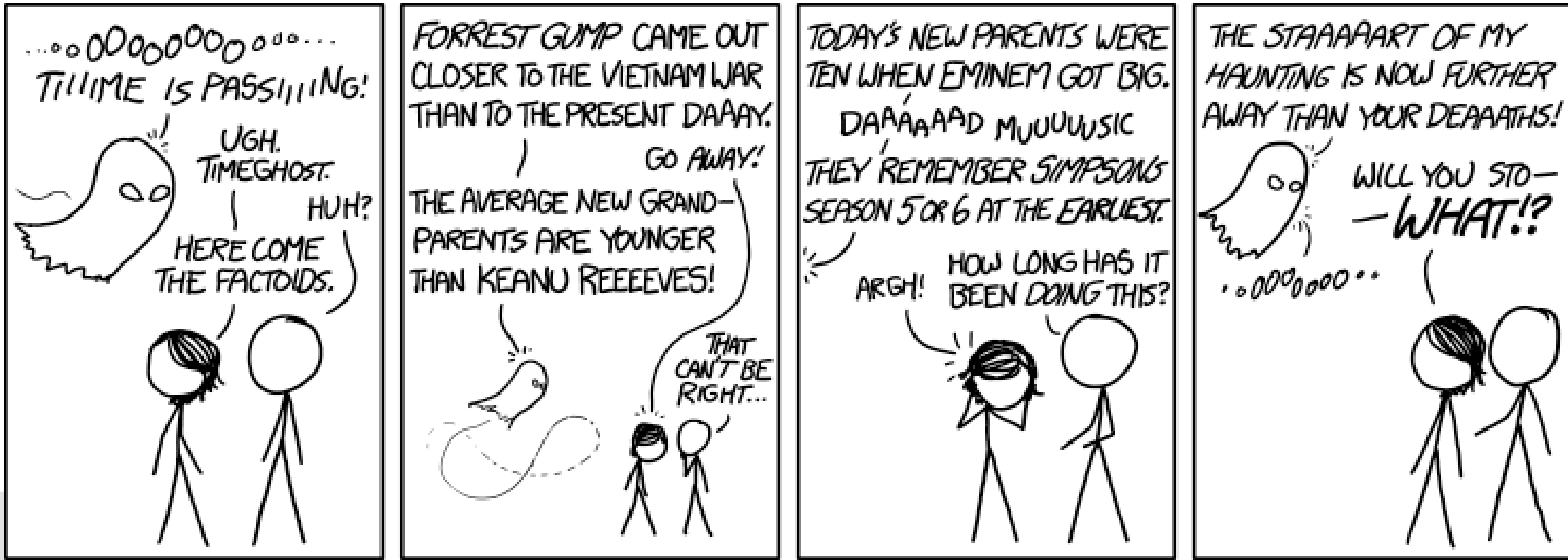
Logic and Computability

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. “
  - p… there is coffee, q… there is cake, r… the workshop is a success
  - $p \wedge q \rightarrow r$

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**. "
  - 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
  - E.g.: The system has to satisfy a property **eventually**.
  - A certain signal has to be high in the **next 5 time steps**.
  - Event A can only happen **10 minutes after** Event B.
- Temporal Logic allows reasoning over system's executions.
  - Introduce **temporal operators**, used additionally to logical operators
- Model Checking
  - Checks whether a model of a **system** meets a given **specification**
  - Specification typically **expressed in temporal logic.**

Outline

- Temporal Logic Formulas
  - Semantics of temporal operators
    - Intuitive explanation
  - Model natural language sentences via temporal logic formulas
- Evaluating System’s Executions
  - Definition of Kripke structures
  - Checking execution paths w.r.t. temporal logic formulas
- Evaluating Systems
  - Semantics of path operators
    - Intuitive explanation
  - Checking Kripke structures w.r.t. temporal logic formulas

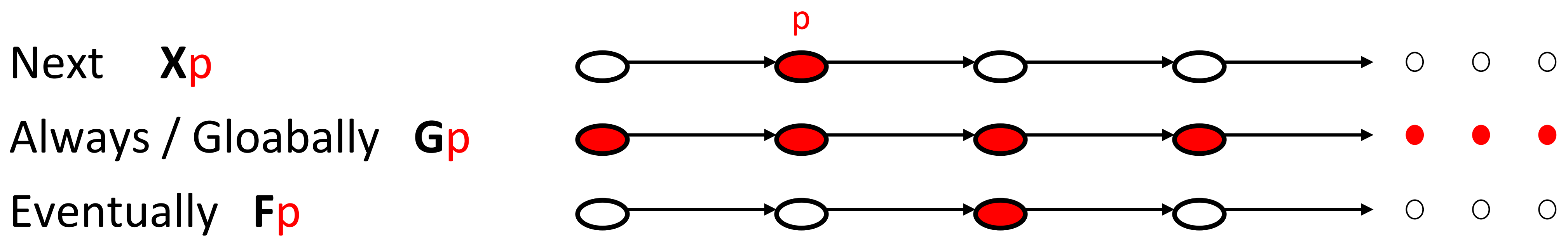
Learning Outcomes

After this lecture…

1. students can explain the semantic of the temporal operators (X,G,F, and U) and the path operators (A and E).
2. students can model natural language sentences via temporal logic.
3. students can define Kripke structures.
4. students can check whether an execution trace satisfies a temporal logic formula.
5. students can check whether a Kripke structure satisfies a temporal logic formula.

Temporal Operators

- Describe properties that hold along an execution path



- A state s satisfies the formula Xp if p is true in the next state.
- A state s satisfies the formula Gp if p is true in every state along the trace.
- A state s satisfies the formula Fp if p is true in s or in a subsequent state along the trace.

Translate in Temporal Logic

Temporal Operators
X… next
G… globally
F… eventually

- r… there is a request, g… arbiter gives grant

- "If there is a request, the arbiter gives a grant in the **next time step**. "
  - $\mathbf{G}(r \to \mathbf{X}g)$

- "If there is a request, the arbiter gives a grant within the **next 2 time steps**."
  - $\mathbf{G}(r \to (\mathbf{X}g \vee \mathbf{XX}g))$

- "If there is a request, the arbiter gives a grant **eventually**. "
  - $\mathbf{G}(r \to \mathbf{F}g)$

