# Introduction to Formal Methods Chapter 04: CTL Model Checking

#### Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it URL: http://disi.unitn.it/rseba/DIDATTICA/fm2020/ Teaching assistant: Enrico Magnago – enrico.magnago@unitn.it

#### CDLM in Informatica, academic year 2019-2020

last update: Monday 18th May, 2020, 14:48

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, and S. Tonetta, who detain its copyright. Some exampes displayed in these slides are taken from [Clarke, Grunberg & Feled, "Model Checking", MIT Press], and their copyright is detained 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.

Roberto Sebastiani

Ch. 04: CTL Model Checking

#### Outline

- CTL Model Checking: general ideas
- CTL Model Checking: a simple example
- Some theoretical issues
- CTL Model Checking: algorithms
- 5 CTL Model Checking: some examples
  - A relevant subcase: invariants
  - Exercises

イロト イポト イヨト イヨト

#### Outline



- 2) CTL Model Checking: a simple example
- 3 Some theoretical issues
- 4 CTL Model Checking: algorithms
- 5 CTL Model Checking: some examples
- 6 A relevant subcase: invariants
- 7) Exercises

イロト イポト イヨト イヨト

#### 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 $\varphi$ :

#### $\mathbf{AG}(p ightarrow \mathbf{AF}q)$

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

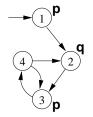
Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 4/72

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  $\varphi$ :

 $AG(p \rightarrow AFq)$ 

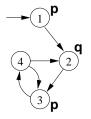
• ...the model checking algorithm checks whether in all initial states of M all the executions of the model satisfy the formula  $(M \models \varphi)$ .

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 4/72

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 φ:

#### $\textbf{AG}(\rho \rightarrow \textbf{AF}q)$

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

Roberto Sebastiani

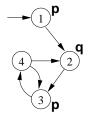
Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 4/72

イロト イポト イヨト イヨト

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  $\varphi$ :

AG(
ho 
ightarrow AFq)

 ...the model checking algorithm checks whether in all initial states of M all the executions of the model satisfy the formula (M ⊨ φ).

Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 4/72

• (10) • (10)

# CTL Model Checking: General Idea

#### Two macro-steps:

- 1 construct the set of states where the formula holds:  $[\varphi] := \{ s \in S : M, s \models \varphi \}$  $([\varphi] \text{ is called the denotation of } \varphi)$
- 2 then compare with the set of initial states  $L \subseteq [1,2]$ 
  - $I \subseteq [\varphi]$  ?

# CTL Model Checking: General Idea

Two macro-steps:

 construct the set of states where the formula holds:
 [φ] := {s ∈ S : M, s ⊨ φ} ([φ] is called the denotation of φ)
 then compare with the set of initial states:

# CTL Model Checking: General Idea

Two macro-steps:

1 construct the set of states where the formula holds:

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

```
([\varphi] is called the denotation of \varphi)
```

2 then compare with the set of initial states:

 $I \subseteq [\varphi]$  ?

#### In order to compute $[\varphi]$ :

- proceed "bottom-up" on the structure of the formula, computing  $[\varphi_i]$  for each subformula  $\varphi_i$  of  $AG(p \rightarrow AFq)$ :
  - [q],
  - [AFq],
  - [p],
  - $[p \rightarrow AFq],$
  - [AG(
    ho 
    ightarrow AFq)]

In order to compute  $[\varphi]$ :

- proceed "bottom-up" on the structure of the formula, computing [φ<sub>i</sub>] for each subformula φ<sub>i</sub> of AG(p → AFq):
  - [q],
    [AFq],
  - [p],
  - $[p \rightarrow AFq]$ ,
  - $[\mathbf{AG}(p \rightarrow \mathbf{AF}q)]$

In order to compute  $[\varphi]$ :

- proceed "bottom-up" on the structure of the formula, computing [φ<sub>i</sub>] for each subformula φ<sub>i</sub> of AG(p → AFq):
  - [q],
  - [AFq],
  - [p],
  - [*p* → AF*q*],
     [AG(*p* → AF*q*)]

In order to compute  $[\varphi]$ :

- proceed "bottom-up" on the structure of the formula, computing [φ<sub>i</sub>] for each subformula φ<sub>i</sub> of AG(p → AFq):
  - [q],
  - [**AF**q],
  - [p],
  - $[p \rightarrow AFq]$ , •  $[AG(p \rightarrow AFq]]$
  - $[\mathbf{AG}(p \rightarrow \mathbf{AF}q)]$

In order to compute  $[\varphi]$ :

- proceed "bottom-up" on the structure of the formula, computing [φ<sub>i</sub>] for each subformula φ<sub>i</sub> of AG(p → AFq):
  - [q],
  - [**AF***q*],
  - [*p*],
  - [*p* → AF*q*],
     [AG(*p* → AF*q*)]

A D K A D K A D K A D K A D K

In order to compute  $[\varphi]$ :

- proceed "bottom-up" on the structure of the formula, computing [φ<sub>i</sub>] for each subformula φ<sub>i</sub> of AG(p → AFq):
  - [q],
  - [**AF***q*],
  - [*p*],
  - $[p \rightarrow \mathbf{AF}q],$
  - $[\mathbf{AG}(p \to \mathbf{AF}q)]$

イロト イポト イヨト イヨト

In order to compute  $[\varphi]$ :

- proceed "bottom-up" on the structure of the formula, computing [φ<sub>i</sub>] for each subformula φ<sub>i</sub> of AG(p → AFq):
  - [q],
  - [**AF***q*],
  - [p],
  - $[p \rightarrow AFq]$ ,
  - $[\mathbf{AG}(p \rightarrow \mathbf{AF}q)]$

#### 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

イロト イポト イヨト イヨト

#### 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

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

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

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

イロト イポト イヨト イヨト

#### Tableaux rules: a quote



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

Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 8/72

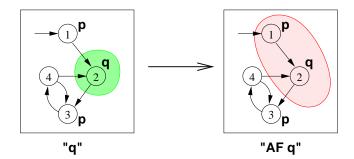
#### Outline



#### CTL Model Checking: a simple example

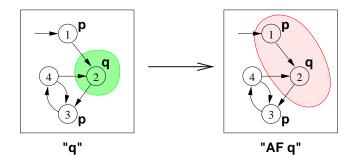
- Some theoretical issues
- 4 CTL Model Checking: algorithms
- 5 CTL Model Checking: some examples
- 6 A relevant subcase: invariants
- Zercises

イロト イポト イヨト イヨト



• Recall the AF tableau rule:  $AFq \leftrightarrow (q \lor AXAFq)$ • Iteration:  $[AFq]^{(1)} = [q]; [AFq]^{(i+1)} = [q] \cup AX[AFq]^{(i)}$ 

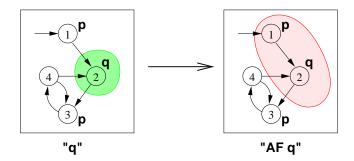
- $[\mathsf{AF}q]^{(1)} = [q] = \{2\}$
- $[\mathbf{AFq}]^{(2)} = [q \lor \mathbf{AXq}] = \{2\} \cup \{1\} = \{1, 2\}$
- $[\mathsf{AF}q]^{(3)} = [q \lor \mathsf{AX}(q \lor \mathsf{AX}q)] = \{2\} \cup \{1\} = \{1, 2\}$



• Recall the AF tableau rule:  $AFq \leftrightarrow (q \lor AXAFq)$ • Iteration:  $[AFq]^{(1)} = [q]; [AFq]^{(i+1)} = [q] \cup AX[AFq]^{(i)}$ 

•  $[\mathbf{AFq}]^{(2)} = [q \lor \mathbf{AXq}] = \{2\} \cup \{1\} = \{1, 2\}$ 

•  $[\mathsf{AF}q]^{(3)} = [q \lor \mathsf{AX}(q \lor \mathsf{AX}q)] = \{2\} \cup \{1\} = \{1, 2\}$ 



- Recall the AF tableau rule:  $AFq \leftrightarrow (q \lor AXAFq)$
- Iteration:  $[AFq]^{(1)} = [q]; [AFq]^{(i+1)} = [q] \cup AX[AFq]^{(i)}$ 
  - $[\mathbf{AF}q]^{(1)} = [q] = \{2\}$
  - $[\mathbf{AFq}]^{(2)}_{(1)} = [q \lor \mathbf{AXq}] = \{2\} \cup \{1\} = \{1,2\}$
  - $[\mathbf{AFq}]^{(3)} = [q \lor \mathbf{AX}(q \lor \mathbf{AX}q)] = \{2\} \cup \{1\} = \{1, 2\}$

