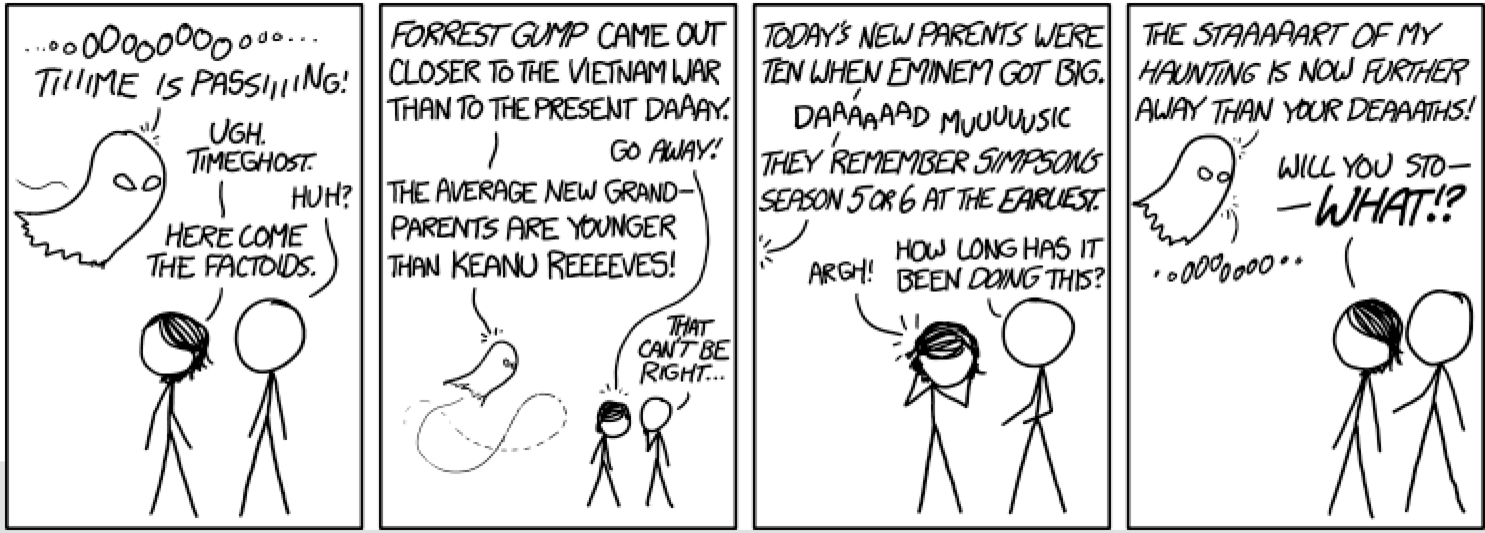
Logic and Computability

Temporal Logic

Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at



https://xkcd.com/1393/

Warm Up – Modelling sentences

Translate the following sentences in propositional logic:

- "If there is coffee and cake, then the workshop is a success. "

Warm Up – Modelling sentences

Translate the following sentences in propositional logic:

- "If there is coffee and cake, then the workshop is a success. "
  - p… there is coffee, q… there is cake, r… the workshop is a success
  - $p \land q \to r$

Warm Up – Modelling sentences

Translate the following sentences in propositional logic:

- “If there is a request, the arbiter gives a grant in the **next time step**. “
- “If there is a request, the arbiter gives a grant within the **next two time steps**. “
- “If there is a request, the arbiter gives a grant **eventually**. “

Warm Up – Modelling sentences

Translate the following sentences in propositional logic:

- “If there is a request, the arbiter gives a grant in the **next time step**. “
  - p… there is a request, q… arbiter gives a grant in the next time step
  - $p \rightarrow \mathrm{q}$
- “If there is a request, the arbiter gives a grant within the **next two time steps**. “
- “If there is a request, the arbiter gives a grant **eventually**. “

Warm Up – Modelling sentences

Translate the following sentences in propositional logic:

- "If there is a request, the arbiter gives a grant in the **next time step**. "
  - p… there is a request, q… arbiter gives a grant in the next time step
  - $p \rightarrow \mathrm{q}$
- "If there is a request, the arbiter gives a grant within the **next two time steps**. "
  - p… there is a request, q… arbiter gives a grant within the next two time steps
  - $p \rightarrow \mathrm{q}$
- "If there is a request, the arbiter gives a grant **eventually**. "

Warm Up – Modelling sentences

Translate the following sentences in propositional logic:

- "If there is a request, the arbiter gives a grant in the **next time step**. "
  - p… there is a request, q… arbiter gives a grant in the next time step
  - $p \rightarrow \mathrm{q}$
- "If there is a request, the arbiter gives a grant within the **next two time steps**. "
  - p… there is a request, q… arbiter gives a grant within the next two time steps
  - $p \rightarrow \mathrm{q}$
- "If there is a request, the arbiter gives a grant **eventually**. "
  - p… there is a request, q… the arbiter gives a grant eventually
  - $p \rightarrow \mathrm{q}$

Motivation

- We want to specify properties of hardware and software
- Temporal operators allow to encode formula about the future of paths
  - a condition will eventually be true
  - a condition will be true until another fact becomes true
  - etc
- Model Checking
  - Checks whether a model of a system meets a given specification
  - Specification typically expressed in temporal logic

Outline

- Temporal Operators
  - Modelling Sentences
- Kripke Structures
  - Temporal Properties on Kripke Structures
- CTL*
  - Path Operators
  - Intuitive Meaning + Syntax

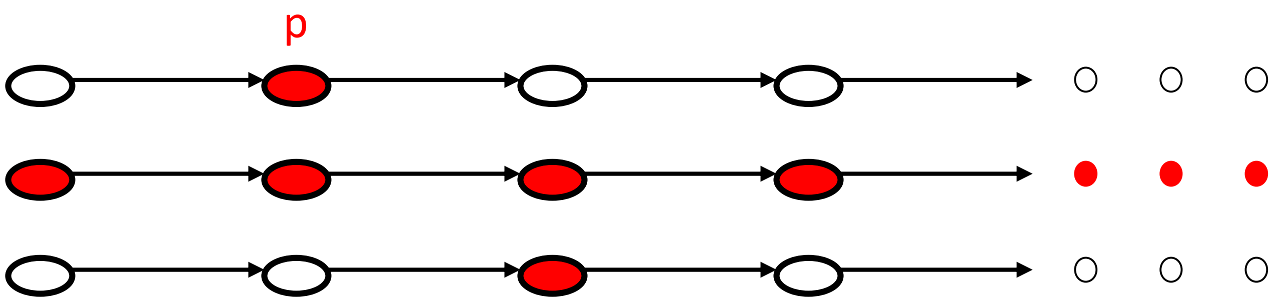
Temporal Operators

- Describe properties that hold along an execution path of a system

Next **X**p

Always / Gloabally **G**p

Finally / Eventually **F**p



Translate in temporal logic

Translate the following sentences in temporal propositional logic:

- “If there is a request, the arbiter gives a grant in the **next time step**. “

- “If there is a request, the arbiter gives a grant within the **next two time steps**. “

- “If there is a request, the arbiter gives a grant **eventually**. “

Temporal Operators
X… next
G… globally
F… eventually

Translate in temporal logic

Translate the following sentences in temporal propositional logic:

- "If there is a request, the arbiter gives a grant in the **next time step**. "
  - r… there is a request, g… arbiter gives grant
  - $G(r \rightarrow Xg)$
- "If there is a request, the arbiter gives a grant within the **next two time steps**. "
- "If there is a request, the arbiter gives a grant **eventually**. "

Temporal Operators
X… next
G… globally
F… eventually

Translate in temporal logic

Translate the following sentences in temporal propositional logic:

- “If there is a request, the arbiter gives a grant in the **next time step**. “
  - r… there is a request, g… arbiter gives grant
  - $G(r \rightarrow Xg)$
- “If there is a request, the arbiter gives a grant within the **next two time steps**. “
  - r… there is a request, g… arbiter gives grant
  - $G(r \rightarrow (Xg \vee XXg))$
- “If there is a request, the arbiter gives a grant **eventually**. “

Temporal Operators
X… next
G… globally
F… eventually

Translate in temporal logic

Translate the following sentences in temporal propositional logic:

- “If there is a request, the arbiter gives a grant in the **next time step**. “
  - r… there is a request, g… arbiter gives grant
  - $G(r \rightarrow Xg)$
- “If there is a request, the arbiter gives a grant within the **next two time steps**. “
  - r… there is a request, g… arbiter gives grant
  - $G(r \rightarrow (Xg \vee XXg))$
- “If there is a request, the arbiter gives a grant **eventually**. “
  - r… there is a request, g… arbiter gives grant
  - $G(r \rightarrow Fg)$

Temporal Operators
X… next
G… globally
F… eventually

Translate the following sentences in temporal propositional logic:

a) The system gives a grant infinitely often.
b) The system sends a request finitely often.

- "The system gives a grant infinitely often."

- "The system sends a request finitely often."

Temporal Operators
X… next
G… globally
F… eventually

Translate the following sentences in temporal propositional logic:

a) The system gives a grant infinitely often.

b) The system sends a request finitely often.

- “The system gives a grant infinitely often.“
  - g… the system sends a grant
  - GF(g)
- “The system sends a request finitely often.“

Temporal Operators

X… next

G… globally

F… eventually

Translate the following sentences in temporal propositional logic:

a) The system gives a grant infinitely often.
b) The system sends a request finitely often.