Temporal Operators

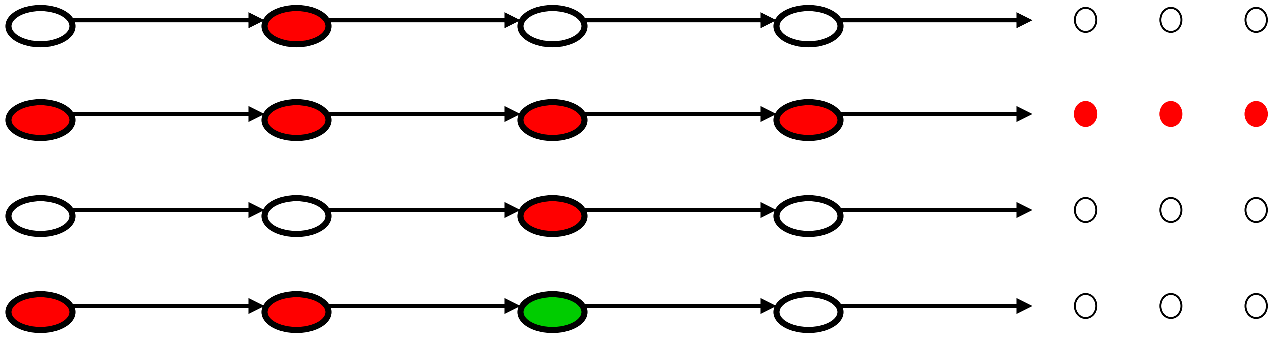
- Describe properties that hold along an execution path

Next Xp

Always / Gloabally Gp

Eventually Fp

Until pUq



Translate in Temporal Logic

Temporal Operators
X... next
G... globally
F... eventually
U... Until

- "The request is high until the arbiter gives a grant."
  - r... request is high, g... the system gives a grant
  - G(r **U** g)

- "The system is in the error state until the temperature is low or the system is turned off."
  - e... system is in error state, l... temperature is low, o... system is turned off
  - $G(e\ \mathbf{U}\ (\mathrm{l} \vee o))$

Translate in Temporal Logic

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

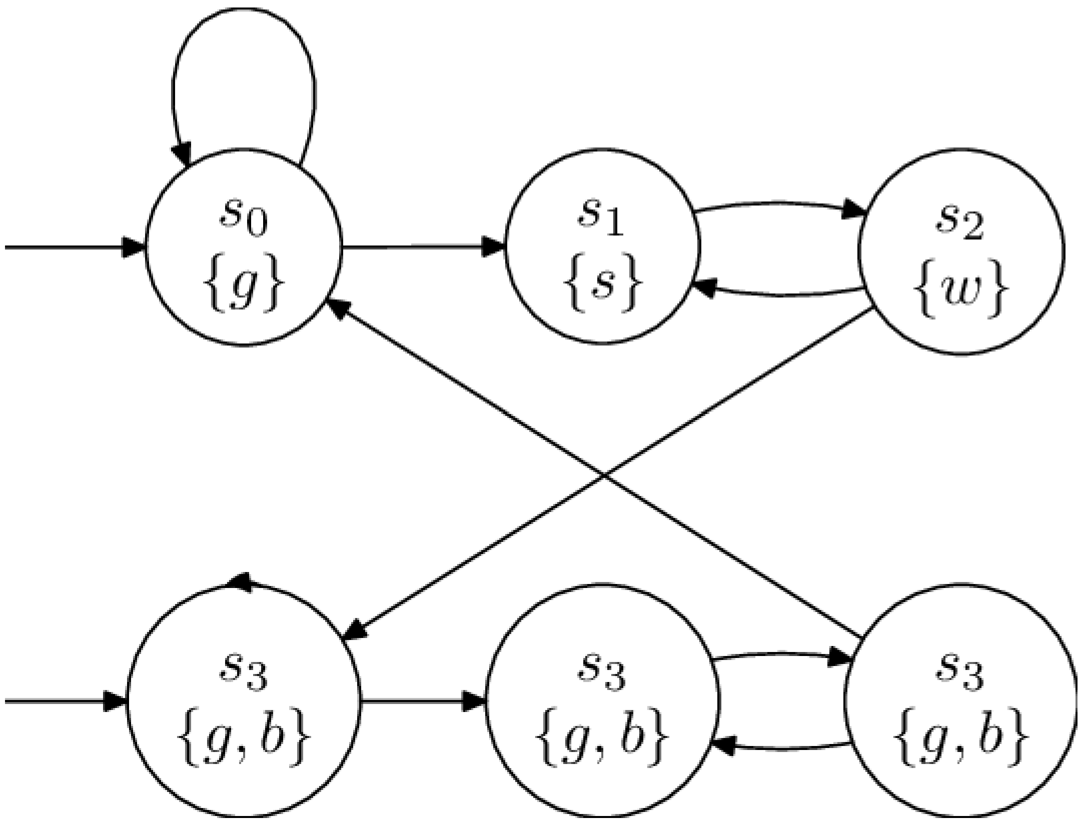
- "The system gives a grant infinitely often."
  - g… the system gives a grant
  - $\mathbf{GF}(\mathrm{g})$
- "The system sends a request finitely often."
  - r… system sends a request
  - $\mathbf{FG}(\neg \mathrm{r})$

Outline

- Temporal Logic Formulas
  - Semantics of temporal operators
    - Intuitive explanation
  - Model natural language sentences via temporal logic formulas
- Evaluating System’s Executions
  - Definition of Kripke structures
  - Checking execution paths w.r.t. temporal logic formulas
- Evaluating Systems
  - Semantics of path operators
    - Intuitive explanation
  - Checking Kripke structures w.r.t. temporal logic formulas

