بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه چهاردهم: پیچیدگی حافظه (۳)

Theory of computation - 002 - S14 - space complexity (3)

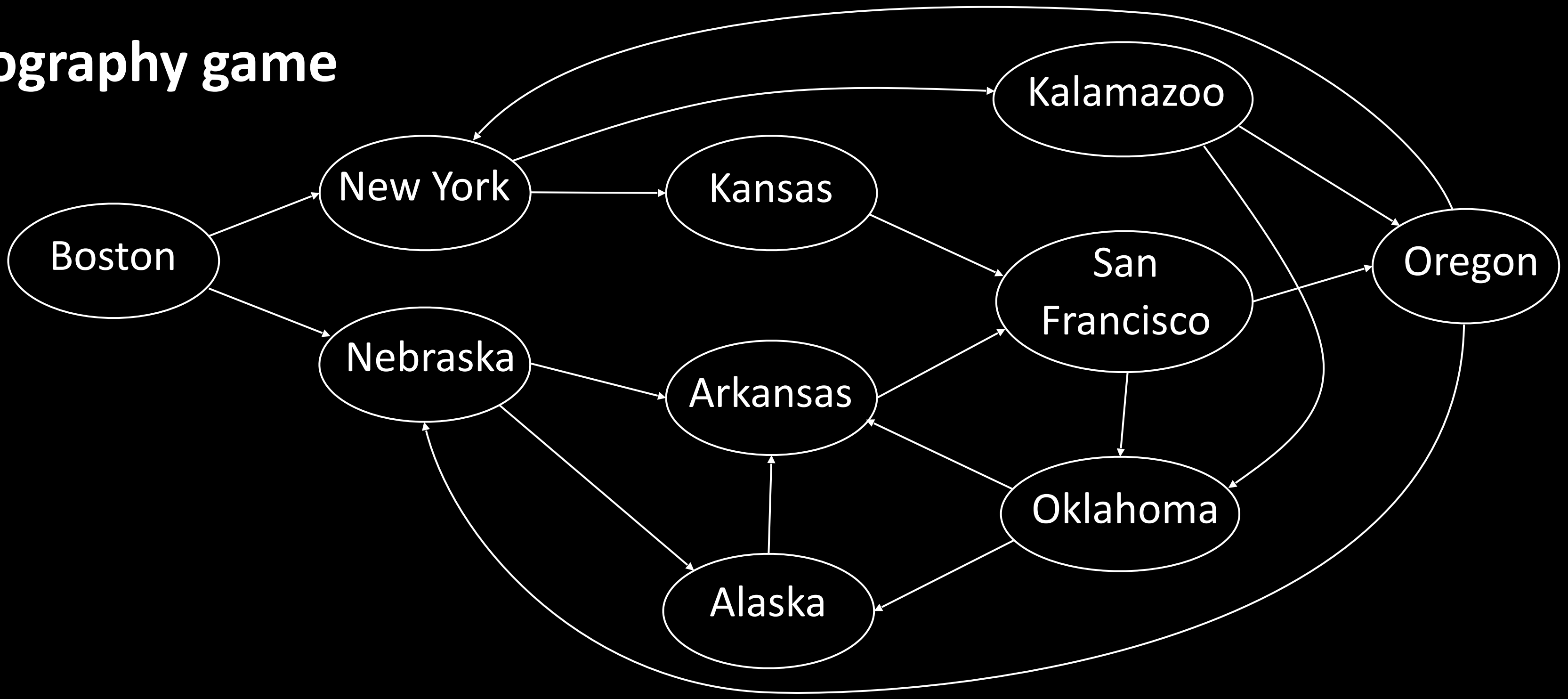
Games and Complexity

Games and Complexity

Geography game

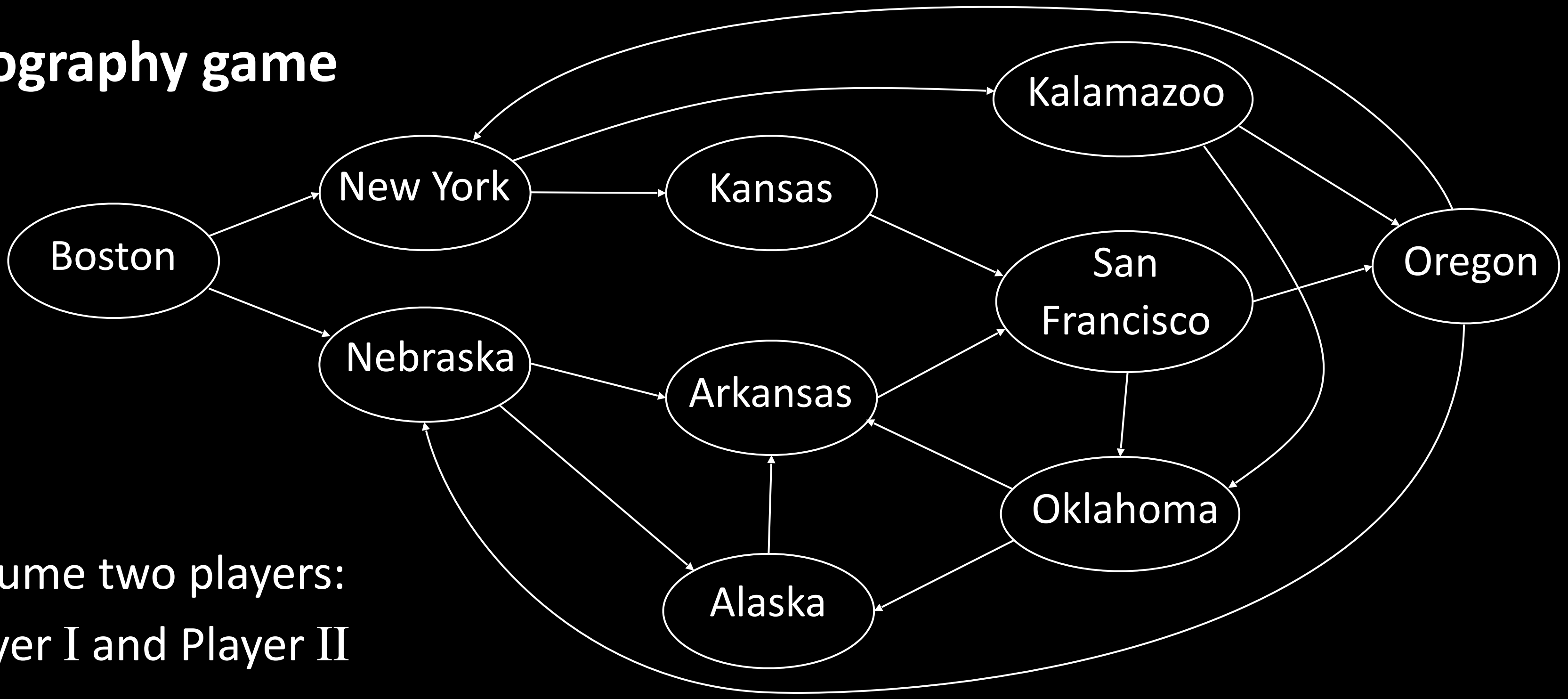
Games and Complexity

Geography game



Games and Complexity

Geography game



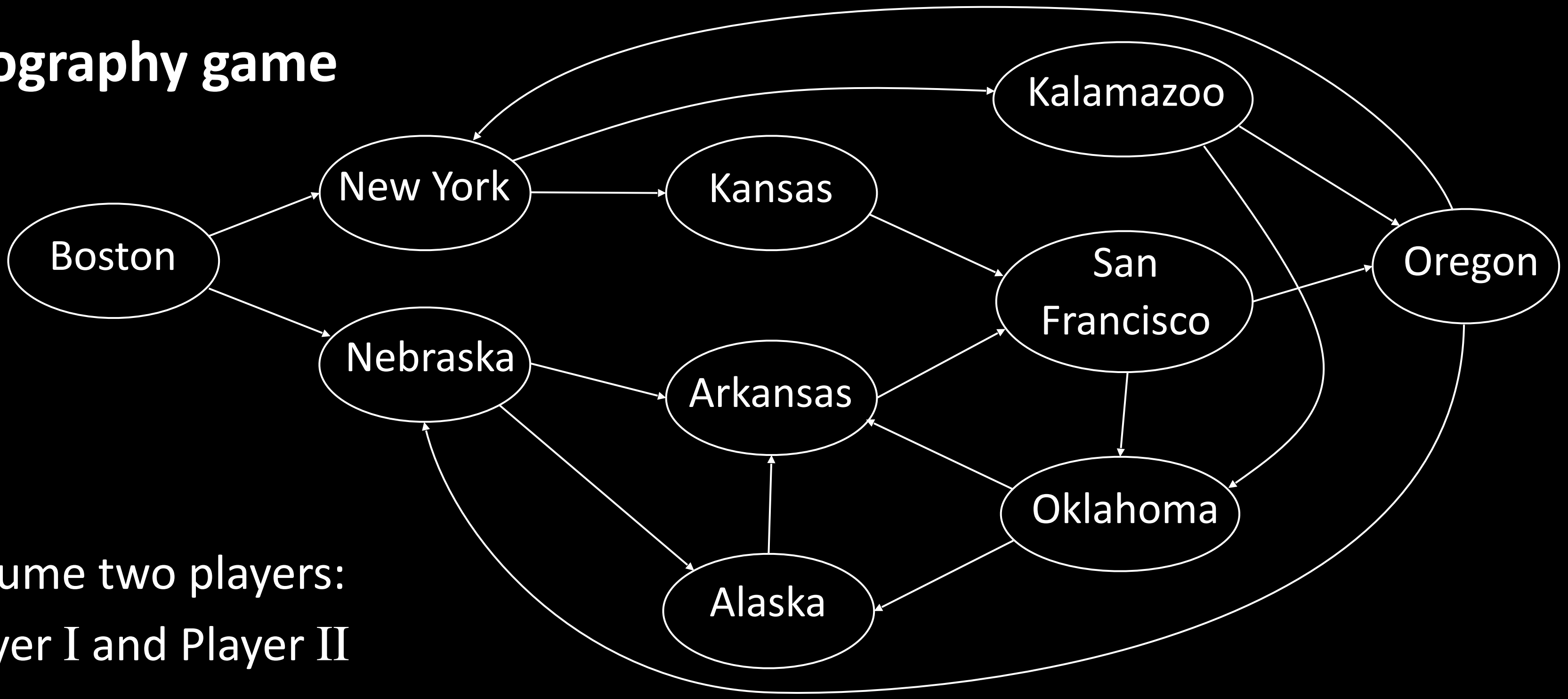
Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