- "The system gives a grant infinitely often."
  - g… the system sends a grant
  - $GF(g)$
- "The system sends a request finitely often."
  - r… system sends a request
  - $FG(\neg r)$

Temporal Operators

X… next
G… globally
F… eventually

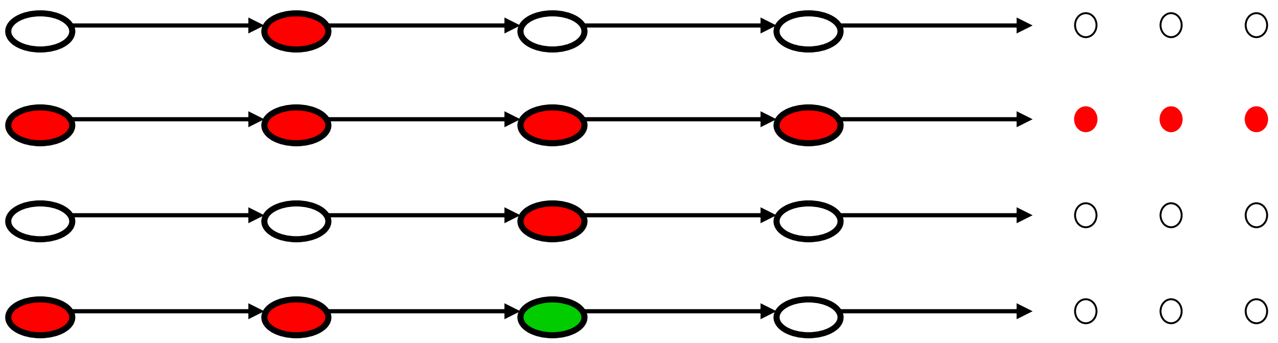
Temporal Operators

Next Xp

Always / Gloabally Gp

Finally / Eventually Fp

Until pUq



Temporal Operators

X… next

G… globally

F… eventually

U… until

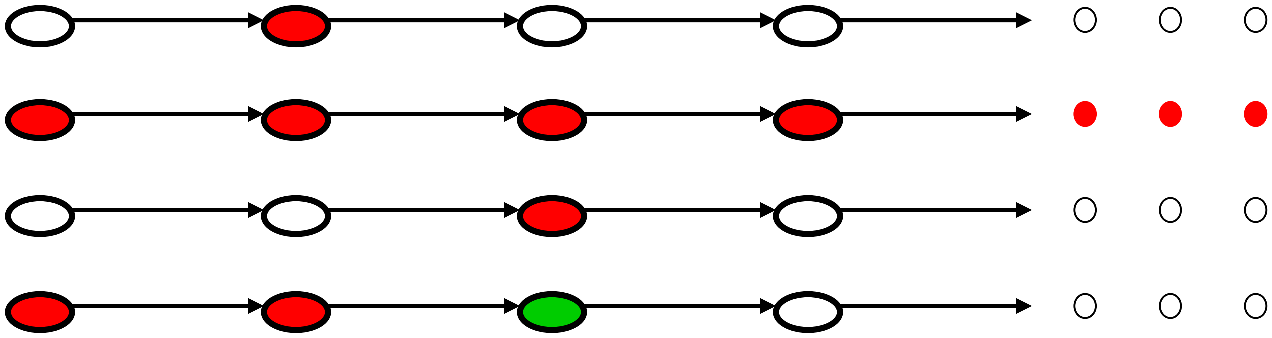
Temporal Operators

Next Xp

Always / Gloabally Gp

Finally / Eventually Fp

Until p**U**q



Translate the following sentences in temporal propositional logic:

- “The request is high until the arbiter gives a grant. “

Temporal Operators
X… next
G… globally
F… eventually
U… until

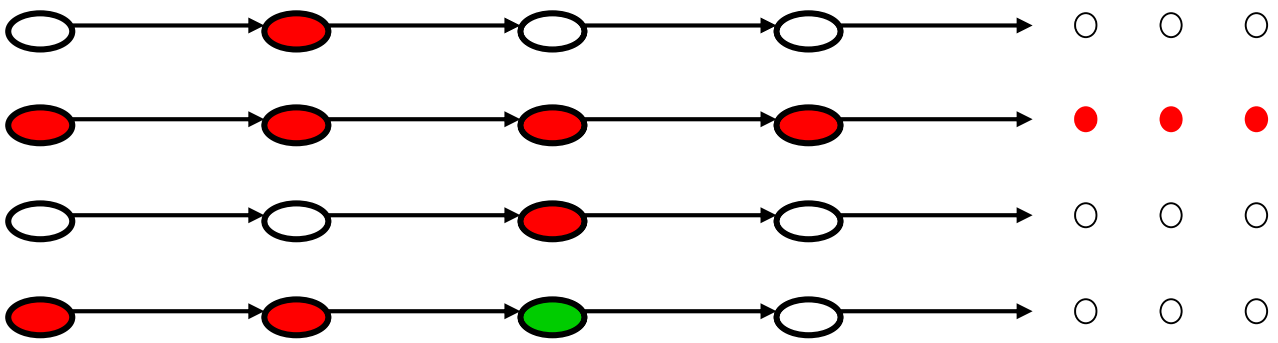
Temporal Operators

Next Xp

Always / Gloabally Gp

Finally / Eventually Fp

Until pUq



Translate the following sentences in temporal propositional logic:

- “The request is high until the arbiter gives a grant. “
  - r… request is high, g… arbiter gives grant
  - r U g

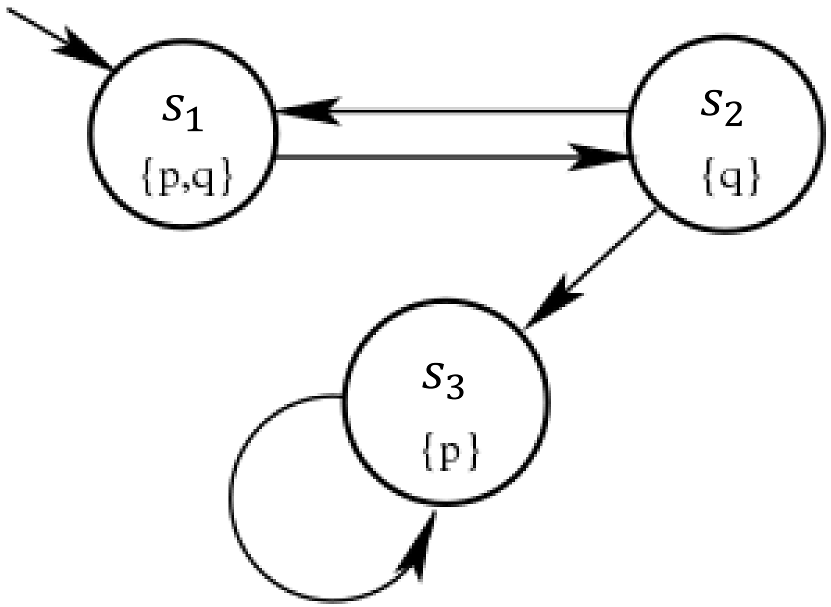
Temporal Operators
X… next
G… globally
F… eventually
U… until

Outline

- Temporal Operators
  - Modelling Sentences
- Kripke Structures
  - Temporal Properties on Kripke Structures
- CTL*
  - Path Operators
  - Intuitive Explanation

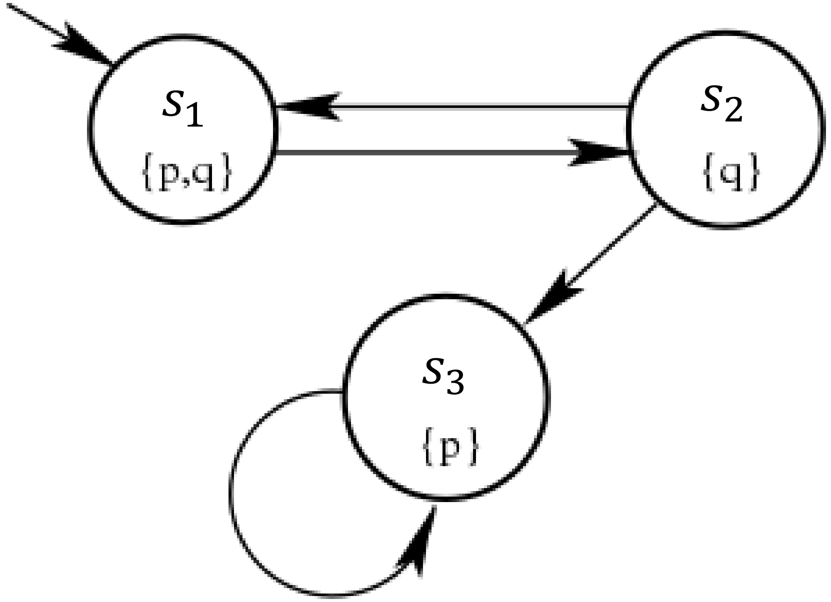
Kripke Structures

- Let AP be a set of Boolean variables (atomic propositions)
- A Kripke Structure is a tuple K=(S, S_0, R, L)
  - Finite Set of States S
  - Set of Initial States $S_0 \subseteq S$
  - Transition Relation $R \subseteq S \times S$
  - Labeling function: $L : S \to 2^{AP}$



Kripke Structure - Example

- In this example, AP = {p,q} and K=(S, S_0, R, L) with
  - $S = \{s_1, s_2, s_3\}$
  - $S_0 = \{s_1\}$
  - $R = \{(s_1, s_2), (s_2, s_1), (s_2, s_3), (s_3, s_3)\}$
  - $L = \{(s_1, \{p, q\}), (s_2, \{q\}), (s_3, \{p\})\}$