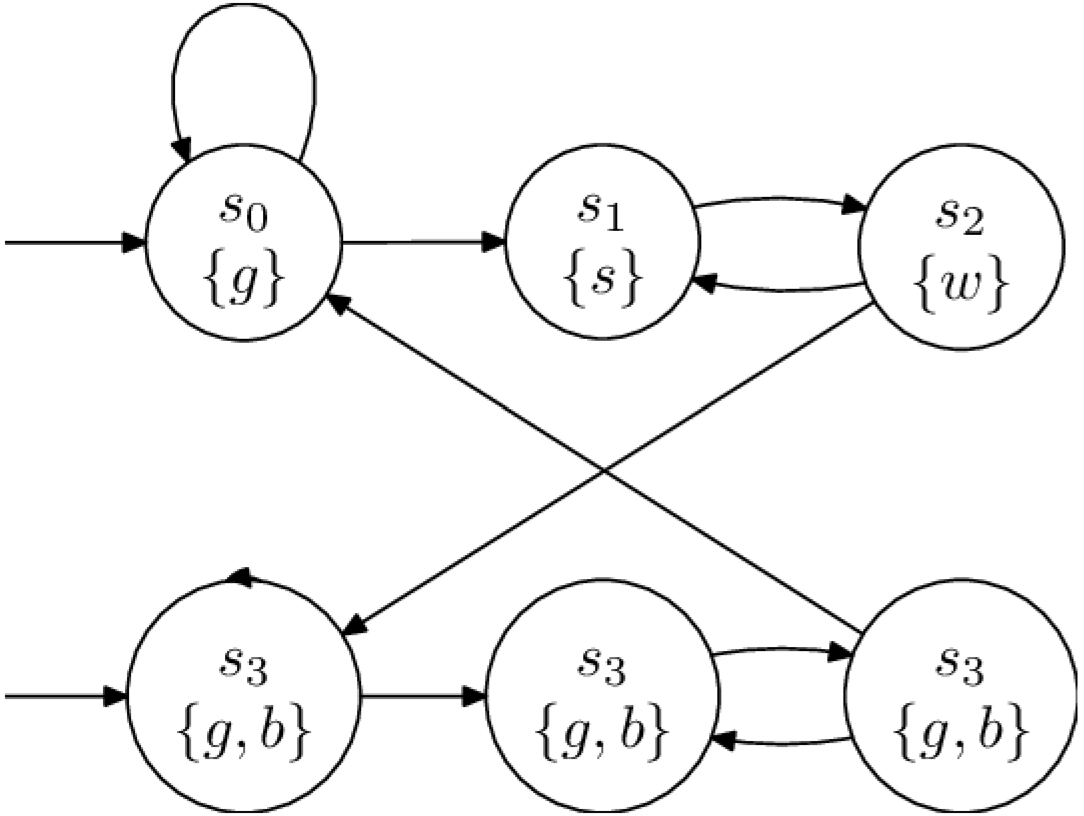
Kripke Structures

- Transition system with **labelling function**
  - Assigns set of atomic propositions to each state



Kripke Structures

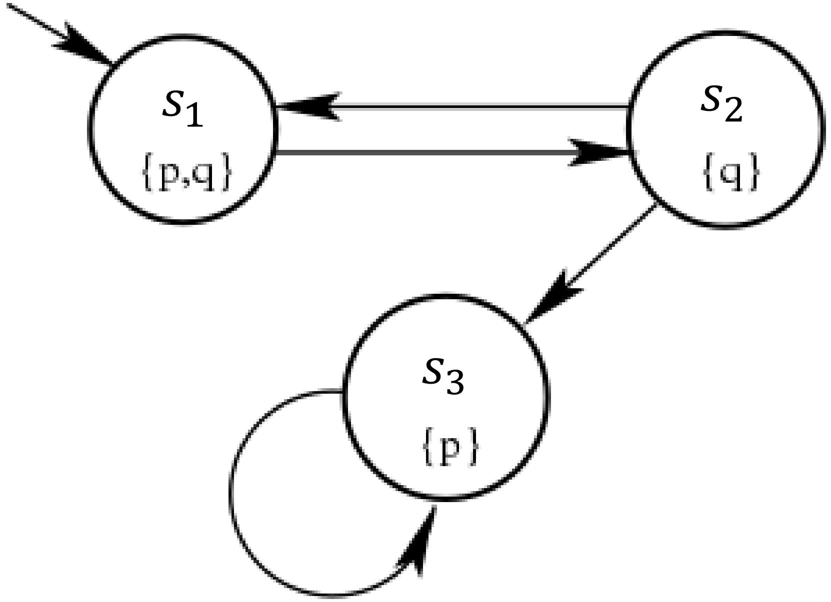
- A Kripke Structure is a tuple $K = (S, S_0, R, L)$
  - Finite Set of States S
  - Set of Initial States $S_0 \subseteq S$
  - Transition Relation $R \subseteq S \times S$
  - Labeling function: $L : S \to 2^{AP}$



Kripke Structure - Example

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

- How does the graph look like?

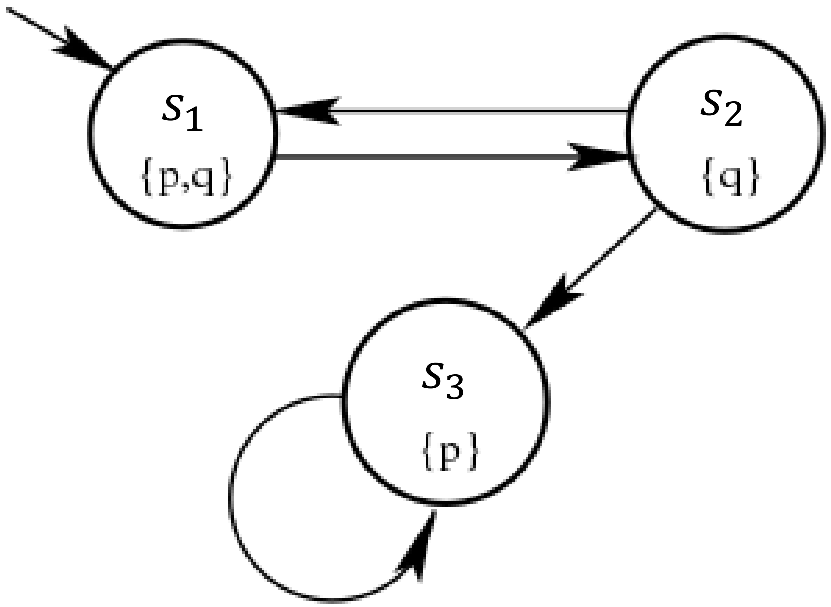


Paths and Words over Kripke Structures

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

- A path is a is a sequence of states $\rho = s_1, s_2, s_3 \dots$ s.t. for each i > 0, $R(s_i, s_{i+1})$ hold

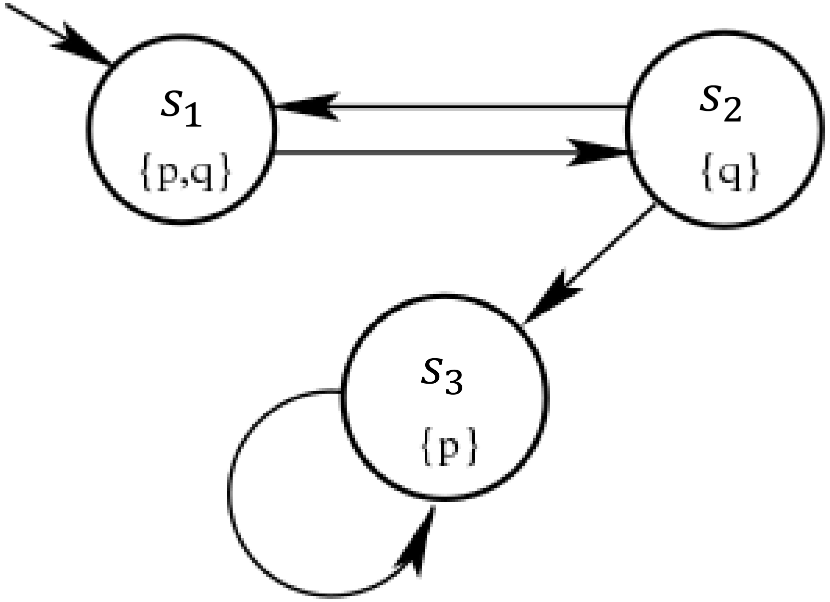
- The *word* of a path ρ is a sequence of sets of atomic propositions $w = L(s_1), L(s_2), L(s_3), \dots$



Paths and Words over Kripke Structures

Example:

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



Temporal Operators

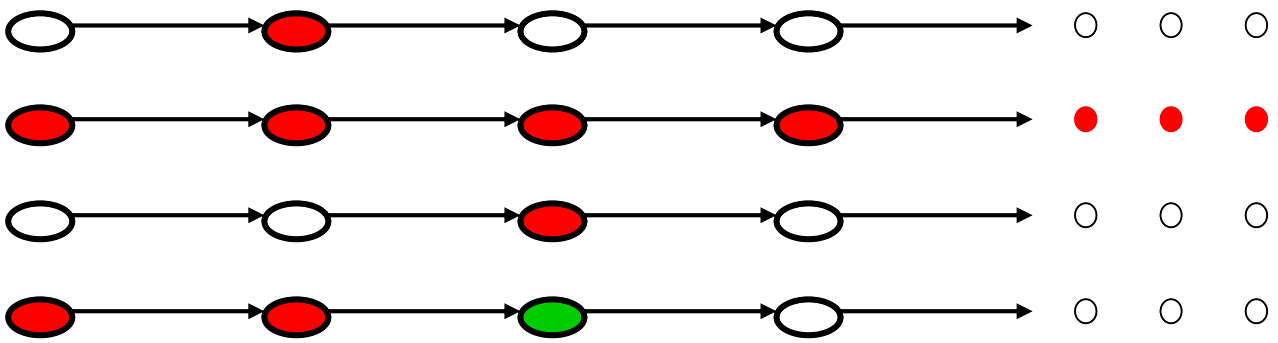
- Describe properties that hold along an execution path of a Kripke structure

Next Xp

Always / Gloabally Gp

Eventually Fp

Until pUq



- A state s satisfies the formula pUq if either q is true in s or p holds in every state (starting from s) until q holds.

Evaluating Traces – Example 1

Given:

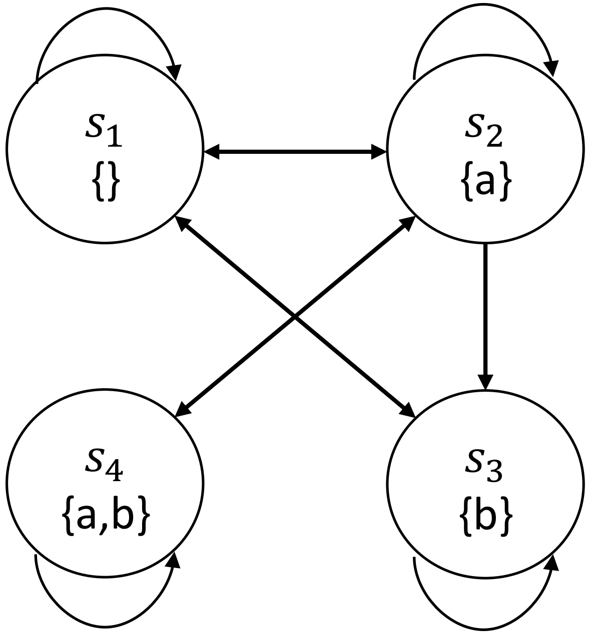
- Trace $\rho = s_1 s_2 s_2 s_3 s_1 s_2 s_4 s_4 s_4 s_4 \ldots$
- Temporal logic formula $\varphi = Xa \vee a\ U\ b$

Does the trace ρ satisfy the formula φ?

Step 1: Compute the word w of ρ

w = {} {a}, {a}, {b}, {}, {a}, $\{a,b\}, \{a,b\}\{a,b\} \ldots$

Step 2: Using w, evaluate φ over ρ



> If the first state s of the trace ρ satisfies φ (i. e. $s \vDash \varphi$), we say that the trace satisfies φ (i. e., $\rho \vDash \varphi$).

Evaluating Traces – Example 1

Given:

- Word $w = \{\}\,\{a\},\{a\},\{b\},\{\},\{a\},\{a,b\}^{\omega}$
- Formula $\varphi = Xa \vee a\;U\;b$
- Evaluate each subformula for each step (=state along the trace)

$\{a,b\}^{\omega}$ ... $\{a,b\}$ infinitely many times

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

Since the first state s of ρ satisfies φ ($s \vDash \varphi$), it holds that ρ satisfies φ ($\rho \vDash \varphi$).

Evaluating Traces – Example 2

Given:

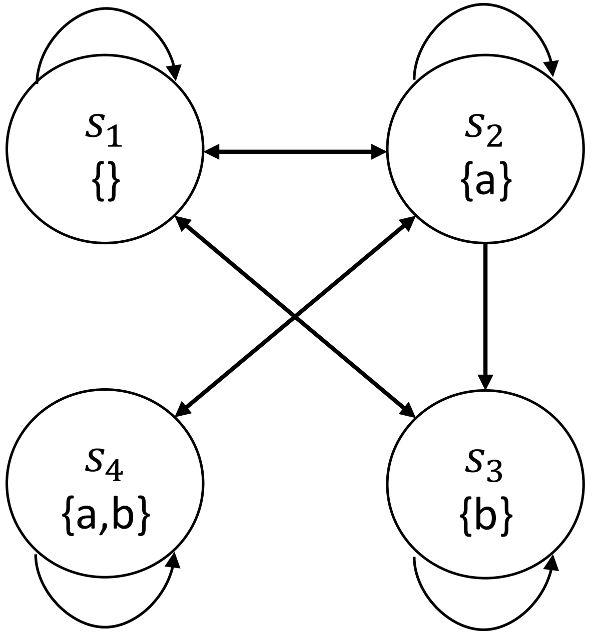
- Trace $\rho = s_1 s_2 s_2 s_4^{\omega}$
- Temporal logic formula $\varphi = Ga \rightarrow Fb$
- Does the trace ρ satisfy the formula φ?

Step 1: Compute the word w of ρ

$\mathrm{w} = \{\}\ \{\mathrm{a}\}, \{\mathrm{a}\}, \{\mathrm{a,b}\}^{\omega}$

Step 2: Using w, evaluate φ over ρ

> If the first state s of the trace ρ satisfies φ (i.e. $s \vDash \varphi$), we say that the trace satisfies φ (i.e., $\rho \vDash \varphi$).



Evaluating Traces – Example 2

Given:

- Word $w = \{\}\ \{a\}, \{a\}, \{a, b\}^{\omega}$
- Formula $\varphi = Ga \rightarrow Fb$
- Does w satisfy φ?

$\{a, b\}^{\omega}$ ... $\{a, b\}$ infinitely many times

Step	0	1	2	ω
a	0	1	1	1
b	0	0	0	1
Ga	0	1	1	1
Fb	1	1	1	1
$Ga \rightarrow Fb$	1	1	1	1

Since the first state s of ρ satisfies φ ($s \vDash \varphi$), it holds that ρ satisfies φ ($\rho \vDash \varphi$).