Games and Complexity

Geography game



Generalized Geography Game

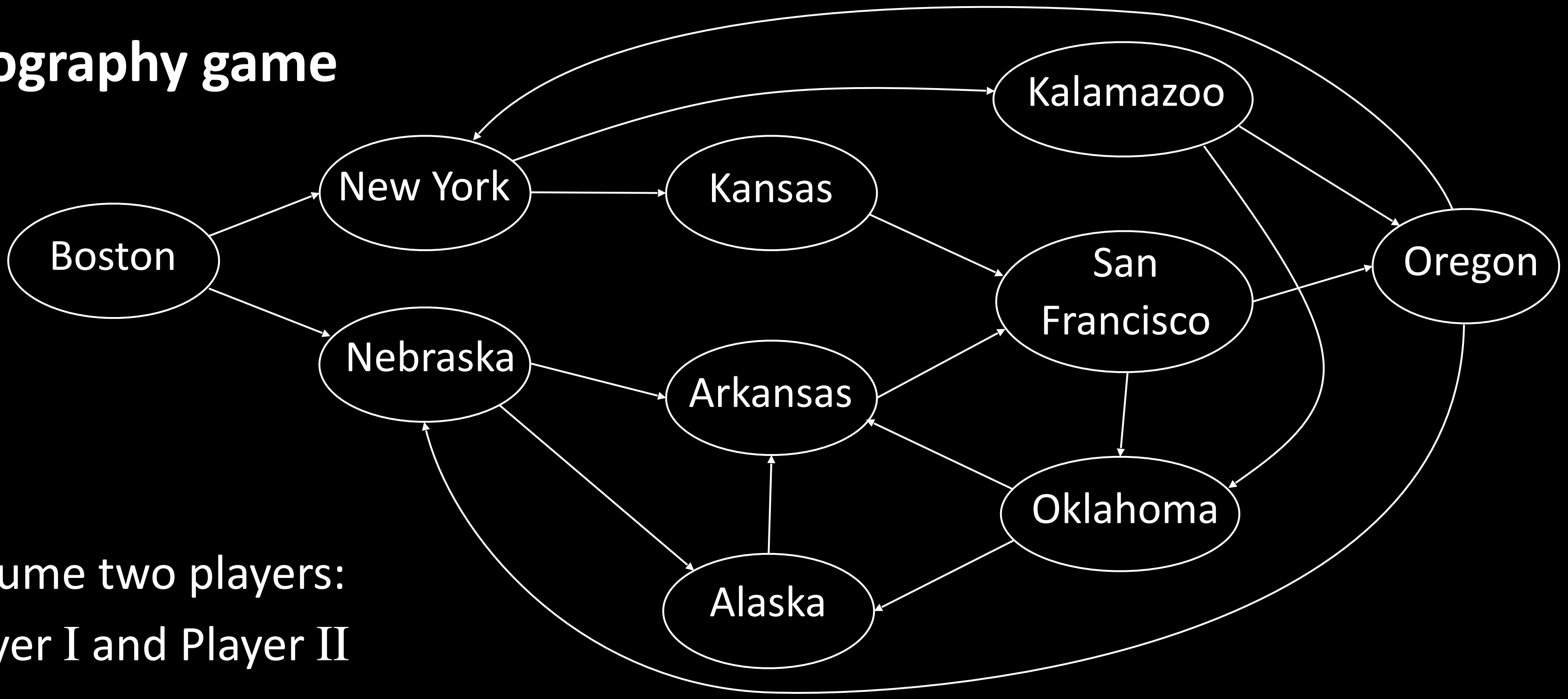
Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

Games and Complexity

Geography game



Generalized Geography Game

Played on any directed graph.

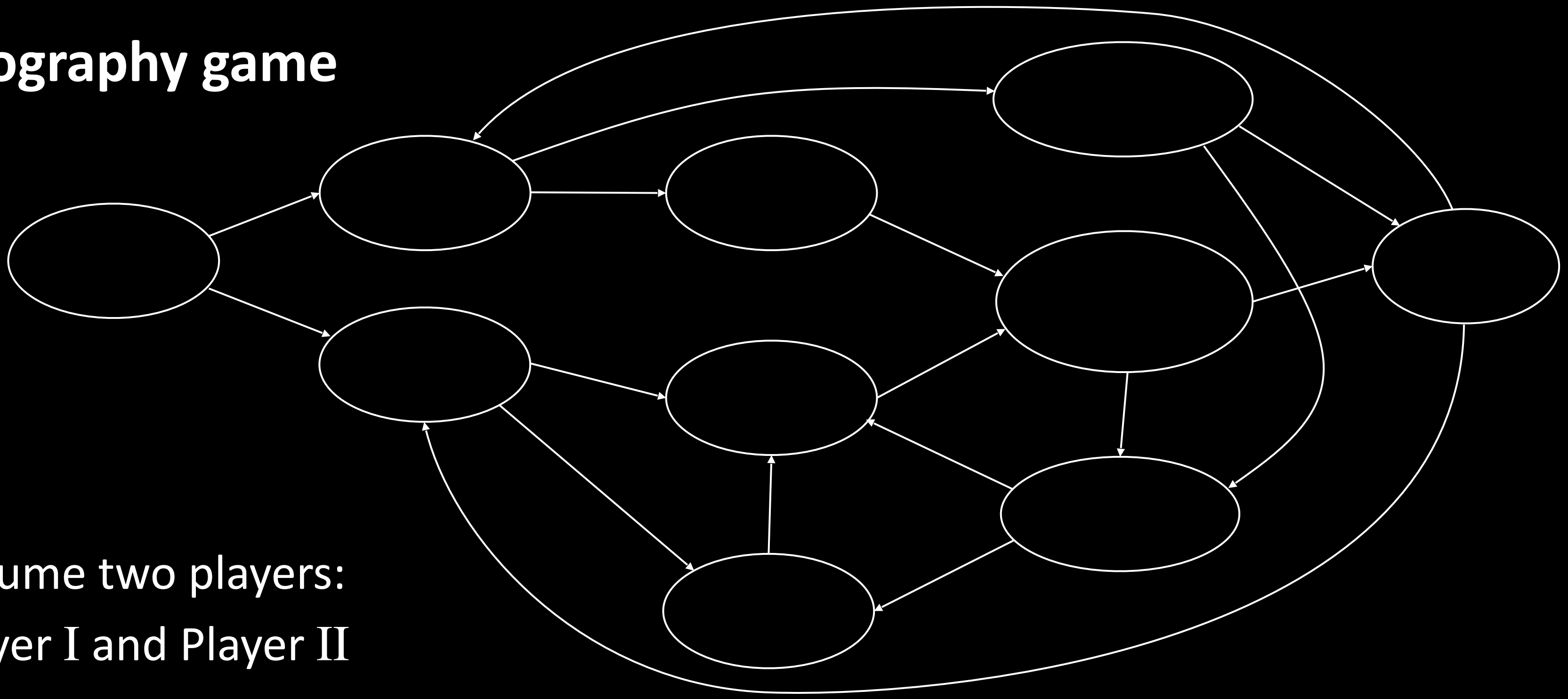
Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

Games and Complexity

Geography game



Generalized Geography Game

Played on any directed graph.

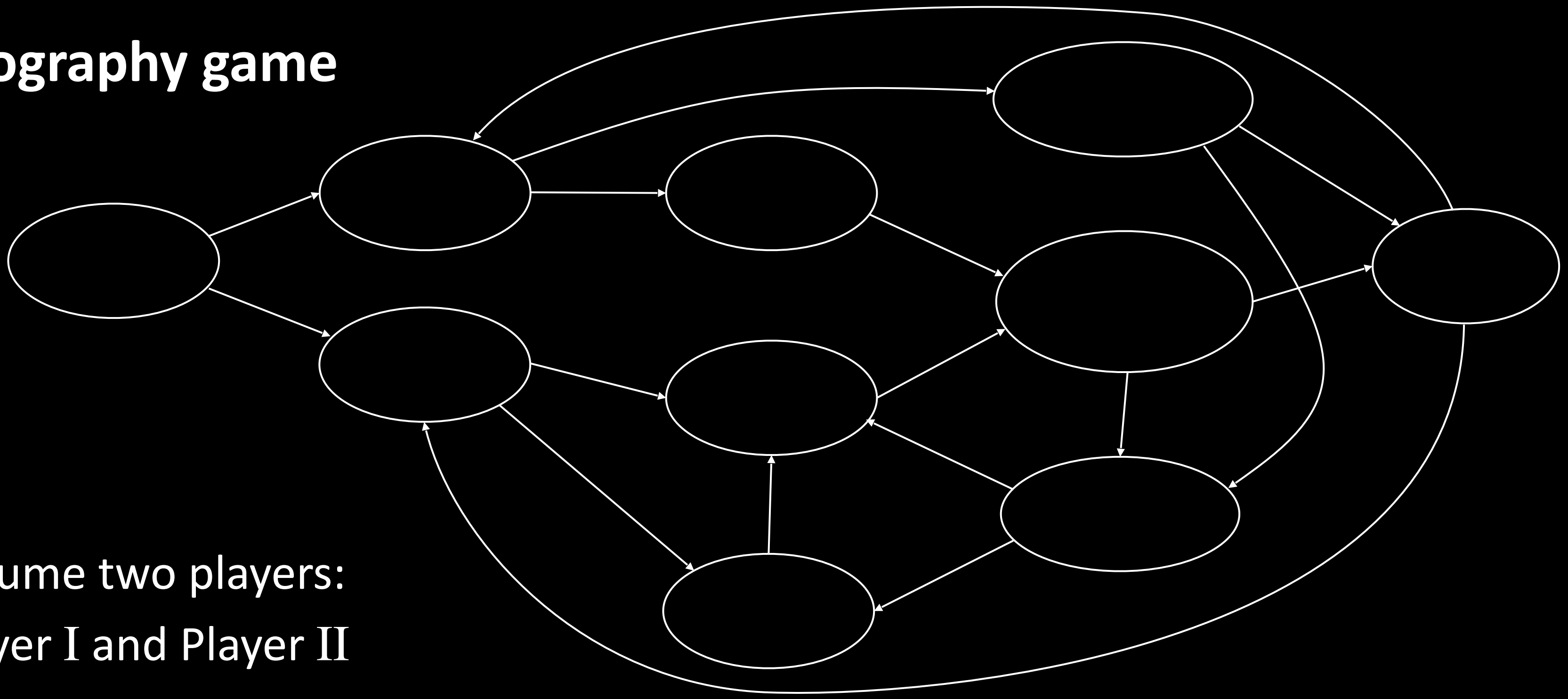
Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed.
The first player stuck (= cannot move) loses.