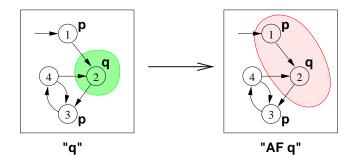
Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020

10/72

イロト イポト イヨト イヨト 二日

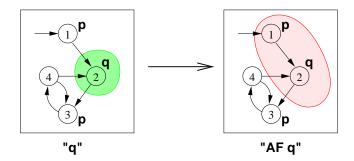


- Recall the AF tableau rule:  $AFq \leftrightarrow (q \lor AXAFq)$
- Iteration:  $[AFq]^{(1)} = [q]; [AFq]^{(i+1)} = [q] \cup AX[AFq]^{(i)}$ 
  - $[\mathbf{AF}q]^{(1)} = [q] = \{2\}$
  - $[\mathbf{AFq}]^{(2)}_{(1)} = [q \lor \mathbf{AXq}] = \{2\} \cup \{1\} = \{1,2\}$
  - $[\mathbf{AFq}]^{(3)} = [q \lor \mathbf{AX}(q \lor \mathbf{AXq})] = \{2\} \cup \{1\} = \{1, 2\}$

Roberto Sebastiani

10/72

イロト イポト イヨト イヨト 二日



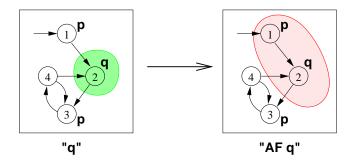
- Recall the AF tableau rule:  $AFq \leftrightarrow (q \lor AXAFq)$
- Iteration:  $[AFq]^{(1)} = [q]; [AFq]^{(i+1)} = [q] \cup AX[AFq]^{(i)}$

• 
$$[\mathbf{AF}q]^{(1)}_{\dots} = [q] = \{2\}$$

• 
$$[\mathbf{AFq}]^{(2)} = [q \lor \mathbf{AXq}] = \{2\} \cup \{1\} = \{1, 2\}$$

•  $[\mathbf{AFq}]^{(3)} = [q \lor \mathbf{AX}(q \lor \mathbf{AXq})] = \{2\} \cup \{1\} = \{1, 2\}$ 

10/72



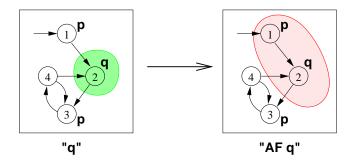
- Recall the AF tableau rule:  $AFq \leftrightarrow (q \lor AXAFq)$
- Iteration:  $[AFq]^{(1)} = [q]; [AFq]^{(i+1)} = [q] \cup AX[AFq]^{(i)}$

• 
$$[\mathbf{AF}q]^{(1)} = [q] = \{2\}$$

• 
$$[\mathbf{AFq}]_{(2)}^{(2)} = [q \lor \mathbf{AXq}] = \{2\} \cup \{1\} = \{1,2\}$$

•  $[\mathbf{AFq}]^{(3)} = [q \lor \mathbf{AX}(q \lor \mathbf{AX}q)] = \{2\} \cup \{1\} = \{1, 2\}$ 

Roberto Sebastiani



- Recall the AF tableau rule:  $AFq \leftrightarrow (q \lor AXAFq)$
- Iteration:  $[AFq]^{(1)} = [q]; [AFq]^{(i+1)} = [q] \cup AX[AFq]^{(i)}$

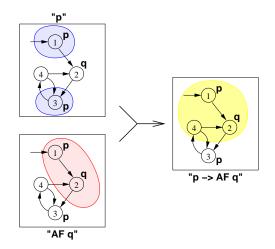
• 
$$[\mathbf{AFq}]^{(1)}_{(1)} = [q] = \{2\}$$

• 
$$[\mathbf{AFq}]_{(2)}^{(2)} = [q \lor \mathbf{AXq}] = \{2\} \cup \{1\} = \{1,2\}$$

[AFq]<sup>(3)</sup> = [q ∨ AX(q ∨ AXq)] = {2} ∪ {1} = {1,2}
 ⇒ (fix point reached)

Roberto Sebastiani

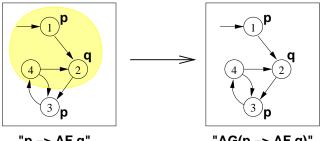
(周) (日) (日) (日) (日)



Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 11/72

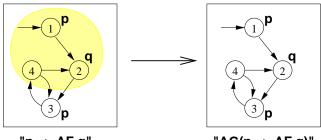
э



"p -> AF q"

"AG(p -> AF q)"

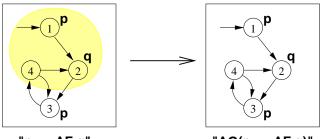
- Recall the AG tableau rule:  $AG\varphi \leftrightarrow (\varphi \land AXAG\varphi)$ • Iteration:  $[\mathbf{AG}\varphi^{(1)}] = [\varphi]; \quad [\mathbf{AG}\varphi]^{(i+1)} = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi]^{(i)}$



"p –> AF q"

"AG(p -> AF q)"

Recall the AG tableau rule: AGφ ↔ (φ ∧ AXAGφ)
 Iteration: [AGφ<sup>(1)</sup>] = [φ]; [AGφ]<sup>(i+1)</sup> = [φ] ∩ AX[AGφ]<sup>(i)</sup>
 [AGφ<sup>(1)</sup>] = [φ] ∧ AX[AGφ<sup>(1)</sup>] = [φ] ∧ AX[AGφ<sup>(1)</sup>]
 [AGφ<sup>(1)</sup>] = [φ] ∧ AX[AGφ<sup>(1)</sup>] = [φ] ∧ AX[AGφ<sup>(1)</sup>]



"p –> AF q"

"AG(p -> AF q)"

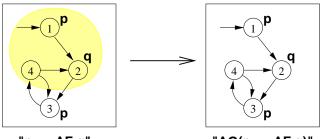
- Recall the AG tableau rule:  $AG\varphi \leftrightarrow (\varphi \land AXAG\varphi)$
- Iteration:  $[\mathbf{AG}\varphi^{(1)}] = [\varphi]; \quad [\mathbf{AG}\varphi]^{(i+1)} = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi]^{(i)}$ 
  - $(1) \quad [\mathsf{AG}\varphi]^{(1)}_{(2)} = [\varphi] = \{1, 2, 4\}$
  - **3**  $\left[\mathsf{AG}\varphi\right]_{(2)}^{(2)} = \left[\varphi\right] \cap \mathsf{AX}\left[\mathsf{AG}\varphi\right]_{(2)}^{(1)} = \{1, 2, 4\} \cap \{1, 3\} = \{1\}$
  - **③**  $[\mathbf{AG}\varphi]^{(3)} = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi]^{(2)} = \{1, 2, 4\} \cap \{\} = \{\}$

Roberto Sebastiani

Monday 18<sup>th</sup> May, 2020

- 3

12/72



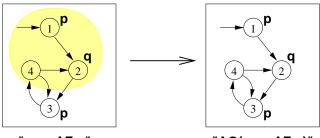
"p –> AF q"

"AG(p -> AF q)"

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

A B A B A B A
 A B A
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 A
 A
 A
 A

- 3



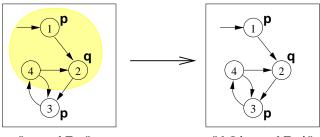
"p –> AF q"

"AG(p -> AF q)"

- Recall the AG tableau rule:  $AG\varphi \leftrightarrow (\varphi \land AXAG\varphi)$
- Iteration:  $[\mathbf{AG}\varphi^{(1)}] = [\varphi]; \quad [\mathbf{AG}\varphi]^{(i+1)} = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi]^{(i)}$ •  $[\mathbf{AG}\varphi]^{(1)} = [\varphi] = \{1, 2, 4\}$ 
  - **2**  $[\mathbf{AG}\varphi]^{(2)} = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi]^{(1)} = \{1, 2, 4\} \cap \{1, 3\} = \{1\}$
  - **(a)**  $[\mathsf{AG}\varphi]^{(3)} = [\varphi] \cap \mathsf{AX}[\mathsf{AG}\varphi]^{(2)} = \{1, 2, 4\} \cap \{\} = \{\}$

・ロト ・ 同ト ・ ヨト ・ ヨト

- 3



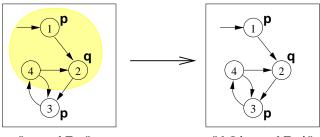
"p -> AF q"

"AG(p -> AF q)"

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

 $\Rightarrow$  (fix point reached)