Outline

- Temporal Logic Formulas
  - Semantics of temporal operators
    - Intuitive explanation
  - Model natural language sentences via temporal logic formulas
- Evaluating System’s Executions
  - Definition of Kripke structures
  - Checking execution paths w.r.t. temporal logic formulas
- Evaluating Systems
  - Semantics of path operators
    - Intuitive explanation
  - Checking Kripke structures w.r.t. temporal logic formulas

Evaluating Systems

- Check whether a Kripke structure K satisfies a formula $\boldsymbol{\varphi}$
- K satisfies φ, if all initial states of $s_0 \in S_0$ satisfy φ
  - $\boldsymbol{K \vDash \varphi}$ **if and only if** $\boldsymbol{\forall s \in S_0 : s \vDash \varphi}$
- We need **path quantifiers** to reason about execution paths of systems.

Computation Tree Logic – CTL*

- Extends propositional logic with
  - **Temporal Operators,** and
  - **Path Quantifiers**
    - A for all paths starting from s have property $\boldsymbol{\varphi}$
    - E there exists a path starting from s have property $\boldsymbol{\varphi}$

Computation Tree Logic – CTL*

- Extends propositional logic with
  - **Temporal Operators,** and
  - **Path Quantifiers**

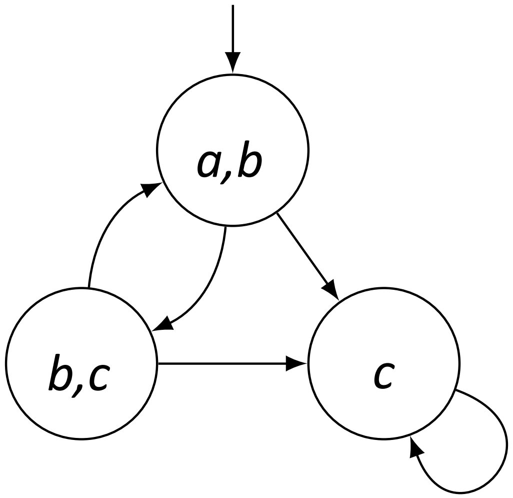
- Kripke structure K satisfies a CTL* formula φ, if all its initial states $s_0 \in S_0$ satisfy φ.
  - $\boldsymbol{K \vDash \varphi}$ **iff** $\boldsymbol{\forall s_0 \in S_0 : s_0 \vDash \varphi}$

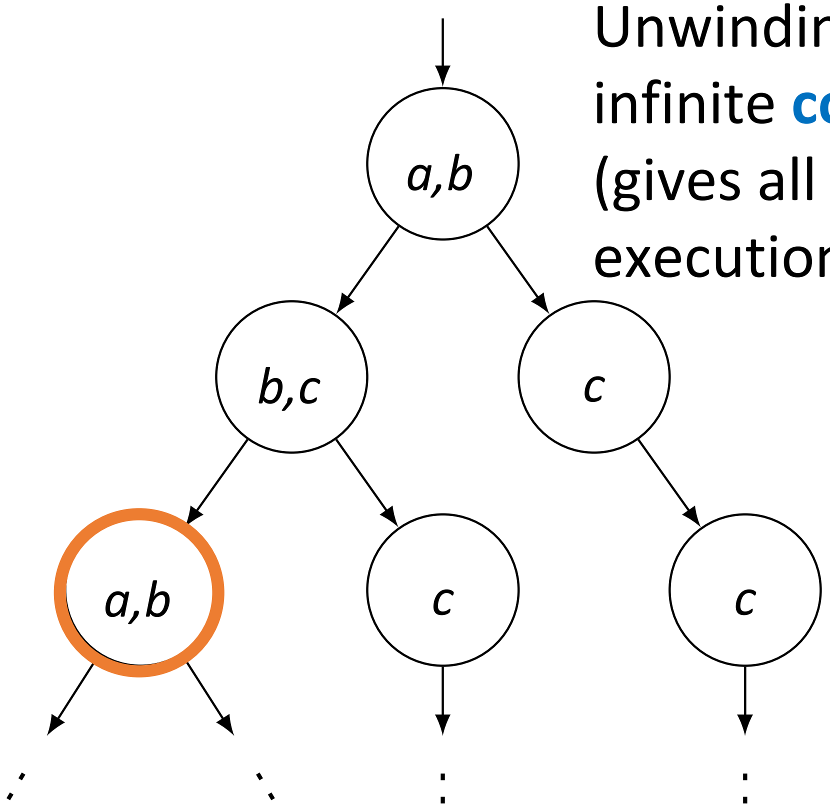
Temporal Operators
X… next
G… globally
F… eventually
U… until
Path Quantifiers
A for **all** paths
E there **exists** a path

Evaluating Kripke Structures – Example 1

- Does the Kripke structure K satisfy $\varphi_1 = EXX(a \wedge b)$?

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



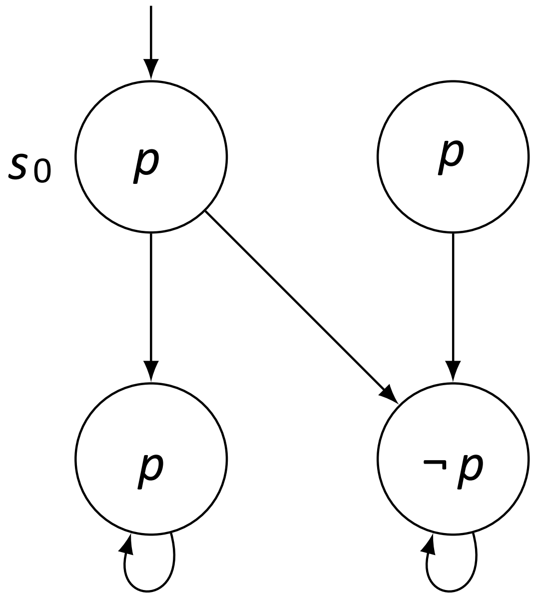


Unwinding of K into infinite **computation tree** (gives all possible execution paths)

$\boldsymbol{s_0 \vDash \mathrm{EXX}\ (a \wedge b)}$.
Thus,
$\boldsymbol{K \vDash \mathrm{EXX}\ (a \wedge b)}$.

Evaluating Kripke Structures – Example 1

- Does the Kripke structure K one of the following formulas?
  - $\varphi_1 = EXp$
  - $\varphi_2 = AXp$



