

# Formal Methods

## Module II: Formal Verification

### Ch. 05: **Explicit-State CTL Model Checking**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it

URL: <http://disi.unitn.it/rseba/DIDATTICA/fm2022/>

Teaching assistant: **Giuseppe Spallitta** – giuseppe.spallitta@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems  
Academic year 2021-2022

last update: Wednesday 20<sup>th</sup> April, 2022, 18:36

*Copyright notice: some material (text, figures) displayed in these slides is courtesy of R. Alur, M. Benerecetti, A. Cimatti, M. Di Natale, P. Pandya, M. Pistore, M. Roveri, C. Tinelli, and S. Tonetta, who retain its copyright. Some examples displayed in these slides are taken from [Clarke, Grunberg & Peled, "Model Checking", MIT Press], and their copyright is retained by the authors. All the other material is copyrighted by Roberto Sebastiani. Every commercial use of this material is strictly forbidden by the copyright laws without the authorization of the authors. No copy of these slides can be displayed in public without containing this copyright notice.*

# Outline

- 1 CTL Model Checking: general ideas
- 2 Some theoretical issues
- 3 CTL Model Checking: algorithms
- 4 CTL Model Checking: some examples
- 5 A relevant subcase: invariants
- 6 Exercises

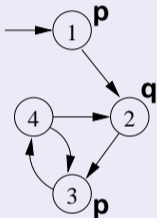
# Outline

- 1 CTL Model Checking: general ideas
- 2 Some theoretical issues
- 3 CTL Model Checking: algorithms
- 4 CTL Model Checking: some examples
- 5 A relevant subcase: invariants
- 6 Exercises

# CTL Model Checking

CTL Model Checking is a formal verification technique where...

- ...the system is represented as a Finite State Machine  $M$ :



- ...the property is expressed a CTL formula  $\varphi$ :

$$\mathbf{AG}(p \rightarrow \mathbf{AF}q)$$

- ...the model checking algorithm checks whether in all initial states of  $M$  all the executions of the model satisfy the formula ( $M \models \varphi$ ).

# CTL Model Checking: General Idea

Two macro-steps:

- 1 construct the set of states where the formula holds:

$$[\varphi] := \{s \in S : M, s \models \varphi\}$$

( $[\varphi]$  is called the **denotation** of  $\varphi$ )

- 2 then compare with the set of initial states:

$$I \subseteq [\varphi] ?$$

## CTL Model Checking: General Idea [cont.]

In order to compute  $[\varphi]$ :

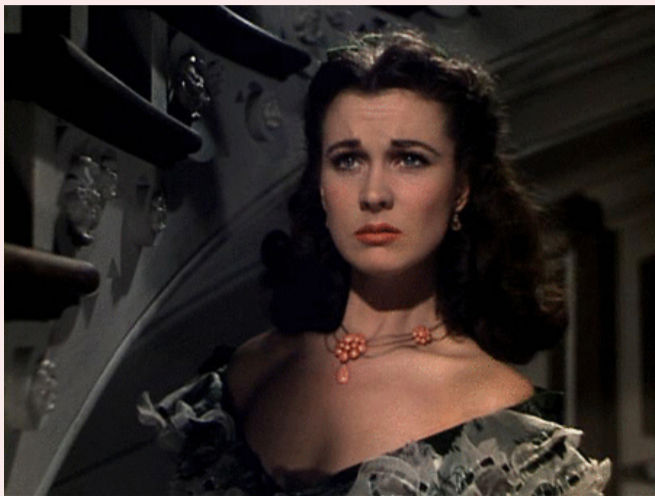
- proceed “bottom-up” on the structure of the formula, computing  $[\varphi_i]$  for each subformula  $\varphi_i$  of  $\mathbf{AG}(p \rightarrow \mathbf{AF}q)$ :
  - $[q]$ ,
  - $[\mathbf{AF}q]$ ,
  - $[p]$ ,
  - $[p \rightarrow \mathbf{AF}q]$ ,
  - $[\mathbf{AG}(p \rightarrow \mathbf{AF}q)]$

## CTL Model Checking: General Idea [cont.]

In order to compute each  $[\varphi_i]$ :

- assign **Propositional atoms** by **labeling function**
- handle **Boolean operators** by standard **set operations**
- handle **temporal operators AX, EX** by computing **pre-images**
- handle **temporal operators AG, EG, AF, EF, AU, EU**, by (implicitly) applying **tableaux rules**, until a **fixpoint** is reached

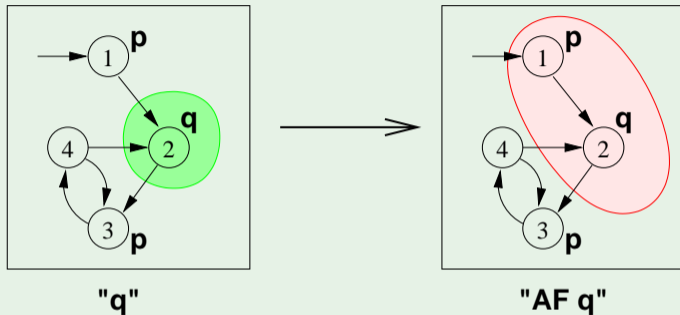
## Tableaux Rules: a Quote



*"After all... tomorrow is another day."  
[Scarlett O'Hara, "Gone with the Wind"]*

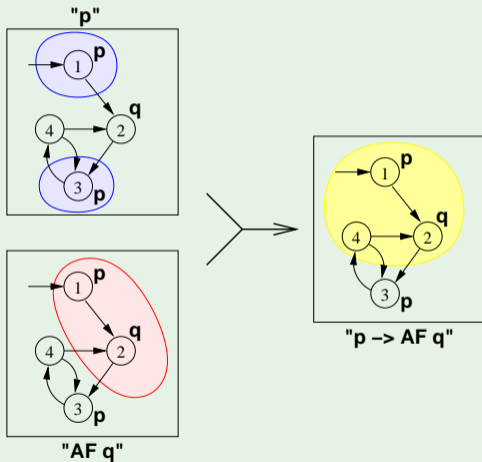


# CTL Model Checking: Example: $\mathbf{AG}(p \rightarrow \mathbf{AF}q)$

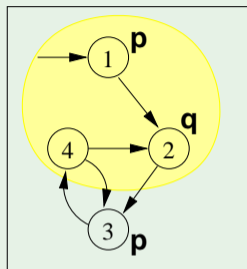


- Recall the **AF** tableau rule:  $\mathbf{AF}q \leftrightarrow (q \vee \mathbf{AXAF}q)$
- Iteration:  $[\mathbf{AF}q]^{(1)} = [q]$ ;  $[\mathbf{AF}q]^{(i+1)} = [q] \cup \mathbf{AX}[\mathbf{AF}q]^{(i)}$ 
  - $[\mathbf{AF}q]^{(1)} = [q] = \{2\}$
  - $[\mathbf{AF}q]^{(2)} = [q \vee \mathbf{AX}q] = \{2\} \cup \{1\} = \{1, 2\}$
  - $[\mathbf{AF}q]^{(3)} = [q \vee \mathbf{AX}(q \vee \mathbf{AX}q)] = \{2\} \cup \{1\} = \{1, 2\}$   
 $\implies$  (fix point reached)

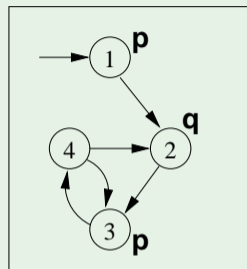
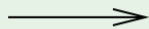
# CTL Model Checking: Example: $\mathbf{AG}(p \rightarrow \mathbf{AF}q)$ [cont.]