(個) (ヨ) (ヨ) (ヨ)



"p –> AF q"

"AG(p -> AF q)"

- Recall the AG tableau rule:  $AG\varphi \leftrightarrow (\varphi \land AXAG\varphi)$
- Iteration:  $[\mathbf{AG}\varphi^{(1)}] = [\varphi]; \quad [\mathbf{AG}\varphi]^{(i+1)} = [\varphi] \cap \mathbf{AX}[\mathbf{AG}\varphi]^{(i)}$

$$[AG\varphi]^{(2)} = [\varphi] \cap AX[AG\varphi]^{(1)} = \{1, 2, 4\} \cap \{1, 3\} = \{1\}$$

 $[AG\varphi]^{(3)} = [\varphi] \cap AX[AG\varphi]^{(2)} = \{1, 2, 4\} \cap \{\} = \{\}$  $\implies (fix point reached)$ 

Roberto Sebastiani

- The set of states where the formula holds is empty
   ⇒ the initial state does not satisfy the property
   ⇒ M ⊭ AG(p → AFq)
- Counterexample: a lazo-shaped path: 1, 2, {3,4}<sup>ω</sup> (satisfying EF(p ∧ EG¬q))

#### Note

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

- The set of states where the formula holds is empty
   ⇒ the initial state does not satisfy the property
   ⇒ M ⊭ AG(p → AFq)
- Counterexample: a lazo-shaped path: 1, 2, {3,4}<sup>ω</sup> (satisfying EF(p ∧ EG¬q))

#### Note

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

- The set of states where the formula holds is empty  $\implies$  the initial state does not satisfy the property  $\implies M \not\models AG(p \rightarrow AFq)$
- Counterexample: a lazo-shaped path: 1, 2, {3,4}<sup>ω</sup> (satisfying EF(p ∧ EG¬q))

#### Note

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

- The set of states where the formula holds is empty  $\implies$  the initial state does not satisfy the property  $\implies M \not\models AG(p \rightarrow AFq)$
- Counterexample: a lazo-shaped path: 1, 2, {3,4}<sup>ω</sup> (satisfying EF(p ∧ EG¬q))

#### Note

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

- The set of states where the formula holds is empty  $\implies$  the initial state does not satisfy the property  $\implies M \not\models AG(p \rightarrow AFq)$
- Counterexample: a lazo-shaped path: 1, 2, {3,4}<sup>ω</sup> (satisfying EF(p ∧ EG¬q))

#### Note

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

## Outline

- CTL Model Checking: general ideas
- 2) CTL Model Checking: a simple example
  - Some theoretical issues
- 4) CTL Model Checking: algorithms
- 5 CTL Model Checking: some examples
- 6 A relevant subcase: invariants
- 7) Exercises

・ 同 ト ・ ヨ ト ・ ヨ ト

## The fixed-point theory of lattice of sets

#### Definition

For any finite set S, the structure (2<sup>S</sup>, ⊆) forms a complete lattice with ∪ as join and ∩ as meet operations.

• A function  $F : 2^S \mapsto 2^S$  is monotonic provided  $S_1 \subseteq S_2 \Rightarrow F(S_1) \subseteq F(S_2)$ .

Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 15/72

## The fixed-point theory of lattice of sets

### Definition

- For any finite set S, the structure (2<sup>S</sup>, ⊆) forms a complete lattice with ∪ as join and ∩ as meet operations.
- A function  $F : 2^S \mapsto 2^S$  is monotonic provided  $S_1 \subseteq S_2 \Rightarrow F(S_1) \subseteq F(S_2)$ .

・ 同 ト ・ ヨ ト ・ ヨ ト

### Definition

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

• Given a function  $F : 2^S \mapsto 2^S$ ,  $a \subseteq S$  is a fixed point of F iff

F(a) = a

- a is a least fixed point (LFP) of *F*, written µx.*F*(x), iff, for every other fixed point a' of *F*, a ⊆ a'
- a is a greatest fixed point (GFP) of *F*, written *vx*.*F*(*x*), iff, for every other fixed point *a*' of *F*, *a*' ⊆ *a*

イロト イポト イヨト イヨト

#### Definition

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

• Given a function  $F : 2^S \mapsto 2^S$ ,  $a \subseteq S$  is a fixed point of F iff

### F(a) = a

- a is a least fixed point (LFP) of *F*, written μx.*F*(x), iff, for every other fixed point a' of *F*, a ⊆ a'
- a is a greatest fixed point (GFP) of *F*, written *vx*.*F*(*x*), iff, for every other fixed point *a*' of *F*, *a*' ⊆ *a*

#### Definition

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

• Given a function  $F : 2^S \mapsto 2^S$ ,  $a \subseteq S$  is a fixed point of F iff

F(a) = a

- a is a least fixed point (LFP) of *F*, written µx.*F*(x), iff, for every other fixed point a' of *F*, a ⊆ a'
- a is a greatest fixed point (GFP) of *F*, written *vx*.*F*(*x*), iff, for every other fixed point *a*' of *F*, *a*' ⊆ *a*

- 3

#### Definition

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

• Given a function  $F : 2^S \mapsto 2^S$ ,  $a \subseteq S$  is a fixed point of F iff

$$F(a) = a$$

- a is a least fixed point (LFP) of *F*, written µx.*F*(x), iff, for every other fixed point a' of *F*, a ⊆ a'
- a is a greatest fixed point (GFP) of F, written vx.F(x), iff, for every other fixed point a' of F, a' ⊆ a

#### 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  $\emptyset \subset F(\emptyset) \subset F(F(\emptyset))$
- the greatest fixed point of F is the limit
  - $S \supseteq F(S) \supseteq F(F(S))$ .

Since 2<sup>s</sup> is finite, convergence is obtained in a finite number of steps.

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

#### 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<sup>S</sup> is finite, convergence is obtained in a finite number of steps.

イロト イポト イヨト イヨト

#### 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* ⊇ *F*(*S*) ⊇ *F*(*F*(*S*)) ...

Since 2<sup>S</sup> is finite, convergence is obtained in a finite number of steps.

#### 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 ⊇ F(S) ⊇ F(F(S))...

Since 2<sup>S</sup> is finite, convergence is obtained in a finite number of steps.

イロト イポト イヨト イヨト

#### 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<sup>S</sup> is finite, convergence is obtained in a finite number of steps.

# **CTL Model Checking and Lattices**

- If *M* = ⟨*S*, *I*, *R*, *L*, *AP*⟩ is a Kripke structure, then ⟨2<sup>S</sup>, ⊆⟩ is a complete lattice
- We identify φ with its denotation [φ]

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

## **CTL Model Checking and Lattices**

- If *M* = ⟨*S*, *I*, *R*, *L*, *AP*⟩ is a Kripke structure, then ⟨2<sup>S</sup>, ⊆⟩ is a complete lattice
- We identify  $\varphi$  with its denotation  $[\varphi]$

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

イロト イポト イヨト イヨト

# **CTL Model Checking and Lattices**

- If *M* = ⟨*S*, *I*, *R*, *L*, *AP*⟩ is a Kripke structure, then ⟨2<sup>S</sup>, ⊆⟩ is a complete lattice
- We identify  $\varphi$  with its denotation  $[\varphi]$
- ⇒ 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 $[\varphi]$

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

### Recursive definition of $[\varphi]$

$$[true] = S$$
  

$$[false] = \{\}$$
  

$$[p] = \{s | p \in L(s)\}$$
  

$$[\neg \varphi_1] = S/[\varphi_1]$$
  

$$[\varphi_1 \land \varphi_2] = [\varphi_1] \cap [\varphi_2]$$
  

$$[EX\varphi] = \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\}$$
  

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

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

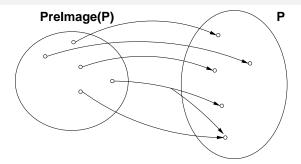
イロト イポト イヨト イヨト

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

### Definition of $[\varphi]$

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

### Recursive definition of $[\varphi]$



•  $[\mathsf{EX}\varphi] = \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\}$ 

• [**EX** $\varphi$ ] is said to be the Pre-image of [ $\varphi$ ] (*Preimage*([ $\varphi$ ]))