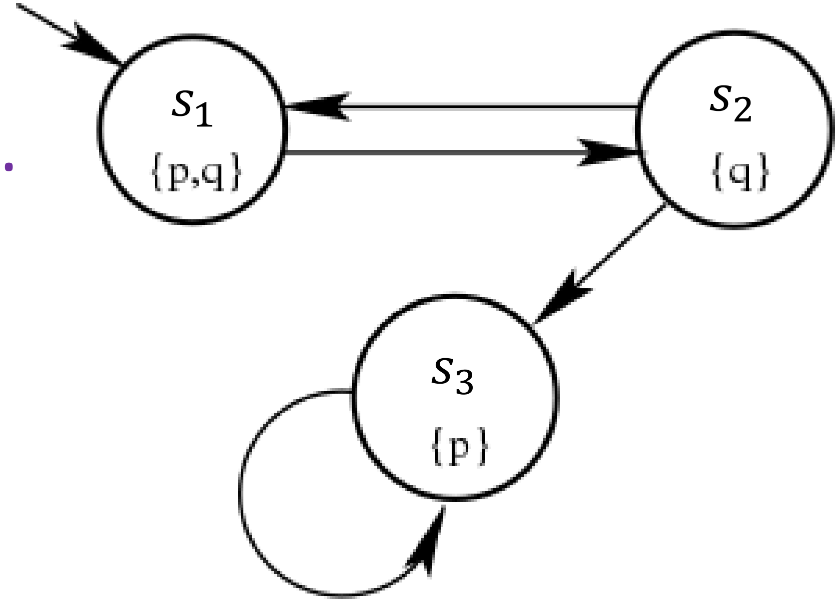


Kripke Structures – Paths and Words

- Given a Kripke Structure K=(S, S_0, R, L)

- A path is a is a sequence of states $\rho = s_1, s_2 \ldots$ s.t. for each i > 0, $R(s_i, s_{i+1})$ hold

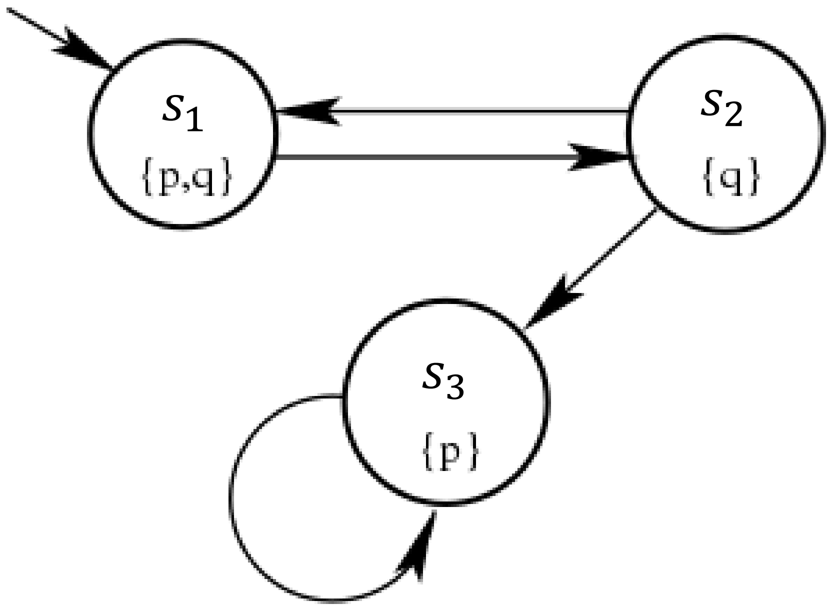
- The *word* on the path ρ is a sequence of sets of the atomic propositions $w = L(s_1), L(s_2), L(s_3), \ldots$



Kripke Structures – Paths and Words - Example

- Given a Kripke Structure K=(S, S_0, R, L)

- Given a path $\rho = s_1, s_2, s_1, s_2, s_3, s_3, s_3, \ldots = s_1, s_2, s_1, s_2, s_3^{\omega}$
- What is the execution word w over the path ρ?

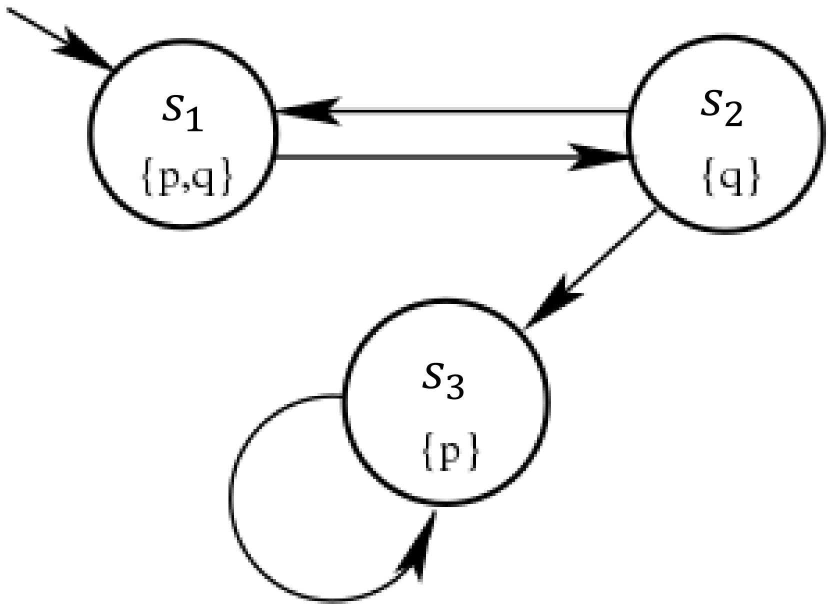


Kripke Structures – Paths and Words - Example

- Given a Kripke Structure K=(S, S_0, R, L)

- Given a path $\rho = s_1, s_2, s_1, s_2, s_3, s_3, s_3, \ldots = s_1, s_2, s_1, s_2, s_3^{\omega}$
- What is the execution word w over the path ρ?

- w = {p, q}, {q}, {p, q}, {q}, $\{p\}^{\omega}$



Given the following execution word w of a Kripke structure. Evaluate the formula φ on w. Evaluate each sub-formula for any execution step using the provided table.

- $w = \{\}\ \{a\}, \{a\}, \{b\}, \{\}, \{a\}, \{a, b\}^{\omega}$
- $\varphi = Xa \vee a\ U\ b$

Step	0	1	2	3	4	5	ω
a	0	1	1	0	0	1	1
b	0	0	0	1	0	0	1
Xa							
aUb							
$Xa \vee aUb$							

Given the following execution word w of a Kripke structure. Evaluate the formula φ on w. Evaluate each sub-formula for any execution step using the provided table.

- $\mathrm{w} = \{\} \{a\}, \{a\}, \{b\}, \{\}, \{a\}, \{a, b\}^{\omega}$
- $\varphi = Xa \vee a\ U\ b$

Step	0	1	2	3	4	5	ω
a	0	1	1	0	0	1	1
b	0	0	0	1	0	0	1
Xa	1	1	0	0	1	1	1
aUb	0	1	1	1	0	1	1
$Xa \vee aUb$	1	1	1	1	1	1	1

Given the following execution word w of a Kripke structure. Evaluate the formula φ on w. Evaluate each sub-formula for any execution step using the provided table.

- w = {} {a}, { }, {a,b,c}, {a}, {a,b}, $(\{a\}, \{a, c\}, \{a, c\})^\omega$
- $\varphi = GFa \rightarrow (FG\neg b \wedge c)$

Step	0	1	2	3	4	5	ω		
a	0	1	0	1	1	1	1	1	1
b	0	0	0	1	0	1	0	0	0
c	0	0	0	1	0	0	0	1	1
GFa									
$FG\neg b$									
$FG\neg b \wedge c$									
φ									

Given the following execution word w of a Kripke structure. Evaluate the formula φ on w. Evaluate each sub-formula for any execution step using the provided table.

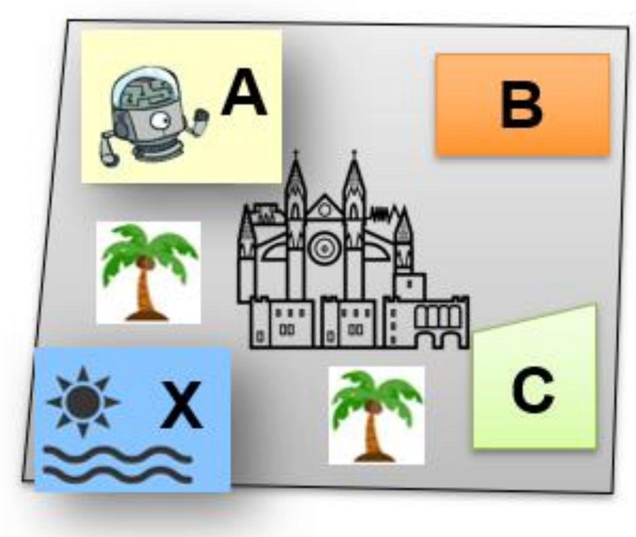
- w = {} {a}, { }, {a,b,c}, {a}, {a,b}, $(\{a\}, \{a, c\}, \{a, c\})^{\omega}$
- $\varphi = GFa \rightarrow (FG\neg b \wedge c)$

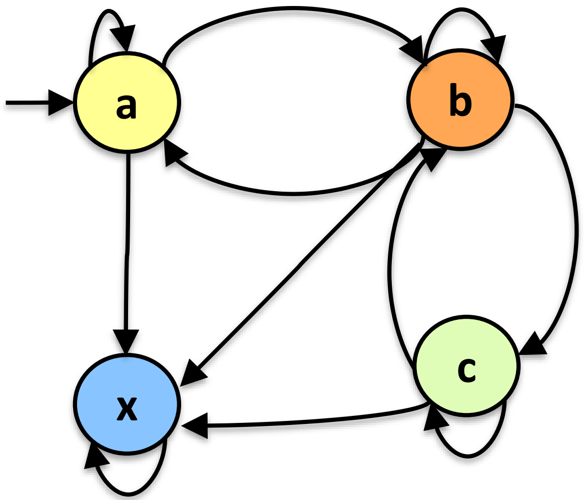
Step	0	1	2	3	4	5	ω		
a	0	1	0	1	1	1	1	1	1
b	0	0	0	1	0	1	0	0	0
c	0	0	0	1	0	0	0	1	1
GFa	1	1	1	1	1	1	1	1	1
$FG\neg b$	1	1	1	1	1	1	1	1	1
$FG\neg b \wedge c$	0	0	0	1	0	0	0	1	1
φ	0	0	0	1	0	0	0	1	1

