Algorithms for CTL

B. Srivathsan

Chennai Mathematical Institute

Model Checking and Systems Verification

January - April 2016

Module 1:

Adequate CTL formulae

Recap of CTL

State formulae

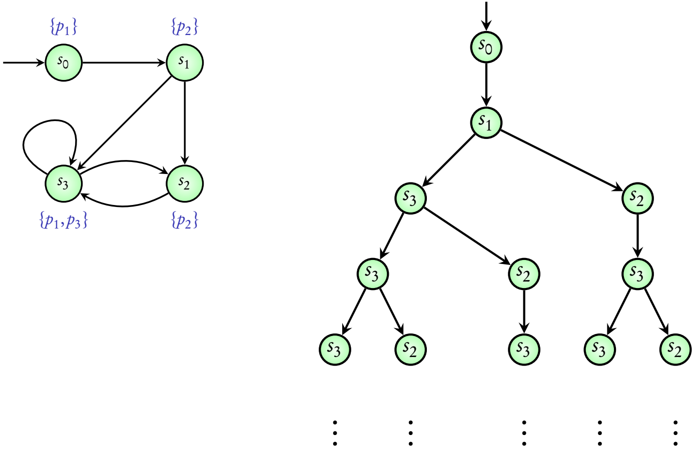
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid E\,\alpha \mid A\,\alpha$$

$p_i \in AP$ ϕ_1, ϕ_2 : State formulae α : Path formula

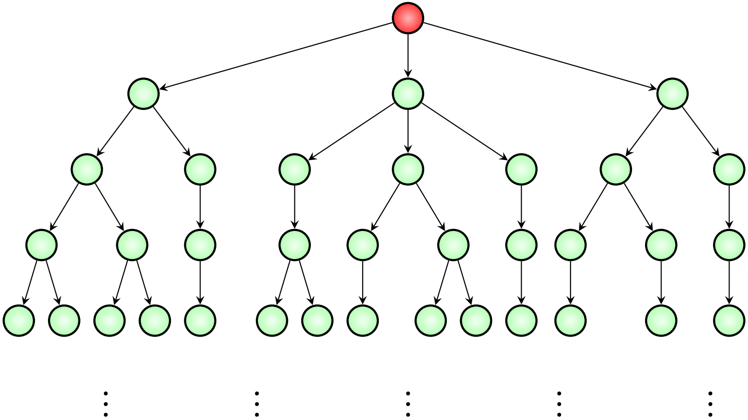
Path formulae

$$\alpha := X\,\phi_1 \mid \phi_1\,U\,\phi_2 \mid F\,\phi_1 \mid G\,\phi_1$$

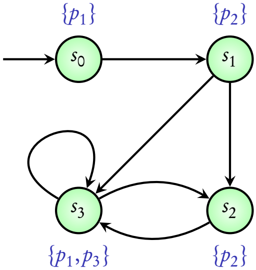
Transition system satisfies CTL state formula ϕ if its **computation tree** satisfies ϕ

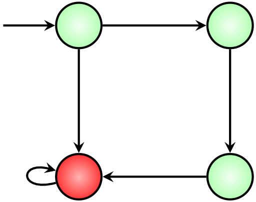


A **tree** satisfies CTL state formula ϕ if its **root** satisfies ϕ

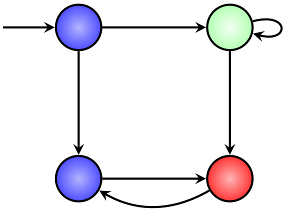


A **state** s in a transition system satisfies a CTL formula ϕ if the computation tree **starting at** s satisfies ϕ

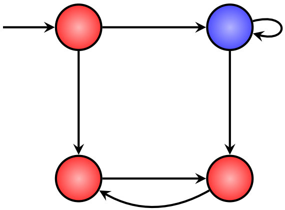




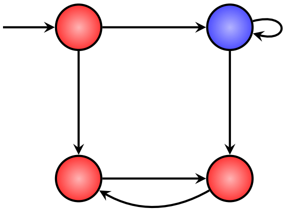
Above transition system satisfies **E X** *red*



Above transition system satisfies **E** *blue* **U** *red*



Above transition system satisfies **E G** *red*



Above transition system satisfies **E G** *red*

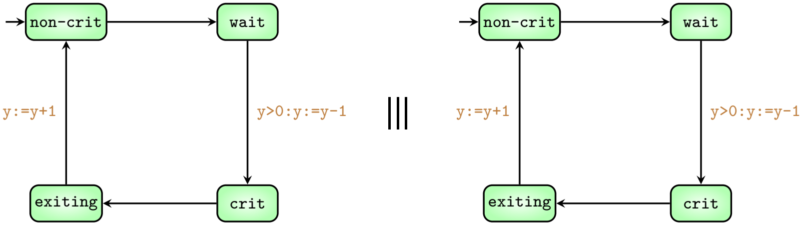
It does not satisfy **A F** *blue*

Mutual exclusion

Atomic propositions AP $= \{ p_1, p_2, p_3, p_4 \}$

p_1: `pr1.location=crit` p_2: `pr1.location=wait`

p_3: `pr2.location=crit` p_4: `pr2.location=wait`



Above system satisfies $\mathbf{A}\,\mathbf{G}\,\neg\,(p_1 \wedge p_3)$

Goal of this unit

Design an **algorithm**:

INPUT: A **transition system** M and a **CTL formula** ϕ

OUTPUT: Does M satisfy ϕ?

Goal of this unit

Design an **algorithm**:

INPUT: A **transition system** M and a **CTL formula** ϕ

OUTPUT: Does M satisfy ϕ?

We will answer a more general question:

Given M and ϕ, find all the states of M that satisfy ϕ

First step

State formulae

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid E\,\alpha \mid A\,\alpha$$

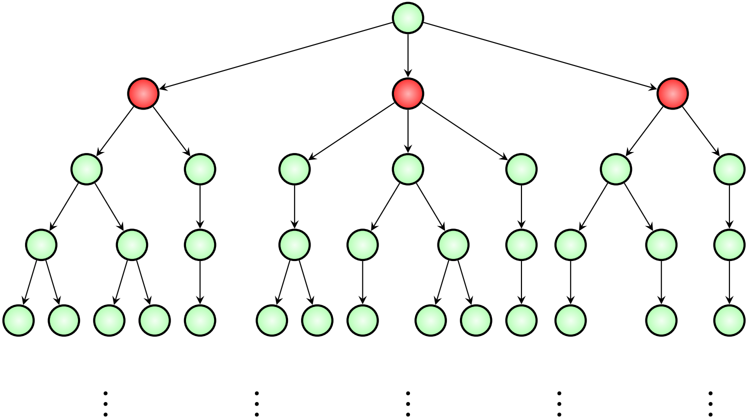
$p_i \in AP$ ϕ_1, ϕ_2 : State formulae α : Path formula

Path formulae

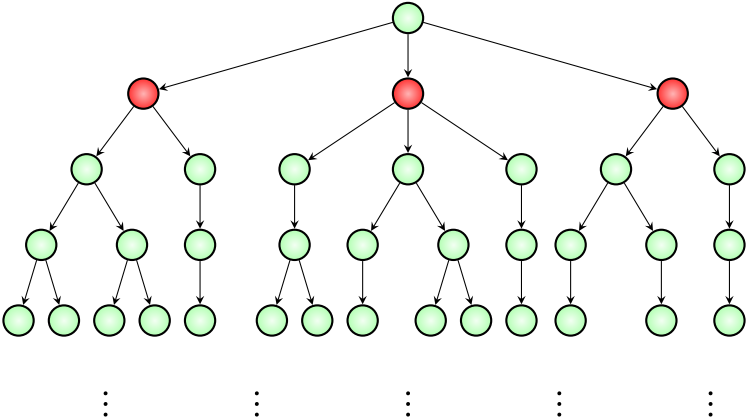
$$\alpha := X\,\phi_1 \mid \phi_1\,U\,\phi_2 \mid F\,\phi_1 \mid G\,\phi_1$$

Rewrite **A** in terms of **E**

A X (*red*) equivalent to ¬ **E X** (¬ *red*)



A X (*red*) equivalent to ¬ **E X** (¬ *red*)



$$\mathbf{A\,X}\,\phi \quad \equiv \quad \neg\,\mathbf{E\,X}\,\neg\,\phi$$

Can we rewrite $\mathbf{A}\,(\phi\ \mathbf{U}\ \psi)$ as $\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$?

