:	[any]	[any]	[any]	[any]	[any]	…	

start:	false	**true**	<u>true</u>	false	false	…	the formula is satisfied by this sequence
cook:	[any]	[any]	**true**	<u>true</u>	[any]	…	
stop:	[any]	[any]	[any]	**true**	**<u>true</u>**	**<u>…</u>**	

start:	true	false	**true**	false	false	…	the formula is not satisfied by this sequence
cook:	[any]	true	[any]	**false**	[any]	…	
stop:	[any]	[any]	true	true	true	…	

Parentheses and Precedence

Consider a basic microwave:

Example 1: $G \boxed{(\text{ start} \Rightarrow X \boxed{(\text{cook } U \text{ stop})})}$

Example 2: $\boxed{G \text{ start}} \Rightarrow \boxed{X\ (\text{cook } U \text{ stop})}$

Example 3: $G \boxed{(\text{ start} \Rightarrow X \text{ cook } U \text{ stop })}$

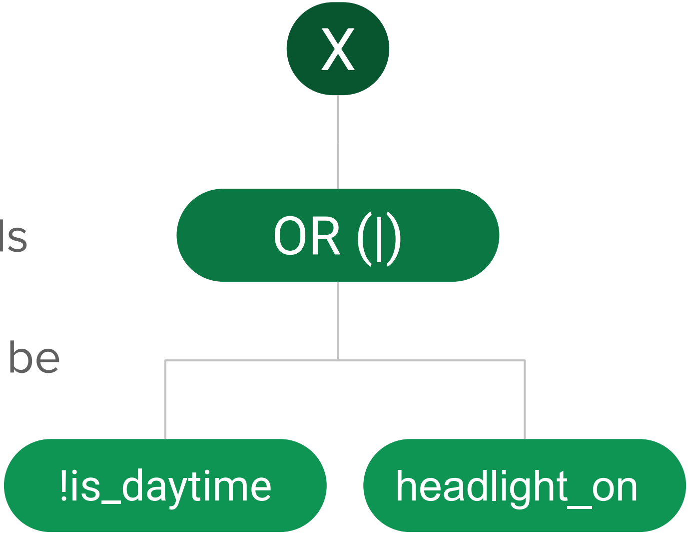
$G\ (\text{ start} \Rightarrow (X \text{ cook})\ U \text{ stop })$

Breaking Down and Translating LTL Formulas (1)

Example: Autonomous Car: Automatic Headlights Activation

Formula: X (**!**is_daytime | headlight_on)

- X (Next-state operator) ensures the condition holds in the next moment.
- | (OR operator) means at least one condition must be true.
- If **not daytime**, headlights **must** be on.
- If **daytime**, headlights **may** be on, but it's not required.

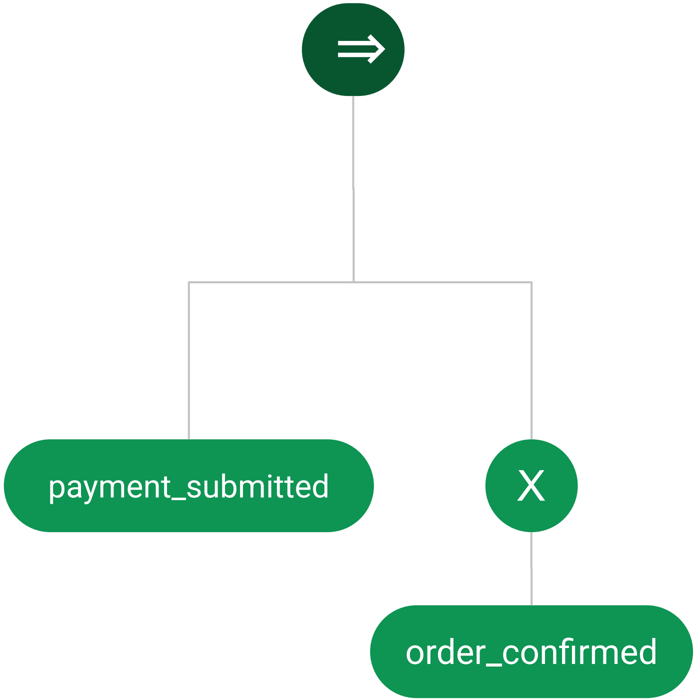


Breaking Down and Translating LTL Formulas (2)

Online Checkout: Payment & Order Confirmation

Formula: $\boxed{\text{payment\_submitted} \Rightarrow X \text{ order\_confirmed}}$

- **⇒** (Implication) means if the first condition is true, the second must follow.
- **X** (Next-state operator) ensures the order is confirmed in the next step.
- If **payment is submitted**, the system **must** confirm the order in the next state.
- If **payment is not submitted**, no guarantee is required.



Key Concepts

Core Operators

Logical:

- → (IMPLIES): If...then
- & (AND): Both true
- | (OR): Either true
- ! (NOT): Must be false

Temporal:

- X (Next): True in next step
- G (Globally): Always true
- F (Eventually): True at some point
- U (Until): True until condition

Example Application

Autopilot Safety Rule:

G(collision_warning) → X(emergency_brake)

"If collision warning is detected, trigger emergency braking in the next step."

Online Checkout:

payment_submitted → X(order_confirmed)

"If payment is submitted, order must be confirmed in the next state."