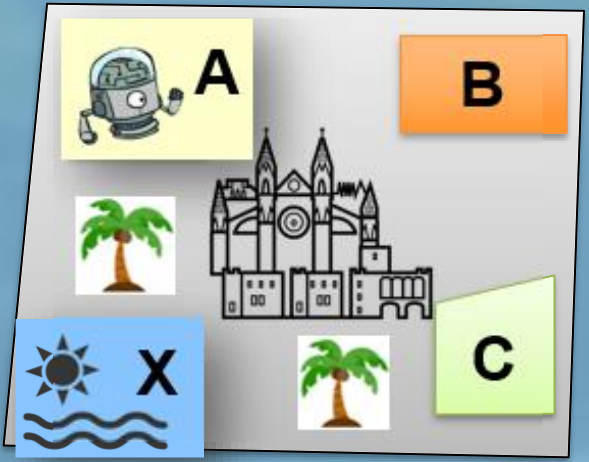
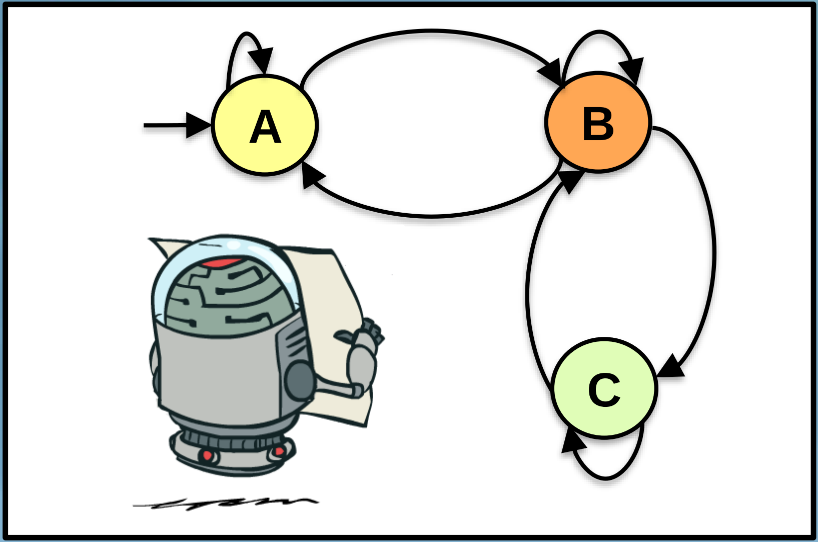
Graz University of Technology
Institute for Applied Information Processing and Communications

Temporal Logic





Model Checking SS24

Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

22nd April 2024

Temporal Logic

- Used to specify the dyamic behavior of systems.
- E.g., A temporal logic formula can express that…
  - …a property has to hold in the next time step.
  - …a property has to hold always.
  - …a property has to hold eventually.
  - …
- MC Question
  - Does the model of the system satisfy a temporal logic formula?
- System model
  - **Kripke structure (today)**
  - I/O Automaton
  - Markov Decision Process / Stochastic Multiplayer Game, ….

Plans for the Next 4 Weeks

- Topic: Model Checking of Temporal Logic Formulas

1. Intro to Temporal Logics: CTL*, LTL, CTL
2. CTL Model Checking – Part 1
3. CTL Model Checking – Part 2
4. LTL Model Checking

- Next: Model Checking of Probabilistic Systems (Stefan’s Part)

Plan for Today

- Motivating Example
- CTL*
  - Informal Explanation of Syntax and Semantics
  - Syntax
  - Semantics
- Sublogics: CTL, LTL

Warm Up

Model sentences in propositional logic.

- “If a sentence as a truth value, then it is a declarative sentence.”

- “A model is an assignment that makes a formula either true or false.”

Warm Up

Model sentences in propositional logic.

- “If a sentence as a truth value, it is a declarative sentence.”

  p... sentence has a truth value, q… sentence is a declarative sentence

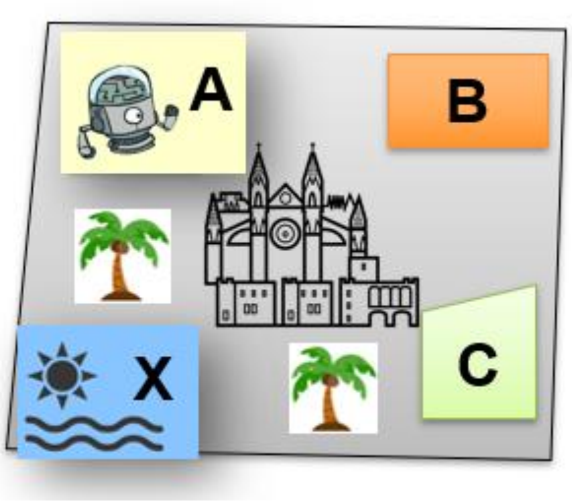
  $$p \rightarrow q$$

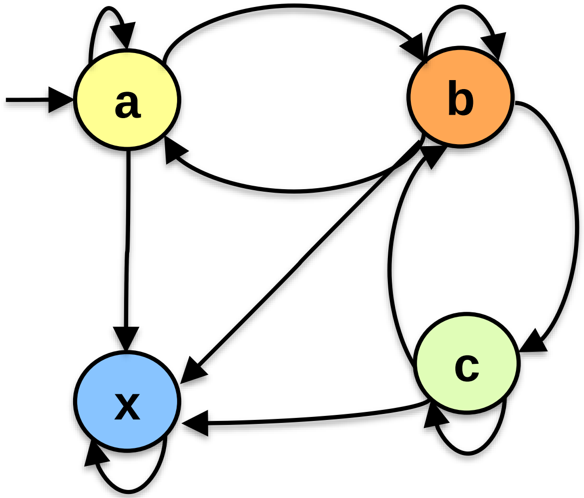
- “A model is an assignment that makes a formula either true or that makes the formula false.”

  p... assignment that makes the formula true,
  q… assignment that makes the formula false

  $$p \oplus q$$

Properties of Kripke Structures





Properties

- For any execution, it is always the case that if the robot visits A, it visits C within the next two steps.
- There exists an execution, in which the robot always visits C within the next two steps after visiting A.

