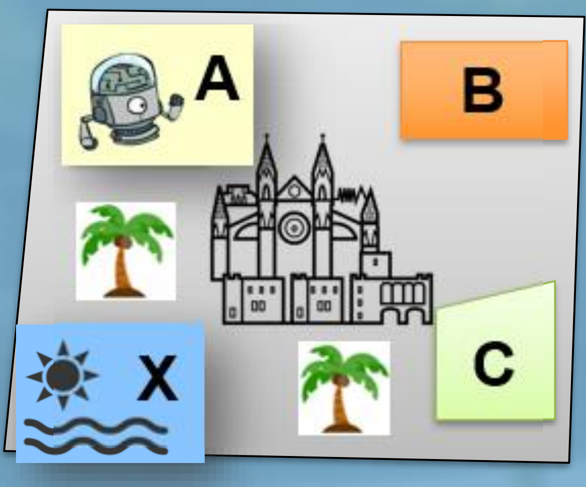
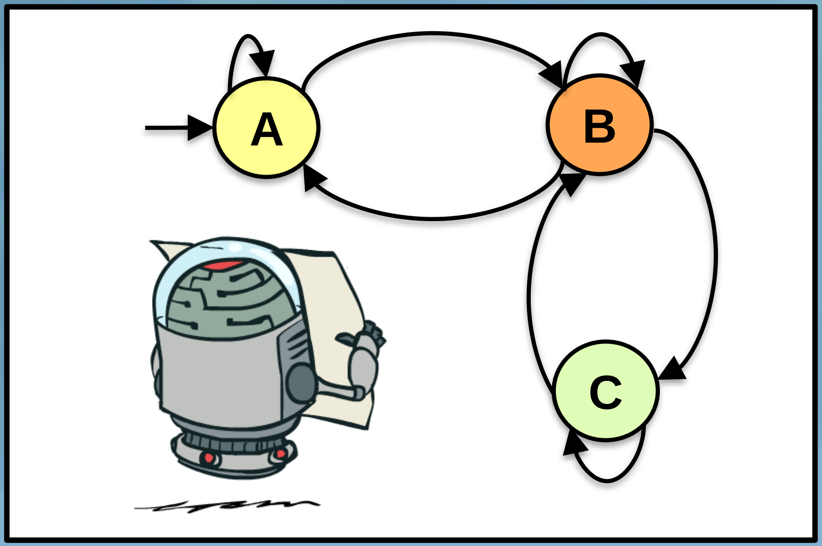
Graz University of Technology
Institute for Applied Information
Processing and Communications

Temporal Logic +
CTL Model Checking





Model Checking SS23
Bettina Könighofer

May 4th, 2023

Part 1 –
Properties of CTL / LTL

Linear Temporal Logic
LTL

LTL is the set of all state formulas, defined below:

State formulas:

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

Path formulas:

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

  where f_1 and f_2 are path formulas

Computation Tree Logic
CTL

CTL is the set of all state formulas, defined below:

- $p \in AP$
- $\neg g_1$, $g_1 \vee g_2$, $g_1 \wedge g_2$
- **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** $g_2)$

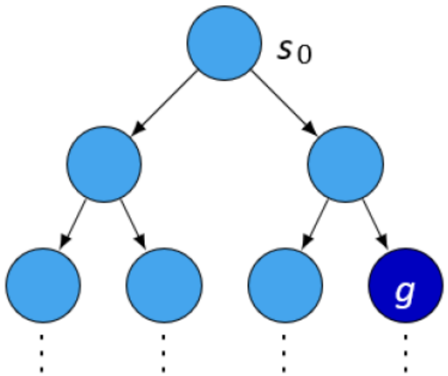
  where g_1 and g_2 are state formulas

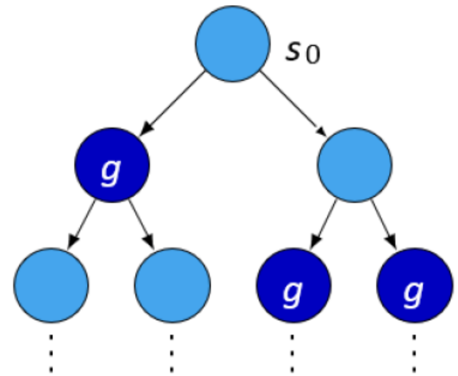
Note, that all sub-formulas of a CTL formula are state formulas

Illustration of CTL Semantics

EFg

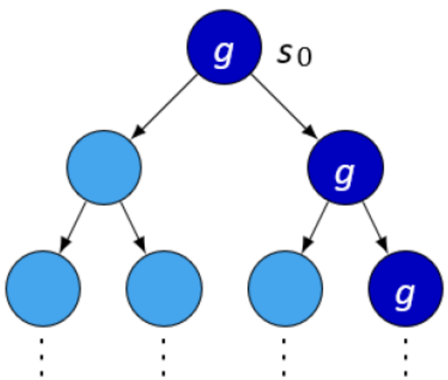
“exists reachable state such that…”

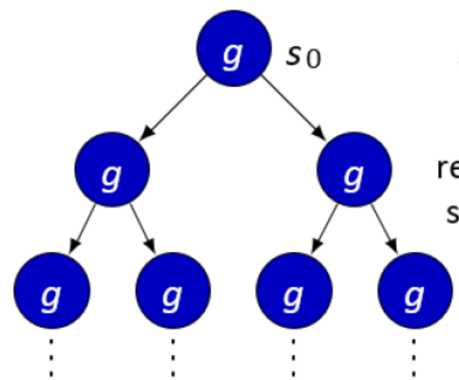




AFg

EGg

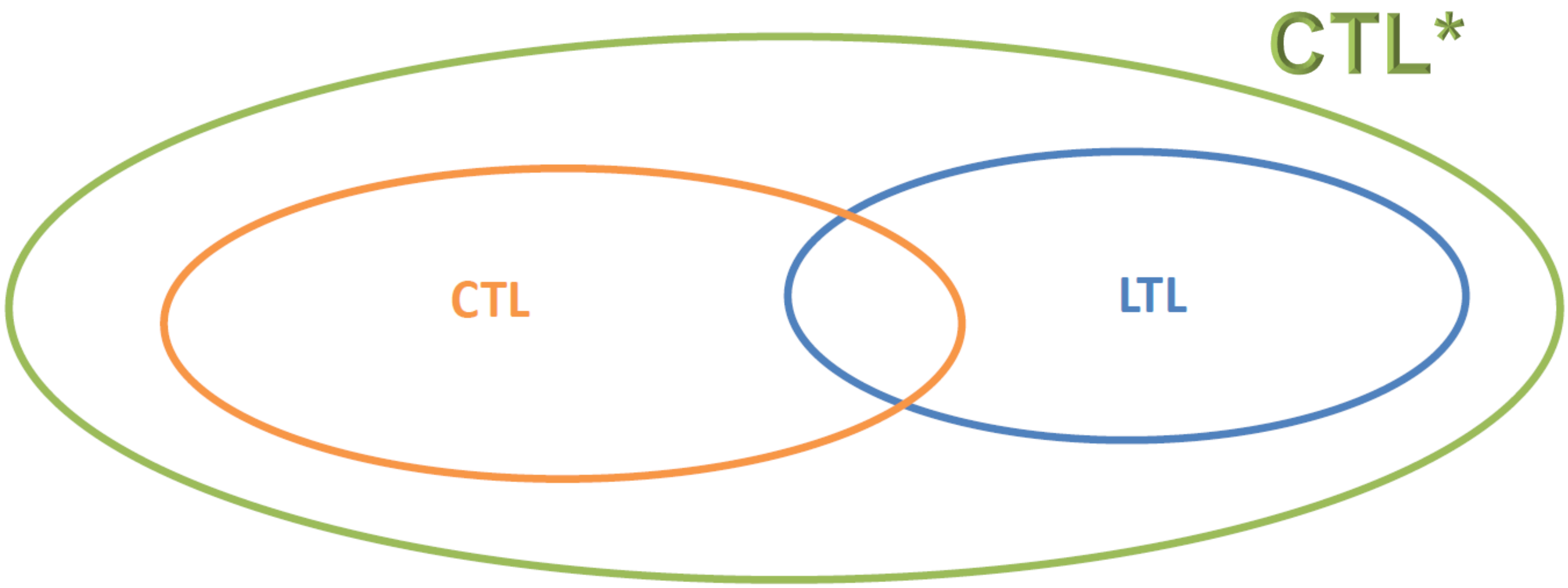




AGg

“all reachable states…”

Relationship between LTL and CTL

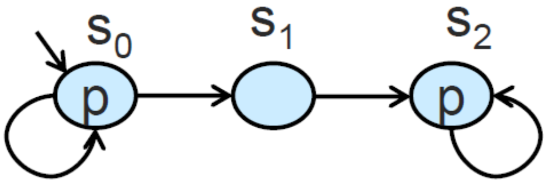


LTL vs CTL

- 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*"

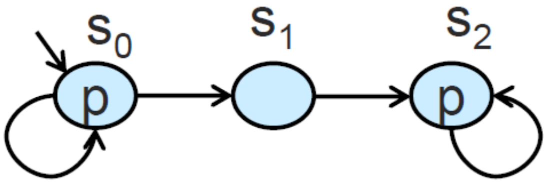
LTL vs CTL

- 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?



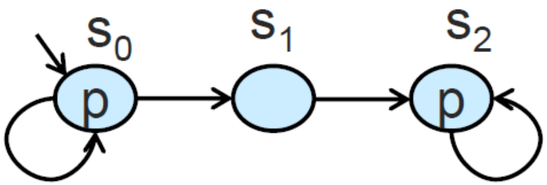
LTL vs CTL

- 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

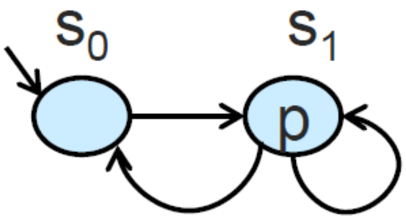
- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
  - $\boldsymbol{AFG\ p}$ = "*for all paths, eventually p always holds*"
- Solution: No
  - But what about: **AFAGp?**
  - **AFAGp** = "*for all paths, there is a point from which all reachable states satisfy p*"
  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



LTL vs CTL

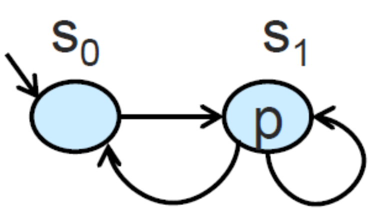
- Exercise:
  - Does the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
  - $\boldsymbol{AFG\ p}$ = "*for all paths, eventually p always holds*"
- Solution: No
  - What about **AFEG p?**

Hint:



LTL vs CTL

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

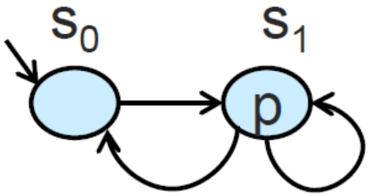


All paths satisfy **FEG**p

- since s_1 sat **EG**p

LTL vs CTL

- Exercise:
  - Dies the LTL formula $\boldsymbol{AFG\ p}$ has an equivalent in CTL?
- Solution: No
  - What about **AFEG p?**
    - “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

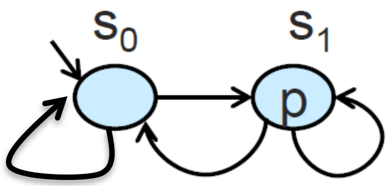
But $s_0,s_1,s_0,s_1,s_0,s_1\ldots$ does not satisfy **FG**p

LTL vs CTL

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

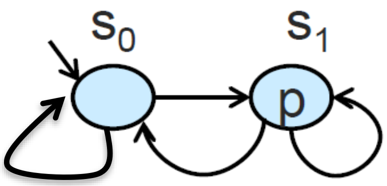


LTL vs CTL

- 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
  - 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
- CTL* is more expressive than either of them

Counterexamples

- Counterexample generation is a central feature of MC

- Given M and φ, such that $M \not\models \varphi$, a counterexample is a behavior of M, demonstrating the violation of φ in M

- To be useful for debugging it should
  - have finite representation
  - be easy-to-understand by human

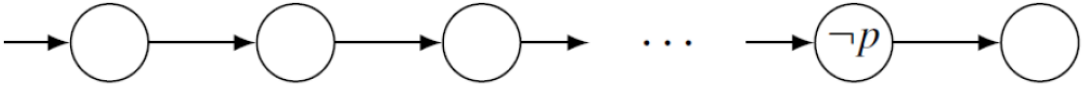
- Simplest form of a counterexample: trace that violates φ

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

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



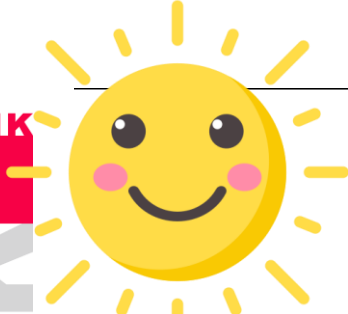
SCOS
Secure & Correct Systems

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?