Games and Complexity

Geography game



Assume two players:

Player I and Player II

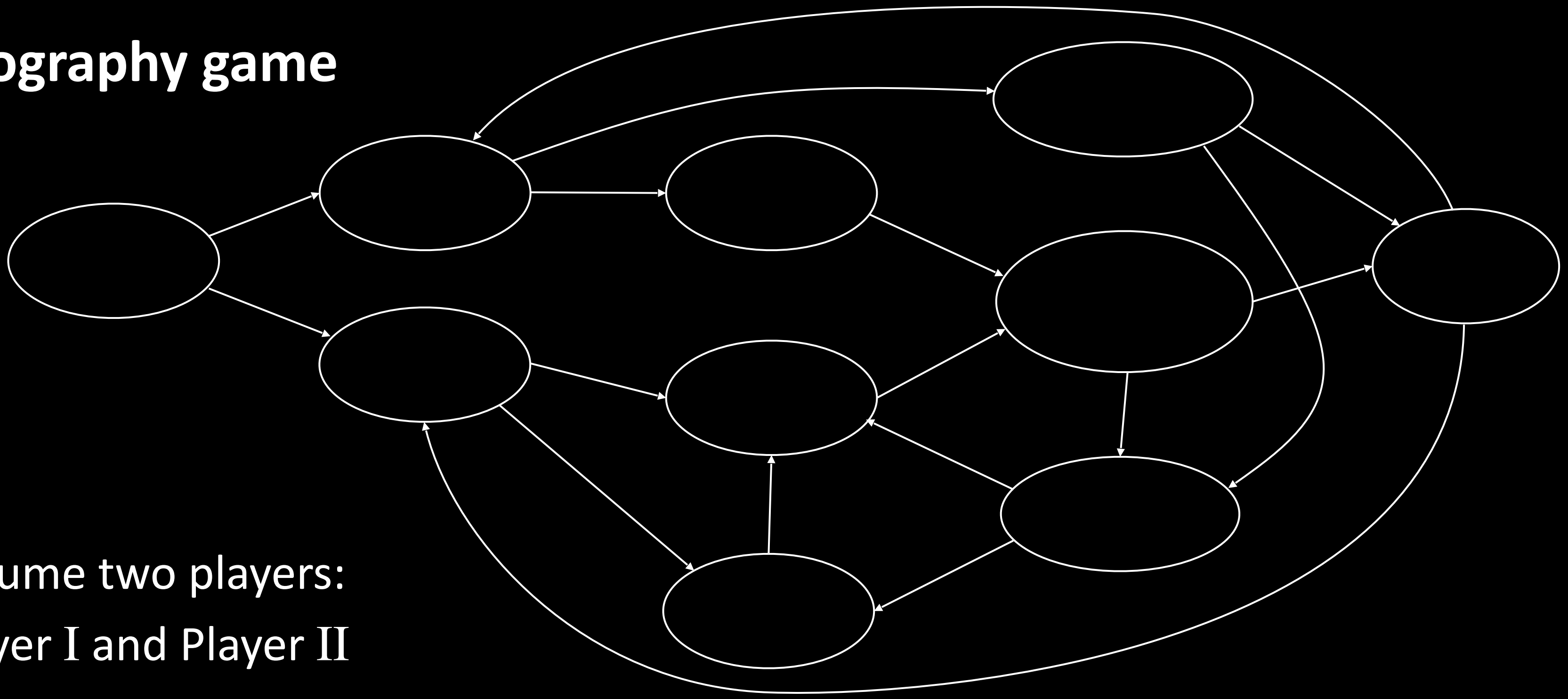
Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

Generalized Geography Game

Played on any directed graph. Players take turns picking nodes that form a simple path.

Games and Complexity

Geography game



Assume two players:

Player I and Player II

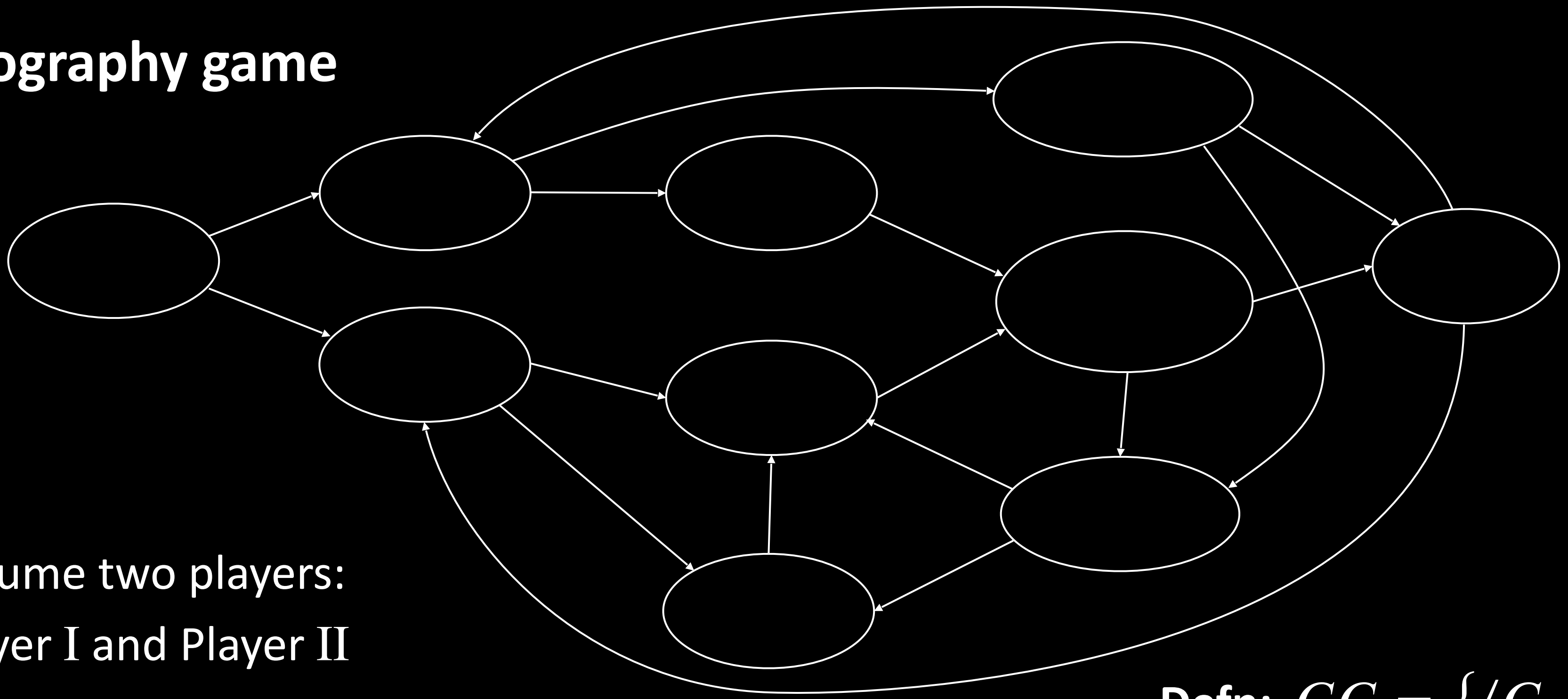
Players take turns picking places that start with the letter which ended the previous place. No repeats allowed.
The first player stuck (= cannot move) loses.

Generalized Geography Game

Played on any directed graph.
Players take turns picking nodes that form a simple path.
The first player stuck loses.