$K \models EX\,p$

$K \not\models AXp$

Example – Mutual Exclusion

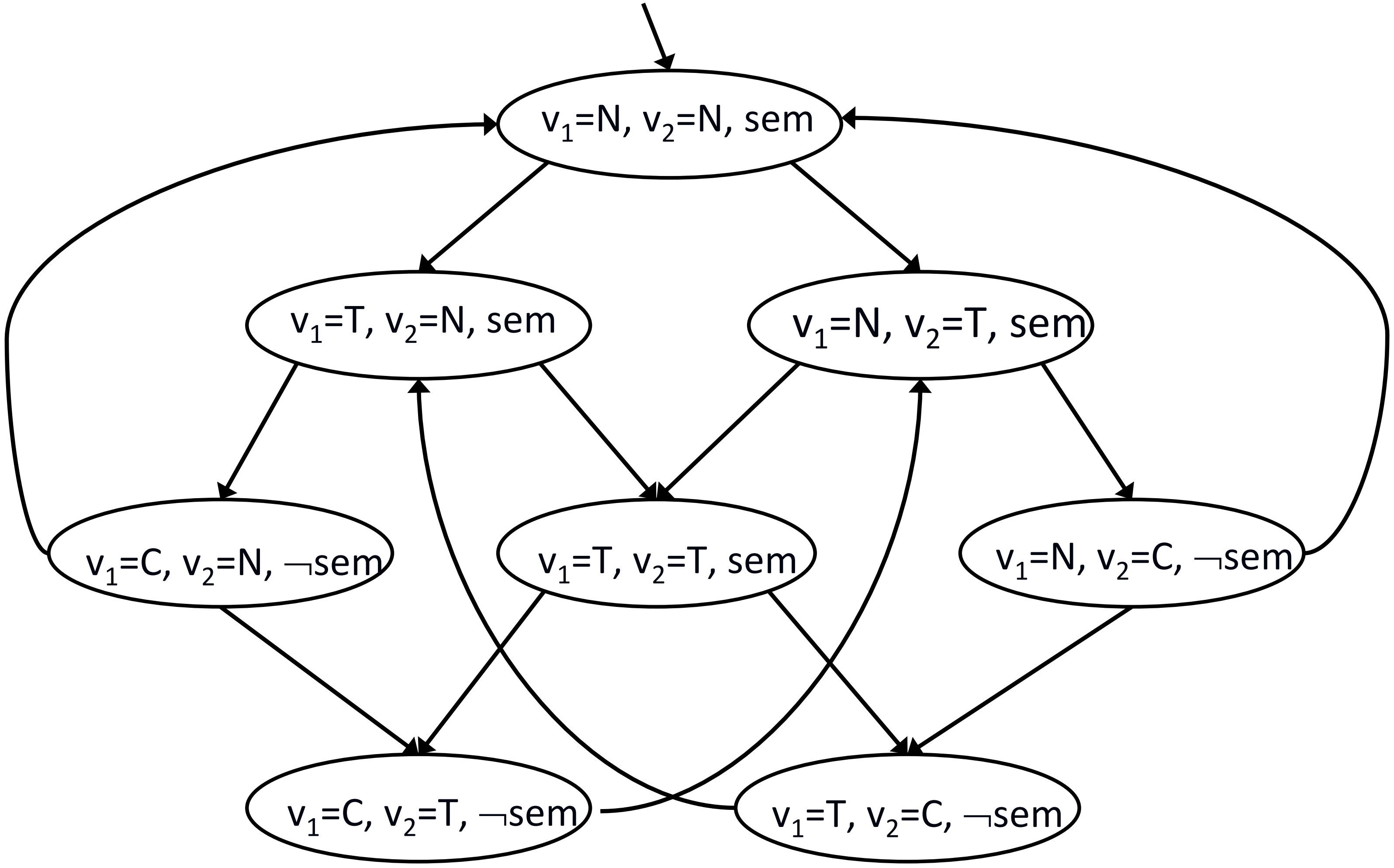
- Two processes P_1 and P_2 with a joint semaphor signal sem
- Each process P_i has a variable v_i describing its state:
  - $v_i = \mathrm{N}$ Non-critical
  - $v_i = \mathrm{T}$ Trying
  - $v_i = \mathrm{C}$ Critical

- Each process runs the following program

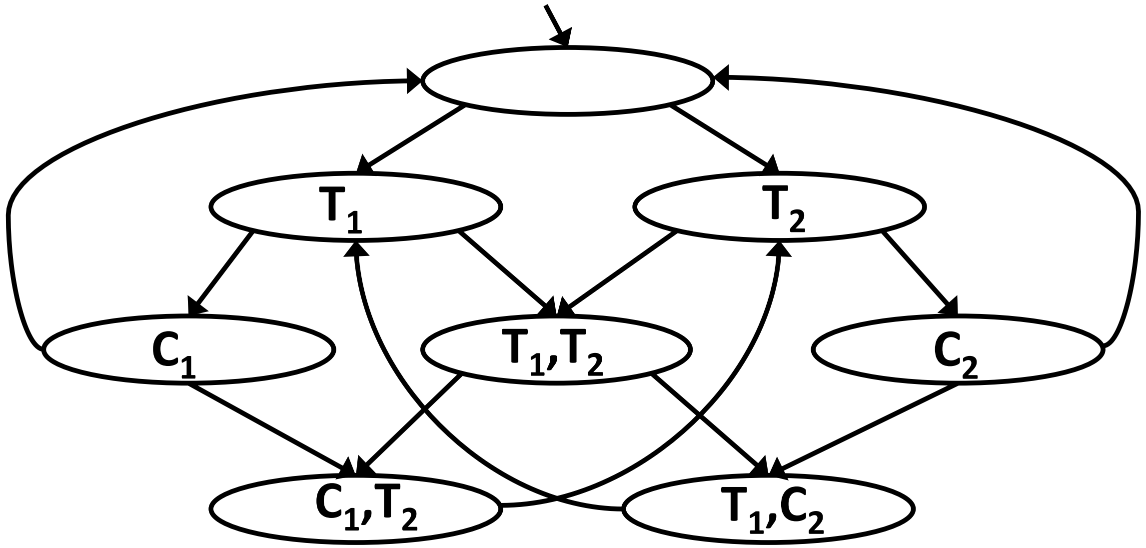
```
while (true) {
        if (v_i == N)  v_i = T;
        else if (v_i == T && sem)  { v_i = C; sem = 0; }
        else if (v_i == C)  {v_i = N; sem = 1; }
    }
```

Atomic action (→ each of the three if/else-if statements)