Write properties as formulas:

For detailed modelling, we need…

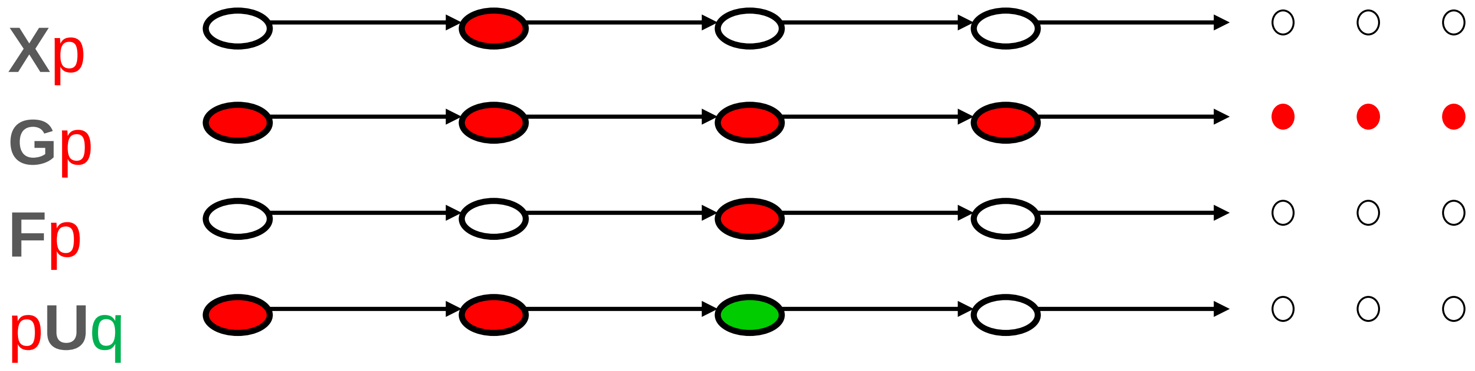
- temporal operators, and
- path quantifiers!

Propositional Temporal Logic

AP – a set of atomic propositions, $p, q \in AP$

Temporal operators

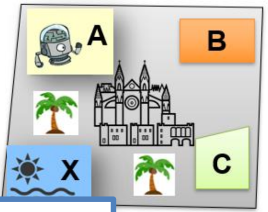
- Describe properties along a given path/execution
- 3 operators to start with:

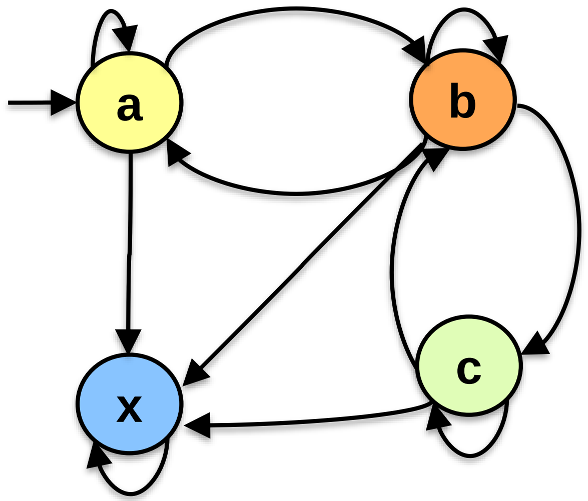


Path quantifiers: **A** for **all** paths

E there exists a path

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

Path quantifiers
A for **all** paths
E there exists a path

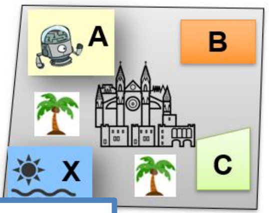
Properties

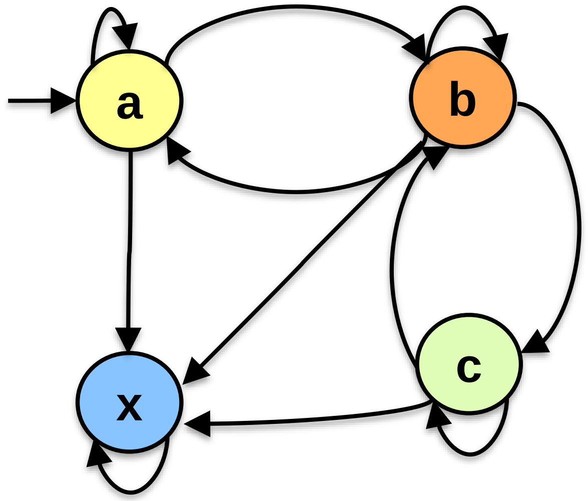
- For any execution, it is always the case that if the robot visits **A**, it visits **C** within the next two steps.

Write properties as formulas:

$$A\,G\,(a \rightarrow Xc \vee XXc)$$

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

Path quantifiers
A for **all** paths
E there exists a path

Properties

- For any execution, it is always the case that if the robot visits **A**, it visits **C** within the next two steps.
- There exists an execution in which it is always the case that if the robot visits **A**, it visits **C** within the next two steps.

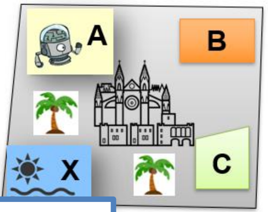
Write properties as formulas:

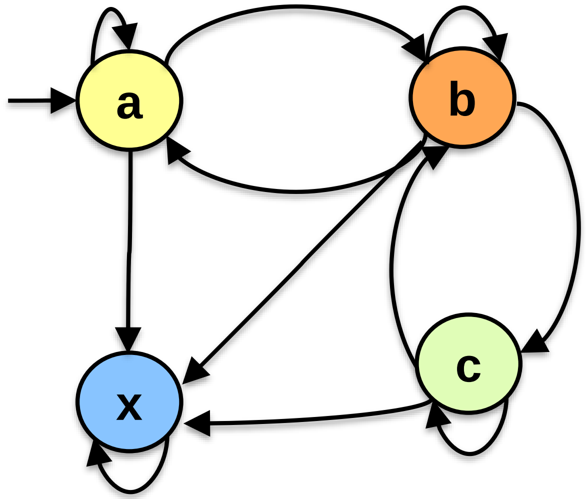
$$A\,G\,(a \rightarrow Xc \vee XXc)$$

$$E\,G\,(a \rightarrow Xc \vee XXc)$$

SCOS Secure & Correct Systems

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

Path quantifiers
A for **all** paths
E there exists a path

Properties

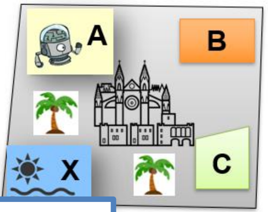
- For any execution it holds that the robot *never* visits **X.**
- There exists an execution in which it holds that the robot *never* visits **X.**

