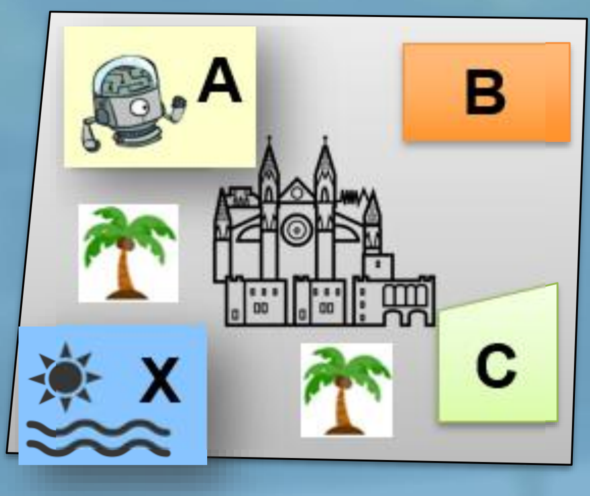
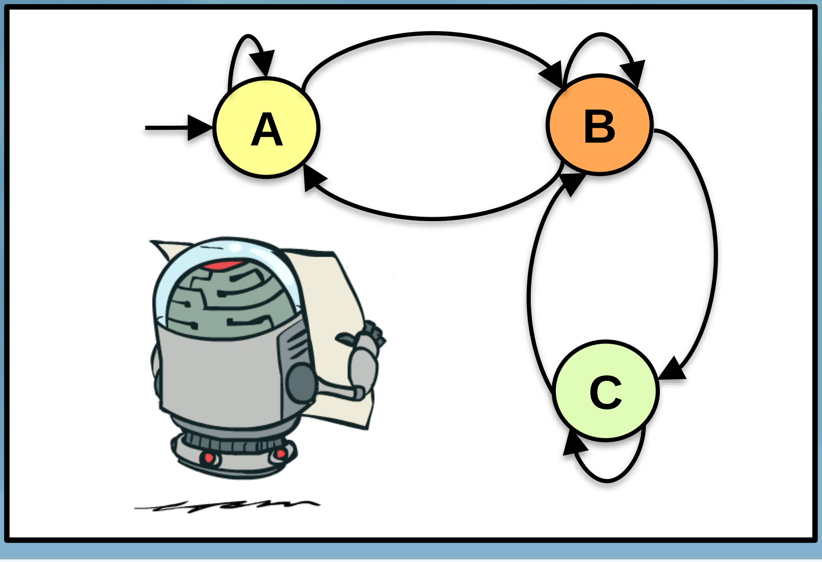
Graz University of Technology
Institute for Applied Information Processing and Communications

Temporal Logic

Bettina Könighofer





Model Checking SS21

April 22nd 2021


ToDo


PLEASE INTERRUPT ME


"TALK TO ME"


WHERE
WHAT
WHEN
WHY
HOW
WHO

Warm Up

Translate sentences to formulas

- "If today is Tuesday, tomorrow is Wednesday."

- "This lecture is exciting and not boring."

Warm Up

Translate sentences to formulas

- "If today is Thursday, then tomorrow is Friday."

  p... today is Tuesday, q… tomorrow is Wednesday

$$p \rightarrow q$$

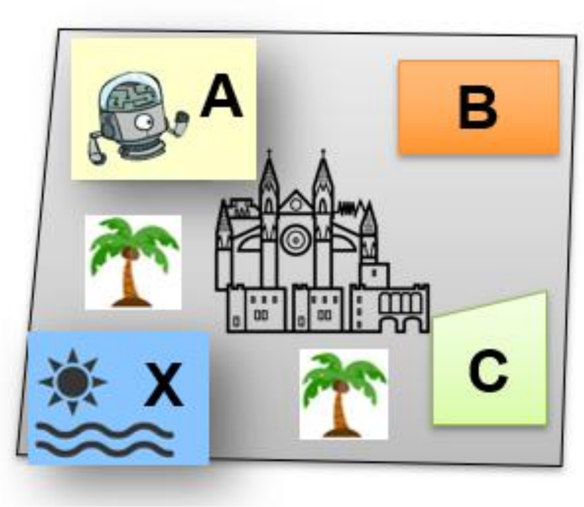
- "This lecture is exciting and not boring."

  p... This lecture is exciting ,
  q… This lecture is boring

$$p \land \neg q$$

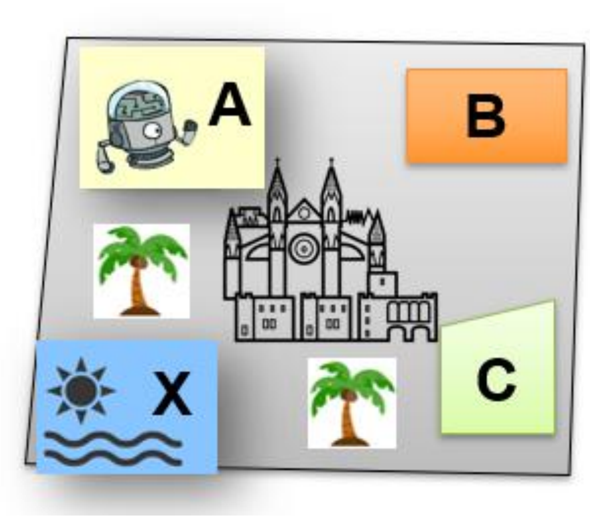
Modeling a reactive system

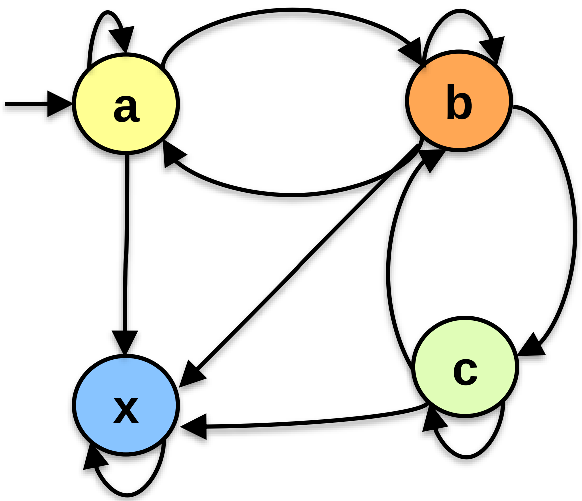
Kripke structure



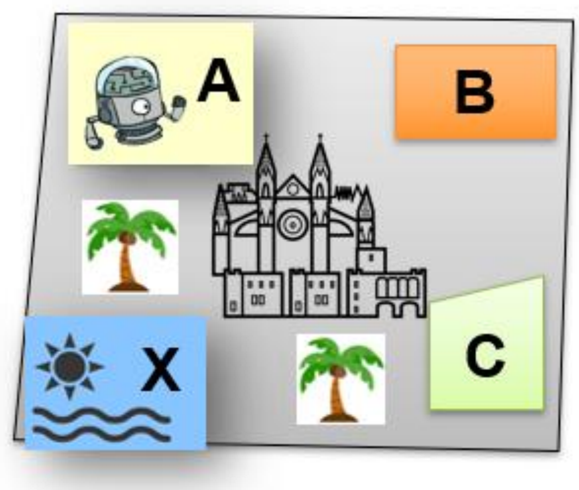
Modeling a reactive system

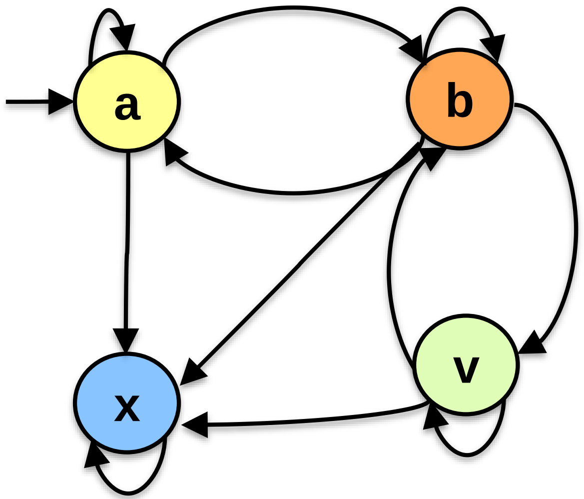
Kripke structure





Properties of Kripke Structures





Properties

ToDo *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

SCOS Secure & Correct Systems

Propositional Temporal Logic

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

Temporal operators:

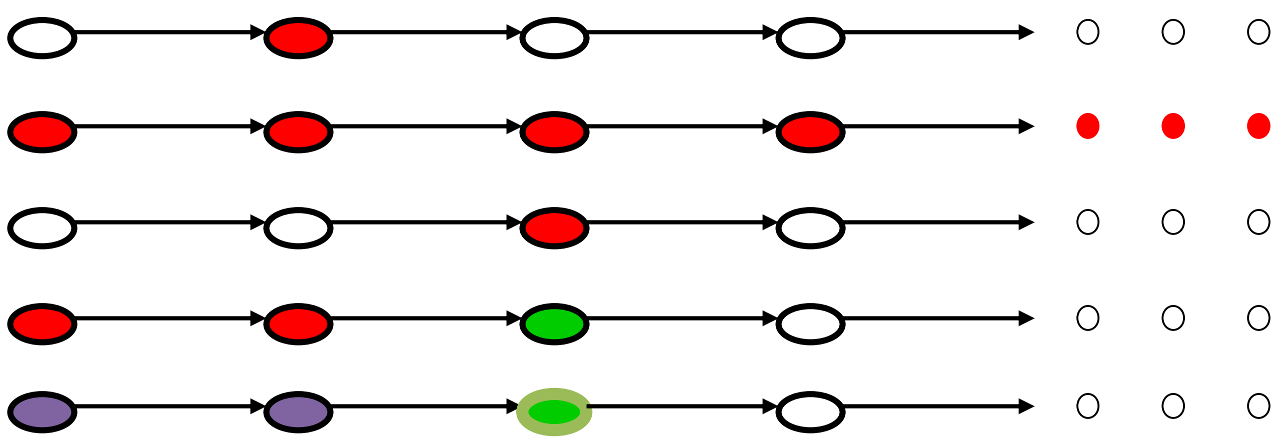
Xp

Gp

Fp

pUq

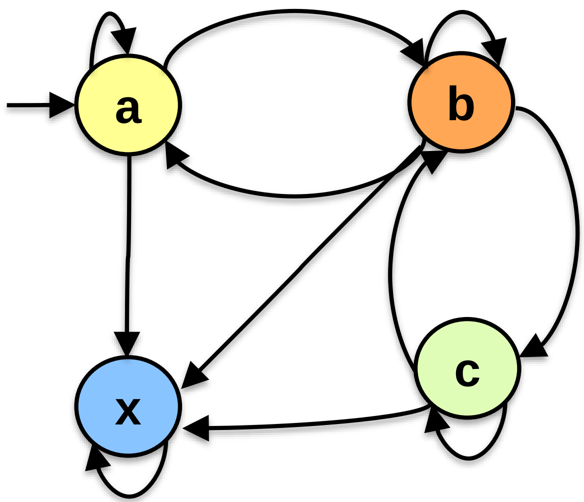
pRq

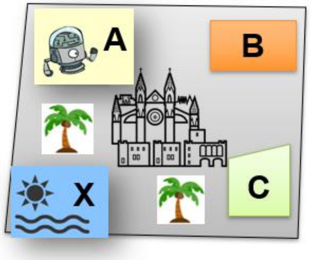


Path quantifiers: **A** for **all** paths

E there **exists** a path

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

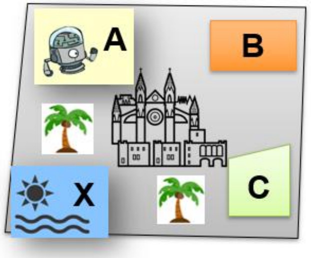
Path quantifiers
A for **all** paths
E there **exists** a path

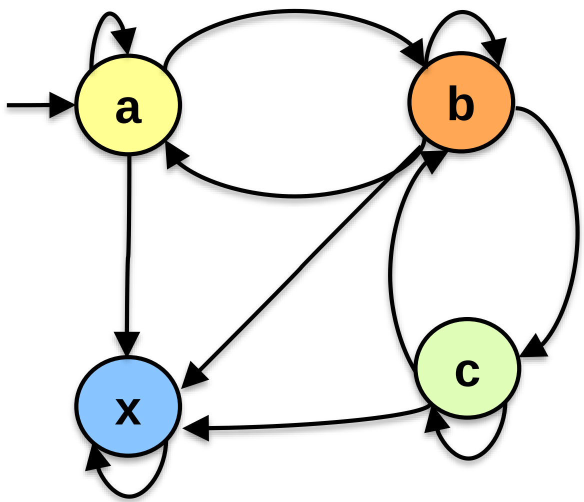
Properties

ToDo *Write properties as formulas*

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

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

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

Properties

Write properties as formulas

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

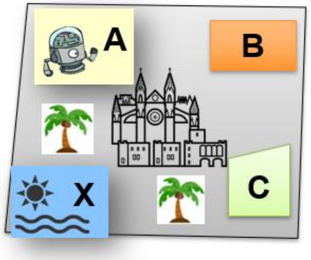
$$A\ G\ (a \rightarrow Xc \vee XXc)$$

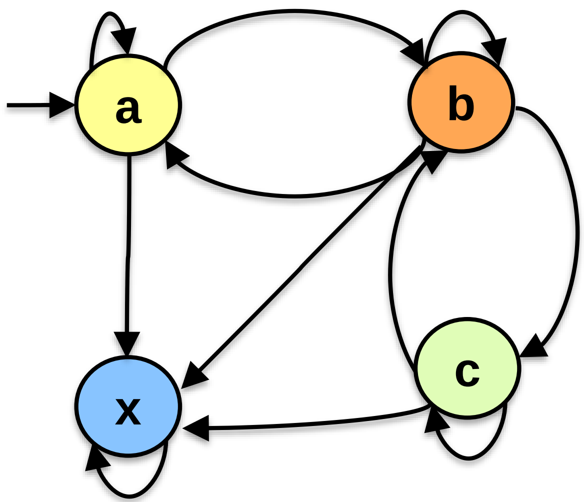
- The robot can visit C within the next two steps after visiting A

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

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

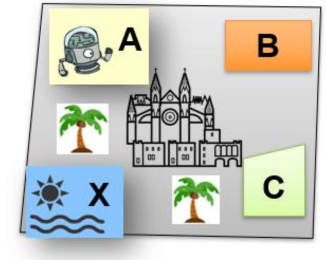
Properties

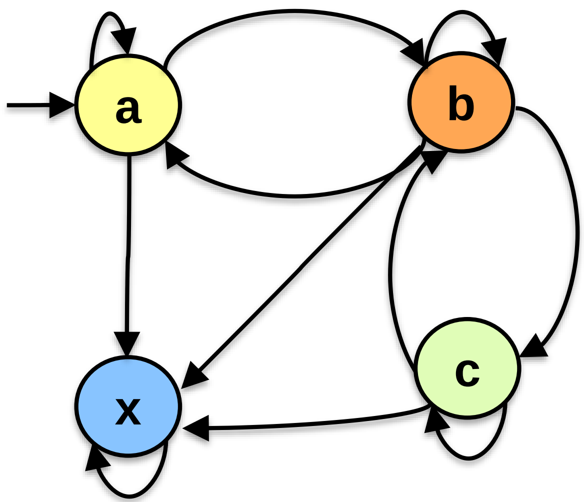
ToDo *Write properties as formulas*

- The robot *never* visits **X**
- It is possible that the robot *never* visits **X**

SCOS
Secure & Correct Systems

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

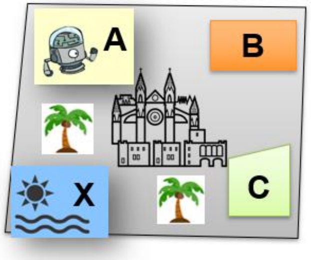
Path quantifiers
A for **all** paths
E there **exists** a path

