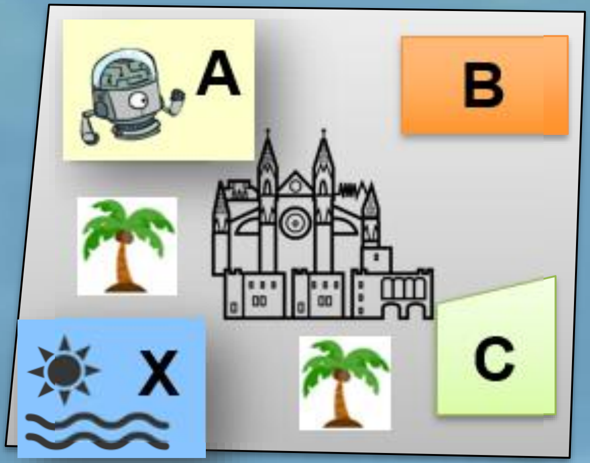
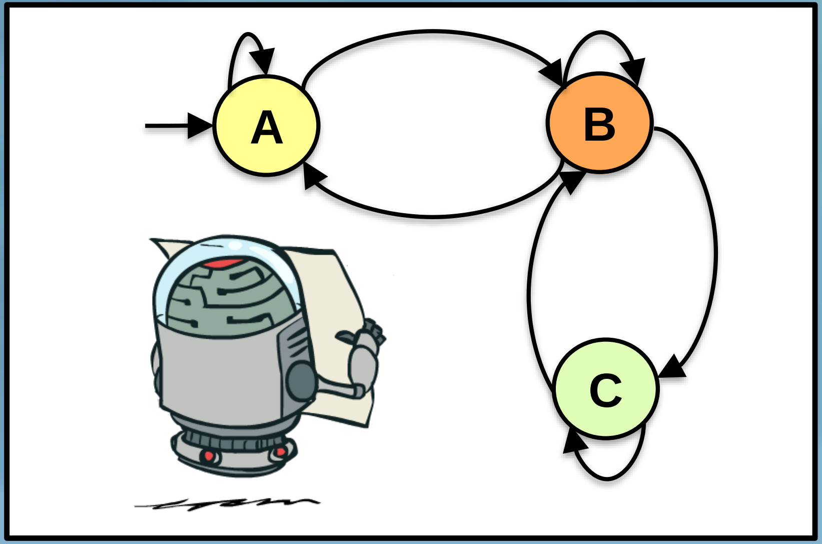
Graz University of Technology
Institute for Applied Information
Processing and Communications

Temporal Logic +
CTL Model Checking





Model Checking SS24

Bettina Könighofer

April 29, 2024

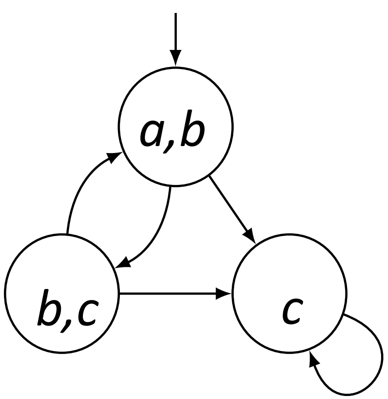
Plan for Today

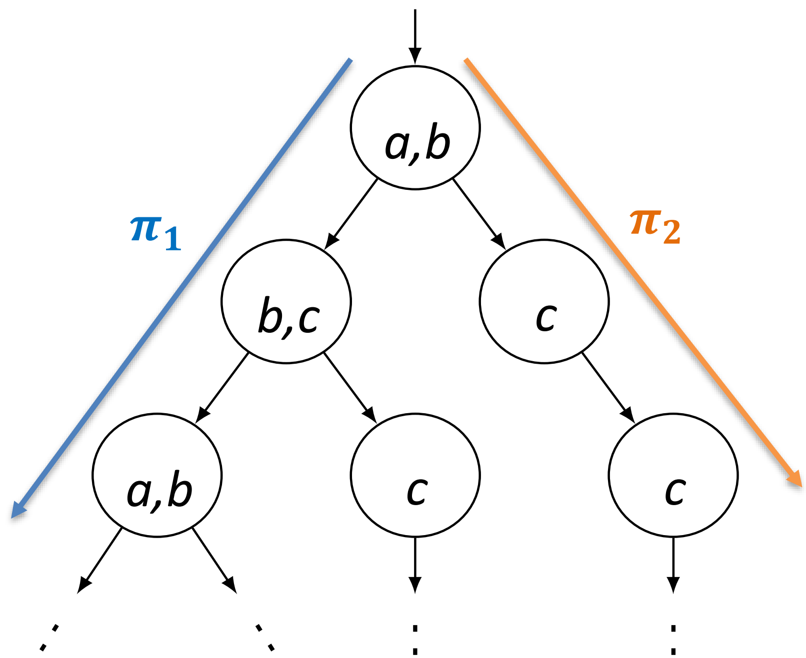
- Presentation of Homework and Recap of Temporal Logic
- Properties of CTL and LTL
- CTL Model Checking

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.



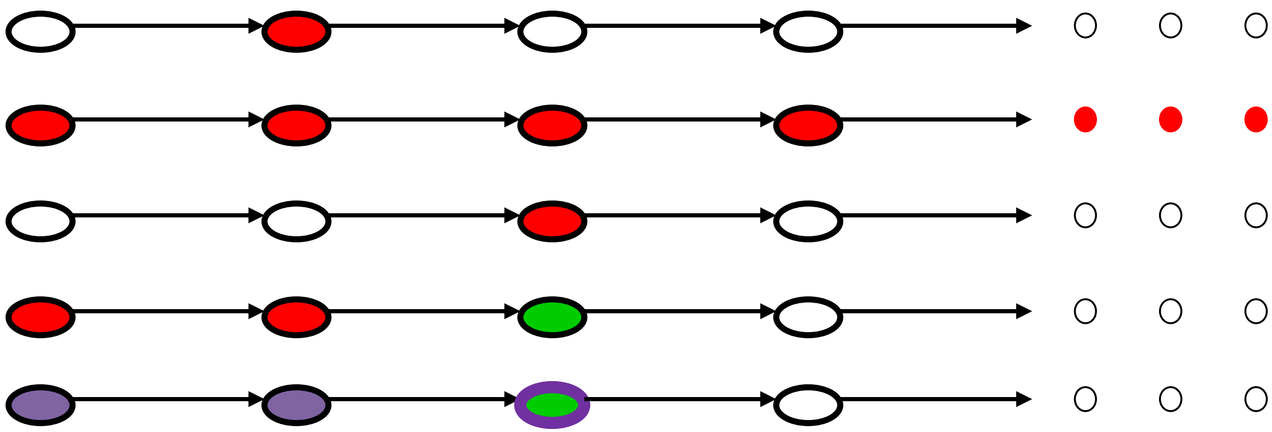


Propositional Temporal Logic

Temporal operators:

- Describe properties that hold along an infinite path π

Xp
Gp
Fp
pUq
pRq



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.

Linear Temporal Logic (LTL) - Syntax

LTL is the set of all state formulas.

State formulas:

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

Path formulas:

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

  where g_1 and g_2 are path formulas.

Computational Tree Logic (CTL) - Syntax

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

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

where f_1 and f_2 are state formulas

Illustration of CTL Semantics

EFg

"exists reachable state such that…"

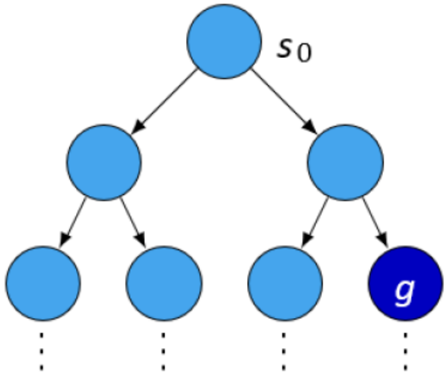
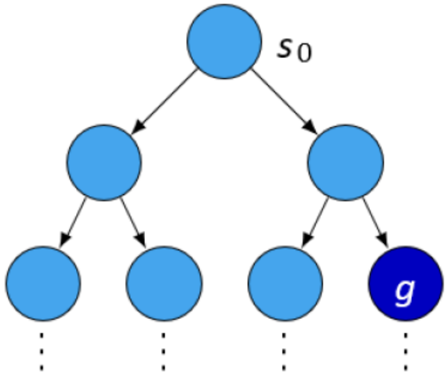
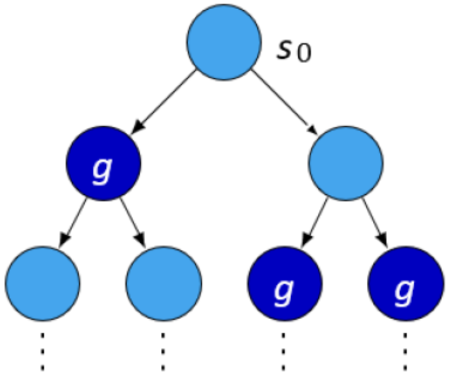


Illustration of CTL Semantics

EFg

“exists reachable state such that…”



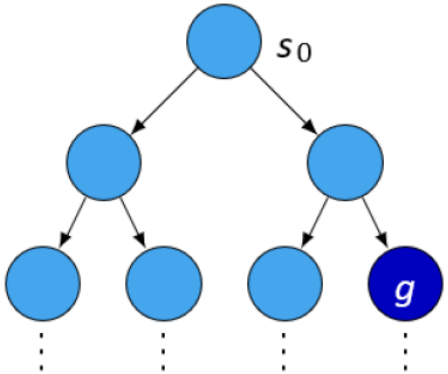


AFg

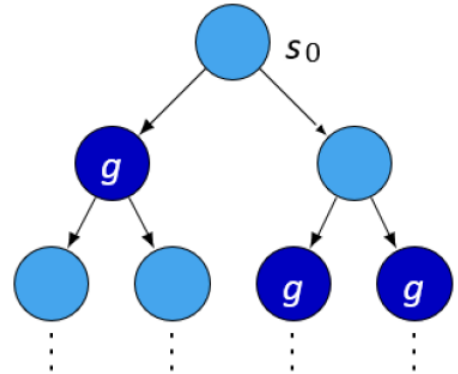
Illustration of CTL Semantics

EFg

"exists reachable state such that…"



AFg



EGg

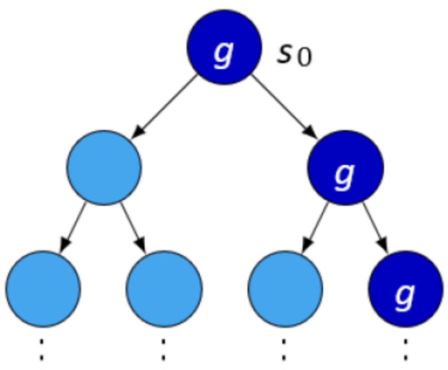
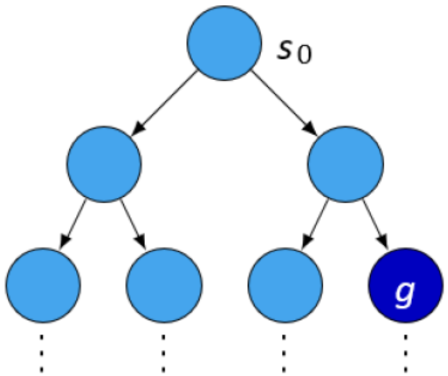


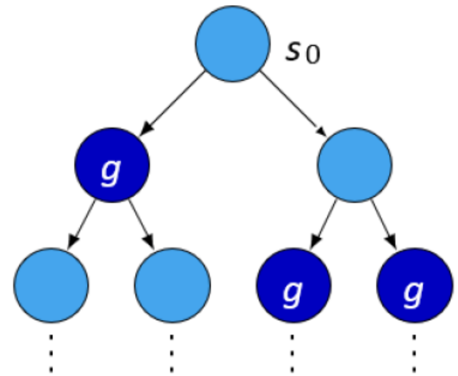
Illustration of CTL Semantics

EFg

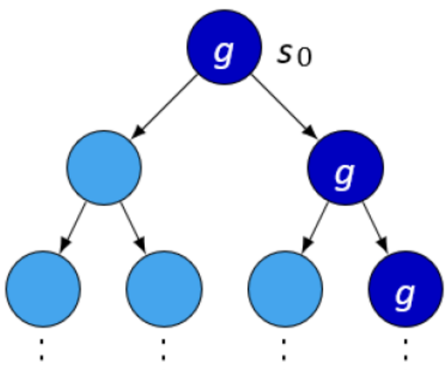
"exists reachable state such that…"



AFg

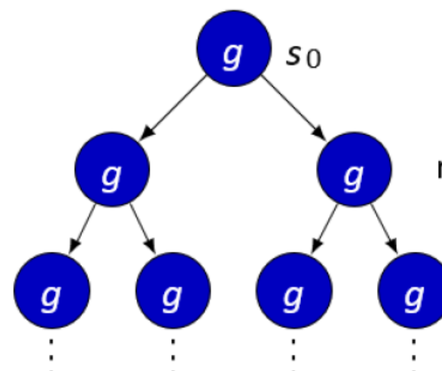


EGg



AGg

"all reachable states…"



Homework

1. "At any time, one can select ten cups of coffee and once selected, ten cups will always eventually be served unless an error occurs."

Homework

1. "At any time, one can select ten cups of coffee and once selected, ten cups will always eventually be served unless an error occurs."

$$AG\ (ten \rightarrow AF\ (served \vee error)\ )$$

Homework

2. “At any time, it is possible to eventually reach an error.”

Homework

2. "At any time, it is possible to eventually reach an error."

$$\varphi \coloneqq AG\,EF\,error$$

Homework

3. "Always, it will happen eventually that the coffee machine remains turned off forever."

Homework

3. "Always, it will happen eventually that the coffee machine remains turned off forever."

$$\varphi \coloneqq AFG\, off$$

Homework

4. “All reachable states can result in 10 cups of coffee.”

Homework

4. “All reachable states can result in 10 cups of coffee.”

$$\varphi \coloneqq AG\ EF\ (coffee)$$

Homework

5. It is never possible that the machine brews 5 cups of coffee in the current time step, and serves 5 more cups within the next 2 seconds.

Homework

5. It is never possible that the machine brews 5 cups of coffee in the current time step, and serves 5 more cups within the next 2 seconds.

$$\varphi \ := AG\neg(5cups \wedge X\ 5cups \wedge XX5cups)$$

Homework

6. The selected amount of coffee will be served within 6 seconds.

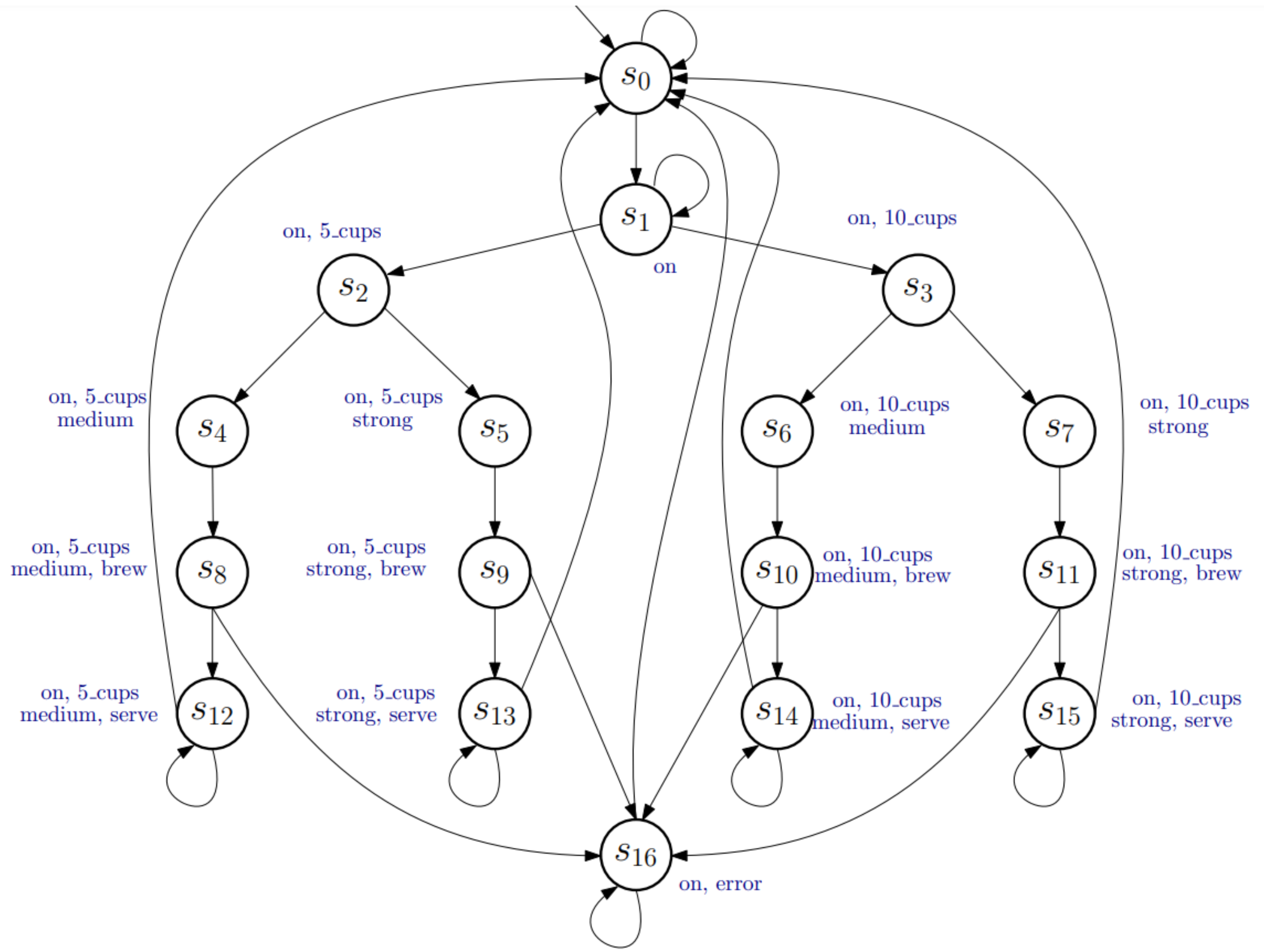
Homework

6. The selected amount of coffee will be served within 6 seconds.

$$\varphi \coloneqq AG(selected \rightarrow (Xserved \vee \cdots \vee XXXXXXserved))$$

Homework

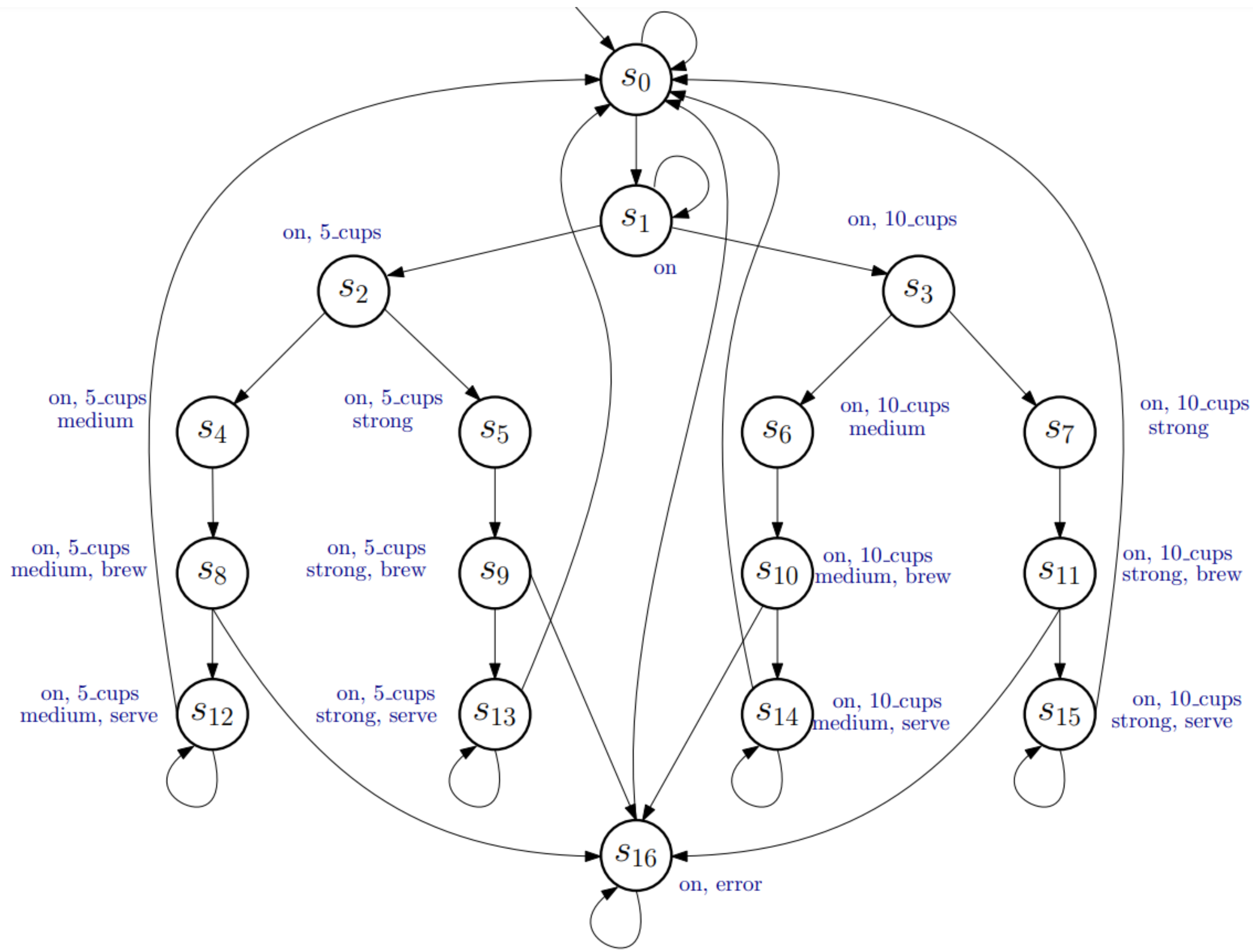
1. $\varphi := A\,((F\;serve)\;U\;(G\;\neg on))$