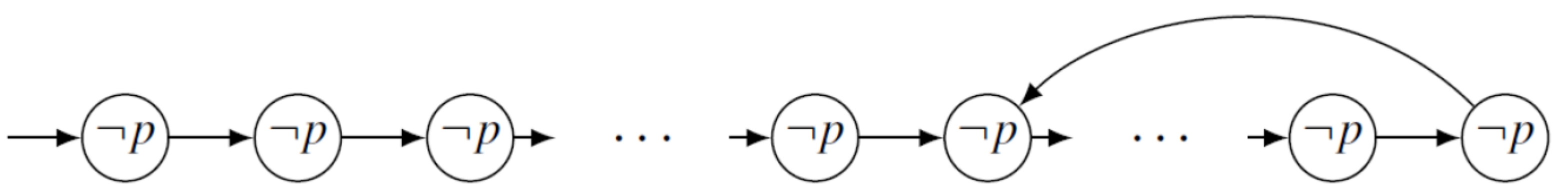


Examples of Counterexamples

- 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

$\mathbf{G}\neg p$



Safety and Liveness Properties

Informally,

- Safety properties guarantee that "something wrong will never happen"
  - Typical example: **AG**p
- Liveness properties guarantee that "something good will eventually happen"
  - Typical examples: **AF**p, **A**(p**U**q)

Safety Properties

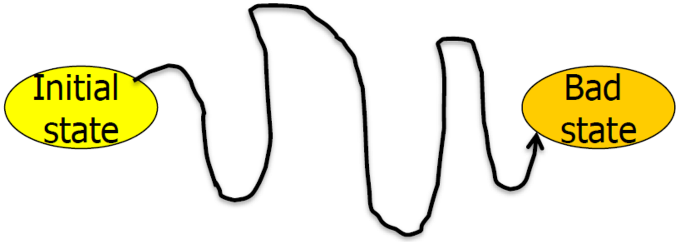
- Nothing “bad” will happen

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

Safety Properties

- Nothing “bad” will happen

- Exercise:
  - A counterexample for a safety property is a **finite (loop-free) path**



Liveness Properties

- Something 'good' will happen.
  - Example: F p

Exercise:

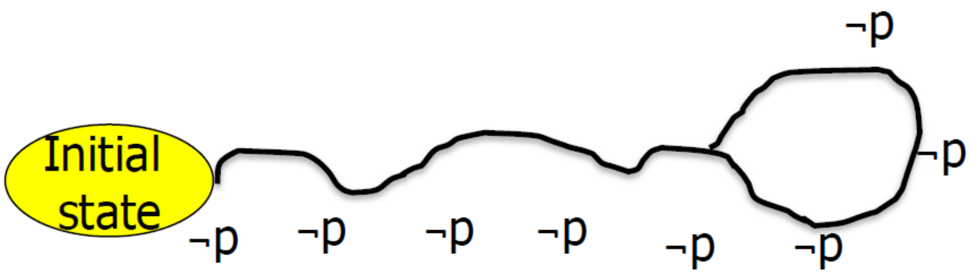
- How does a CE for a **Liveness** property look like?

Liveness Properties

- Something 'good' will happen.
  - Example: F p

Exercise:

- A counterexample is an infinite trace with lasso-shape, showing that this good thing NEVER happened



Part 2 –
CTL Model Checking

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 $[\![f]\!]_M = \{ s \in S \mid M,s \vDash f \}$, i.e., all states satisfying f
  - Check $S_0 \subseteq [\![f]\!]_M$ to conclude that $M \vDash f$

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;

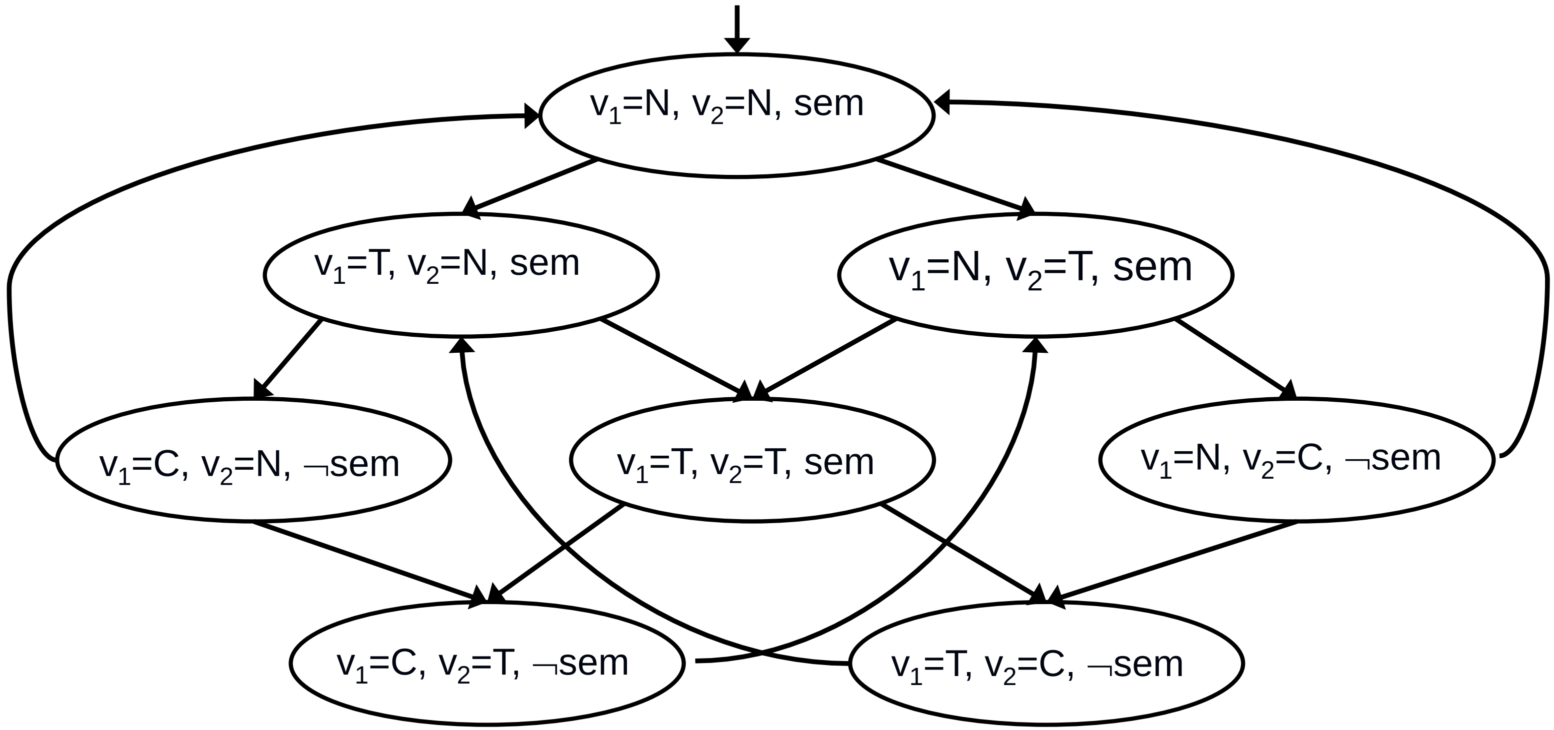
else if (v_i == T && sem) { v_i = C; sem = 0; }

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

}

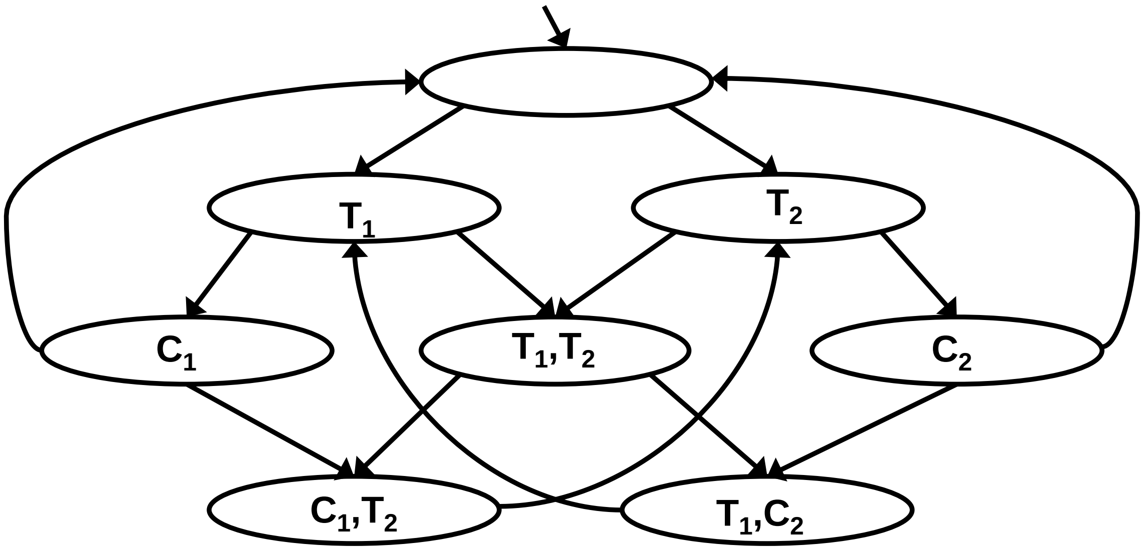
- The full program is: $P_1||P_2$
- Initial state: (v_1=N, v_2=N, sem)
- The execution is interleaving

Illustrative Example: Mutual Exclusion



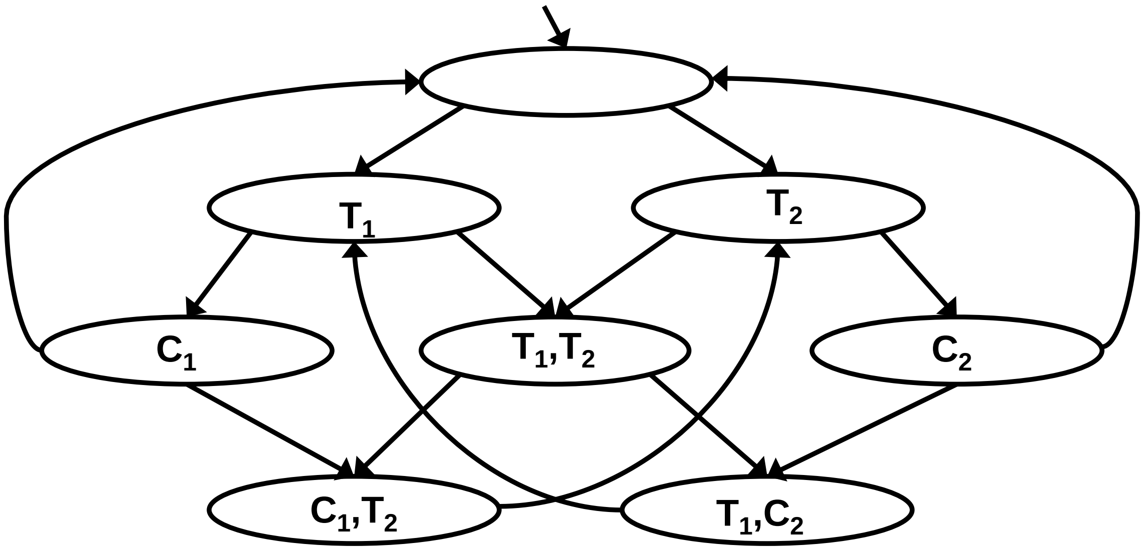
- We define atomic propositions: $AP=\{C_1,C_2,T_1,T_2)$
- A state is labeled with T_i if $v_i=T$
- A state is labeled with C_i if $v_i=C$

Illustrative Example: Mutual Exclusion



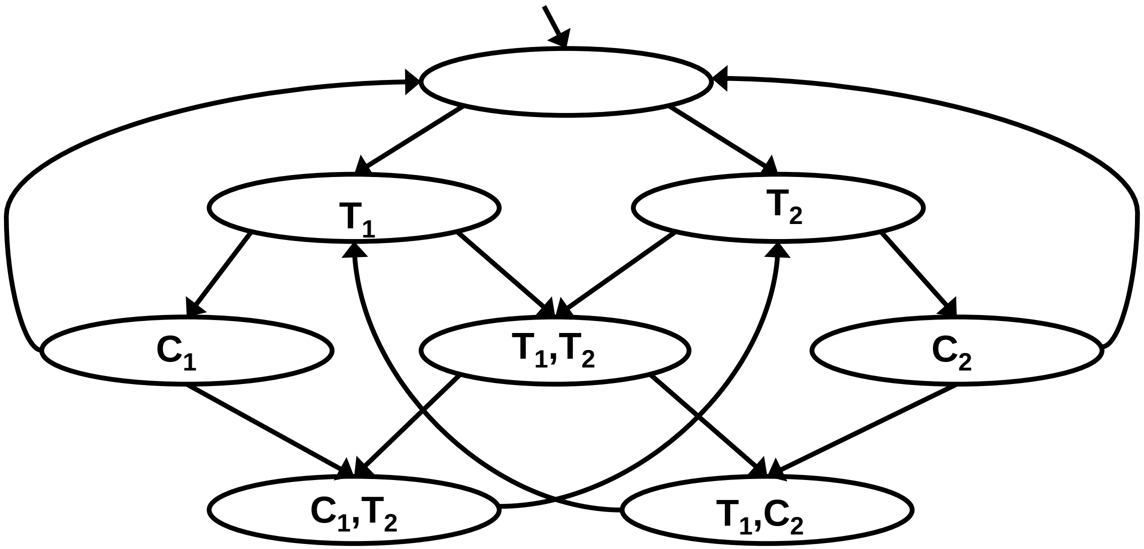
- We define atomic propositions: $AP=\{C_1,C_2,T_1,T_2)$
- A state is labeled with T_i if v_i=T
- A state is labeled with C_i if v_i=C