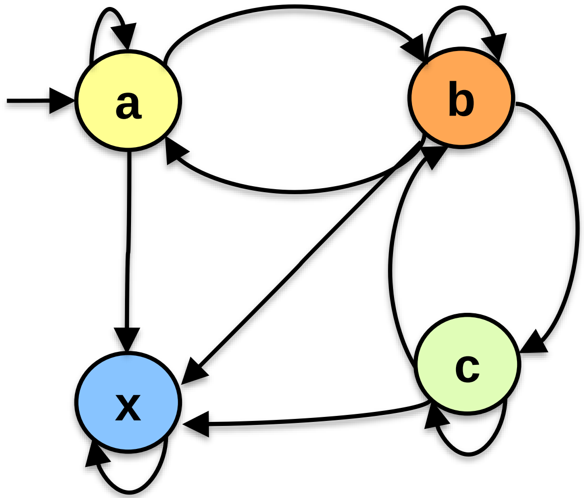
Properties

Write properties as formulas

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

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

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

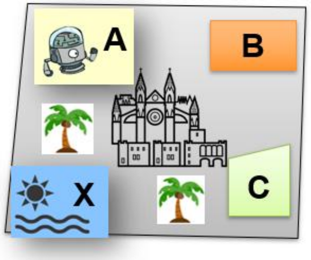
Properties

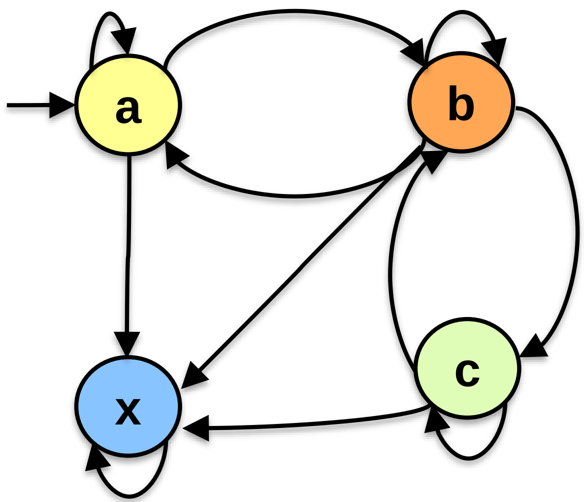
ToDo *Write properties as formulas*

- The robot can visit A and C *infinitely often.*
- The robot always visits A *infinitely often*, but C only *finitely often*.

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

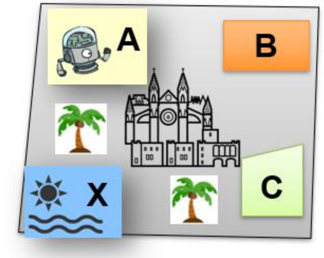
Properties

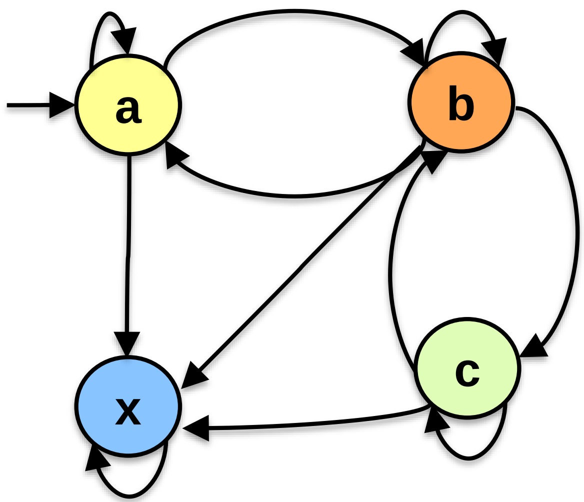
Write properties as formulas

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

Secure & Correct Systems

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

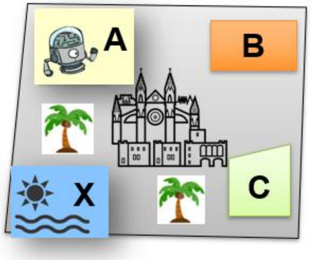
Path quantifiers
A for **all** paths
E there **exists** a path

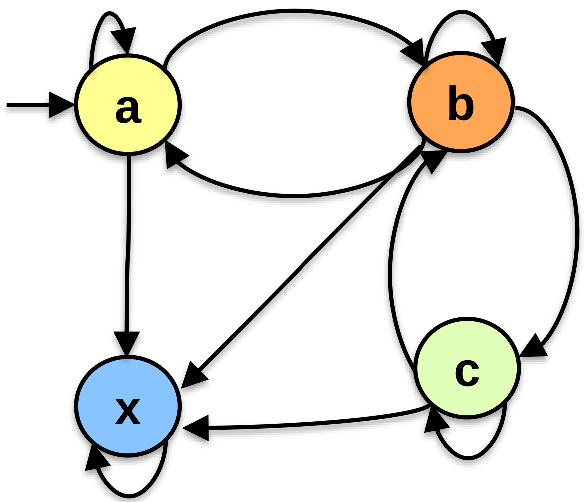
Properties

Write properties as formulas

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

Properties of Kripke Structures





Temporal Operators
X… next
G… globally
F… eventually

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

Properties

Write properties as formulas

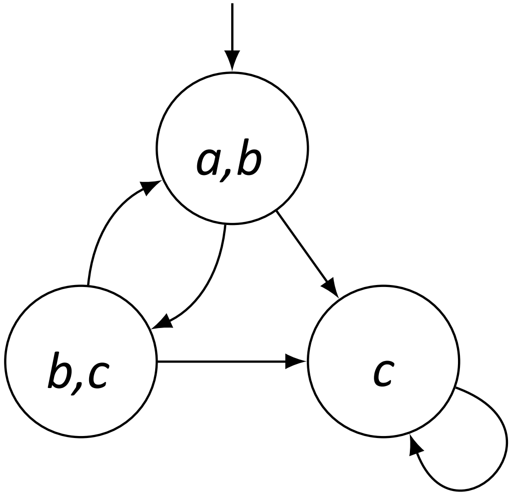
- If the robot visits **A** *infinitely often,* it should also visit **C** *finitely often*.