Homework

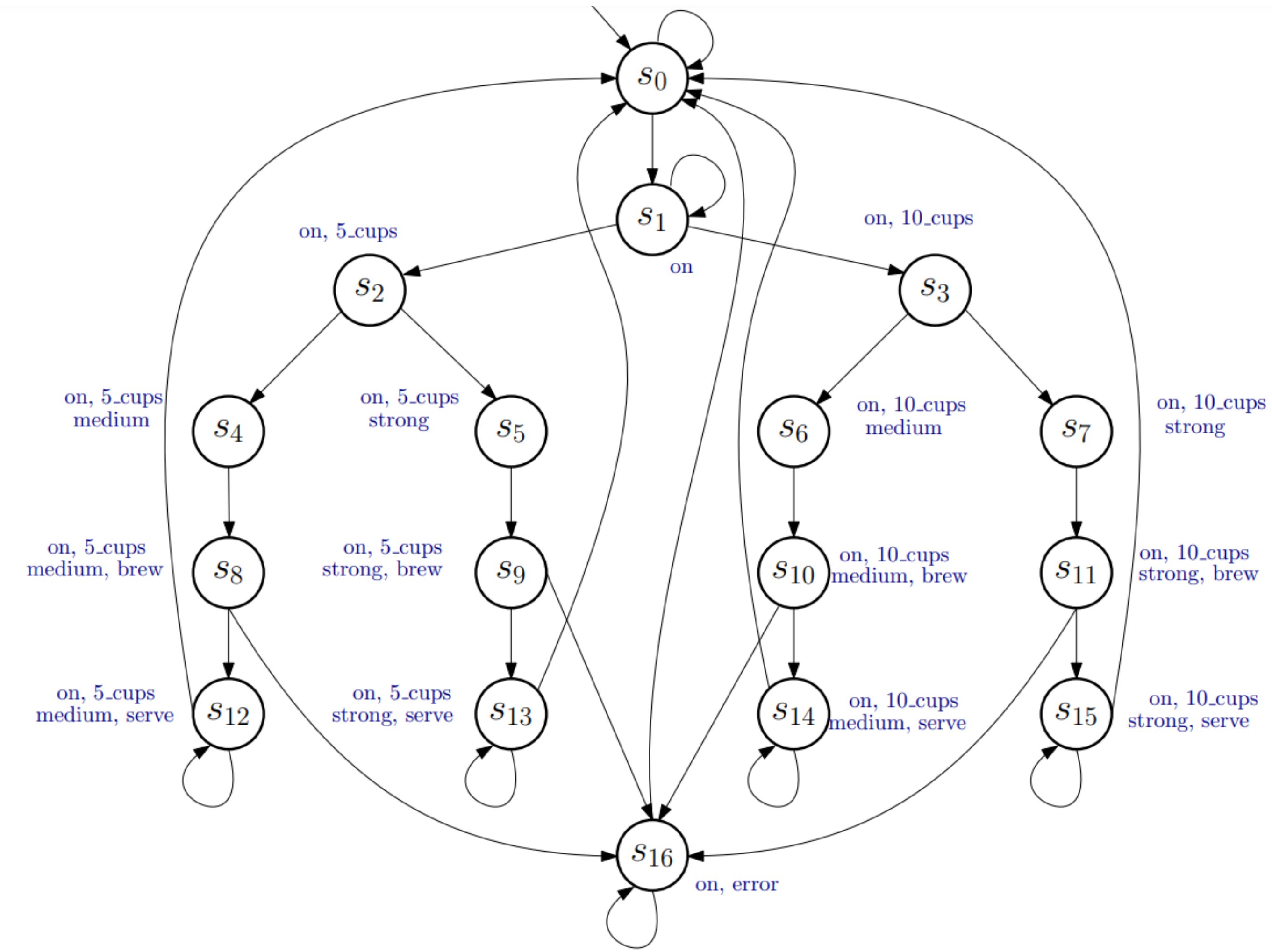
1. $\varphi := A\left((F\ serve)U\ (G\ \neg on)\right)$

NO



Homework

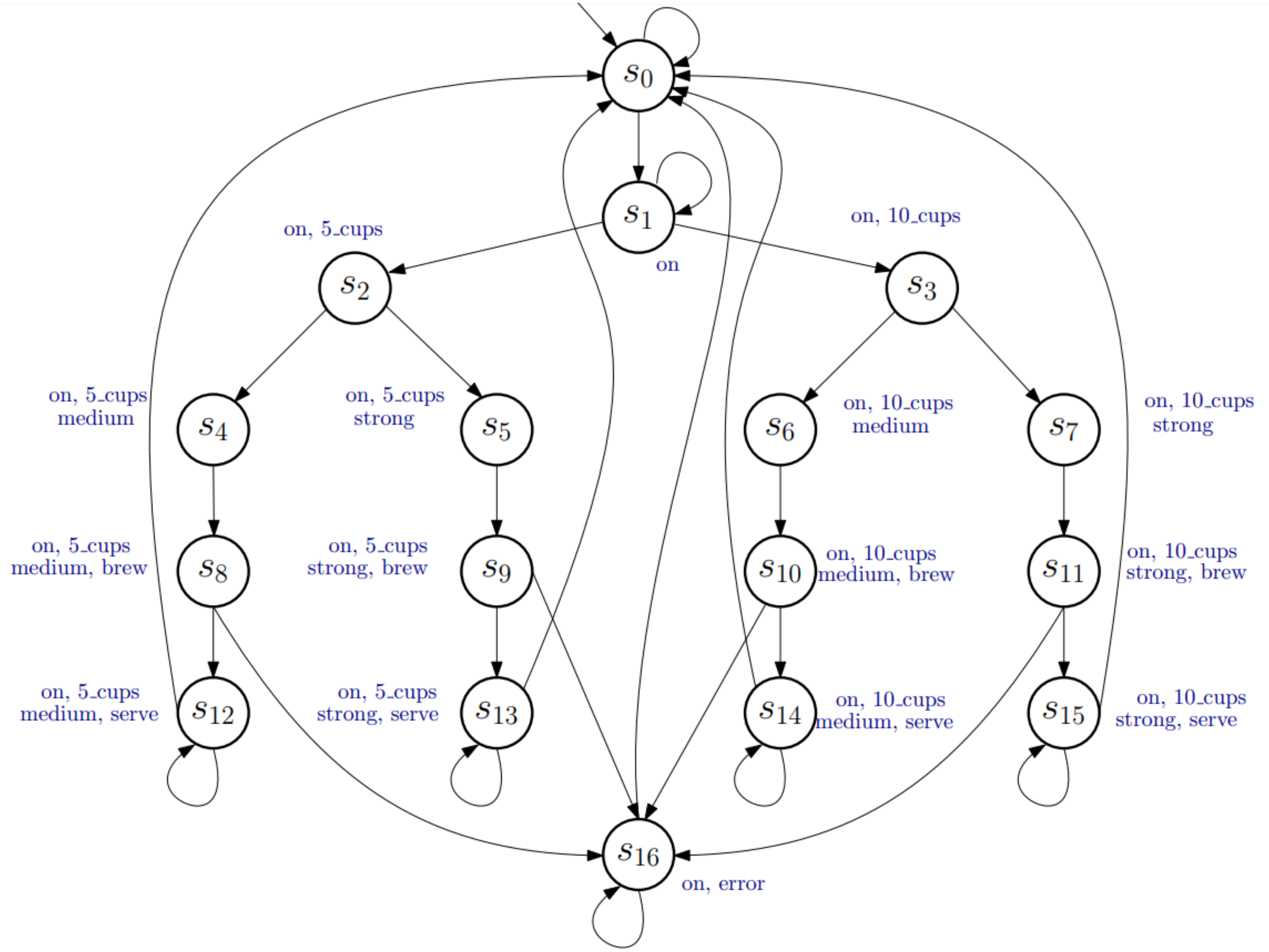
2. φ := AG(serve → (X¬on) ∨ (XX¬on) ∨ (XXX¬on))



Homework

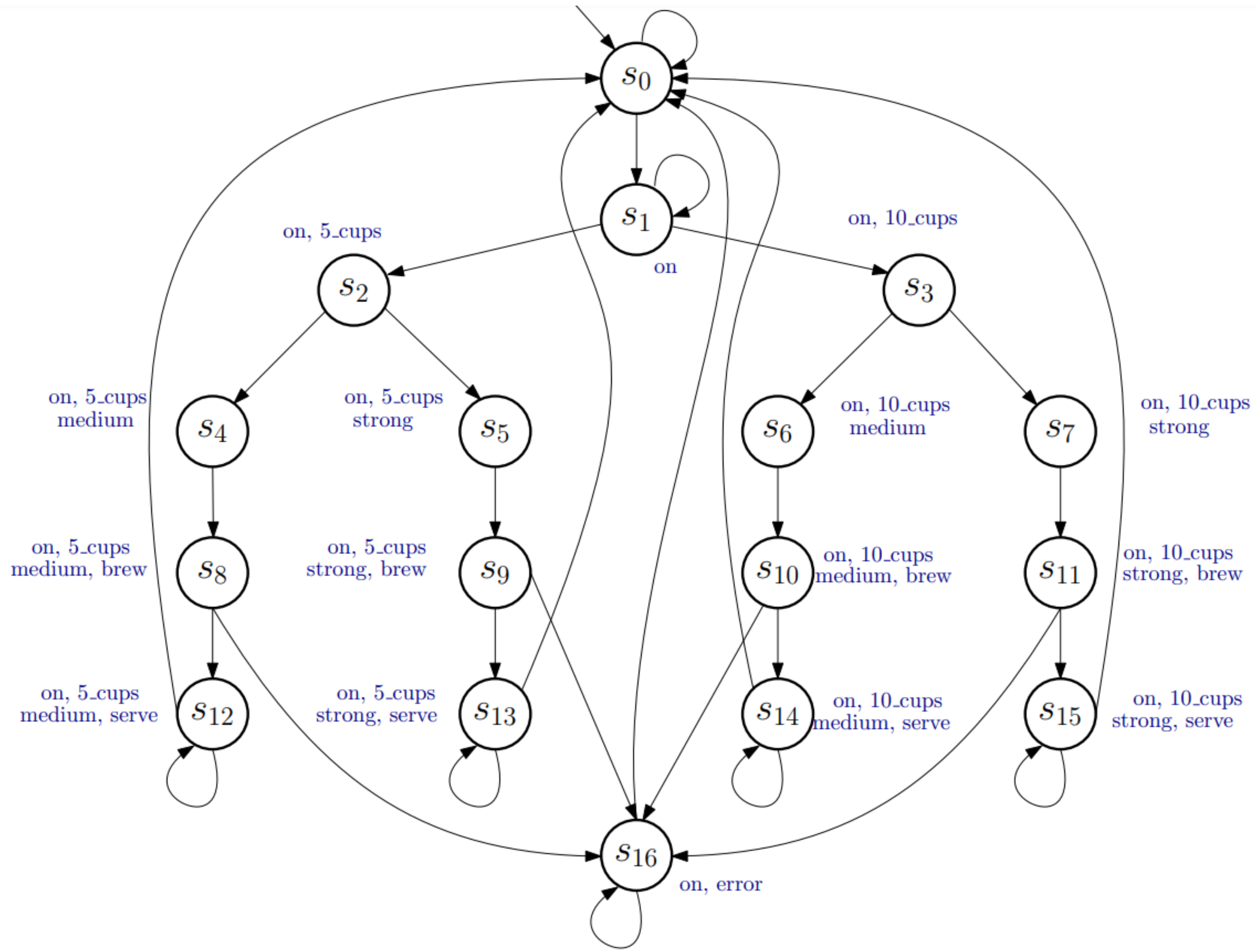
2. $\varphi := AG(\text{ serve} \rightarrow (X\neg\text{on}) \vee (XX\neg\text{on}) \vee (XXX\neg\text{on}))$

NO



Homework

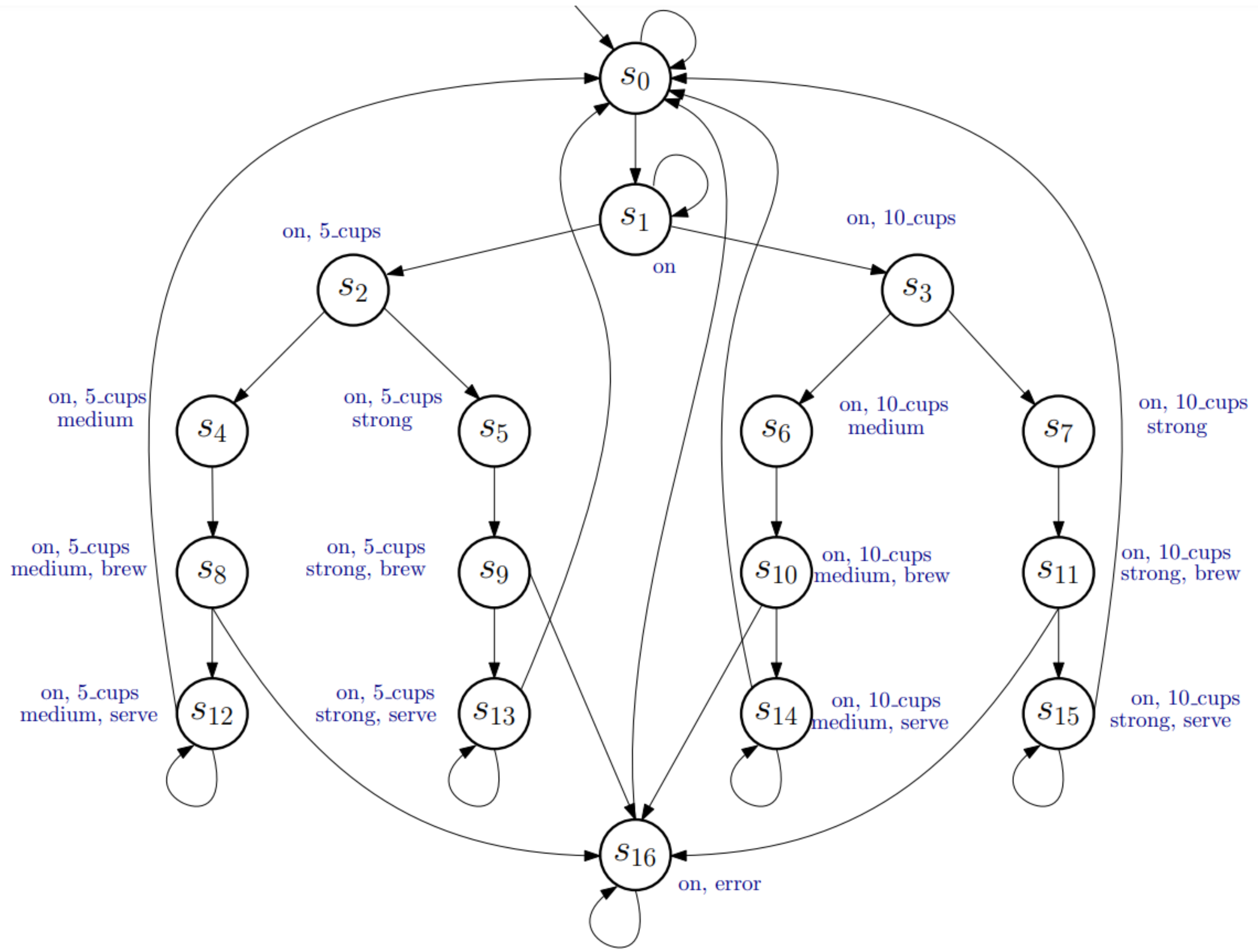
3. $\varphi := EF\big(F\ error\ \rightarrow EF\ (10cups\ \wedge\ serve)\big)$



Homework

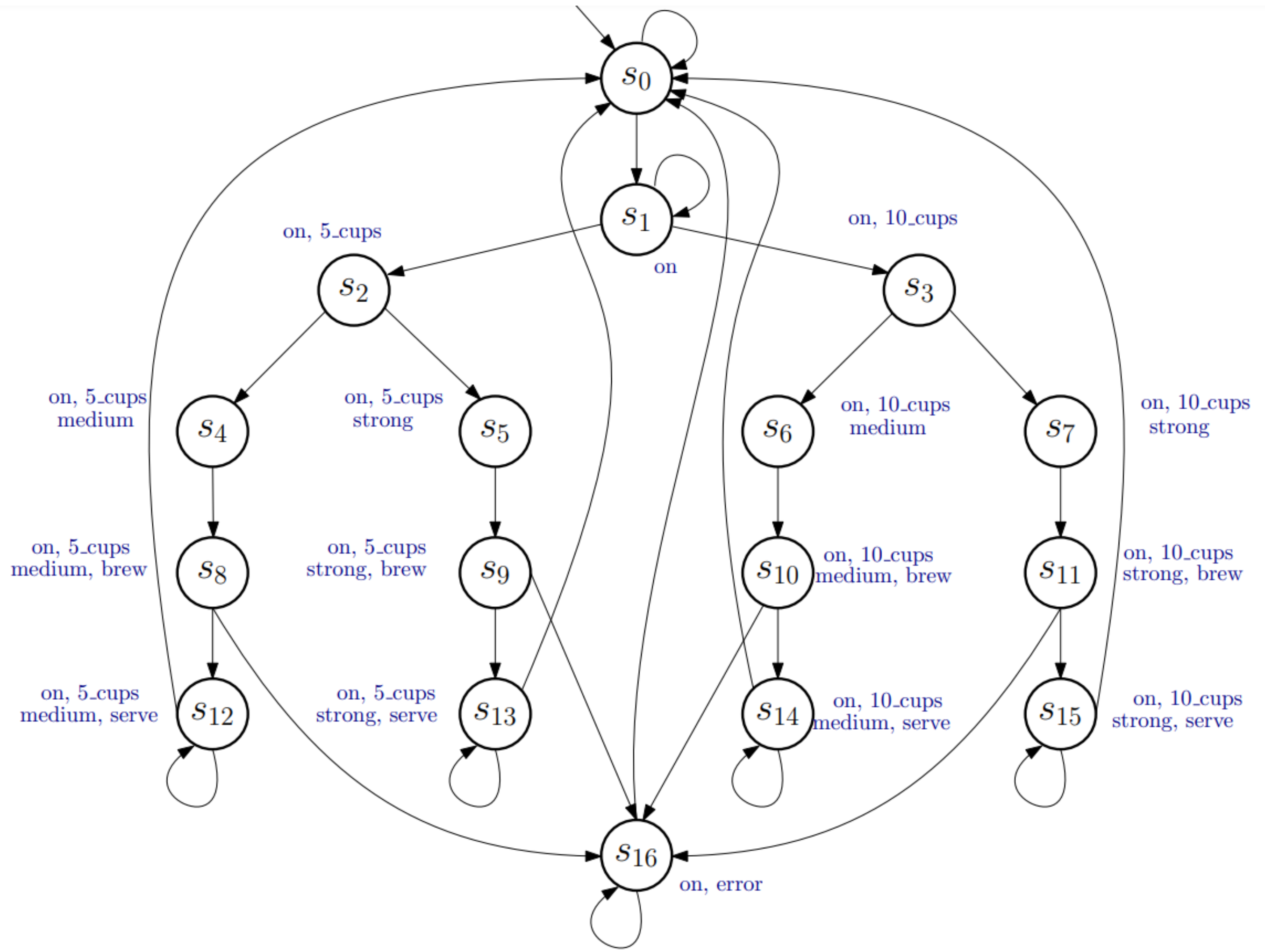
3. $\varphi := EF\big(F\ error\ \rightarrow EF\ (10cups\ \wedge\ serve)\big)$

YES



Homework

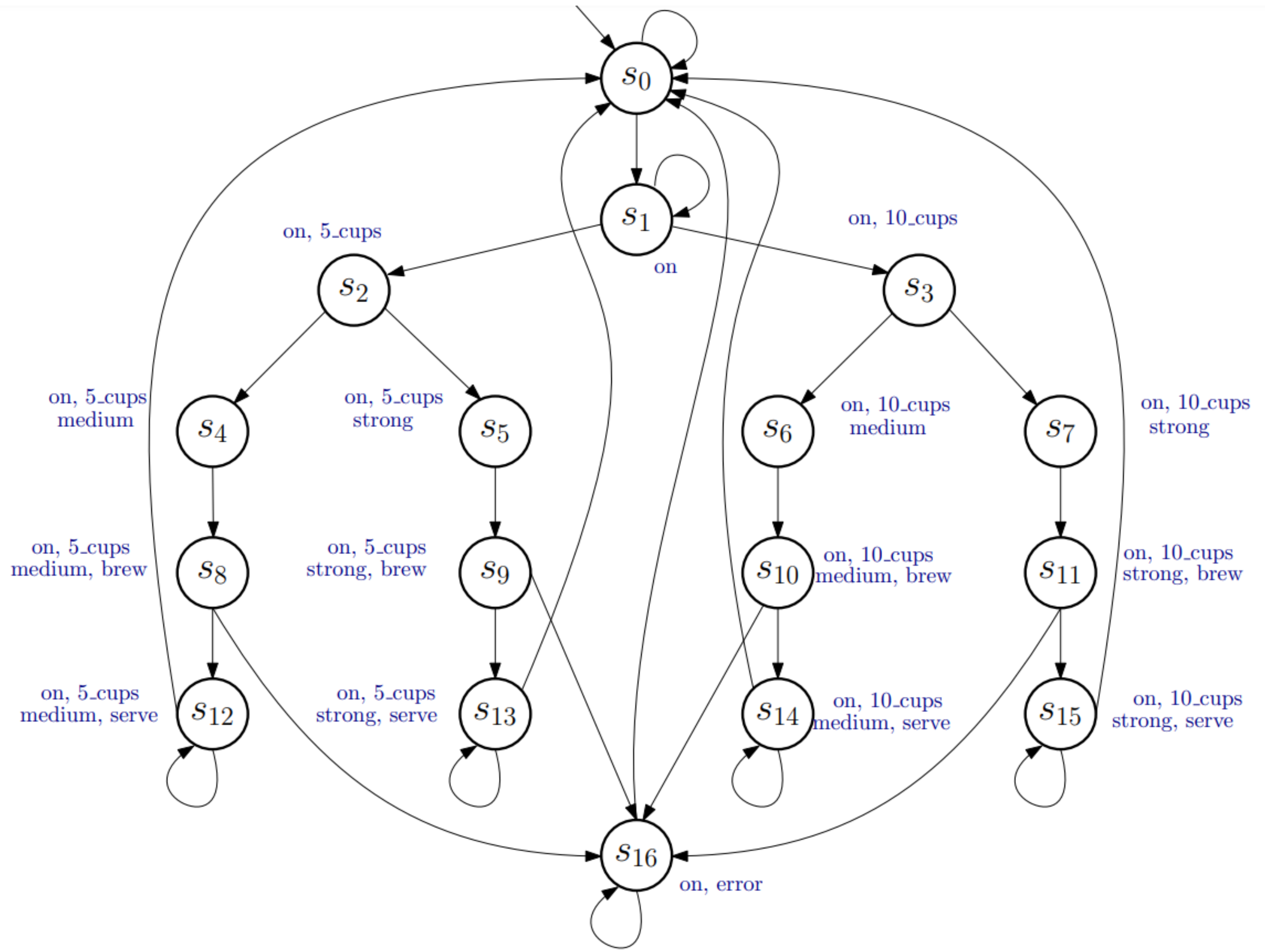
4. $\varphi := AF(serve) \rightarrow (EF\,GF(\neg on))$