Outline

- Temporal Operators
  - Modelling Sentences
- Kripke Structures
  - Temporal Properties on Kripke Structures
- CTL*
  - Path Operators
  - Intuitive Explanation

Properties of Kripke Structures - Example

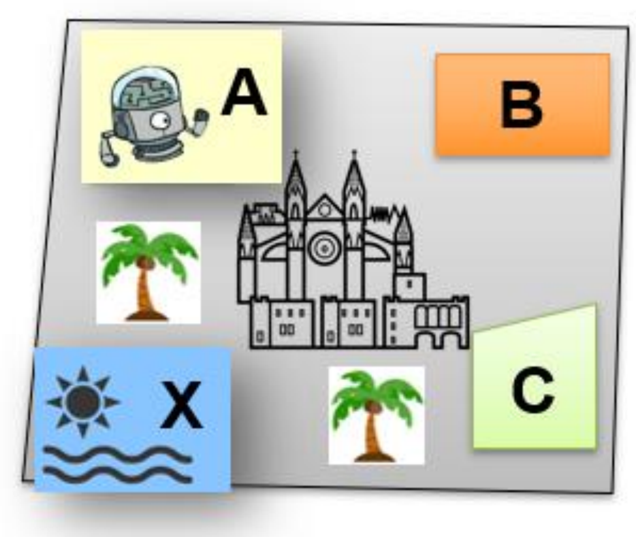


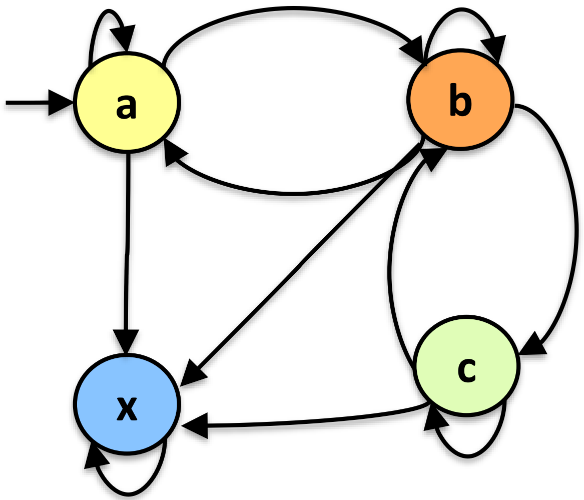


Kripke Structure K=(S, S_0, R, L)

- Robot navigating within a city
- Kripke structure models its allowed movements
  - E.g., if the robot enters the sea, it cannot leave it anymore

Properties of Kripke Structures - Example





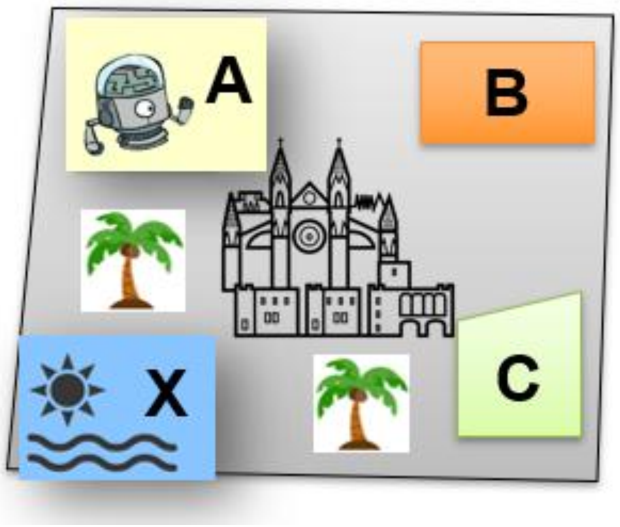
Specify the following properties in temporal logic:

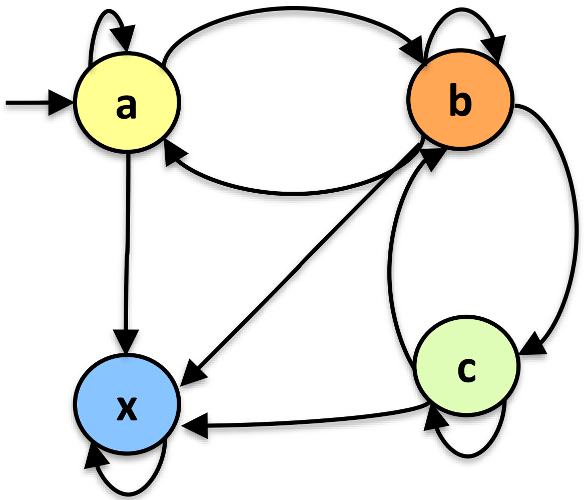
- It is always the case that the robot *never* visits **X.**

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

Temporal Operators
X… next
G… globally
F… eventually
U… until

Properties of Kripke Structures - Example





Temporal Operators
X… next
G… globally
F… eventually
U… until
Path quantifiers
A for **all** paths
E there **exists** a path

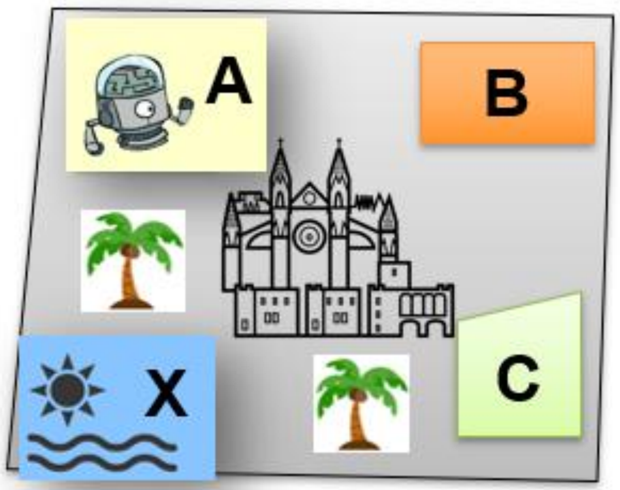
Specify the following properties in temporal logic:

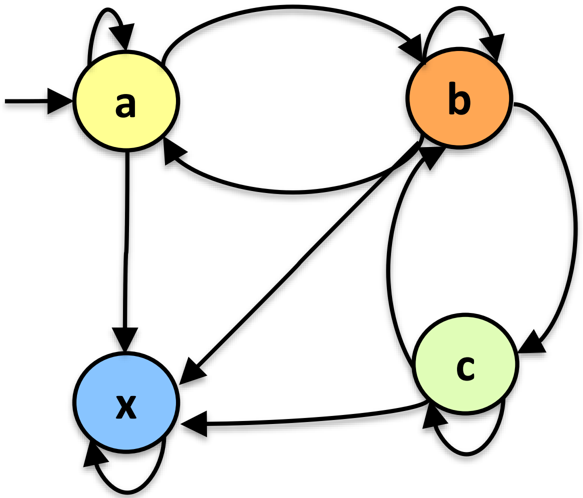
- It is always the case that the robot *never* visits **X.**
  - $A\ G\ \neg x$
- It is possible that the robot *never* visits **X.**
  - $E\ G\ \neg x$

Translate the following sentences in CTL*.

- For any execution, it always holds that whenever the robot visits A, it visits C within the next two steps.

- There exists an execution such that the robot visits C within the next two steps after visiting A



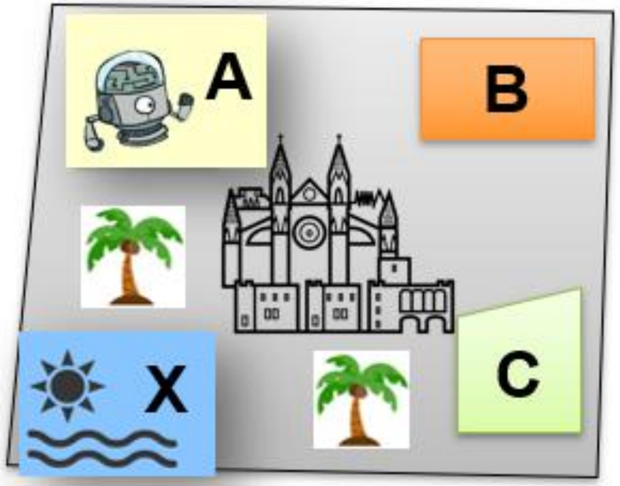


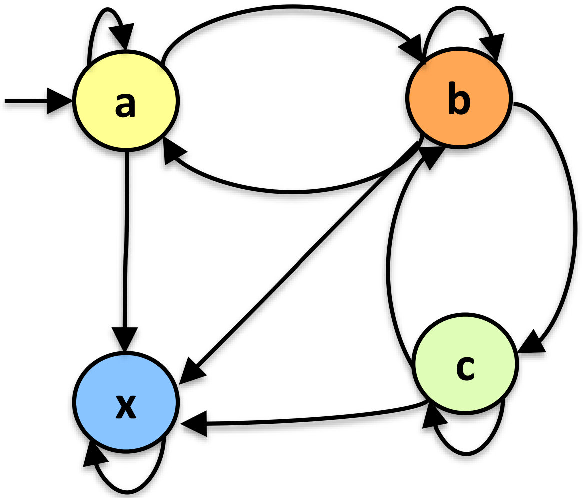
Temporal Operators
X… next
G… globally
F… eventually
U… until
Path quantifiers
A for **all** paths
E there **exists** a path

Translate the following sentences in CTL*.