Can we rewrite $\mathbf{A}\,(\phi\ \mathbf{U}\ \psi)$ as $\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$?

No: $\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$ is **not a CTL formula**

State formulae

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid E\,\alpha \mid A\,\alpha$$

$p_i \in AP$ ϕ_1, ϕ_2 : State formulae α : Path formula

Path formulae

$$\alpha := X\,\phi_1 \mid \phi_1\,U\,\phi_2 \mid F\,\phi_1 \mid G\,\phi_1$$

Can we rewrite $\mathbf{A}\,(\phi\ \mathbf{U}\ \psi)$ as $\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$?

No: $\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$ is **not a CTL formula**

State formulae

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid E\,\alpha \mid A\,\alpha$$

$p_i \in AP$ ϕ_1, ϕ_2 : State formulae α : Path formula

Path formulae

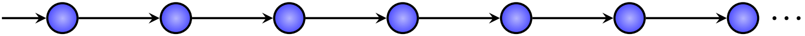
$$\alpha := X\,\phi_1 \mid \phi_1\,U\,\phi_2 \mid F\,\phi_1 \mid G\,\phi_1$$

CTL **does not allow** negation of path formula!

Coming next: Rewrite **A U** in terms of **E U** and **E G**

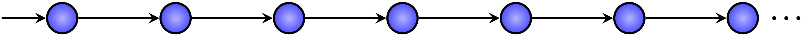
$\neg\,(\mathit{blue}\ \mathbf{U}\ \mathit{red})$

$\neg\,(\mathit{blue}\ \mathbf{U}\ \mathit{red})$



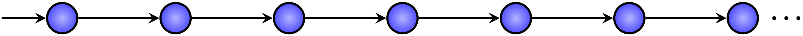
$\neg\,(\mathit{blue}\ \mathbf{U}\ \mathit{red})$

$\mathbf{G}\ \neg\,\mathit{red}$



$\neg\,(\mathit{blue}\ \mathbf{U}\ \mathit{red})$

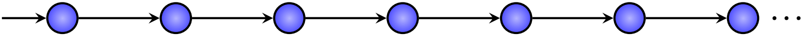
$\mathbf{G}\,\neg\,\mathit{red}$



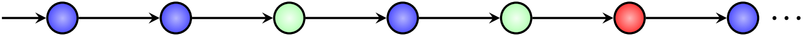
or

$\neg\,(\mathit{blue}\ \mathbf{U}\ \mathit{red})$

$\mathbf{G}\,\neg\,\mathit{red}$

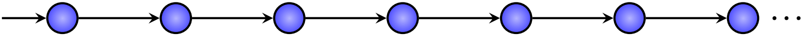


or

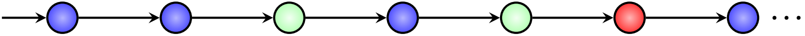


$\neg\,(\mathit{blue}\ \mathbf{U}\ \mathit{red})$

$\mathbf{G}\,\neg\,\mathit{red}$

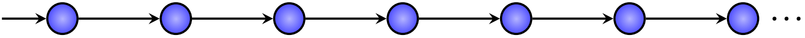


or $(\neg\,\mathit{red})\ \mathbf{U}\ (\neg\,\mathit{blue}\ \wedge\ \neg\,\mathit{red})$

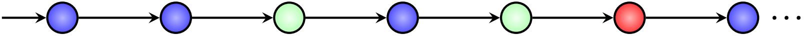


$\neg\,(\mathit{blue}\ \mathbf{U}\ \mathit{red})$

$\mathbf{G}\,\neg\,\mathit{red}$



or $(\neg\,\mathit{red})\ \mathbf{U}\ (\neg\,\mathit{blue}\ \wedge\ \neg\,\mathit{red})$



$$\neg\,(\phi\ \mathbf{U}\ \psi)\quad\equiv\quad\mathbf{G}\,\neg\,\psi\ \vee\ (\neg\psi\ \mathbf{U}\ (\neg\phi\ \wedge\ \neg\psi))$$

$$\mathbf{A}\,(\phi\;\mathbf{U}\;\psi)$$

$$\mathbf{A}\,(\phi\;\mathbf{U}\;\psi)$$

$$\equiv$$

$$\neg\,\mathbf{E}\,\neg\,(\phi\;\mathbf{U}\;\psi)$$

$$\mathbf{A}\,(\phi\ \mathbf{U}\ \psi)$$

$$\equiv$$

$$\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$$

(Not a CTL formula)

$$\mathbf{A}\,(\phi\ \mathbf{U}\ \psi)$$

$$\equiv$$

$$\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$$

(Not a CTL formula)

$$\equiv$$

$$\neg\,(\ \mathbf{E}\,\mathbf{G}\,\neg\,\psi\ \ \vee\ \ \mathbf{E}\,(\neg\psi\ \mathbf{U}\ (\neg\psi\ \wedge\ \neg\,\phi))\ )$$

$$\mathbf{A}\,(\phi\ \mathbf{U}\ \psi)$$

$$\equiv$$

$$\neg\,\mathbf{E}\,\neg\,(\phi\ \mathbf{U}\ \psi)$$

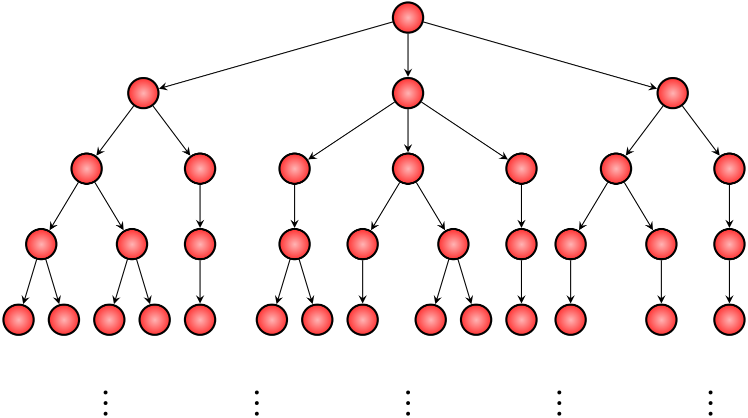
(Not a CTL formula)

$$\equiv$$

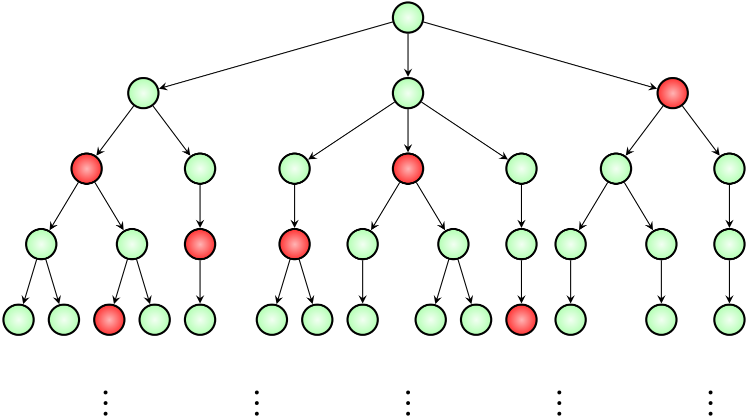
$$\neg\,(\ \mathbf{E}\,\mathbf{G}\,\neg\,\psi\ \ \vee\ \ \mathbf{E}\,(\neg\psi\ \mathbf{U}\,(\neg\psi\ \wedge\ \neg\,\phi))\ )$$

(A CTL formula!)