Key step of every CTL M.C. operation

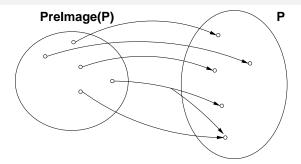
#### Note

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

Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 20/72



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

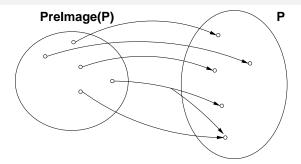
#### Note

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

Roberto Sebastiani

Ch. 04: CTL Model Checking

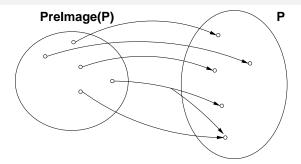
Monday 18<sup>th</sup> May, 2020 20/72



- $[\mathsf{EX}\varphi] = \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\}$
- [**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:  $X \subseteq X' \Longrightarrow Preimage(X) \subseteq Preimage(X')$ 



- $[\mathsf{EX}\varphi] = \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\}$
- [**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:  $X \subseteq X' \Longrightarrow Preimage(X) \subseteq Preimage(X')$ 

Roberto Sebastiani

Ch. 04: CTL Model Checking

### • $\nu Z.([\beta] \cap [\mathbf{E}XZ])$ : greatest fixed point of the function $F_{\beta}: 2^{S} \mapsto 2^{S}$ , s.t. $F_{\beta}([\varphi]) = ([\beta] \cap Preimage([\varphi]))$ $= ([\beta] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$

•  $F_{\beta}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta}(a) \subseteq F_{\beta}(a')$ 

- (Tarski's theorem):  $\nu x.F_{\beta}(x)$  always exists
- (Kleene's theorem): *νx*.*F<sub>β</sub>(x)* can be computed as the limit
   *S* ⊇ *F<sub>β</sub>(S)* ⊇ *F<sub>β</sub>(F<sub>β</sub>(S))* ⊇ ..., in a finite number of steps.

## Theorem (Clarke & Emerson) $[\mathbf{EG}\beta] = \nu Z.([\beta] \cap [\mathbf{EX}Z])$

### • $\nu Z.([\beta] \cap [\mathbf{E}XZ])$ : greatest fixed point of the function $F_{\beta}: 2^{S} \mapsto 2^{S}$ , s.t. $F_{\beta}([\varphi]) = ([\beta] \cap Preimage([\varphi]))$ $= ([\beta] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$

### • $F_{\beta}$ Monotonic: $a \subseteq a' \Longrightarrow F_{\beta}(a) \subseteq F_{\beta}(a')$

- (Tarski's theorem):  $\nu x.F_{\beta}(x)$  always exists
- (Kleene's theorem): νx.F<sub>β</sub>(x) can be computed as the limit S ⊇ F<sub>β</sub>(S) ⊇ F<sub>β</sub>(F<sub>β</sub>(S)) ⊇ ..., in a finite number of steps.

## Theorem (Clarke & Emerson) $[\mathbf{EG}\beta] = \nu Z.([\beta] \cap [\mathbf{EX}Z])$

### • $\nu Z.([\beta] \cap [\mathbf{E}XZ])$ : greatest fixed point of the function $F_{\beta}: 2^{S} \longmapsto 2^{S}$ , s.t. $F_{\beta}([\varphi]) = ([\beta] \cap Preimage([\varphi]))$ $= ([\beta] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$

### • $F_{\beta}$ Monotonic: $a \subseteq a' \Longrightarrow F_{\beta}(a) \subseteq F_{\beta}(a')$

• (Tarski's theorem):  $\nu x.F_{\beta}(x)$  always exists

(Kleene's theorem): νx.F<sub>β</sub>(x) can be computed as the limit
 S ⊇ F<sub>β</sub>(S) ⊇ F<sub>β</sub>(F<sub>β</sub>(S)) ⊇ ..., in a finite number of steps.

## Theorem (Clarke & Emerson) [EG $\beta$ ] = $\nu Z$ .( [ $\beta$ ] $\cap$ [EXZ] )

### • $\nu Z.([\beta] \cap [\mathbf{E}XZ])$ : greatest fixed point of the function $F_{\beta}: 2^{S} \longmapsto 2^{S}$ , s.t. $F_{\beta}([\varphi]) = ([\beta] \cap Preimage([\varphi]))$ $= ([\beta] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$

- $F_{\beta}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta}(a) \subseteq F_{\beta}(a')$ 
  - (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) $[\mathbf{EG}\beta] = \nu Z.([\beta] \cap [\mathbf{EX}Z])$

### • $\nu Z.([\beta] \cap [\mathbf{E}XZ])$ : greatest fixed point of the function $F_{\beta}: 2^{S} \longmapsto 2^{S}$ , s.t. $F_{\beta}([\varphi]) = ([\beta] \cap Preimage([\varphi]))$ $= ([\beta] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$

- $F_{\beta}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta}(a) \subseteq F_{\beta}(a')$ 
  - (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) $[\mathbf{EG}\beta] = \nu Z.([\beta] \cap [\mathbf{EX}Z])$

# Case EG [cont.]

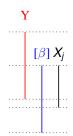
- We can compute X := [EGβ] inductively as follows:

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

- Noticing that  $X_1 = [\beta]$  and  $X_{j+1} \subseteq X_j$  for every  $j \ge 0$ , and that  $([\beta] \cap Y) \subseteq X_j \subseteq [\beta] \Longrightarrow ([\beta] \cap Y) = (X_j \cap Y)$ , we can use instead the following inductive schema:
  - $X_1 := [\beta]$ •  $X_{j+1} := X_j \cap Preimage(X_j)$

# Case EG [cont.]

- We can compute X := [EGβ] inductively as follows:
  - $\begin{array}{rcl} X_0 & := & S \\ X_1 & := & F_{\beta}(S) & = & [\beta] \\ X_2 & := & F_{\beta}(F_{\beta}(S)) & = & [\beta] \cap \textit{Preimage}(X_1) \\ & \ddots & \end{array}$
  - $X_{j+1} := F_{\beta}^{j+1}(S) = [\beta] \cap Preimage(X_j)$
- Noticing that  $X_1 = [\beta]$  and  $X_{j+1} \subseteq X_j$  for every  $j \ge 0$ , and that  $([\beta] \cap Y) \subseteq X_j \subseteq [\beta] \Longrightarrow ([\beta] \cap Y) = (X_j \cap Y)$ , we can use instead the following inductive schema:
  - $X_1 := [\beta]$ •  $X_{j+1} := X_j \cap Preimage(X_j)$



イロト 不得 トイヨト イヨト 二日

- $\mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}XZ]))$ : least fixed point of the function  $F_{\beta_1,\beta_2}: 2^S \mapsto 2^S$ , s.t.  $F_{\beta_1,\beta_2}([\varphi]) = [\beta_2] \cup ([\beta_1] \cap Preimage([\varphi]))$  $= [\beta_2] \cup ([\beta_1] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$
- $F_{\beta_1,\beta_2}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta_1,\beta_2}(a) \subseteq F_{\beta_1,\beta_2}(a')$ 
  - (Tarski's theorem): μx.F<sub>β1,β2</sub>(x) always exists
  - (Kleene's theorem): μx.F<sub>β1,β2</sub>(x) can be computed as the limit
     Ø ⊆ F<sub>β1,β2</sub>(Ø) ⊆ F<sub>β1,β2</sub>(F<sub>β1,β2</sub>(Ø)) ⊆ ..., in a finite number of steps.

#### Theorem (Clarke & Emerson) $[\mathbf{E}(\beta_1 \mathbf{U}\beta_2)] = \mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}\mathbf{X}Z]))$

- $\mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}XZ]))$ : least fixed point of the function  $F_{\beta_1,\beta_2}: 2^S \mapsto 2^S$ , s.t.  $F_{\beta_1,\beta_2}([\varphi]) = [\beta_2] \cup ([\beta_1] \cap Preimage([\varphi]))$  $= [\beta_2] \cup ([\beta_1] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$
- $F_{\beta_1,\beta_2}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta_1,\beta_2}(a) \subseteq F_{\beta_1,\beta_2}(a')$ 
  - (Tarski's theorem): μx. F<sub>β1,β2</sub>(x) always exists
  - (Kleene's theorem):  $\mu x.F_{\beta_1,\beta_2}(x)$  can be computed as the limit  $\emptyset \subseteq F_{\beta_1,\beta_2}(\emptyset) \subseteq F_{\beta_1,\beta_2}(F_{\beta_1,\beta_2}(\emptyset)) \subseteq \ldots$ , in a finite number of steps.