Illustrative Example: Mutual Exclusion



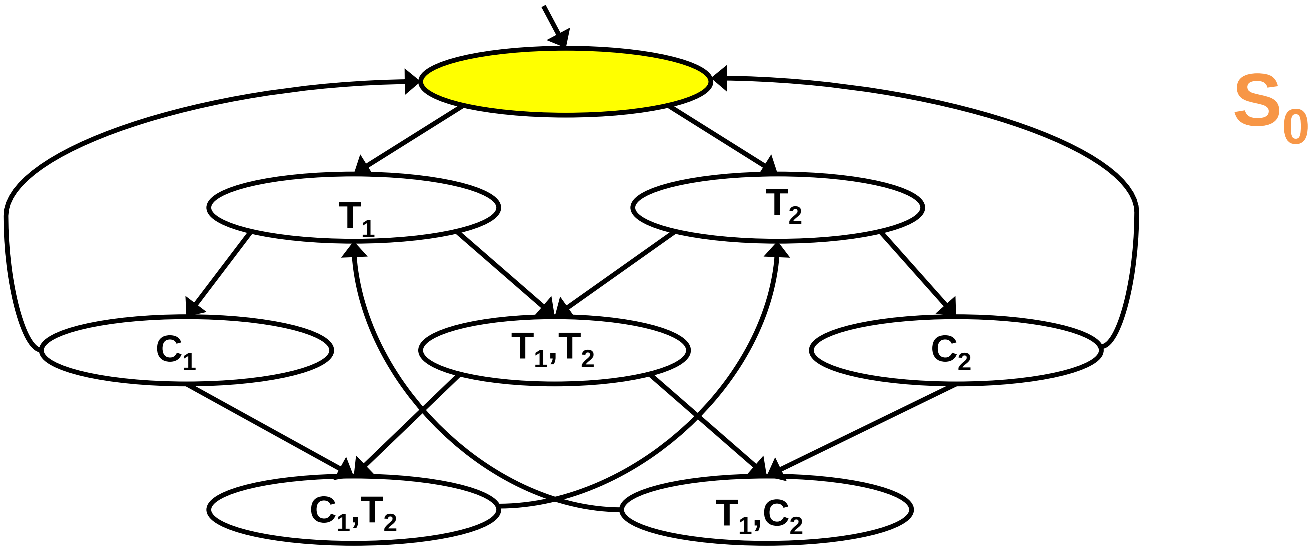
- Does it hold that $M \models f$?
  - Property 1: $f := \mathbf{AG}\neg(C_1 \wedge C_2)$
  - Compute $[\![f]\!]_M = \{\, s \in S \mid M,s \models f \,\}$ and check $S_0 \subseteq [\![f]\!]_M$

Illustrative Example: Mutual Exclusion



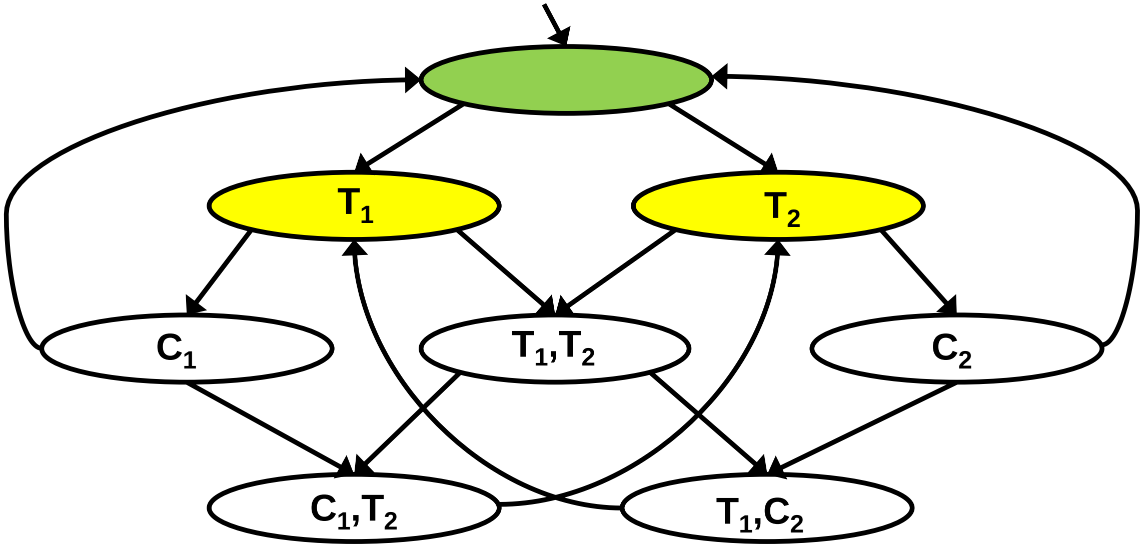
- Does it hold that $M \models f$?
  - Property 1: $f := \mathbf{AG}\neg(C_1 \wedge C_2)$
- $S_i \equiv$ reachable states from an initial state after **i** steps

Illustrative Example: Mutual Exclusion



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

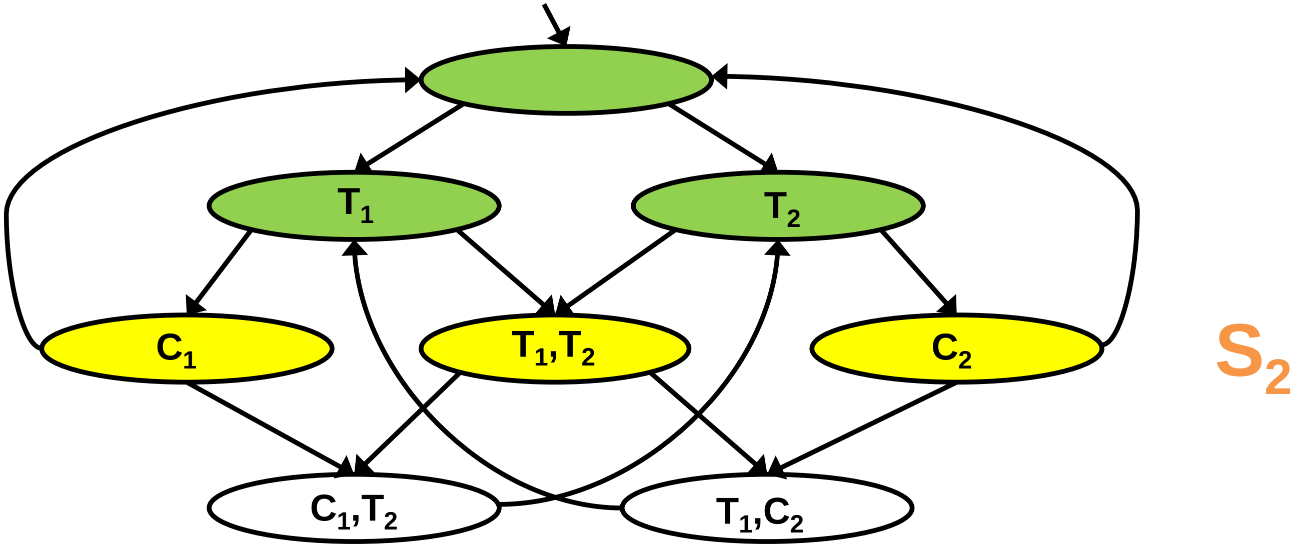
Illustrative Example: Mutual Exclusion



S_1

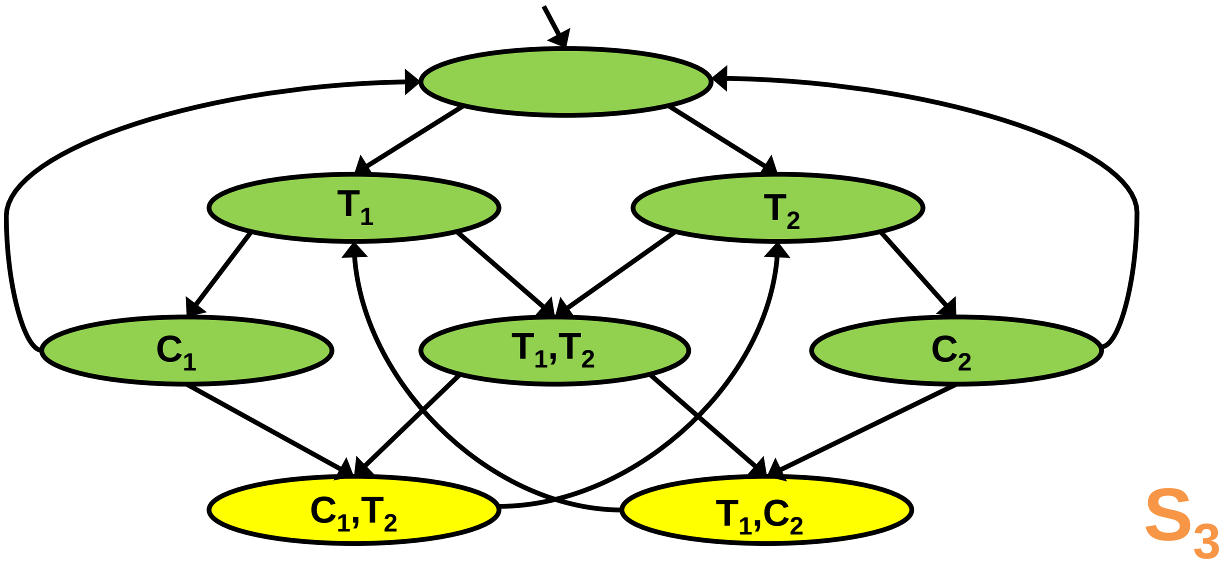
- Does it hold that $M \models f$?
  - Property 1: $f := \mathbf{AG}\neg(C_1 \wedge C_2)$
- $S_i \equiv$ reachable states from an initial state after **i** steps

Illustrative Example: Mutual Exclusion



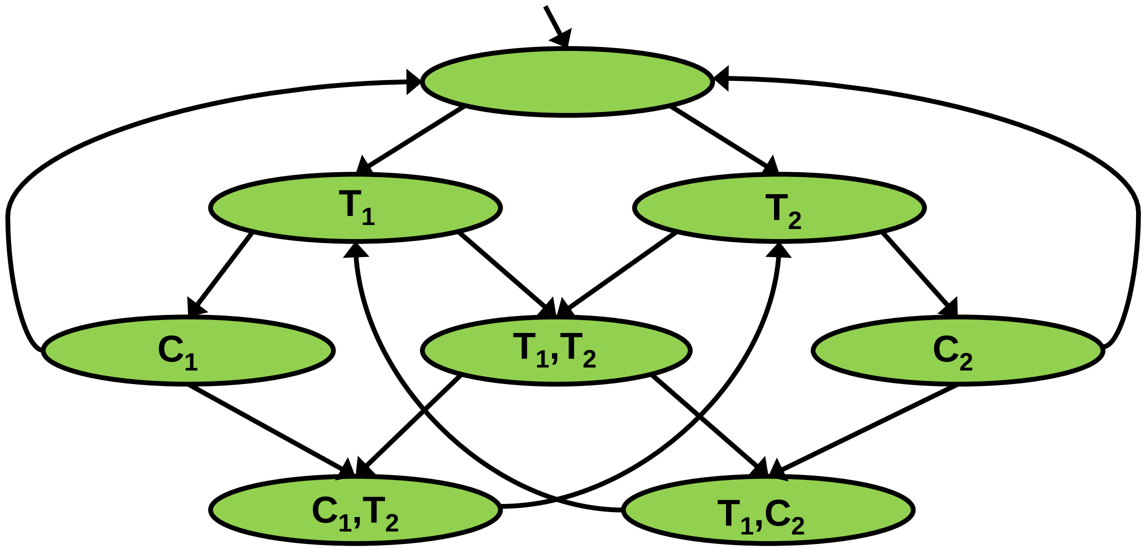
- Does it hold that $M \models f$?
  - Property 1: $f := \mathbf{AG}\neg(C_1 \wedge C_2)$
- $S_i \equiv$ reachable states from an initial state after **i** steps