# CTL Model Checking: Example: $\mathbf{AG}(p \rightarrow \mathbf{AF}q)$ [cont.]



" $p \rightarrow \mathbf{AF} q$ "



" $\mathbf{AG}(p \rightarrow \mathbf{AF} q)$ "

- Recall the **AG** tableau rule:  $\mathbf{AG}\varphi \leftrightarrow (\varphi \wedge \mathbf{AXAG}\varphi)$
- Iteration:  $[\mathbf{AG}\varphi^{(1)}] = [\varphi]$ ;  $[\mathbf{AG}\varphi^{(i+1)}] = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi^{(i)}]$ 
  - $[\mathbf{AG}\varphi^{(1)}] = [\varphi] = \{1, 2, 4\}$
  - $[\mathbf{AG}\varphi^{(2)}] = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi^{(1)}] = \{1, 2, 4\} \cap \{1, 3\} = \{1\}$
  - $[\mathbf{AG}\varphi^{(3)}] = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi^{(2)}] = \{1, 2, 4\} \cap \{\} = \{\}$   
 $\implies$  (fix point reached)

## CTL Model Checking: Example: $\mathbf{AG}(p \rightarrow \mathbf{AF}q)$ [cont.]

- The set of states where the formula holds is empty  
 $\implies$  the initial state does not satisfy the property  
 $\implies M \not\models \mathbf{AG}(p \rightarrow \mathbf{AF}q)$
- **Counterexample:** a lazo-shaped path:  $1, 2, \{3, 4\}^\omega$  (satisfying  $\mathbf{EF}(p \wedge \mathbf{EG}\neg q)$ )

### Note

Counter-example reconstruction in general is not trivial, based on intermediate sets.

# Outline

- 1 CTL Model Checking: general ideas
- 2 Some theoretical issues**
- 3 CTL Model Checking: algorithms
- 4 CTL Model Checking: some examples
- 5 A relevant subcase: invariants
- 6 Exercises

# The fixed-point theory of lattice of sets

## Definition

Let  $2^S$  denote the power set of  $S$ , i.e., the set of all subsets of  $S$ .

- For any finite set  $S$ , the structure  $\langle 2^S, \subseteq \rangle$  forms a **complete lattice** with  $\cup$  as join and  $\cap$  as meet operations.
- A function  $F : 2^S \mapsto 2^S$  is **monotonic** provided  $S_1 \subseteq S_2 \Rightarrow F(S_1) \subseteq F(S_2)$ .

# Fixed Points

## Definition

Let  $\langle 2^S, \subseteq \rangle$  be a complete lattice,  $S$  finite.

- Given a function  $F : 2^S \rightarrow 2^S$ ,  $a \subseteq S$  is a **fixed point** of  $F$  iff

$$F(a) = a$$

- $a$  is a **least fixed point** (LFP) of  $F$ , written  $\mu x.F(x)$ , iff, for every other fixed point  $a'$  of  $F$ ,  $a \subseteq a'$
- $a$  is a **greatest fixed point** (GFP) of  $F$ , written  $\nu x.F(x)$ , iff, for every other fixed point  $a'$  of  $F$ ,  $a' \subseteq a$

# Iteratively computing fixed points

## Tarski's Theorem

A monotonic function over a complete finite lattice has a least and a greatest fixed point.

## (A corollary of) Kleene's Theorem

A monotonic function  $F$  over a complete finite lattice has a least and a greatest fixed point, which can be computed as follows:

- the least fixed point of  $F$  is the limit of the chain  $\emptyset \subseteq F(\emptyset) \subseteq F(F(\emptyset)) \dots$ ,
- the greatest fixed point of  $F$  is the limit of chain  $S \supseteq F(S) \supseteq F(F(S)) \dots$

Since  $2^S$  is finite, convergence is obtained in a **finite number of steps**.



# CTL Model Checking and Lattices

- If  $M = \langle S, I, R, L, AP \rangle$  is a Kripke structure, then  $\langle 2^S, \subseteq \rangle$  is a complete lattice
- We identify  $\varphi$  with its **denotation**  $[\varphi]$

$\implies$  we can see logical operators as **functions**  $F : 2^S \mapsto 2^S$  on the complete lattice  $\langle 2^S, \subseteq \rangle$

# Denotation of a CTL formula $\varphi$ : $[\varphi]$

## Definition of $[\varphi]$

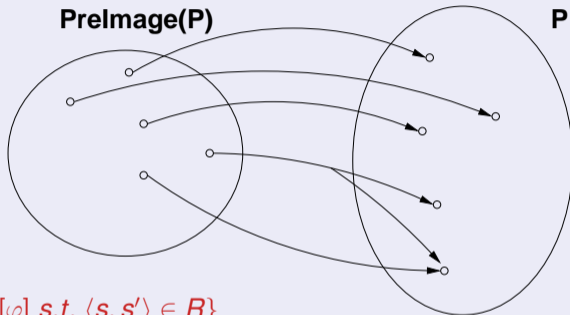
$$[\varphi] := \{s \in S : M, s \models \varphi\}$$

## Recursive definition of $[\varphi]$

$$\begin{aligned} [\top] &= S \\ [\perp] &= \{\} \\ [p] &= \{s \mid p \in L(s)\} \\ [\neg\varphi_1] &= S / [\varphi_1] \\ [\varphi_1 \wedge \varphi_2] &= [\varphi_1] \cap [\varphi_2] \\ [\mathbf{EX}\varphi] &= \{s \mid \exists s' \in [\varphi] \text{ s.t. } \langle s, s' \rangle \in R\} \\ [\mathbf{EG}\beta] &= \nu Z. ( [\beta] \cap [\mathbf{EX}Z] ) \\ [\mathbf{E}(\beta_1 \mathbf{U} \beta_2)] &= \mu Z. ( [\beta_2] \cup ([\beta_1] \cap [\mathbf{EX}Z]) ) \end{aligned}$$

# Case EX

Consider  $\mathbf{EX}\varphi$ :



- $[\mathbf{EX}\varphi] = \{s \mid \exists s' \in [\varphi] \text{ s.t. } \langle s, s' \rangle \in R\}$
- $[\mathbf{EX}\varphi]$  is said to be the **Pre-image** of  $[\varphi]$  ( $Preimage([\varphi])$ )
- Key step of every CTL M.C. operation

## Note

Preimage() is monotonic:  $X \subseteq X' \implies Preimage(X) \subseteq Preimage(X')$

# Case EG

Consider  $\mathbf{EG}\beta$ :

- $\nu Z. ([\beta] \cap [\mathbf{EXZ}])$ : greatest fixed point of the function  $F_\beta : 2^S \mapsto 2^S$ , s.t.

$$\begin{aligned} F_\beta([\varphi]) &= ([\beta] \cap \text{Preimage}([\varphi])) \\ &= ([\beta] \cap \{s \mid \exists s' \in [\varphi] \text{ s.t. } \langle s, s' \rangle \in R\}) \end{aligned}$$

- $F_\beta$  Monotonic:  $a \subseteq a' \implies F_\beta(a) \subseteq F_\beta(a')$ 
  - (Tarski's theorem):  $\nu x. F_\beta(x)$  always exists
  - (Kleene's theorem):  $\nu x. F_\beta(x)$  can be computed as the limit  $S \supseteq F_\beta(S) \supseteq F_\beta(F_\beta(S)) \supseteq \dots$ , in a finite number of steps.