Homework

4. $\varphi := AF(serve) \rightarrow (EF\ GF(\neg on))$

YES

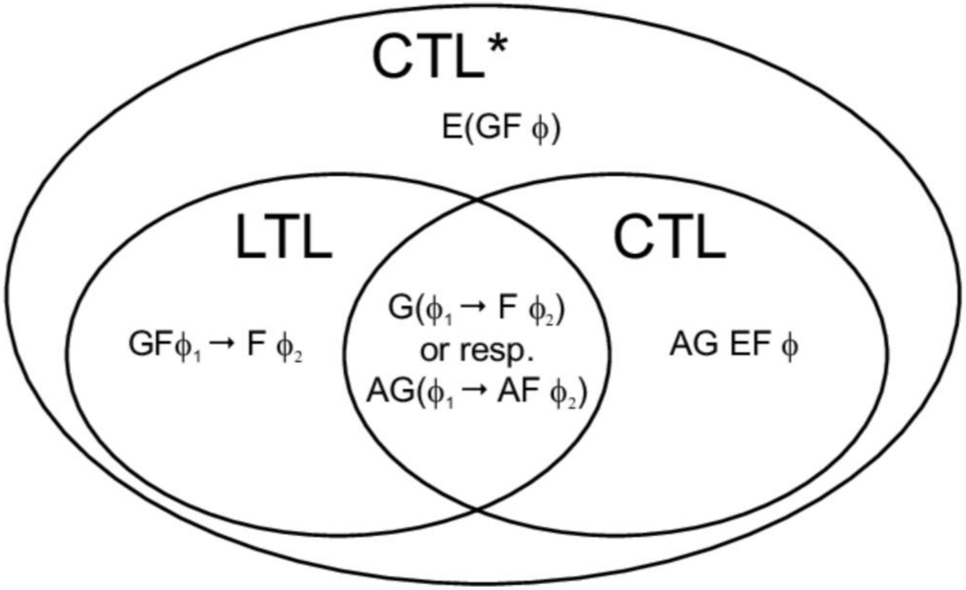


Plan for Today

- Presentation of Homework and Recap of Temporal Logic
- Properties of CTL and LTL
  - LTL vs CTL
  - Counterexamples
  - Safety and Liveness Properties
- CTL Model Checking

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**



LTL vs CTL

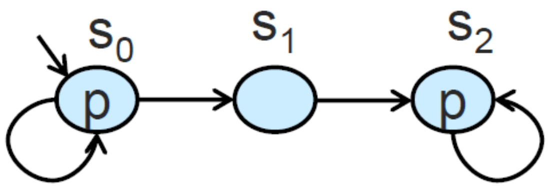
- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
  - $\boldsymbol{AFG\ p}$ = "*for all paths, eventually p always holds*"

LTL vs CTL

- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
  - $\boldsymbol{AFG\ p}$ = "*for all paths, eventually p always holds*"
- Solution: No
  - But what about: **AFAGp?**
  - **AFAGp** = "*for all paths, there is a point from which all reachable states satisfy p*"

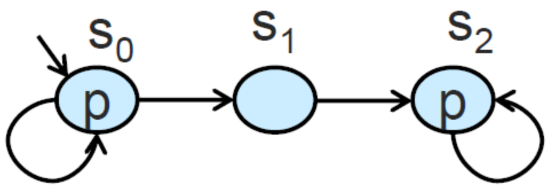
LTL vs CTL

- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\,p}$ has an equivalent in CTL?
  - $\boldsymbol{AFG\,p}$ = "*for all paths, eventually p always holds*"
- Solution: No
  - But what about: **AFAGp?**
  - **AFAGp** = "*for all paths, there is a point from which all reachable states satisfy p*"
    - Consider the given model:
    - Does **AFGp** hold?
    - Does **AFAGp** hold?



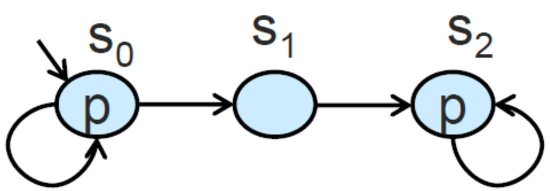
LTL vs CTL

- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
  - $\boldsymbol{AFG\ p}$ = "*for all paths, eventually p always holds*"
- Solution: No
  - But what about: **AFAGp?**
  - **AFAGp** = "*for all paths, there is a point from which all reachable states satisfy p*"
    - Consider the given model:
    - **AFGp** holds
      - All paths satisfy FGp
      - $s_0, s_0, s_0, \ldots$
      - $s_0, s_0, \ldots s_0, s_1, s_2, s_2, s_2, \ldots.$



LTL vs CTL

- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
  - $\boldsymbol{AFG\ p}$ = "*for all paths, eventually p always holds*"
- Solution: No
  - But what about: **AFAGp?**
  - **AFAGp** = "*for all paths, there is a point from which all reachable states satisfy p*"
    - Consider the given model:
    - **AFGp** holds
    - AFAG**p** does not hold
      - $s_0, s_0, s_0, \ldots$ does not satisfy FAGp

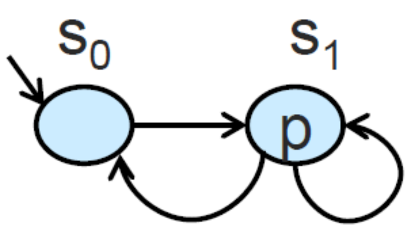


LTL vs CTL

- Exercise:
  - Does the LTL formula $AFG\ p$ has an equivalent in CTL?
  - $AFG\ p$ = "*for all paths, eventually p always holds*"

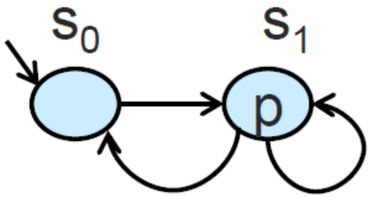
ToDo

- Solution: No
  - What about **AFEG p?**

Hint: 

LTL vs CTL

- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
- Solution: No
  - “in every path there is a point from which there is a path where p globally holds”

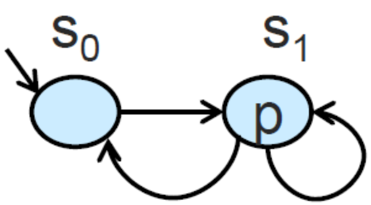


All paths satisfy **FEG**p

- since s_1 sat **EG**p

LTL vs CTL

- Exercise:
  - Does the LTL formula $AFG\ p$ has an equivalent in CTL?
- Solution: No
  - “in every path there is a point from which there is a path where p globally holds”



All paths satisfy **FEG**p

- since s_1 sat **EG**p

But $s_0,s_1,s_0,s_1,s_0,s_1\ldots$ does not satisfy **FG**p

LTL vs CTL

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?

LTL vs CTL

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?
  - **AG(EF p) = "***From* **… "**

LTL vs CTL

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?
  - **AG(EF p) = "***From all reachable states, it is possible to reach a state that satisfies p"*

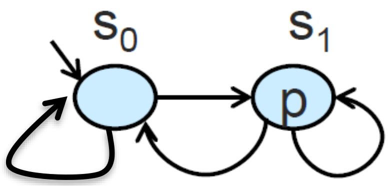
LTL vs CTL

ToDo

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?
  - **AG(EF p) = "***From all reachable states, it is possible to reach a state that satisfies p"*

- What about **AGF p = "***In all paths, p holds infinitely often"?*
  - Does AG(EFp) hold?
  - Does AGFp hold?

Hint:

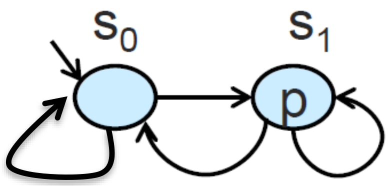


LTL vs CTL

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?
  - **AG(EF p) = "***From all reachable states, it is possible to reach a state that satisfies p"*

- What about **AGF p = "***In all paths, p holds infinitely often"*
  - AG(EFp) holds
    - All reachable states (s_0, s_1) satisfy EFp

Hint:

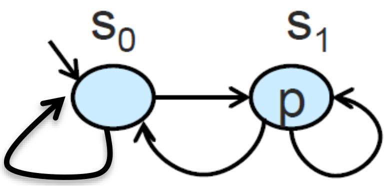


LTL vs CTL

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?
  - **AG(EF p) = "***From all reachable states, it is possible to reach a state that satisfies p"*

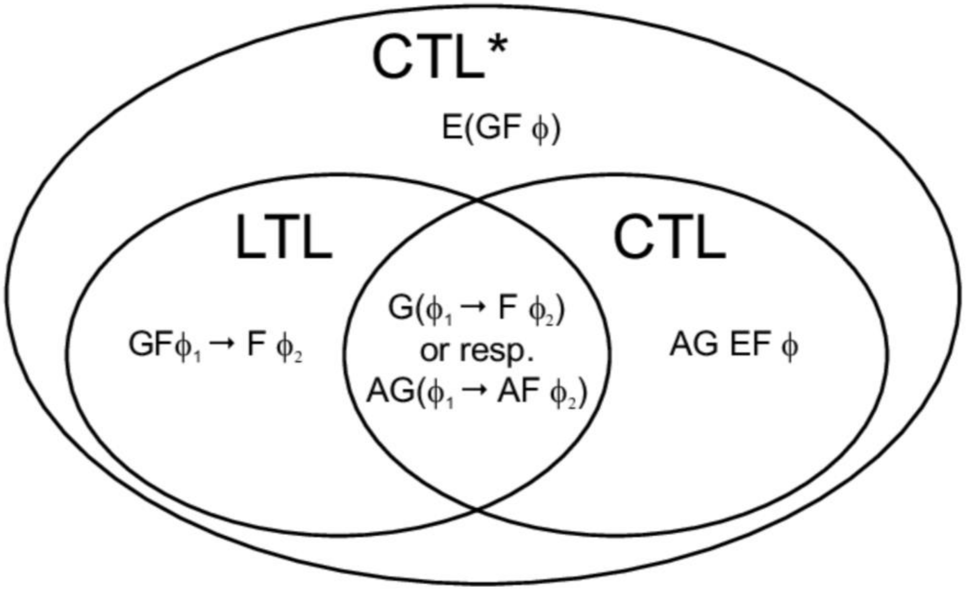
- What about **AGF p = "***In all paths, p holds infinitely often"*
  - AG(EFp) holds
    - All reachable states (s_0, s_1) satisfy EFp
  - AGFp does not hold
    - s_0, s_0, s_0 ... does not satisfy GFp

Hint:



LTL vs CTL

- The expressive powers of LTL and CTL are incomparable. That is,
  - There is an LTL formula that has no equivalent CTL formula
  - There is a CTL formula that has no equivalent LTL formula



Plan for Today

- Presentation of Homework and Recap of Temporal Logic
- Properties of CTL and LTL
  - LTL vs CTL
  - Counterexamples
  - Safety and Liveness Properties
- CTL Model Checking

Counterexamples

- Given M and φ s.t. $M \not\models \varphi$.
  A counterexample is trace π of M violating φ.

- Counterexample generation is a central feature of MC
- Used for debugging
  - Should have finite representation
  - Easy-to-understand by human

Examples of Counterexamples

- For **AX**p:
  A transition from an initial state to a state violating p

Examples of Counterexamples

- For **AX**p:
  A transition from an initial state to a state violating p
  - Counterexample for **AX**p is a witness for **EX**¬ p

Examples of Counterexamples

- For **AX**p:
  A transition from an initial state to a state violating p
  - Counterexample for **AX**p is a witness for **EX**¬ p

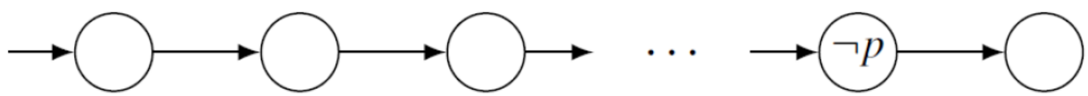
- For **AG**p:
  A finite path from an initial state to a state violating p

Examples of Counterexamples

- For **AX**p:
  A transition from an initial state to a state violating p
  - Counterexample for **AX**p is a witness for **EX**¬ p

- For **AG**p:
  A finite path from an initial state to a state violating p
  - Counterexample for **AG**p is a witness for **EF**¬ p

$\mathbf{F}\,\neg p$



Examples of Counterexamples

- For **AF**p:

How does a counterexample for AFp look like?

Examples of Counterexamples

- For **AF**p:
  An infinite path, all of its states violating p (satisfying ¬p)
  - Counterexample for **AF**p is a witness for **EG**¬ p

Examples of Counterexamples

- For **AF**p:
  An infinite path, all of its states violating p (satisfying ¬p)
  - Counterexample for **AF**p is a witness for **EG**¬ p

ToDo Exercise:

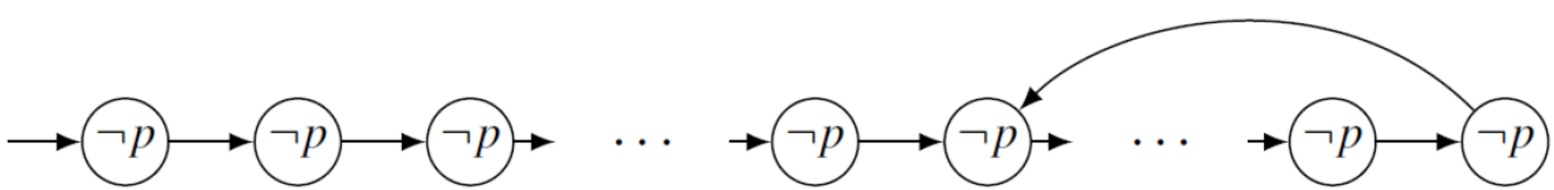
- How do we get a finite representation for the CE?

Examples of Counterexamples

- For **AF**p:
  An infinite path, all of its states violating p (satisfying ¬p)
  - Counterexample for **AF**p is a witness for **EG**¬ p

- A finite representation for violation of AFp:
  - A lasso, which is a path of the form $\pi = \pi_0 (\pi_1)^\omega$
  - π_0 and π_1 are finite paths
  - ω indicates infinitely many repetitions of π_1

$\mathbf{G}\neg p$



Plan for Today

- Presentation of Homework and Recap of Temporal Logic
- Properties of CTL and LTL
  - LTL vs CTL
  - Counterexamples
  - Safety and Liveness Properties
- CTL Model Checking

Safety and Liveness Properties

Informally,

- Safety properties guarantee that
  "something bad will never happen"
  - Typical example: $AG\neg p$

Safety and Liveness Properties

Informally,

- Safety properties guarantee that
  "something bad will never happen"
  - Typical example: $AG \neg p$

- Liveness properties guarantee that
  "something good will happen eventually"
  - Typical examples: $AF\, p$, $A(pUq)$

Safety and Liveness Properties

Informally,

- Safety properties guarantee that
  "something bad will never happen"
  - Typical example: $AG\neg p$

Exercise:

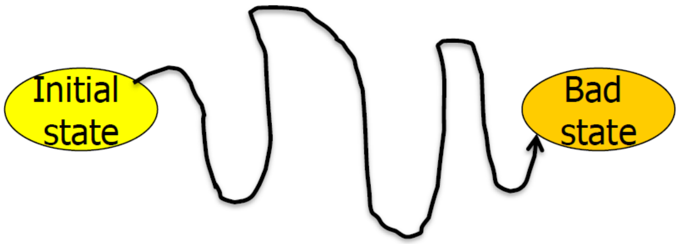
- How does a **counterexample** for a **safety** property look like?

Safety and Liveness Properties

Informally,

- Safety properties guarantee that
  "something bad will never happen"
  - Typical example: $AG\neg p$

- Exercise:
  - How does a **counterexample** for a **safety** property look like?
  - A counterexample for a safety property is a **finite (loop-free) path**



Safety and Liveness Properties

Informally,

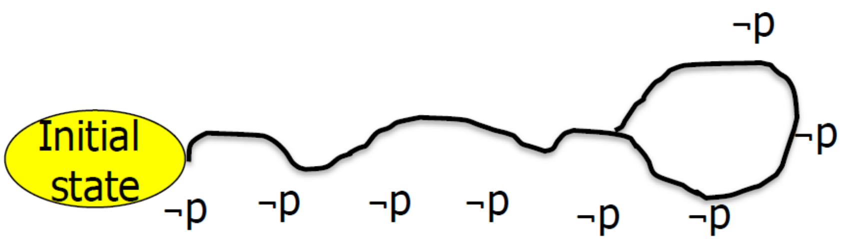
- Liveness properties guarantee that
  "something good will happen eventually"
  - Typical examples: $AF\ p$
- Exercise:

  - How does a **counterexample** for a **liveness** property look like?

Safety and Liveness Properties

Informally,

- Liveness properties guarantee that
  "something good will happen eventually"
  - Typical examples: $AF\ p$
- Exercise:
  - How does a **counterexample** for a **liveness** property look like?
  - A counterexample is **an infinite** trace showing that this good thing NEVER happened



Plan for Today

- Presentation of Homework and Recap of Temporal Logic
- Properties of CTL and LTL
  - LTL vs CTL
  - Counterexamples
  - Safety and Liveness Properties
- CTL Model Checking
  - MC Problem Definition
  - Illustrative Example for CTL Model Checking
  - Algorithm for CTL MC

The Model Checking Problem

- Given a Kripke structure M and a CTL formula f
- Model Checking Problem:
  - $M \models f$, i.e., M is a model for f

The Model Checking Problem

- Given a Kripke structure M and a CTL formula f
- Model Checking Problem:
  - $M \vDash f$, i.e., M is a model for f
- Alternative Definition
  - Compute $[\![\mathrm{f}]\!]_{\mathrm{M}} = \{s \in S \mid M, s \vDash f\}$ i.e., all states satisfying f
  - Check $S_0 \subseteq [\![\mathrm{f}]\!]_{\mathrm{M}}$ to conclude that $M \vDash f$

Illustrative Example: Mutual Exclusion

Illustrative Example: Mutual Exclusion

- Two processes with a joint semaphor 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

Illustrative Example: Mutual Exclusion

- Each process runs the following program:

P_i :: while (true) {

Atomic action

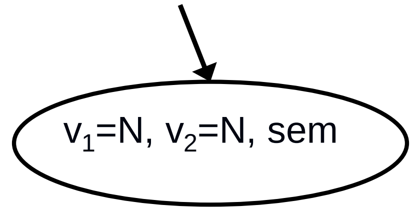
if (v_i == N) v_i = T;

else if (v_i == T && sem) { v_i = C; sem = 0; }

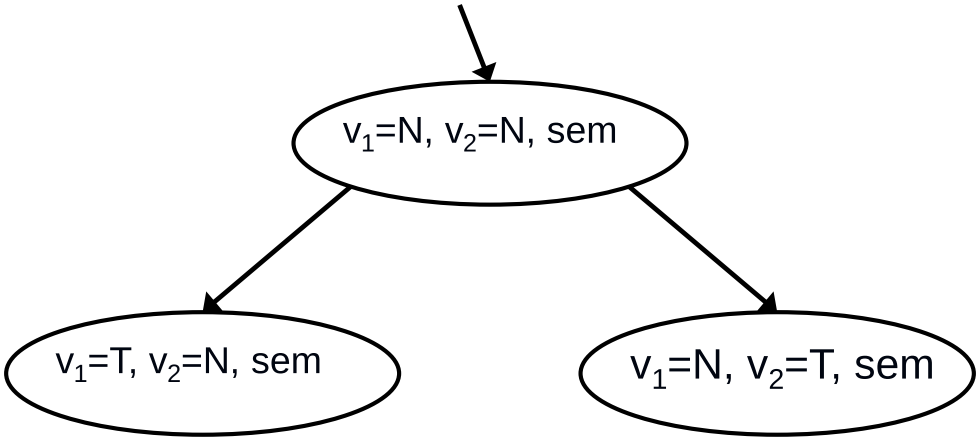
else if (v_i == C) {v_i = N; sem = 1; }

}

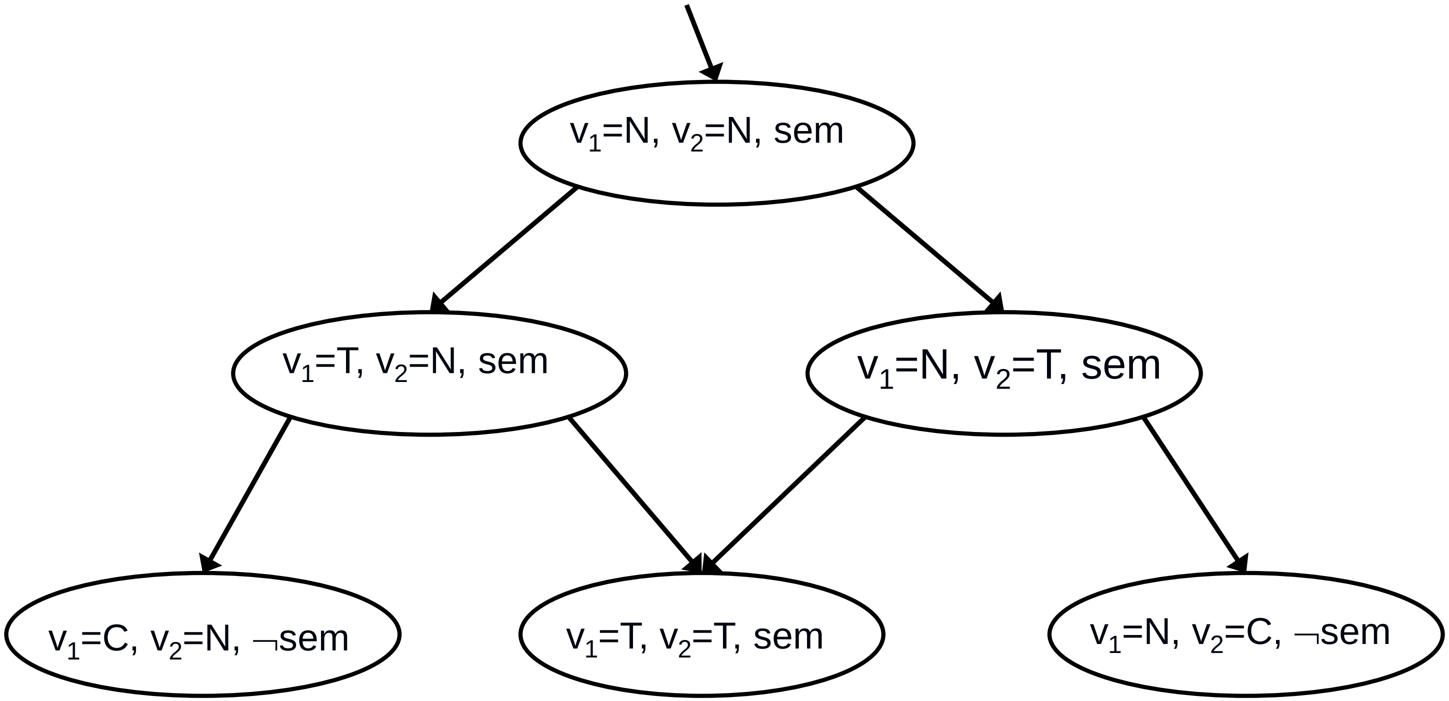
Illustrative Example: Mutual Exclusion



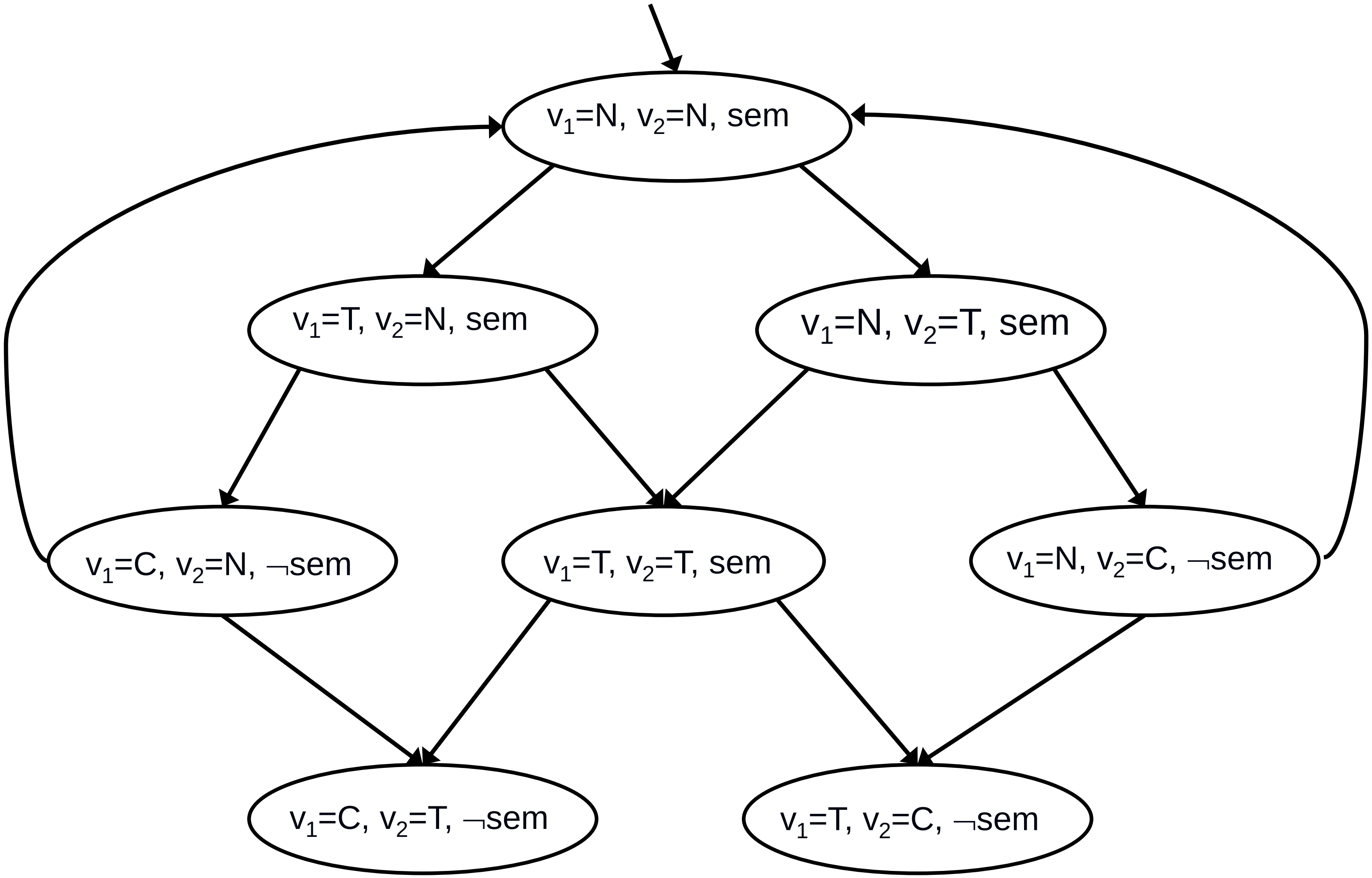
Illustrative Example: Mutual Exclusion



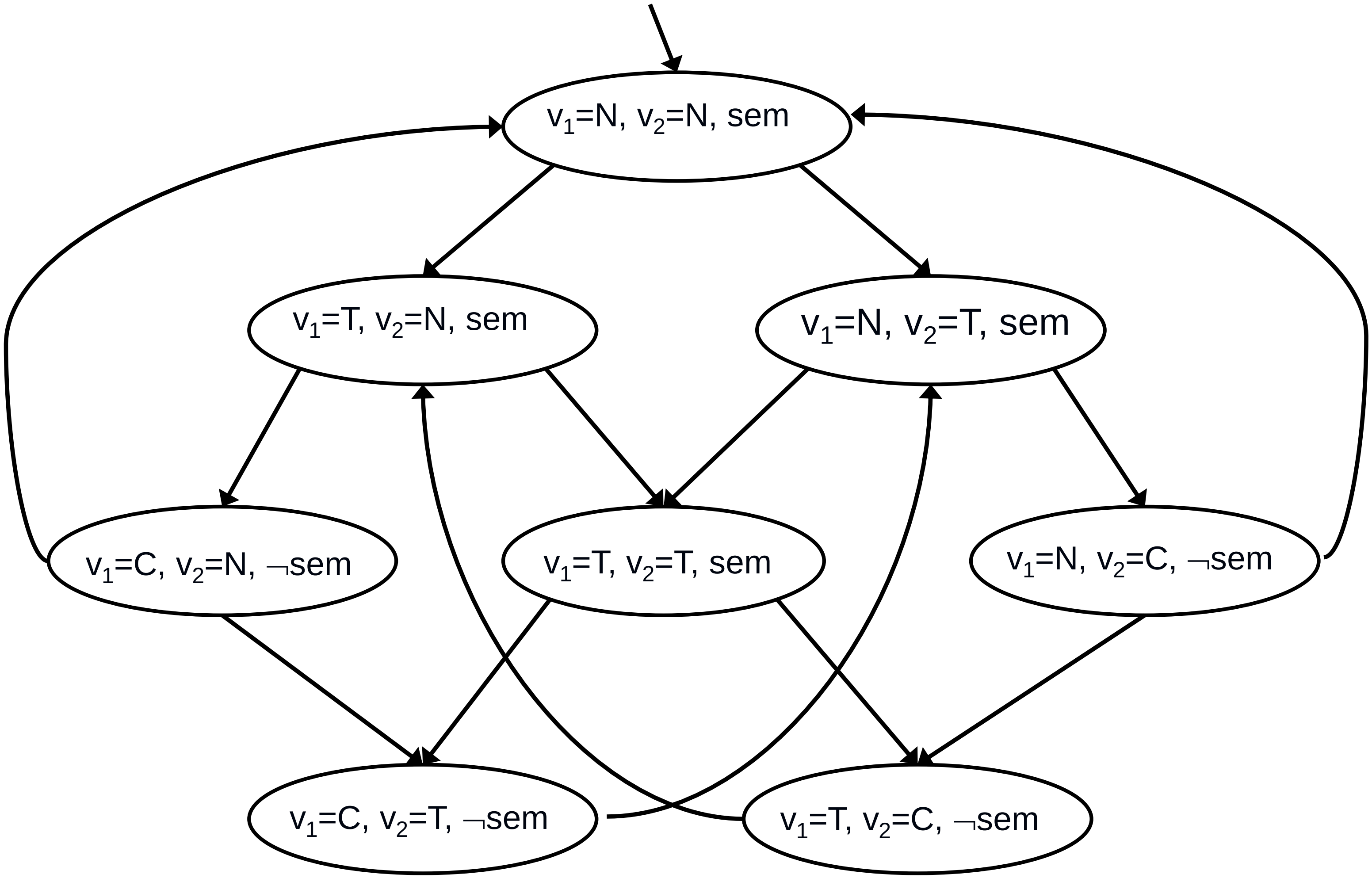
Illustrative Example: Mutual Exclusion



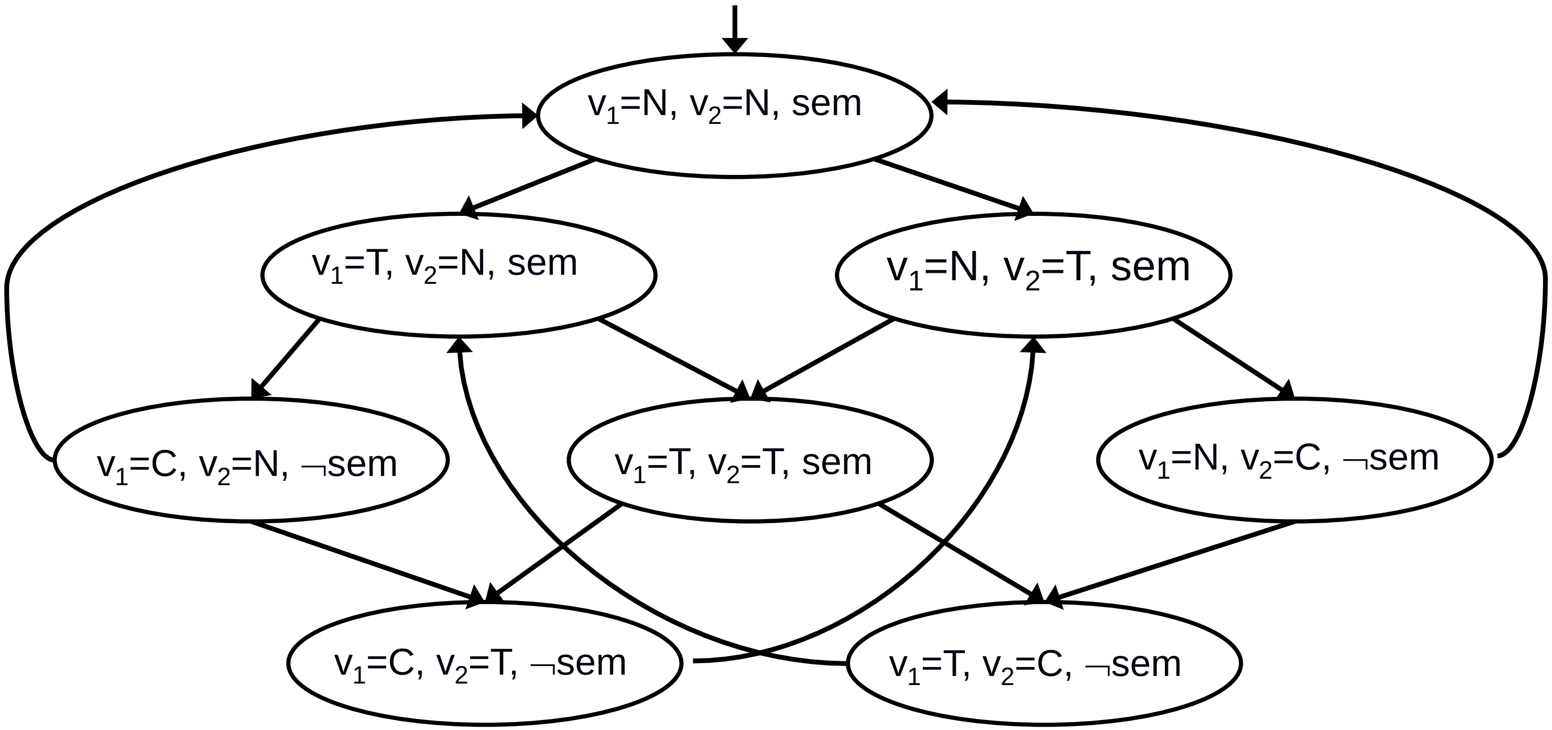
Illustrative Example: Mutual Exclusion



Illustrative Example: Mutual Exclusion

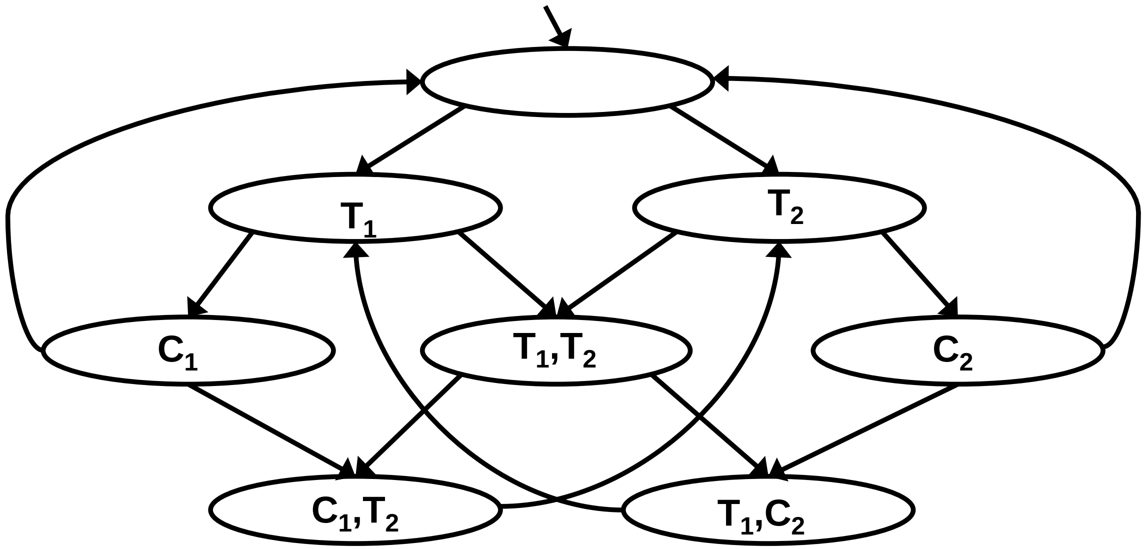


Illustrative Example: Mutual Exclusion



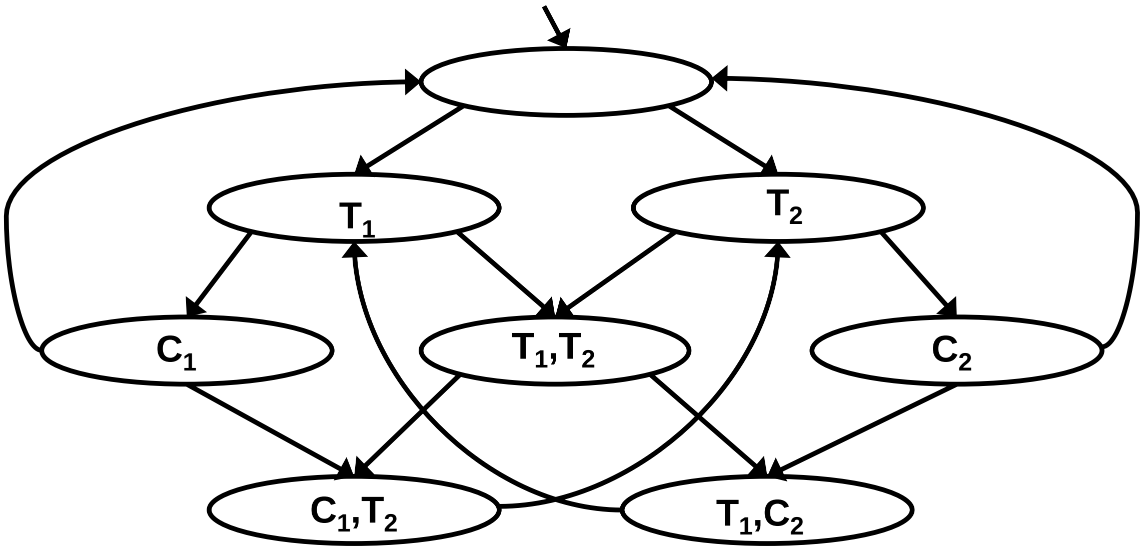
- We define atomic propositions: $AP = \{C_1, C_2, T_1, T_2)$
- A state is labeled with T_i if $v_i = T$
- A state is labeled with C_i if $v_i = C$

Illustrative Example: Mutual Exclusion



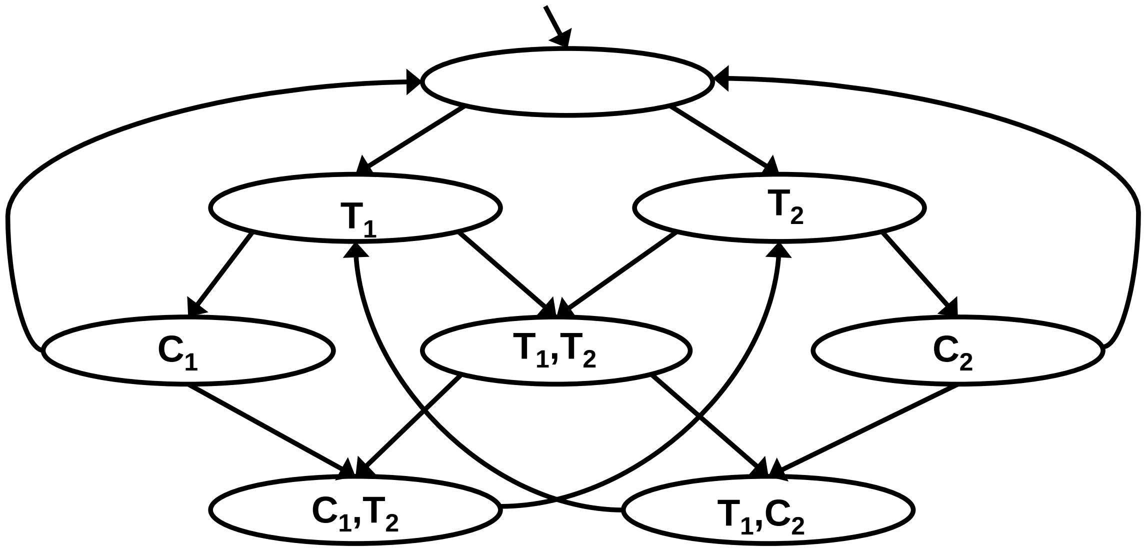
- We define atomic propositions: $AP = \{C_1, C_2, T_1, T_2)$
- A state is labeled with T_i if $v_i = T$
- A state is labeled with C_i if $v_i = C$

Illustrative Example: Mutual Exclusion



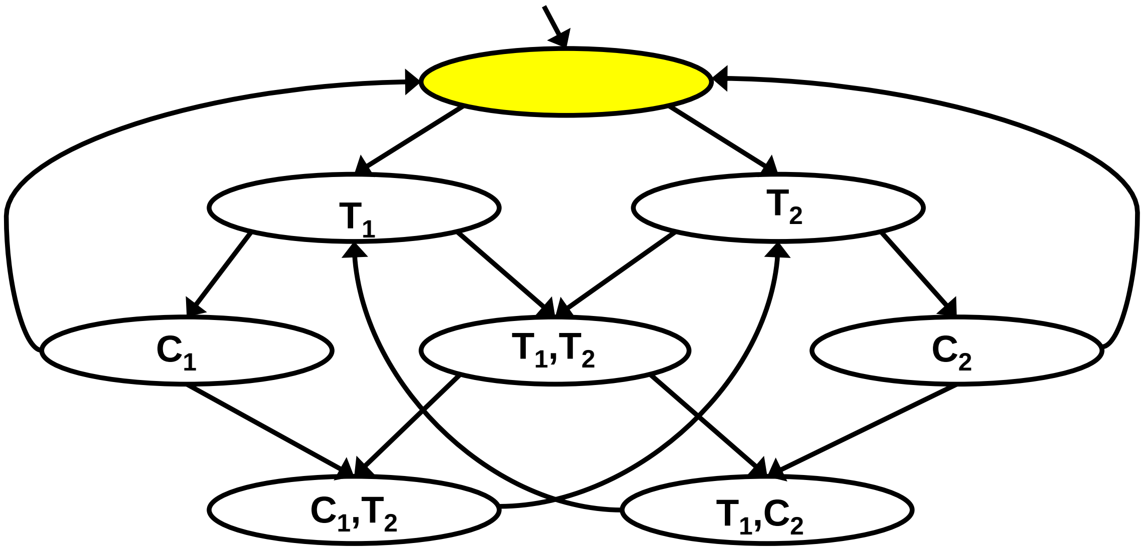
- Does it hold that $M \models f$?
  - Property 1: $f := \boldsymbol{AG}\neg(C_1 \wedge C_2)$
  - Compute $[\![f]\!]_M = \{ s \in S \mid M,s \models f \}$ and check $S_0 \subseteq [\![f]\!]_M$

Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 1: $f := \boldsymbol{AG}\neg(C_1 \wedge C_2)$
- S_i ≡ reachable states from an initial state after i steps

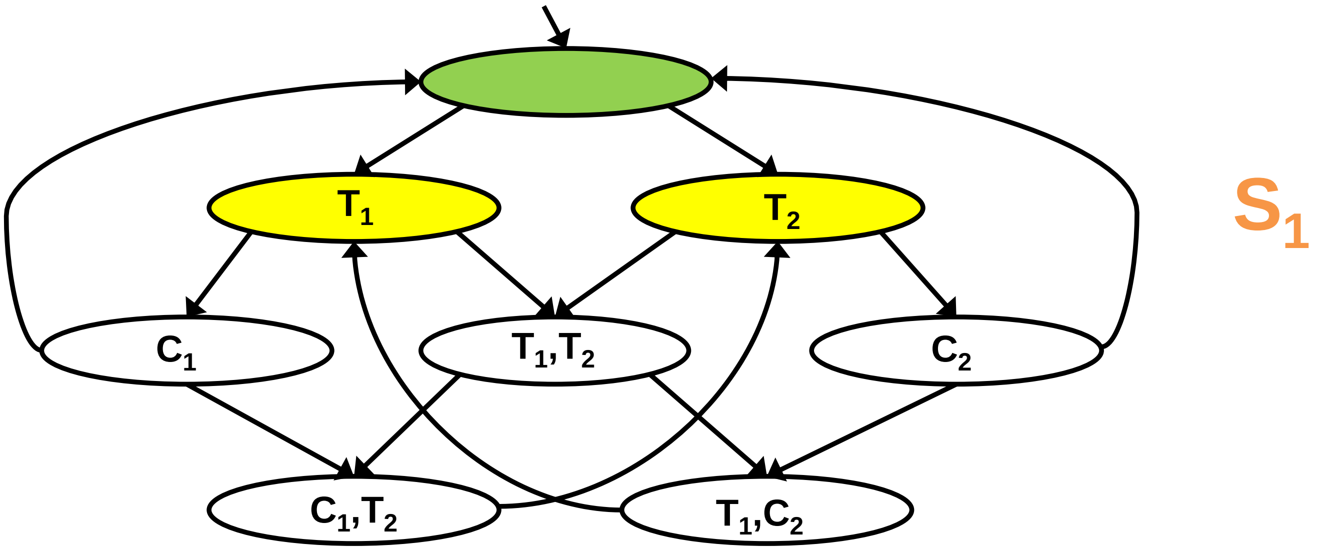
Illustrative Example: Mutual Exclusion



S_0

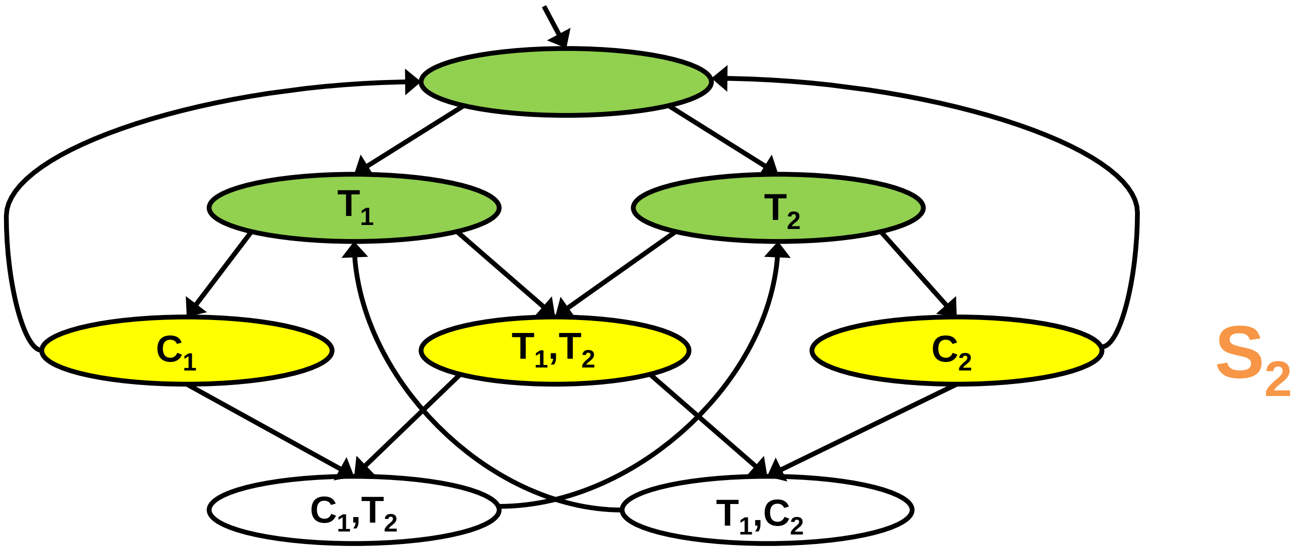
- Does it hold that M ⊨ f?
  - Property 1: $f := \boldsymbol{AG}\neg(C_1 \wedge C_2)$
- $S_i \equiv$ reachable states from an initial state after i steps

Illustrative Example: Mutual Exclusion



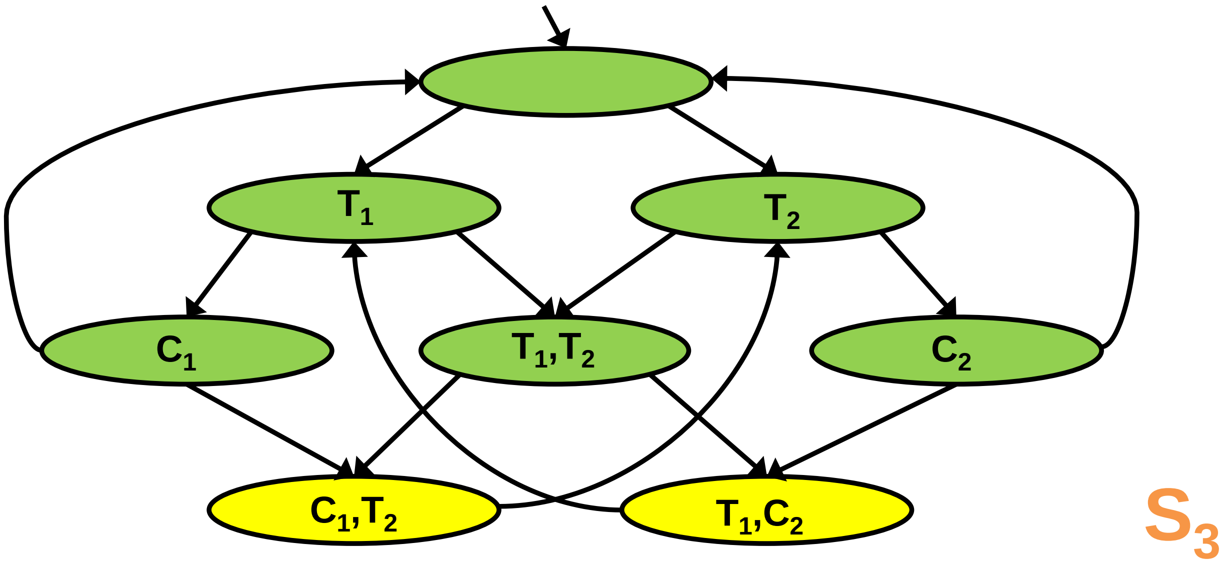
- Does it hold that M ⊨ f?
  - Property 1: $f := \boldsymbol{AG}\neg(C_1 \wedge C_2)$
- S_i ≡ reachable states from an initial state after **i** steps

Illustrative Example: Mutual Exclusion



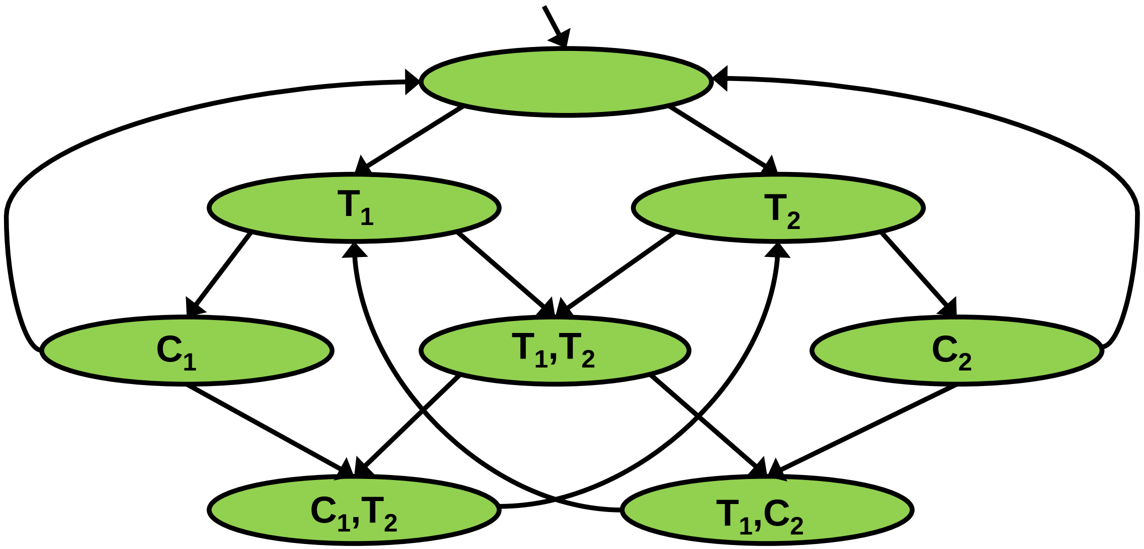
- Does it hold that M ⊨ f?
  - Property 1: $f := \boldsymbol{AG}\neg(C_1 \wedge C_2)$
- S_i ≡ reachable states from an initial state after i steps

Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 1: $f := \boldsymbol{AG}\neg(C_1 \wedge C_2)$
- S_i ≡ reachable states from an initial state after i steps

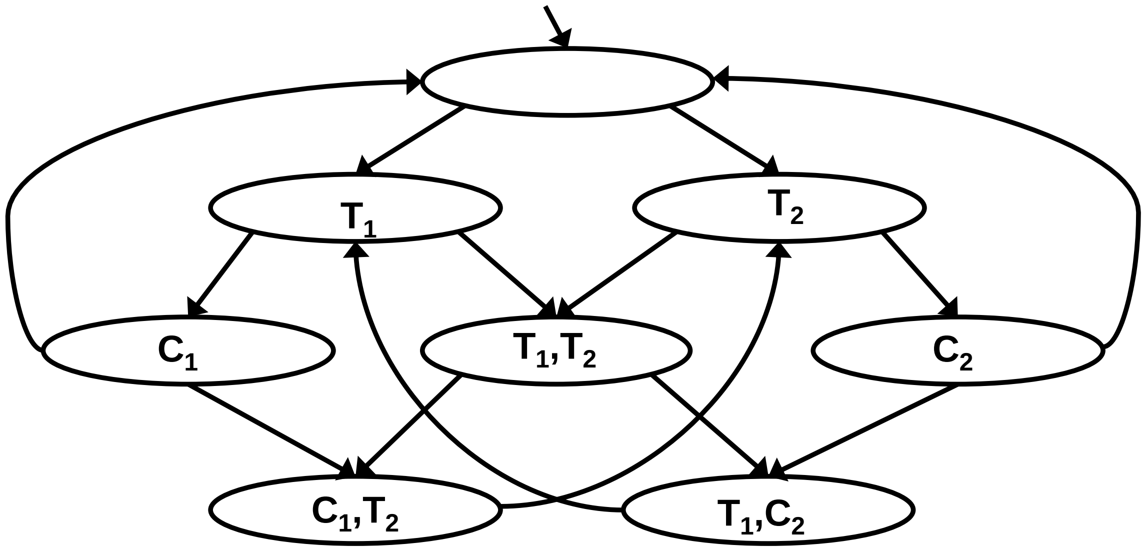
Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 1: $f := \boldsymbol{AG}\neg(C_1 \wedge C_2)$

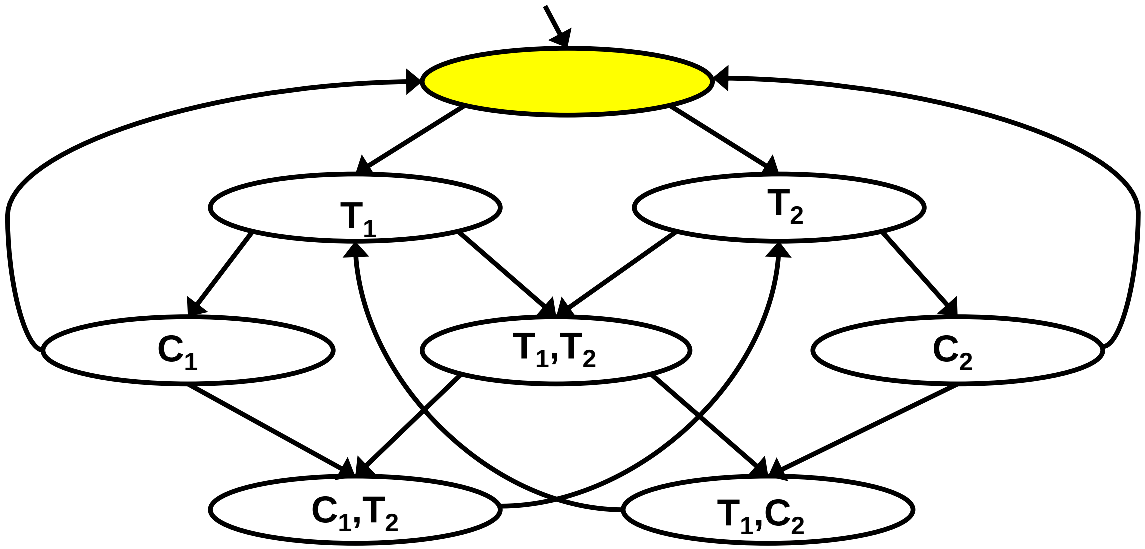
$M \vDash AG \neg (C_1 \wedge C_2)$

Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 2: $f := \boldsymbol{AG}\neg(T_1 \wedge T_2)$

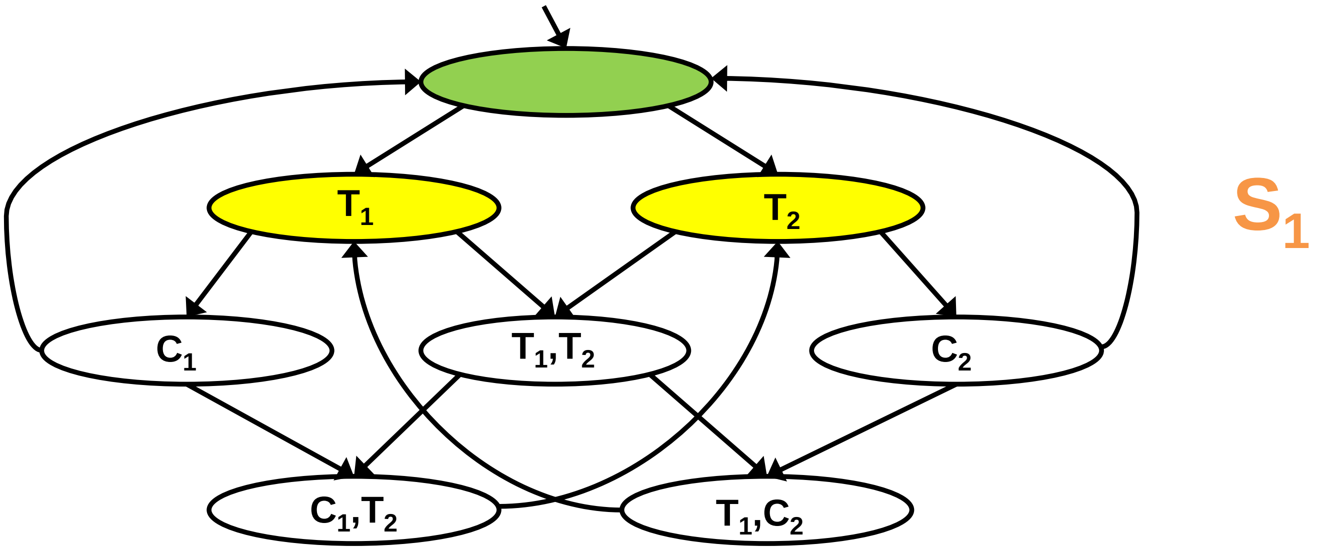
Illustrative Example: Mutual Exclusion



S_0

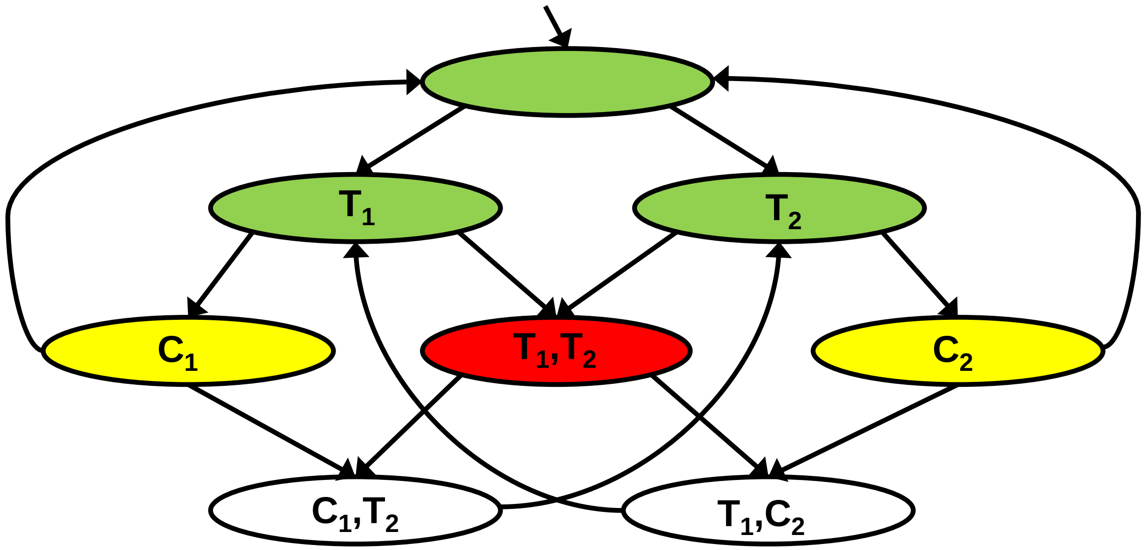
- Does it hold that M ⊨ f?
  - Property 2: $f := \boldsymbol{AG}\neg(T_1 \wedge T_2)$
- S_i ≡ reachable states from an initial state after **i** steps

Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 2: $f := \boldsymbol{AG}\neg(T_1 \wedge T_2)$
- S_i ≡ reachable states from an initial state after i steps

Illustrative Example: Mutual Exclusion

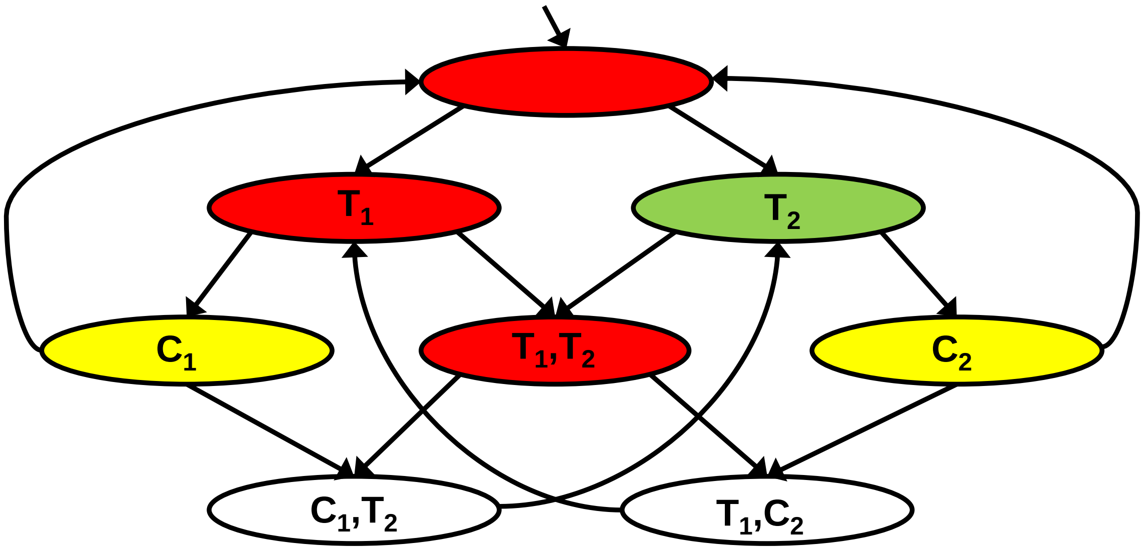


S_3

- Does it hold that $M \models f$?
  - Property 2: $f := \boldsymbol{AG}\neg(T_1 \wedge T_2)$