A G (*red*) equivalent to ¬ **E F** (¬ *red*)



$\mathbf{A}\,\mathbf{F}\,(\mathit{red})$ equivalent to $\neg\,\mathbf{E}\,\mathbf{G}\,(\neg\,\mathit{red})$



First step

State formulae

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid E\,\alpha \mid A\,\alpha$$

$p_i \in AP$ ϕ_1, ϕ_2 : State formulae α : Path formula

Path formulae

$$\alpha := X\,\phi_1 \mid \phi_1\,U\,\phi_2 \mid F\,\phi_1 \mid G\,\phi_1$$

Rewrite **A** in terms of **E**

First step

State formulae

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid E\,\alpha \mid A\,\alpha$$

$p_i \in AP$ ϕ_1, ϕ_2 : State formulae α : Path formula

Path formulae

$$\alpha := X\,\phi_1 \mid \phi_1\,U\,\phi_2 \mid F\,\phi_1 \mid G\,\phi_1$$

Rewrite **A** in terms of **E** **Done!**

All CTL formulas can be written in terms of

E X , **E U** , **E G** and **E F**

All CTL formulas can be written in terms of

$\mathbf{E\,X}$, $\mathbf{E\,U}$, $\mathbf{E\,G}$ and $\mathbf{E\,F}$

Moreover $\mathbf{E\,F}\ \phi \quad \equiv \quad \mathbf{E}\ (\text{true}\ \mathbf{U}\ \phi)$

All CTL formulas can be written in terms of

E X , **E U** , **E G** and **E F**

Moreover $\mathbf{E}\,\mathbf{F}\ \phi \quad \equiv \quad \mathbf{E}\ (\text{true}\ \mathbf{U}\ \phi\ )$

E X, **E U** and **E G** are adequate to describe all CTL formulas

Existential Normal Form (ENF) for CTL

State formulae

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid E\,X\,\phi \mid E\,(\phi_1\, U\, \phi_2) \mid E\,G\,\phi$$

$p_i \in AP$ $\quad$ ϕ, ϕ_1, ϕ_2 : State formulae

Existential Normal Form (ENF) for CTL

State formulae

$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid E\,X\,\phi \mid E\,(\phi_1\,U\,\phi_2) \mid E\,G\,\phi$$

$p_i \in AP$ $\qquad$ ϕ, ϕ_1, ϕ_2 : State formulae

Theorem

For **every** CTL formula there exists an **equivalent** CTL formula in ENF

Module 2:
EX, EU and EG

CTL model-checking problem

Given transition system M and a CTL formula ϕ, find all states of M that satisfy ϕ

CTL model-checking problem

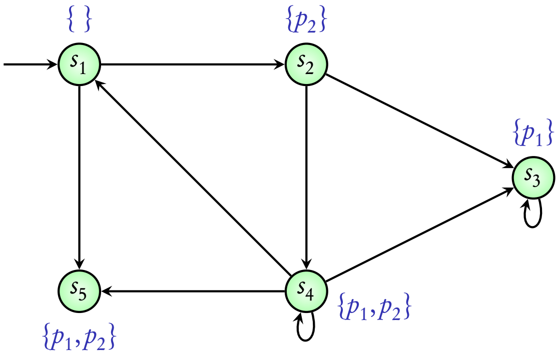
Given transition system M and a CTL formula ϕ, find all states of M that satisfy ϕ

In this unit: Special case when ϕ is either **E X**, **E U** or **E G**

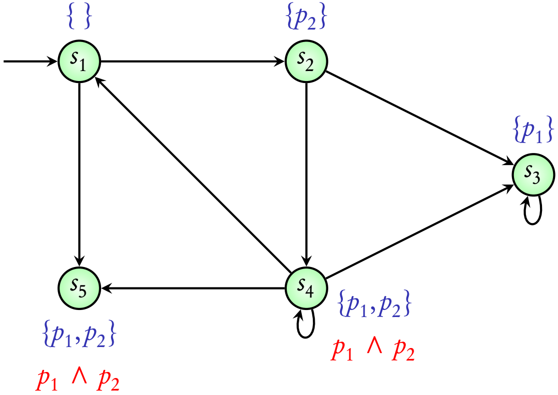
Part 1:

Algorithm for E X

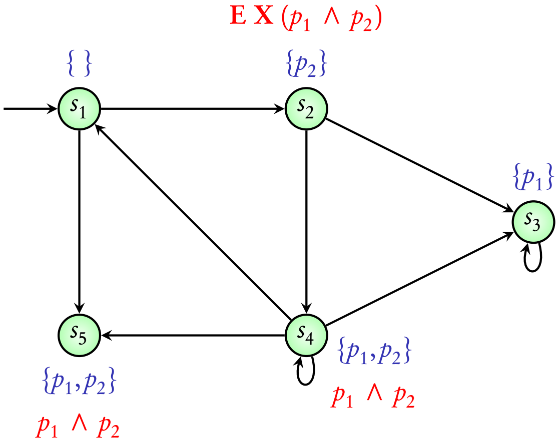
$\mathbf{E}\,\mathbf{X}\,(p_1 \wedge p_2)$



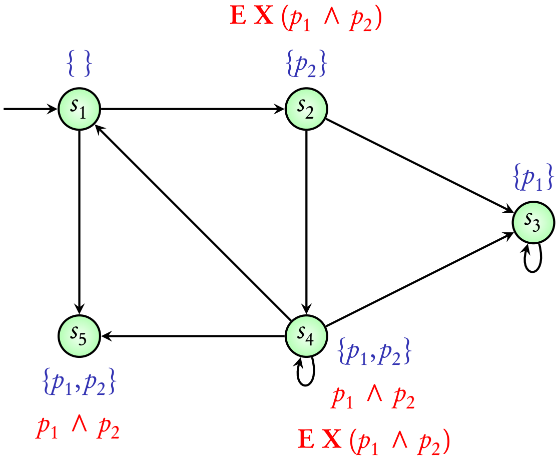
$\mathbf{E}\,\mathbf{X}\,(p_1 \land p_2)$



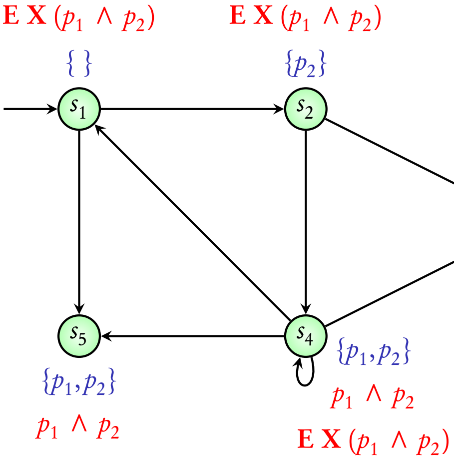
$\mathbf{E\,X}\,(p_1 \wedge p_2)$



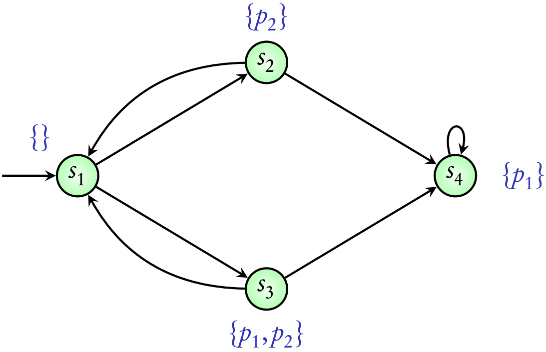
$\mathbf{E}\,\mathbf{X}\,(p_1 \wedge p_2)$



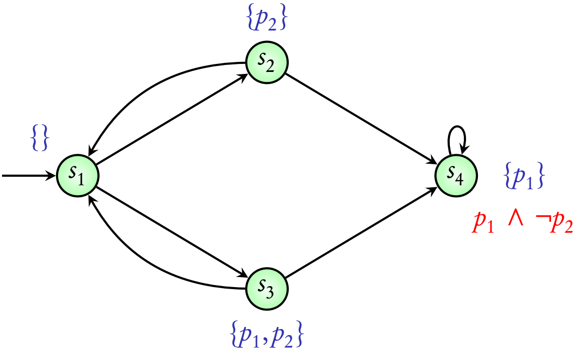
$\mathbf{E\,X}\,(p_1 \wedge p_2)$



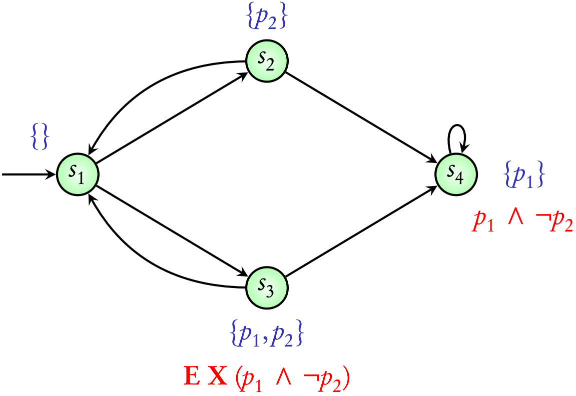
$$\mathbf{E}\,\mathbf{X}\,(p_1 \wedge \neg p_2)$$