Example – Mutual Exclusion

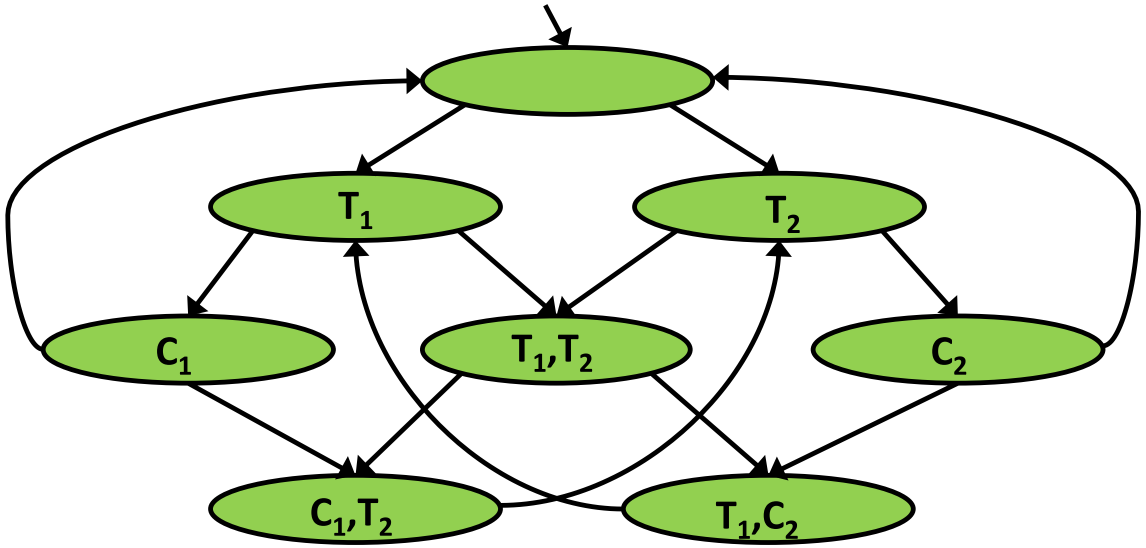


Example – Mutual Exclusion



- Simpler Representation

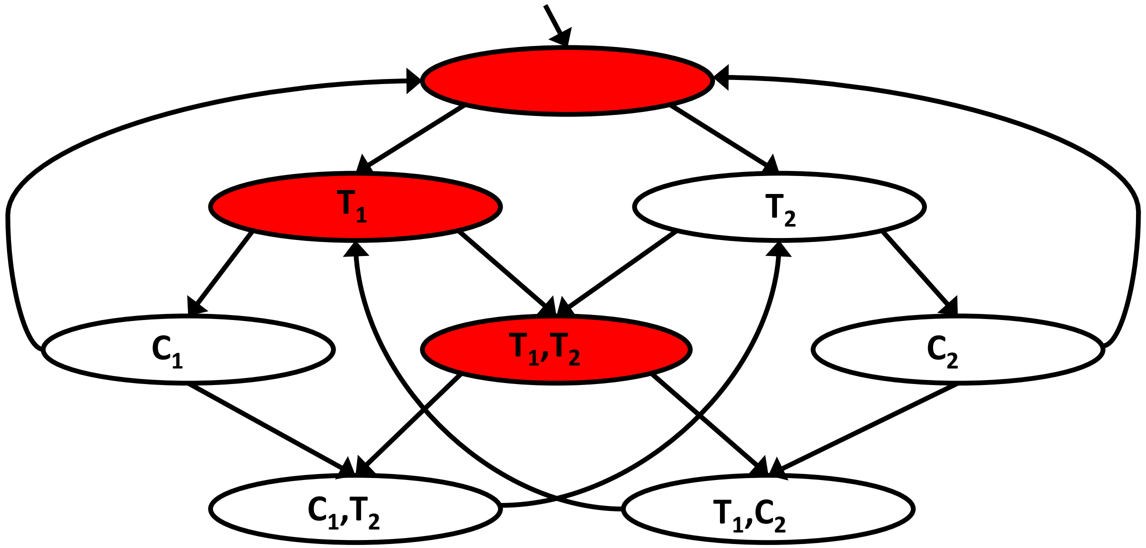
Example – Mutual Exclusion



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

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

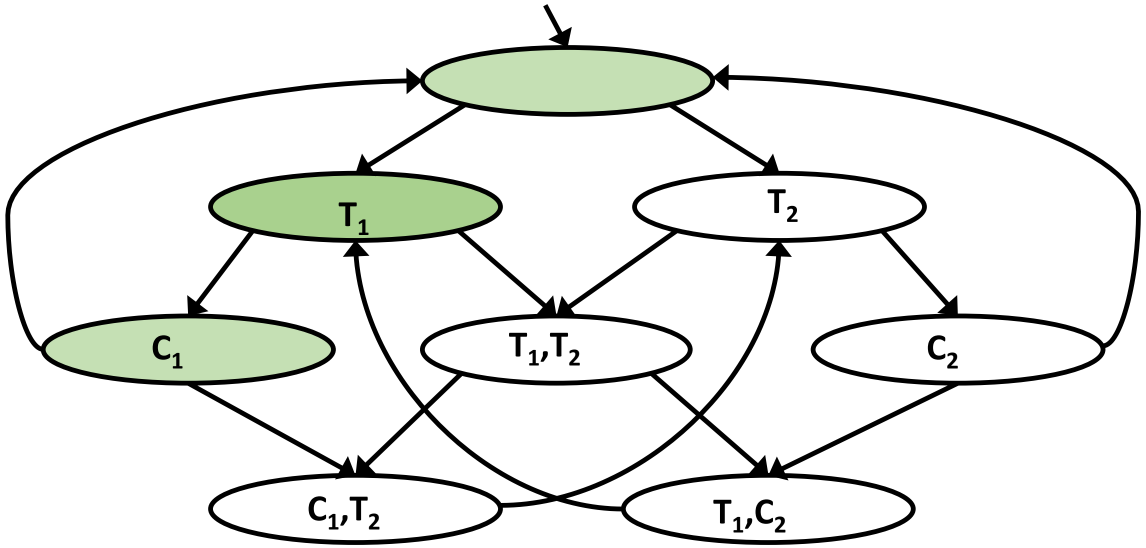
Example – Mutual Exclusion



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

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

Example – Mutual Exclusion

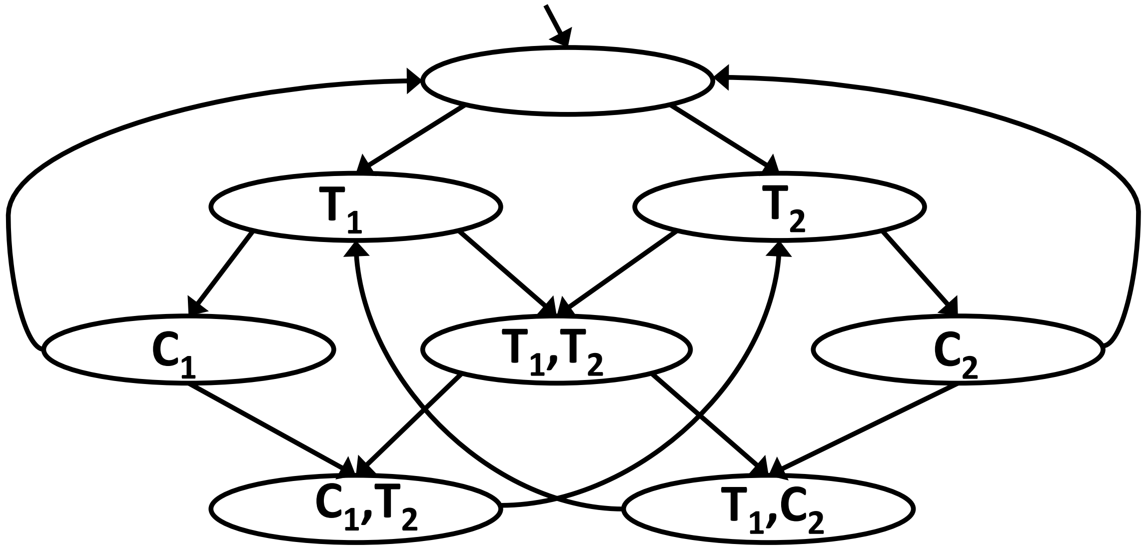


Temporal Operators
X… next
G… globally
F… eventually
U… until
Path quantifiers
A for **all** paths
E there **exists** a path