Illustrative Example: Mutual Exclusion



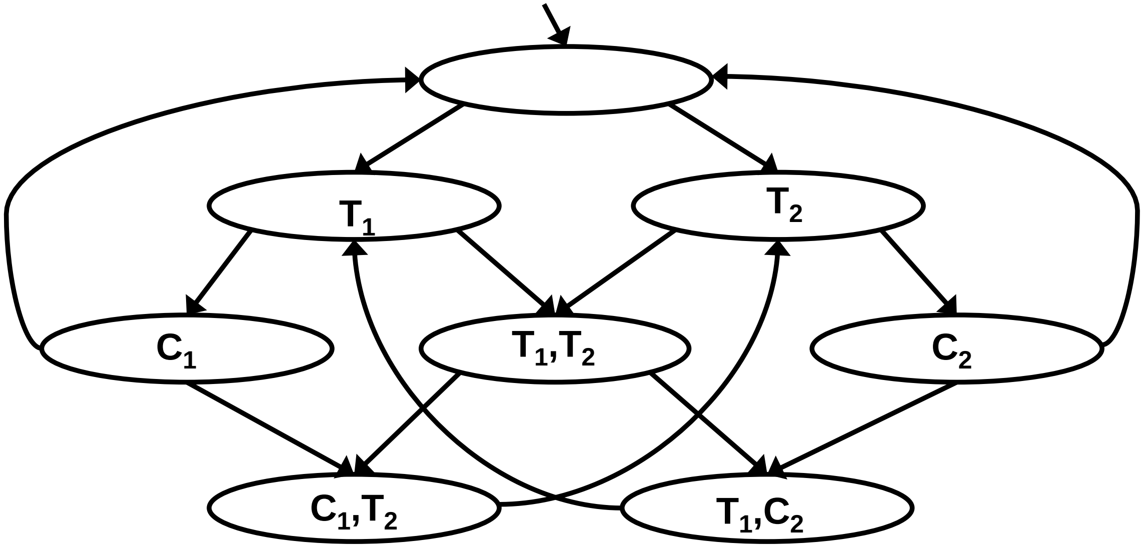
- Does it hold that M ⊨ f?
  - Property 1: f := **AG**$\neg(C_1 \wedge C_2)$
- S_i ≡ reachable states from an initial state after **i** steps

Illustrative Example: Mutual Exclusion



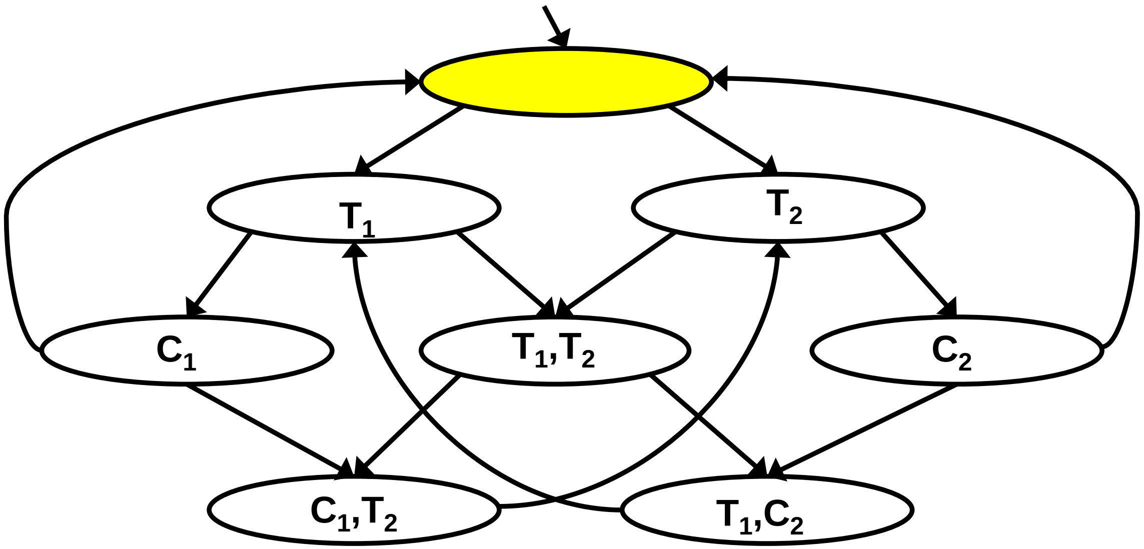
- Does it hold that $M \vDash f$?
  - Property 1: $f := \mathbf{AG}\neg(C_1 \wedge C_2)$ ✓ $M \vDash AG\ \neg\ (C_1 \wedge C_2)$

Illustrative Example: Mutual Exclusion



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

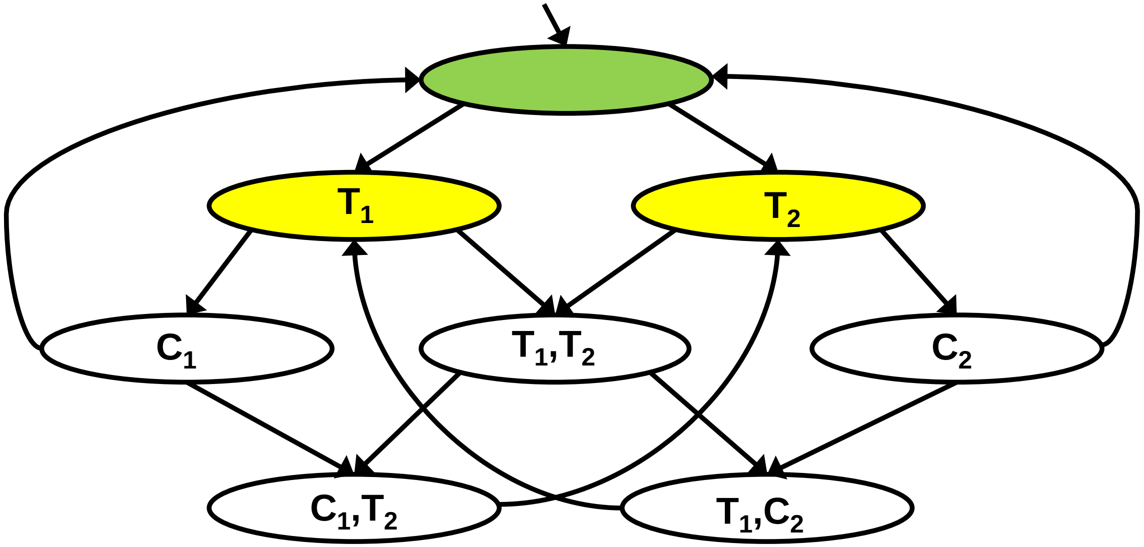
Illustrative Example: Mutual Exclusion



S_0

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

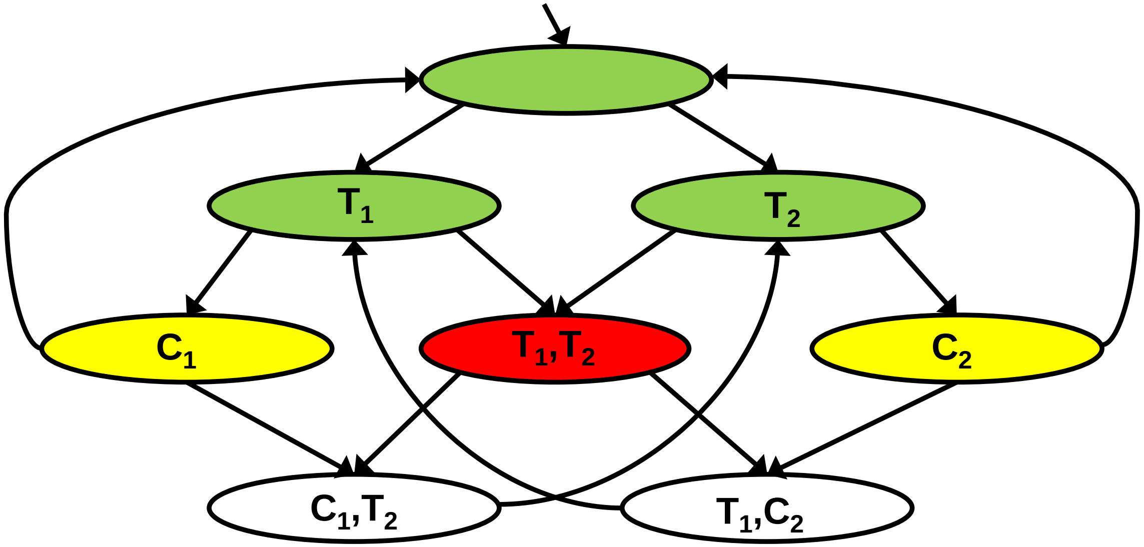
Illustrative Example: Mutual Exclusion



S_1

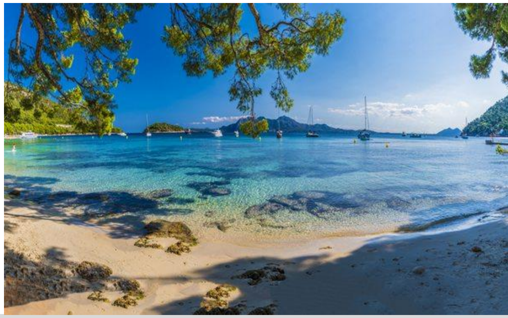
- Does it hold that $M \models f$?
  - Property 2: $f := \mathbf{AG}\neg(T_1 \wedge T_2)$
- $S_i \equiv$ reachable states from an initial state after **i** steps

Illustrative Example: Mutual Exclusion



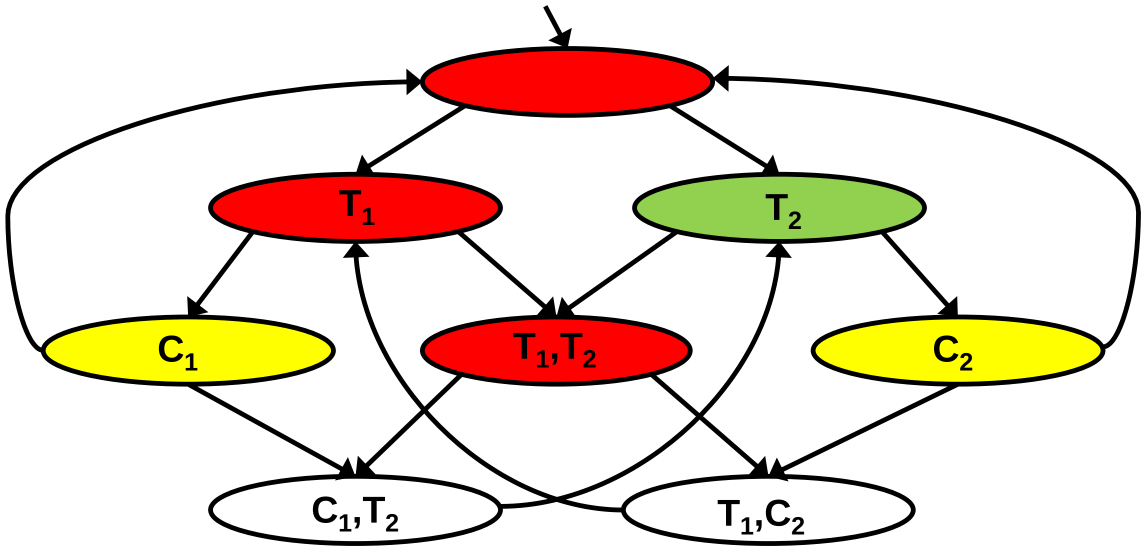
S_3

- Does it hold that $M \models f$?
  - Property 2: $f := \mathbf{AG}\neg(T_1 \wedge T_2)$ ✗ $M \not\models AG \neg (T_1 \wedge T_2)$



SCOS
Secure & Correct Systems

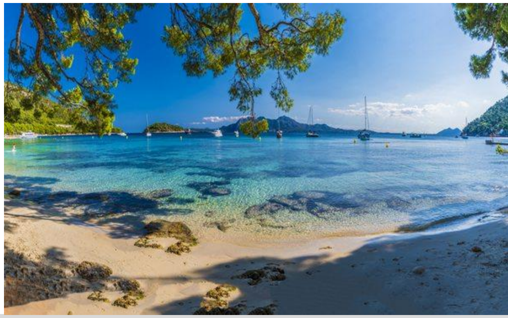
Illustrative Example: Mutual Exclusion



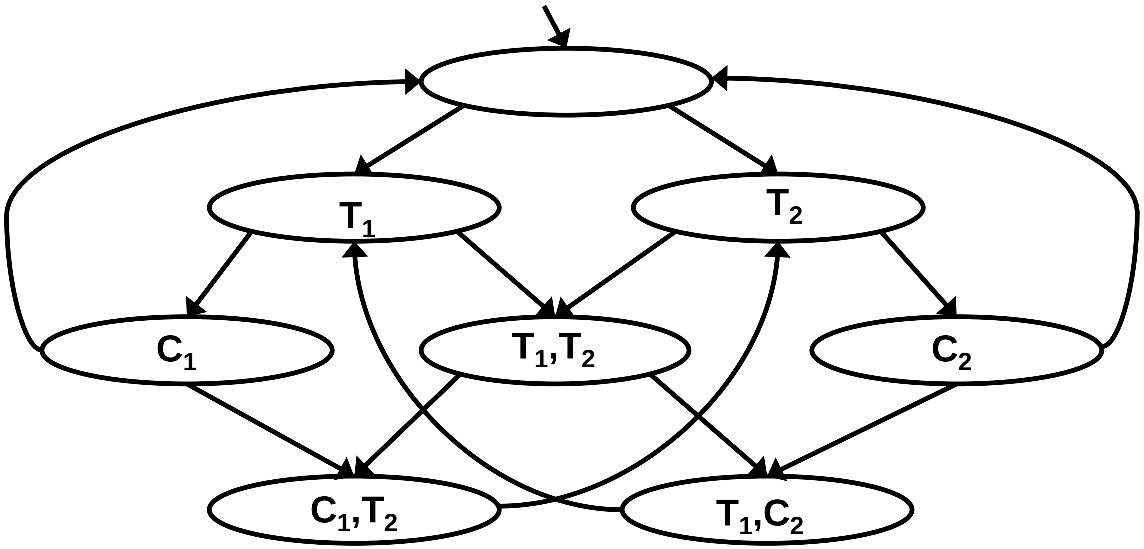
S_3

- Does it hold that $M \models f$?
  - Property 2: $f := \mathbf{AG}\neg(T_1 \wedge T_2)$ ✗ $M \not\models AG \neg (T_1 \wedge T_2)$