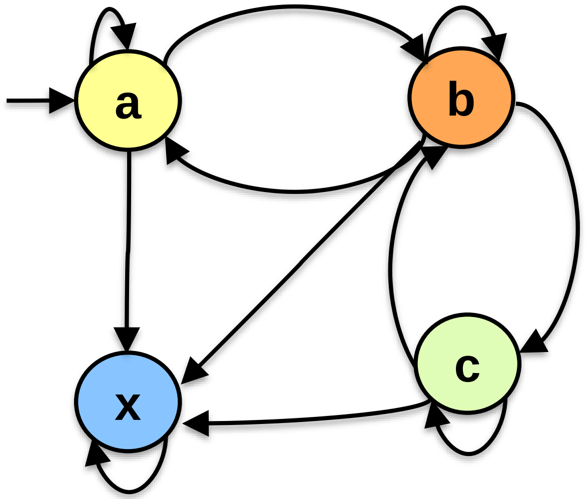
Write properties as formulas:

$$A\ G\ (\neg x)$$

$$E\ G\ (\neg x)$$

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

Path quantifiers
A for **all** paths
E there **exists** a path

Properties

- There exists an execution in which it holds that the robot visits **A** *infinitely often* and **C** *infinitely often.*
- For any execution, it holds that the robot visits **A** *infinitely often*, but **C** only *finitely often*.

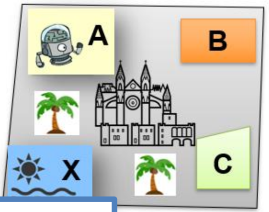
Write properties as formulas:

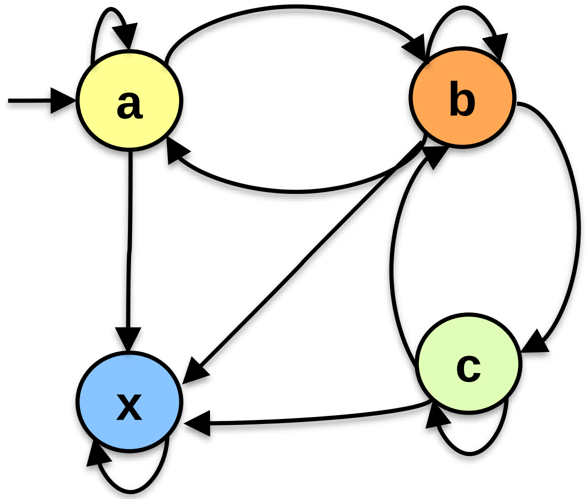
$$E\ (GF\ a \wedge GF\ c)$$

$$A\ (GF\ a \wedge FG \neg c)$$

SCOS Secure & Correct Systems

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

Path quantifiers
A for **all** paths
E there **exists** a path

Properties

- For any execution, it holds that if the robot visits **A** *infinitely often,* it also visits **C** *finitely often.*

Write properties as formulas:

$$A\,(GF\,a \rightarrow FG\neg c)$$

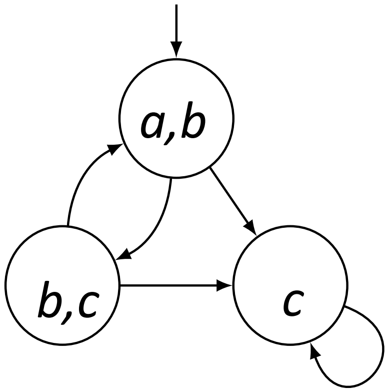
Plan for Today

- Motivating Example and Intuitive Explanation of Temporal Operators
- CTL*
  - Informal Explanation of Syntax and Semantics
  - Syntax
  - Semantics
- LTL
- CTL

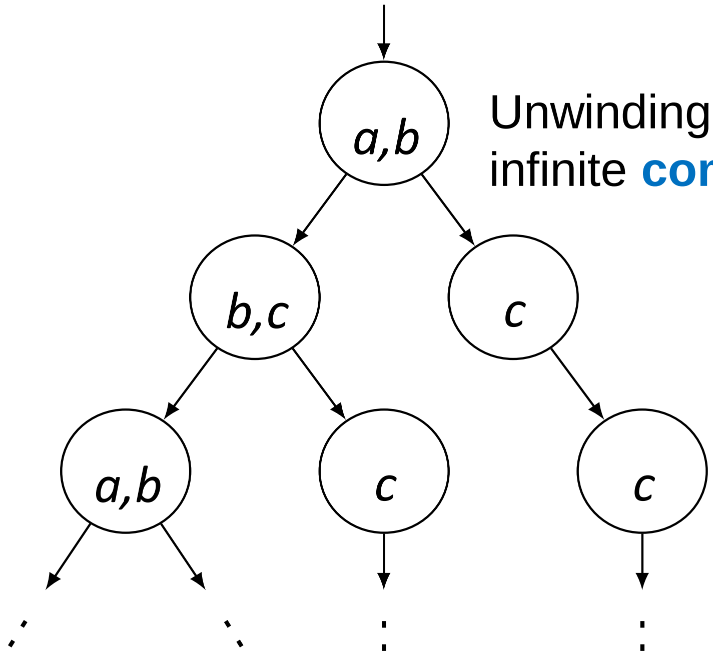
Computation Tree Logic - CTL*

- Defines properties of Computation Trees of Kripke structures.
- Computation Tree
  - Shows all possible executions starting form initial state.
  - All branches of the tree are infinite.