Games and Complexity

Geography game



Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

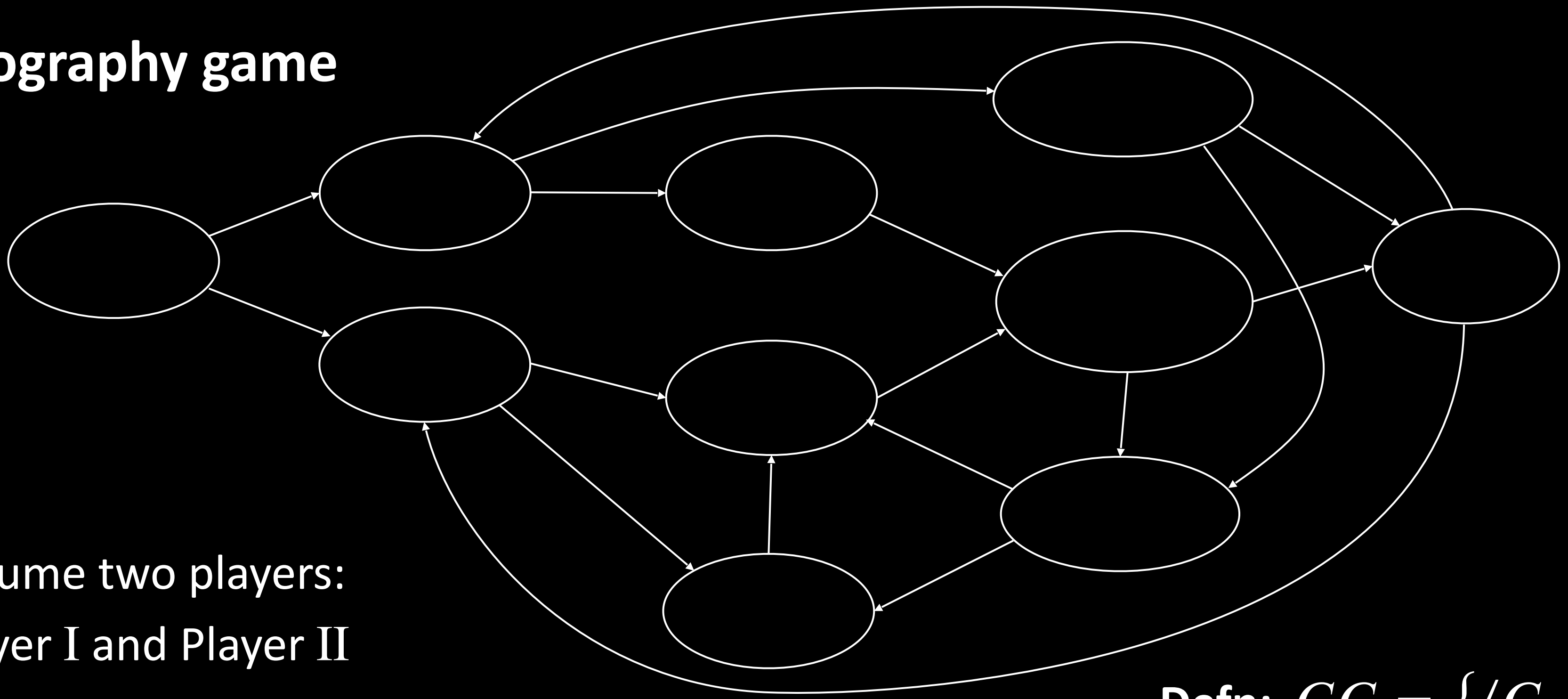
Generalized Geography Game

Played on any directed graph.
Players take turns picking nodes that form a simple path.
The first player stuck loses.

Defn: $GG = \{\langle G, a\rangle \mid$ Player I has a forced win in Generalized Geography on graph G starting at node $a\}$.

Games and Complexity

Geography game



Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

Generalized Geography Game

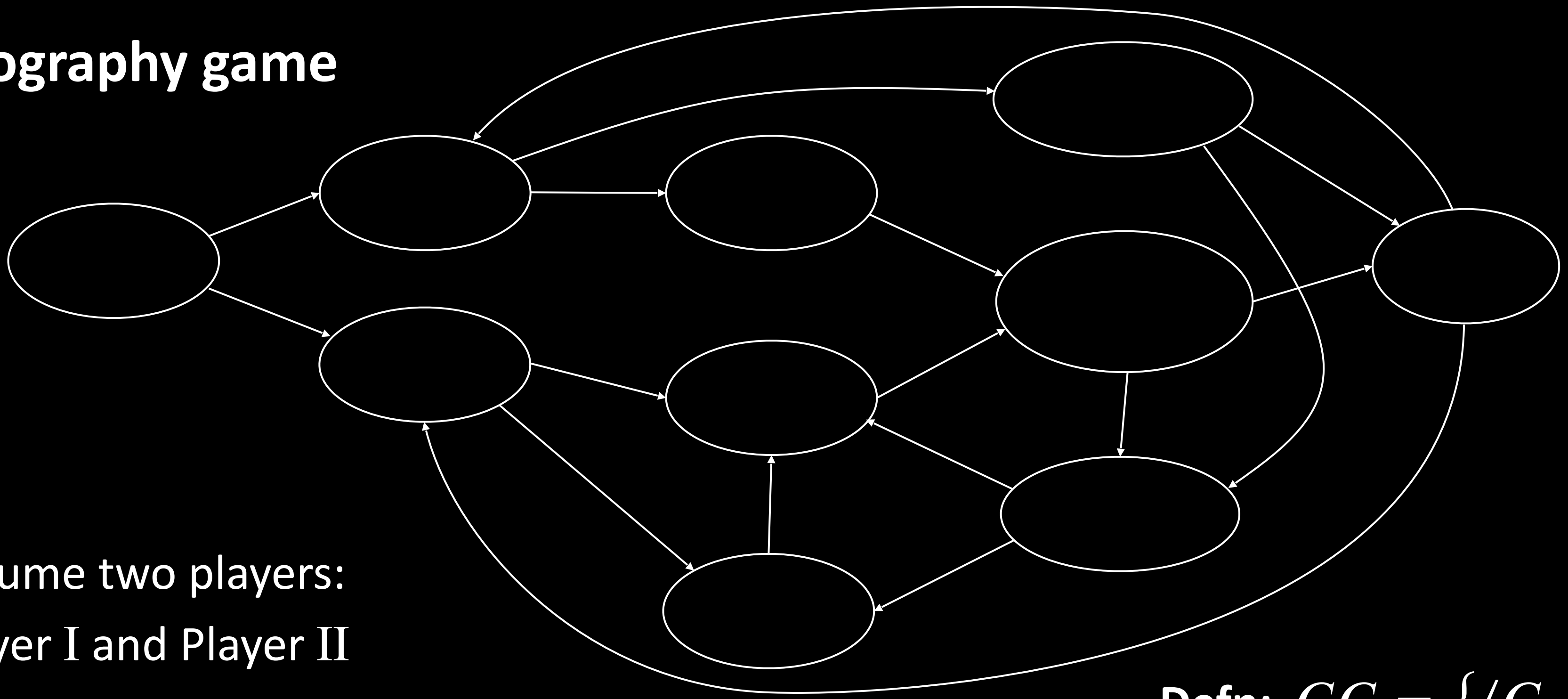
Played on any directed graph.
Players take turns picking nodes that form a simple path.
The first player stuck loses.

Defn: $GG = \{\langle G, a\rangle \mid$ Player I has a forced win in Generalized Geography on graph G starting at node $a\}$.

"forced win" also called a "winning strategy" means that the player will win if both players play optimally.

Games and Complexity

Geography game



Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

Generalized Geography Game

Played on any directed graph.
Players take turns picking nodes that form a simple path.
The first player stuck loses.

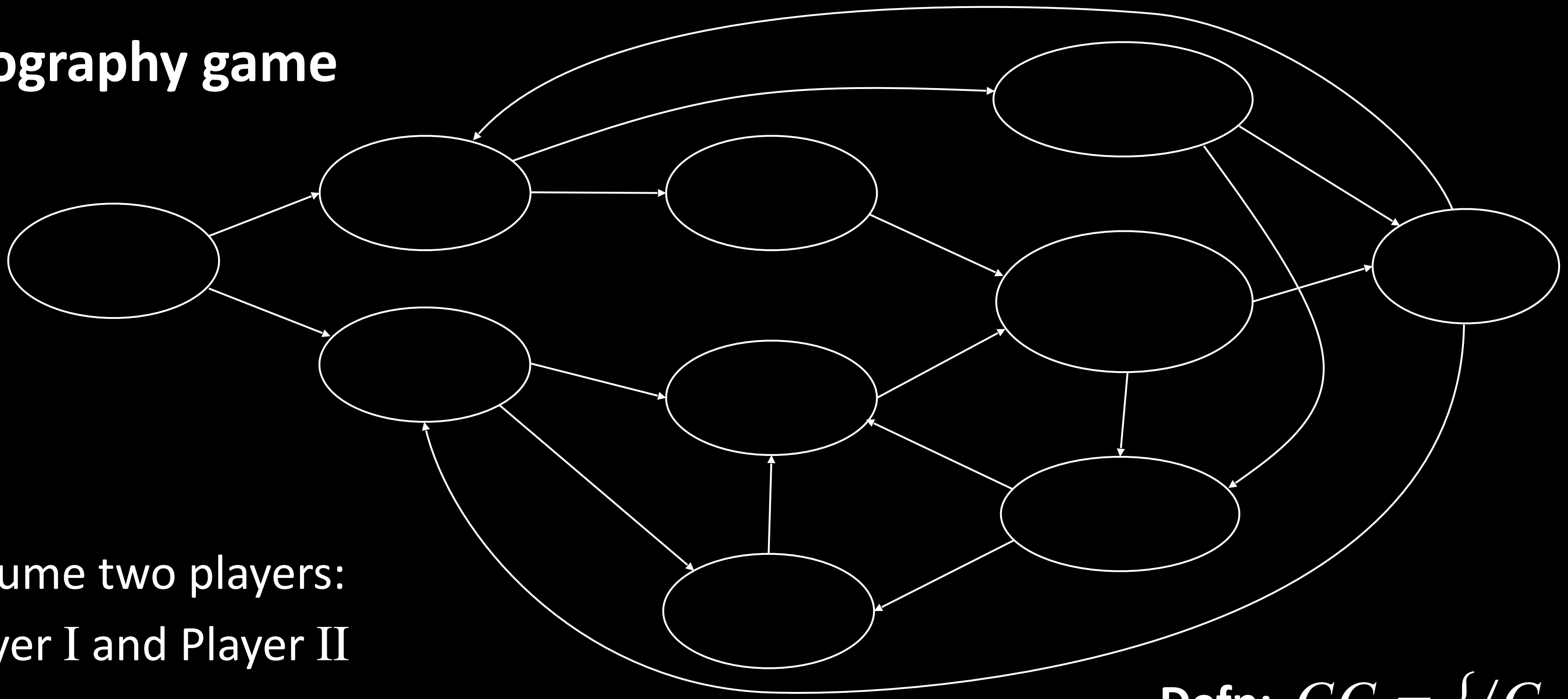
Defn: $GG = \{\langle G, a \rangle \mid$ Player I has a <u>forced win</u> in Generalized Geography on graph G starting at node $a\}$.