- Model checker returns a **counterexample**

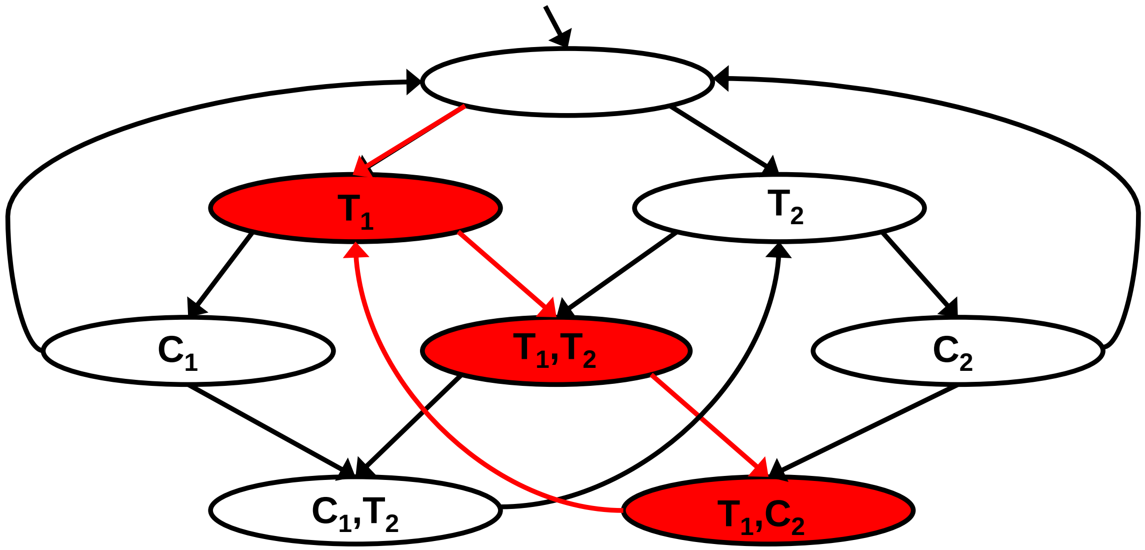


Illustrative Example: Mutual Exclusion



- Does it hold that $M \models f$?
  - Property 3: $f := \textbf{AG}\ ((T_1 \rightarrow \textbf{F}\ C_1) \wedge (T_2 \rightarrow \textbf{F}\ C_2))$
- In case $M \not\models f$, compute a counterexample

Illustrative Example: Mutual Exclusion

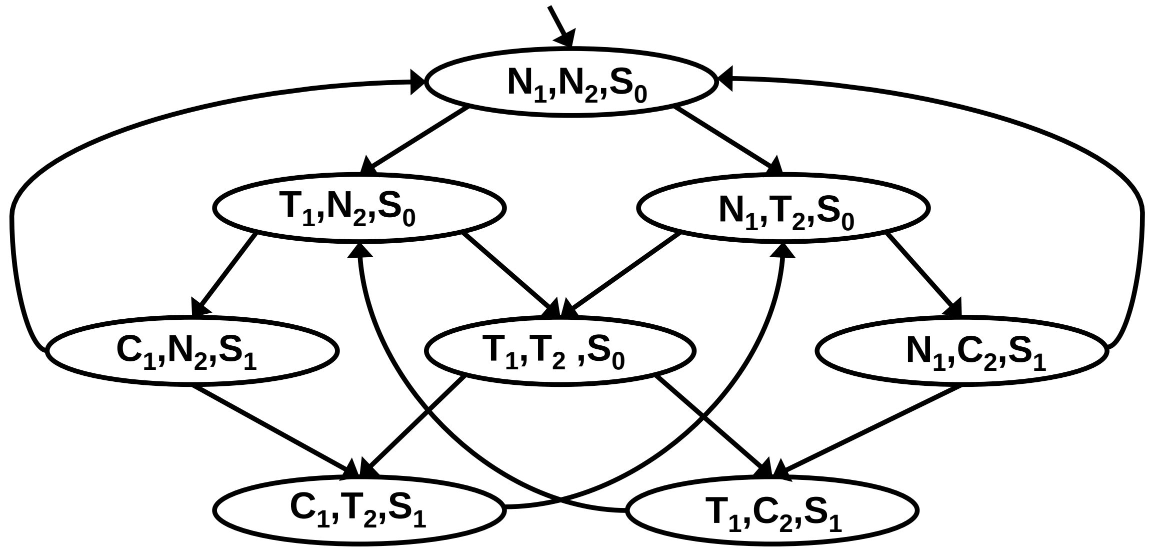


- Does it hold that $M \models f$?
  - Property 3: $f := \mathbf{AG}\,((T_1 \rightarrow \mathbf{F}\,C_1) \wedge (T_2 \rightarrow \mathbf{F}\,C_2))$
- In case $M \not\models f$, compute a counterexample

$M \not\models AG\,((T_1 \rightarrow F\,C_1) \wedge (T_2 \rightarrow F\,C_2))$

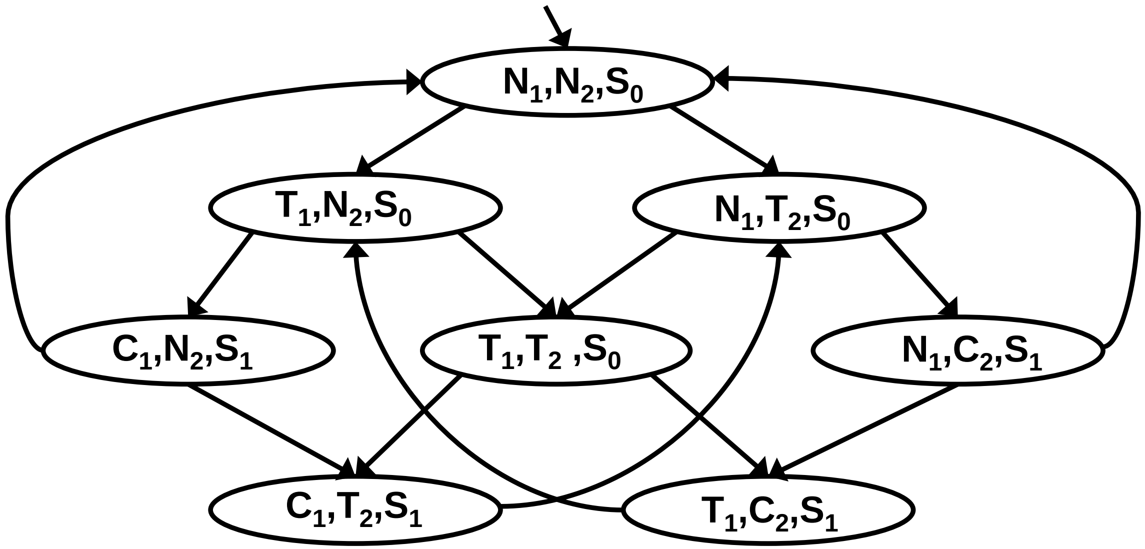
SCOS Secure & Correct Systems

Illustrative Example: Mutual Exclusion



- Does it hold that $M \models f$?
  - Property 4: $f := \mathbf{AG}\,\mathbf{EF}\,(N_1 \wedge N_2 \wedge S_0)$
- How would you express property 4 in natural language?
- In case $M \not\models f$, compute a counterexample

Illustrative Example: Mutual Exclusion



- Does it hold that $M \models f$? ✓
  - Property 4: $f := \mathbf{AG}\,\mathbf{EF}\,(N_1 \wedge N_2 \wedge S_0)$
- ***No matter where you are***
  there is always a way
  to get to the initial state (restart)

CTL Model Checking

Receives:

- A Kripke structure M, modeling a system
- A CTL formula f, describing a property

- Determines whether $M \models f$

- Alternatively definition, MC returns $[\![f]\!] = \{ s \in S \mid M,s \models f \}$
  - M is omitted from $[\![f]\!]_M$ when clear from the context

SCOS
Secure & Correct Systems

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 → **AF** grant**)**
  - Check grant, request
  - Then check **AF** grant
  - Next check request → **AF** grant
  - Finally check **AG(** request → **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
- We check subformula g of f only after all subformulas of g have already been checked
- For subformula g, the algorithm adds g to label(s) for every state s that satisfies g
- When we finish checking g, the following holds:
  - $g \in label(s) \Leftrightarrow M,s \models g$

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

- $M \models f$ if and only if $f \in label(s)$ for all initial states s of M
  - $M \models f$ if and only if $S_0 \subseteq [\![f]\!]_M$

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 Atomic Propositions

- Procedure for labeling the states satisfying $p \in AP$:

$$p \in label(s) \Leftrightarrow \underbrace{p \in L(s)}_{\text{Defined by M}}$$

Model Checking ¬, ∨- Formulas

- Let f_1 and f_2 be subformulas that have already been checked
  - added to label(s), when needed

ToDo Give the procedures for labeling the states satisfying $\neg f_1$ and $f_1 \vee f_2$

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

- Let f_1 and f_2 be subformulas that have already been checked
  - added to label(s), when needed
- Give the procedures for labeling the states satisfying $\neg f_1$ and $f_1 \vee f_2$
  - $\neg f_1$ 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 $g = EX\, f_1$

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

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 $g = EX\, f_1$

- 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)

```
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 $g = E(f_1 U f_2)$

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
```

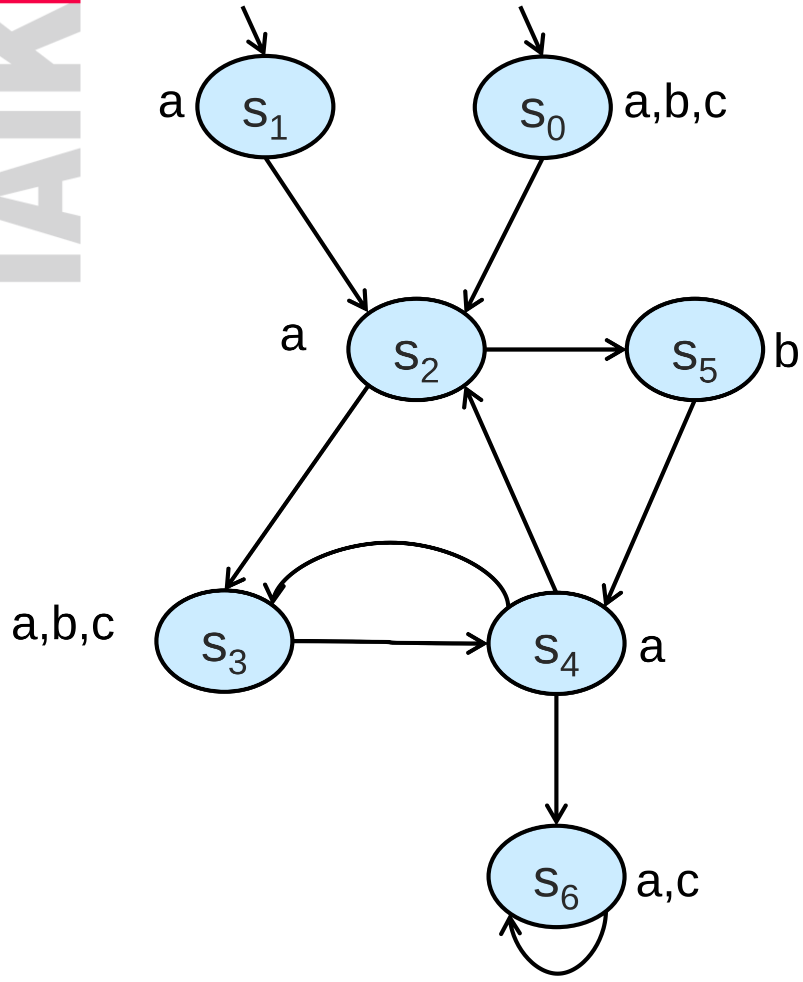
Model Checking $g = E(f_1 U f_2)$

- 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) := 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



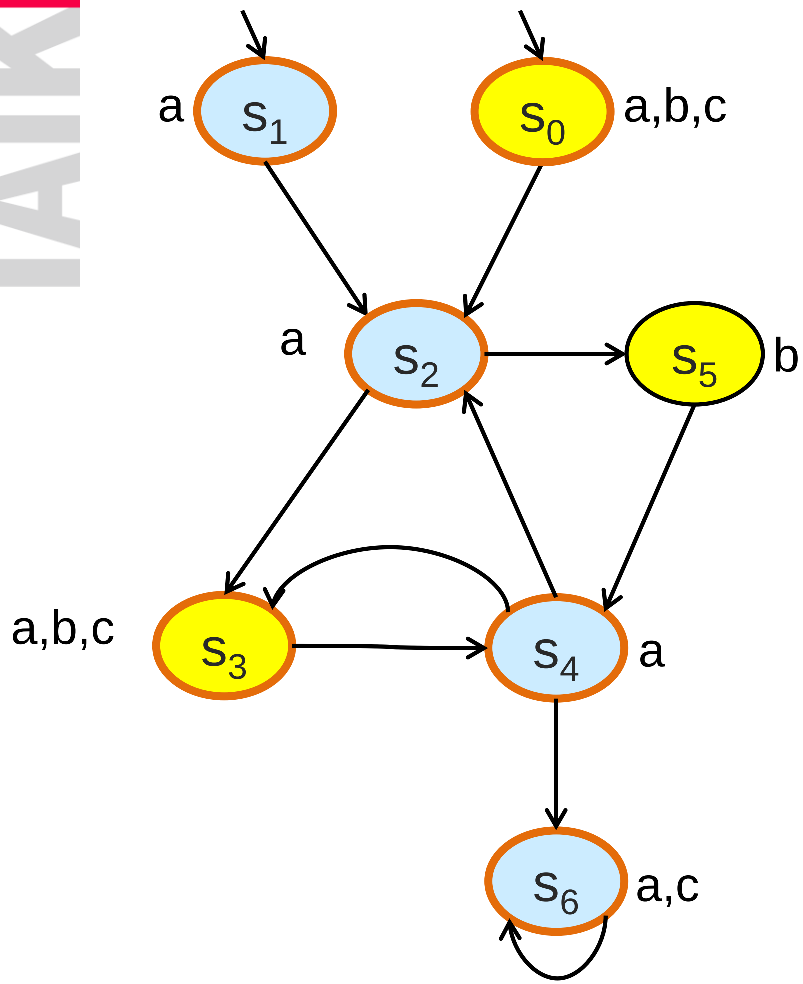
ToDo Does it hold that M ⊨ f?