Theorem (Clarke & Emerson)  $[\mathbf{E}(\beta_1 \mathbf{U}\beta_2)] = \mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}\mathbf{X}\mathbf{Z}]))$ 

- $\mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}XZ]))$ : least fixed point of the function  $F_{\beta_1,\beta_2}: 2^S \mapsto 2^S$ , s.t.  $F_{\beta_1,\beta_2}([\varphi]) = [\beta_2] \cup ([\beta_1] \cap Preimage([\varphi]))$  $= [\beta_2] \cup ([\beta_1] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$
- $F_{\beta_1,\beta_2}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta_1,\beta_2}(a) \subseteq F_{\beta_1,\beta_2}(a')$ 
  - (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  $\emptyset \subseteq F_{\beta_1,\beta_2}(\emptyset) \subseteq F_{\beta_1,\beta_2}(F_{\beta_1,\beta_2}(\emptyset)) \subseteq \ldots$ , in a finite number of steps.

Theorem (Clarke & Emerson)  $[\mathbf{E}(\beta_1 \mathbf{U}\beta_2)] = \mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}\mathbf{X}\mathbf{Z}]))$ 

- $\mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}XZ]))$ : least fixed point of the function  $F_{\beta_1,\beta_2}: 2^S \mapsto 2^S$ , s.t.  $F_{\beta_1,\beta_2}([\varphi]) = [\beta_2] \cup ([\beta_1] \cap Preimage([\varphi]))$  $= [\beta_2] \cup ([\beta_1] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$
- $F_{\beta_1,\beta_2}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta_1,\beta_2}(a) \subseteq F_{\beta_1,\beta_2}(a')$ 
  - (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  $\emptyset \subseteq F_{\beta_1,\beta_2}(\emptyset) \subseteq F_{\beta_1,\beta_2}(F_{\beta_1,\beta_2}(\emptyset)) \subseteq \ldots$ , in a finite number of steps.

# Theorem (Clarke & Emerson) $[\mathbf{E}(\beta_1\mathbf{U}\beta_2)] = \mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}\mathbf{X}\mathbf{Z}]))$

- $\mu Z.([\beta_2] \cup ([\beta_1] \cap [\mathbf{E}XZ]))$ : least fixed point of the function  $F_{\beta_1,\beta_2}: 2^S \mapsto 2^S$ , s.t.  $F_{\beta_1,\beta_2}([\varphi]) = [\beta_2] \cup ([\beta_1] \cap Preimage([\varphi]))$  $= [\beta_2] \cup ([\beta_1] \cap \{s \mid \exists s' \in [\varphi] \ s.t. \ \langle s, s' \rangle \in R\})$
- $F_{\beta_1,\beta_2}$  Monotonic:  $a \subseteq a' \Longrightarrow F_{\beta_1,\beta_2}(a) \subseteq F_{\beta_1,\beta_2}(a')$ 
  - (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  $\emptyset \subseteq F_{\beta_1,\beta_2}(\emptyset) \subseteq F_{\beta_1,\beta_2}(\mathcal{F}_{\beta_1,\beta_2}(\emptyset)) \subseteq \ldots$ , in a finite number of steps.

Theorem (Clarke & Emerson)

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

# Case EU [cont.]

- We can compute X := [E(β<sub>1</sub>Uβ<sub>2</sub>)] inductively as follows:

  - $X_{j+1} := F_{\beta_1,\beta_2}^{j+1}(\emptyset)) = [\beta_2] \cup ([\beta_1] \cap Preimage(X_j))$
- Noticing that  $X_1 = [\beta_2]$  and  $X_{j+1} \supseteq X_j$  for every  $j \ge 0$ , and that  $([\beta_2] \cup Y) \supseteq X_j \supseteq [\beta_2] \Longrightarrow ([\beta_2] \cup Y) = (X_j \cup Y)$ , we can use instead the following inductive schema:

• 
$$X_1 := [\beta_2]$$
  
•  $X_{j+1} := X_j \cup ([\beta_1] \cap Preimage(X_j))$ 

# Case EU [cont.]

- We can compute X := [E(β<sub>1</sub>Uβ<sub>2</sub>)] inductively as follows:

  - $X_{j+1} := F_{\beta_1,\beta_2}^{j+1}(\emptyset)) = [\beta_2] \cup ([\beta_1] \cap Preimage(X_j))$
- Noticing that  $X_1 = [\beta_2]$  and  $X_{j+1} \supseteq X_j$  for every  $j \ge 0$ , and that  $([\beta_2] \cup Y) \supseteq X_j \supseteq [\beta_2] \Longrightarrow ([\beta_2] \cup Y) = (X_j \cup Y)$ , we can use instead the following inductive schema:

• 
$$X_1 := [\beta_2]$$
  
•  $X_{j+1} := X_j \cup ([\beta_1] \cap Preimage(X_j))$ 

イロト イポト イヨト イヨト

··[β<sub>2</sub>]

# A relevant subcase: EF

#### • $\mathbf{EF}\beta = \mathbf{E}(\top \mathbf{U}\beta)$

- $[\top] = S \Longrightarrow [\top] \cap Preimage(X_j) = Preimage(X_j)$
- We can compute  $X := [\mathbf{EF}\beta]$  inductively as follows:
  - X<sub>1</sub> := [β]
     X<sub>i+1</sub> := X<sub>i</sub> ∪ Preimage(X<sub>i</sub>)
  - $X_{j+1} := X_j \cup Preimage(X_j)$

# A relevant subcase: EF

#### • $\mathbf{EF}\beta = \mathbf{E}(\top \mathbf{U}\beta)$

#### • $[\top] = S \Longrightarrow [\top] \cap Preimage(X_j) = Preimage(X_j)$

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

• 
$$X_1 := [\beta]$$
  
•  $X_{i+1} := X_i \cup Preimage(X_i)$ 

# A relevant subcase: EF

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

#### Outline

- CTL Model Checking: general ideas
- 2 CTL Model Checking: a simple example
- 3 Some theoretical issues
  - CTL Model Checking: algorithms
- 5 CTL Model Checking: some examples
- 6 A relevant subcase: invariants
- Zercises

・ 同 ト ・ ヨ ト ・ ヨ ト

#### Assume φ written in terms of ¬, ∧, EX, EU, EG

A general M.C. algorithm (fix-point):

1. for every  $\varphi_i \in Sub(\varphi)$ , find  $[\varphi_i]$ 

2. Check if  $I \subseteq [\varphi]$ 

• Subformulas  $Sub(\varphi)$  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: appy recursively the tableaux rules, until a fixpoint is reached

- Assume φ written in terms of ¬, ∧, EX, EU, EG
- A general M.C. algorithm (fix-point):
  - 1. for every  $\varphi_i \in Sub(\varphi)$ , find  $[\varphi_i]$
  - 2. Check if  $I \subseteq [\varphi]$
- Subformulas  $Sub(\varphi)$  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: appy recursively the tableaux rules, until a fixpoint is reached

- Assume φ written in terms of ¬, ∧, EX, EU, EG
- A general M.C. algorithm (fix-point):
  - 1. for every  $\varphi_i \in Sub(\varphi)$ , find  $[\varphi_i]$
  - 2. Check if  $I \subseteq [\varphi]$
- Subformulas  $Sub(\varphi)$  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: appy recursively the tableaux rules, until a fixpoint is reached

- Assume φ written in terms of ¬, ∧, EX, EU, EG
- A general M.C. algorithm (fix-point):
  - 1. for every  $\varphi_i \in Sub(\varphi)$ , find  $[\varphi_i]$
  - 2. Check if  $I \subseteq [\varphi]$
- Subformulas  $Sub(\varphi)$  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: appy recursively the tableaux rules, until a fixpoint is reached

- Assume φ written in terms of ¬, ∧, EX, EU, EG
- A general M.C. algorithm (fix-point):
  - 1. for every  $\varphi_i \in Sub(\varphi)$ , find  $[\varphi_i]$
  - 2. Check if  $I \subseteq [\varphi]$
- Subformulas  $Sub(\varphi)$  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: appy recursively the tableaux rules, until a fixpoint is reached

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

- Assume φ written in terms of ¬, ∧, EX, EU, EG
- A general M.C. algorithm (fix-point):
  - 1. for every  $\varphi_i \in Sub(\varphi)$ , find  $[\varphi_i]$
  - 2. Check if  $I \subseteq [\varphi]$
- Subformulas  $Sub(\varphi)$  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: appy recursively the tableaux rules, until a fixpoint is reached

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

A D K A D K A D K A D K B K

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

# General M.C. Procedure

```
state set Check(CTL formula \beta) {
    case \beta of
    true:
                    return S:
    false:
                    return {};
                    return {s \mid p \in L(s)};
    p:
    \neg \beta_1:
                    return S / Check(\beta_1):
    \beta_1 \wedge \beta_2:
                    return Check(\beta_1) \cap Check(\beta_2);
    \mathbf{EX}\beta_1:
                    return Prelmage(Check(\beta_1));
    EG\beta_1:
                    return Check EG(Check(\beta_1));
    \mathbf{E}(\beta_1 \mathbf{U} \beta_2):
                    return Check EU(Check(\beta_1),Check(\beta_2));
```

