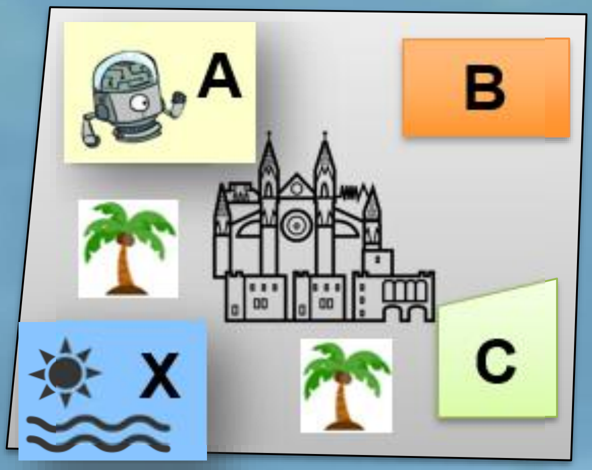
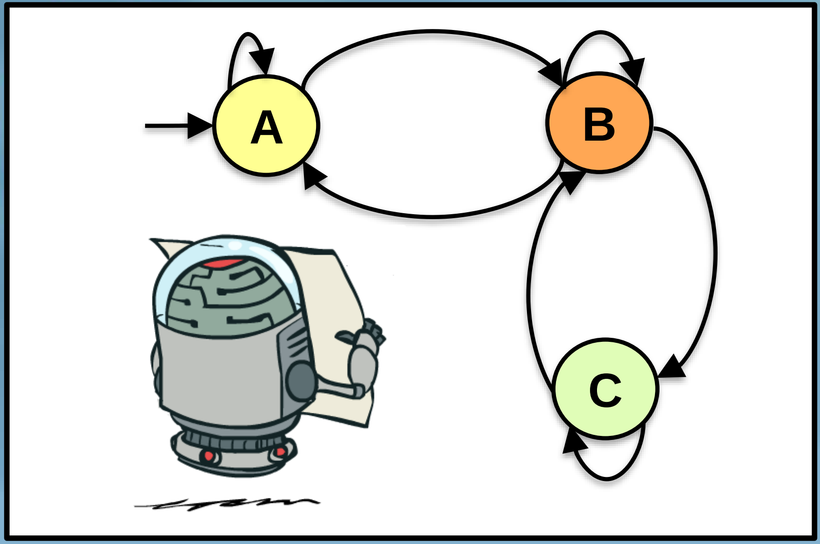
Graz University of Technology
Institute for Applied Information Processing and Communications

Temporal Logic





Model Checking SS23

Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

April 27 2023


ToDo


PLEASE INTERRUPT ME


"TALK TO ME"


WHERE
WHAT
WHEN
WHY
HOW
WHO

Warm Up

Translate sentences to formulas

- "If a sentence has a truth value, it is a declarative sentence."

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

Warm Up

Translate sentences to formulas

- "If a sentence has 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."

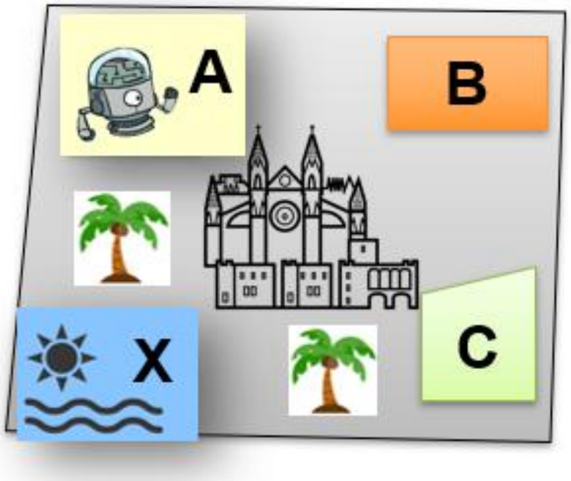
  p... assignment that makes the formula true,
  q… assignment that makes the formula false

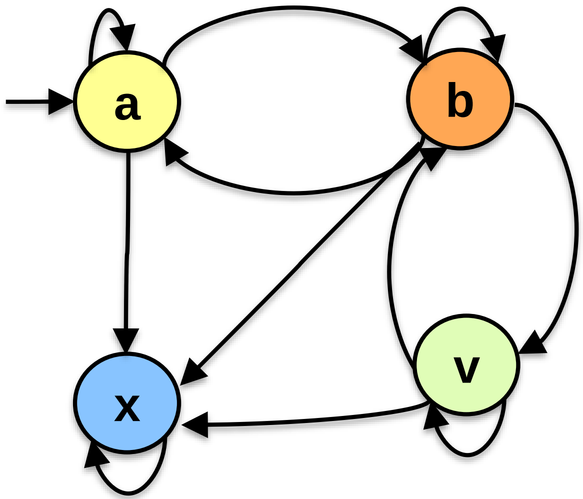
  $$p \oplus q$$

Temporal Logic

- Used to specify the dyamic behavior of systems
- MC Question:
  - Does the model of the system satisfy a temporal logic formula?
- System model:
  - **Kripke structure (today)**
  - I/O Automaton
  - Multiplayer Game
  - Markov Decision Process / Stochastic Multiplayer Game

Properties of Kripke Structures





Properties

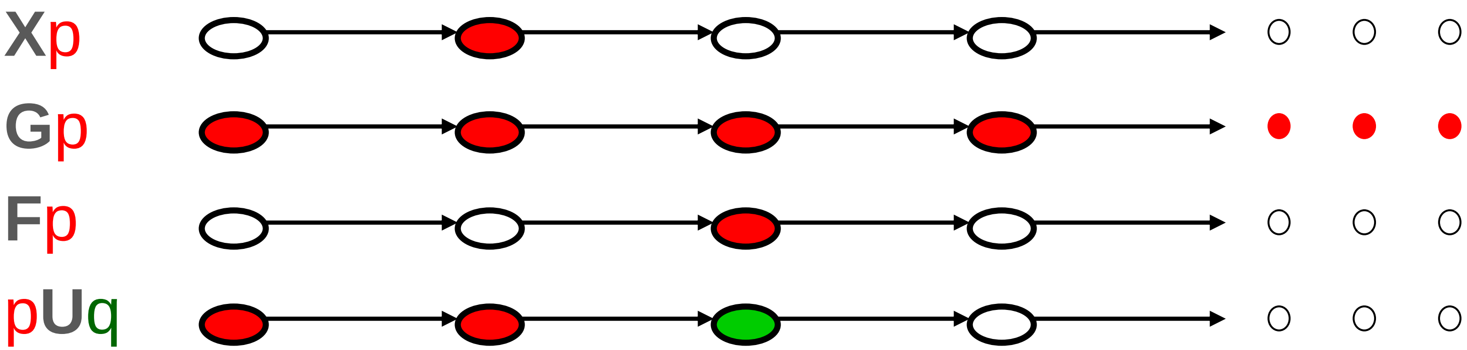
Write properties as formulas

- Always when the robot visits A, it visits C within the next two steps.
- The robot can visit C within the next two steps after visiting A