- For any execution it always holds that whenever the robot visits A, it visits C within the next two steps.
  - $A\,G\,(a \rightarrow Xc \vee XXc)$

- There exists an execution such that the robot visits C within the next two steps after visiting A
  - $E\,G\,(a \rightarrow Xc \vee XXc)$

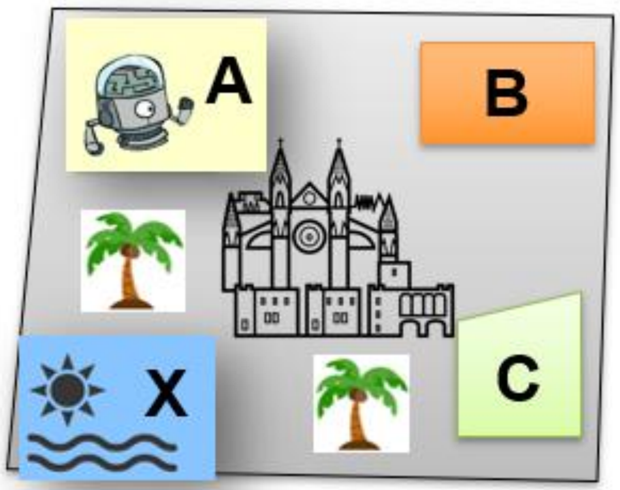


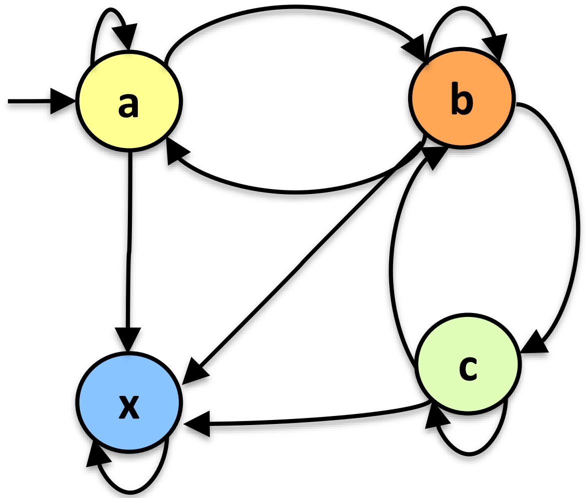


Temporal Operators
X… next
G… globally
F… eventually
U… until
Path quantifiers
A for **all** paths
E there **exists** a path

Translate the following sentences in CTL*.

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





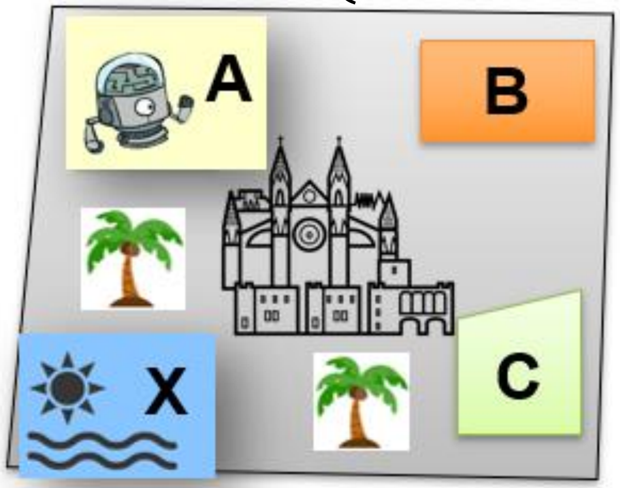
Temporal Operators
X… next
G… globally
F… eventually
U… until
Path quantifiers
A for **all** paths
E there **exists** a path

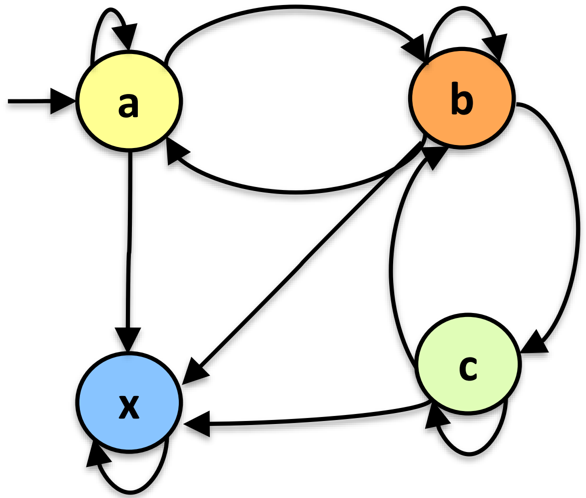
Translate the following sentences in CTL*.

- The robot can visit **A** *infinitely often* and **C** *infinitely often*
  - $E\ (GF\ a \land GF\ c)$

- Always, the robot visits **A** *infinitely often* and **C** *infinitely often*
  - $A\ (GF\ a \land FG\ \neg c)$

- If the robot visits **A** *infinitely often,* it should also visit **C** *finitely often.*
  - $A\ (GF\ a \rightarrow GFc)$



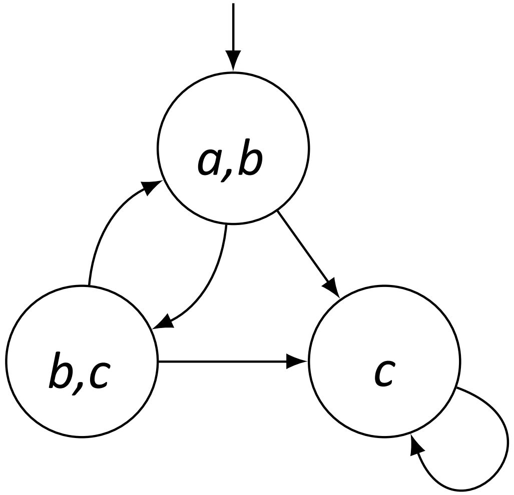


Temporal Operators
X… next
G… globally
F… eventually
U… until
Path quantifiers
A for **all** paths
E there **exists** a path

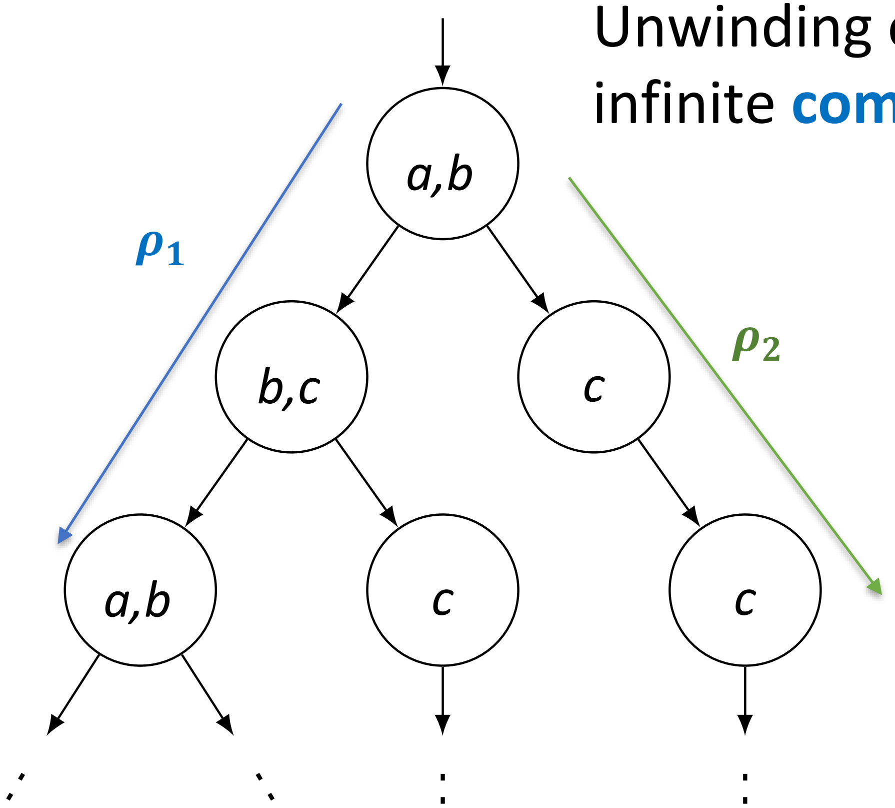
Computation Tree Logic – CTL*

- Defines properties of computation trees of Kripke structures

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



Unwinding of K into
infinite **computation tree**



Computation Tree Logic – CTL*

- Propositional Logic extended with
- Path quantifiers:
  - A for all paths starting from s have property $\boldsymbol{\varphi}$
  - E there exists a path starting from s have property $\boldsymbol{\varphi}$
  - Use combination of A and E to describe branching structure in tree
- Temporal operators
  - **NeXt** - $\boldsymbol{X\varphi}$: $\boldsymbol{\varphi}$ has to hold at the next state
  - **Finally** $-\mathrm{F}\,\boldsymbol{\varphi}$*:* eventually $\boldsymbol{\varphi}$ has to hold (somewhere on the subsequent path)
  - **Globally** - $\mathrm{G}\,\boldsymbol{\varphi}$: φ has to hold on the entire subsequent path.
  - **Until** – $\varphi\ U\ \psi$: ψ has to hold *at least* until φ becomes true,
which must hold at the current or a future position.