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

### PreImage

```
state_set PreImage(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;
```

# Check\_EG

```
\begin{array}{l} \textbf{state\_set Check\_EG(state\_set [\beta]) } \\ X' := [\beta]; \ j := 1; \\ \textbf{repeat} \\ X' := X'; \ j := j + 1; \\ X' := X \cap PreImage(X); \\ \textbf{until } (X' = X); \\ \textbf{return } X; \\ \end{array}
```

# Check\_EU

```
\begin{array}{l} \textbf{state\_set Check\_EU(state\_set [\beta_1], [\beta_2]) } \\ X' := [\beta_2]; \ j := 1; \\ \textbf{repeat} \\ X := X'; \ j := j + 1; \\ X' := X \cup ([\beta_1] \cap \textit{PreImage}(X)); \\ \textbf{until } (X' = X); \\ \textbf{return } X; \\ \end{array}
```

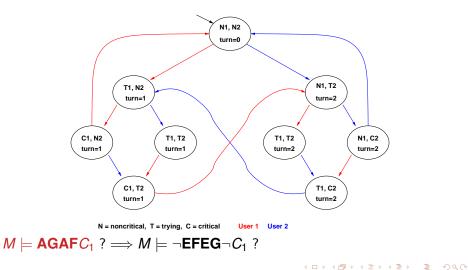
# A relevant subcase: Check\_EF

```
\begin{array}{l} \textbf{state\_set Check\_EF(state\_set [\beta]) } \{ \\ X' := [\beta]; \ j := 1; \\ \textbf{repeat} \\ X' := X'; \ j := j + 1; \\ X' := X \cup PreImage(X); \\ \textbf{until } (X' = X); \\ \textbf{return } X; \\ \} \end{array}
```

#### Outline

- CTL Model Checking: general ideas
- 2 CTL Model Checking: a simple example
- 3 Some theoretical issues
- 4) CTL Model Checking: algorithms
- 5 CTL Model Checking: some examples
  - 6 A relevant subcase: invariants
  - 7 Exercises

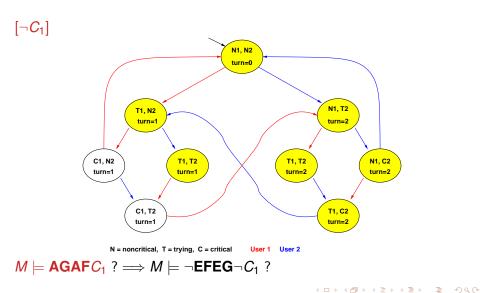
・ 同 ト ・ ヨ ト ・ ヨ ト



Roberto Sebastiani

Ch. 04: CTL Model Checking

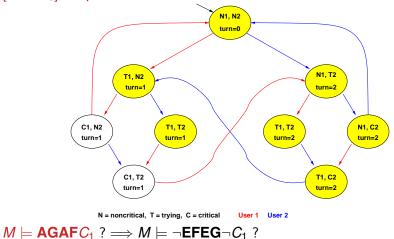
Monday 18<sup>th</sup> May, 2020 34/72



Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 35/72

[**EG**¬*C*<sub>1</sub>], step 0:



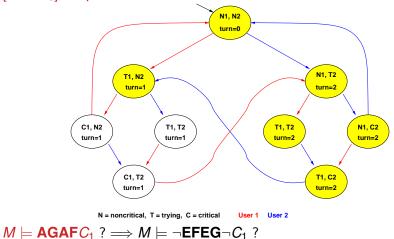
Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 36/72

イロト イポト イヨト イヨト 一日

[**EG**¬*C*<sub>1</sub>], step 1:

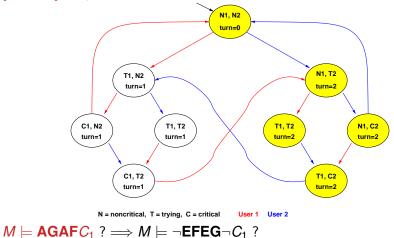


Roberto Sebastiani

Ch. 04: CTL Model Checking

・ ロ ト 4 日 ト 4 王 ト 4 王 ト 王 今 Q ペ
Monday 18<sup>th</sup> May, 2020 37/72

[**EG**¬*C*<sub>1</sub>], step 2:

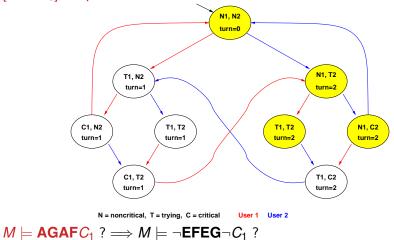


Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 38/72

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

[**EG** $\neg$ *C*<sub>1</sub>], step 3:

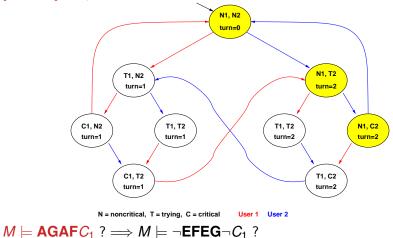


Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 39/72

<ロト <回 > < 三 > < 三 > - 三 =

[**EG** $\neg$ *C*<sub>1</sub>], step 4:



Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 40/72

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

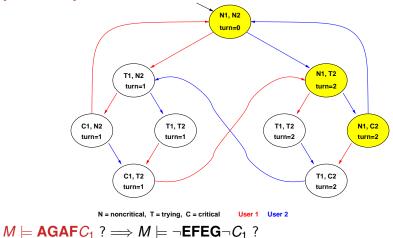
#### [**EG** $\neg$ *C*<sub>1</sub>], FIXPOINT! N1, N2 turn=0 N1. T2 T1. N2 turn=1 turn=2 C1, N2 T1, T2 T1, T2 N1, C2 turn=1 turn=1 turn=2 turn=2 C1, T2 T1, C2 turn=2 turn=1 N = noncritical, T = trying, C = critical User 1 User 2 $M \models \mathsf{AGAFC}_1 ? \Longrightarrow M \models \neg \mathsf{EFEG} \neg C_1 ?$

Roberto Sebastiani

Ch. 04: CTL Model Checking

・ ロ ト 4 日 ト 4 王 ト 4 王 ト 王 今 Q ペ
Monday 18<sup>th</sup> May, 2020 41/72

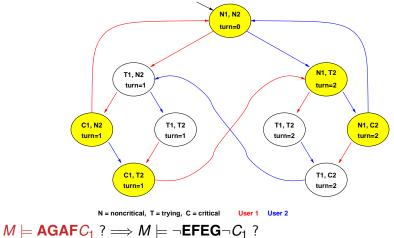
#### $[\mathbf{EFEG} \neg C_1], \text{STEP 0}$



Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 42/72

# [**EFEG**¬*C*<sub>1</sub>], STEP 1

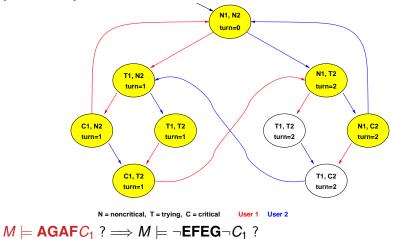


Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 43/72

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

#### $[\mathbf{EFEG} \neg C_1], \text{STEP 2}$



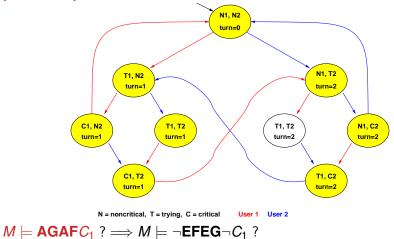
Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 44/72

イロト イポト イヨト イヨト 二日

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



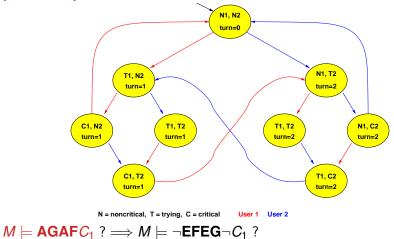
Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 45/72

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

#### $[\mathbf{EFEG} \neg C_1], \text{STEP 4}$



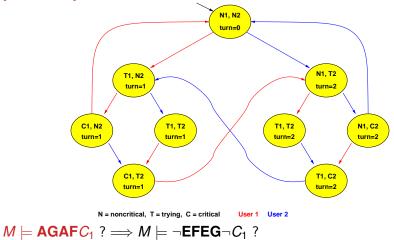
Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 46/72

イロト イポト イヨト イヨト 二日

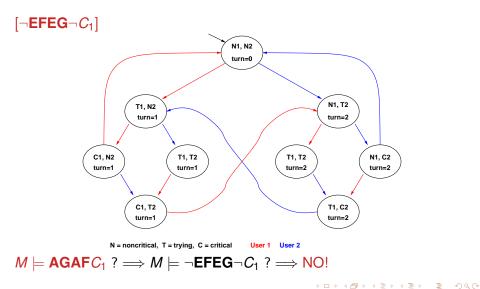
#### [**EFEG** $\neg$ *C*<sub>1</sub>], FIXPOINT!