Propositional Temporal Logic

AP – a set of atomic propositions, p,q∈AP

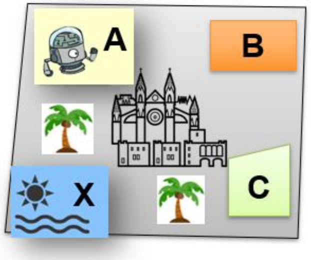
Temporal operators:

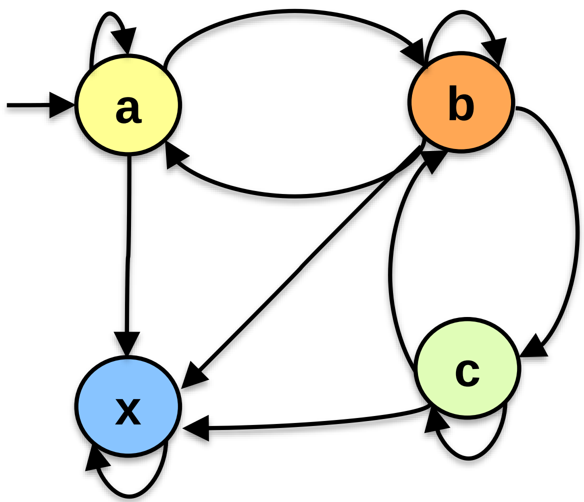


Path quantifiers: **A** for **all** paths

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

ToDo *Write properties as formulas*

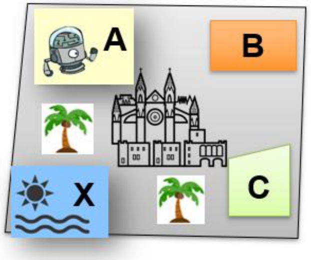
- Always when the robot visits A, it visits C within the next two steps.

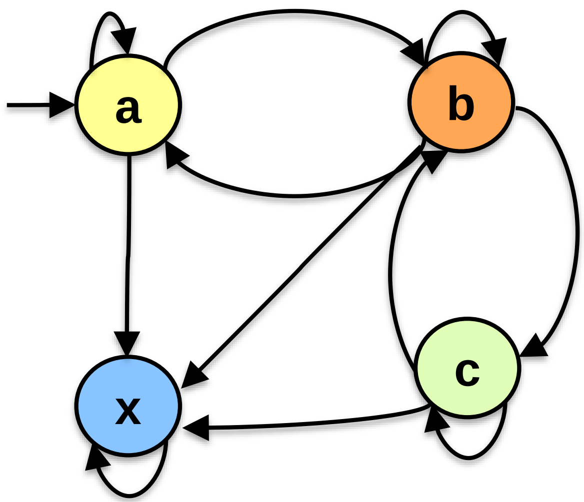
$$A\,G\,(a \rightarrow Xc \lor XXc)$$

- The robot can visit C within the next two steps after visiting A

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

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

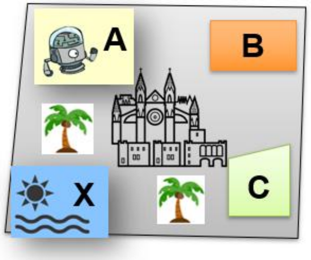
Path quantifiers
A for **all** paths
E there **exists** a path

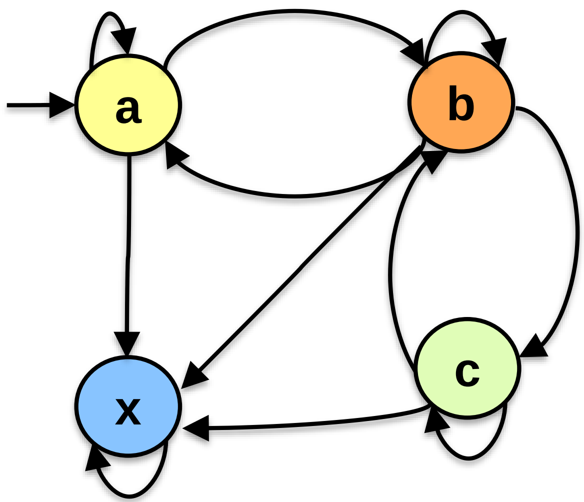
Properties

ToDo *Write properties as formulas*

- The robot *never* visits **X** $A\ G\ \neg x$
- It is possible that the robot *never* visits **X** $E\ G\ \neg x$

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

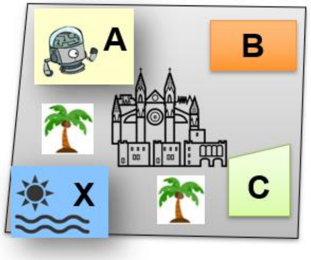
Path quantifiers
A for **all** paths
E there **exists** a path

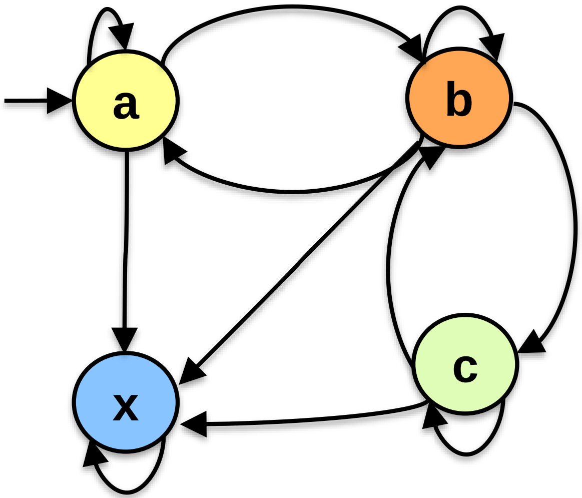
Properties

ToDo *Write properties as formulas*

- The robot can visit **A** and **C** *infinitely often.* $A\ (GF\ a \land GF\ c)$
- The robot always visits **A** *infinitely often*, but **C** only *finitely often*. $E\ (GF\ a \land FG \neg c)$

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

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

Properties

ToDo *Write properties as formulas*

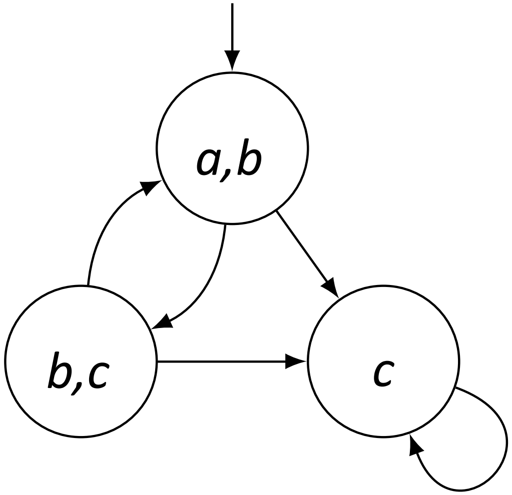
- If the robot visits **A** *infinitely often,* it should visit **C** only *finitely often*.