"forced win" also called a "winning strategy" means that the player will win if both players play optimally.

Theorem: GG is PSPACE-complete

Games and Complexity

Geography game



Assume two players:

Player I and Player II

Players take turns picking places that start with the letter which ended the previous place. No repeats allowed. The first player stuck (= cannot move) loses.

Generalized Geography Game

Played on any directed graph.
Players take turns picking nodes that form a simple path.
The first player stuck loses.

Defn: $GG = \{\langle G, a \rangle \mid$ Player I has a <u>forced win</u> in Generalized Geography on graph G starting at node $a\}$.

"forced win" also called a "winning strategy" means that the player will win if both players play optimally.

Theorem: GG is PSPACE-complete

Check-in 19.1

Games and Complexity

Geography game

Check-in 19.1

Let G be the graph below.

Which player has a winning strategy in the Generalized Geography game starting at node a?

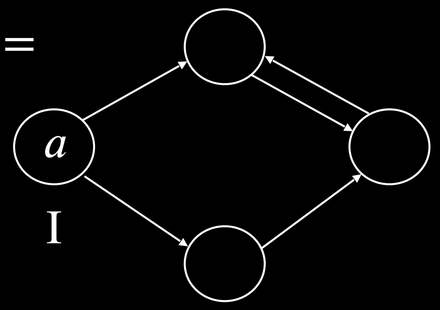
(a) Player I

(b) Player II

(c) Neither player

(d) Both players

$G =$



Generalized Geography Game

Played on any directed graph.
Players take turns picking nodes that form a simple path.
The first player stuck loses.

Defn: $GG = \{\langle G, a\rangle \mid$ Player I has a forced win in Generalized Geography on graph G starting at node $a\}$.

"forced win" also called a "winning strategy" means that the player will win if both players play optimally.

Theorem: GG is PSPACE-complete

Check-in 19.1

Games and Quantifiers

The Formula Game

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots (\exists / \forall) x_k \overbrace{\left[(\cdots) \land \cdots \land (\cdots) \right]}^{\psi}$

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots (\exists / \forall) x_k \overbrace{\left[(\cdots) \wedge \cdots \wedge (\cdots) \right]}^{\psi}$

There are two Players "∃" and "∀".

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots (\exists/\forall) x_k \overbrace{\left[(\ \cdots) \wedge \cdots \wedge (\ \cdots) \right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots (\exists/\forall) x_k \overbrace{\left[(\cdots) \wedge \cdots \wedge (\cdots) \right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots (\exists / \forall) x_k \overbrace{\left[(\ \cdots) \land \cdots \land (\ \cdots) \right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots (\exists / \forall) x_k \ \overbrace{\left[(\ \cdots) \wedge \cdots \wedge (\ \cdots) \right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:

Player ∃ wins if the assignment satisfies ψ.

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots (\exists/\forall) x_k \overbrace{\left[\, (\ \cdots) \wedge \cdots \wedge (\ \cdots) \,\right]}^{\psi}$

There are two Players “∃” and “∀”.

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:

Player ∃ wins if the assignment satisfies ψ.

Player ∀ wins if not.

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots (\exists/\forall) x_k \ \overbrace{\left[\, (\,\cdots) \wedge \cdots \wedge (\,\cdots) \,\right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:

Player ∃ wins if the assignment satisfies ψ.

Player ∀ wins if not.

Claim: Player ∃ has a forced win in the formula game on ϕ iff ϕ is TRUE.

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots (\exists/\forall) x_k \overbrace{\Big[(\ \cdots) \wedge \cdots \wedge (\ \cdots) \Big]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:

Player ∃ wins if the assignment satisfies ψ.

Player ∀ wins if not.

Claim: Player ∃ has a forced win in the formula game on ϕ iff ϕ is TRUE.

Therefore $\{\langle\phi\rangle \mid \text{Player } \exists \text{ has a forced win on } \phi\} = TQBF$.

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots (\exists/\forall) x_k \overbrace{\left[\, (\ \cdots) \land \cdots \land (\ \cdots) \,\right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.
Player ∀ assigns values to the ∀-quantified variables.
The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:
Player ∃ wins if the assignment satisfies ψ.
Player ∀ wins if not.

Claim: Player ∃ has a forced win in the formula game on ϕ iff ϕ is TRUE.

Therefore $\{\langle\phi\rangle \mid \text{Player } \exists \text{ has a forced win on } \phi\} = TQBF$.

Next: show $TQBF \leq_P GG$.

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots (\exists/\forall) x_k\ \overbrace{\left[\,(\ \cdots) \wedge \cdots \wedge (\ \cdots)\,\right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:

Player ∃ wins if the assignment satisfies ψ.

Player ∀ wins if not.

Claim: Player ∃ has a forced win in the formula game on ϕ iff ϕ is TRUE.

Therefore $\{\langle\phi\rangle \mid$ Player ∃ has a forced win on $\phi\} = TQBF$.

Next: show $TQBF \leq_P GG$.

Check-in 19.2

Games and Quantifiers

The Formula Game

Given QBF $\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots (\exists/\forall) x_k \overbrace{\left[\, (\,\cdots) \wedge \cdots \wedge (\,\cdots) \, \right]}^{\psi}$

There are two Players "∃" and "∀".

Player ∃ assigns values to the ∃-quantified variables.

Player ∀ assigns values to the ∀-quantified variables.

The players choose the values according to the order of the quantifiers in ϕ.

After all variables have been assigned values, we determine the winner:

Player ∃ wins if the assignment satisfies ψ.

Player ∀ wins if not.

Claim: Player ∃ has a forced win in the formula game on ϕ iff ϕ is TRUE.

Therefore $\{\langle\phi\rangle \mid \text{Player } \exists \text{ has a forced win on } \phi\} = TQBF$.

Next: show $TQBF \leq_P GG$.

Check-in 19.2

Which player has a winning strategy in the formula game on

$$\phi \ = \ \exists x \ \forall y \left[(x \vee y) \wedge (\overline{x} \vee \overline{y}) \right]$$

(a) ∃-player

(b) ∀-player

(c) Neither player

GG is PSPACE-complete

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

Give reduction f from $TQBF$ to GG. $f\big(\langle \phi \rangle\big) = \langle G, a \rangle$

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

Give reduction f from $TQBF$ to GG. $f\big(\langle\phi\rangle\big) = \langle G, a\rangle$

Construct G to mimic the formula game on ϕ.

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

Give reduction f from $TQBF$ to GG. $f\big(\langle\phi\rangle\big) = \langle G, a\rangle$

Construct G to mimic the formula game on ϕ.

G has Players I and II

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

Give reduction f from $TQBF$ to GG. $f\big(\langle\phi\rangle\big) = \langle G, a\rangle$

Construct G to mimic the formula game on ϕ.

G has Players I and II

Player I plays role of ∃-Player in ϕ. Ditto for Player II and the ∀-Player.

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

Give reduction f from $TQBF$ to GG. $f(\langle \phi \rangle) = \langle G, a \rangle$

Construct G to mimic the formula game on ϕ.

G has Players I and II

Player I plays role of ∃-Player in ϕ. Ditto for Player II and the ∀-Player.

$$\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots (\exists/\forall) x_k \left[\underbrace{(\ \cdots) \wedge \cdots \wedge (\ \cdots)}_{\text{assume in cnf}} \right]$$

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

Give reduction f from $TQBF$ to GG. $f(\langle \phi \rangle) = \langle G, a \rangle$

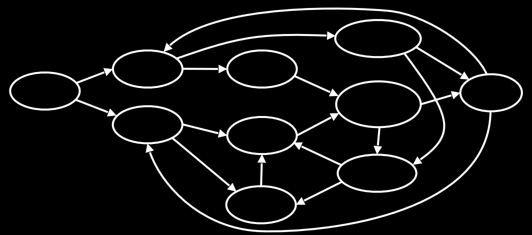
Construct G to mimic the formula game on ϕ.

G has Players I and II

Player I plays role of ∃-Player in ϕ. Ditto for Player II and the ∀-Player.

$$\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots (\exists/\forall) x_k \left[\underbrace{(\;\cdots) \wedge \cdots \wedge (\;\cdots)}_{\text{assume in cnf}} \right]$$

$\big\downarrow f$

$G =$ 

GG is PSPACE-complete

Theorem: GG is PSPACE-complete

Proof: 1) $GG \in$ PSPACE (recursive algorithm, exercise)

2) $TQBF \leq_{\mathrm{P}} GG$

Give reduction f from $TQBF$ to GG. $f(\langle\phi\rangle) = \langle G, a\rangle$

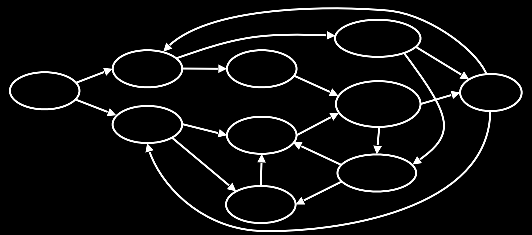
Construct G to mimic the formula game on ϕ.