Theorem (Clarke & Emerson)

$$[\mathbf{EG}\beta] = \nu Z. ([\beta] \cap [\mathbf{EXZ}])$$

## Case **EG** [cont.]

- We can compute  $X := [\mathbf{EG}\beta]$  inductively as follows:

$$X_0 := S$$

$$X_1 := F_\beta(S) = [\beta]$$

$$X_2 := F_\beta(F_\beta(S)) = [\beta] \cap \text{Preimage}(X_1)$$

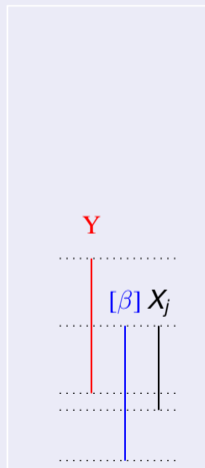
...

$$X_{j+1} := F_\beta^{j+1}(S) = [\beta] \cap \text{Preimage}(X_j)$$

- Noticing that  $X_1 = [\beta]$  and  $X_{j+1} \subseteq X_j$  for every  $j \geq 0$ , and that  $([\beta] \cap Y) \subseteq X_j \subseteq [\beta] \implies ([\beta] \cap Y) = (X_j \cap Y)$ ,

we can use instead the following inductive schema:

- $X_1 := [\beta]$
- $X_{j+1} := X_j \cap \text{Preimage}(X_j)$



# Case EU

Consider  $\mathbf{E}(\beta_1 \mathbf{U} \beta_2)$ :

- $\mu Z. ([\beta_2] \cup ([\beta_1] \cap [\mathbf{EXZ}] ))$ : least fixed point of the function  $F_{\beta_1, \beta_2} : 2^S \mapsto 2^S$ , s.t.  
$$\begin{aligned} F_{\beta_1, \beta_2}([\varphi]) &= [\beta_2] \cup ([\beta_1] \cap \text{Preimage}([\varphi])) \\ &= [\beta_2] \cup ([\beta_1] \cap \{s \mid \exists s' \in [\varphi] \text{ s.t. } \langle s, s' \rangle \in R\}) \end{aligned}$$
- $F_{\beta_1, \beta_2}$  Monotonic:  $a \subseteq a' \implies F_{\beta_1, \beta_2}(a) \subseteq F_{\beta_1, \beta_2}(a')$ 
  - (Tarski's theorem):  $\mu x. F_{\beta_1, \beta_2}(x)$  always exists
  - (Kleene's theorem):  $\mu x. F_{\beta_1, \beta_2}(x)$  can be computed as the limit  $\emptyset \subseteq F_{\beta_1, \beta_2}(\emptyset) \subseteq F_{\beta_1, \beta_2}(F_{\beta_1, \beta_2}(\emptyset)) \subseteq \dots$ , in a finite number of steps.

Theorem (Clarke & Emerson)

$$[\mathbf{E}(\beta_1 \mathbf{U} \beta_2)] = \mu Z. ([\beta_2] \cup ([\beta_1] \cap [\mathbf{EXZ}] ))$$

## Case **EU** [cont.]

- We can compute  $X := [\mathbf{E}(\beta_1 \mathbf{U} \beta_2)]$  inductively as follows:

$$X_0 := \emptyset$$

$$X_1 := F_{\beta_1, \beta_2}(\emptyset) = [\beta_2]$$

$$X_2 := F_{\beta_1, \beta_2}(F_{\beta_1, \beta_2}(\emptyset)) = [\beta_2] \cup ([\beta_1] \cap \text{Preimage}(X_1))$$

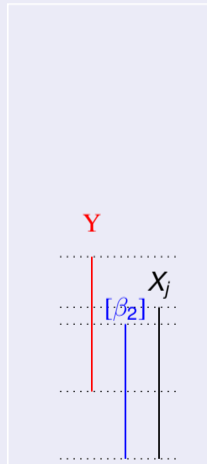
...

$$X_{j+1} := F_{\beta_1, \beta_2}^{j+1}(\emptyset) = [\beta_2] \cup ([\beta_1] \cap \text{Preimage}(X_j))$$

- Noticing that  $X_1 = [\beta_2]$  and  $X_{j+1} \supseteq X_j$  for every  $j \geq 0$ , and that  $([\beta_2] \cup Y) \supseteq X_j \supseteq [\beta_2] \implies ([\beta_2] \cup Y) = (X_j \cup Y)$ , we can use instead the following inductive schema:

- $X_1 := [\beta_2]$

- $X_{j+1} := X_j \cup ([\beta_1] \cap \text{Preimage}(X_j))$



## A relevant subcase: **EF**

- $\mathbf{EF}\beta = \mathbf{E}(\mathbf{TU}\beta)$
- $[T] = S \implies [T] \cap \text{Preimage}(X_j) = \text{Preimage}(X_j)$
- We can compute  $X := [\mathbf{EF}\beta]$  inductively as follows:
  - $X_1 := [\beta]$
  - $X_{j+1} := X_j \cup \text{Preimage}(X_j)$



# Outline

- 1 CTL Model Checking: general ideas
- 2 Some theoretical issues
- 3 CTL Model Checking: algorithms**
- 4 CTL Model Checking: some examples
- 5 A relevant subcase: invariants
- 6 Exercises

# General Schema

- Assume  $\varphi$  written in terms of  $\neg, \wedge, \mathbf{EX}, \mathbf{EU}, \mathbf{EG}$
- A general M.C. algorithm (**fix-point**):
  1. for every  $\varphi_i \in \text{Sub}(\varphi)$ , find  $[\varphi_i]$
  2. Check if  $I \subseteq [\varphi]$
- Subformulas  $\text{Sub}(\varphi)$  of  $\varphi$  are checked bottom-up
- To compute each  $[\varphi_i]$ : if the main operator of  $\varphi_i$  is a
  - **Propositional atoms**: apply labeling function
  - **Boolean operator**: apply standard set operations
  - **temporal operator**: apply recursively the tableaux rules, until a **fixpoint** is reached

## General M.C. Procedure

```
state_set Check(CTL_formula  $\beta$ ) {  
  case  $\beta$  of  
     $\top$ :           return  $S$ ;  
     $\perp$ :           return  $\{\}$ ;  
     $p$ :           return  $\{s \mid p \in L(s)\}$ ;  
     $\neg\beta_1$ :       return  $S / \text{Check}(\beta_1)$ ;  
     $\beta_1 \wedge \beta_2$ : return  $\text{Check}(\beta_1) \cap \text{Check}(\beta_2)$ ;  
    EX $\beta_1$ :       return  $\text{PreImage}(\text{Check}(\beta_1))$ ;  
    EG $\beta_1$ :       return  $\text{Check\_EG}(\text{Check}(\beta_1))$ ;  
    E( $\beta_1 \mathbf{U} \beta_2$ ): return  $\text{Check\_EU}(\text{Check}(\beta_1), \text{Check}(\beta_2))$ ;  
}
```

# Prelmage

Compute  $[EX\beta]$

```
state_set Prelmage(state_set  $[\beta]$ ) {  
   $X := \{\}$ ;  
  for each  $s \in S$  do  
    for each  $s'$  s.t.  $s' \in [\beta]$  and  $\langle s, s' \rangle \in R$  do  
       $X := X \cup \{s\}$ ;  
return  $X$ ;  
}
```

Compute **[EG $\beta$ ]**

```
state_set Check_EG(state_set [ $\beta$ ]) {  
   $X' := [\beta]; j := 1;$   
  repeat  
     $X := X'; j := j + 1;$   
     $X' := X \cap \text{Prelmage}(X);$   
  until ( $X' = X$ );  
  return  $X;$   
}
```

# Check\_EU

Compute  $[E(\beta_1 U \beta_2)]$

```
state_set Check_EU(state_set  $[\beta_1], [\beta_2]$ ) {  
   $X' := [\beta_2]; j := 1;$   
  repeat  
     $X := X'; j := j + 1;$   
     $X' := X \cup ([\beta_1] \cap \text{Prelmage}(X));$   
  until ( $X' = X$ );  
  return  $X;$   
}
```