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

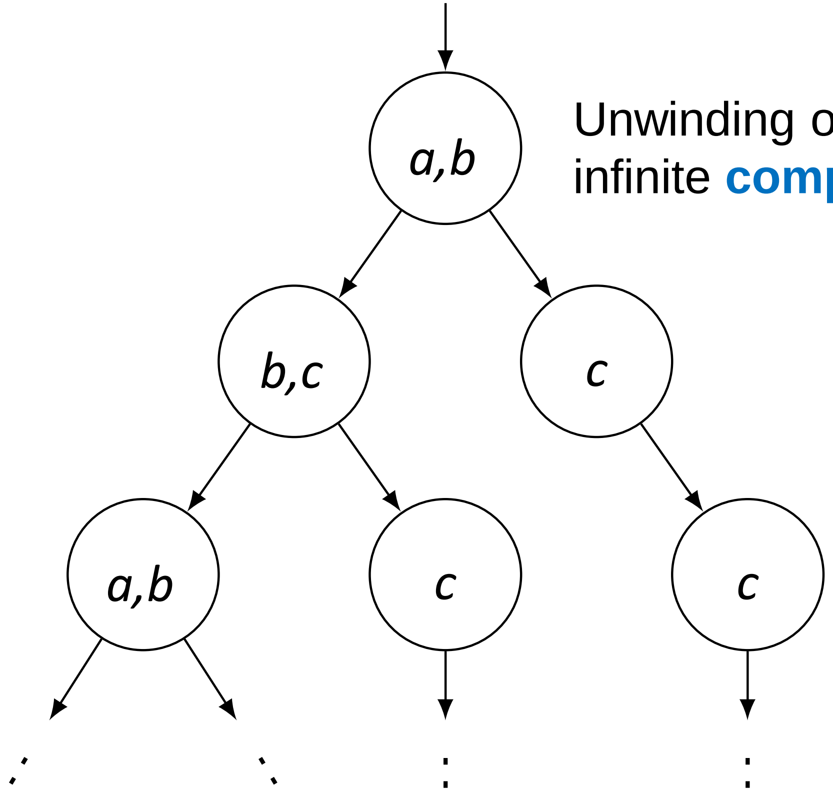
Computation Tree Logic - CTL*

- Defines properties of computation trees of Kripke structures

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

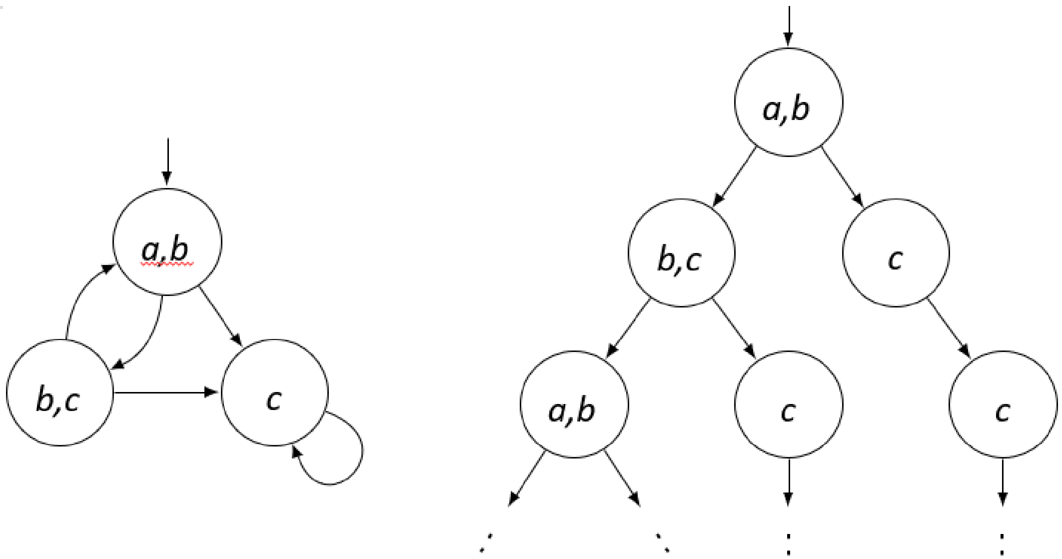


Unwinding of M into infinite **computation tree**



Paths and Suffixes

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



Propositional Temporal Logic

Temporal operators:

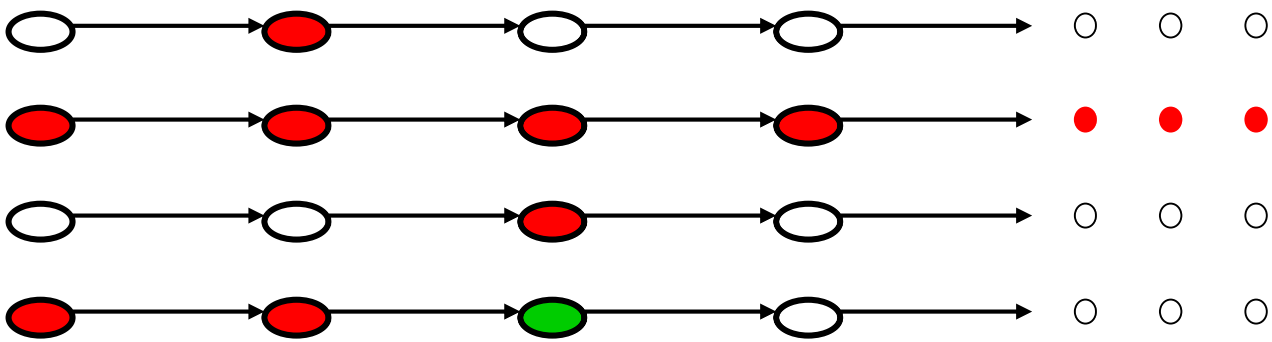
- Describe properties that hold along an infinite path π

Xp

Gp

Fp

p**U**q



p**U**q holds if there is a state on π where q holds,
and at every preceding state on π (if it exists), p holds

Propositional Temporal Logic

Temporal operators:

- Describe properties that hold along an infinite path π

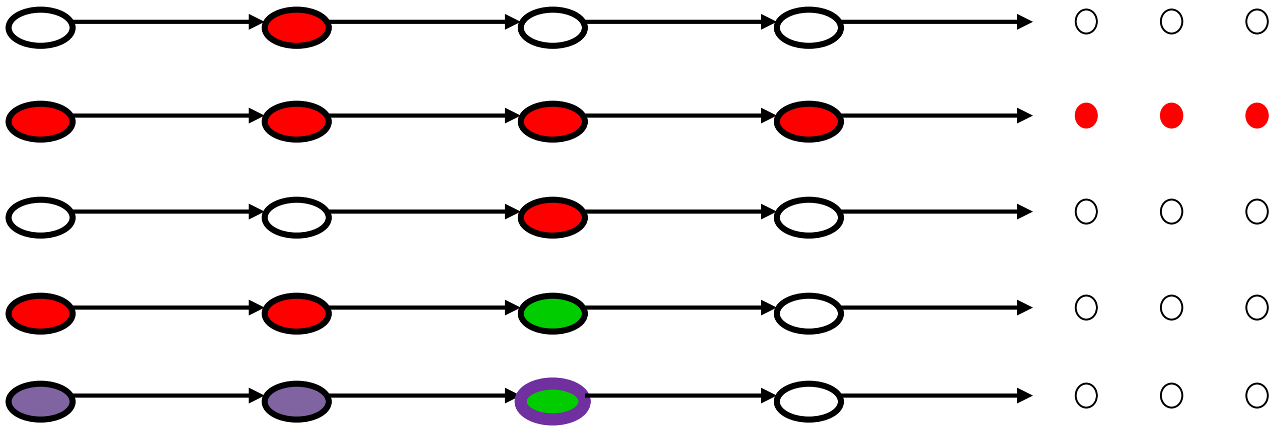
Xp

Gp

Fp

pUq

pRq



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

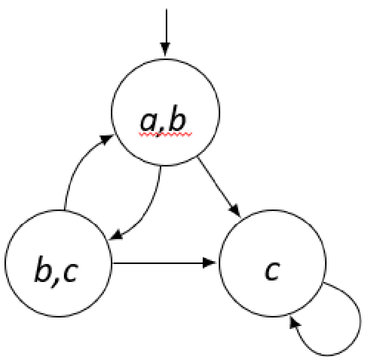
Propositional Temporal Logic

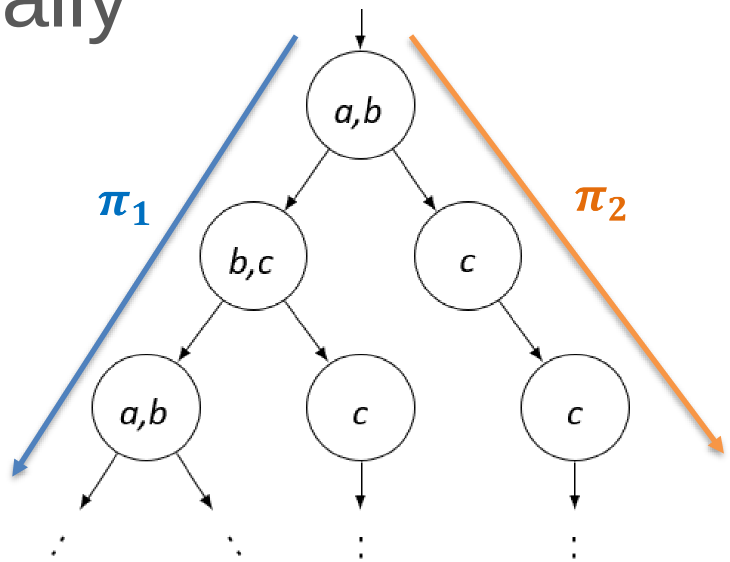
Path quantifiers: A, E

- Are used in a particular state s.
- They specify that all of the paths or some of the paths starting from s have property φ

- **A** for **all** paths starting from **s** have property φ
- **E** there **exists** a path starting from **s** have property φ
- Use combination of **A and E** to describe branching structure in tree

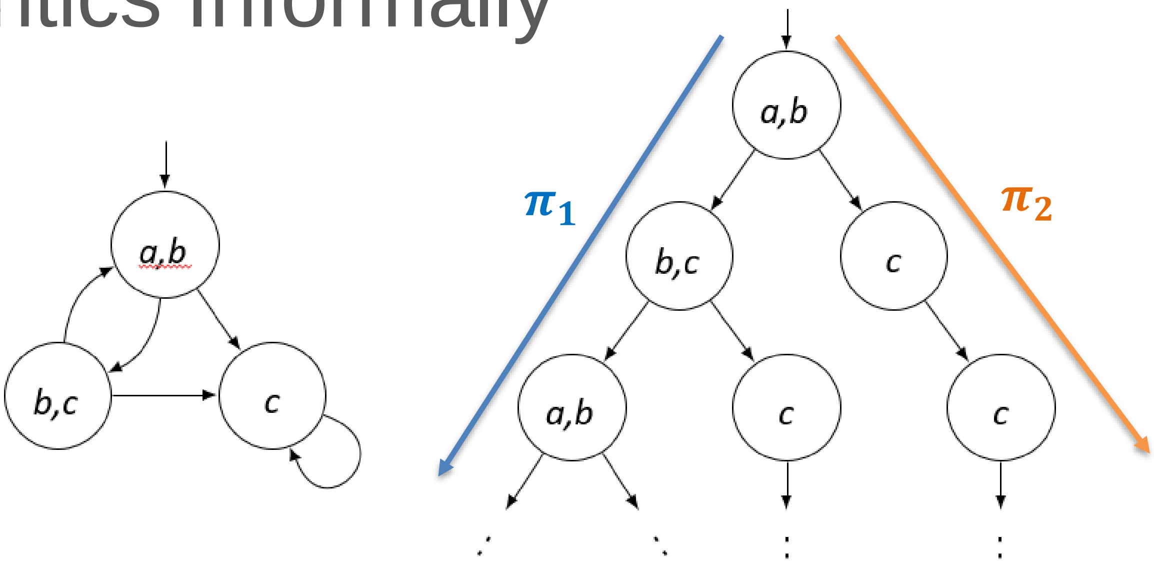
State Formulas and Path Formulas Semantics Informally





- Path Formulas:
  - $\pi_1 \vDash \mathrm{G}\mathrm{b}$
  - $\pi_2 \nvDash \mathrm{G}\mathrm{b}$
- State Formulas:
  - $s_0 \vDash \mathrm{EG}\ \mathrm{b}$
  - $s_0 \nvDash \mathrm{AG}\ \mathrm{b}$

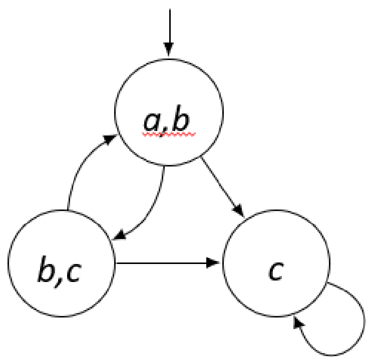
State Formulas and Path Formulas Semantics Informally

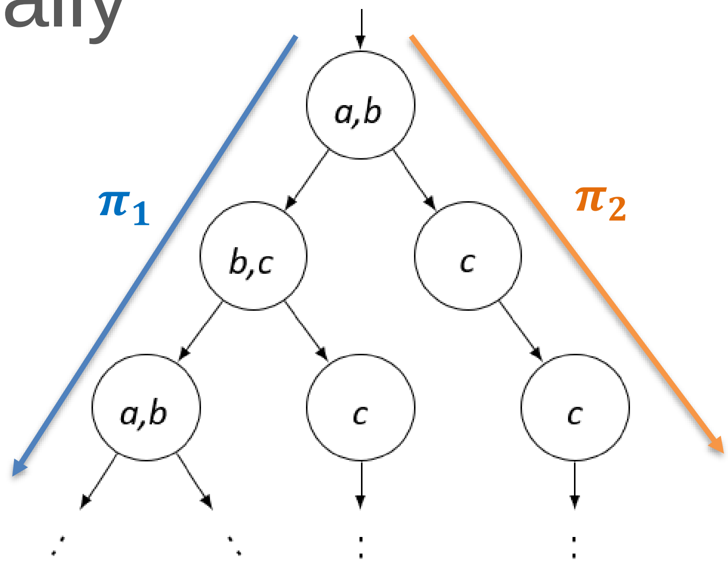


ToDo Does s_0 satisfy the following formula?