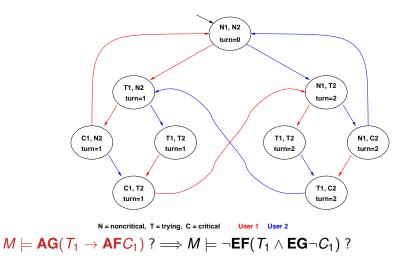
Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 47/72

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

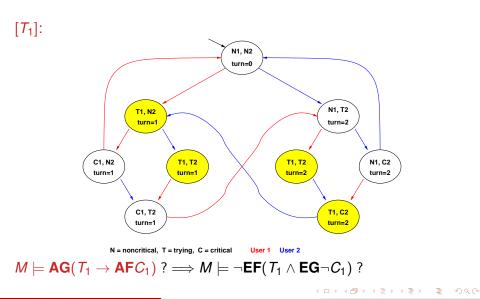


Ch. 04: CTL Model Checking



Roberto Sebastiani

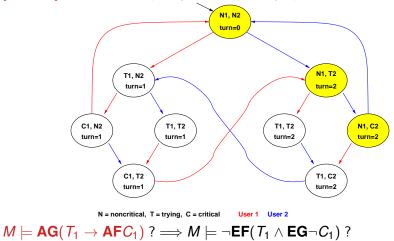
Ch. 04: CTL Model Checking



Ch. 04: CTL Model Checking

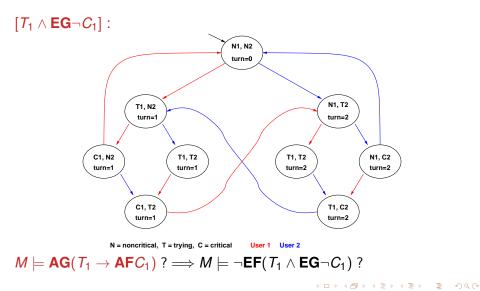
Monday 18<sup>th</sup> May, 2020

[**EG** $\neg$ *C*<sub>1</sub>], STEPS 0-4: (see previous example)



Ch. 04: CTL Model Checking

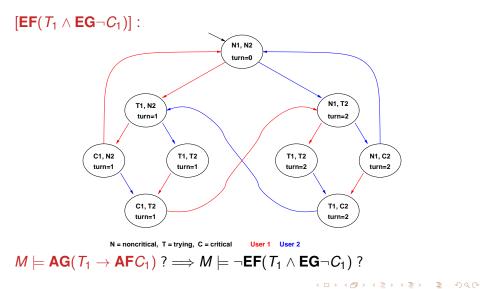
< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □



Roberto Sebastiani

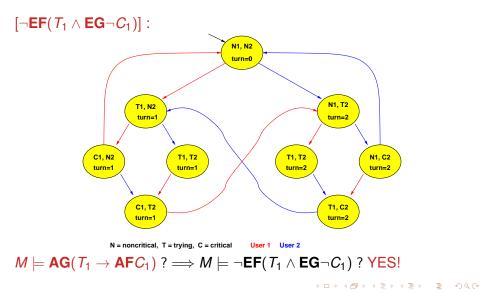
Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020



Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020



Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020

#### CTL Model Checking: some examples



# The property verified is...

Roberto Sebastiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020

#### Homework

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

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 56/72

э

イロト イロト イヨト イヨト

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

- 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
  - $\Longrightarrow \textit{O}(|\textit{M}| \cdot |arphi|)$  steps
- Step 2: check  $I \subseteq [\varphi]$ : O(|M|)
- $\Longrightarrow O(|M| \cdot |\varphi|)$

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

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

- 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
  - $\Longrightarrow O(|M| \cdot |arphi|)$  steps
- Step 2: check  $I \subseteq [\varphi]$ : O(|M|)
- $\implies O(|M| \cdot |\varphi|)$

#### Outline

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

Ch. 04: CTL Model Checking

< □ > < ⊇ > < ⊇ >
 Monday 18<sup>th</sup> May, 2020

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

 $Y' := Y \cup PreImage(Y)$ 

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 computations:

 $Y' := Y \cup PreImage(Y)$ 

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

Roberto Sebastiani

Ch. 04: CTL Model Checking

- Invariant properties have the form AG p (e.g., AG¬bad)
- Checking invariants is the negation of a reachability problem:
  - is there a reachable state that is also a bad state?

 $(\mathbf{AG} \neg bad = \neg \mathbf{EF} bad)$ 

Standard M.C. algorithm reasons backward from the bad by

• Better algorithm: reasons backward from the bad by iteratively

**Roberto Sebastiani** 

Ch. 04: CTL Model Checking

- 3

- Invariant properties have the form **AG p** (e.g., **AG**¬*bad*)
- Checking invariants is the negation of a reachability problem:
  - is there a reachable state that is also a bad state?

 $(\mathbf{AG} \neg bad = \neg \mathbf{EF} bad)$ 

• Standard M.C. algorithm reasons backward from the bad by iteratively applying PreImage computations:

 $Y' := Y \cup PreImage(Y)$ 

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 computations:

 $Y' := Y \cup PreImage(Y)$ 

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

Roberto Sebastiani

Ch. 04: CTL Model Checking

・ イヨト イヨト ヨ つへの Monday 18<sup>th</sup> May, 2020 59/72

- Invariant properties have the form **AG p** (e.g., **AG**¬*bad*)
- Checking invariants is the negation of a reachability problem:
  - is there a reachable state that is also a bad state?

 $(\mathbf{AG} \neg bad = \neg \mathbf{EF} bad)$ 

• Standard M.C. algorithm reasons backward from the bad by iteratively applying PreImage computations:

 $Y' := Y \cup PreImage(Y)$ 

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 computations:

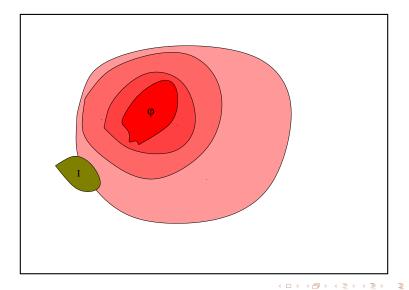
 $Y' := Y \cup PreImage(Y)$ 

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

Roberto Sebastiani

Ch. 04: CTL Model Checking

## Model Checking of Invariants [cont.]



Roberto Sebastiani

Monday 18<sup>th</sup> May, 2020

#### 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:

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

• Simplest form: compute the set of reachable states.

Monday 18<sup>th</sup> May, 2020 61/72

A B A B A B A
 A B A
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 B
 A
 A
 A
 A
 A

#### 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:

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

• Simplest form: compute the set of reachable states.

イロト イポト イヨト イヨト

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:

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

• Simplest form: compute the set of reachable states.

Ro	ber	to :	Se	bast	tiani

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 61/72

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:

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

• Simplest form: compute the set of reachable states.

Monday 18<sup>th</sup> May, 2020 61/72

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:

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

• Simplest form: compute the set of reachable states.