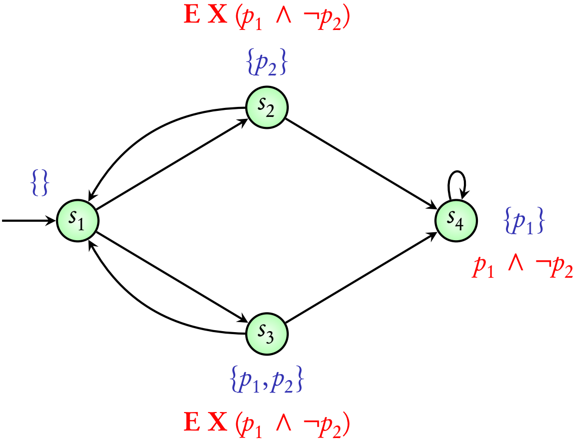
$\mathbf{E}\,\mathbf{X}\,(p_1 \wedge \neg p_2)$



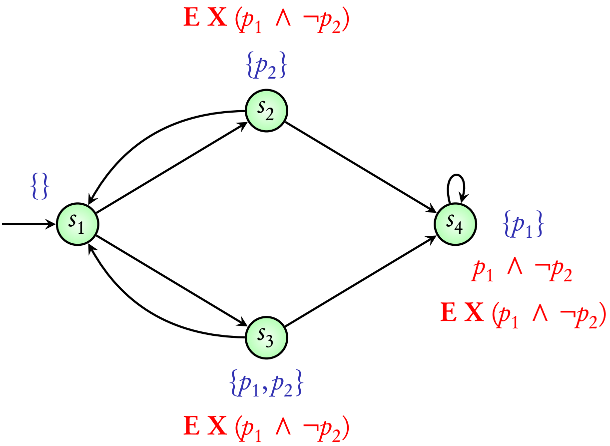
$$\mathbf{E}\,\mathbf{X}\,(p_1 \land \neg p_2)$$



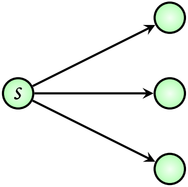
$\mathbf{E}\,\mathbf{X}\,(p_1 \wedge \neg p_2)$



$\mathbf{E}\,\mathbf{X}\,(p_1 \land \neg p_2)$

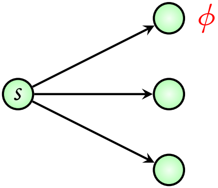


Algorithm for E X ϕ



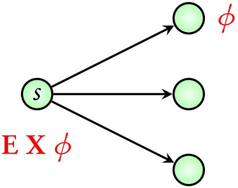
Algorithm for E X ϕ

Suppose states satisfying ϕ have been labelled



Algorithm for E X ϕ

Suppose states satisfying ϕ have been labelled

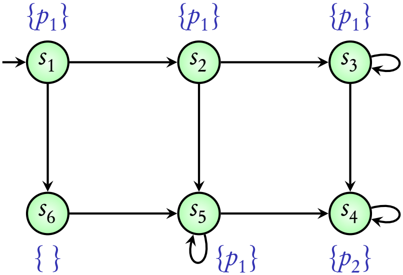


State s is labelled with **E X** ϕ if there **exists a successor** which is labelled ϕ

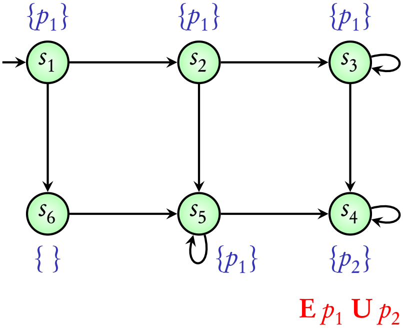
Part 2:

Algorithm for E U

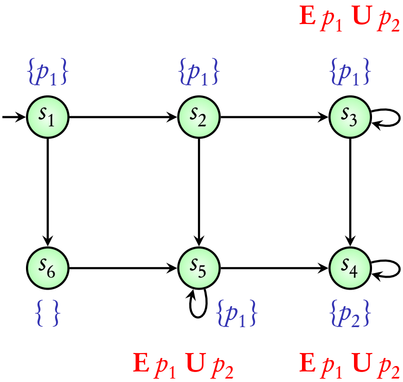
$\mathbf{E}\,(p_1\;\mathbf{U}\;p_2)$



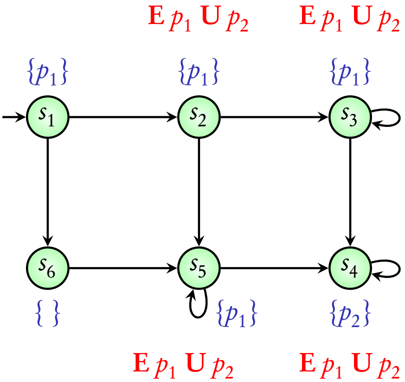
$\mathbf{E}\,(p_1\ \mathbf{U}\ p_2)$



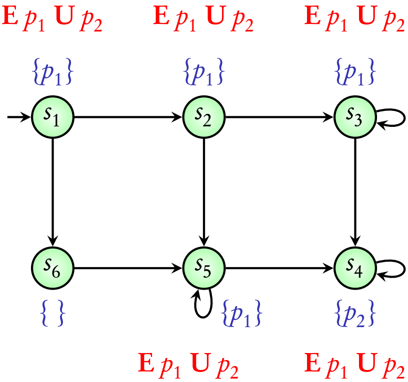
$\mathbf{E}\,(p_1\ \mathbf{U}\ p_2)$



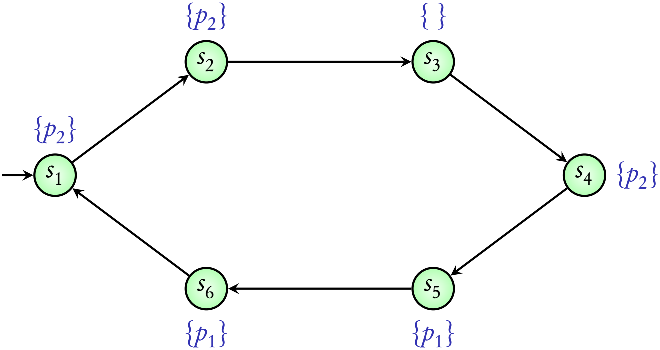
$\mathbf{E}\,(p_1\,\mathbf{U}\,p_2)$



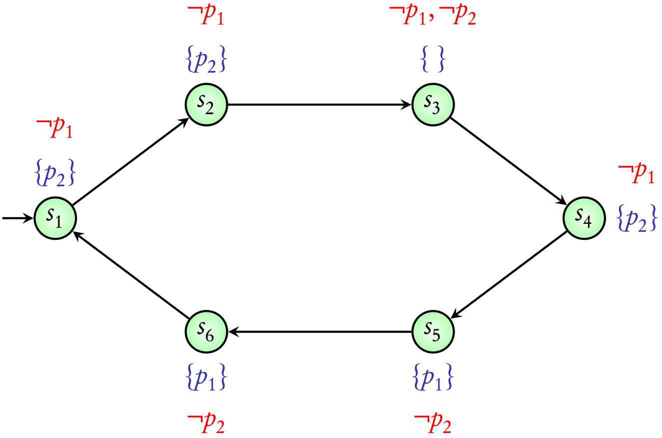
$\mathbf{E}\,(p_1\;\mathbf{U}\;p_2)$



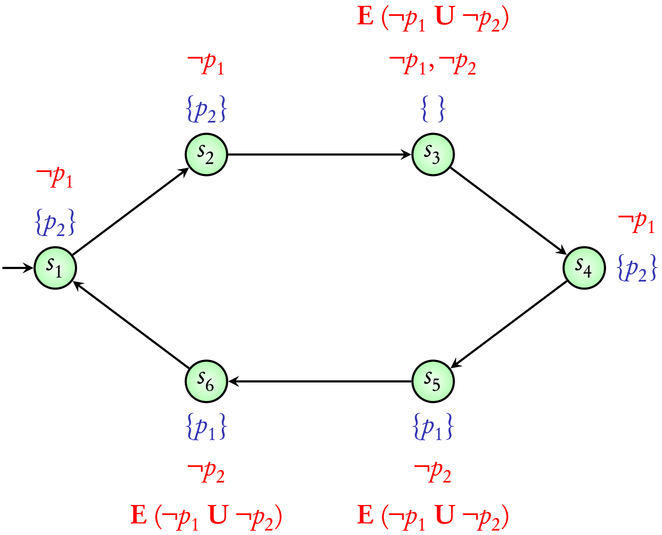
$\mathbf{E}\,(\neg\, p_1\ \mathbf{U}\ \neg\, p_2)$