Kripke structure M,
labeled with $AP = \{a, b, c\}$

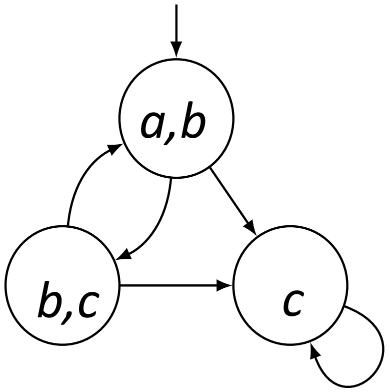


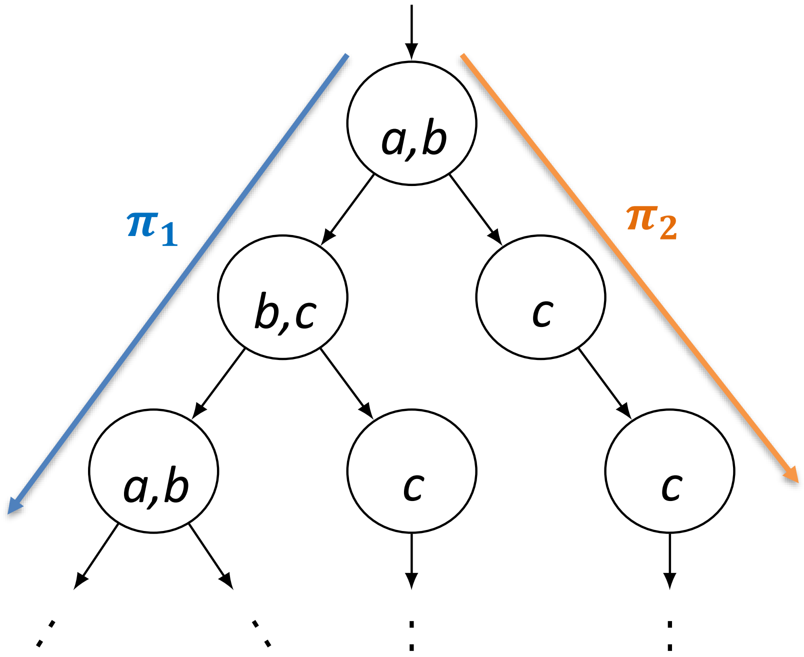
Unwinding of M into
infinite **computation tree**



Paths

- $\pi = s_0, s_1, \ldots$ is an *infinite* **path** in M if
  - s_0 is an initial state, and
  - for all $i \geq 0$, $(s_i, s_{i+1}) \in R$

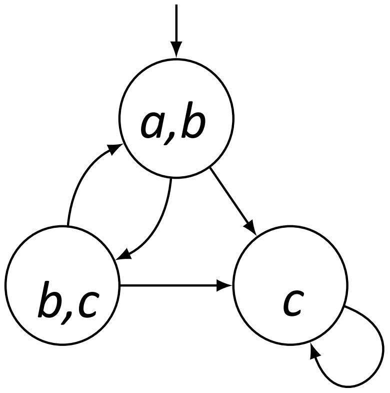


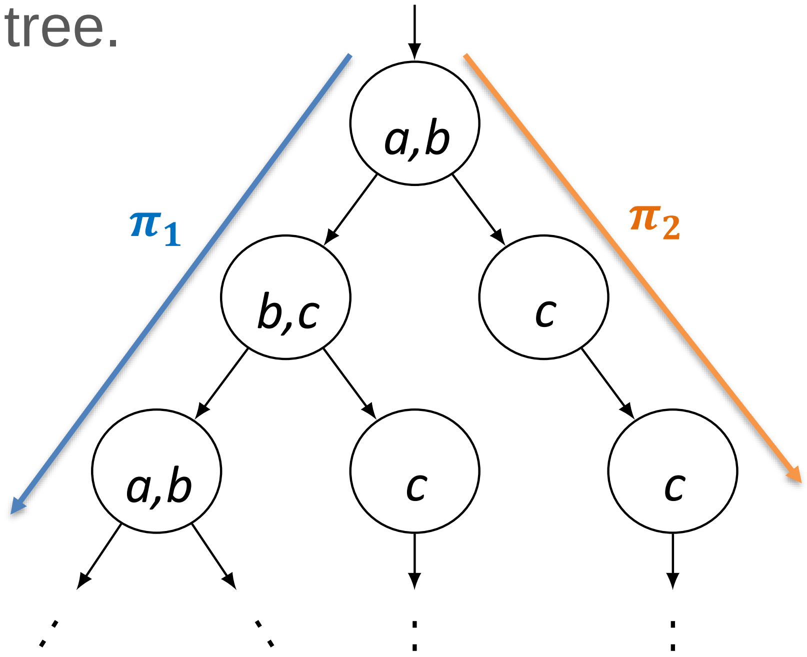


Propositional Temporal Logic

Path quantifiers: A, E

- **A** specifies that **all** paths starting from **s** have property φ.
- **E** specifies that **some** paths starting from **s** have property φ.
- Use combination of **A and E** to describe branching structure in tree.

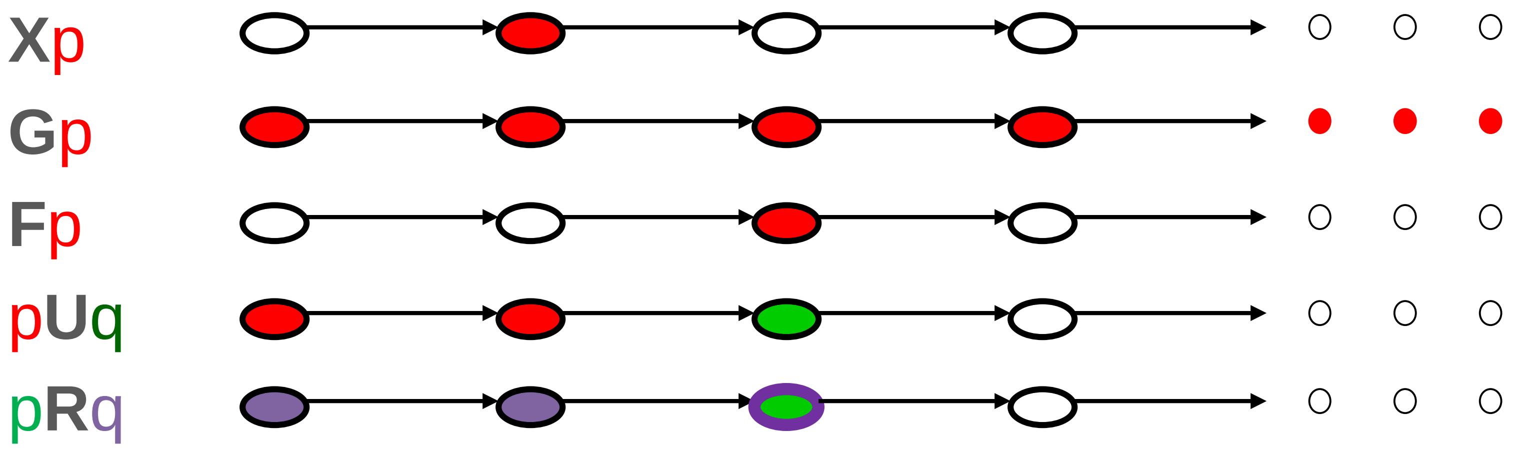




Propositional Temporal Logic

Temporal operators:

- Describe properties that hold along an infinite path π

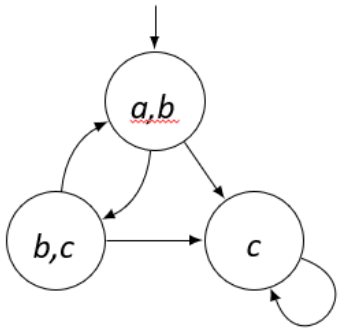


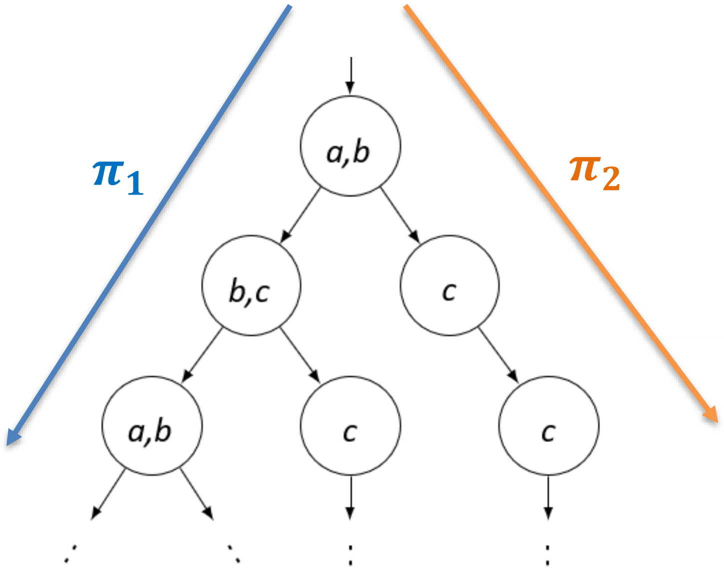
p**R**q "p release q":

p**R**q requires that q holds along π up to and including the first state where p holds. However, p is not required to hold eventually.

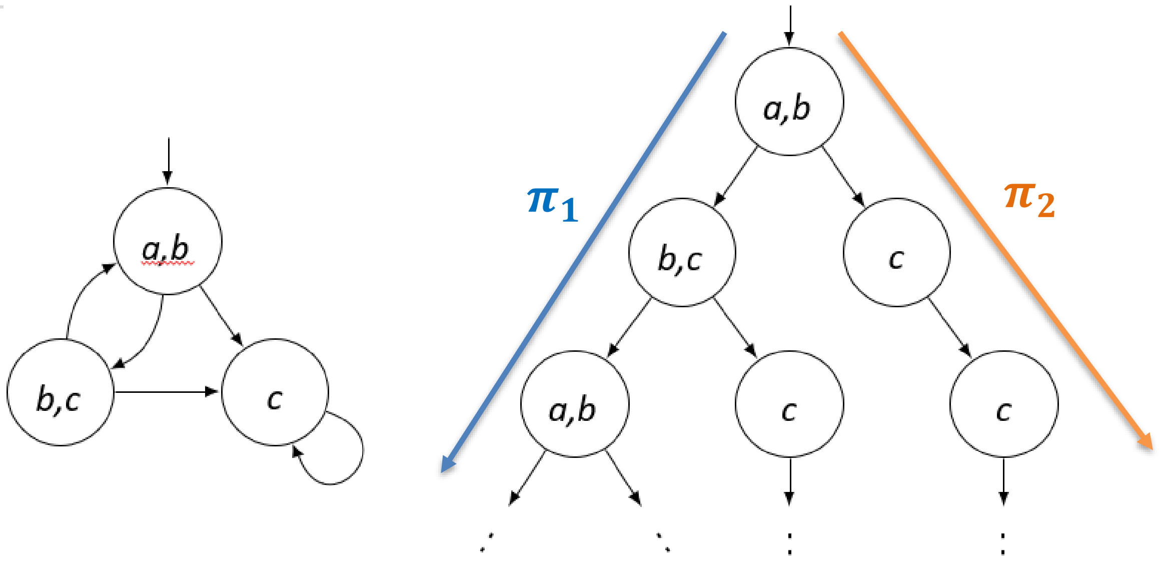
Informal Semantics of State and Path Formulas

- Illustrate CTL* Semantics on Example
- Path Formulas:
  - On π_1, b holds at every state. → $\pi_1 \vDash Gb$
  - On π_2, b does not hold at every state. → $\pi_2 \nvDash Gb$
- State Formulas:
  - There is a path from s_0 that satisfies Gb →$s_0 \vDash \mathrm{EG\,b}$
  - Not all paths from s_0 satisfy Gb →$s_0 \nvDash \mathrm{AG\,b}$





Informal Semantics of State and Path Formulas



- Does s_0 satisfy the following formula?
  - $s_0 \vDash \text{EXX}\,(\text{a} \wedge b)$
  - $s_0 \nvDash \text{EXAX}\,(\text{a} \wedge b)$

Plan for Today

- Motivating Example
- CTL*
  - Informal Explanation of Syntax and Semantics
  - Syntax
  - Semantics
- Sublogics: CTL, LTL

Syntax of CTL*

Two types of formulas in the inductive definition

- State formulas (true in a specific state)
- Path formulas (true along a specific path)

CTL* formulas are the set of all state formulas

Syntax of CTL*: State Formulas

Inductive definition of state formulas:

- If $p \in AP$, then p is a state formula.
- If f_1 and f_2 are state formulas, so are $\neg f_1$, $f_1 \vee f_2$, and $f_1 \wedge f_2$.
- If g is a path formula, then $\boldsymbol{E}g, \boldsymbol{A}g$ are state formulas.