## A relevant subcase: Check\_EF

Compute  $[EF\beta]$

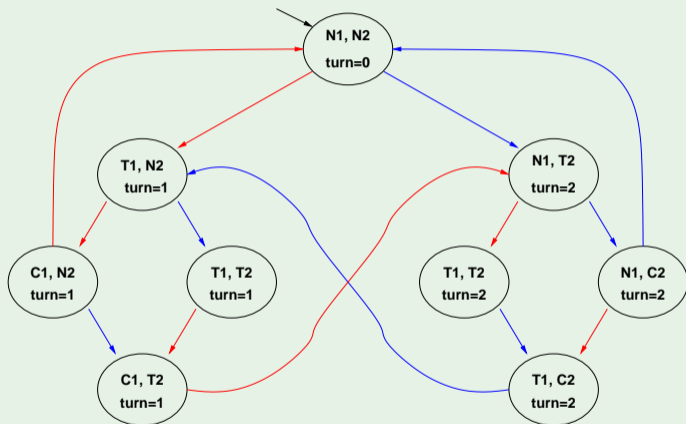
```
state_set Check_EF(state_set  $[\beta]$ ) {  
   $X' := [\beta]; j := 1;$   
  repeat  
     $X := X'; j := j + 1;$   
     $X' := X \cup \text{Prelmage}(X);$   
  until ( $X' = X$ );  
  return  $X;$   
}
```

# Outline

- 1 CTL Model Checking: general ideas
- 2 Some theoretical issues
- 3 CTL Model Checking: algorithms
- 4 CTL Model Checking: some examples**
- 5 A relevant subcase: invariants
- 6 Exercises



# Example 1: fairness



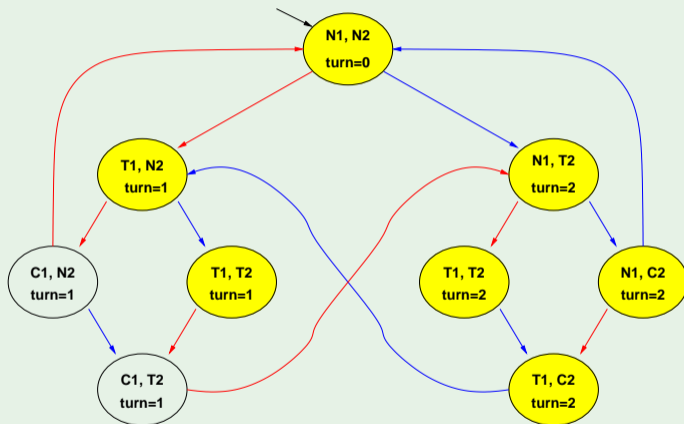
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

$[\neg C_1]$



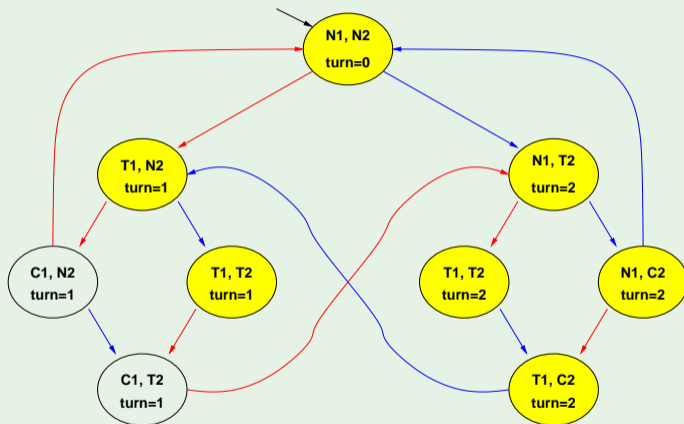
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

[ $\text{EG}\neg C_1$ ], step 0:



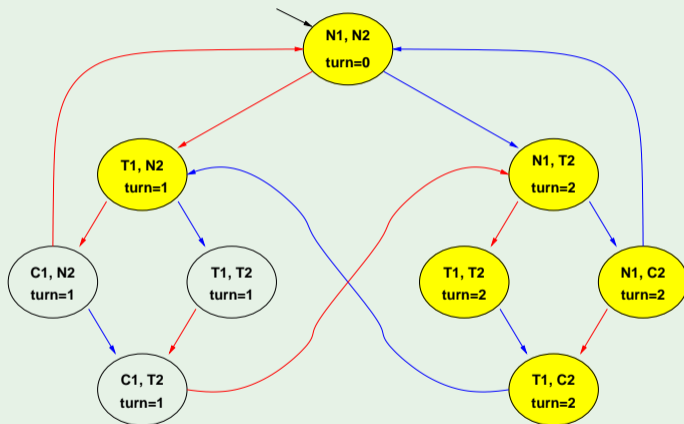
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG}\neg C_1 ?$

# Example 1: fairness

[ $\text{EG}\neg C_1$ ], step 1:



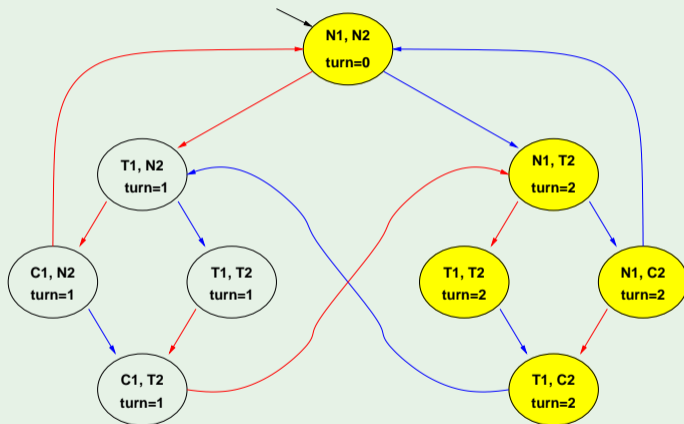
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG}\neg C_1 ?$

# Example 1: fairness

[ $\text{EG}\neg C_1$ ], step 2:



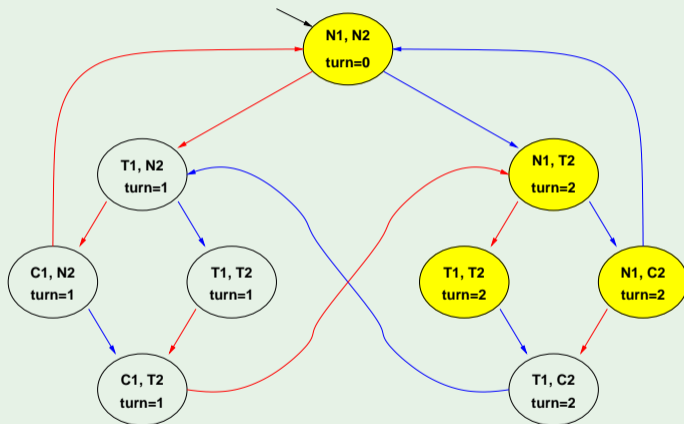
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG}\neg C_1 ?$

# Example 1: fairness

[ $EG\neg C_1$ ], step 3:



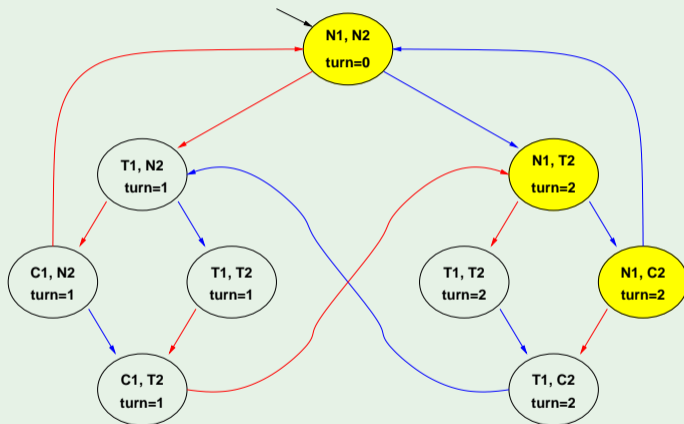
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG}\neg C_1 ?$

# Example 1: fairness

[ $\text{EG}\neg C_1$ ], step 4:



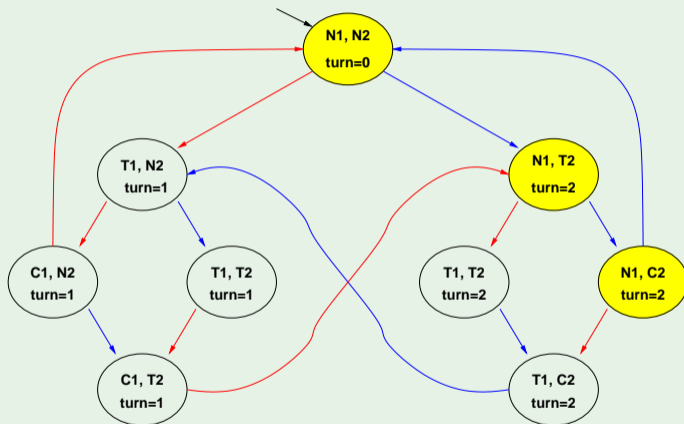
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG}\neg C_1 ?$

# Example 1: fairness

[EG $\neg$ C<sub>1</sub>], FIXPOINT!