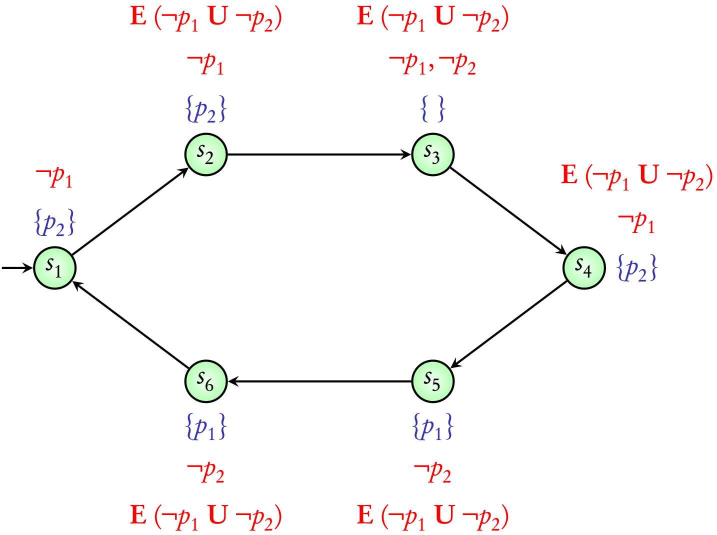
$\mathbf{E}\,(\neg\, p_1\ \mathbf{U}\ \neg\, p_2)$



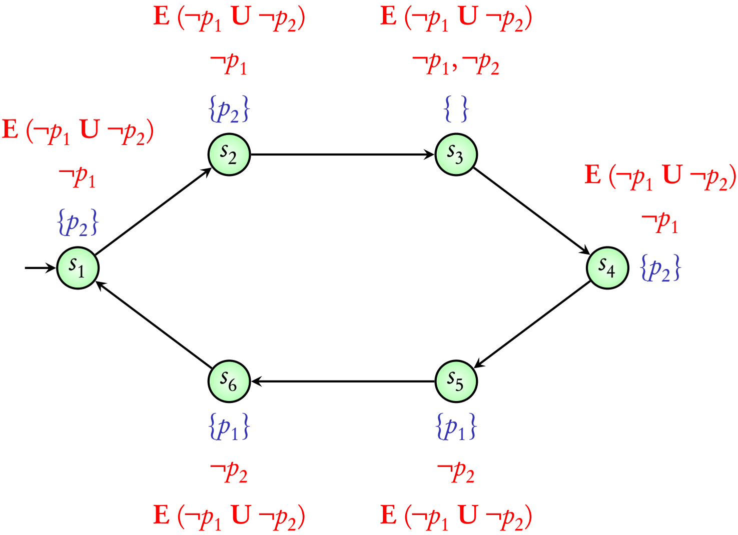
$\mathbf{E}\,(\neg\, p_1\ \mathbf{U}\ \neg\, p_2)$



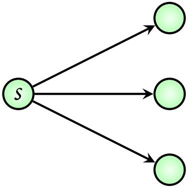
$\mathbf{E}\,(\neg\, p_1\ \mathbf{U}\ \neg\, p_2)$



$\mathbf{E}\,(\neg\, p_1\ \mathbf{U}\ \neg\, p_2)$



Algorithm for E (ϕ_1 U ϕ_2)

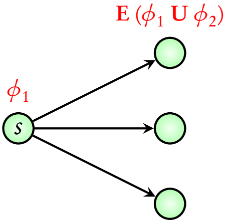


- If any state is labelled with ϕ_2, label it with **E** (ϕ_1 **U** ϕ_2)
- ***Repeat***:

  Label any state with **E** (ϕ_1 **U** ϕ_2) if it is labelled with ϕ_1 and at least one successor is labelled with **E** (ϕ_1 **U** ϕ_2)

  until no change

Algorithm for E (ϕ_1 U ϕ_2)

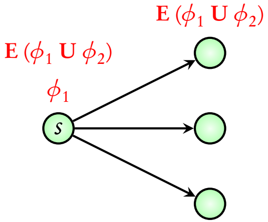


- If any state is labelled with ϕ_2, label it with **E** (ϕ_1 **U** ϕ_2)
- ***Repeat***:

  Label any state with **E** (ϕ_1 **U** ϕ_2) if it is labelled with ϕ_1 and at least one successor is labelled with **E** (ϕ_1 **U** ϕ_2)

  until no change

Algorithm for E (ϕ_1 U ϕ_2)



- If any state is labelled with ϕ_2, label it with **E** (ϕ_1 **U** ϕ_2)
- ***Repeat***:

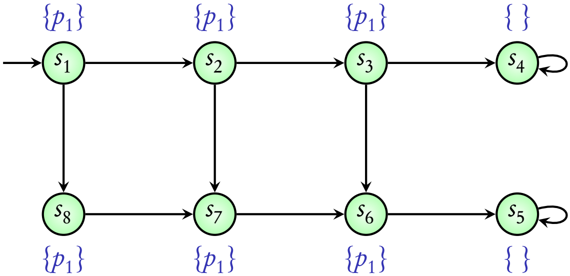
  Label any state with **E** (ϕ_1 **U** ϕ_2) if it is labelled with ϕ_1 and at least one successor is labelled with **E** (ϕ_1 **U** ϕ_2)

  until no change

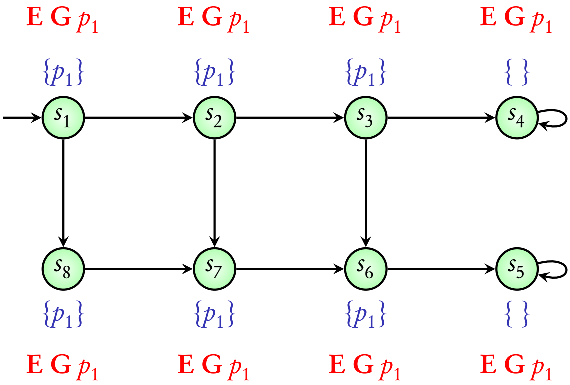
Part 3:

Algorithm for E G

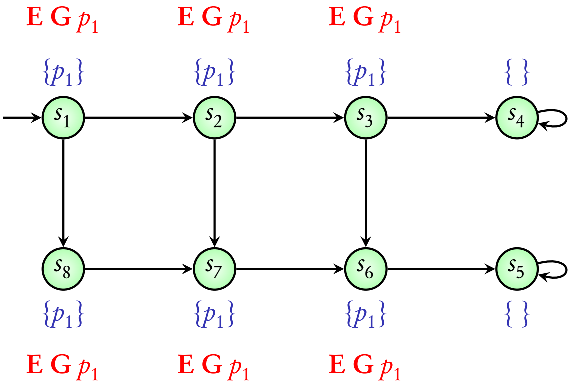
$\mathbf{E}\,\mathbf{G}\,p_1$



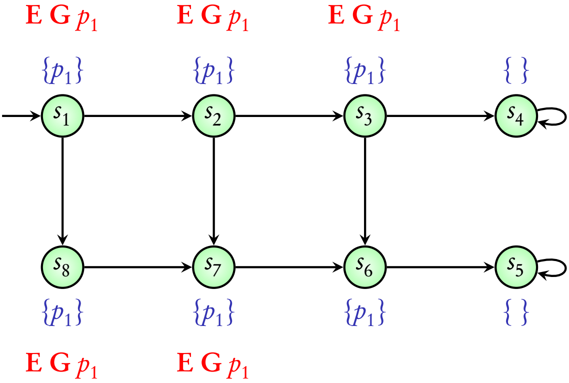
$\mathbf{E}\,\mathbf{G}\,p_1$



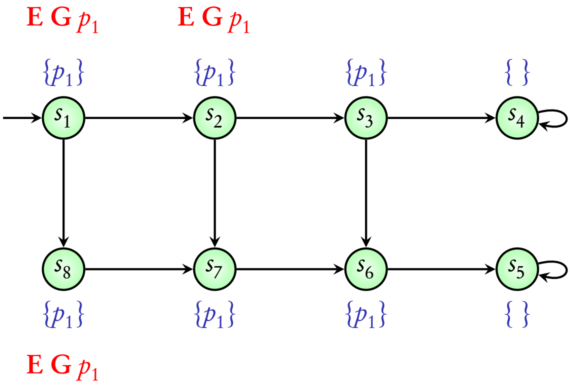
$\mathbf{E}\,\mathbf{G}\,p_1$



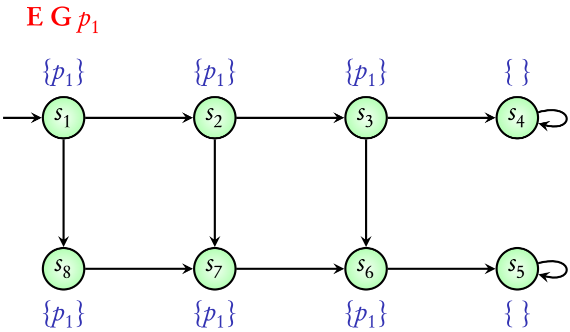
$\mathbf{E}\,\mathbf{G}\,p_1$



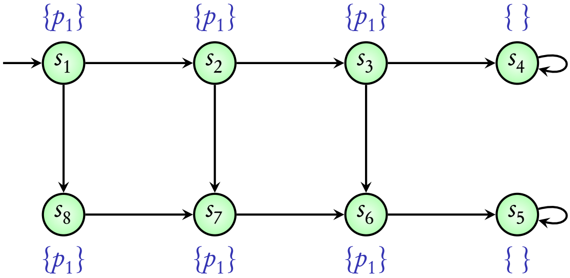
$\mathbf{E}\,\mathbf{G}\,p_1$



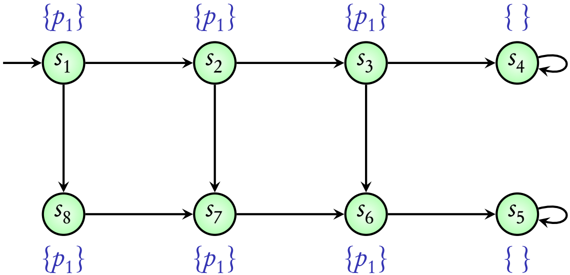
$\mathbf{E}\,\mathbf{G}\,p_1$



$\mathbf{E}\,\mathbf{G}\,p_1$

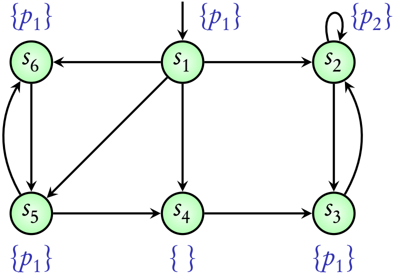


$\mathbf{E}\ \mathbf{G}\ p_1$

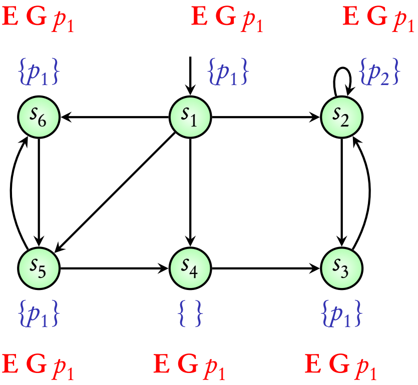


No state of the above transition system satisfies $\mathbf{E}\ \mathbf{G}\ p_1$

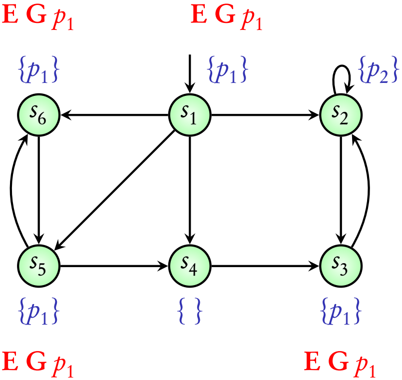
$\mathbf{E\,G}\,p_1$



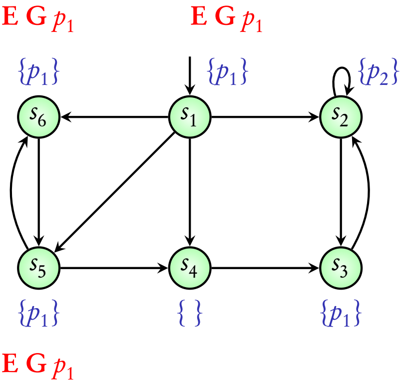
$\mathbf{E}\,\mathbf{G}\,p_1$



$\mathbf{E}\,\mathbf{G}\,p_1$



$\mathbf{E}\,\mathbf{G}\,p_1$



Algorithm for E G ϕ

Algorithm for $\mathbf{E}\,\mathbf{G}\,\phi$

- Label all states with $\mathbf{E}\,\mathbf{G}\,\phi$

Algorithm for $\mathbf{E}\ \mathbf{G}\ \phi$

- Label all states with $\mathbf{E}\ \mathbf{G}\ \phi$
- If any state is **not** labelled with ϕ, **delete** the label $\mathbf{E}\ \mathbf{G}\ \phi$

Algorithm for E G ϕ

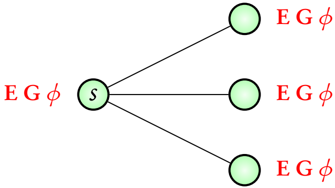
- Label all states with **E G** ϕ
- If any state is **not** labelled with ϕ, **delete** the label **E G** ϕ
- *Repeat*:

  Delete the label **E G** ϕ from a state if **none** of its successors is labelled with **E G** ϕ

  until no change

Algorithm for E G ϕ

- Label all states with **E G ϕ**
- If any state is **not** labelled with ϕ, **delete** the label **E G ϕ**



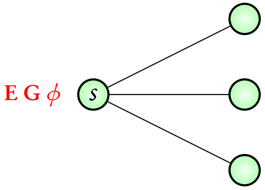
- *Repeat*:

  Delete the label **E G ϕ** from a state if **none** of its successors is labelled with **E G ϕ**

  until no change

Algorithm for E G ϕ

- Label all states with **E G ϕ**
- If any state is **not** labelled with ϕ, **delete** the label **E G ϕ**



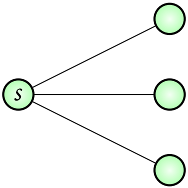
- *Repeat*:

  Delete the label **E G ϕ** from a state if **none** of its successors is labelled with **E G ϕ**

  until no change

Algorithm for E G ϕ

- Label all states with **E G** ϕ
- If any state is **not** labelled with ϕ, **delete** the label **E G** ϕ



- *Repeat*:

  Delete the label **E G** ϕ from a state if **none** of its successors is labelled with **E G** ϕ

  until no change

Summary

Algorithms

EX, EU, EG

Module 3:

Final algorithm

CTL model-checking problem

Given transition system M and a CTL formula ϕ, find all states of M that satisfy ϕ

CTL model-checking problem

Given transition system M and a CTL formula ϕ, find all states of M that satisfy ϕ

- **Module 1:** Every CTL formula can be written using **EX, EU, EG**
- **Module 2:** Labelling algorithms for **EX, EU, EG**

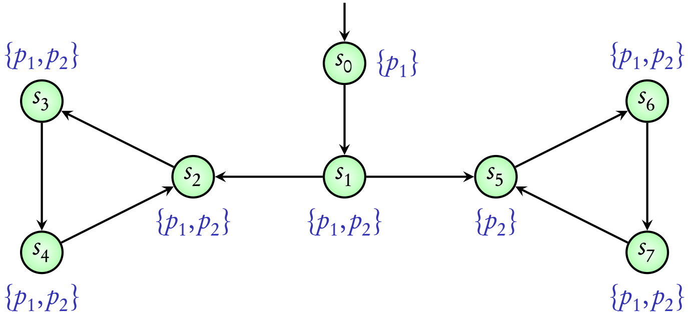
Coming next: Generic algorithm for a CTL formula

State formulae

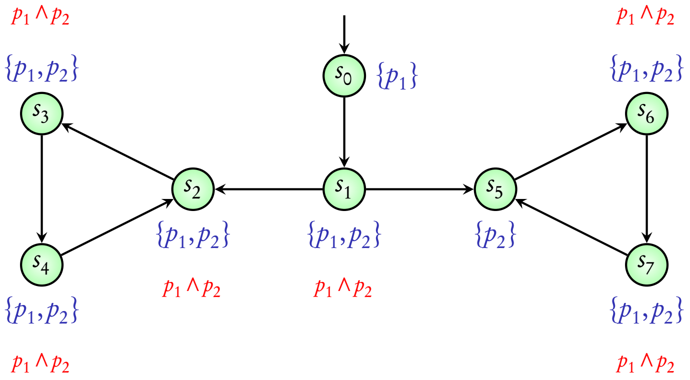
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid E\,X\,\phi \mid E\,(\phi_1\,U\,\phi_2) \mid E\,G\,\phi$$