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

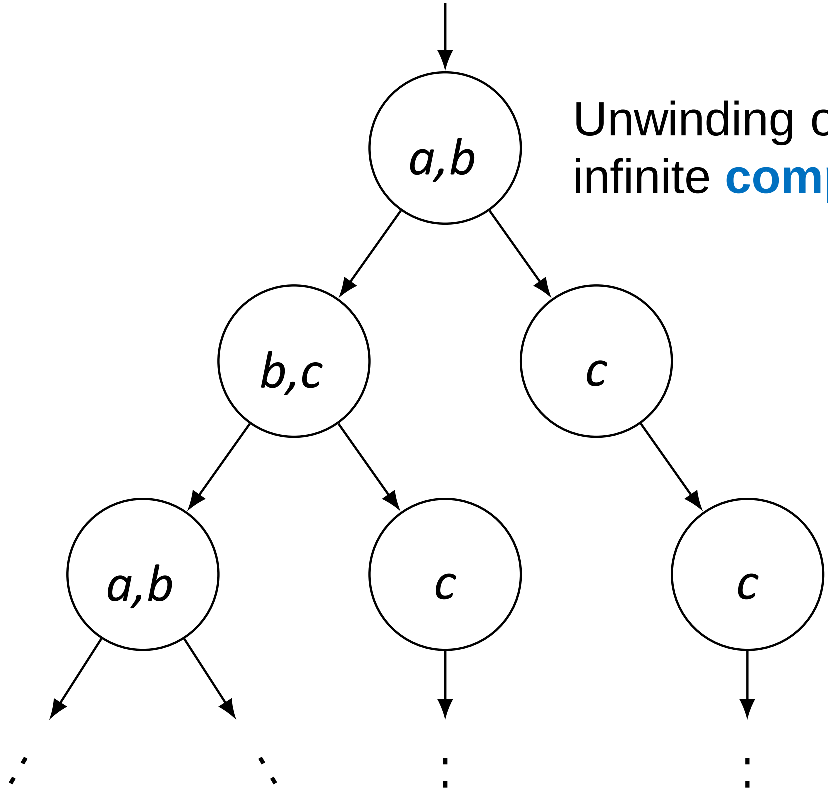
Computation Tree Logic - CTL*

- Defines properties of computation trees of Kripke structures

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

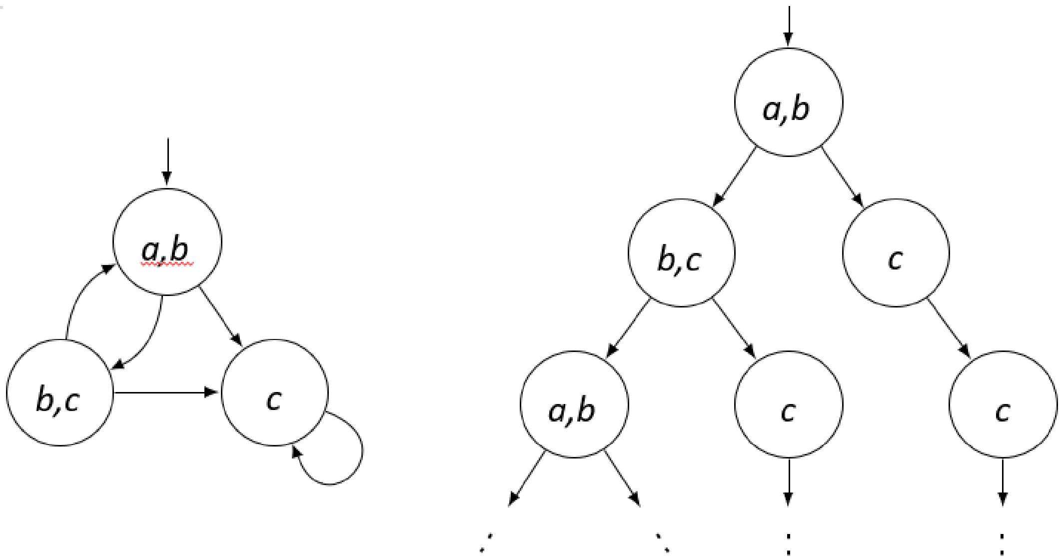


Unwinding of M into
infinite **computation tree**



Paths and Suffixes

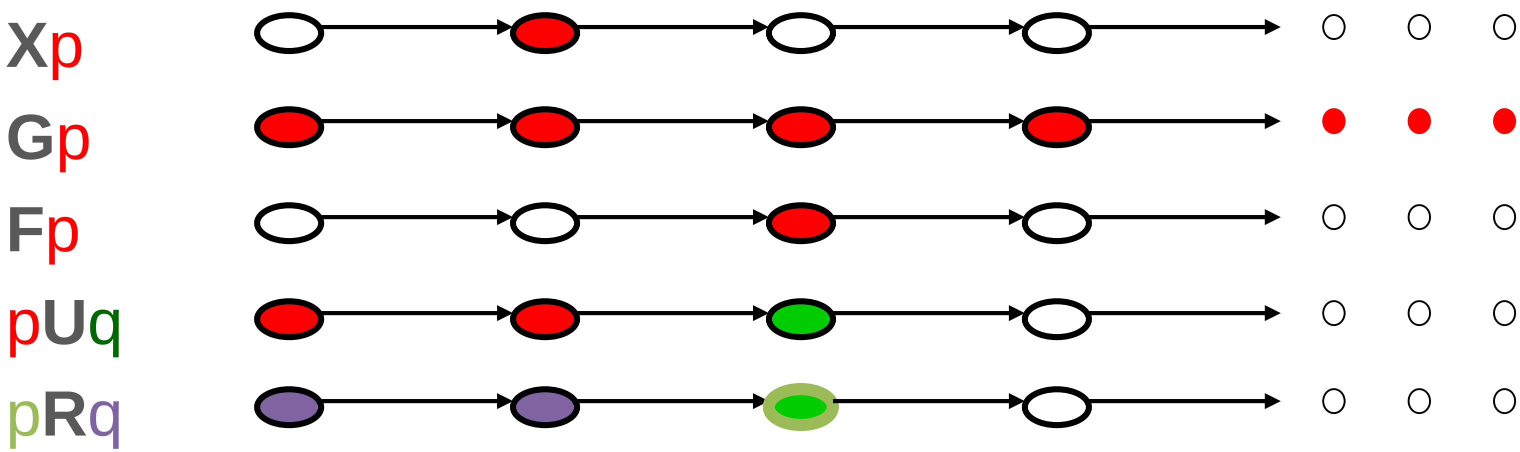
- $\pi = s_0, s_1, \ldots$ is an *infinite* path in M from a state s if
  - $s = s_0$ and
  - for all $i \geq 0$, $(s_i, s_{i+1}) \in R$



Propositional Temporal Logic

Temporal operators:

- Describe properties that hold along π

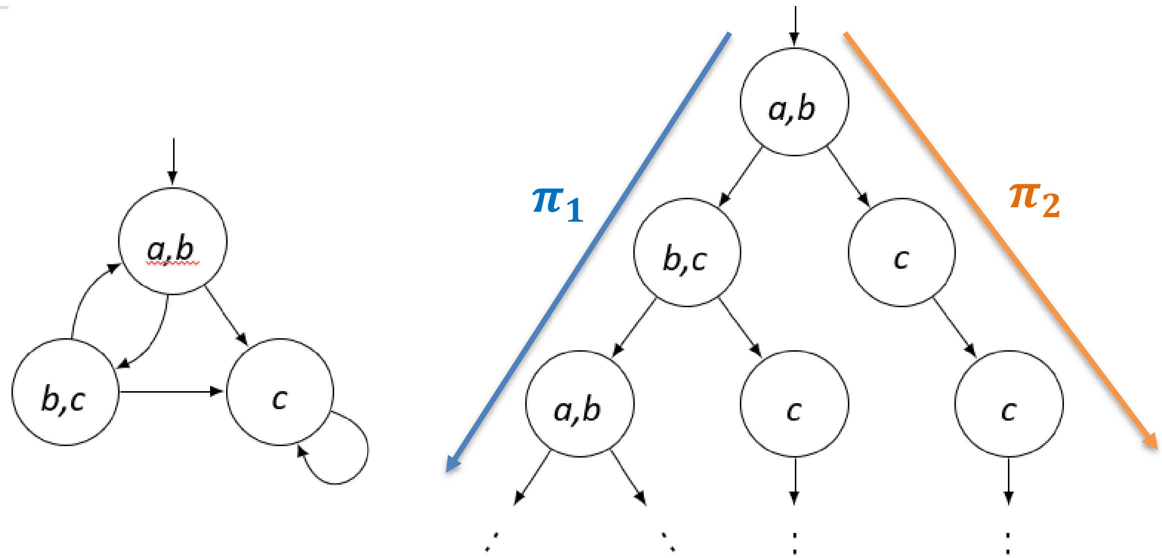


Propositional Temporal Logic

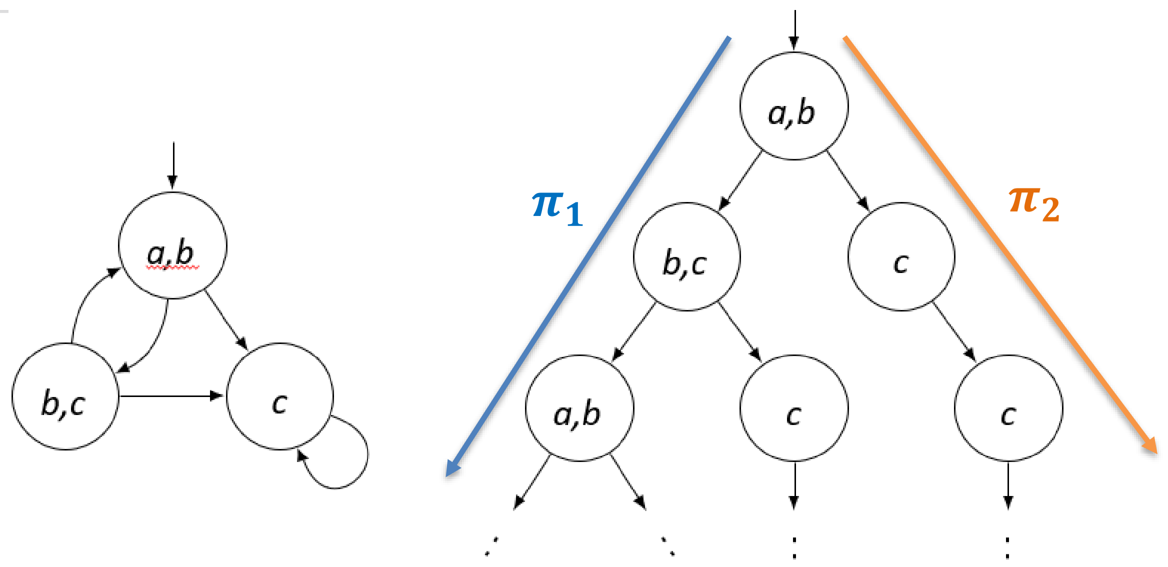
Path quantifiers:

- **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

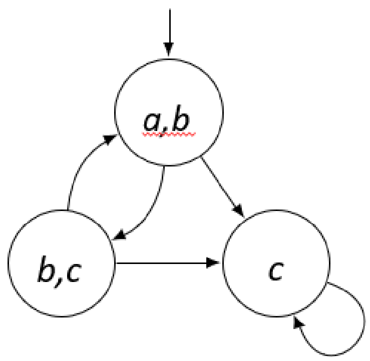


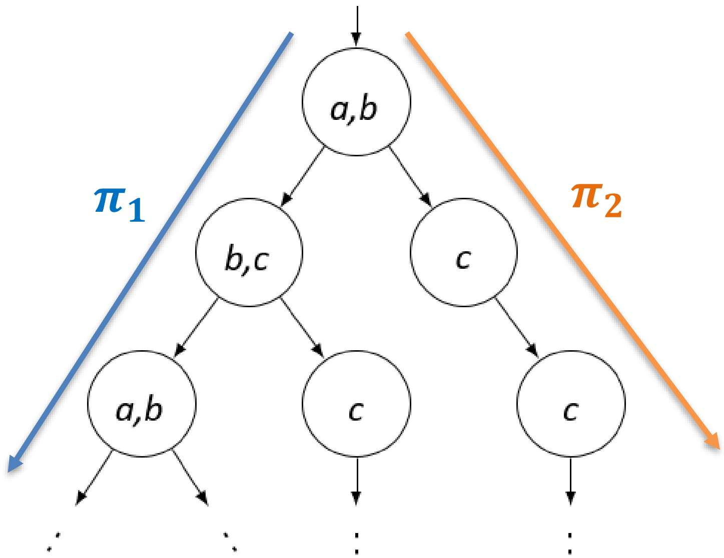
State Formulas and Path Formulas



- Path Formulas:
  - $\pi_1 \models \mathrm{Gb}$
  - $\pi_2 \not\models \mathrm{Gb}$

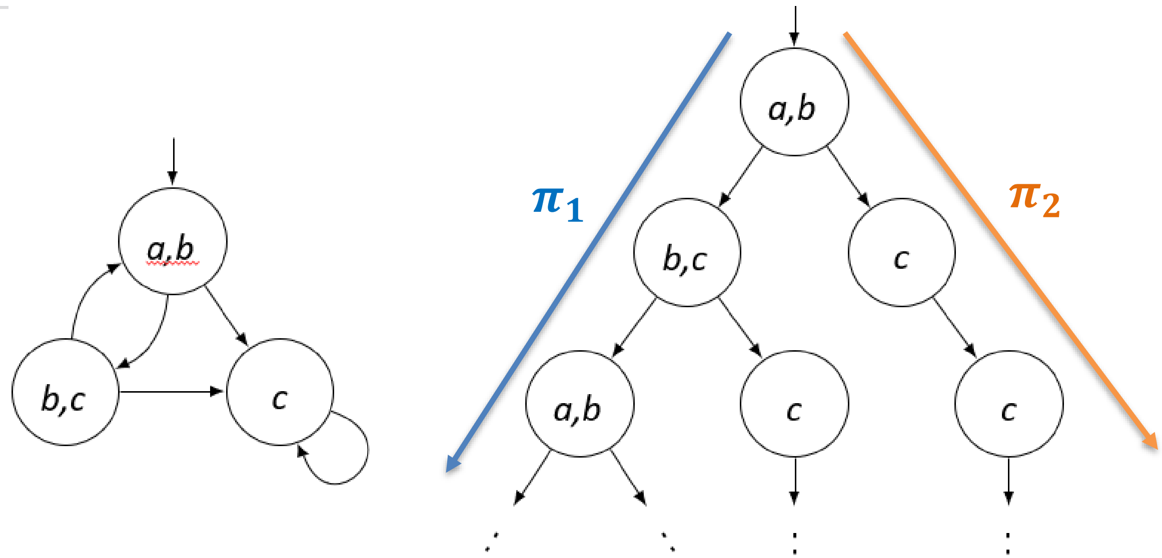
State Formulas and Path Formulas





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

State Formulas and Path Formulas

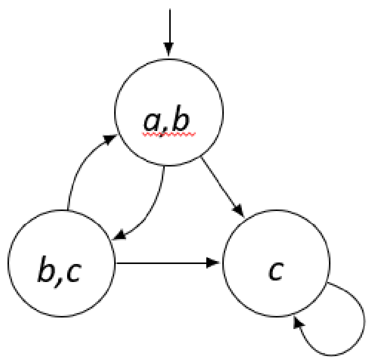


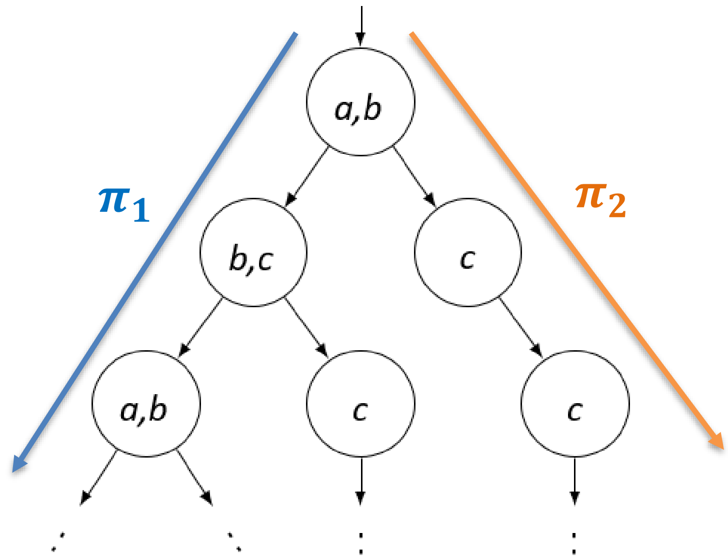
ToDo

Does s_0 satisfy the following formula?

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

State Formulas and Path Formulas





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

Syntax of CTL*

Two types of formulas in the inductive definition

- State formulas
- Path formulas

Syntax of CTL*: State Formulas

State formulas are true in a specific state

Syntax of CTL*: State Formulas

State formulas are true in a specific state

Inductive definition of state formulas:

Syntax of CTL*: State Formulas

State formulas are true in a specific state

Inductive definition of state formulas:

- $p \in AP$

Syntax of CTL*: State Formulas

State formulas are true in a specific state

Inductive definition of state formulas:

- $p \in AP$
- $\neg f_1, f_1 \vee f_2, f_1 \wedge f_2$ 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:

- $p \in AP$
- $\neg f_1, f_1 \vee f_2, f_1 \wedge f_2$ 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

Syntax of CTL*: Path Formulas

Path formulas are true along a specific path

Inductive definition of path formulas:

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

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
- $\neg g_1, g_1 \vee g_2, g_1 \wedge g_2, \boldsymbol{X} g_1, \boldsymbol{G} g_1, \boldsymbol{F} g_1, g_1 \boldsymbol{U} g_2$, $g_1 \boldsymbol{R} g_2$ where g_1, g_2 are path formulas

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
- $\neg g_1, g_1 \vee g_2, g_1 \wedge g_2, \boldsymbol{X} g_1, \boldsymbol{G} g_1, \boldsymbol{F} g_1, g_1 \boldsymbol{U} g_2$, $g_1 \boldsymbol{R} g_2$ where g_1, g_2 are path formulas

CTL* is the set of all state formulas

Semantics of CTL*

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

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:
  - $M, s \vDash f$ … the **state** formula f holds in state s of M

SCOS
Secure & Correct Systems

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:
  - $M, s \vDash f$ … the **state** formula f holds in state s of M