Ch. 04: CTL Model Checking

Monday 18<sup>th</sup> May, 2020 61/72

Alternative algorithm (often more efficient): forward checking

- Compute the set of bad states [bad]
- Compute the set of initial states /
- 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:

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

• 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';  $Y' := Y \cup Image(Y);$ } return Y; }

Y=reachable

イロト イポト イヨト イヨト

#### Computing Reachable states: advanced

```
State_Set Compute_reachable() {

Y := F := I; j := 1;

while (F \neq \emptyset)

j := j + 1;

F := Image(F) \setminus Y;

Y := Y \cup F;

}

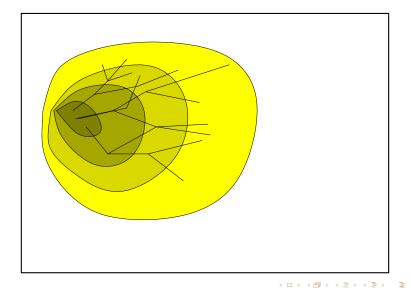
return Y;

}
```

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

イロト イポト イヨト イヨト

#### Computing Reachable states [cont.]



Roberto Sebastiani

Monday 18<sup>th</sup> May, 2020 64/72

#### Checking of Invariant Properties: basic

```
bool Forward Check EF(State Set BAD) {
    Y := I; Y' := \emptyset; i := 1;
   while (Y' \neq Y) and (Y' \cap BAD) = \emptyset
        i := i + 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;

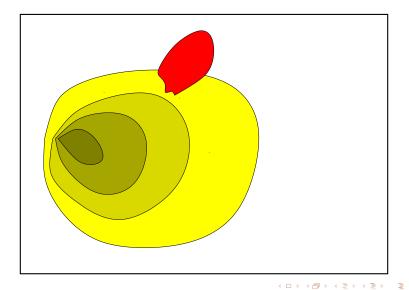
く 同 ト く ヨ ト く ヨ ト 一

#### Checking of Invariant Properties: advanced

```
bool Forward Check EF(State Set BAD) {
    Y := F := I; i := 1;
   while (F \neq \emptyset) and (F \cap BAD) = \emptyset
        i := i + 1;
         F := Image(F) \setminus Y;
         Y := Y \cup F:
    if (F \cap BAD) \neq \emptyset // counter-example
         return true
    else
                         // fixpoint reached
         return false
```

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

#### Checking of Invariant Properties [cont.]



Monday 18<sup>th</sup> May, 2020 67/72

#### 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 ∩ 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],..., *t*[*n*] is our counterexample

イロト イポト イヨト イヨト

- 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],..., *t*[*n*] is our counterexample

- 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],..., *t*[*n*] is our counterexample

- 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
- $t[0], t[1], \ldots, t[n]$  is our counterexample

- 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],..., *t*[*n*] is our counterexample

イロト 不得 トイヨト イヨト 二日

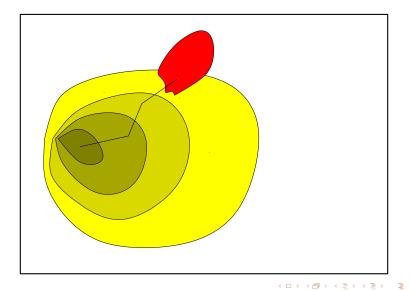
- if layer *n* intersects with the bad states, then the property is violated
- a counterexample can be reconstructed proceeding backwards
  - (i) select any state of  $BAD \cap F[n]$  (we know it is satisfiable), call it t[n]
  - (ii) compute *Preimage*(*t*[*n*]), i.e. the states that can result in *t*[*n*] 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], ..., *t*[*n*] is our counterexample

Ch. 04: CTL Model Checking

<ロ > < 合 > < 言 > < 言 > 三 Monday 18<sup>th</sup> May, 2020

68/72

- 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], ..., *t*[*n*] is our counterexample



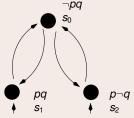
Monday 18<sup>th</sup> May, 2020 69/72

#### Outline

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

・ 同 ト ・ ヨ ト ・ ヨ ト

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

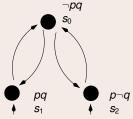


(a) Rewrite  $\varphi$  into an equivalent formula  $\varphi'$  expressed in terms of EX, EG, EU/EF only.

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

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

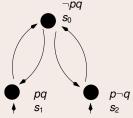
Consider the Kripke Model *M* below, and the CTL property  $\varphi \stackrel{\text{def}}{=} \mathbf{AG}((p \land q) \rightarrow \mathbf{EG}q)$ .



- (a) Rewrite  $\varphi$  into an equivalent formula  $\varphi'$  expressed in terms of **EX**, **EG**, **EU**/**EF** only. [Solution:  $\varphi' = \neg \mathbf{EF} \neg ((\neg p \lor \neg q) \lor \mathbf{EG}q) = \neg \mathbf{EF}((p \land q) \land \neg \mathbf{EG}q)$ ]
- (b) Compute bottom-up the denotations of all subformulas of  $\varphi'$ . (Ex:  $[p] = \{s_1, s_2\}$ )

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

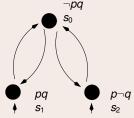
Consider the Kripke Model *M* below, and the CTL property  $\varphi \stackrel{\text{def}}{=} \mathbf{AG}((p \land q) \to \mathbf{EG}q)$ .



- (a) Rewrite  $\varphi$  into an equivalent formula  $\varphi'$  expressed in terms of **EX**, **EG**, **EU**/**EF** only. [Solution:  $\varphi' = \neg \mathbf{EF} \neg ((\neg p \lor \neg q) \lor \mathbf{EG}q) = \neg \mathbf{EF}((p \land q) \land \neg \mathbf{EG}q)$ ]
- (b) Compute bottom-up the denotations of all subformulas of  $\varphi'$ . (Ex:  $[p] = \{s_1, s_2\}$ ) [Solution:

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

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



- (a) Rewrite  $\varphi$  into an equivalent formula  $\varphi'$  expressed in terms of **EX**, **EG**, **EU**/**EF** only. [Solution:  $\varphi' = \neg \mathbf{EF} \neg ((\neg p \lor \neg q) \lor \mathbf{EG}q) = \neg \mathbf{EF}((p \land q) \land \neg \mathbf{EG}q)$ ]
- (b) Compute bottom-up the denotations of all subformulas of  $\varphi'$ . (Ex:  $[p] = \{s_1, s_2\}$ ) [Solution:

Consider the Kripke Model *M* below, and the CTL property  $AG(AFp \rightarrow AFq)$ .



(a) Rewrite  $\varphi$  into an equivalent formula  $\varphi' \exp^{\hat{s}_i}$  espective of EX, EG, EU/EF only.

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

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

Consider the Kripke Model *M* below, and the CTL property  $AG(AFp \rightarrow AFq)$ .



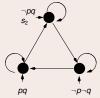
(a) Rewrite  $\varphi$  into an equivalent formula  $\varphi'$  expressed in terms of **EX**, **EG**, **EU**/**EF** only. [Solution:

 $\varphi' = \mathsf{AG}(\mathsf{AF}\rho \to \mathsf{AF}q) = \neg \mathsf{EF} \neg (\neg \mathsf{EG} \neg \rho \to \neg \mathsf{EG} \neg q) = \neg \mathsf{EF}(\neg \mathsf{EG} \neg \rho \land \mathsf{EG} \neg q) ]$ 

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

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

Consider the Kripke Model *M* below, and the CTL property  $AG(AFp \rightarrow AFq)$ .



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

 $\varphi' = \mathsf{AG}(\mathsf{AF}p \to \mathsf{AF}q) = \neg \mathsf{EF} \neg (\neg \mathsf{EG} \neg p \to \neg \mathsf{EG} \neg q) = \neg \mathsf{EF}(\neg \mathsf{EG} \neg p \land \mathsf{EG} \neg q) ]$ 

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

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

72/72

Consider the Kripke Model *M* below, and the CTL property  $AG(AFp \rightarrow AFq)$ .



(a) Rewrite  $\varphi$  into an equivalent formula  $\varphi' \exp^{i \theta}$  espectives of EX, EG, EU/EF only. [Solution:

 $\varphi' = \mathsf{AG}(\mathsf{AF}p \to \mathsf{AF}q) = \neg \mathsf{EF} \neg (\neg \mathsf{EG} \neg p \to \neg \mathsf{EG} \neg q) = \neg \mathsf{EF}(\neg \mathsf{EG} \neg p \land \mathsf{EG} \neg q) ]$ 

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

 $\begin{bmatrix} p \end{bmatrix} = \{s_0\} \quad [\neg q] = \{s_1\} \\ [\neg p] = \{s_1, s_2\} \quad [EG\neg q] = \{s_1\} \\ [EG\neg p] = \{s_1, s_2\} \quad [\neg EG\neg p \land EG\neg q] = \{\} \\ [\neg EG\neg p] = \{s_0\} \quad [EF(\neg EG\neg p \land EG\neg q)] = \{\} \\ [q] = \{s_0, s_2\} \quad [\neg EF(\neg EG\neg p \land EG\neg q)] = \{s_0, s_1, s_2\} \\ (c) \text{ As a consequence of point } (b), \text{ say whether } M \models \varphi \text{ or not.} \\ [Solution: Yes, \{s_0, s_1, s_2\} \subseteq [\varphi']. ] \\ \hline \text{Roberto Sebastiani} \qquad Ch. 04: CTL Model Checking \qquad Monday 18^{th} May, 2020 \qquad 72/72 \\ \hline \end{array}$