Computation Tree Logic – CTL* - Syntax

- A CTL* formula is a „state formula“

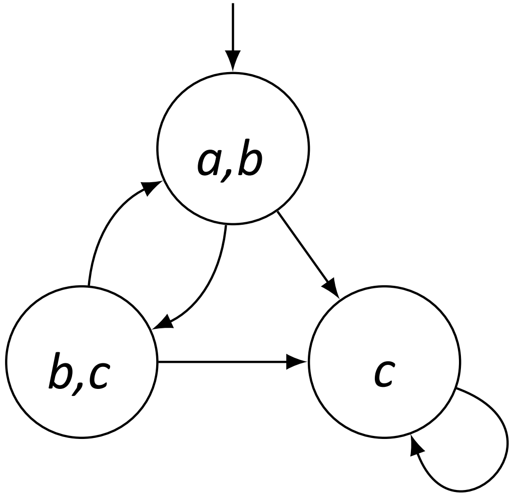
- State formula: $\varphi \coloneqq p \mid \neg\varphi \mid \varphi \vee \varphi | \varphi \wedge \varphi | Ef \mid Af$
  with f beeing a path formula, and p beeing an atomic propostion

- Path formula: $f \coloneqq \varphi \mid \neg f \mid f \vee f | f \wedge f | Xf \mid Ff \mid Gf \mid fUf$

Given the following Kripke structure. Does s_0 satisfy the following formulas?

- $\varphi_1 = EXX(a \wedge b)$

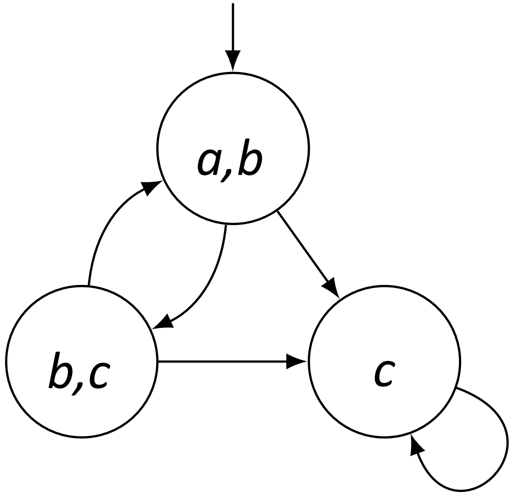
Kripke structure K,
labeled with $AP = \{a, b, c\}$



Given the following Kripke structure. Does s_0 satisfy the following formulas?

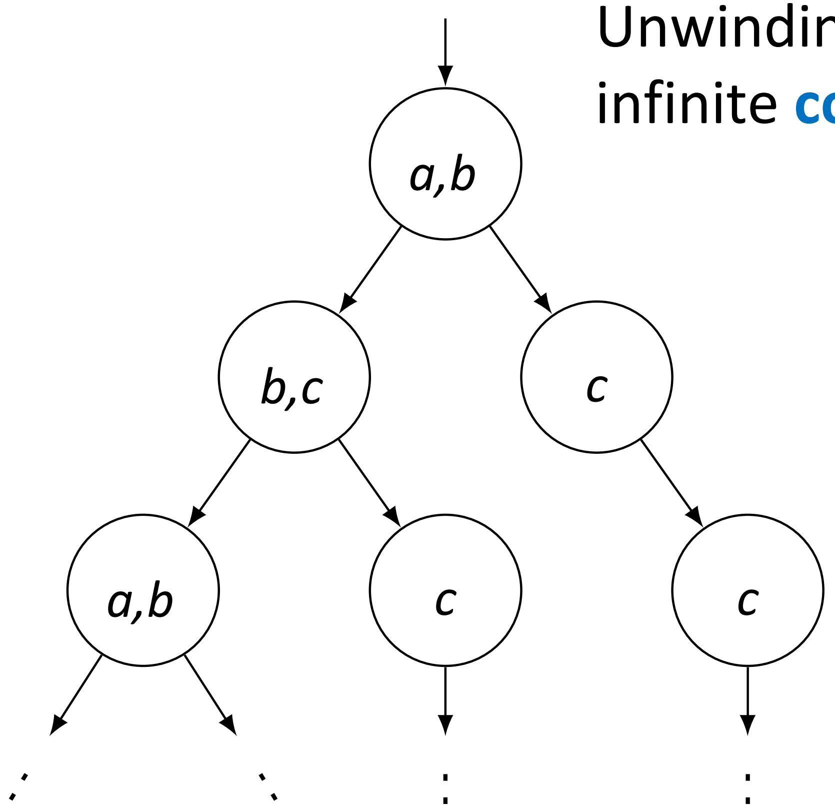
- $\varphi_1 = EXX(a \wedge b)$

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



- $s_0 \vDash \mathrm{EXX}\,(\mathrm{a} \wedge b)$

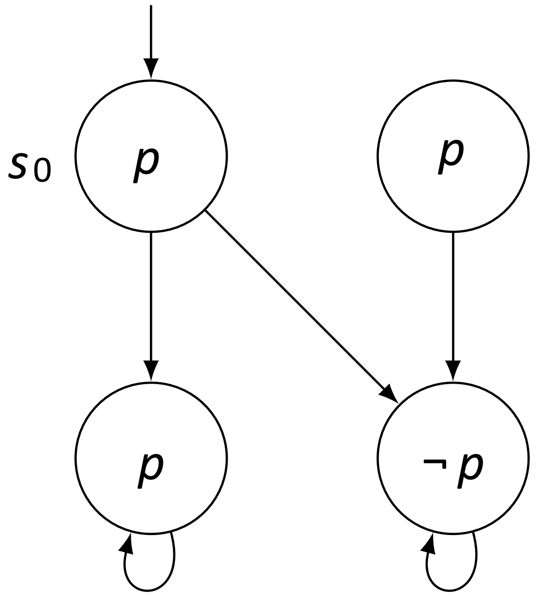
Unwinding of K into
infinite **computation tree**



Given the following Kripke structure K.

Does s_0 satisfy the following formulas?

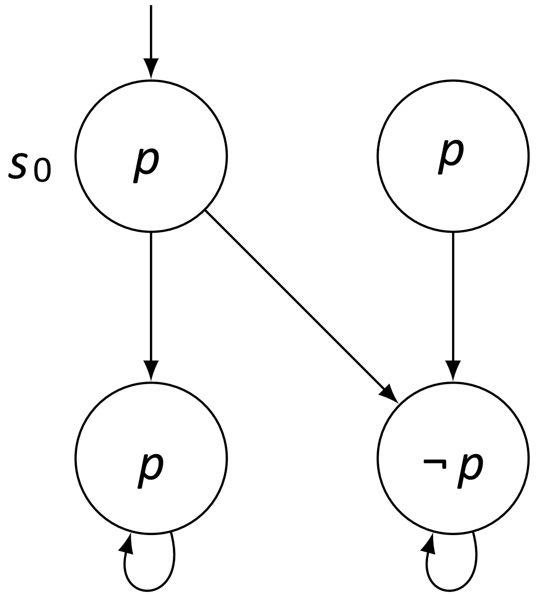
- $\varphi_1 = EXp$
- $\varphi_2 = EG\neg p$



Given the following Kripke structure K.

Does s_0 satisfy the following formulas? Explain your answer.

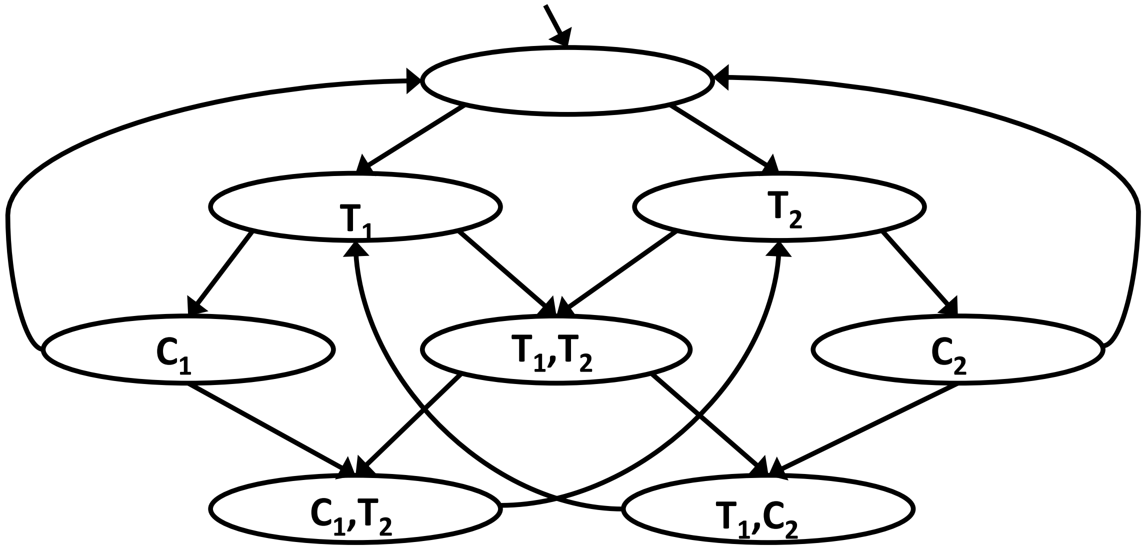
- $\varphi_1 = EXp$
- $\varphi_2 = EG\neg p$