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

Semantics of CTL*

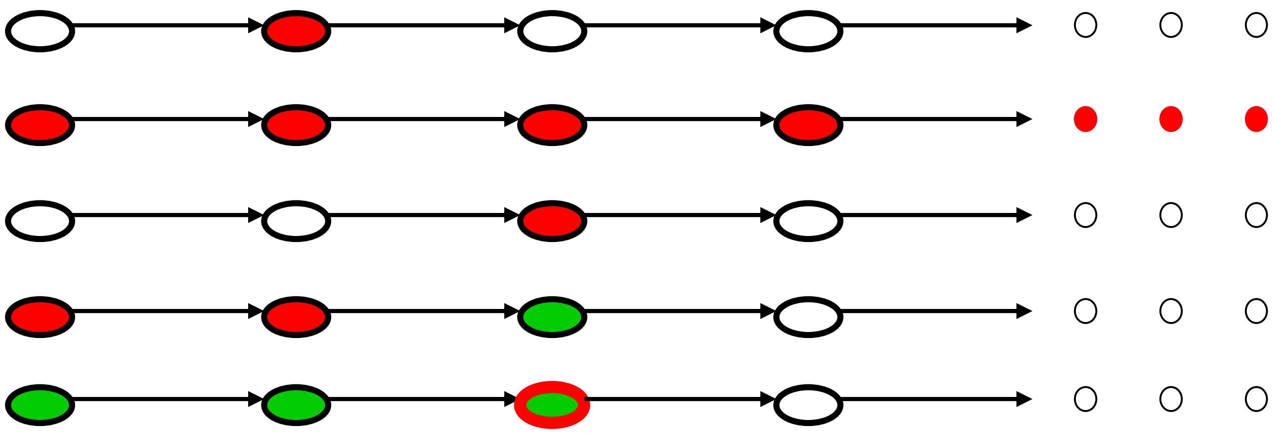
State formulas:

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

Semantics of path formulas - summary

If p,q are state formulas, then:

Xp
Gp
Fp
p**U**q
p**R**q

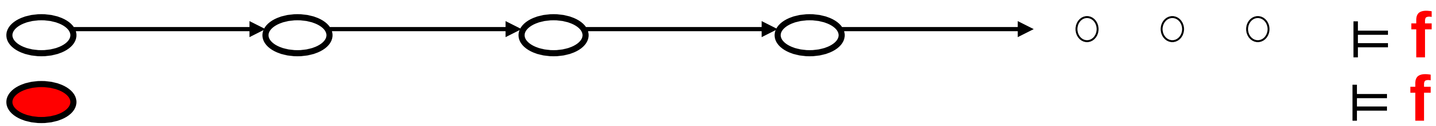


But in the general case, they can be path formulas

Semantics of CTL*

Path formulas:

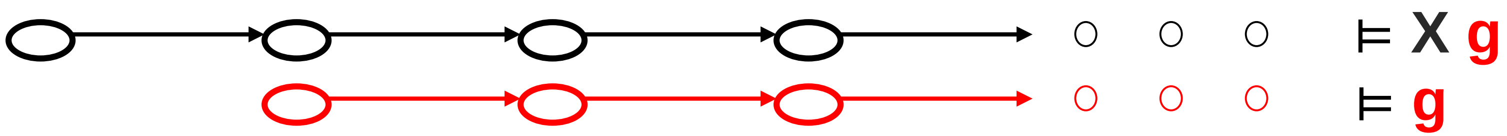
- $M, \pi \models f$, where f is a state formula $\Leftrightarrow M, s_0 \models f$



Semantics of CTL*

Path formulas:

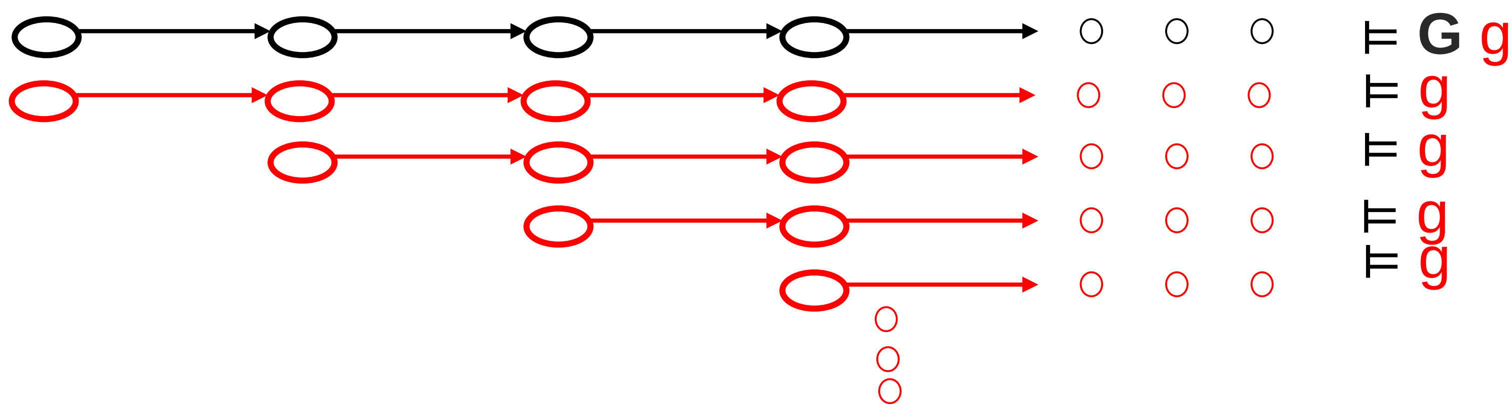
- M, $\pi \models \mathbf{X}g$, where g is a path formula $\Leftrightarrow$ M, $\pi^1 \models g$



Semantics of CTL*

Path formulas:

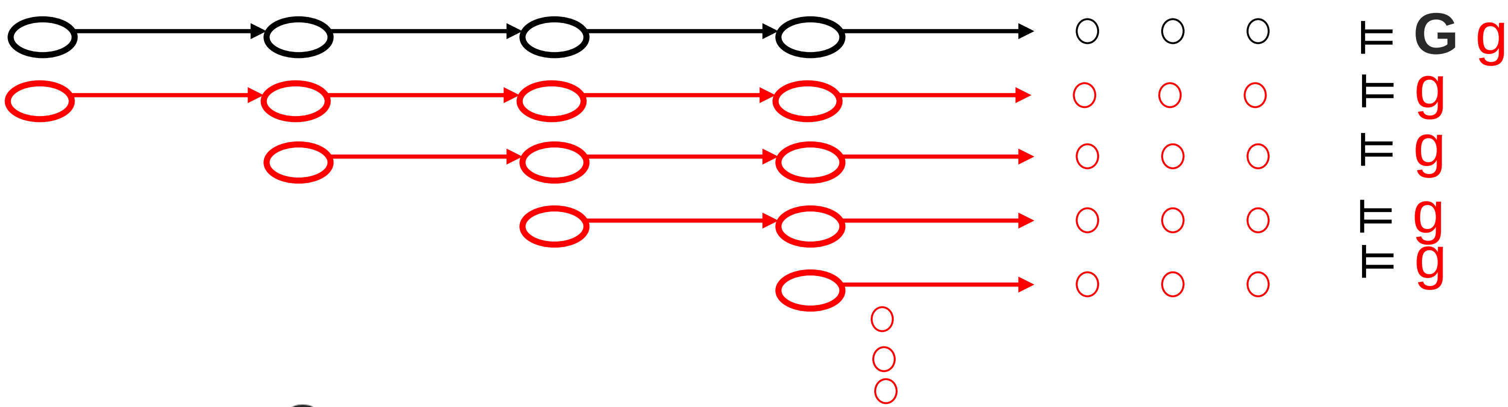
- M, $\pi \vDash \mathbf{G}g \Leftrightarrow$ for every $i \geq 0$, M, $\pi^i \vDash g$



Semantics of CTL*

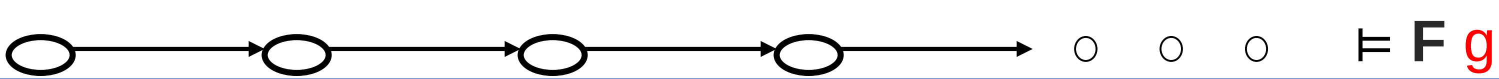
Path formulas:

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



⊨ G g

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

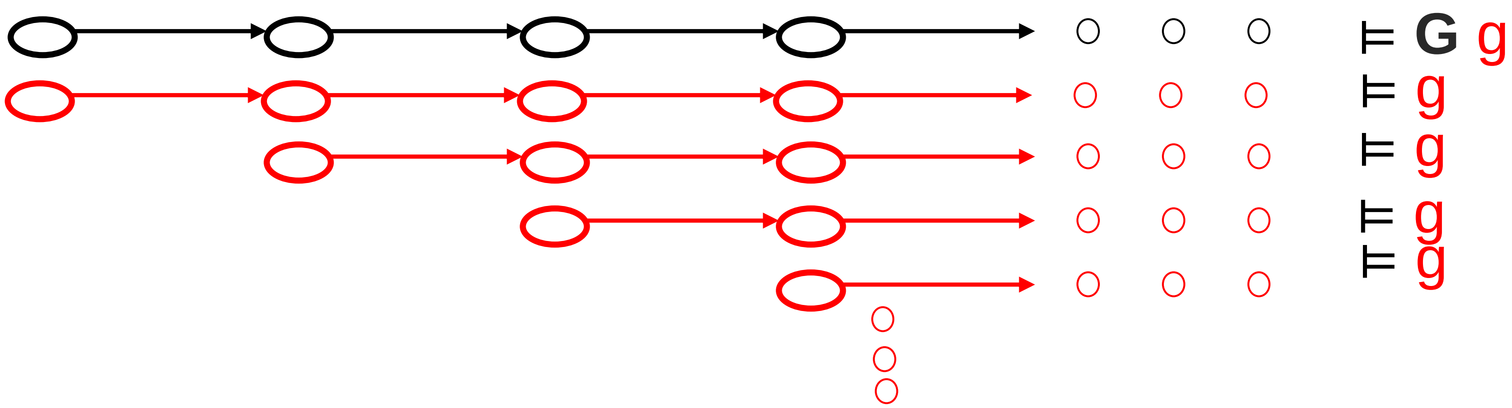


⊨ G g

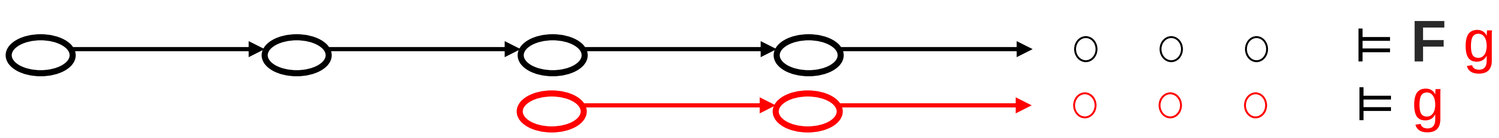
Semantics of CTL*

Path formulas:

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



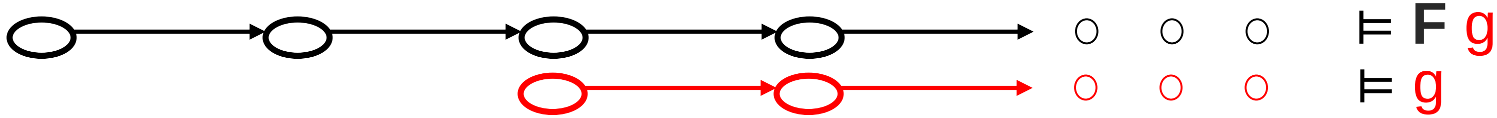
- $M, \pi \vDash \mathbf{F}g \Leftrightarrow$ there exists $k \geq 0$, such that $M, \pi^k \vDash g$



Semantics of CTL*

Path formulas:

- M, π ⊨ **F**g ⇔ there exists k ≥0, such that M, π^k ⊨ g

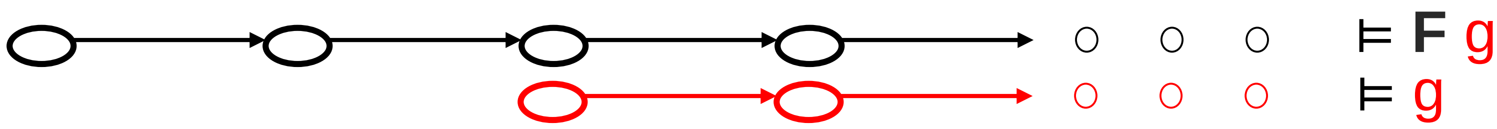


- M, π ⊨ g_1 **U** g_2 ⇔

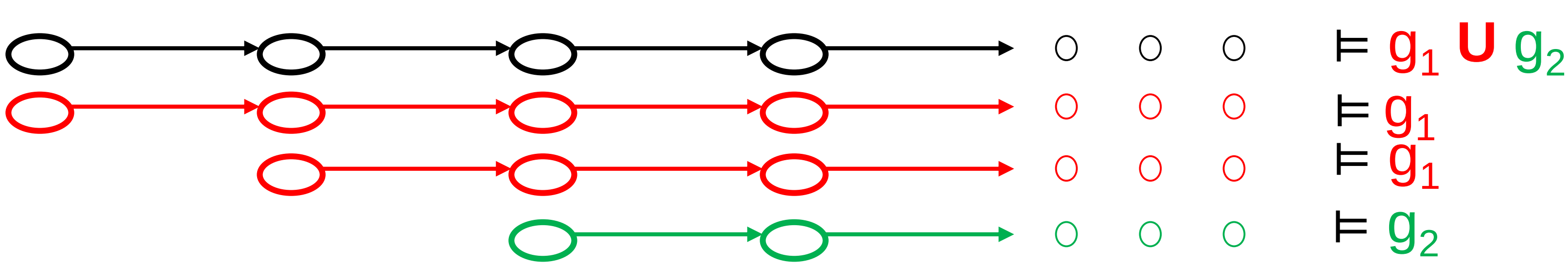
Semantics of CTL*

Path formulas:

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



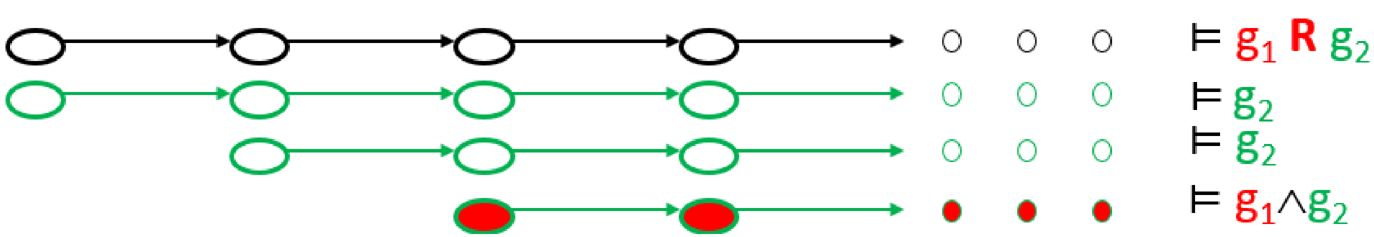
- $M, \pi \models g_1 \mathbf{U}\, g_2 \Leftrightarrow$ there exists $k \geq 0$, such that $M, \pi^k \models g_2$ and for every $0 \leq j < k$, $M, \pi^j \models g_1$



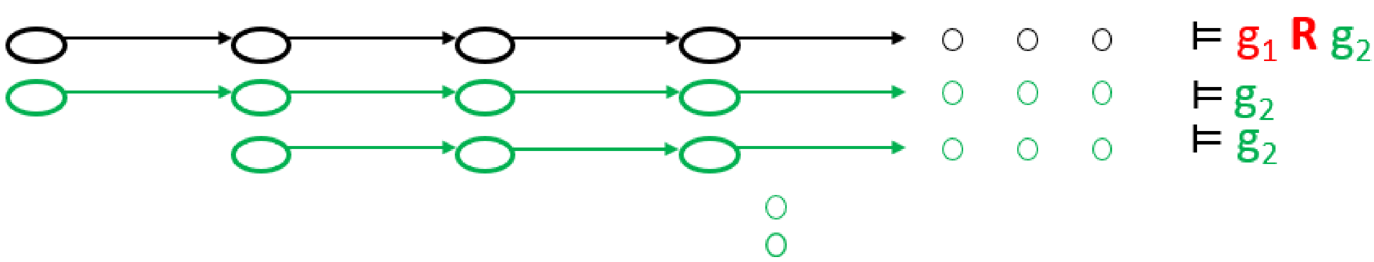
Semantics of CTL*

Path formulas:

- $M, \pi \models g_1 \mathbf{R}\, g_2 \Leftrightarrow$ for all $j \geq 0$, if for every $i<j$ $M, \pi^i \not\models g_1$ then $M, \pi^j \models g_2$



or



More about R („release“)