$M \not\models AG\,\neg(T_1 \wedge T_2)$

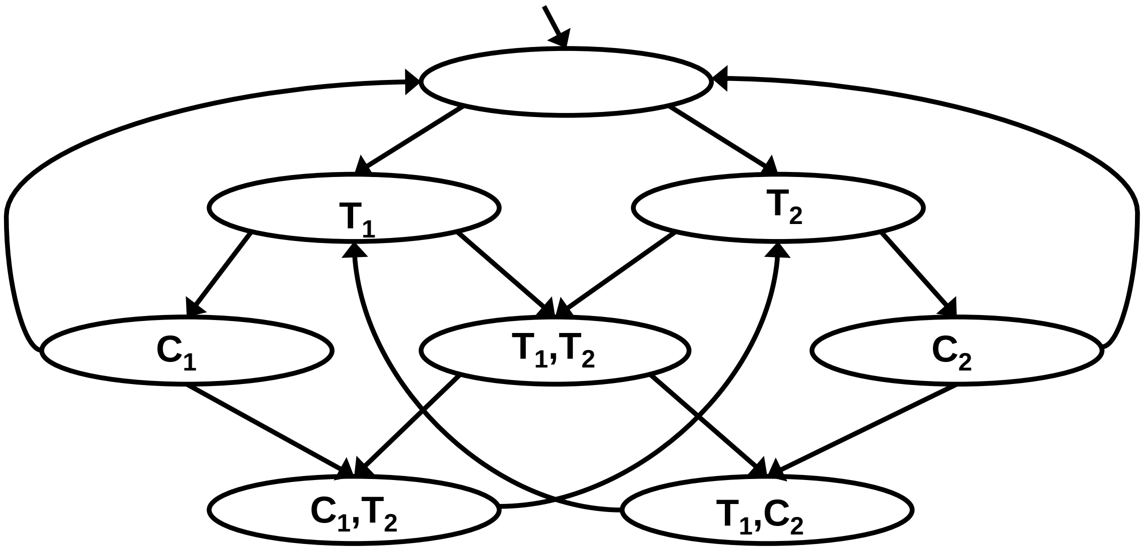
Illustrative Example: Mutual Exclusion



S_3

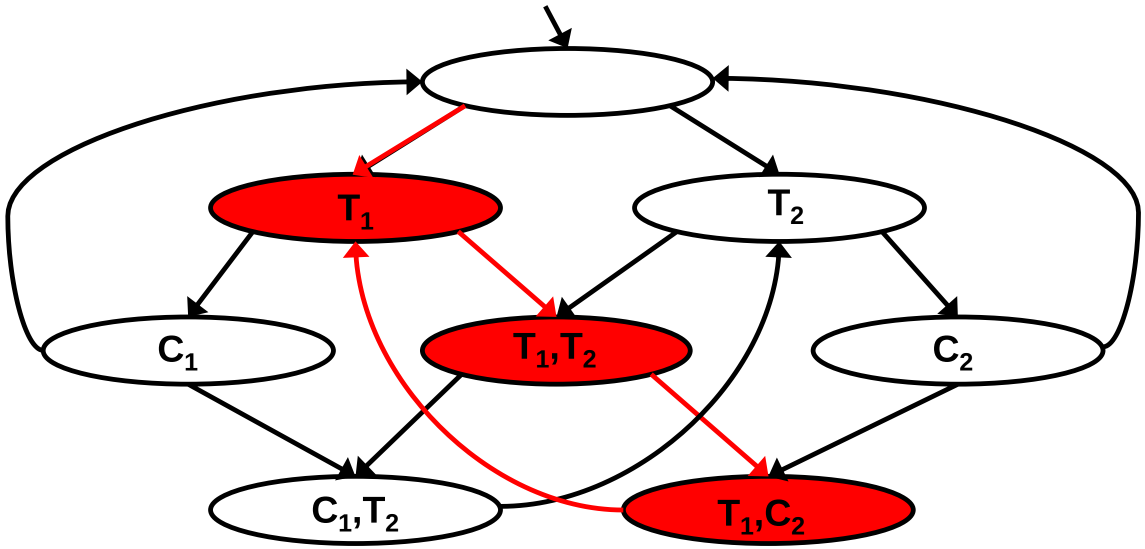
- Does it hold that $M \models f$?
  - Property 2: $f := AG\neg(T_1 \wedge T_2)$ $\qquad M \not\models AG\neg(T_1 \wedge T_2)$
- Model checker returns a **counterexample**

Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 3: $f := \boldsymbol{AG}\,((T_1 \rightarrow \boldsymbol{F}\,C_1) \wedge (T_2 \rightarrow \boldsymbol{F}\,C_2))$
- In case M ⊭ f, compute a counterexample

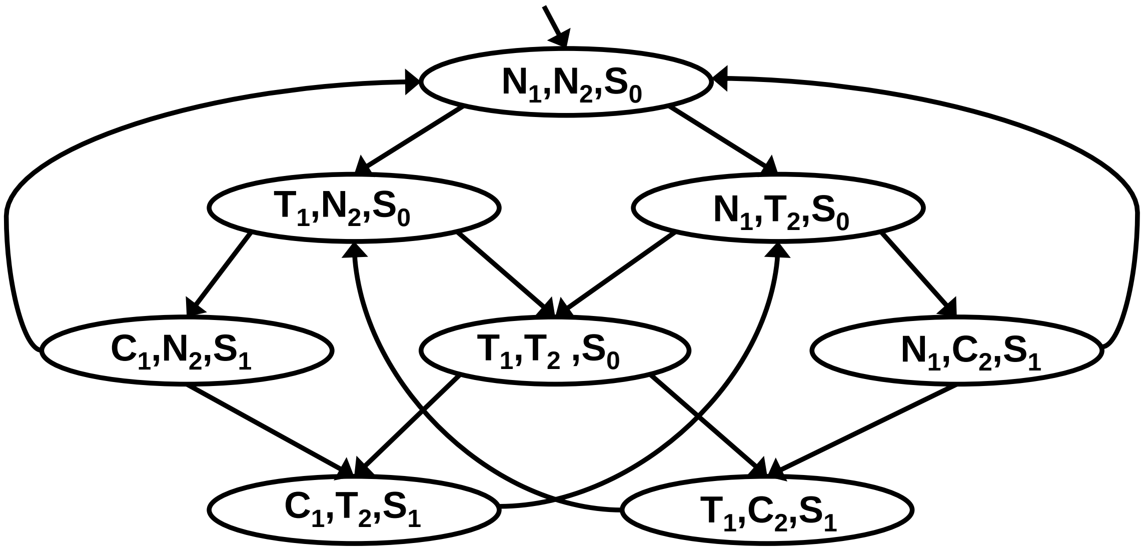
Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 3: $f := \boldsymbol{AG}\,((T_1 \rightarrow \boldsymbol{F}\,C_1) \wedge (T_2 \rightarrow \boldsymbol{F}\,C_2))$
- In case M ⊭ f, compute a counterexample

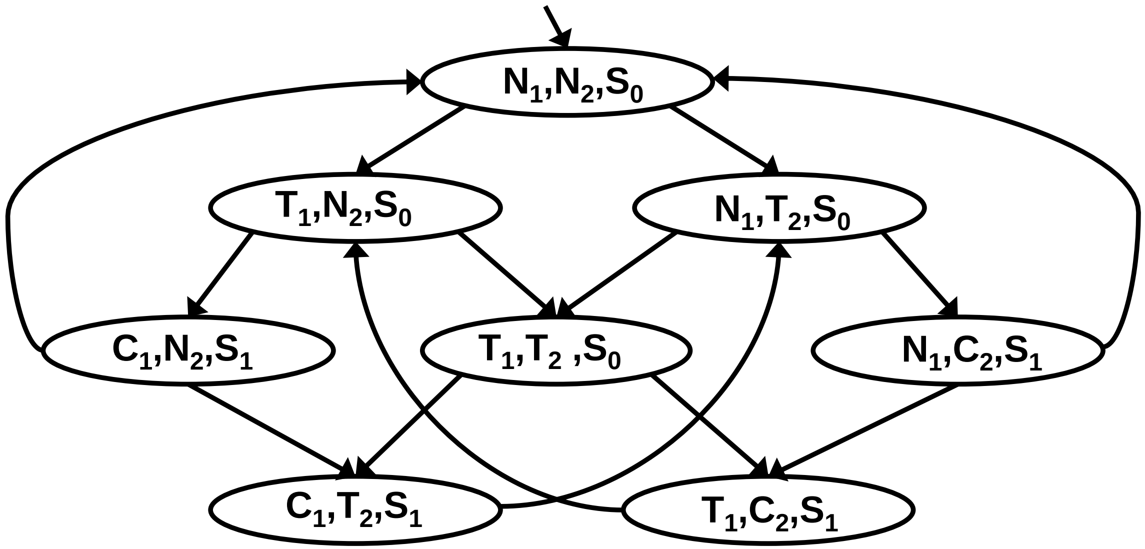
M ⊭ AG (($T_1 \rightarrow$ F C_1) ∧ ($T_2 \rightarrow$ F C_2))

Illustrative Example: Mutual Exclusion



- Does it hold that M ⊨ f?
  - Property 4: $f :=\ \boldsymbol{AG\,EF}\,(N_1 \wedge N_2 \wedge S_0)$
- How would you express Property 4 in natural language?
- In case M ⊭ f, compute a counterexample

Illustrative Example: Mutual Exclusion



- Does it hold that $M \vDash f$?
  - Property 4: $f := \mathbf{AG}\ \mathbf{EF}\ (N_1 \wedge N_2 \wedge S_0)$

$M \vDash \mathbf{AG}\ \mathbf{EF}\ (N_1 \wedge N_2 \wedge S_0)$

- ***"No matter where you are***
  there is always a way
  to get to the initial state (restart)"

Plan for Today

- Presentation of Homework and Recap of Temporal Logic
- Properties of CTL and LTL
  - LTL vs CTL
  - Counterexamples
  - Safety and Liveness Properties
- CTL Model Checking
  - MC Problem Definition
  - Illustrative Example for CTL Model Checking
  - Algorithm for CTL MC

CTL Model Checking Algorithm

Receives

- Given a Kripke structure M and a CTL formula f

MC Returns:

- Whether $M \models f$, i.e., M is a model for f

CTL Model Checking Algorithm

Receives

- Given a Kripke structure M and a CTL formula f

MC Returns:

- Whether $M \models f$, i.e., M is a model for f

Or (Alternative Definition):

- $[\![f]\!]_M = \{s \in S \mid M, s \models f\}$ i.e., all states satisfying f
  - M is omitted from $[\![f]\!]_M$ when clear from the context

CTL Model Checking $M \models f$

Iterative algorithm:
Compute $[\![g]\!]_M$ for every subformula g of f

CTL Model Checking $M \models f$

Iterative algorithm:
Compute $[\![g]\!]_M$ for every subformula g of f

- Work iteratively on subformulas of f
  - from **simpler** to **complex** subformulas

CTL Model Checking $M \models f$

Iterative algorithm:
Compute $[\![g]\!]_M$ for every subformula g of f

- Work iteratively on subformulas of f
  - from **simpler** to **complex** subformulas

- For checking **AG(** request $\rightarrow$ **AF** grant**)**

CTL Model Checking $M \models f$

Iterative algorithm:
Compute $[\![g]\!]_M$ for every subformula g of f

- Work iteratively on subformulas of f
  - from **simpler** to **complex** subformulas

- For checking **AG(** request $\rightarrow$ **AF** grant**)**
  - Check grant, request

CTL Model Checking $M \models f$

Iterative algorithm:
Compute $[\![g]\!]_M$ for every subformula g of f

- Work iteratively on subformulas of f
  - from **simpler** to **complex** subformulas

- For checking **AG(** request $\rightarrow$ **AF** grant**)**
  - Check grant, request
  - Then check **AF** grant

CTL Model Checking $M \vDash f$

Iterative algorithm:
Compute $[\![g]\!]_M$ for every subformula g of f

- Work iteratively on subformulas of f
  - from **simpler** to **complex** subformulas

- For checking **AG(** request $\rightarrow$ **AF** grant**)**
  - Check grant, request
  - Then check **AF** grant
  - Next check request $\rightarrow$ **AF** grant

CTL Model Checking $M \models f$

Iterative algorithm:
Compute $[\![g]\!]_M$ for every subformula g of f

- Work iteratively on subformulas of f
  - from **simpler** to **complex** subformulas

- For checking **AG(** request $\rightarrow$ **AF** grant**)**
  - Check grant, request
  - Then check **AF** grant
  - Next check request $\rightarrow$ **AF** grant
  - Finally check **AG(** request $\rightarrow$ **AF** grant**)**

CTL Model Checking $M \models f$

- For each s, computes label(s), which is the set of subformulas of f that are true in s

CTL Model Checking $M \vDash f$

- For each s, computes label(s), which is the set of subformulas of f that are true in s

For every subformula g of f:

- The algorithm adds g to label(s) for every state s that satisfies g
  - $g \in \text{label}(s) \Leftrightarrow M, s \vDash g$

CTL Model Checking $M \vDash f$

- For each s, computes label(s), which is the set of subformulas of f that are true in s

For every subformula g of f:

- The algorithm adds g to label(s) for every state s that satisfies g
  - $g \in$ label(s) $\Leftrightarrow M, s \vDash g$

$M \vDash f$ if and only if $f \in$ label(s) for all initial states $s \in S_0$ of M

Minimal set of operators for CTL

- All CTL formulas can be transformed to use only the operators:
  - $\neg$, $\vee$, **EX**, **EU**, **EG**
- MC algorithm needs to handle **AP** (atomic propositions) and **$\neg$, $\vee$, EX, EU, EG**

Model Checking $AP, \neg, \vee$- Formulas

Procedure for labeling the states:

- For $p \in AP$
  - $p \in \text{label}(s)$ if and only if $\underbrace{p \in \mathrm{L}(s)}_{\text{Defined by } M}$

Model Checking $AP, \neg, \vee$- Formulas

Procedure for labeling the states:

- For $p \in AP$
  - $p \in$ label(s) if and only if $p \in$ L(s)
- For subformulas f_1 and f_2 that have already been checked (added to label(s), when needed)

Model Checking $AP, \neg, \vee$- Formulas

Procedure for labeling the states:

- For $p \in AP$
  - $p \in$ label(s) if and only if $p \in$ L(s)
- For subformulas f_1 and f_2 that have already been checked (added to label(s), when needed)
  - $\neg f_1$ add to label(s) if and only if $f_1 \notin label(s)$

Model Checking $AP, \neg, \vee$- Formulas

Procedure for labeling the states:

- For $p \in AP$
  - $p \in$ label(s) if and only if $p \in$ L(s)
- For subformulas f_1 and f_2 that have already been checked (added to label(s), when needed)
  - $\neg f_1$ add to label(s) if and only if $f_1 \notin label(s)$
  - $f_1 \vee f_2$ add to label(s) if and only if
    $f_1 \in labels(s)$ **or** $f_2 \in label(s)$

Model Checking AP, $\neg$,$\vee$- Formulas

Procedure for labeling the states:

- For $p \in AP$
  - $p \in label(s)$ if and only if $p \in L(s)$
- For subformulas f_1 and f_2 that have already been checked (added to label(s), when needed)
  - $\neg f_1$ add to $label(s)$ if and only if $f_1 \notin label(s)$
  - $f_1 \vee f_2$ add to $label(s)$) if and only if $f_1 \in label(s)$ **or** $f_2 \in label(s)$

Give the procedures for labeling states satisfying $EX f_1$

Model Checking $AP, \neg, \vee$- Formulas

Procedure for labeling the states:

- For $p \in AP$
  - $p \in label(s)$ if and only if $p \in \mathrm{L}(s)$
- For subformulas f_1 and f_2 that have already been checked (added to label(s), when needed)
  - $\neg f_1$ add to $label(s)$ if and only if $f_1 \notin label(s)$
  - $f_1 \vee f_2$ add to $label(s)$) if and only if $f_1 \in label(s)$ **or** $f_2 \in label(s)$

Give the procedures for labeling states satisfying $\mathrm{EX} f_1$

- Add $\mathrm{EX} f_1$ to $label(s)$ if and only if s has a successor t such that $f_1 \in label(t)$

Model Checking $g = EX\, f_1$

- Give the procedures for labeling states satisfying $EX f_1$
  - Add g to label(s) if and only if s has a successor t such that $f_1 \in$ label(t)

Model Checking $g = EX\, f_1$

- Add $EX f_1$ to $label(s)$ if and only if s has a successor t such that $f_1 \in label(t)$

```
procedure CheckEX (f1)
    T := { t | f1 ∈ label(t) }
    while T ≠ ∅  do
        choose t ∈T;  T := T \ {t};
        for all s  such that  R(s,t) do
            if EX f1 ∉ label(s) then
                label(s) : = label(s) ∪ { EX f1};
```

Model Checking $g = E(f_1 U f_2)$

ToDo Procedures for labeling states satisfying $E(f_1 U f_2)$

- Think how you can rewrite the procedure CheckEX

```
procedure CheckEX (f1)
  T := { t | f1 ∈ label(t) }

while T ≠ ∅  do
    choose t ∈T;  T := T \ {t};
    for all s  such that  R(s,t) do
        if EX f1 ∉ label(s) then
            label(s) : = label(s) ∪ { EX f1};
```

```
procedure CheckEU (f1,f2)
  T :=

  for all t∈T do
     label(t) :=

  while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
```

Model Checking $g = E(f_1 U f_2)$

- Procedures for labeling states satisfying $E(f_1 U f_2)$
  - Rewriting the procedure CheckEX

```
procedure CheckEX (f1)
  T := { t | f1 ∈ label(t) }

while T ≠ ∅  do
    choose t ∈T;  T := T \ {t};
    for all s  such that  R(s,t) do
        if EX f1 ∉ label(s) then
           label(s) : = label(s) ∪ { EX f1};
```

```
procedure CheckEU (f1,f2)
  T := { t | f2 ∈ label(t) }

  for all t∈T do
     label(t) :=

  while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
```

Model Checking $g = E(f_1 U f_2)$

- Procedures for labeling states satisfying $E(f_1 U f_2)$
  - Rewriting the procedure CheckEX

```
procedure CheckEX (f1)
  T := { t | f1 ∈ label(t) }

while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
          if EX f1 ∉ label(s) then
             label(s) : = label(s) ∪ { EX f1};
```

```
procedure CheckEU (f1,f2)
  T := { t | f2 ∈ label(t) }

  for all t∈T do
     label(t) := label(t) ∪ { E(f1 U f2) }

  while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
```

Model Checking $g = E(f_1 U f_2)$

- Procedures for labeling states satisfying $E(f_1 U f_2)$
  - Rewriting the procedure CheckEX

```
procedure CheckEX (f1)
  T := { t | f1 ∈ label(t) }

while T ≠ ∅  do
    choose t ∈T;  T := T \ {t};
    for all s  such that  R(s,t) do
        if EX f1 ∉ label(s) then
           label(s) : = label(s) ∪ { EX f1};
```

```
procedure CheckEU (f1,f2)
  T := { t | f2 ∈ label(t) }

  for all t∈T do
     label(t) := label(t) ∪ { E(f1 U f2) }

  while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
        if E(f1 U f2) ∉ label(s) and f1 ∈ label(s) then
           label(s) : = label(s) ∪ {E(f1 U f2) };
```

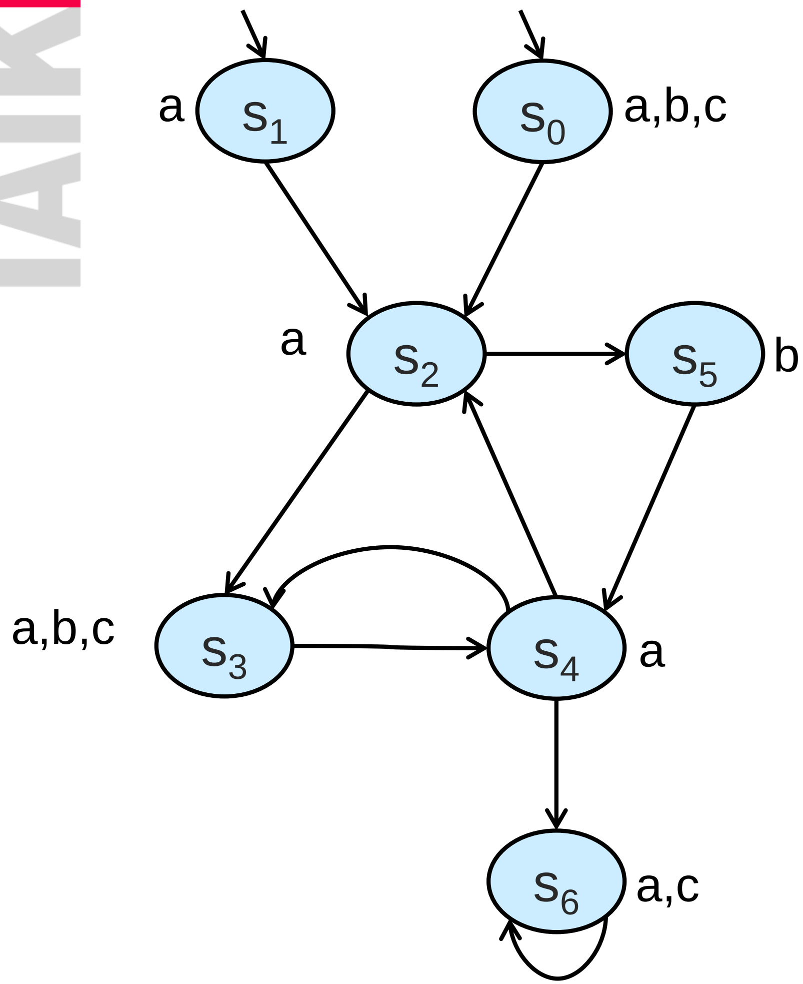
Model Checking $g = E(f_1 U f_2)$

- Procedures for labeling states satisfying $E(f_1 U f_2)$
  - Rewriting the procedure CheckEX

```
procedure CheckEX (f1)
  T := { t | f1 ∈ label(t) }

while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
          if EX f1 ∉ label(s) then
               label(s) : = label(s) ∪ { EX f1};
```

```
procedure CheckEU (f1,f2)
  T := { t | f2 ∈ label(t) }

  for all t∈T do
     label(t) := label(t) ∪ { E(f1 U f2) }

  while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
          if E(f1 U f2) ∉ label(s) and f1 ∈ label(s) then
               label(s) : = label(s) ∪ {E(f1 U f2) };
               T : = T ∪ {s}
```

Example: Model Checking U Formulas



Does it hold that M ⊨ f?