N = noncritical, T = trying, C = critical

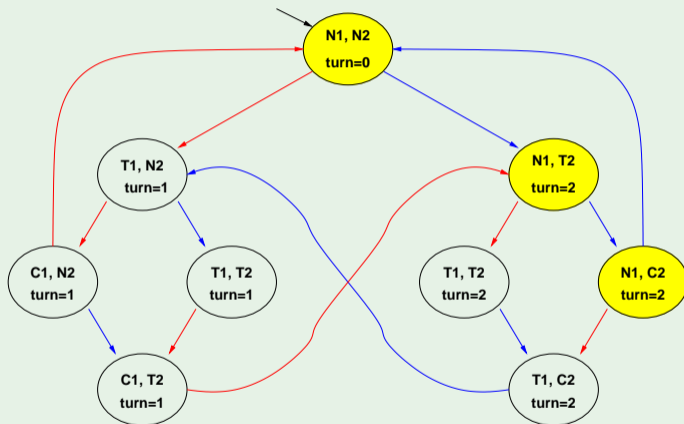
User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$



# Example 1: fairness

[EFEG $\neg$ C<sub>1</sub>], STEP 0



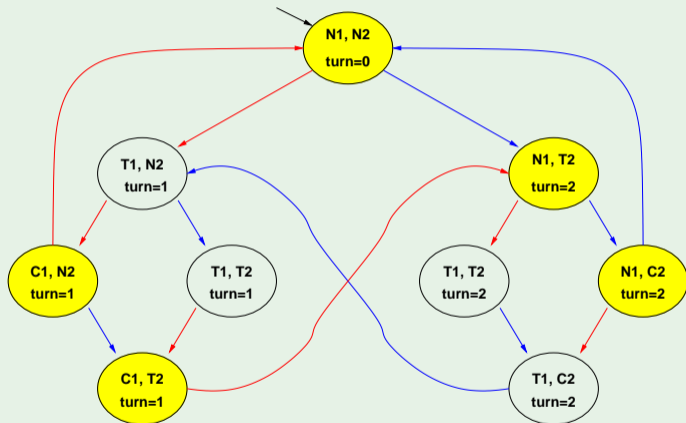
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

[EFEG $\neg$ C<sub>1</sub>], STEP 1



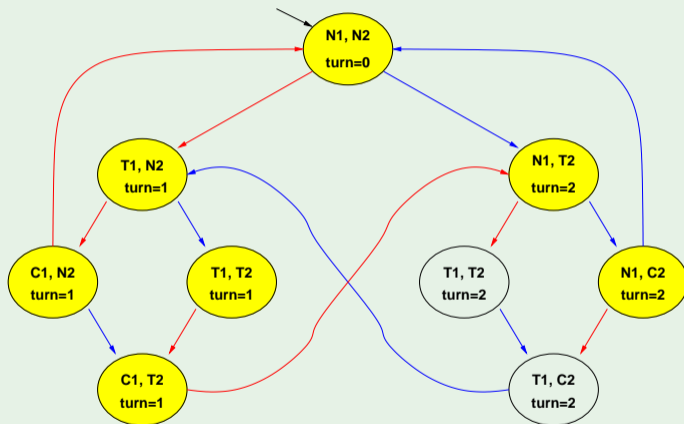
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

[EFEG $\neg$ C<sub>1</sub>], STEP 2

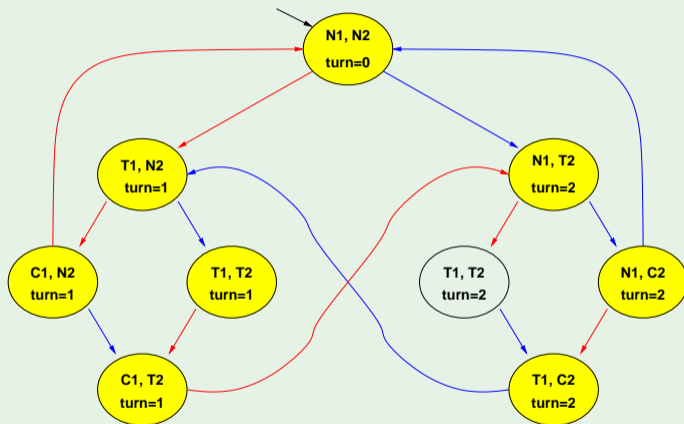


N = noncritical, T = trying, C = critical    User 1    User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

[EFEG $\neg$ C<sub>1</sub>], STEP 3



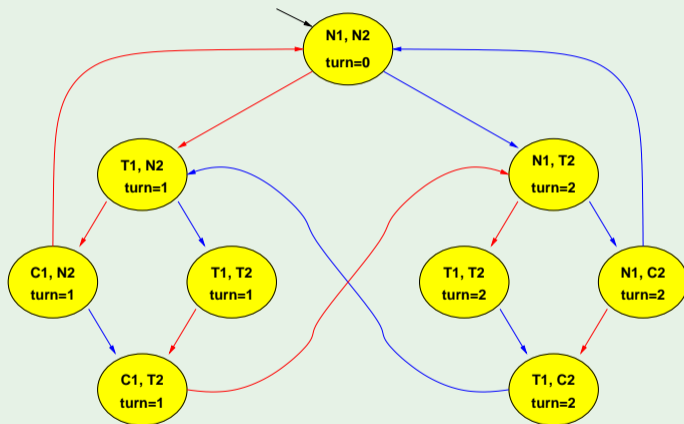
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

[EFEG $\neg$ C<sub>1</sub>], STEP 4



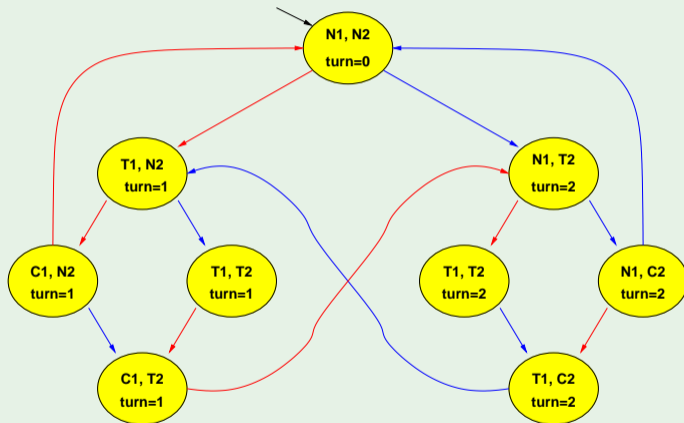
N = noncritical, T = trying, C = critical

User 1 User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

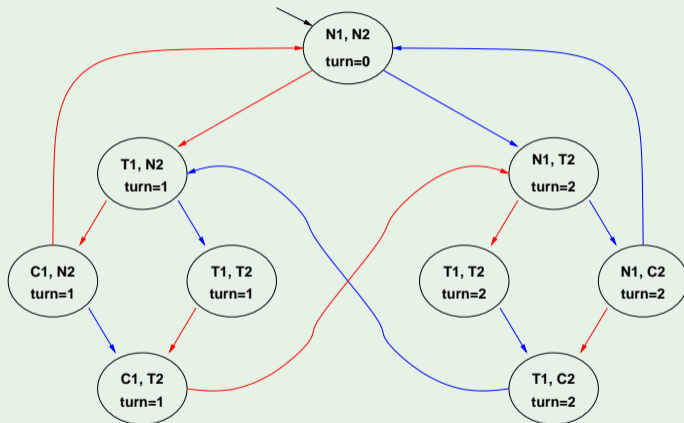
[EFEG $\neg$ C<sub>1</sub>], FIXPOINT!