$K \vDash EX\,p$

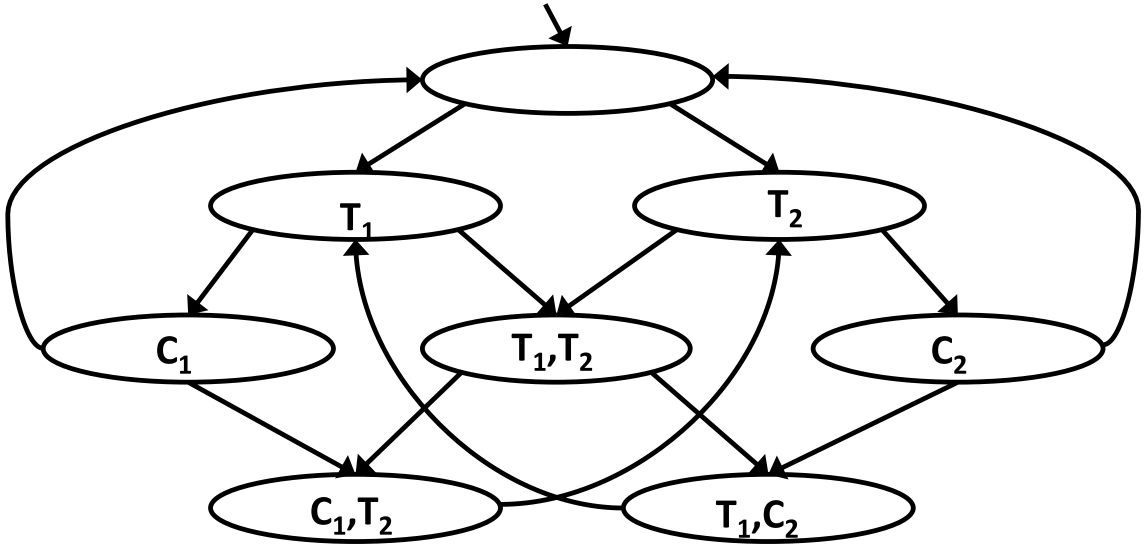
$K \nvDash EG\neg p$

Example – Mutual Exclusion



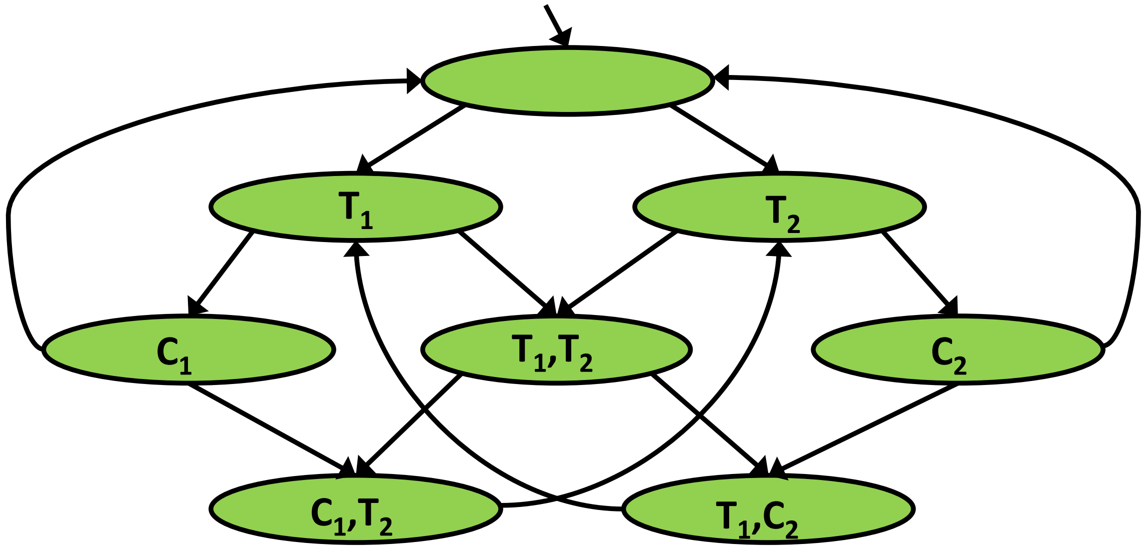
- Two processes with a joint Boolean signal sem
- Each process P_i has a variable v_i describing its state:
  - v_i = N Non-critical
  - v_i = T Trying
  - v_i = C Critical

Example – Mutual Exclusion



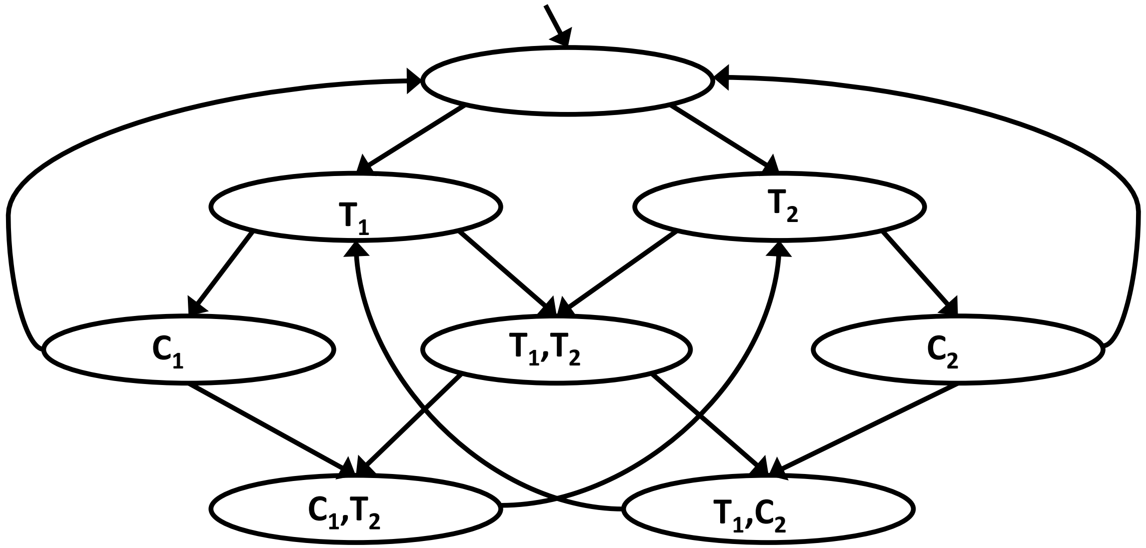
- Does it hold that K $\models \varphi$?
  - Property 1: $\varphi := \mathbf{AG}\neg(C_1 \wedge C_2)$

Example – Mutual Exclusion



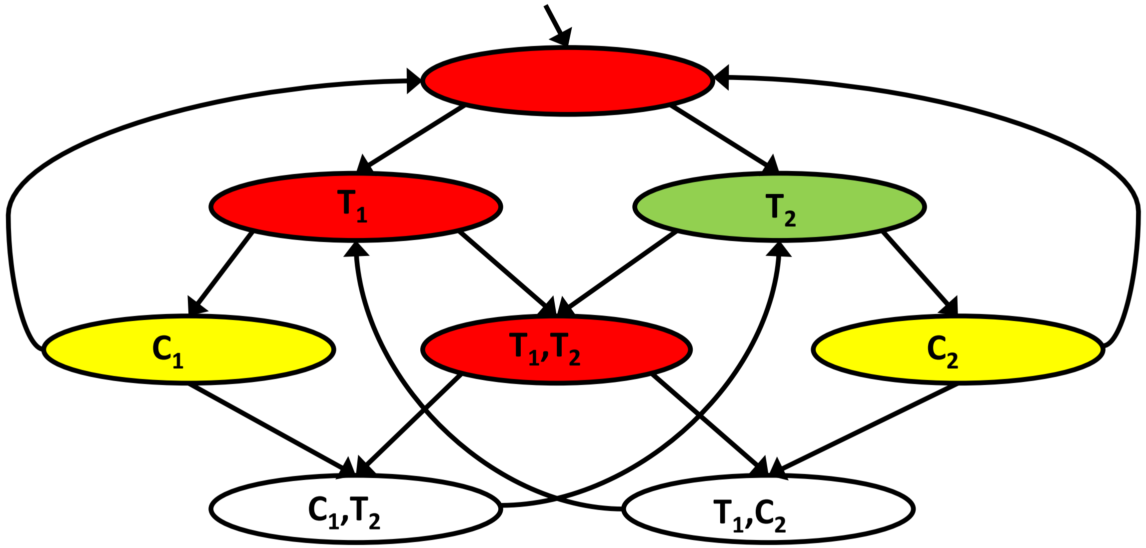
- Does it hold that K $\models$ φ?
  - Property 1: φ := $\mathbf{AG}\neg(C_1 \wedge C_2)$ ✓

Example – Mutual Exclusion



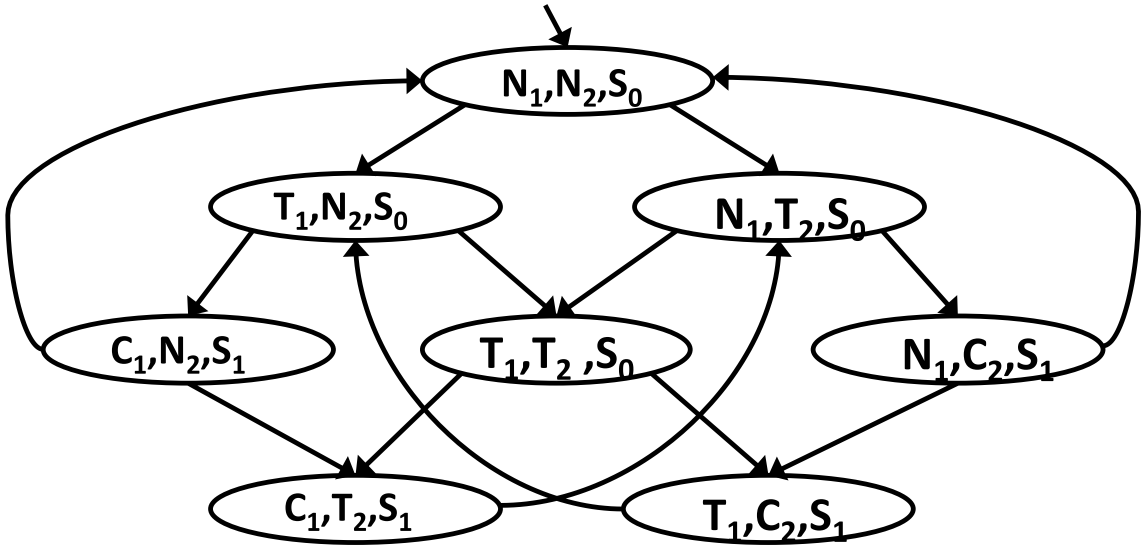
- Does it hold that K $\vDash \varphi$?
  - Property 2: φ := $\mathbf{AG}\neg(T_1 \wedge T_2)$