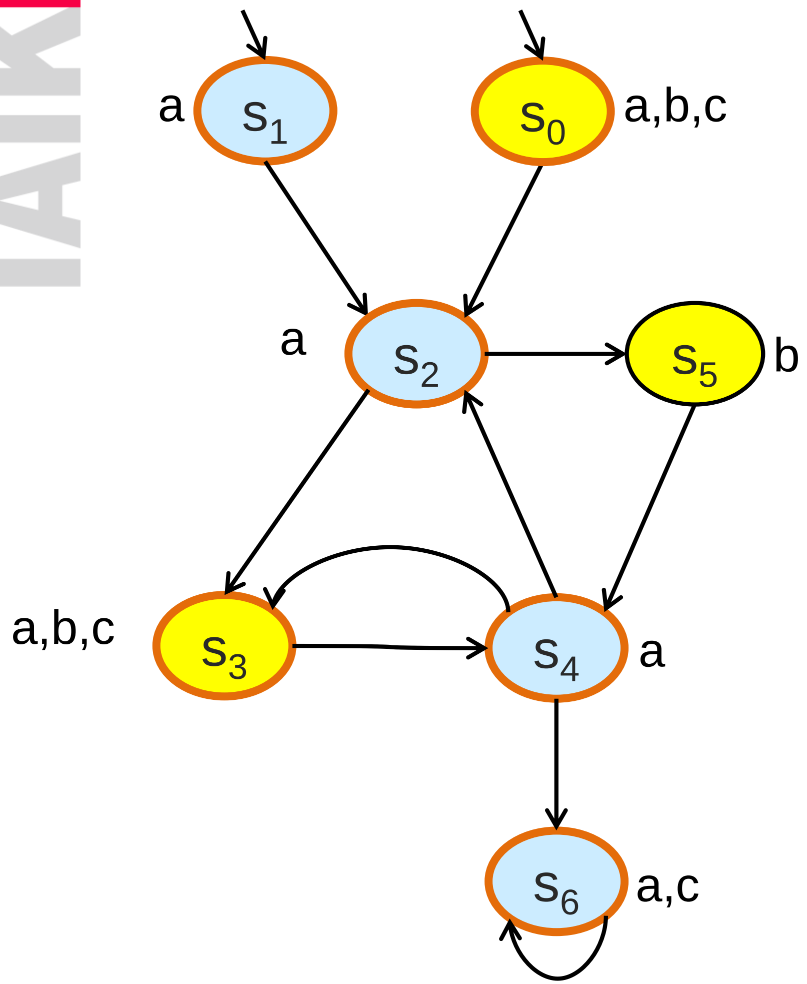
- $f := E(aUb)$

procedure CheckEU (f_1,f_2)
  T := { t | f_2 ∈ label(t) }

  for all t∈T **do**
    label(t) := label(t) ∪ { E(f_1 U f_2) }

  while T ≠ ∅ **do**
    choose t ∈T; T := T \ {t};
    for all s **such that** R(s,t) **do**
      if E(f_1 U f_2) ∉ label(s) **and** f_1 ∈ label(s) **then**
        label(s) : = label(s) ∪ {E(f_1 U f_2) };
        T : = T ∪ {s}

Example: Model Checking U Formulas



Does it hold that M ⊨ f?

- $f := E(aUb)$

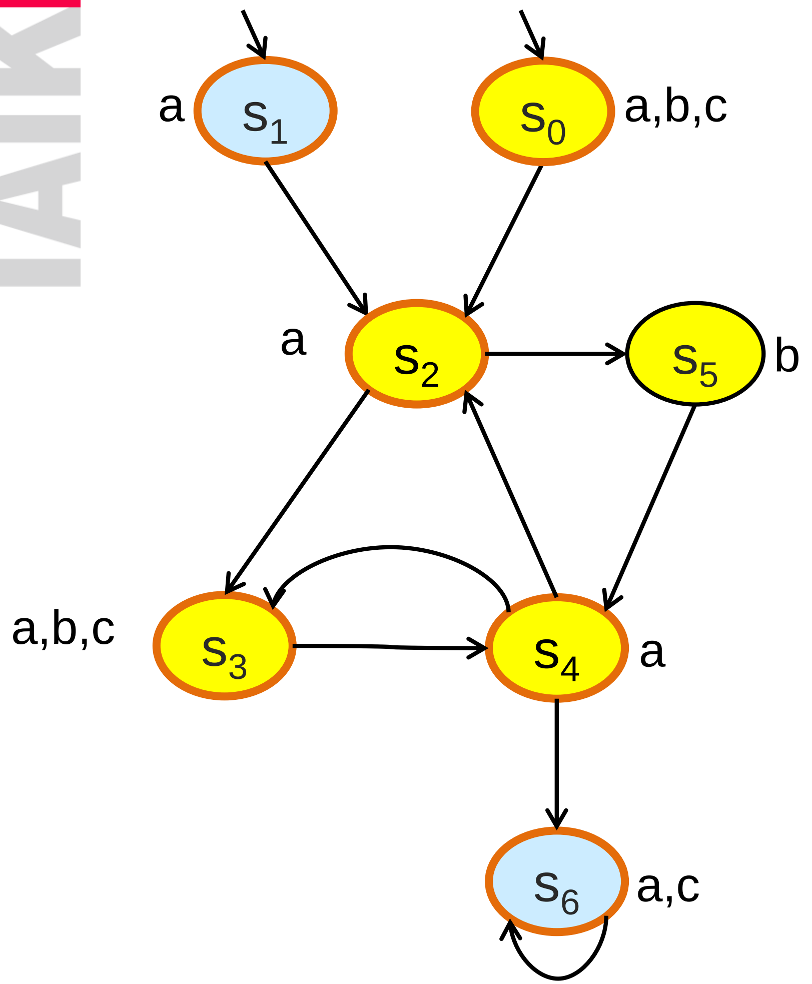
[[E(aUb)]] = {0,3,5}

procedure CheckEU (f_1,f_2)
  T := { t | f_2 ∈ label(t) }

  for all t∈T do
    label(t) := label(t) ∪ { E(f_1 U f_2) }

  while T ≠ ∅ do
    choose t ∈T; T := T \ {t};
    for all s such that R(s,t) do
      if E(f_1 U f_2) ∉ label(s) and f_1 ∈ label(s) then
        label(s) : = label(s) ∪ {E(f_1 U f_2) };
        T : = T ∪ {s}

Example: Model Checking *U* Formulas



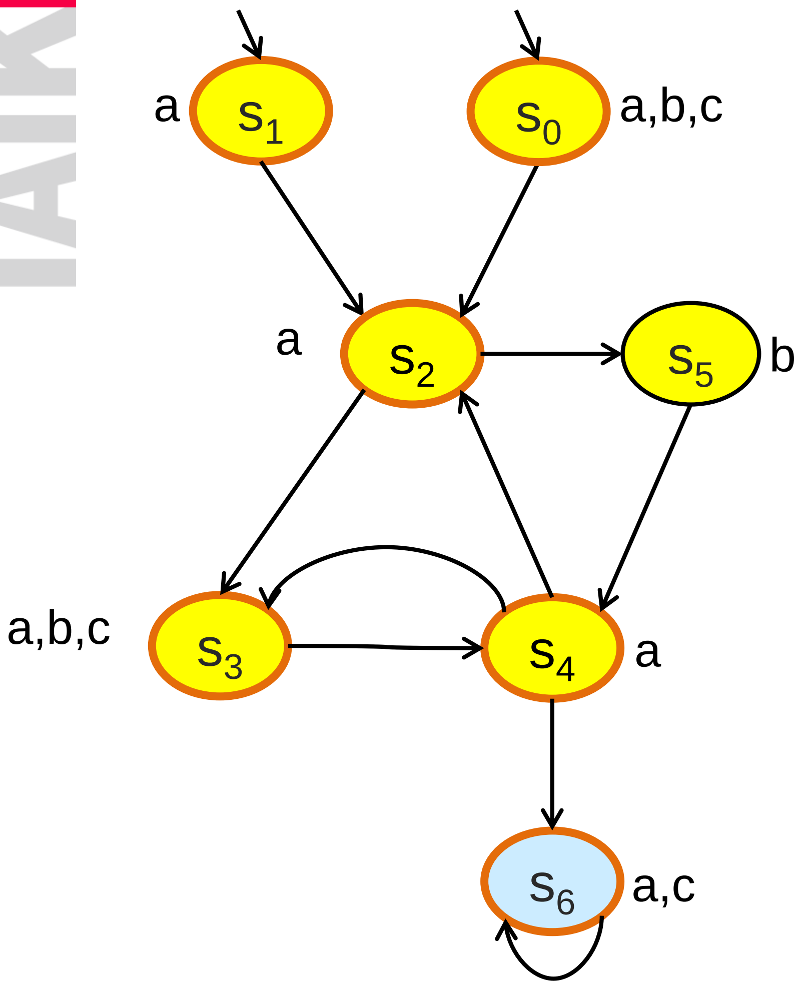
Does it hold that M ⊨ f?

- $f := E(aUb)$

[[E(aUb)]] = {0,2,3,4,5}

```
procedure CheckEU (f1,f2)
  T := { t | f2 ∈ label(t) }

  for all t∈T do
     label(t) := label(t) ∪ { E(f1 U f2) }

  while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
        if E(f1 U f2) ∉ label(s) and f1 ∈ label(s) then
           label(s) : = label(s) ∪ {E(f1 U f2) };
           T : = T ∪ {s}
```

Example: Model Checking *U* Formulas



Does it hold that M ⊨ f?

- $f := E(aUb)$

✓ M ⊨ E(aUb)

[[E(aUb)]] = {0,1,2,3,4,5}

```
procedure CheckEU (f1,f2)
  T := { t | f2 ∈ label(t) }

  for all t∈T do
     label(t) := label(t) ∪ { E(f1 U f2) }

  while T ≠ ∅  do
     choose t ∈T;  T := T \ {t};
     for all s  such that  R(s,t) do
        if E(f1 U f2) ∉ label(s) and f1 ∈ label(s) then
           label(s) : = label(s) ∪ {E(f1 U f2) };
           T : = T ∪ {s}
```

Model Checking $g = EGf_1$

Observation:

s ⊨ **EG** f_1

if and only if

There is a path π, starting at s, such that $\pi \models$ **G** f_1

Model Checking $g = EGf_1$

Observation:

s ⊨ **EG** f_1

if and only if

There is a path π, starting at s, such that $\pi \models \mathbf{G}\, f_1$

if and only if

There is a path from s to a strongly connected component, where all states satisfy f_1

Model Checking $g = EGf_1$

- A Strongly Connected Component (SCC) in a graph
  is a subgraph C such that every node in C is reachable from any other node in C via nodes in C

Model Checking $g = EGf_1$

- A Strongly Connected Component (SCC) in a graph is a subgraph C such that every node in C is reachable from any other node in C via nodes in C

- An SCC C is maximal (MSCC) if it is not contained in any other SCC in the graph
  - Possible to find all MSCC in linear time O(|S|+|R|) (Tarjan)

Model Checking $g = EGf_1$

- A Strongly Connected Component (SCC) in a graph
  is a subgraph C such that every node in C is reachable from any other node in C via nodes in C

- An SCC C is maximal (MSCC) if it is not contained in any other SCC in the graph
  - Possible to find all MSCC in linear time O(|S|+|R|) (Tarjan)

- C is nontrivial if it contains at least one edge.
  Otherwise, it is trivial.

Model Checking $g = EGf_1$

- Reduced structure for M and f_1:
  - Remove from M all states such that $f_1 \notin labels(s)$

Model Checking $g = EGf_1$

- Reduced structure for M and f_1:
  - Remove from M all states such that $f_1 \notin labels(s)$

- Resulting model: $M' = (S', R', L')$
  - $S' = \{ s \mid M, s \vDash f_1\}$
  - $R' = (S' \times S') \cap R$
  - $L'(s') = L(s')$ for every $s' \in S'$

Model Checking $g = EGf_1$

- Reduced structure for M and f_1:
  - Remove from M all states such that $f_1 \notin labels(s)$

- Resulting model: $M' = (S', R', L')$
  - $\mathrm{S}' = \{\, \mathrm{s} \mid \mathrm{M}, s \vDash f_1\}$
  - $\mathrm{R}' = (S' \times S') \cap R$
  - $L'(s') = L(s')$ for every $s' \in S'$

- Theorem: $M, s \vDash EG\ f_1$ if and only if

  $s \in S'$ and

  there is $a\ path$ in M' from s to some state t

  in a nontrivial MSCC of M'.

Model Checking $g = EGf_1$

procedure CheckEG (f_1)
S′ := {s | $f_1 \in$ label(s) }
MSCC := { C | C is a nontrivial MSCC of M′ }
T := $\cup_{C \in MSCC}$ { s | s $\in$ C}

for all t$\in$T **do**
label(t) := label(t) $\cup$ { EG f_1}

Model Checking $g = EG f_1$

procedure CheckEG (f_1)
  $S' := \{s \mid f_1 \in label(s)\}$
  MSCC := { C | C is a nontrivial MSCC of M′ }
  $T := \cup_{C \in MSCC} \{ s \mid s \in C\}$

  for all $t \in T$ do
    $label(t) := label(t) \cup \{ EG\ f_1\}$

  while $T \neq \varnothing$ do
    choose $t \in T$; $T := T \setminus \{t\}$;
    for all $s \in S'$ such that $R'(s,t)$ do
      if $EG\ f_1 \notin label(s)$ then
        $label(s) := label(s) \cup \{EG\ f_1\}$;
        $T := T \cup \{s\}$

Model Checking Complexity

Steps per Subformula

- MC Atomic Propositions
    -
- MC ¬, ∨ formulas
    -
- MC g = EX f_1
    -
- MC $g = E(f_1 U f_2)$
    -
- MC $g = EG f_1$

Model Checking Complexity

Steps per Subformula

- MC Atomic Propositions
  - O(|S|) steps
- MC ¬, ∨ formulas
  -
- MC g = EX f_1
  -
- MC $g = E(f_1 U f_2)$
  -
- MC $g = EG f_1$

Model Checking Complexity

Steps per Subformula

- MC Atomic Propositions
  - O(|S|) steps
- MC $\neg, \vee$ formulas
  - O(|S|) steps
- MC g = EX f_1
  -
- MC $g = E(f_1 U f_2)$
  -
- MC $g = EG f_1$

Model Checking Complexity

Steps per Subformula

- MC Atomic Propositions
  - O(|S|) steps
- MC $\neg$, $\vee$ formulas
  - O(|S|) steps
- MC g = EX f_1
  - Add g to label(s) iff s has a successor t such that $f_1 \in$ label(t)
  - O(|S| + |R|)
- MC $g = E(f_1 U f_2)$
  -
- MC $g = EG f_1$

Model Checking Complexity

Steps per Subformula

- MC Atomic Propositions
  - O(|S|) steps
- MC $\neg$, $\vee$ formulas
  - O(|S|) steps
- MC g = EX f_1
  - Add g to label(s) iff s has a successor t such that $f_1 \in$ label(t)
  - O(|S| + |R|)
- MC $g = E(f_1 U f_2)$
  - O(|S| + |R|)
- MC $g = EG f_1$

Model Checking Complexity

Steps per Subformula

- MC $g = EGf_1$
  - Computing M′ : O (|S| + |R|)
  - Computing MSCCs using Tarjan's algorithm: O (|S′| + |R′|)
  - Labeling all states in MSCCs: O (|S′|)
  - Backward traversal: O (|S′| + |R′|)

  - => Overall: O (|S| + |R|)

Model Checking Complexity

Steps per Subformula

- MC Atomic Propositions
  - O(|S|) steps
- MC $\neg$, $\vee$ formulas
  - O(|S|) steps
- MC $g = EX\ f_1$
  - Add g to label(s) iff s has a successor t such that $f_1 \in$ label(t)
  - O(|S| + |R|)
- MC $g = E(f_1 U f_2)$
  - O(|S| + |R|)
- MC $g = EG f_1$
  - O(|S| + |R|)

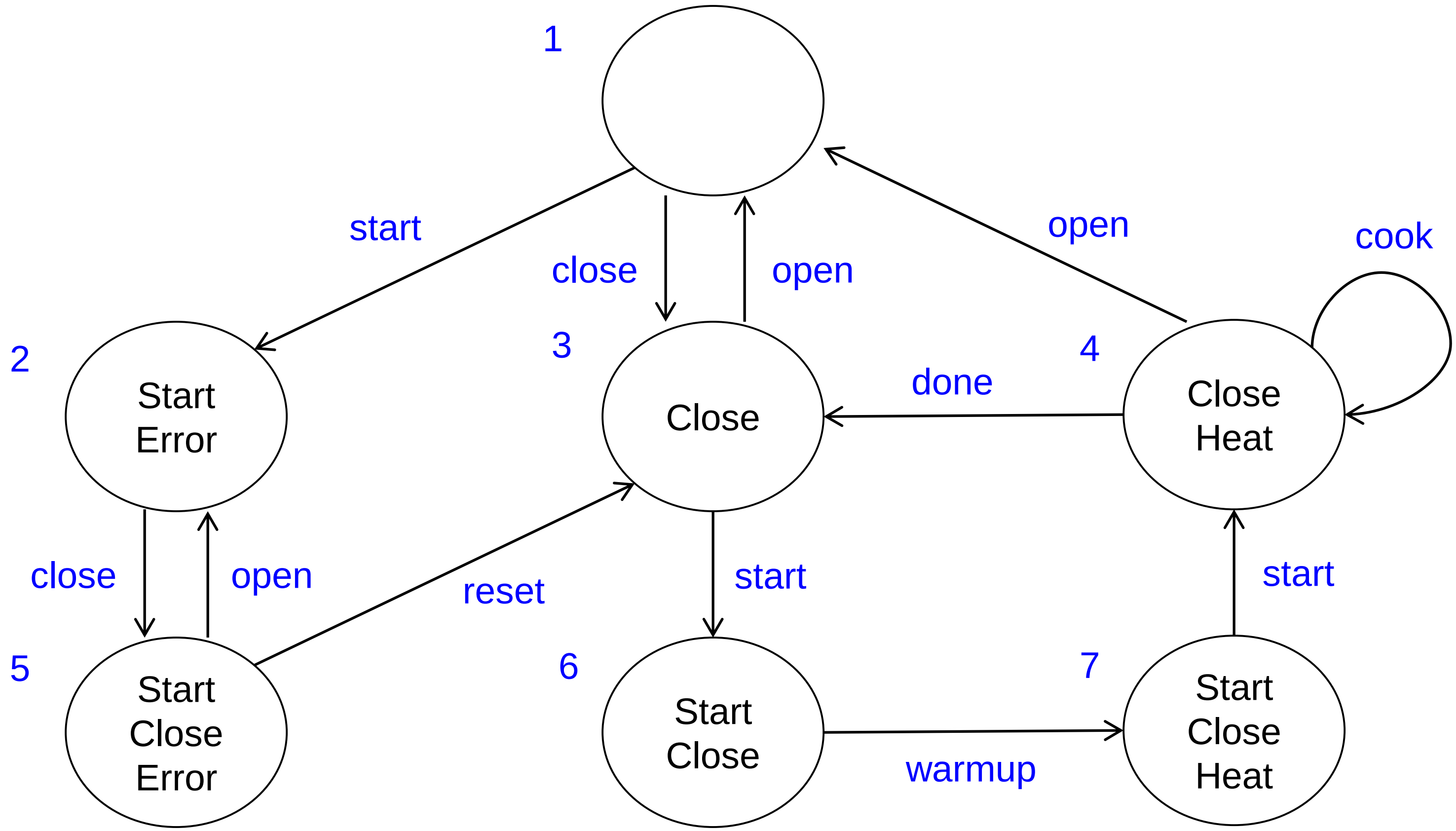
Model Checking Complexity

- Each subformula
  - $O(|S|+|R|) = O(|M|)$
- Number of subformulas in f:
  - $O(|f|)$
- Total
  - $O(|M| \times |f|)$

- For comparison
  - Complexity of MC for LTL and CTL* is $O(|M| \times 2^{|f|})$