- $f := E(aUb)$

```
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}
```


SCOS Secure & Correct Systems

Example: Model Checking U Formulas

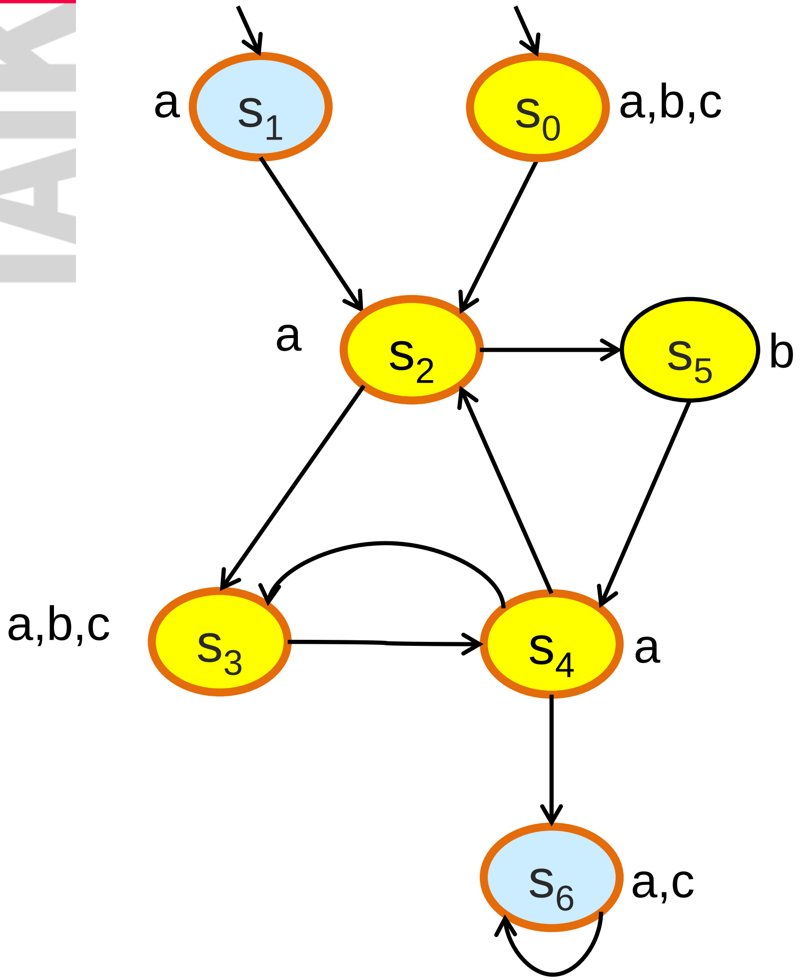


Does it hold that M ⊨ f?

- $f := E(aUb)$

```
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

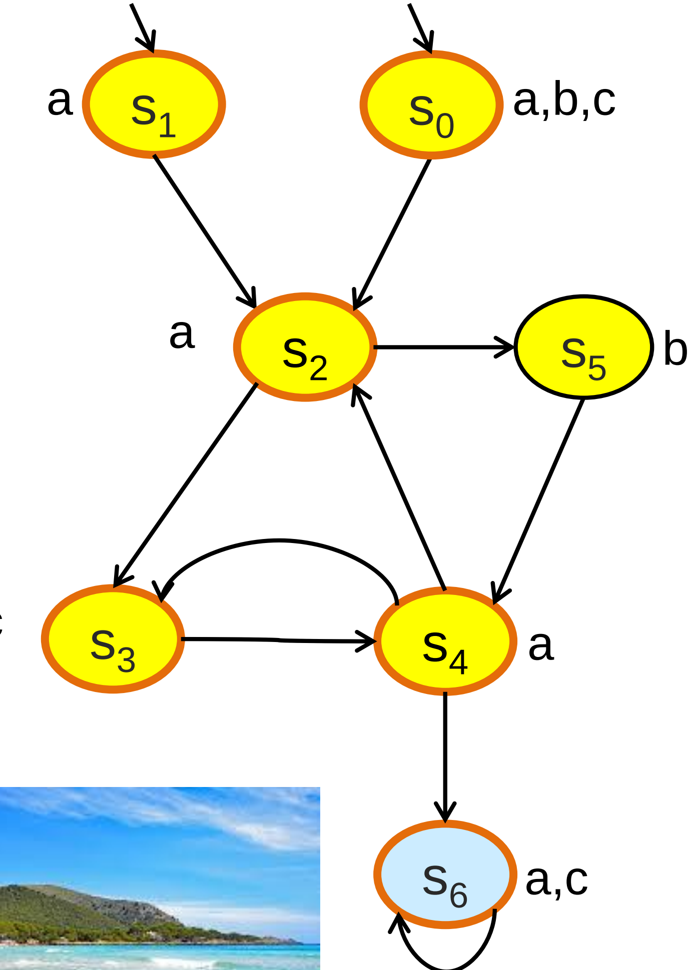


Does it hold that M ⊨ f?

- $f := E(aUb)$

```
procedure CheckEU (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)]] = {0,3,5,4}

```
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 $g = EG f_1$

Observation:

s ⊨ **EG** f_1

iff

There is a path π, starting at s, such that π ⊨ **G** f_1

iff

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

- 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 $g = EG f_1$

- Reduced structure for M and f_1:
  - Remove from M all states such that $f_1 \notin$ label(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 $s' \in S'$

- Theorem: $M,s \vDash EG\, f_1$ iff

1. $s \in S'$ and
2. There is $a\ path$ in M' from s to some state t in a nontrivial MSCC of M'

Model Checking $g = EG f_1$

```
procedure CheckEG (f1)
  S′ := {s | f1 ∈ label(s) }
  MSCC := { C | C is a nontrivial MSCC of M′ }
  T := ∪C ∈MSCC { s | s ∈ C}