G has Players I and II

Player I plays role of ∃-Player in ϕ. Ditto for Player II and the ∀-Player.

$$\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots (\exists/\forall) x_k \left[\underbrace{(\cdots) \wedge \cdots \wedge (\cdots)}_{\text{assume in cnf}} \right]$$

$$\downarrow f$$

$G =$ 

Constructing the GG graph G

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, (\, x_1 \vee \overline{x_2} \vee x_3 \,) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge \cdots \wedge (\, \cdots) \,]$

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3 \,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k} \,]$

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3 \,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k} \,]$

I = ∃ II = ∀

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3 \,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k} \,]$

$G =$

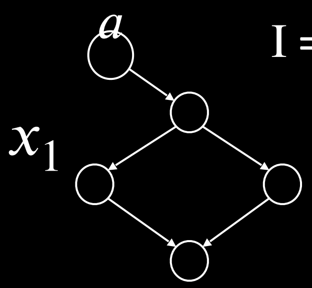
I = $\exists$ II = $\forall$

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$



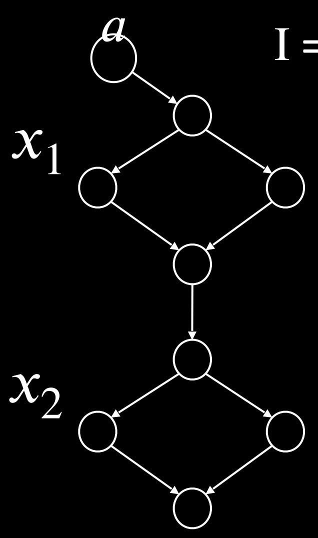
I = $\exists$ II = $\forall$

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k}\,]$

$G =$



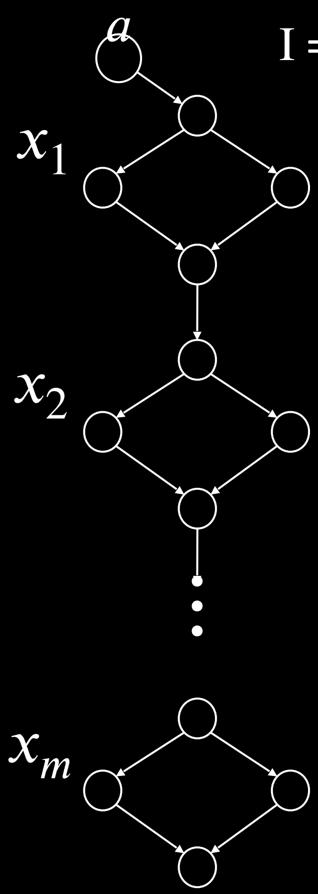
I = ∃ II = ∀

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\cdots)}_{c_k} \,]$

$G =$



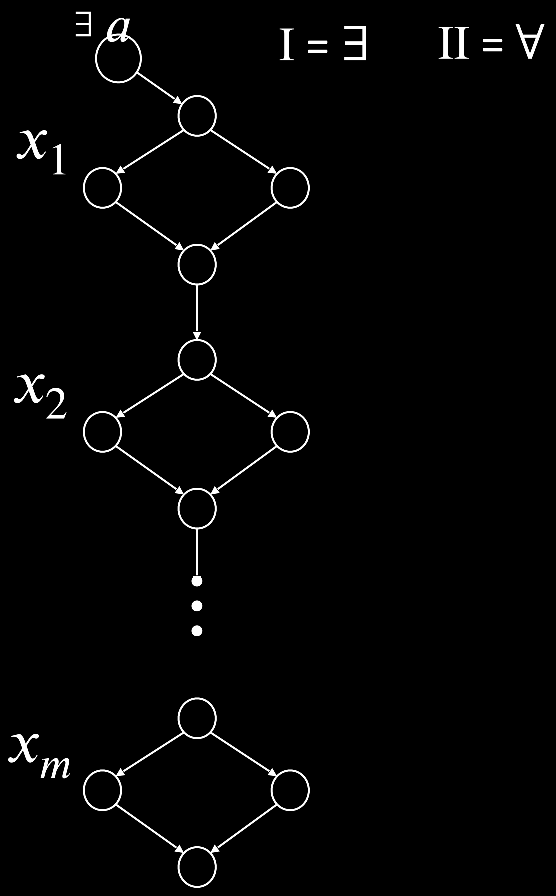
I = ∃ II = ∀

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

$G =$

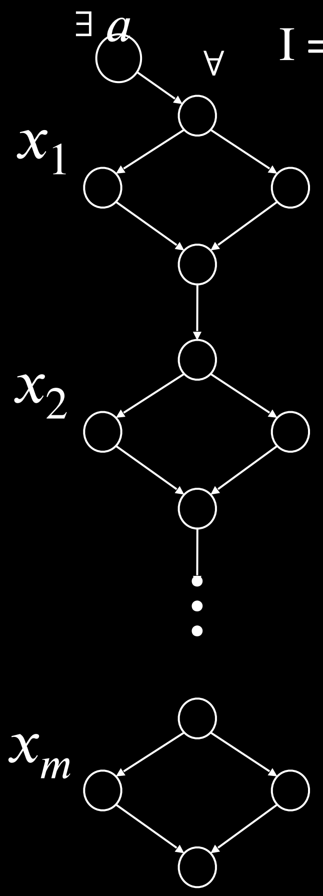


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$

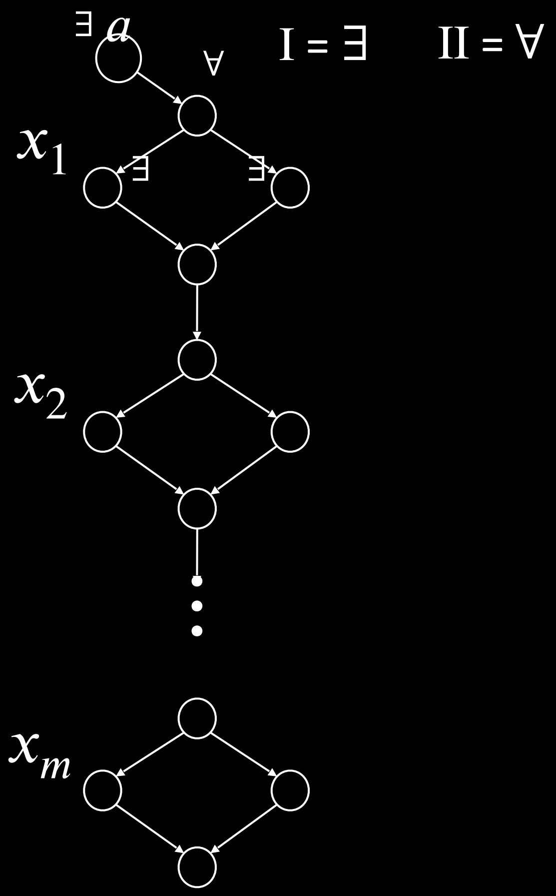


I = ∃ II = ∀

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

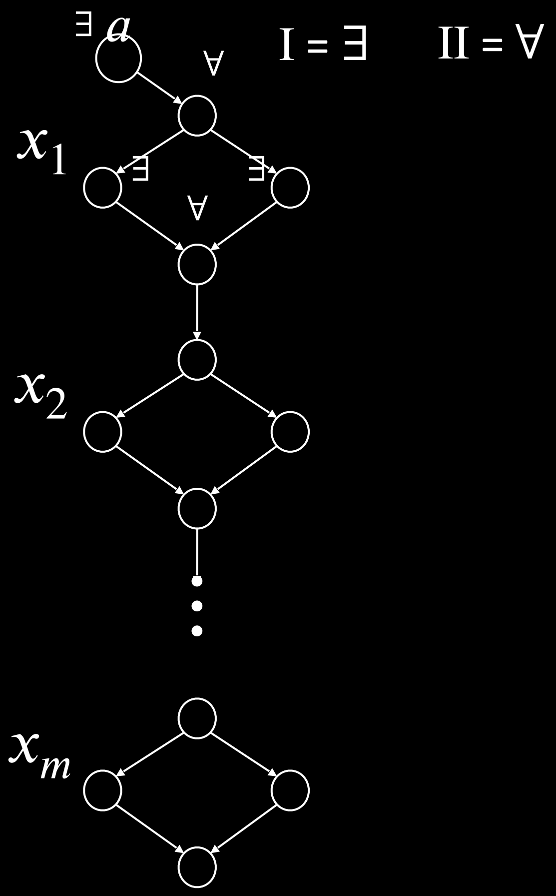
$G =$ 

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\,[\,(\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

$G =$

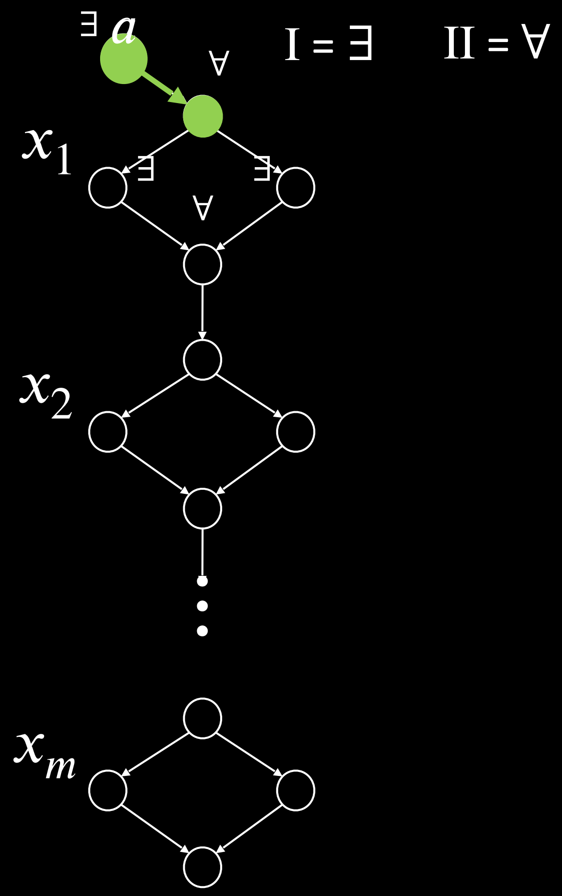


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\,[\,\underbrace{(\,x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\,\cdots)}_{c_k}\,]$