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

ToDo Rewrite it using U, F, G, or X

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

More about 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

- $M \models f \iff$

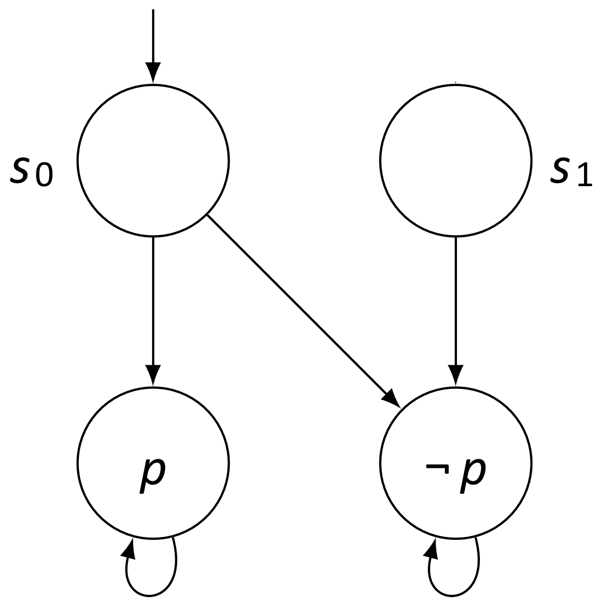
Semantics of CTL*

- $M \models f \Leftrightarrow$ for all initial states $s_0 \in S_{0:}$ $M, s_0 \models f$

Semantics of CTL*

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

- Example: Does $M \models EX\ p$ or $M \models \neg EX\ p$?

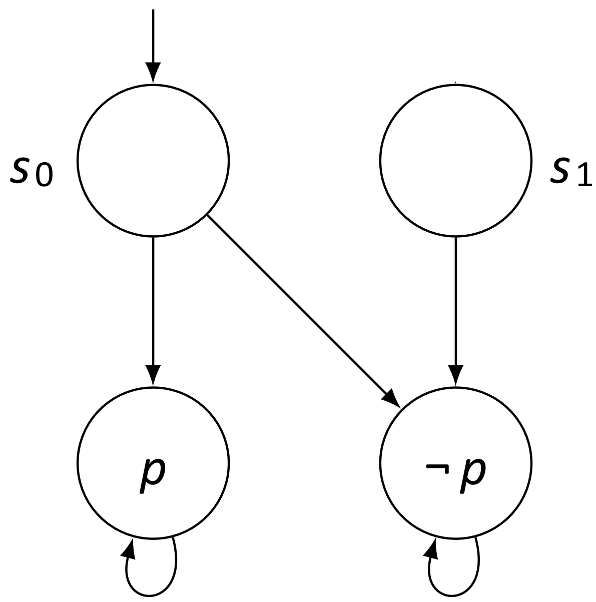


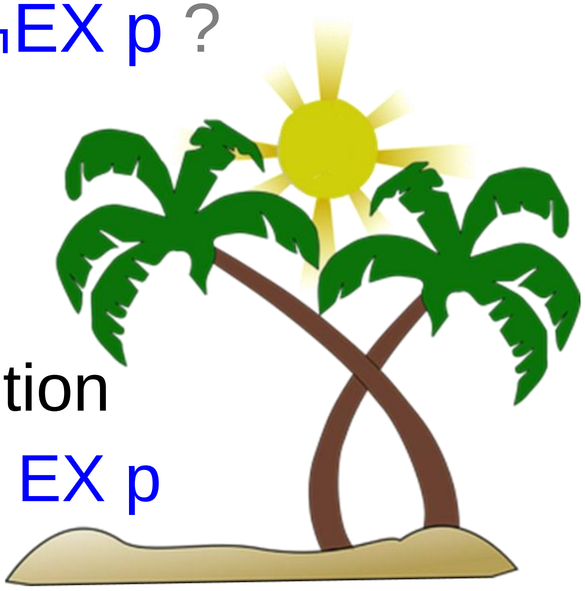
Semantics of CTL*

- $M \models f \;\Leftrightarrow\;$ for all initial states $s_0 \in S_0$, $\;M, s_0 \models f$

ToDo

- Example: Does $M \models EX\ p$ or $M \models \neg EX\ p$?





Solution

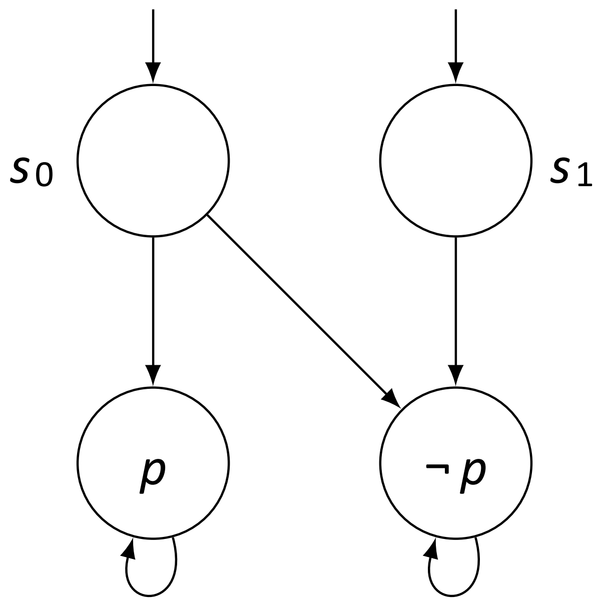
$M \models EX\ p$

Semantics of CTL*

- $M \models f \;\Leftrightarrow\;$ for all initial states $s_0 \in S_0$, $\; M, s_0 \models f$

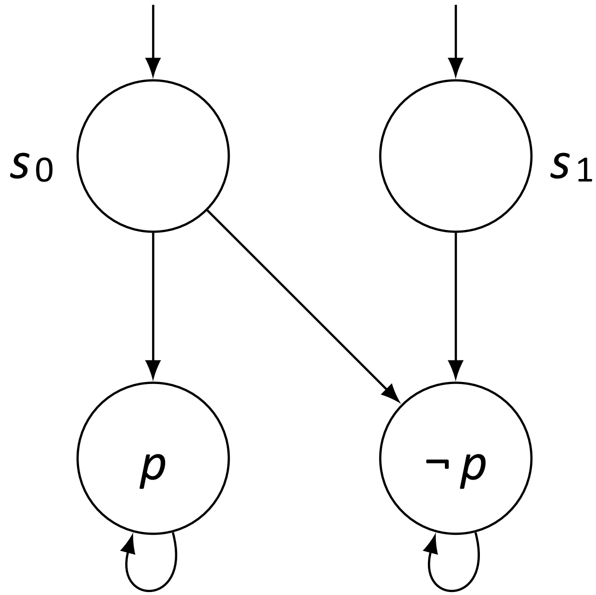
ToDo

- Example: Does $M \models EX\, p$ or $M \models \neg EX\, p$?



Semantics of CTL*

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



Neither

Exercise 1

Question:

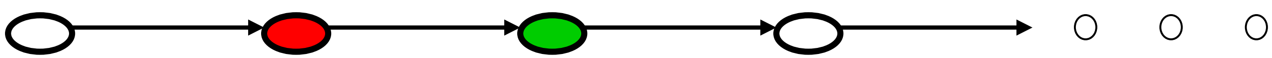
- Given a, b $\in$ AP
  How do all paths that satisfy (**F**b) **U** a look like?

Exercise 1

Question:

- Given a, b ∈ AP
  How do all paths that satisfy **(F**b**) U** a look like?

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



Exercise 2

Question:

For p ∈ AP, what are the meaning of the following formulas? That is, when does π satisfy each of the formulas:

- $\pi \models \mathbf{GF}\ p$
- $\pi \models \mathbf{FG}\ p$

Exercise 2

Question:

For p $\in$ AP, what are the meaning of the following formulas? That is, when does π satisfy each of the formulas:

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

SCOS
Secure & Correct Systems

Exercise 2

Question:

For $p \in AP$, what are the meaning of the following formulas? That is, when does s satisfy each of the 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 $\neg$p along π

Exercise 2

Question:

For $p \in AP$, what are the meaning of the following formulas? That is, when does s satisfy each of the 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 $\neg p$ along π

Exercise 3

Question:

When does π satisfy the formula:

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

Answer:

Exercise 3

Question:

When does π satisfy the formula:

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

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 $\vee$, $\neg$, **X**, **U**, **E** are sufficient to express any CTL* formula:

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

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

- Example:
  Transforming a formula into NNF:

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

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:

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

Useful sublogics of CTL*

- CTL, ACTL and ACTL* are branching-time temporal logics
  - 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:

- $\mathbf{A}f$ 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!