Inductive definition of path formulas:

- If f is a state formula, then f is also a path formula.
- If g_1, g_2 are path formulas, then $\neg g_1$, $g_1 \vee g_2, g_1 \wedge g_2$,
  $\boldsymbol{X} g_1$, $\boldsymbol{G} g_1$, $\boldsymbol{F} g_1$, $g_1 \boldsymbol{U}\, g_2$, $g_1 \boldsymbol{R}\, g_2$
  are path formulas.

CTL* is the set of all state formulas!

Plan for Today

- Motivating Example
- CTL*
  - Informal Explanation of Syntax and Semantics
  - Syntax
  - Semantics
- Sublogics: CTL, LTL

Semantics of CTL*

- Kripke Structure $M = (S, S_0, R, AP, L)$
- $\pi = s_0, s_1, \ldots$ is an infinite **path** in M
- π^i – the **suffix** of π, starting at s_i

- For state formulas:
  - $\boldsymbol{M, s} \vDash \boldsymbol{f}$ … the **state** formula f holds in state s of M

- For path formulas:
  - $\boldsymbol{M, \pi} \vDash \boldsymbol{g}$ … the **path** formula g holds along π in M

Semantics of CTL*

- Let g_1 and g_2 be path formulas and f_1 and f_2 be state formulas.
- $\models$ is inductively defined via the structure of the formula.

State formulas:

- $M, s \models p \quad \Leftrightarrow p \in L(s)$ for $p \in AP$
- $M, s \models \mathbf{E}\, g_1 \quad \Leftrightarrow$ there is a path π from s s.t. $M, \pi \models g_1$
- $M, s \models \mathbf{A}\, g_1 \quad \Leftrightarrow$ for every path π from s s.t. $M, \pi \models g_1$
- Boolean combination ($\wedge$, $\vee$, $\neg$) – the usual semantics

Semantics of CTL*

Let g_1 and g_2 be path formulas and f_1 and f_2 be state formulas.

Path formulas:

- $M, \pi \vDash f_1$ $\Leftrightarrow$ s is the first state of π and $M, s \vDash f_1$
- $M, \pi \vDash$ X g_1 $\Leftrightarrow$ $M, \pi^1 \vDash g_1$
- $M, \pi \vDash$ G g_1 $\Leftrightarrow$ for every $i \geq 0$, $M, \pi^i \vDash g_1$
- $M, \pi \vDash$ Fg_1 $\Leftrightarrow$ there exists $k \geq 0$, $M, \pi^k \vDash g_1$
- $M, \pi \vDash g_1$ U g_2 $\Leftrightarrow$ there exists $k \geq 0$, $M, \pi^k \vDash g_2$ and for every $0 \leq j < k$, $M, \pi^j \vDash g_1$

$M \vDash f_1$ $\Leftrightarrow$ for all initial states $s_0 \in S_0$: $M, s_0 \vDash f_1$

Properties of CTL*

The operators **∨, ¬, X, U, E** are sufficient to express any CTL* formula:

- $f_1 \wedge f_2 \equiv \neg(\neg f_1 \vee \neg f_2)$
- $\boldsymbol{F}\, g_1 \equiv true\, \boldsymbol{U}\, g_1$
- $\boldsymbol{G}\, g_1 \equiv \neg \boldsymbol{F} \neg g_1$
- $\boldsymbol{A}\, f \equiv \neg \boldsymbol{E} \neg f$
- $g_1\, \boldsymbol{R}\, g_2 \equiv \neg(\neg g_1\, \boldsymbol{U} \neg g_2)$ or
  $g_1\, \boldsymbol{R}\, g_2 \equiv \left(g_2\, \boldsymbol{U}\, (g_1 \wedge g_2)\right) \vee G\, g_2$
  - Intuitively, once g_1 becomes true, it “releases” g_2.
    If g_1 never becomes true then g_2 stays true forever
  - Rewrite it using the operators U, F, G, or X

Negation Normal Form (NNF)

- Formulas in Negation Normal Form (NNF) are formulas in which negations are applied only to atomic propositions

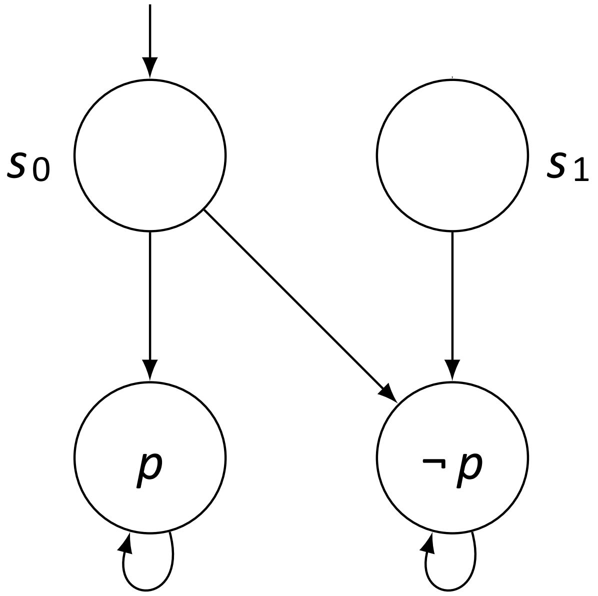
- Every CTL* formula is equivalent to a CTL* formula in NNF
- Negations can be "pushed" inwards.