- s_0 ☐ EXX $(a \wedge b)$
- s_0 ☐ EXAX $(a \wedge b)$

State Formulas and Path Formulas Semantics Informally





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

SCOS Secure & Correct Systems

Syntax of CTL*

Two types of formulas in the inductive definition

- State formulas
- Path formulas

CTL* formulas are the set of all state formulas

Syntax of CTL*: State Formulas

State formulas are true in a specific state

Inductive definition of state formulas:

- $p \in AP$ is a state formula
- $\neg f_1, f_1 \vee f_2, f_1 \wedge f_2$ where f_1, f_2 are state formulas

Syntax of CTL*: State Formulas

State formulas are true in a specific state

Inductive definition of state formulas:

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

Syntax of CTL*: Path Formulas

Path formulas are true along a specific path

Inductive definition of path formulas:

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

- CTL* is the set of all state formulas = CTL* formulas are Boolean variables, temporal properties with a leading path quantifier, and Boolean combinations thereof.

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

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

Semantics of CTL*

- g is a path formula

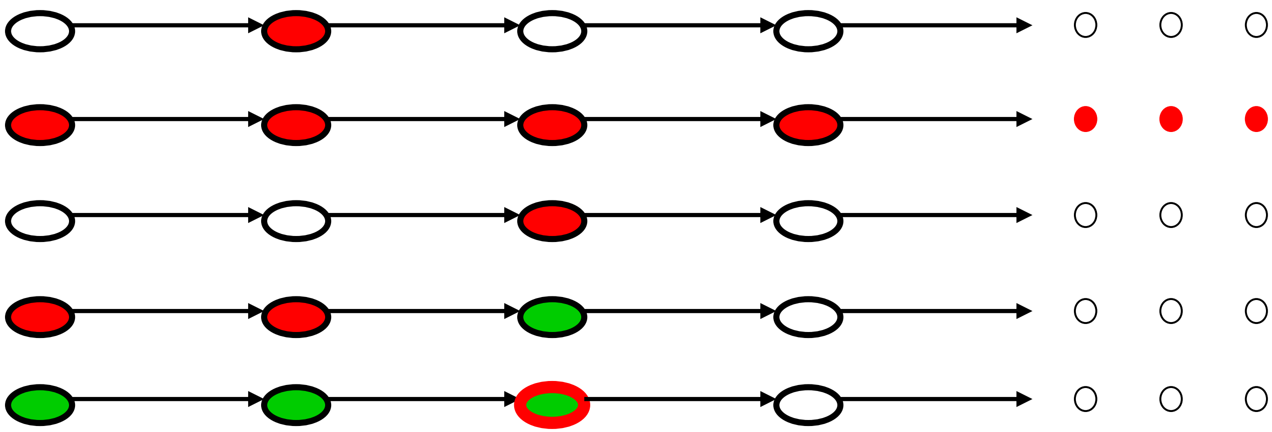
State formulas:

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

Semantics of path formulas - summary

If p,q are state formulas, then:

Xp
Gp
Fp
pUq
pRq

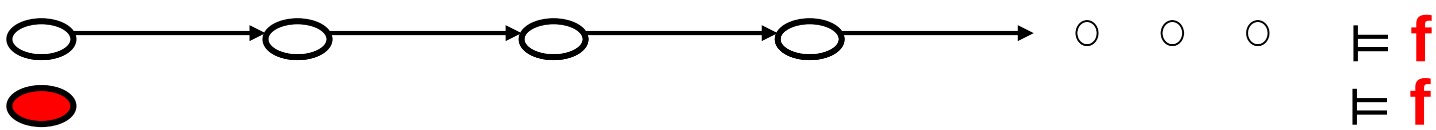


But in the general case p and q can be path formulas

Semantics of CTL*

Path formulas:

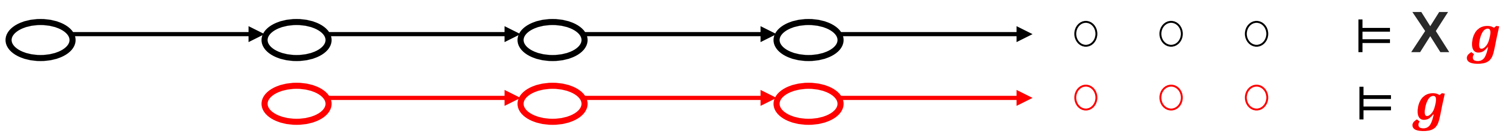
- $M, s \models$ f, where f is a state formula $\Leftrightarrow$ $M, s_0 \models$ f



Semantics of CTL*

Path formulas:

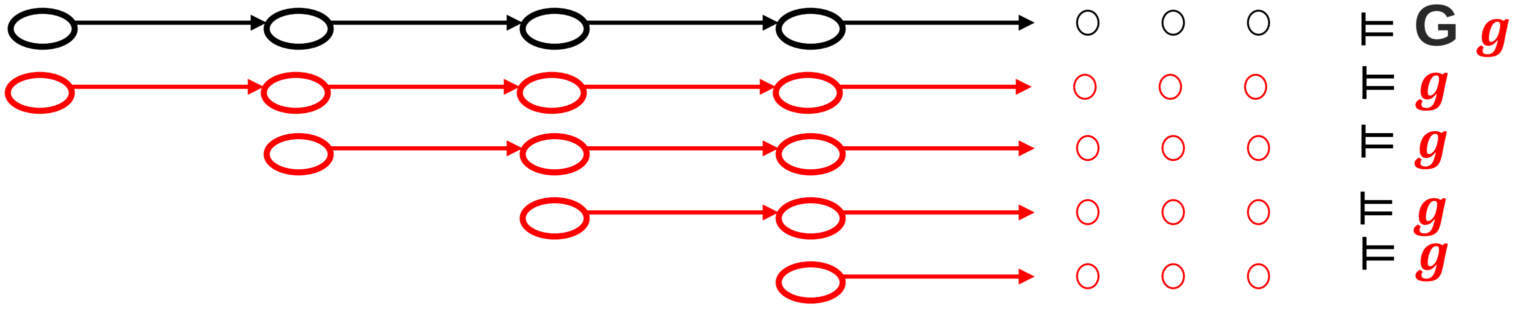
- $M, \pi \models \mathbf{X}\ g$, where g is a path formula $\Leftrightarrow$ $M, \pi^1 \models g$



Semantics of CTL*

Path formulas:

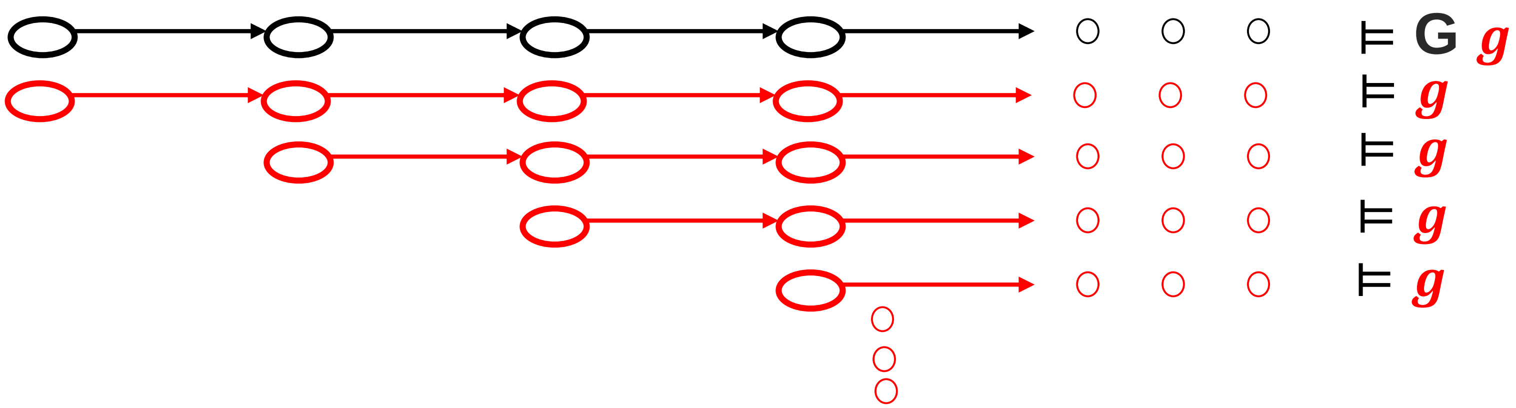
- $M, \pi \vDash \mathbf{G}g \Leftrightarrow$ for every i ≥0, $M, \pi^i \vDash g$



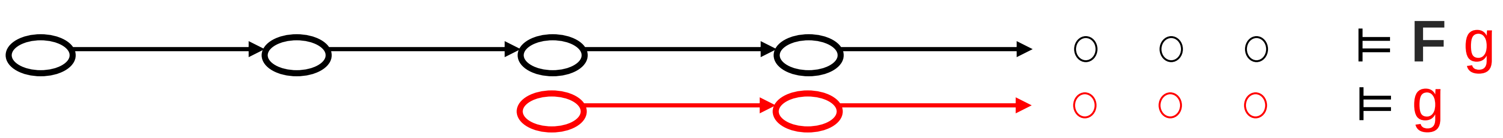
Semantics of CTL*

Path formulas:

- $M, \pi \vDash \mathbf{G}g \Leftrightarrow$ for every i ≥0, $M, \pi^i \vDash g$



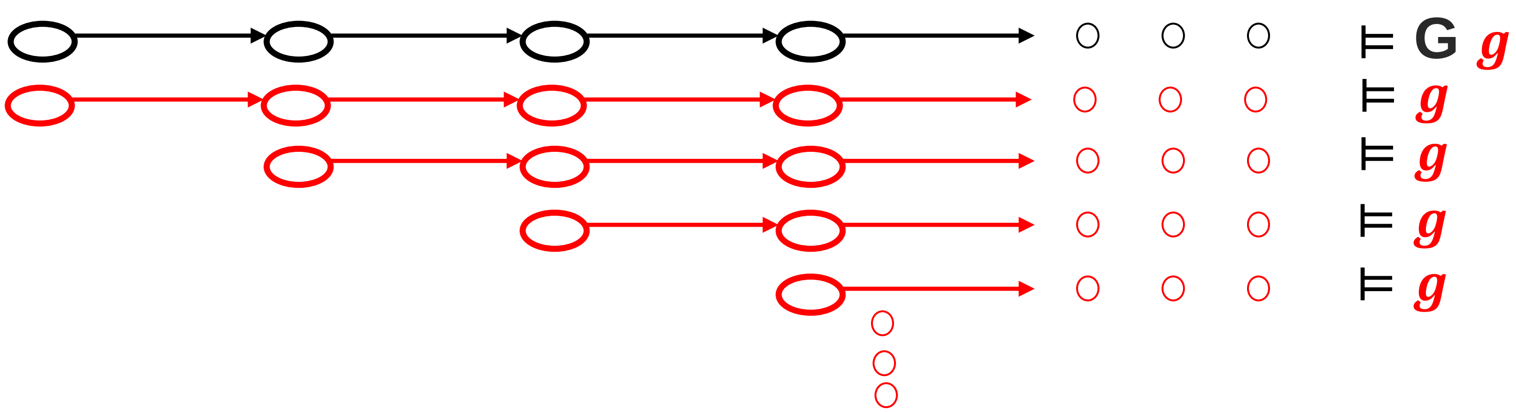
- $M, \pi \vDash \mathbf{F}g \Leftrightarrow$



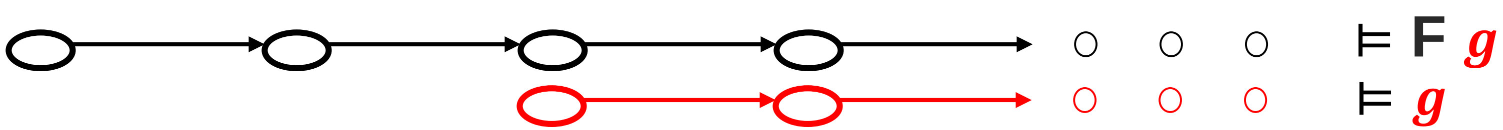
Semantics of CTL*

Path formulas:

- $M, \pi \models \mathbf{G}g \Leftrightarrow$ for every i ≥0, $M, \pi^{i} \models g$



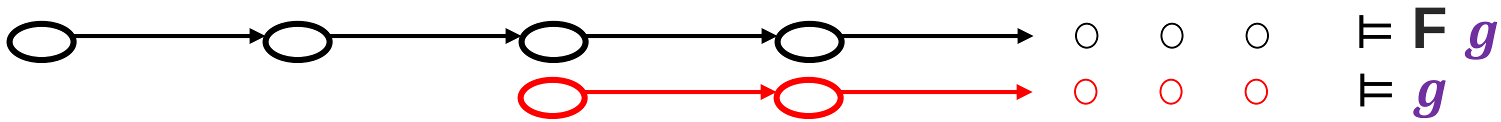
- $M, \pi \models \mathbf{F}g \Leftrightarrow$ there exists k ≥0, such that $M, \pi^{k} \models g$



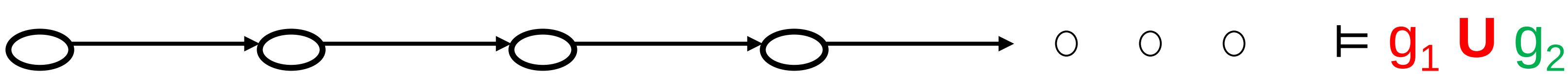
Semantics of CTL*

Path formulas:

- $M, \pi \vDash \mathbf{F}\ g \Leftrightarrow$ there exists k ≥0, such that M, $\pi^k \vDash g$



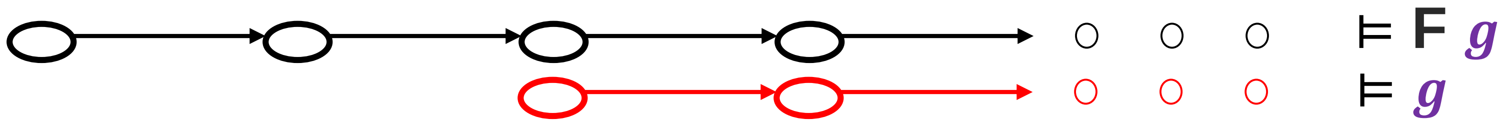
- $M, \pi \vDash g_1\ \mathbf{U}\ g_2 \Leftrightarrow$



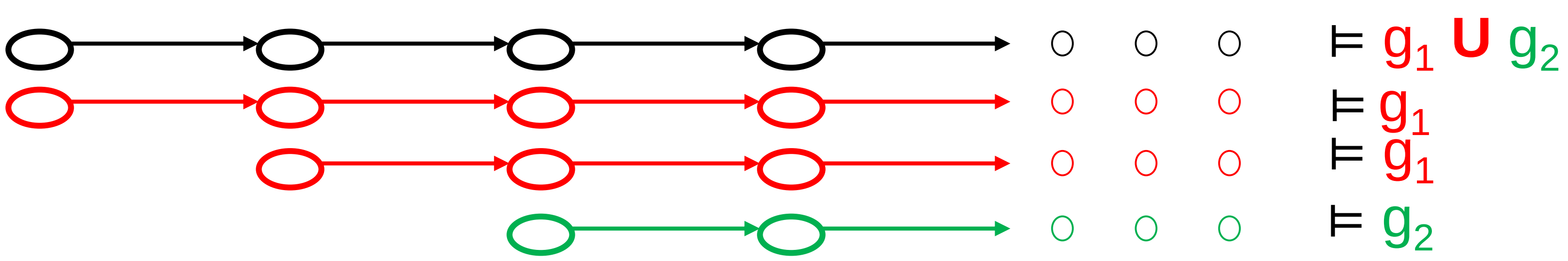
Semantics of CTL*

Path formulas:

- $M, \pi \vDash \mathbf{F}\ g \Leftrightarrow$ there exists k ≥0, such that M, $\pi^k \vDash g$



- $M, \pi \vDash g_1\ \mathbf{U}\ g_2 \Leftrightarrow$ there exists k ≥0, such that M,$\pi^k \vDash g_2$
  and for every 0 ≤j<k, M, $\pi^j \vDash g_1$



R („release“)

- 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 U, F, G, or X

- g_1 **R** g_2 $\equiv$

R („release“)

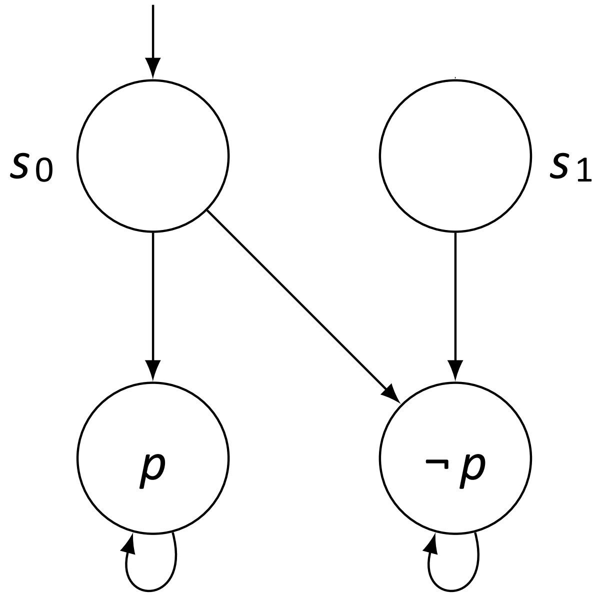
- Intuitively, once g_1 becomes true, it “releases” g_2.
  If g_1 never becomes true then g_2 stays true forever

- $g_1 \mathbf{R}\, g_2 \equiv (g_2\ \mathbf{U}\ (g_1 \wedge g_2)) \vee \mathbf{G}\ g_2$

Semantics of CTL*

ToDo

- $M \models f \Leftrightarrow$ for all initial states $s_0 \in S_0$: $M, s_0 \models f$
- Example: Does $M \models \mathrm{EX}\ p$ or $M \models \neg \mathrm{EX}\ p$?