Example – Mutual Exclusion



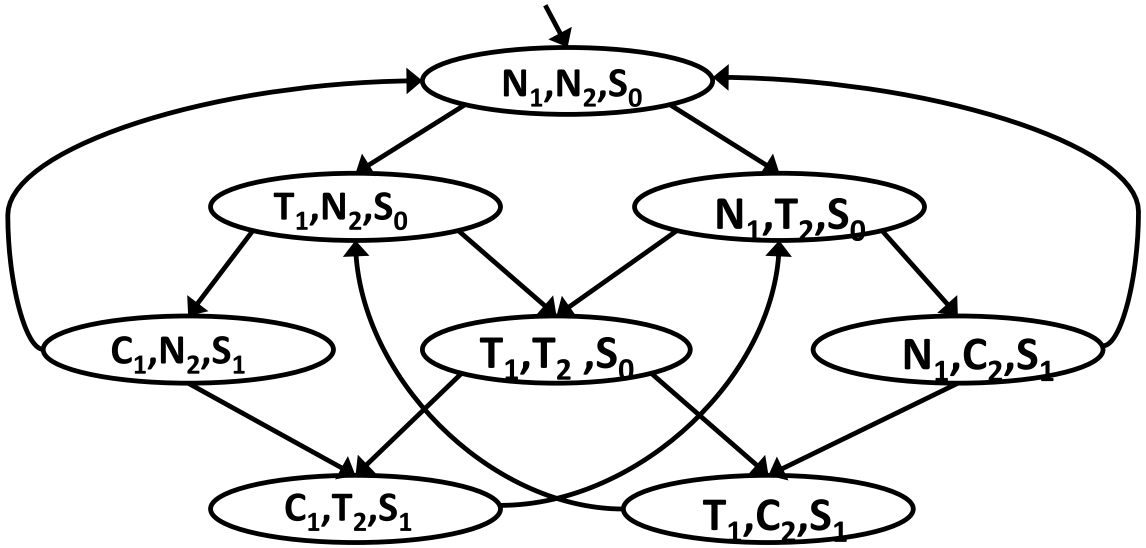
- Does it hold that $K \vDash \varphi$?
  - Property 2: φ := $\mathbf{AG}\neg(T_1 \wedge T_2)$ ✗

Example – Mutual Exclusion



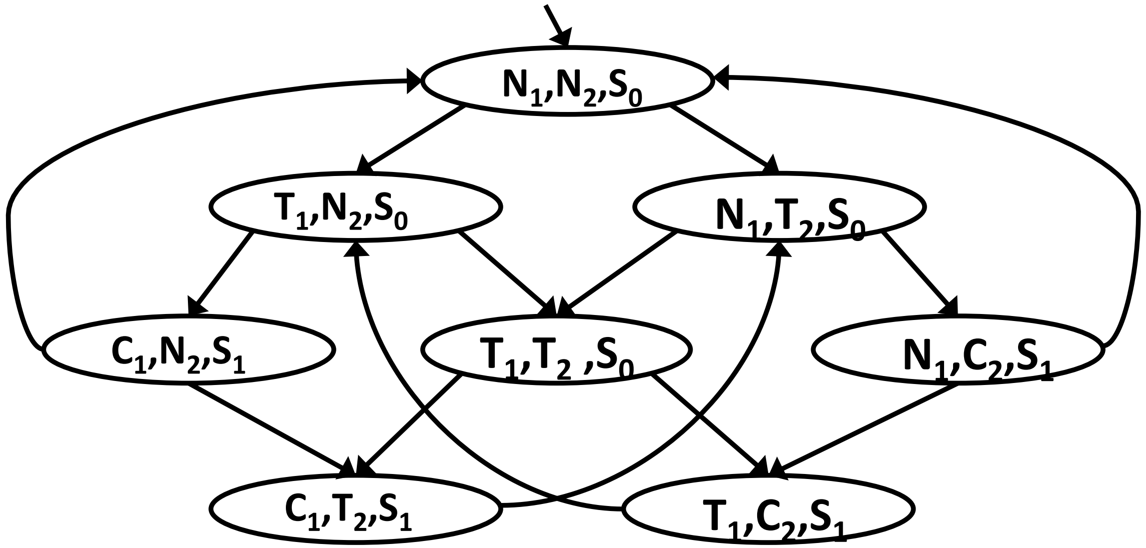
- Does it hold that $K \vDash \varphi$?
  - Property 3: $\varphi := \mathbf{AG}\ \mathbf{EF}\ (N_1 \wedge N_2 \wedge S_0)$

Example – Mutual Exclusion



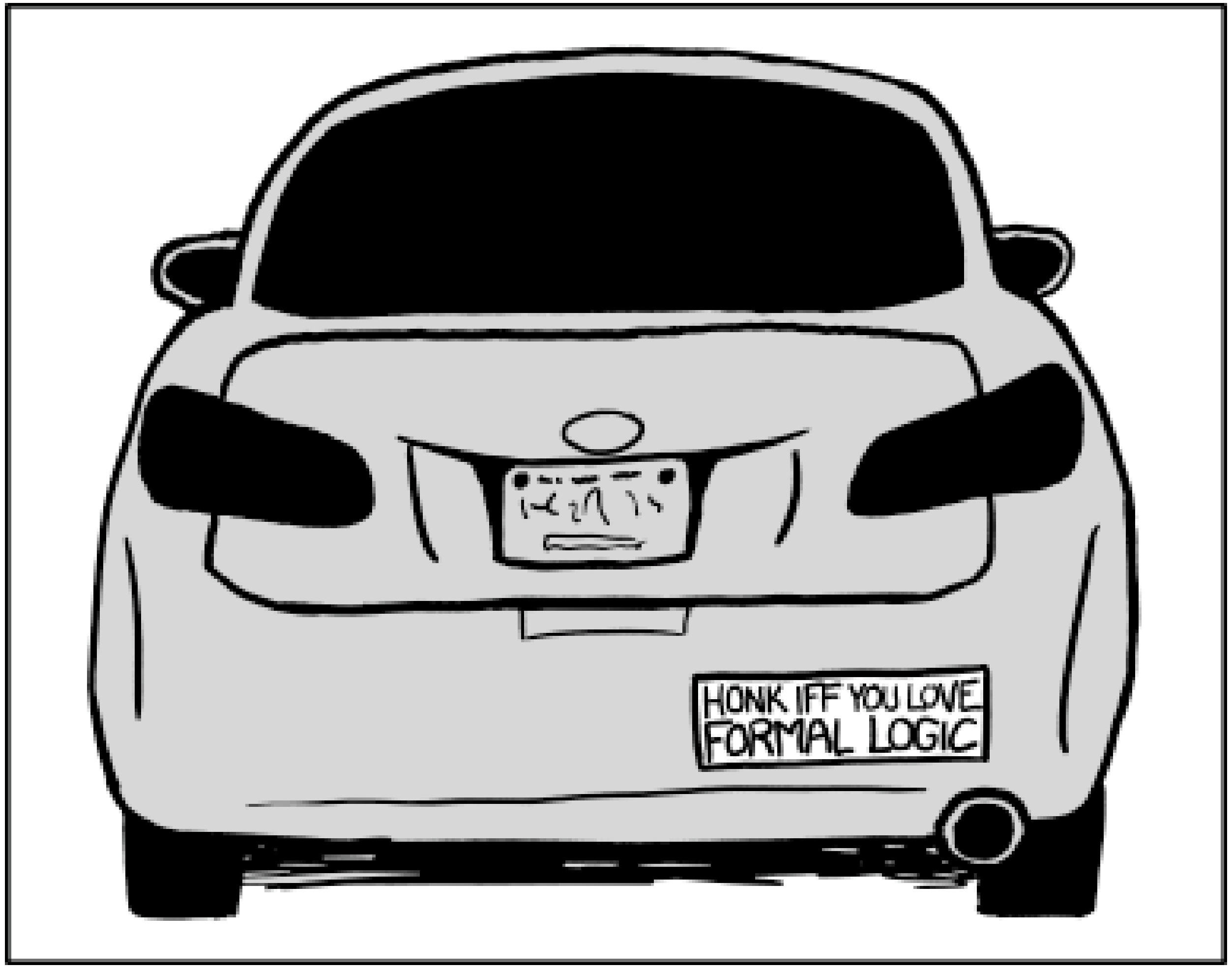
- Does it hold that $K \vDash \varphi$?
  - Property 3: $\varphi := \mathbf{AG}\ \mathbf{EF}\ (N_1 \wedge N_2 \wedge S_0)$
- No matter where you are there is always a way to get to the initial state (restart)

Example – Mutual Exclusion



- Does it hold that $K \vDash \varphi$?
  - Property 3: $\varphi := \mathbf{AG}\ \mathbf{EF}\ (N_1 \wedge N_2 \wedge S_0)$ ✓
- No matter where you are there is always a way to get to the initial state (restart)

Thank You



https://xkcd.com/1033/