  for all t∈T do
      label(t) := label(t) ∪ { EG f1}

  while T ≠ ∅  do
      choose t ∈T;  T := T \ {t};
      for all s ∈S’  such that  R’(s,t) do
          if EG f1 ∉ label(s) then
              label(s) : = label(s) ∪ {EG f1};
              T : = T ∪ {s}
```

Model Checking Complexity

Steps per Subformula

- MC Atomic Propositions
  -
- MC $\neg$, $\vee$ 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
  - 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 $g = EG f_1$
  - 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

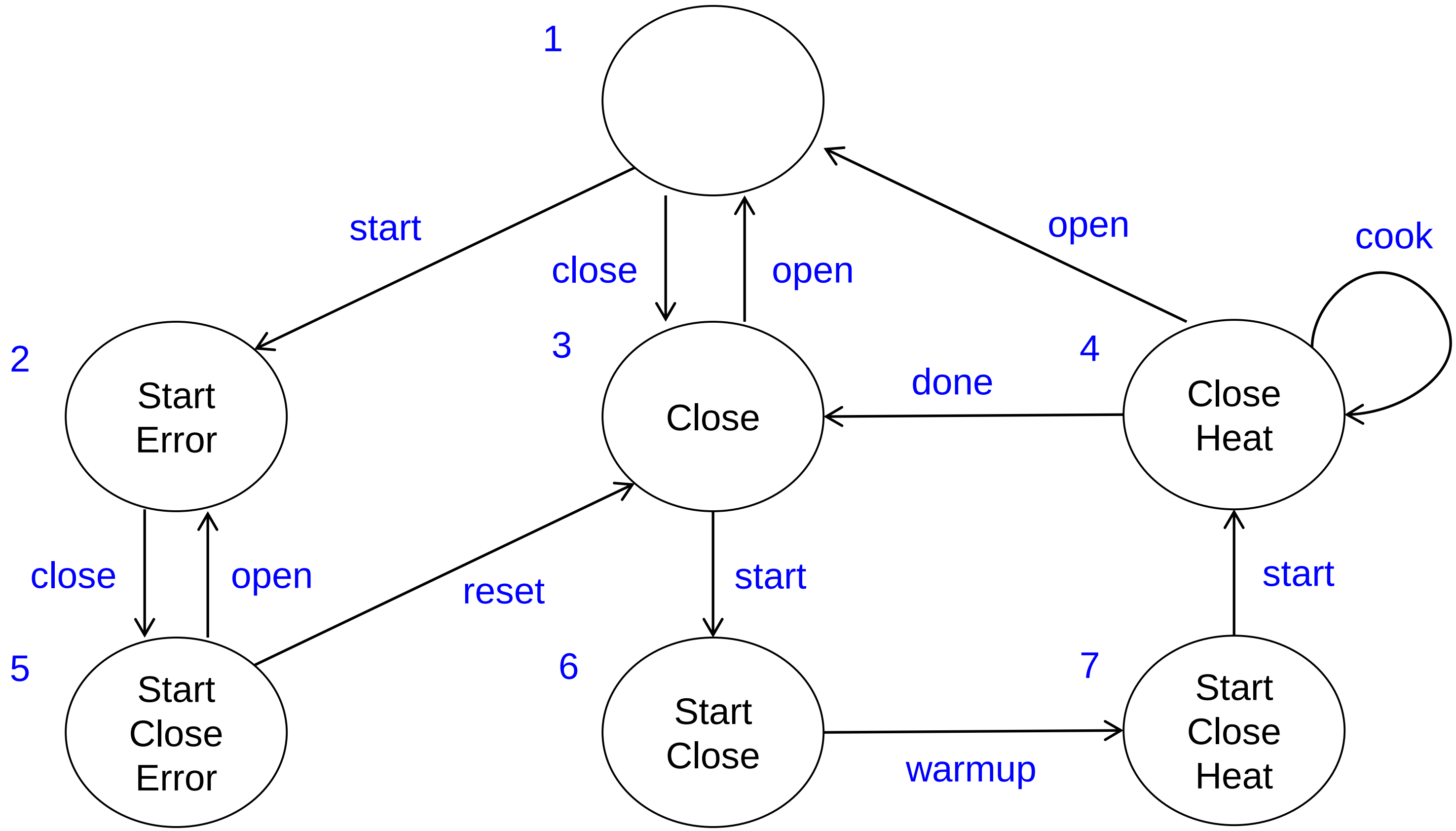
- MC Atomic Propositions
  - O(|S|) steps
- MC $\neg$, $\vee$ formulas
  - O(|S|) steps
- MC $g = EX\ f_1$
  - Add g to label(s) iff s has a successor t such that $f_1 \in$ label(t)
  - O(|S| + |R|)
- MC $g = E(f_1 U f_2)$
  - O(|S| + |R|)
- MC $g = EG f_1$
  - O(|S| + |R|)

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 $M \models f$?
  - $f := AG\,(Start \rightarrow AF\ Heat)$

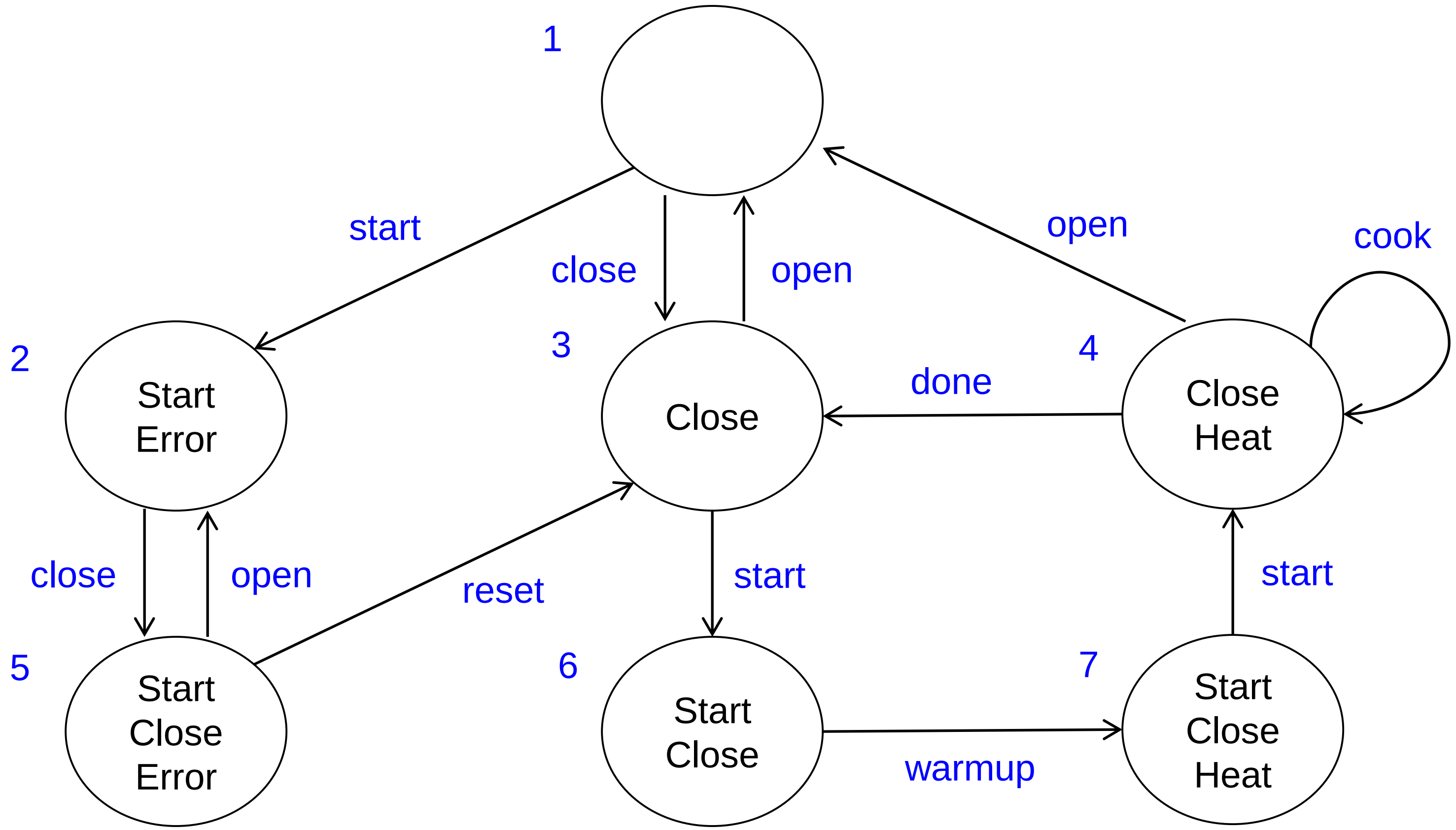


Microwave Example

- Step 1: Rewrite the formula
  - AG (Start $\rightarrow$ AF Heat) $\equiv$
  - $\neg$EF (Start $\wedge$ EG $\neg$Heat) $\equiv$
  - $\neg$E (true U (Start $\wedge$ EG $\neg$Heat))