$G =$

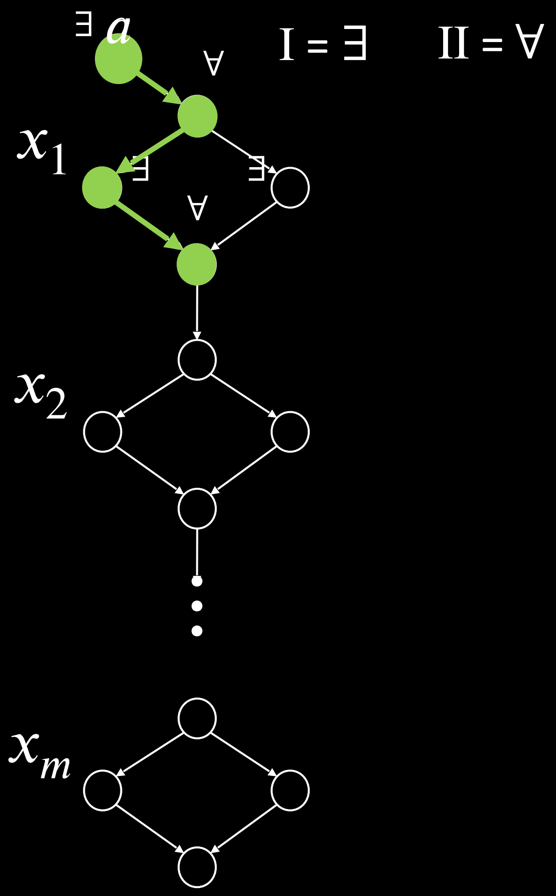


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$

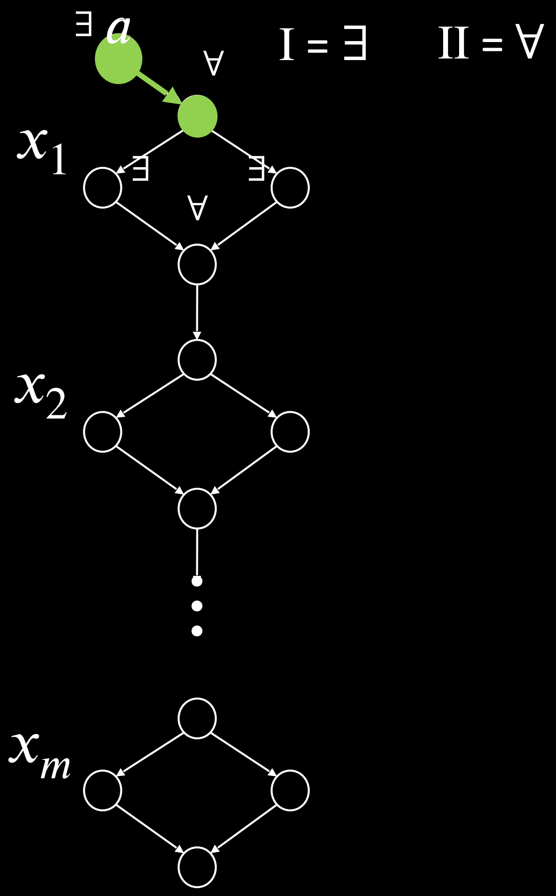


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \, \forall x_2 \, \exists x_3 \cdots \, \forall x_k \, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3 \,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k} \,]$

$G =$

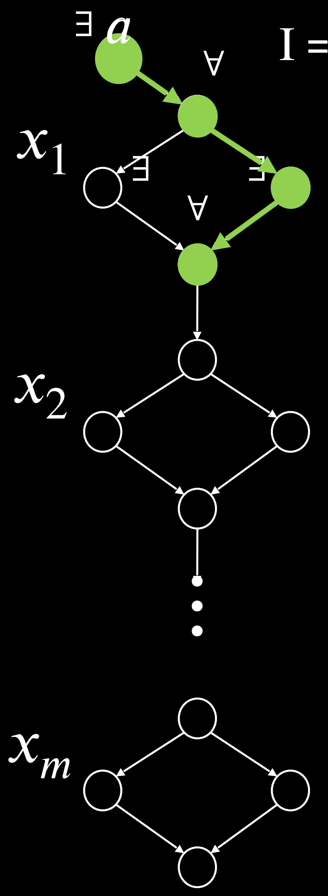


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$



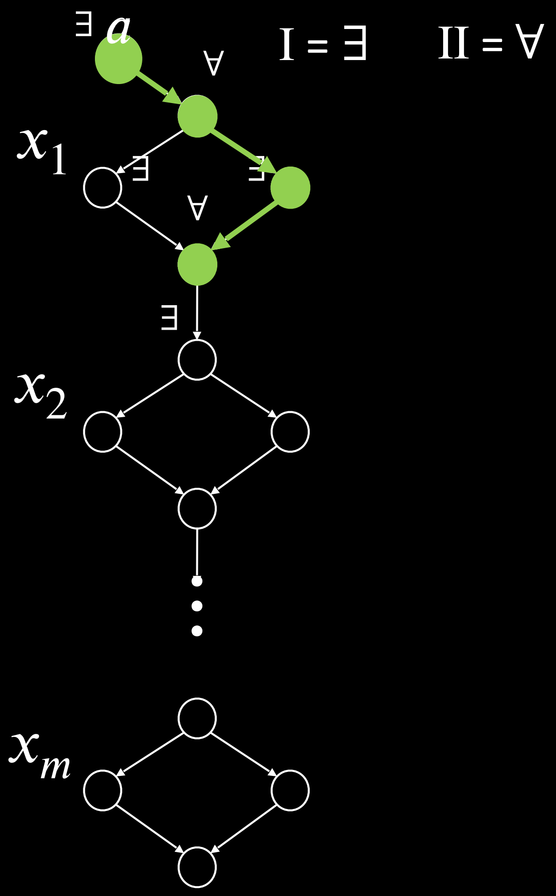
I = ∃ II = ∀

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\,[\,(\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

$G =$

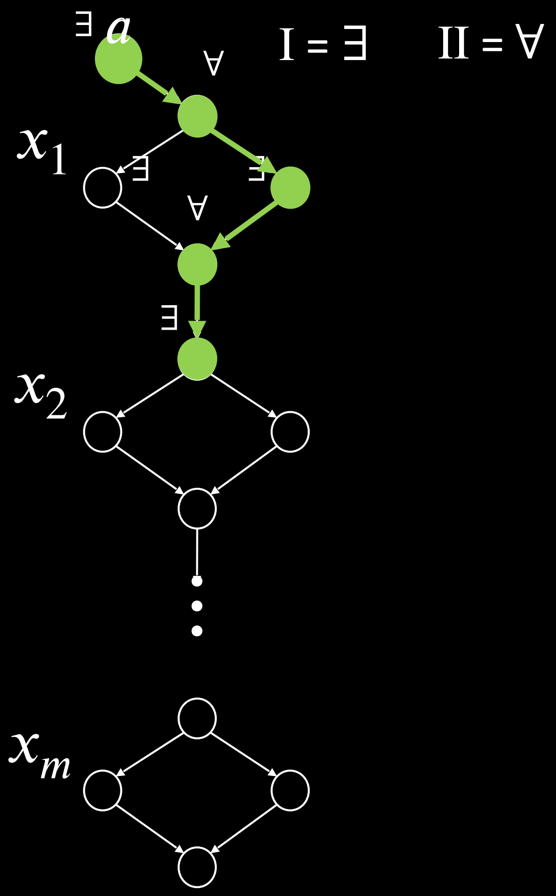


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k}) \,]$

$G =$

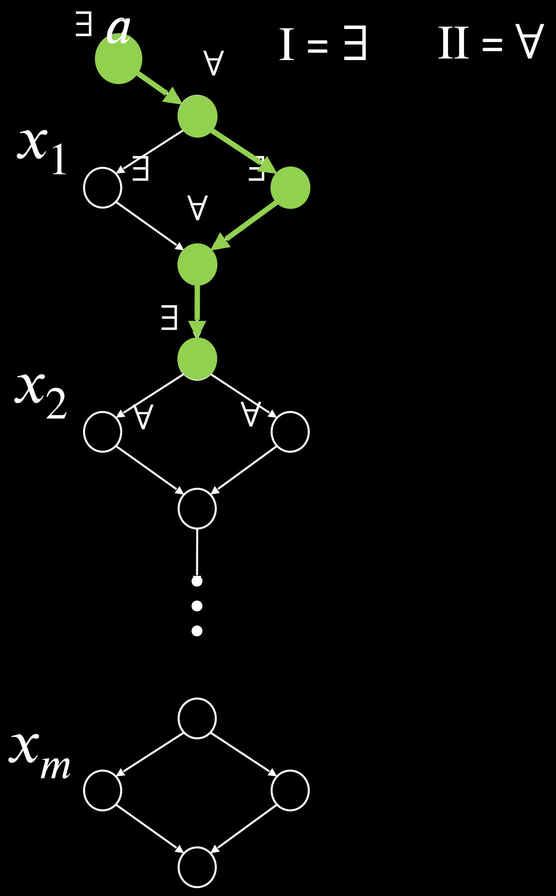


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k}\,]$

$G =$

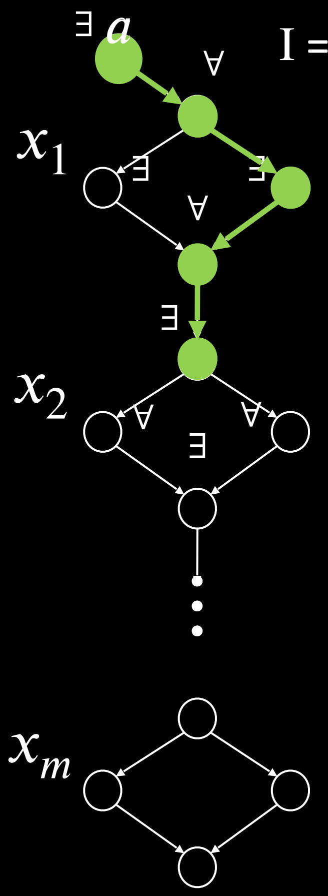


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\,[\,(\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