$p_i \in AP$ $\qquad$ ϕ, ϕ_1, ϕ_2 : State formulae

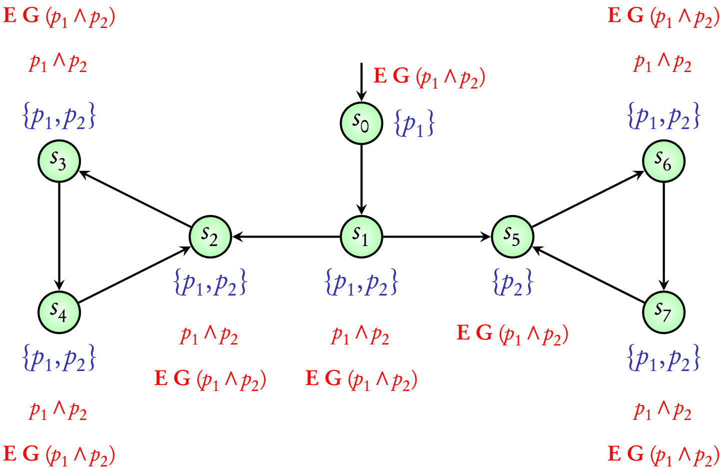
$\mathbf{E\,X\,E\,G}\,(p_1 \wedge p_2)$



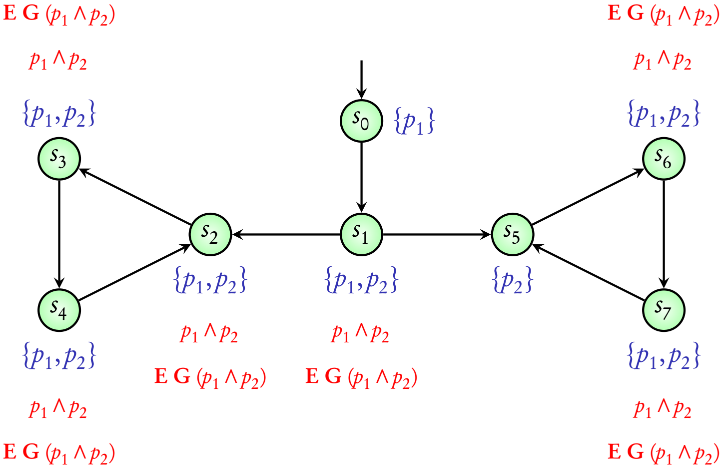
$\mathbf{E\,X\,E\,G}\,(p_1 \wedge p_2)$



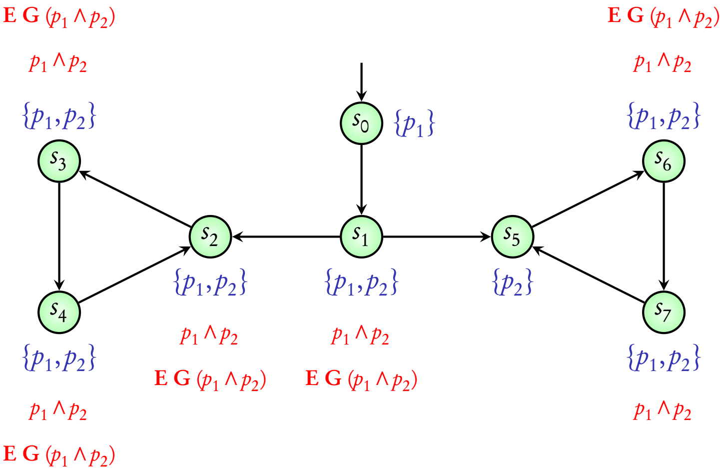
$\mathbf{E\,X\,E\,G}\,(p_1 \wedge p_2)$



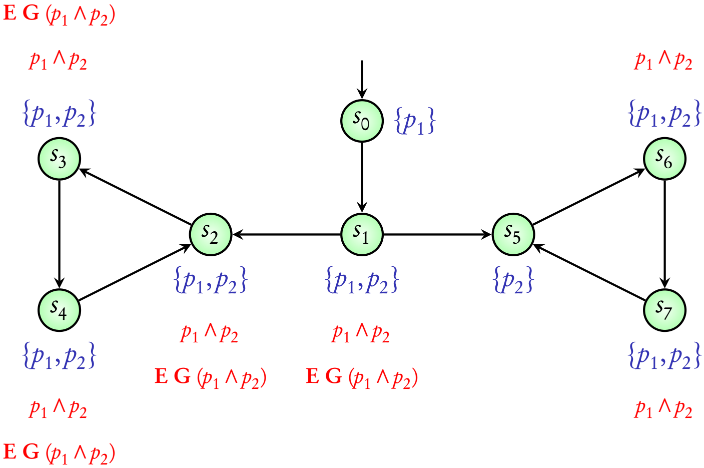
$\mathbf{E\,X\,E\,G}\,(p_1 \wedge p_2)$



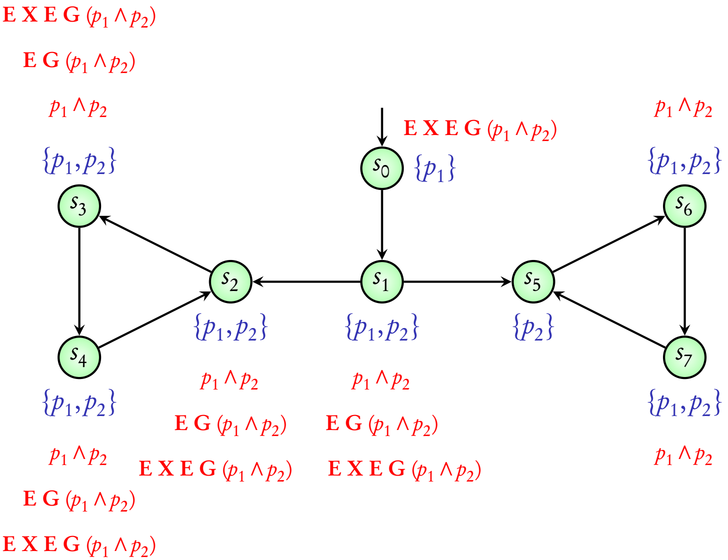
$\mathbf{E\,X\,E\,G}\,(p_1 \wedge p_2)$



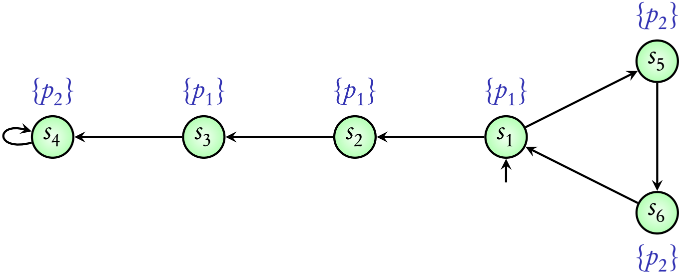
$\mathbf{E\,X\,E\,G}\,(p_1 \wedge p_2)$



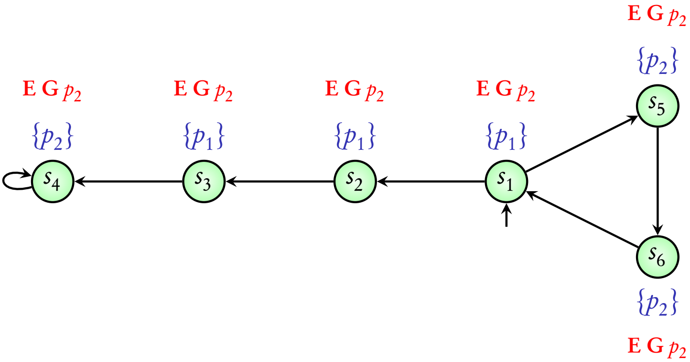
$\mathbf{E\,X\,E\,G}\,(p_1 \wedge p_2)$



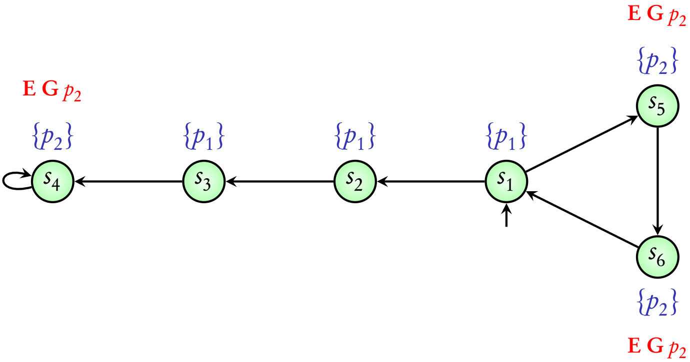
$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\, \mathbf{G}\, p_2)$