Microwave Example

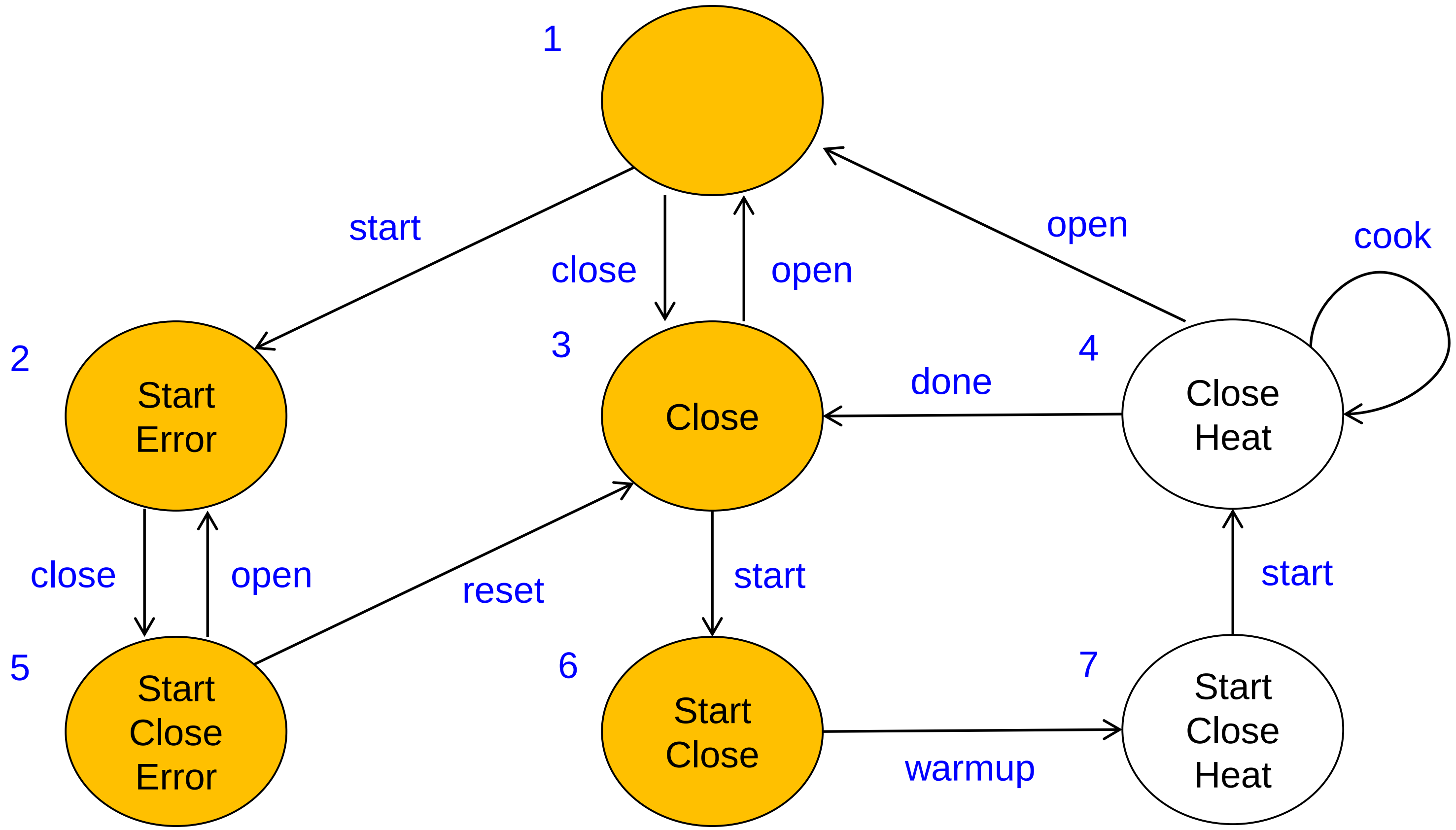
- Use the proposed algorithm to compute if $M \models f$?
  - $f := \neg E\ (\mathrm{true}\ U\ (\mathrm{Start} \wedge EG\ \neg \mathrm{Heat}))$



$$f := \neg E\ (true\ U\ (Start \wedge EG\ \neg Heat))$$

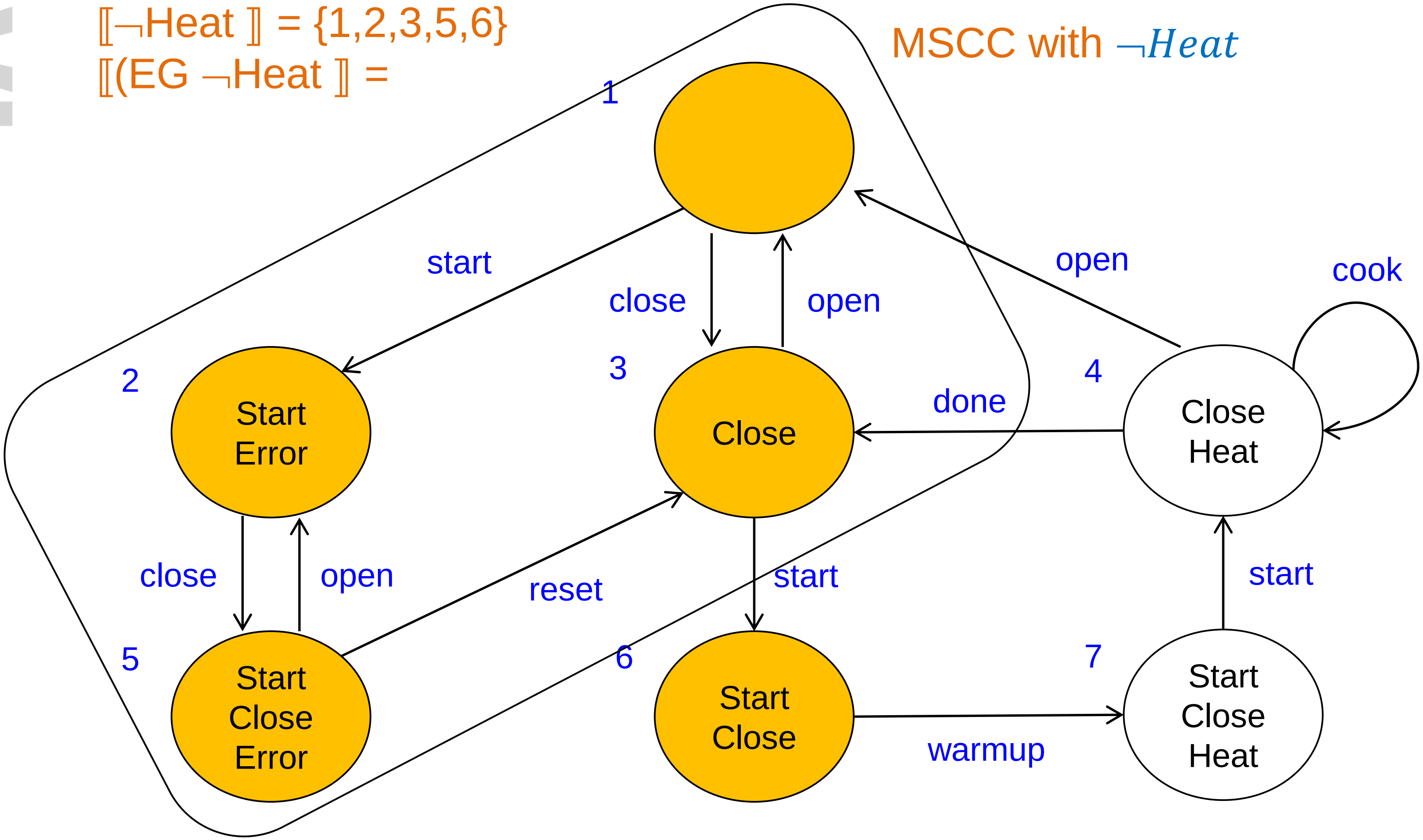
$[\![start]\!] = \{2,5,6,7\}$

$[\![\neg Heat]\!] = \{1,2,3,5,6\}$



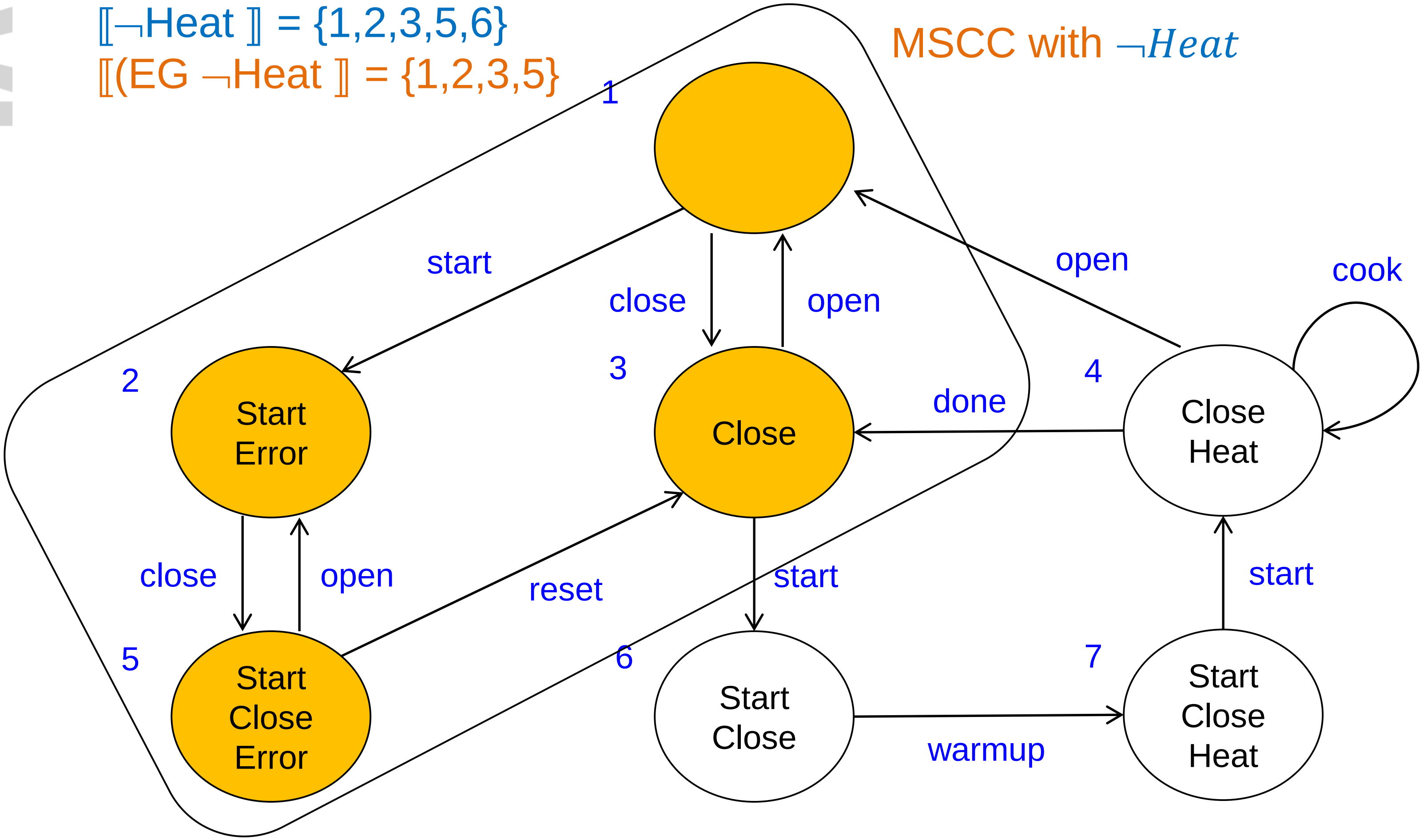
$$f := \neg E\ (true\ U\ (Start \wedge EG\ \neg Heat))$$

⟦start⟧ = {2,5,6,7}
⟦¬Heat ⟧ = {1,2,3,5,6}
⟦(EG ¬Heat ⟧ =



$$f := \neg E\ (true\ U\ (Start \wedge EG\ \neg Heat))$$

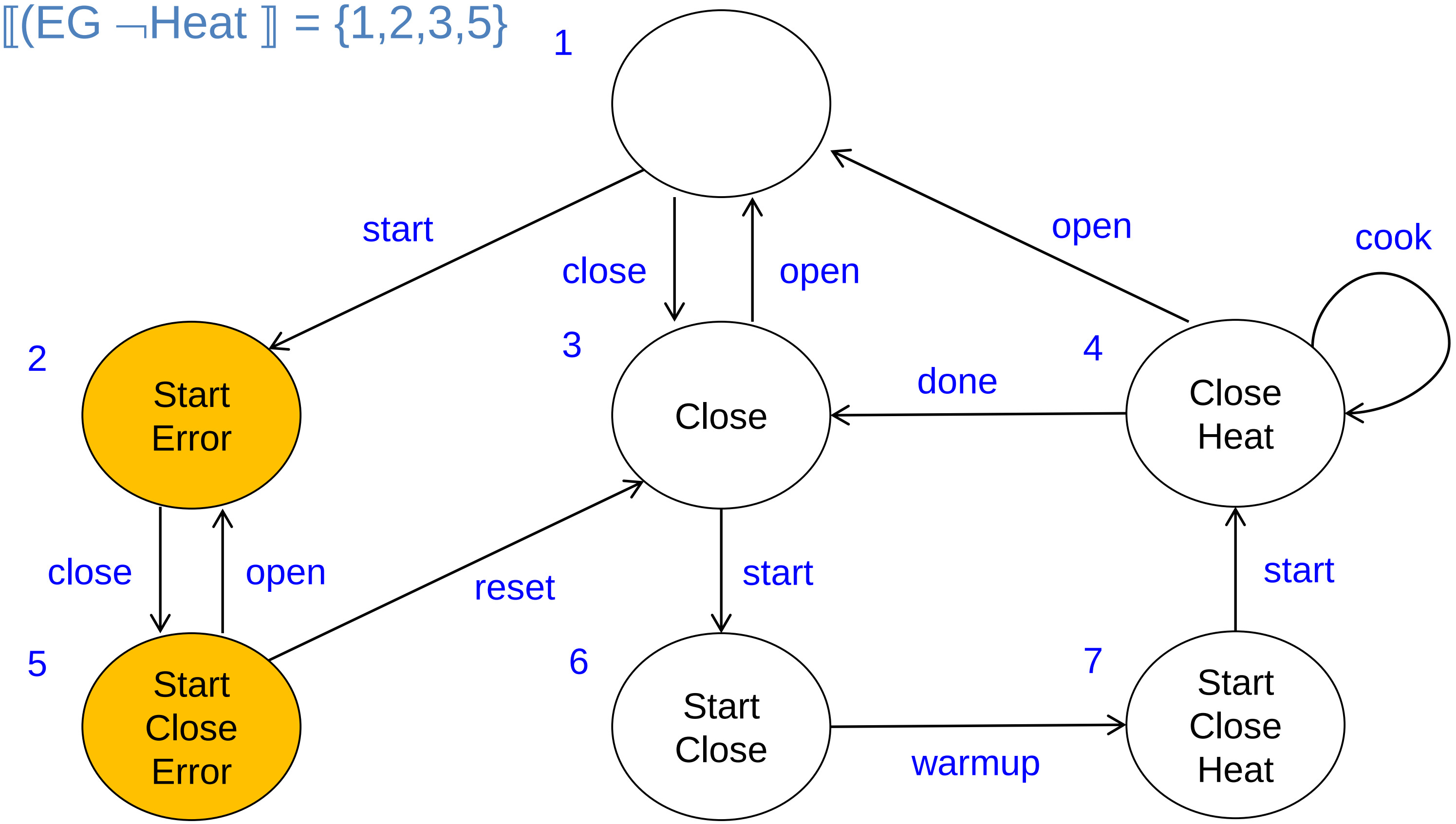
⟦start⟧ = {2,5,6,7}
⟦¬Heat ⟧ = {1,2,3,5,6}
⟦(EG ¬Heat ⟧ = {1,2,3,5}



$$f := \neg E\ (true\ U\ (Start \wedge EG\ \neg Heat))$$

⟦start⟧ = {2,5,6,7}
⟦¬Heat ⟧ = {1,2,3,5,6}
⟦(EG ¬Heat ⟧ = {1,2,3,5}

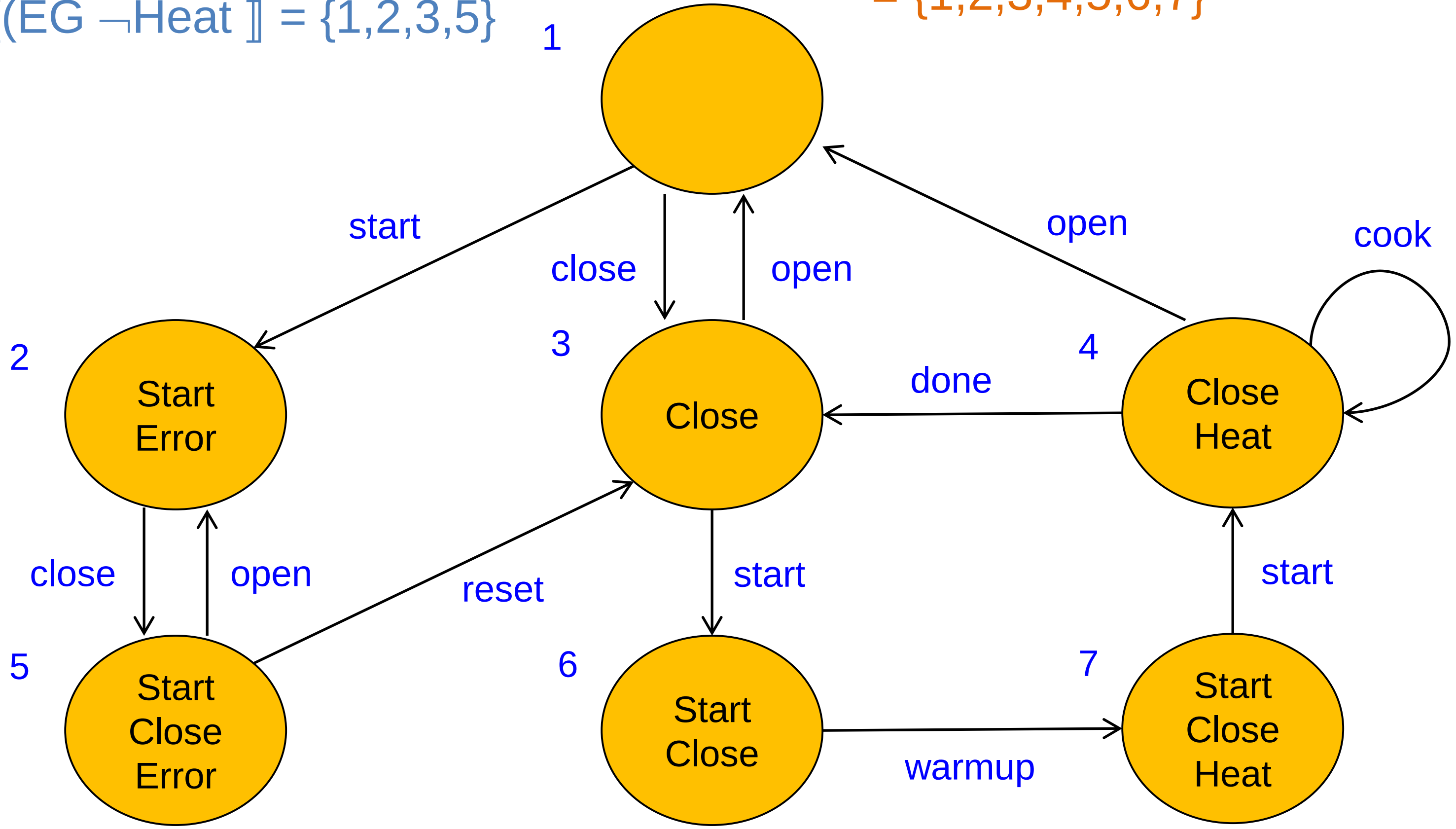
⟦ Start ∧ EG ¬Heat ⟧ = {2, 5}



$$f := \neg E\ (true\ U\ (Start \wedge EG\ \neg Heat))$$

⟦start⟧ = {2,5,6,7}
⟦¬Heat ⟧ = {1,2,3,5,6}
⟦(EG ¬Heat ⟧ = {1,2,3,5}

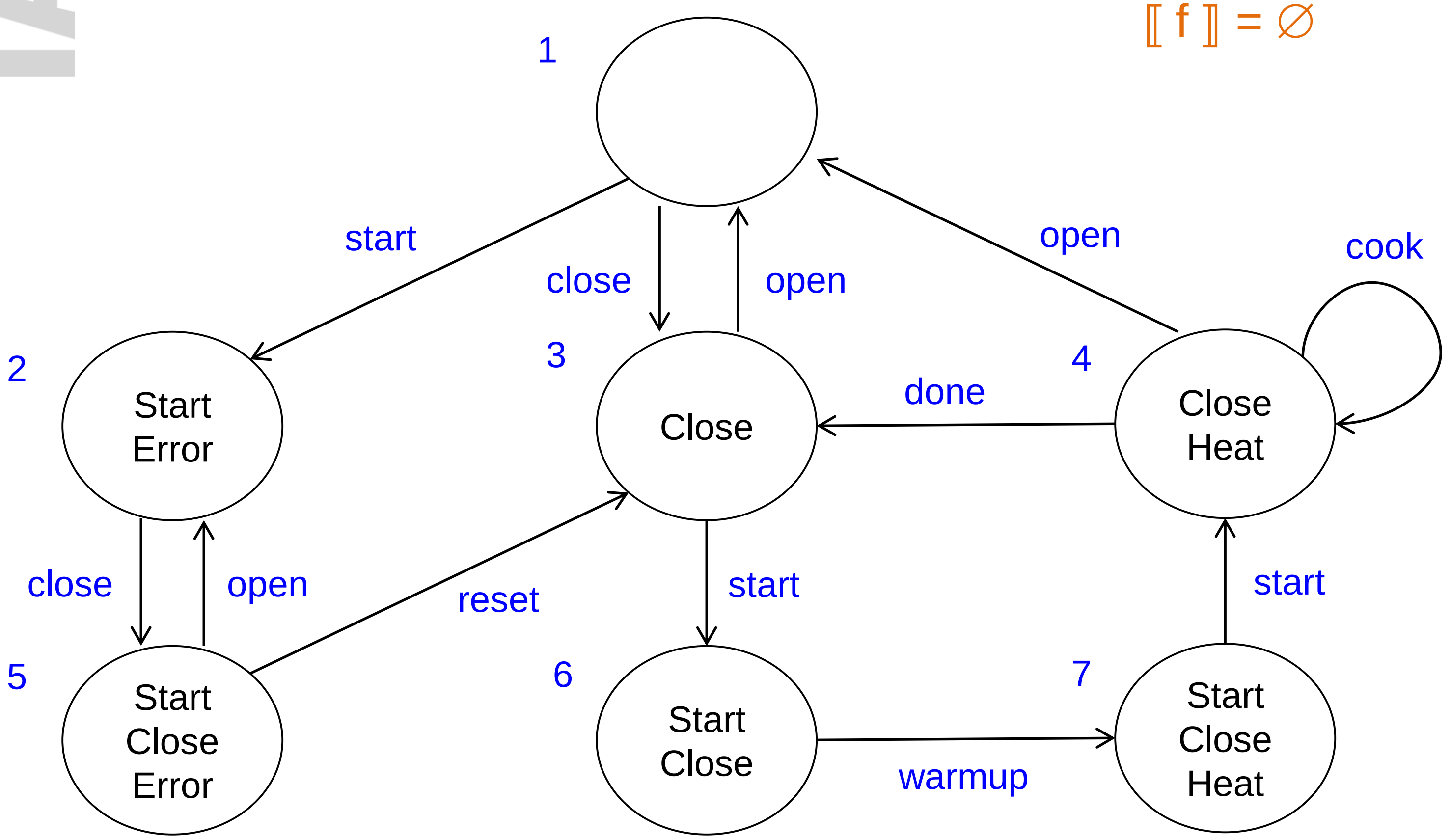
⟦ Start ∧ EG ¬Heat ⟧ = {2, 5}
⟦ E (true U (Start ∧ EG ¬Heat)) ⟧
= {1,2,3,4,5,6,7}



$f := \neg E\,(true\ U\ (Start \wedge EG\ \neg Heat))$

$[\![start]\!] = \{2,5,6,7\}$
$[\![\neg Heat]\!] = \{1,2,3,5,6\}$
$[\![(EG\ \neg Heat]\!] = \{1,2,3,5\}$

$[\![\ Start \wedge EG\ \neg Heat\ ]\!] = \{2, 5\}$
$[\![\ EU\ ]\!] = \{1,2,3,4,5,6,7\}$
$[\![\ f\ ]\!] = \varnothing$



Secure & Correct Systems