- $\neg \mathbf{E}\, f \equiv \mathbf{A} \neg f$
  $\neg \mathbf{G}\, f \equiv \mathbf{F} \neg f$
  $\neg \mathbf{X}\, f \equiv \mathbf{X} \neg f$
  $\neg \mathbf{(}\, f\, \mathbf{U}\, g\, \mathbf{)} \equiv \mathbf{(} \neg f\, \mathbf{R} \neg g\, \mathbf{)}$

Example 1: Semantics of CTL*

$\boldsymbol{M} \vDash \boldsymbol{f}_1 \Leftrightarrow$ for all initial states $s_0 \in S_{0}$: $M, s_0 \vDash \boldsymbol{f}_1$

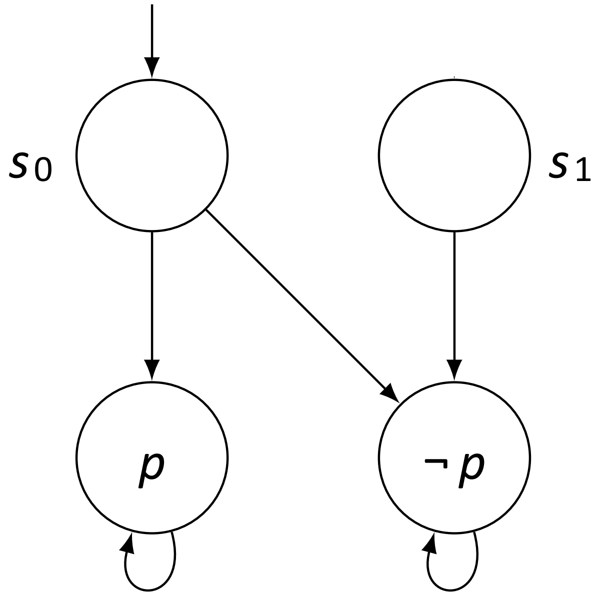
- Does $\boldsymbol{M} \vDash \mathrm{EX}\, p$ or $\boldsymbol{M} \vDash \neg\mathrm{EX}\, p$?



Example 1: Semantics of CTL*

$M \vDash f_1 \;\Leftrightarrow\;$ for all initial states $s_0 \in S_0$: $M, s_0 \vDash f_1$

- Does $M \vDash \mathrm{EX}\, p$ or $M \vDash \neg\mathrm{EX}\, p$?

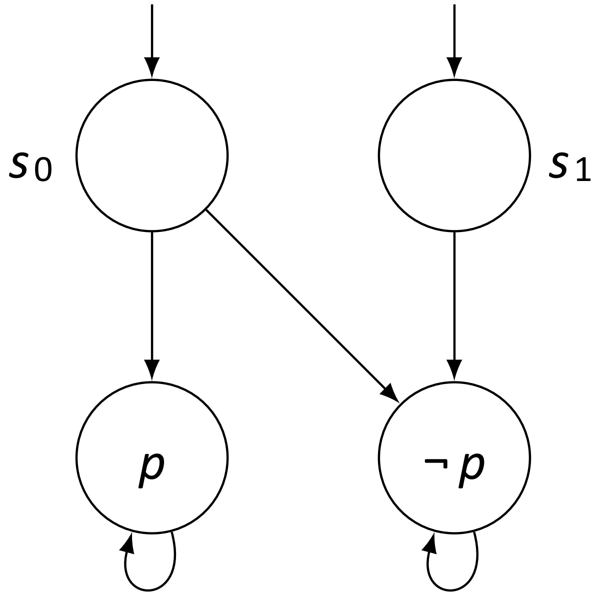


Solution:
$M \vDash \mathrm{EX}\, p$

Example 2: Semantics of CTL*

$\boldsymbol{M} \vDash \boldsymbol{f_1}$ $\Leftrightarrow$ for all initial states $s_0 \in S_0$: $M, s_0 \vDash \boldsymbol{f_1}$

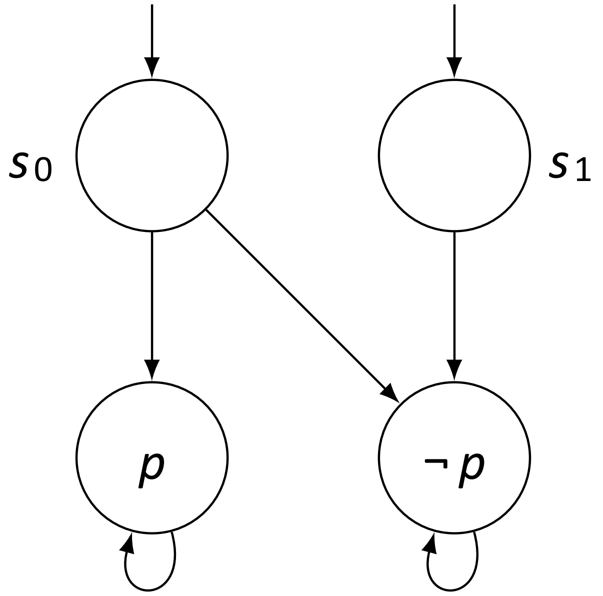
- Does $\boldsymbol{M} \vDash \text{EX}\ p$ or $\boldsymbol{M} \vDash \neg\text{EX}\ p$?



Example 2: Semantics of CTL*

$\boldsymbol{M} \vDash \boldsymbol{f_1} \Leftrightarrow$ for all initial states $s_0 \in S_0$: $M, s_0 \vDash \boldsymbol{f_1}$

- Does $\boldsymbol{M} \vDash \mathrm{EX}\, p$ or $\boldsymbol{M} \vDash \neg\mathrm{EX}\, p$?



Neither

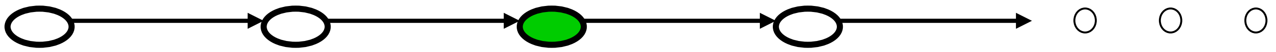
Note, such a situation never happens when M has a single initial state.

Example 3: Semantics of CTL*

Question:

- Given a, b ∈ AP
  How does a path satisfying **F(**a **U** b**)** look like?

F (a **U** b)



Example 4: Semantics of CTL*

Question:

For $p \in AP$, what is the meaning of the following formulas?

- $\pi \vDash$ **GF** p Infinitely often p along π
- $\pi \vDash$ **FG** p Finitely often $\neg p$ along π

Example 5: Semantics of CTL*

Question:

For p ∈ AP, what are the meaning of the following formulas?