$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ?$

# Example 1: fairness

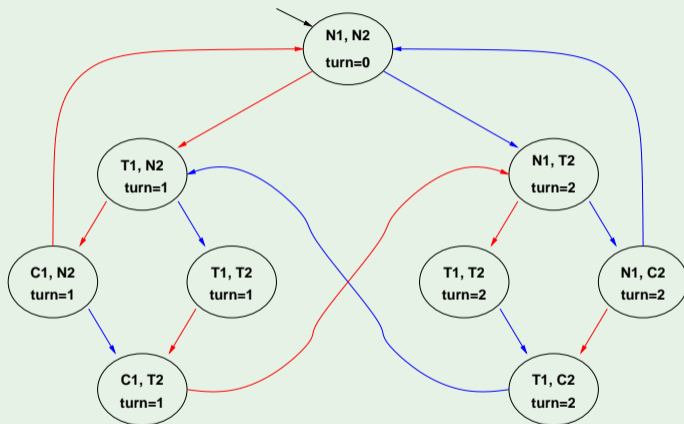
$[\neg \text{EFEG} \neg C_1]$



N = noncritical, T = trying, C = critical    User 1    User 2

$M \models \text{AGAF}C_1 ? \implies M \models \neg \text{EFEG} \neg C_1 ? \implies \text{NO!}$

## Example 2: liveness



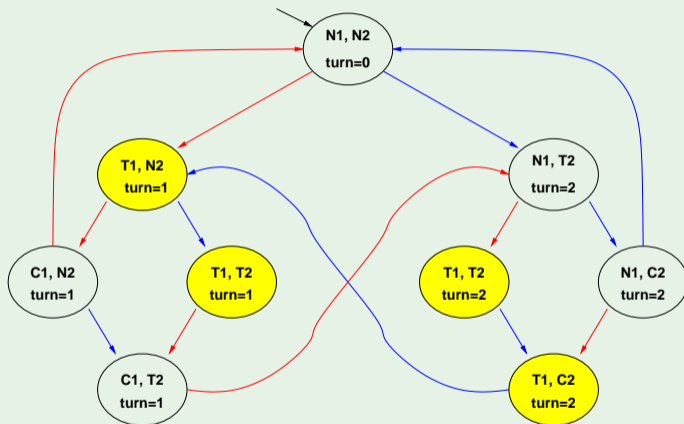
N = noncritical, T = trying, C = critical    User 1    User 2

$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AFC}_1) ? \implies M \models \neg \mathbf{EF}(T_1 \wedge \mathbf{EG} \neg C_1) ?$



## Example 2: liveness

$[T_1]$ :

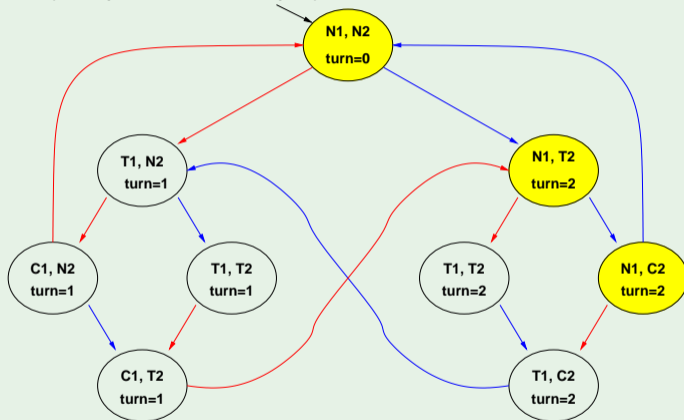


N = noncritical, T = trying, C = critical    User 1    User 2

$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AFC}_1) ? \implies M \models \neg \mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1) ?$

## Example 2: liveness

[ $\mathbf{EG}\neg C_1$ ], STEPS 0-4: (see previous example)

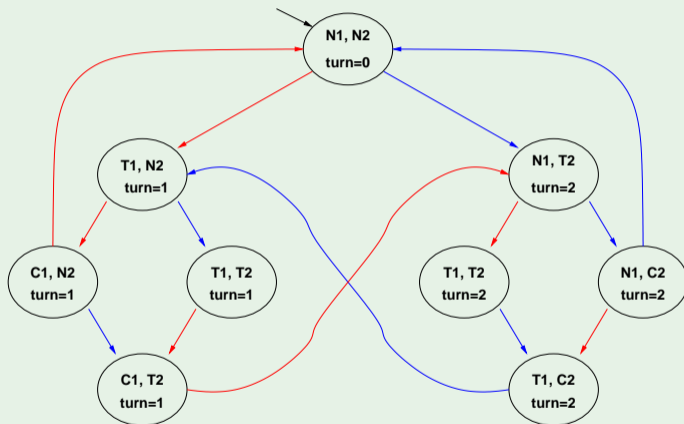


N = noncritical, T = trying, C = critical    User 1    User 2

$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AFC}_1) ? \implies M \models \neg \mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1) ?$

## Example 2: liveness

$[T_1 \wedge \mathbf{EG}\neg C_1]$  :

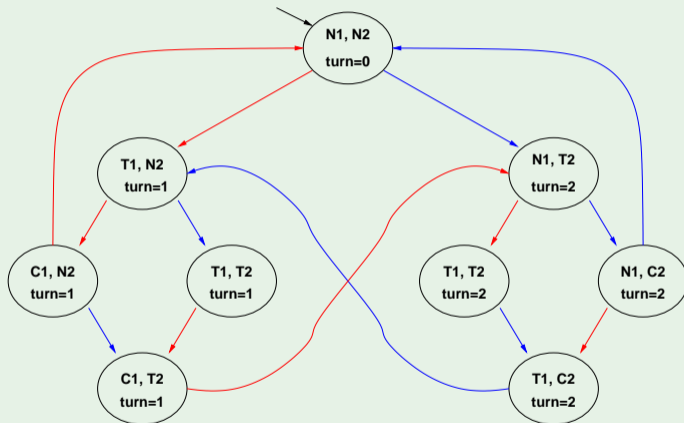


N = noncritical, T = trying, C = critical    User 1    User 2

$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AFC}_1) ? \implies M \models \neg \mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1) ?$

## Example 2: liveness

$[EF(T_1 \wedge EG\neg C_1)] :$

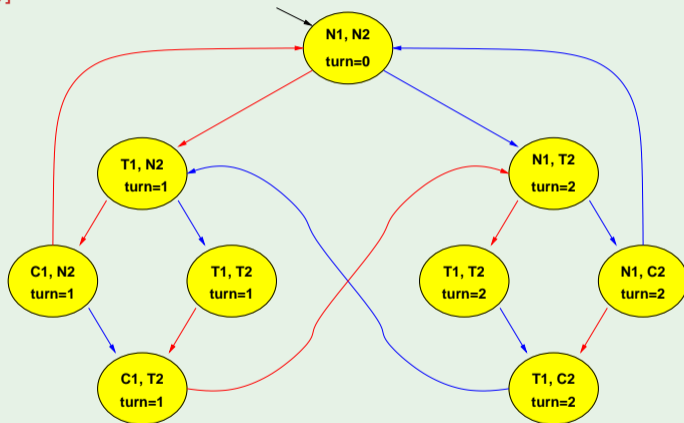


N = noncritical, T = trying, C = critical    User 1    User 2

$M \models AG(T_1 \rightarrow AFC_1) ? \implies M \models \neg EF(T_1 \wedge EG\neg C_1) ?$

## Example 2: liveness

$[\neg \mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)] :$



N = noncritical, T = trying, C = critical    User 1    User 2

$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AFC}_1) ? \implies M \models \neg \mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1) ? \text{ YES!}$



*The property verified is...*

# Homework

Apply the same process to all the CTL examples of Chapter 3.