- Does it hold that $K \vDash \varphi$? ✓
  - Property 3: φ := **EG**$\neg(T_1 \wedge T_2)$

Example – Mutual Exclusion

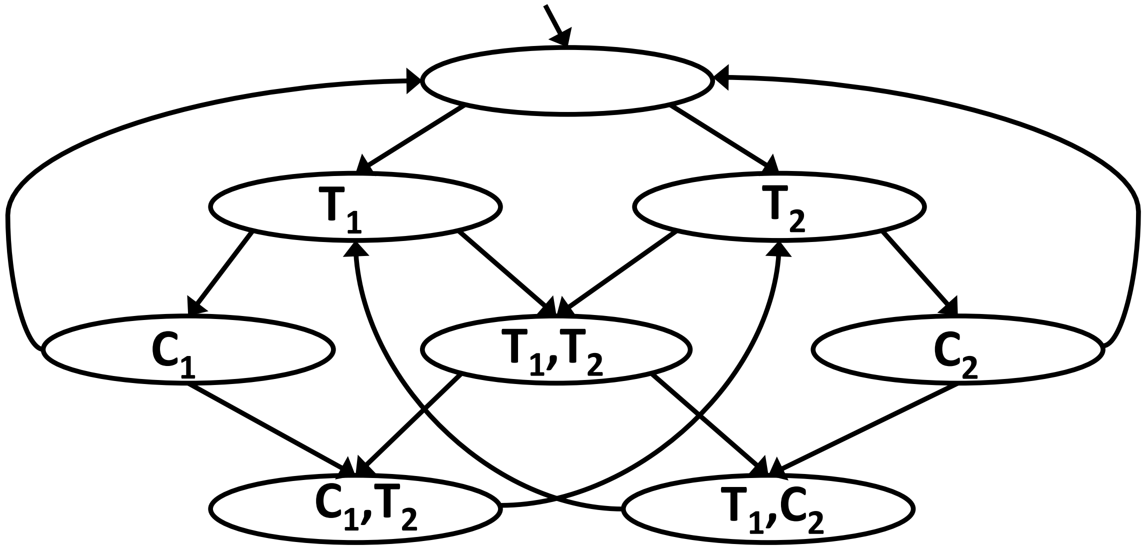
Temporal Operators
X… next
G… globally
F… eventually
U… until
Path quantifiers
A for **all** paths
E there **exists** a path



- Does it hold that $K \vDash \varphi$?
  - Property 4: $\varphi := \mathbf{AG}\ \mathbf{EF}\ (T_1)$

No matter where you are it is always possible to reach the state labeled with T_1.

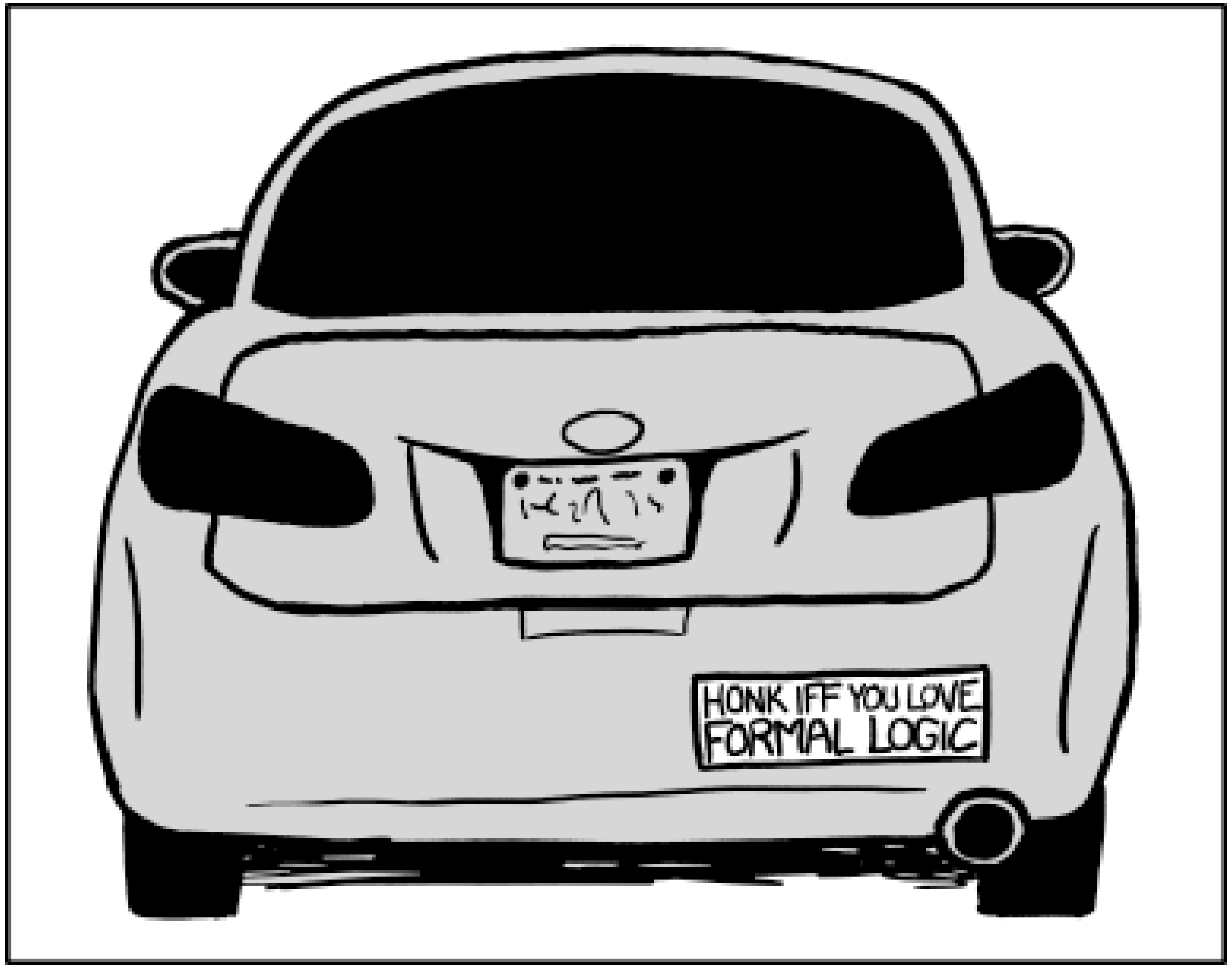
Example – Mutual Exclusion



- Does it hold that $K \models \varphi$? ✓
  - Property 4: $\varphi := \mathbf{AG}\ \mathbf{EF}\ (T_1)$

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

Thank You



https://xkcd.com/1033/