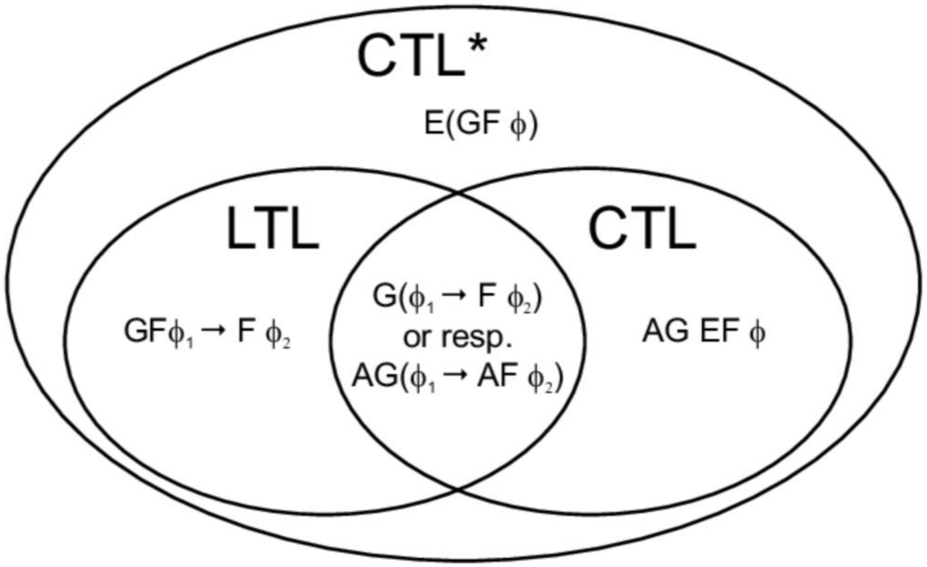
- $s \models$ **EGF** p	There exists a path with satisfies infinitely often p
- $s \models$ **EG EF** p	There exists a path in which we can reach p from all states

Plan for Today

- Motivating Example
- CTL*
  - Informal Explanation of Syntax and Semantics
  - Syntax
  - Semantics
- Sublogics: CTL, LTL

Useful sublogics of CTL*

- **CTL** and **LTL** are the two most used sub-logics of **CTL***
- Differ on how temporal operators and path quantifiers can be combined.
- **CTL*** allows any combination of temporal operators and path quantifiers. It includes both **LTL** and **CTL**.



Linear Temporal Logic (LTL)

LTL consists of state formulas of the form
A g, where g is a path formula, containing no path quantifiers.

- Describes the paths in the computation tree, using only **one, outermost universal quantification.**
- Typically when writing formulas in LTL, the path quantifier is omitted.
- Examples:
  - $GF\ \varphi$
  - $G(\varphi \rightarrow F\ \psi)$
  - $G(\varphi \rightarrow XXX\ \psi)$
  - …

LTL - Syntax

LTL is the set of all state formulas.

State formulas:

- $\mathbf{A}g$ where g is a path formula

Path formulas:

- $p \quad AP$
- $\neg g_1,\ g_1 \vee g_2,\ g_1 \wedge g_2,\ \boldsymbol{X} g_1,\ \boldsymbol{G} g_1,\ \boldsymbol{F} g_1,\ g_1 \boldsymbol{U} g_2,\ g_1 \boldsymbol{R} g_2$

  where g_1 and g_2 are path formulas.

Computation Tree Logic (CTL)

CTL consists of state formulas, where path quantifiers and temporal operators appear in pairs: **AG**, **AU**, **AX**, **AF**, **AR**, **EG**, **EU**, **EX**, **EF**, **ER**

- Examples:
  - $E(\varphi U \psi)$
  - $EF(\varphi) \wedge EG\psi$
  - $AF\ AG\ \varphi$ ...

CTL - Syntax

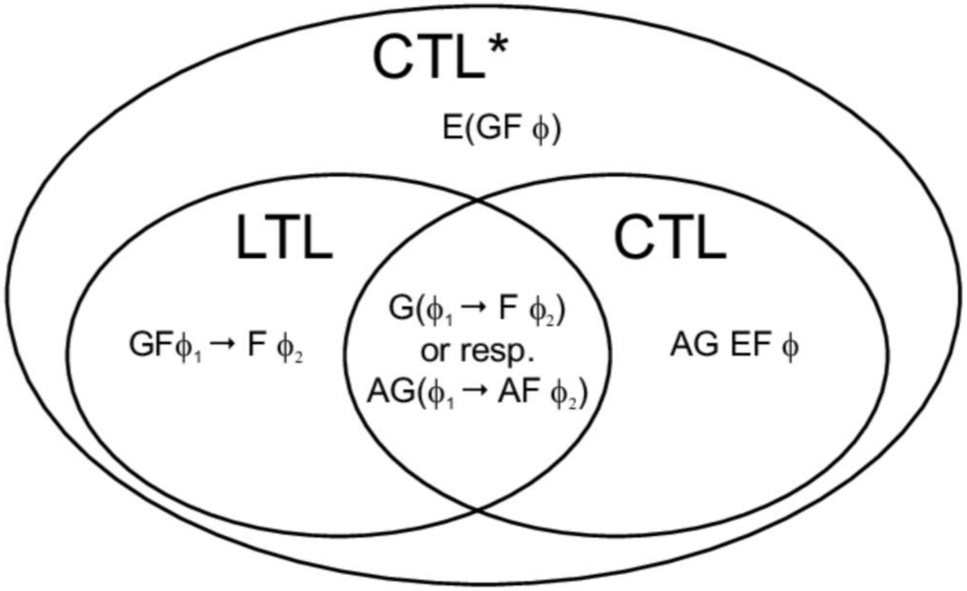
CTL is the set of all state formulas, defined below (by means of state formulas only):

- $p \quad AP$
- $\neg f_1,\ f_1 \vee f_2,\ f_1 \wedge f_2$
- $\boldsymbol{AX}\, f_1, \boldsymbol{AG}\, f_1, \boldsymbol{AF}\, f_1, \boldsymbol{A}\,(f_1\, \boldsymbol{U}\, f_2), \boldsymbol{A}\,(f_1\, \boldsymbol{R}\, f_2)$
- $\boldsymbol{EX}\, f_1,\ \ \boldsymbol{EG}\, f_1,\ \ \boldsymbol{EF}\, f_1,\ \ \boldsymbol{E}\,(f_1\, \boldsymbol{U}\, f_2),\ \ \boldsymbol{E}\,(f_1\, \boldsymbol{R}\, f_2)$

where f_1 and f_2 are state formulas

LTL/CTL/CTL*

- **Linear Temporal Logic (LTL)** consists of state formulas of the form **A**g, where g is a path formula, containing no path quantifiers.
- **CTL** consists of state formulas, where path quantifiers and temporal operators appear in pairs: **AG**, **AU**, **AX**, **AF**, **AR**, **EG**, **EU**, **EX**, **EF**, **ER**



THANK YOU!