# Complexity of CTL Model Checking: $M \models \varphi$

- Step 1: compute  $[\varphi]$ 
  - Compute  $[\varphi]$  bottom-up on the  $O(|\varphi|)$  sub-formulas of  $\varphi$ :  
 $O(|\varphi|)$  steps...
  - ... each requiring at most exploring  $O(|M|)$  states

$\implies O(|M| \cdot |\varphi|)$  steps

- Step 2: check  $I \subseteq [\varphi]$ :  $O(|M|)$

$\implies O(|M| \cdot |\varphi|)$



# Outline

- 1 CTL Model Checking: general ideas
- 2 Some theoretical issues
- 3 CTL Model Checking: algorithms
- 4 CTL Model Checking: some examples
- 5 A relevant subcase: invariants**
- 6 Exercises

# Model Checking of Invariants

- Invariant properties have the form **AG p** (e.g., **AG** $\neg$ *bad*)
- Checking invariants is the negation of a reachability problem:
  - is there a reachable state that is also a bad state? (**AG** $\neg$ *bad* =  $\neg$ **EF***bad*)
- Standard M.C. algorithm reasons **backward** from the *bad* by iteratively applying PreImage:

$$Y' := Y \cup \text{PreImage}(Y)$$

until a fixed point is reached.

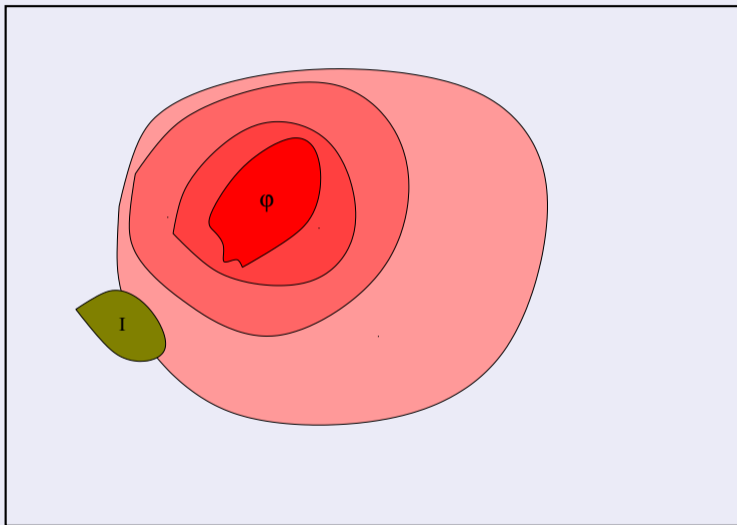
Then the complement is computed and *I* is checked for inclusion in the resulting set.

- Better algorithm: reasons **backward** from the *bad* by iteratively applying PreImage:

$$Y' := Y \cup \text{PreImage}(Y)$$

until (i) it intersect [*I*] or (ii) a fixed point is reached

## Model Checking of Invariants [cont.]



# Forward Model Checking of Invariants

Alternative algorithm (often more efficient): **forward checking**

- Compute the set of bad states  $[bad]$
- Compute the set of initial states  $I$
- Compute incrementally the **set of reachable states from  $I$**  until (i) it intersect  $[bad]$  or (ii) a fixed point is reached
- Basic step is the **(Forward) Image**:

$$Image(Y) \stackrel{\text{def}}{=} \{s' \mid s \in Y \text{ and } R(s, s') \text{ holds}\}$$

- Simplest form: compute the set of reachable states.

## Computing Reachable states: basic

```
State_Set Compute_reachable() {  
     $Y' := I; Y := \emptyset; j := 1;$   
    while ( $Y' \neq Y$ )  
         $j := j + 1;$   
         $Y := Y';$   
         $Y' := Y \cup \text{Image}(Y);$   
    }  
return  $Y;$   
}
```

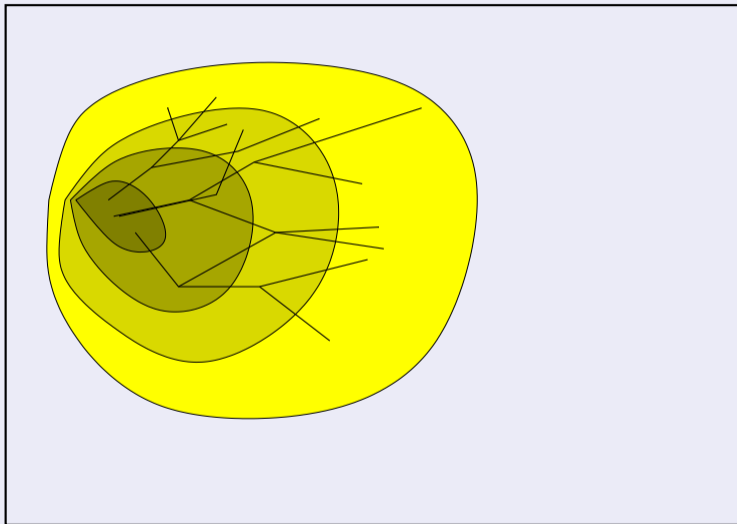
$Y = \text{reachable}$

## Computing Reachable states: advanced

```
State_Set Compute_reachable() {  
     $Y := F := I; j := 1;$   
    while ( $F \neq \emptyset$ )  
         $j := j + 1;$   
         $F := \text{Image}(F) \setminus Y;$   
         $Y := Y \cup F;$   
    }  
return  $Y;$   
}
```

$Y$ =reachable;  $F$ =frontier (new)

## Computing Reachable states [cont.]



## Checking of Invariant Properties: basic

```
bool Forward_Check_EF(State_Set BAD) {  
    Y := I; Y' :=  $\emptyset$ ; j := 1;  
    while (Y'  $\neq$  Y) and (Y'  $\cap$  BAD) =  $\emptyset$   
        j := j + 1;  
        Y := Y';  
        Y' := Y  $\cup$  Image(Y);  
    }  
    if (Y'  $\cap$  BAD)  $\neq$   $\emptyset$  // counter-example  
        return true  
    else // fixpoint reached  
        return false  
    }  
    }  
  
Y=reachable;
```