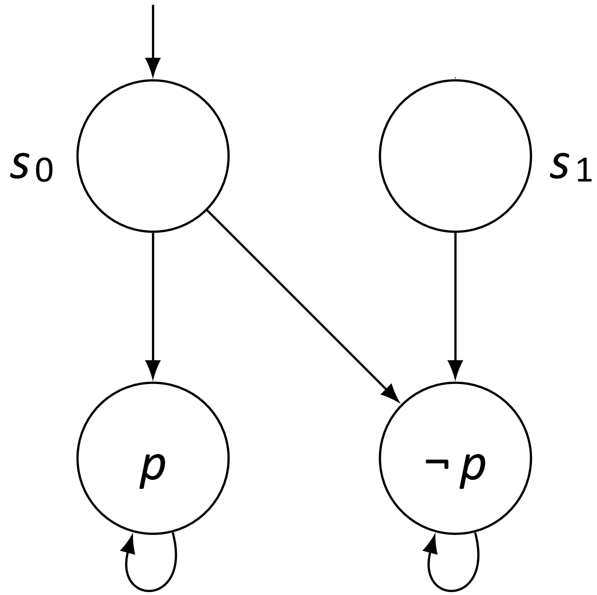


Semantics of CTL*

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

ToDo

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

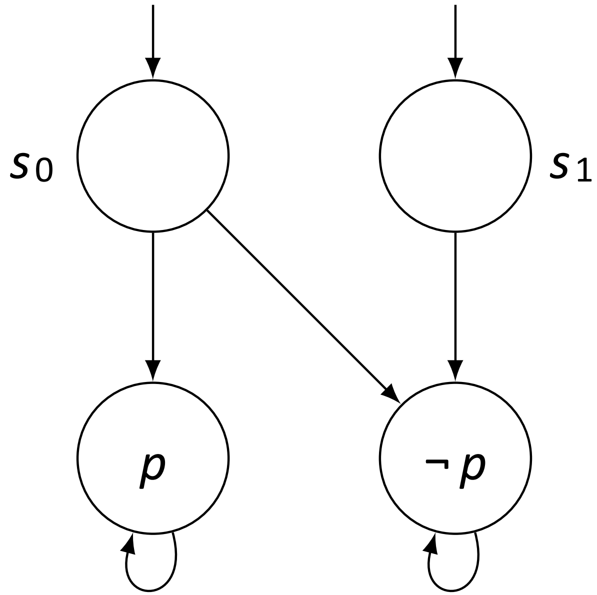


Solution

$M \vDash \text{EX}\ p$

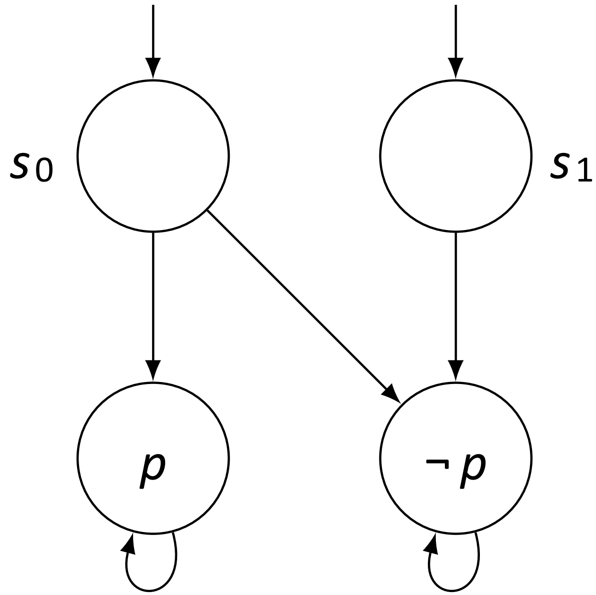
Semantics of CTL*

- $M \vDash f \Leftrightarrow$ for all initial states $s_0 \in S_0$: $M, s_0 \vDash f$
- Example: Does $M \vDash \mathrm{EX}\ p$ or $M \vDash \neg\mathrm{EX}\ p$?



Semantics of CTL*

- $M \vDash f \Leftrightarrow$ for all initial states $s_0 \in S_0$: $M, s_0 \vDash f$
- Example: Does $M \vDash \text{EX}\, p$ or $M \vDash \neg\text{EX}\, p$?



Neither

Holds in s_1 but not in s_0.
Note, such a situation never happens when M has a single initial state.

Exercise 1

Question:

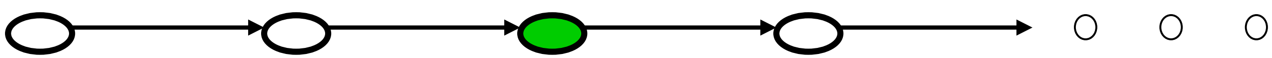
- Given a, b ∈ AP
  How does a path satisfying **F(**a **U** b**)** look like?

Exercise 1

Question:

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

F (a **U** b**)**



Exercise 2

Question:

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

- $\pi \models$ **GF** p
- $\pi \models$ **FG** p

Exercise 2

Question:

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

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

Exercise 2

Question:

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

- $s \models$ **EGF** p
- $s \models$ **EG EF** p

- $\pi \models$ **GF** p   Infinitely often p along π
- $\pi \models$ **FG** p   Finitely often ¬p along π

Exercise 2

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

- $\pi \models$ **GF** p Infinitely often p along π
- $\pi \models$ **FG** p Finitely often ¬p along π

Exercise 3

Question:

When does π satisfy the formula:
(Formulate it without an Until operator)

- $\pi \models (\mathbf{G}a)\ \mathbf{U}\ (\mathbf{G}b)$

Answer:

Exercise 3

Question:

When does π satisfy the formula:
(Formulate it without an Until operator)

- $\pi \models (\mathbf{G}a)\ \mathbf{U}\ (\mathbf{G}b)$

Answer:

- $(\mathbf{G}a)\ \mathbf{U}\ (\mathbf{G}b) \equiv \mathbf{G}b \lor (\mathbf{G}a \land \mathbf{FG}b)$

Properties of CTL*

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

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$
- $f\ \mathbf{R}\ g \equiv \neg(\neg f\ \mathbf{U}\ \neg g)$
- $\mathbf{F}\ f \equiv true\ \mathbf{U}\ f$
- $\mathbf{G}\ f \equiv \neg\ \mathbf{F}\ \neg f$
- $\mathbf{A}\ (f) \equiv \neg\ \mathbf{E}\ (\neg f)$

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

Negation Normal Form (NNF)

- Negations can be "pushed" inwards.
  $\neg\, \mathbf{E}\, f \;\equiv\; \mathbf{A}\, \neg f$
  $\neg\, \mathbf{G}\, f \;\equiv\; \mathbf{F}\, \neg f$
  $\neg\, \mathbf{X}\, f \;\equiv\; \mathbf{X}\, \neg f$
  $\neg\, (\, f\, \mathbf{U}\, g\,) \equiv (\, \neg f\, \mathbf{R}\, \neg g\,)$

- Example:
  Transforming a formula into NNF:

- $\neg(\, (a\, \mathbf{U}\, b\,) \vee \mathbf{F}\, c) \equiv$

Negation Normal Form (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:
  Transforming a formula into NNF:

- $\neg(\ (a\ \mathbf{U}\ b\ ) \vee \mathbf{F}\ c) \equiv (\neg(a\ \mathbf{U}\ b) \wedge \neg\mathbf{F}\ c) \equiv$
  $(((\neg a)\ \mathbf{R}\ (\neg b)) \wedge (\mathbf{G}\ \neg c)$

SCOS
Secure & Correct Systems

Useful sublogics of CTL*

- CTL are Computation tree logic
  - Can describe the branching of the computation tree by applying nested path quantifications
- LTL is a linear-time temporal logic
  - Describes the paths in the computation tree, using only **one, outermost universal quantification**

- CTL and LTL are most widely used

LTL/CTL/CTL*

LTL consists of state formulas of the form **A** f

- f is a path formula, containing no path quantifiers
- LTL is interpreted over infinite computation paths

CTL consists of state formulas, where path quantifiers and temporal operators appear in pairs:

- **AG**, **AU**, **AX**, **AF**, **AR**, **EG**, **EU**, **EX**, **EF**, **ER**
- CTL is interpreted over infinite computation trees

CTL* allows any combination of temporal operators and path quantifiers. It includes both LTL and CTL

LTL

State formulas:

- **A**f where f is a path formula

Path formulas:

- $p \in AP$
- $\neg f_1$, $f_1 \vee f_2$, $f_1 \wedge f_2$, $\mathbf{X} f_1$, $\mathbf{G} f_1$, $\mathbf{F} f_1$, $f_1 \mathbf{U} f_2$, $f_1 \mathbf{R} f_2$

  where f_1 and f_2 are path formulas

LTL is the set of all state formulas

CTL

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

- $p \in AP$
- $\neg g_1, \; g_1 \vee g_2, \; g_1 \wedge g_2$
- **AX** g_1, **AG** g_1, **AF** g_1, **A** $(g_1$ **U** $g_2)$, **A** $(g_1$ **R** $g_2)$
- **EX** g_1, **EG** g_1, **EF** g_1, **E** $(g_1$ **U** $g_2)$, **E** $(g_1$ **R** $g_2)$

where g_1 and g_2 are state formulas


THANK
YOU!