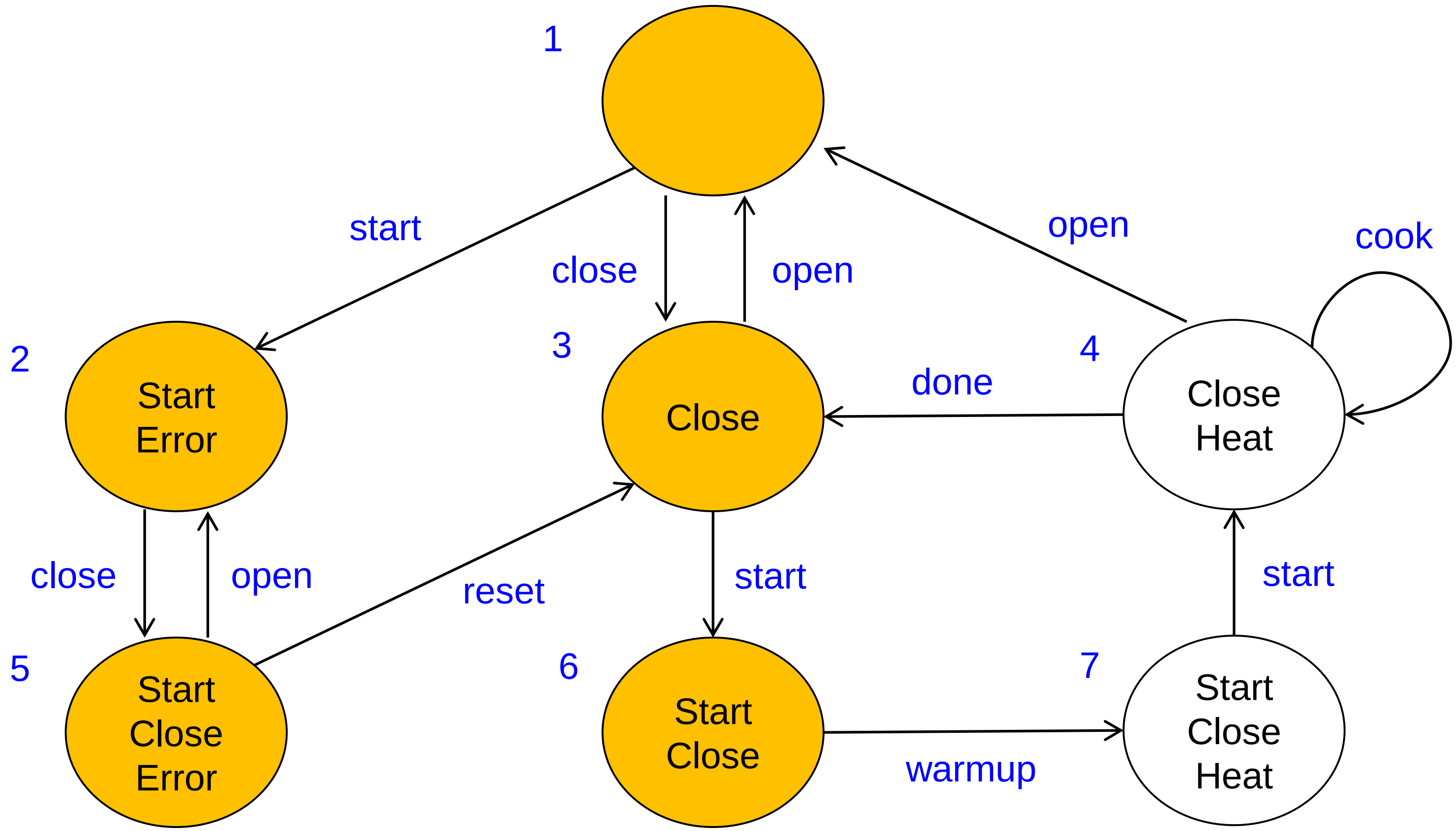
Microwave Example

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



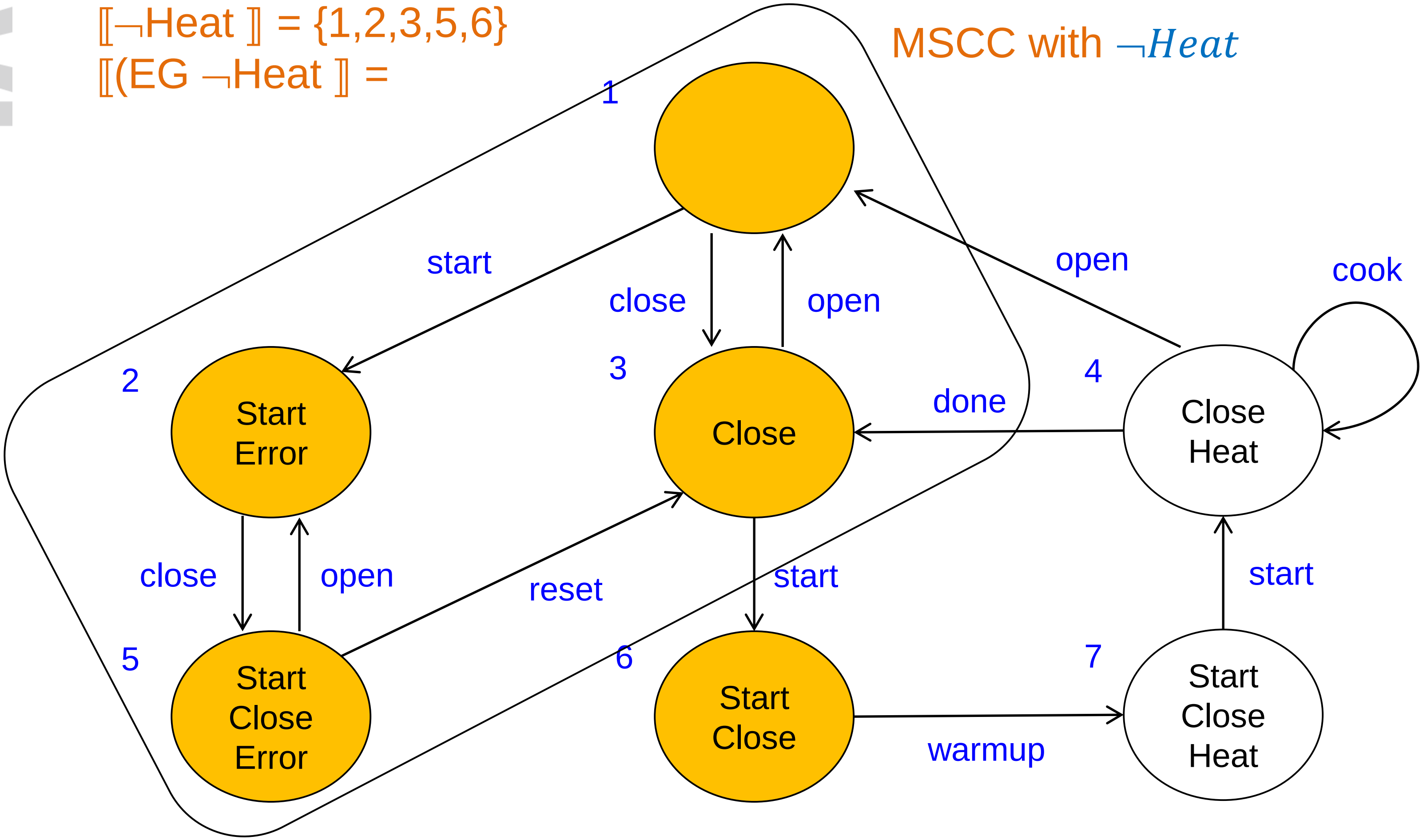
$$f := \neg E\,(true\ U\ (Start \wedge EG\ \neg Heat))$$

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



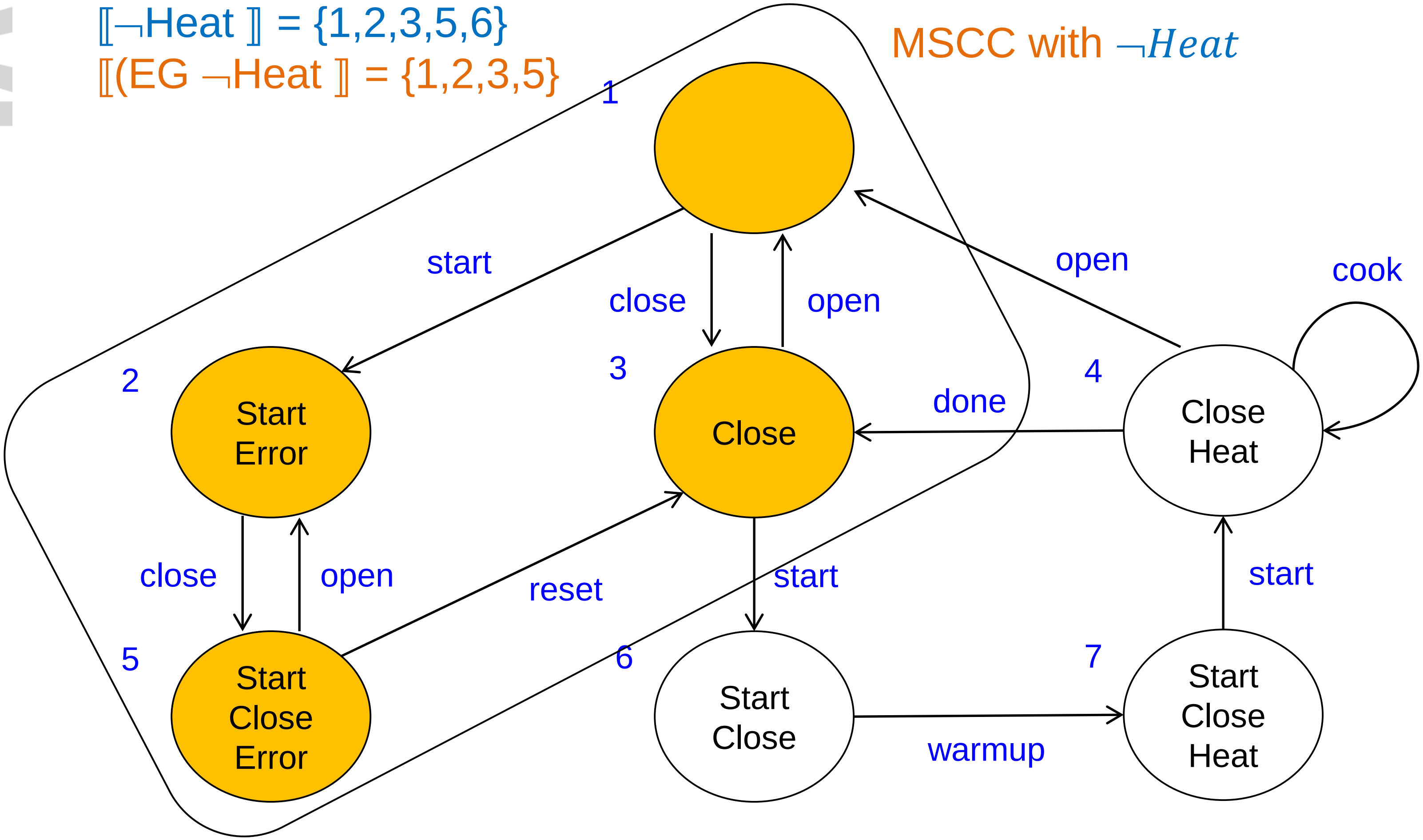
$$f := \neg E\ (true\ U\ (Start \wedge EG\ \neg Heat))$$

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



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

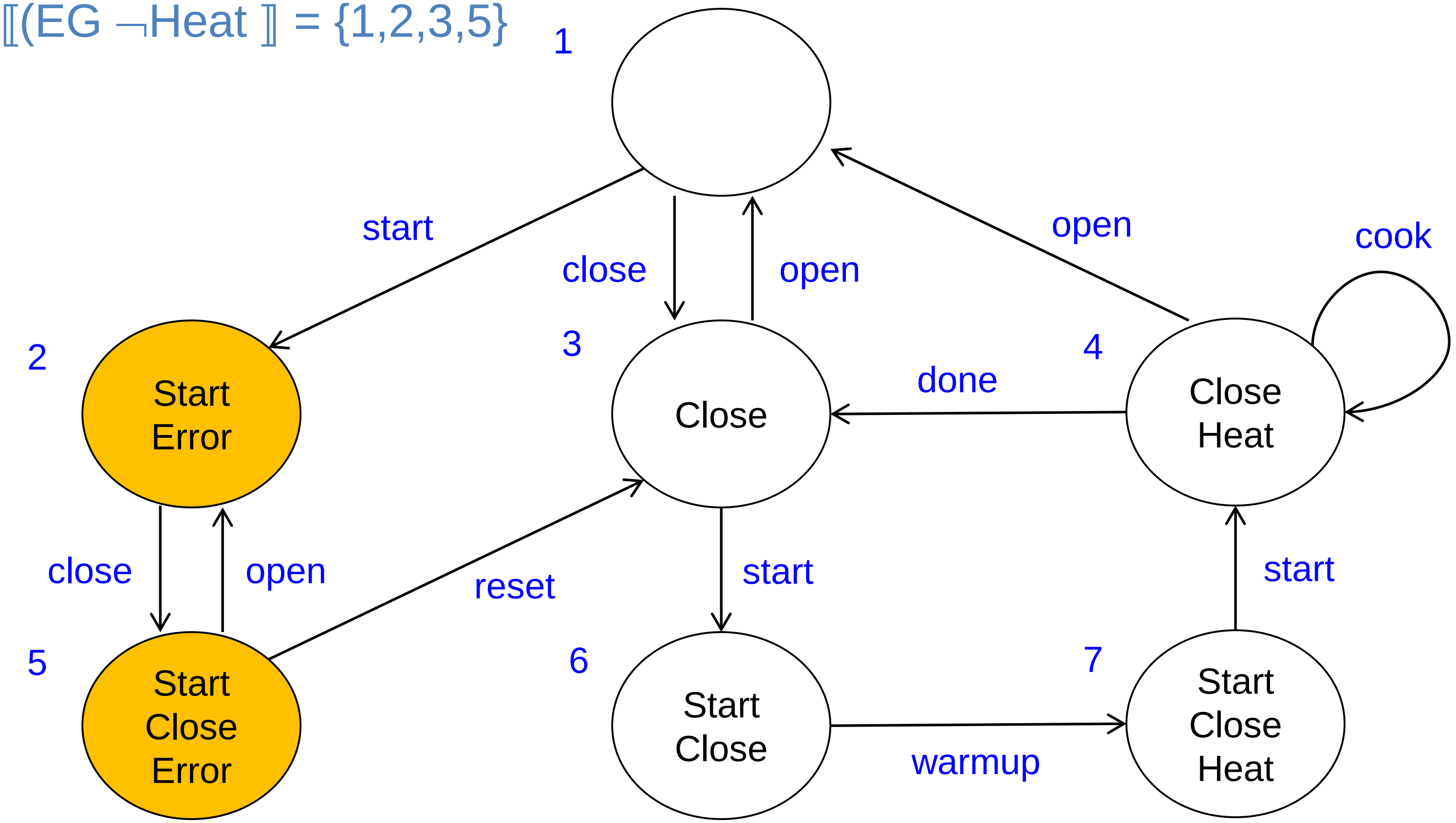
⟦start⟧ = {2,5,6,7}
⟦¬Heat ⟧ = {1,2,3,5,6}
⟦(EG ¬Heat ⟧ = {1,2,3,5}



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

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

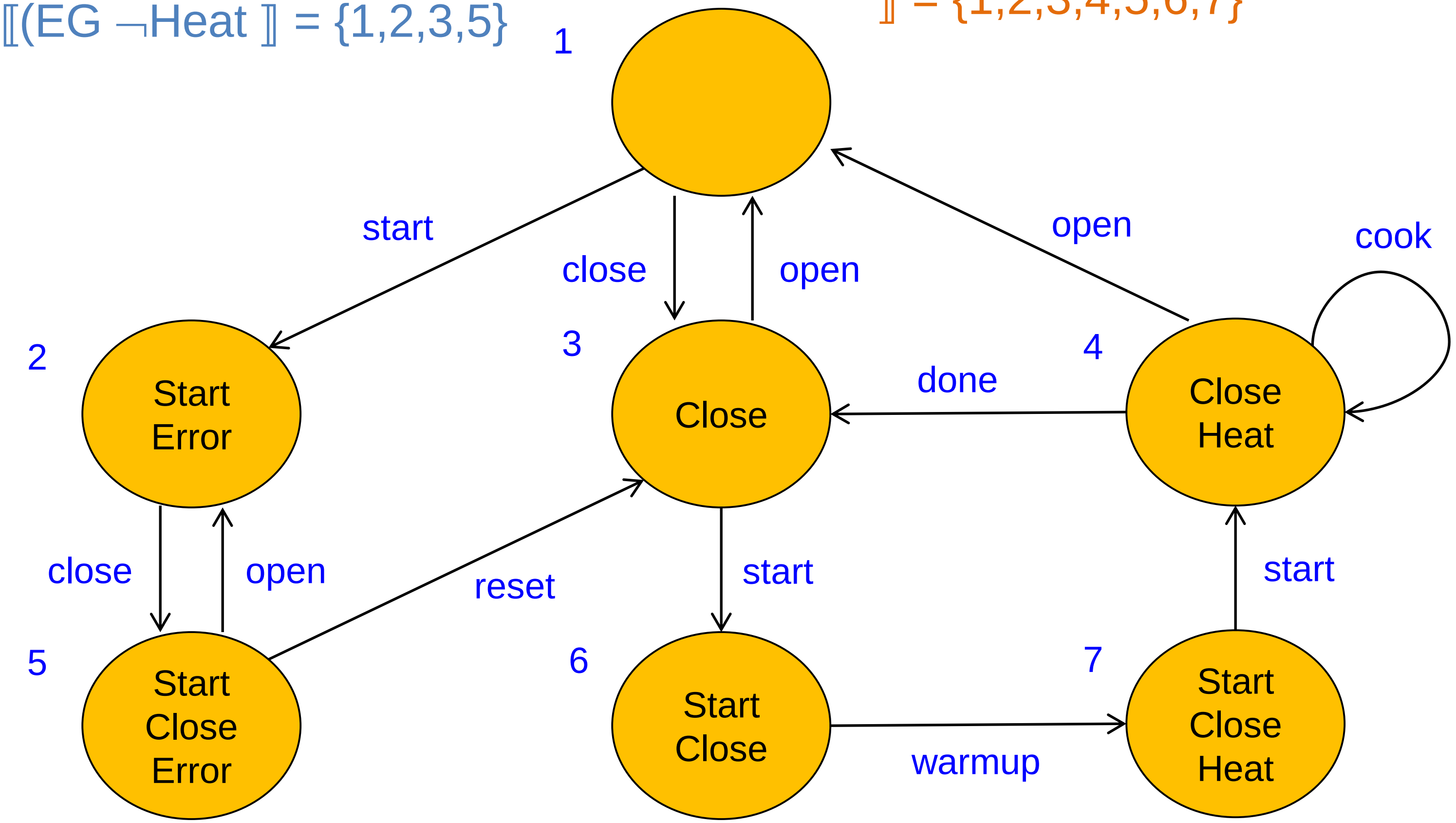
⟦ Start ∧ EG ¬Heat ⟧ = {2, 5}



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

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

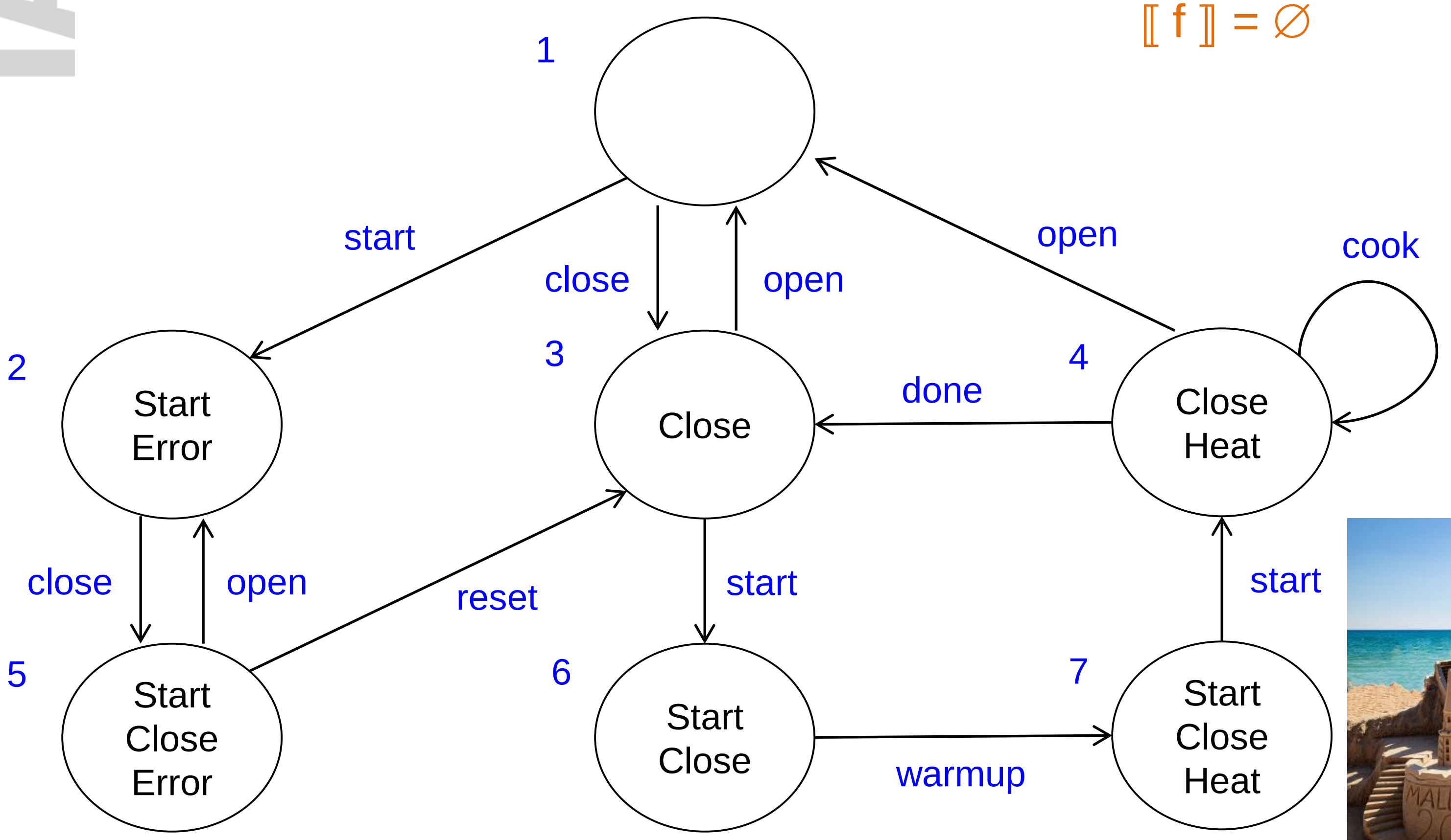
⟦ Start ∧ EG ¬Heat ⟧ = {2, 5}
⟦ E (true U (Start ∧ EG ¬Heat)) ⟧ = {1,2,3,4,5,6,7}



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

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

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



SCOS
Secure & Correct Systems


THANK
YOU!