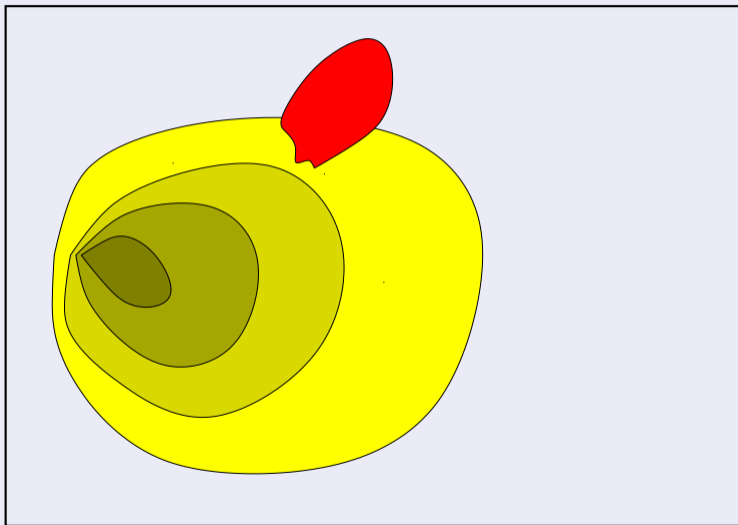


## Checking of Invariant Properties: advanced

```
bool Forward_Check_EF(State_Set BAD) {  
    Y := F := I; j := 1;  
    while (F ≠ ∅) and (F ∩ BAD) = ∅  
        j := j + 1;  
        F := Image(F) \ Y;  
        Y := Y ∪ F;  
    }  
    if (F ∩ BAD) ≠ ∅ // counter-example  
        return true  
    else // fixpoint reached  
        return false  
}
```

*Y*=reachable;*F*=frontier (new)

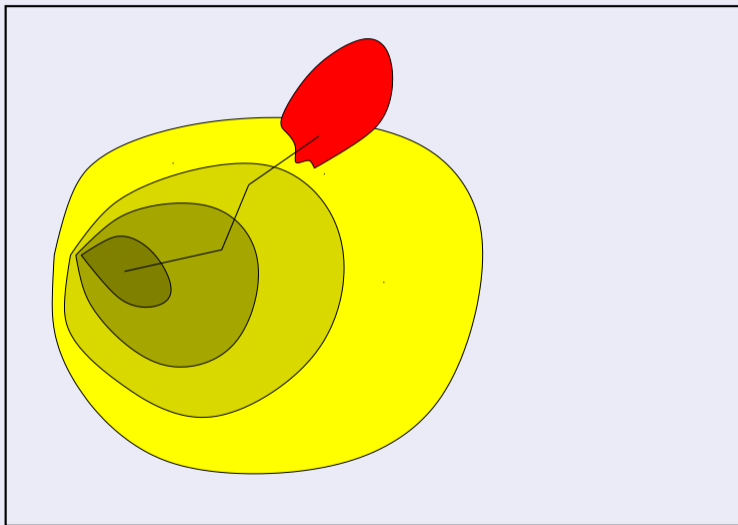
## Checking of Invariant Properties [cont.]



## Checking of Invariants: Counterexamples

- if layer  $n$  intersects with the bad states, then the property is violated
- a counterexample can be reconstructed proceeding backwards
  - (i) select any state of  $BAD \cap F[n]$  (we know it is satisfiable), call it  $t[n]$
  - (ii) compute  $Preimage(t[n])$ , i.e. the states that can result in  $t[n]$  in one step
  - (iii) compute  $Preimage(t[n]) \cap F[n - 1]$ , and select one state  $t[n - 1]$
- iterate (i)-(iii) until the initial states are reached
- $t[0], t[1], \dots, t[n]$  is our counterexample

## Checking of Invariants: Counterexamples [cont.]

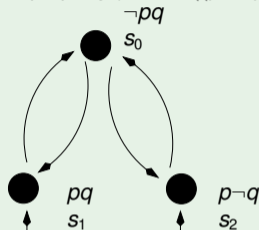


# Outline

- 1 CTL Model Checking: general ideas
- 2 Some theoretical issues
- 3 CTL Model Checking: algorithms
- 4 CTL Model Checking: some examples
- 5 A relevant subcase: invariants
- 6 Exercises**

# Ex: CTL Model Checking

Consider the Kripke Model  $M$  below, and the CTL property  $\varphi \stackrel{\text{def}}{=} \mathbf{AG}((p \wedge q) \rightarrow \mathbf{EG}q)$ .



(a) Rewrite  $\varphi$  into an equivalent formula  $\varphi'$  expressed in terms of **EX**, **EG**, **EU/EF** only.

[ Solution:  $\varphi' = \neg \mathbf{EF} \neg ((\neg p \vee \neg q) \vee \mathbf{EG}q) = \neg \mathbf{EF}((p \wedge q) \wedge \neg \mathbf{EG}q)$  ]

(b) Compute bottom-up the denotations of all subformulas of  $\varphi'$ . (Ex:  $[p] = \{s_1, s_2\}$ )

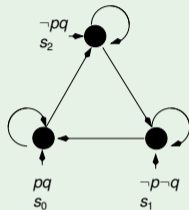
|             |                  |     |                |   |     |                     |   |
|-------------|------------------|-----|----------------|---|-----|---------------------|---|
| [ Solution: | $[p]$            | $=$ | $\{s_1, s_2\}$ | $[\neg \mathbf{EG}q]$                                       | $=$ | $\{s_2\}$           |   |
|             | $[q]$            | $=$ | $\{s_0, s_1\}$ | $[(p \wedge q) \wedge \neg \mathbf{EG}q]$                   | $=$ | $\{\}$              |   |
|             | $[(p \wedge q)]$ | $=$ | $\{s_1\}$      | $[\mathbf{EF}((p \wedge q) \wedge \neg \mathbf{EG}q)]$      | $=$ | $\{\}$              | ] |
|             | $[\mathbf{EG}q]$ | $=$ | $\{s_0, s_1\}$ | $[\neg \mathbf{EF}((p \wedge q) \wedge \neg \mathbf{EG}q)]$ | $=$ | $\{s_0, s_1, s_2\}$ |   |

(c) As a consequence of point (b), say whether  $M \models \varphi$  or not.

[ Solution: Yes,  $\{s_1, s_2\} \subseteq [\varphi']$ . ]

# Ex: CTL Model Checking

Consider the Kripke Model  $M$  below, and the CTL property  $\mathbf{AG}(\mathbf{AF}p \rightarrow \mathbf{AF}q)$ .



(a) Rewrite  $\varphi$  into an equivalent formula  $\varphi'$  expressed in terms of **EX**, **EG**, **EU/EF** only.

[ Solution:  $\varphi' = \mathbf{AG}(\mathbf{AF}p \rightarrow \mathbf{AF}q) = \neg \mathbf{EF} \neg (\neg \mathbf{EG} \neg p \rightarrow \neg \mathbf{EG} \neg q) = \neg \mathbf{EF}(\neg \mathbf{EG} \neg p \wedge \mathbf{EG} \neg q)$  ]

(b) Compute bottom-up the denotations of all subformulas of  $\varphi'$ . (Ex:  $[p] = \{s_1, s_2\}$ )

|             |                             |     |                |   |     |                     |   |
|-------------|-----------------------------|-----|----------------|---|-----|---------------------|---|
|             | $[p]$                       | $=$ | $\{s_0\}$      | $[\neg q]$  | $=$ | $\{s_1\}$           |   |
|             | $[\neg p]$                  | $=$ | $\{s_1, s_2\}$ | $[\mathbf{EG} \neg q]$  | $=$ | $\{s_1\}$           |   |
| [ Solution: | $[\mathbf{EG} \neg p]$      | $=$ | $\{s_1, s_2\}$ | $[\neg \mathbf{EG} \neg p \wedge \mathbf{EG} \neg q]$                   | $=$ | $\{\}$              | ] |
|             | $[\neg \mathbf{EG} \neg p]$ | $=$ | $\{s_0\}$      | $[\mathbf{EF}(\neg \mathbf{EG} \neg p \wedge \mathbf{EG} \neg q)]$      | $=$ | $\{\}$              |   |
|             | $[q]$                       | $=$ | $\{s_0, s_2\}$ | $[\neg \mathbf{EF}(\neg \mathbf{EG} \neg p \wedge \mathbf{EG} \neg q)]$ | $=$ | $\{s_0, s_1, s_2\}$ |   |

(c) As a consequence of point (b), say whether  $M \models \varphi$  or not.

[ Solution: Yes,  $\{s_0, s_1, s_2\} \subseteq [\varphi']$ . ]