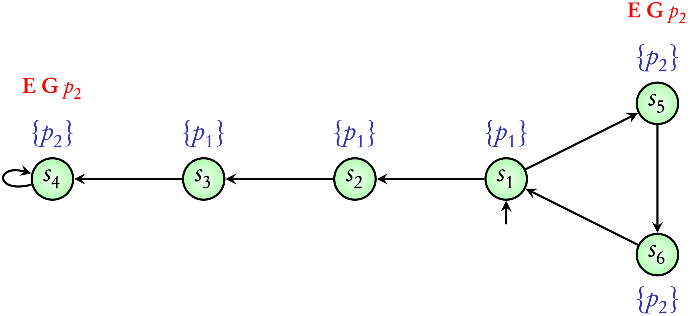
$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\, \mathbf{G}\, p_2)$



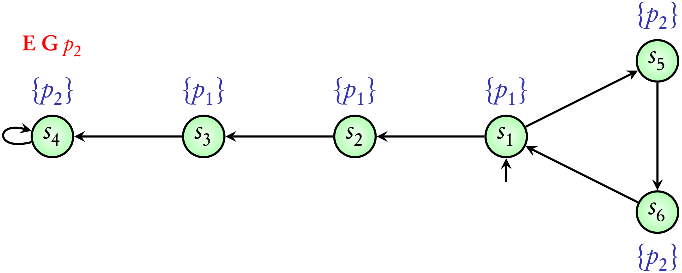
$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\, \mathbf{G}\, p_2)$



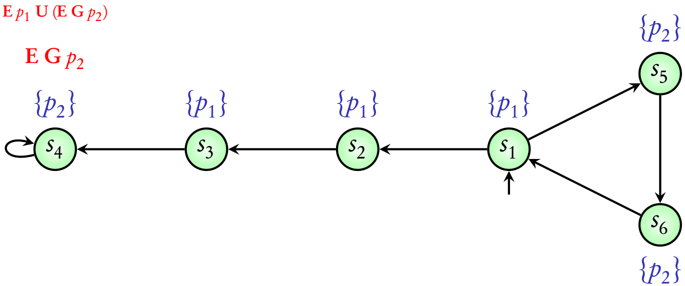
$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\, \mathbf{G}\, p_2)$



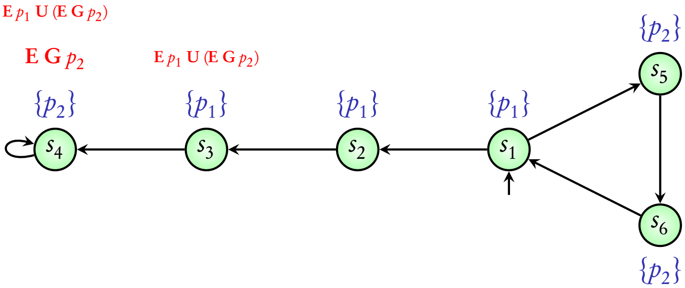
$\mathbf{E}\ p_1\ \mathbf{U}\ (\mathbf{E}\ \mathbf{G}\ p_2)$



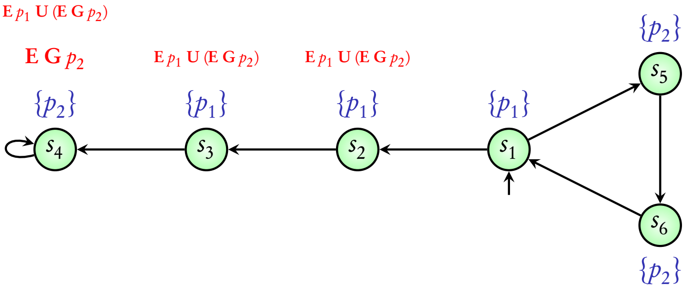
$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\,\mathbf{G}\, p_2)$



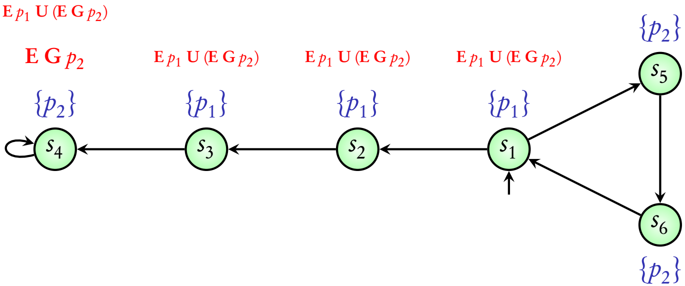
$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\, \mathbf{G}\, p_2)$



$$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\, \mathbf{G}\, p_2)$$



$\mathbf{E}\, p_1\, \mathbf{U}\, (\mathbf{E}\, \mathbf{G}\, p_2)$



function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

end case
end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

ϕ is *true* : **return** S

end case

end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

ϕ is *true* : **return** S

ϕ is p_i : **return** {states containing p_i }

end case
end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

ϕ is *true* : **return** S

ϕ is p_i : **return** {states containing p_i }

ϕ is $\phi_1 \wedge \phi_2$: **return** SAT(ϕ_1) $\cap$ SAT(ϕ_2)

end case
end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

ϕ is *true* : **return** S

ϕ is p_i : **return** {states containing p_i }

ϕ is $\phi_1 \wedge \phi_2$: **return** SAT(ϕ_1) $\cap$ SAT(ϕ_2)

ϕ is $\neg\phi_1$: **return** $S -$ SAT(ϕ_1)

end case

end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

ϕ is *true* : **return** S

ϕ is p_i : **return** {states containing p_i }

ϕ is $\phi_1 \wedge \phi_2$: **return** SAT(ϕ_1) $\cap$ SAT(ϕ_2)

ϕ is $\neg\phi_1$: **return** S – SAT(ϕ_1)

ϕ is **E X** ϕ_1 : **return** $\text{SAT}_{\text{EX}}(\phi_1)$ /* procedure seen in Module 2 */

end case
end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

ϕ is *true* : **return** S

ϕ is p_i : **return** {states containing p_i }

ϕ is $\phi_1 \wedge \phi_2$: **return** SAT(ϕ_1) $\cap$ SAT(ϕ_2)

ϕ is $\neg\phi_1$: **return** S – SAT(ϕ_1)

ϕ is **E X** ϕ_1 : **return** $\text{SAT}_{\text{EX}}(\phi_1)$ /* procedure seen in Module 2 */

ϕ is **E** (ϕ_1 **U** ϕ_2) : **return** $\text{SAT}_{\text{EU}}(\phi_1, \phi_2)$ /* procedure seen in Module 2 */

end case
end function

function SAT(ϕ)

/* Input: **Transition system** M with state set S, **CTL formula** ϕ in ENF */

/* Output: Set of states satisfying ϕ */

case

ϕ is *true* : **return** S

ϕ is p_i : **return** {states containing p_i }

ϕ is $\phi_1 \wedge \phi_2$: **return** $\mathrm{SAT}(\phi_1) \cap \mathrm{SAT}(\phi_2)$

ϕ is $\neg\phi_1$: **return** $S - \mathrm{SAT}(\phi_1)$

ϕ is **E X** ϕ_1 : **return** $\mathrm{SAT}_{\mathrm{EX}}(\phi_1)$ /* procedure seen in Module 2 */

ϕ is **E** (ϕ_1 **U** ϕ_2) : **return** $\mathrm{SAT}_{\mathrm{EU}}(\phi_1, \phi_2)$ /* procedure seen in Module 2 */

ϕ is **E G** ϕ_1 : **return** $\mathrm{SAT}_{\mathrm{EG}}(\phi_1)$ /* procedure seen in Module 2 */

end case
end function

CTL model-checking algorithm

Reference: Logic in Computer Science, *by Huth and Ryan* - Section 3.6.1