$G =$



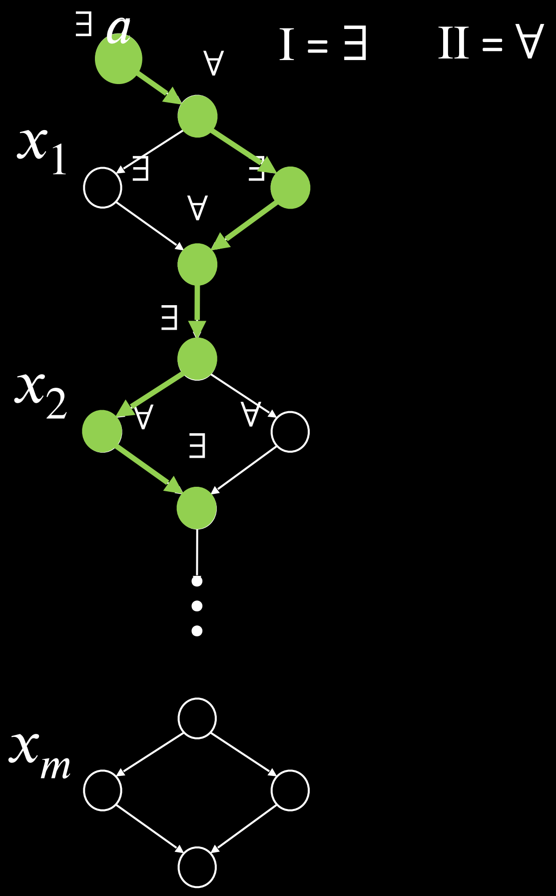
I = ∃ II = ∀

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$

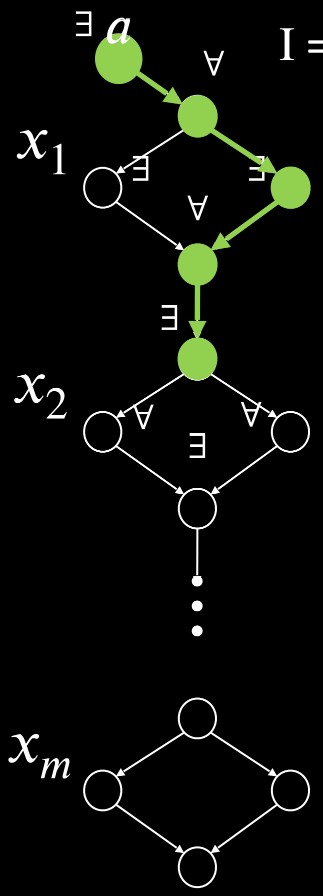


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots \ \forall x_k \, [\, \underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\cdots)}_{c_k} \,]$

$G =$



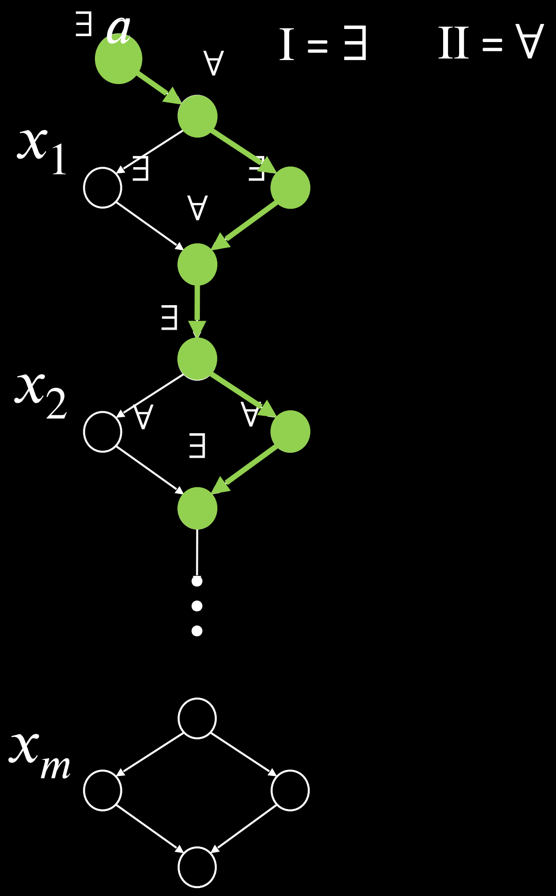
I = ∃ II = ∀

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \ \forall x_2 \ \exists x_3 \cdots \ \forall x_k \left[(\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k}) \right]$

$G =$

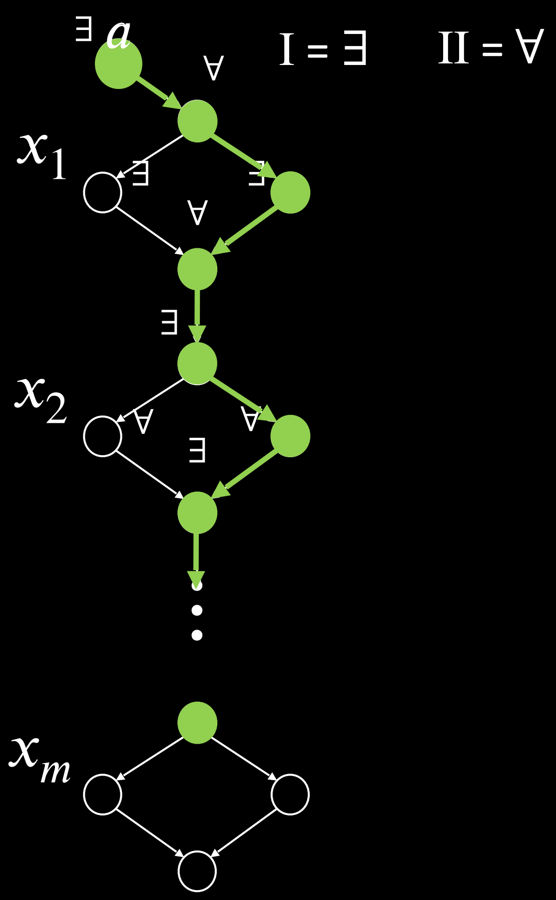


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k}\,]$

$G =$

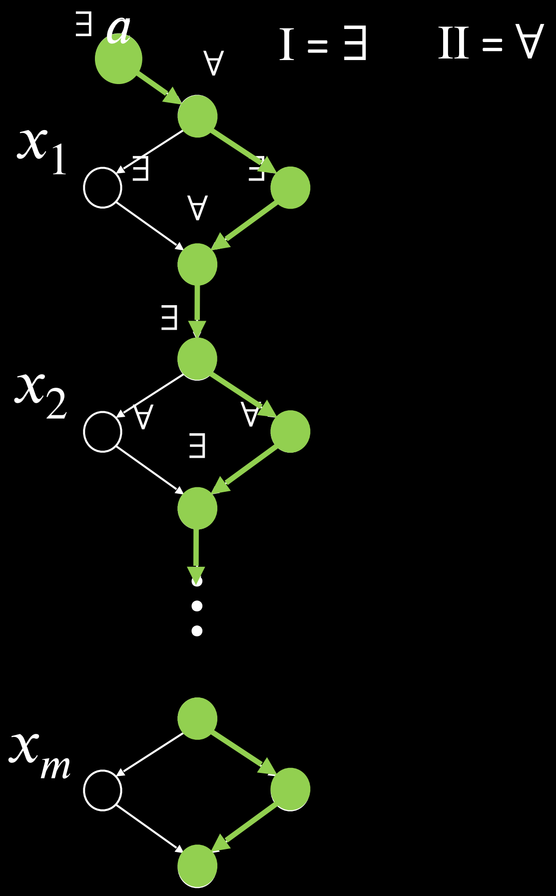


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k}) \,]$

$G =$

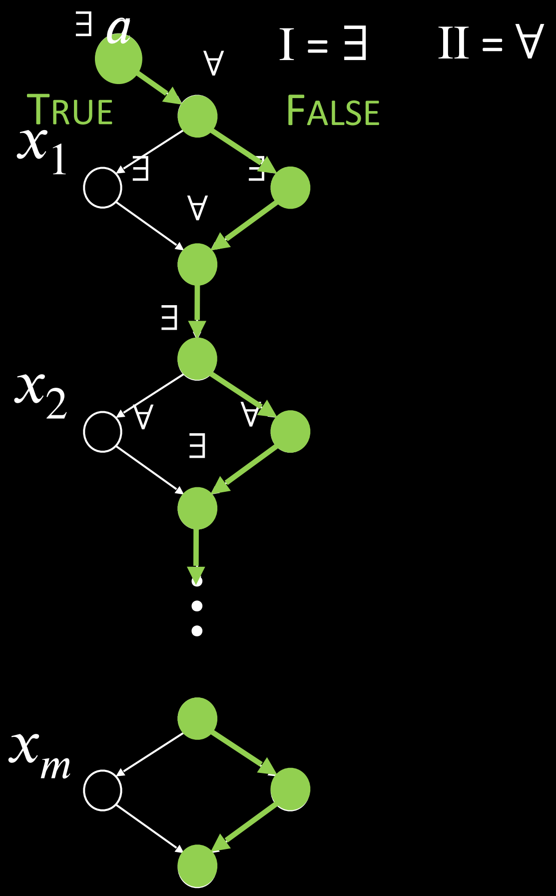


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k}\,]$

$G =$

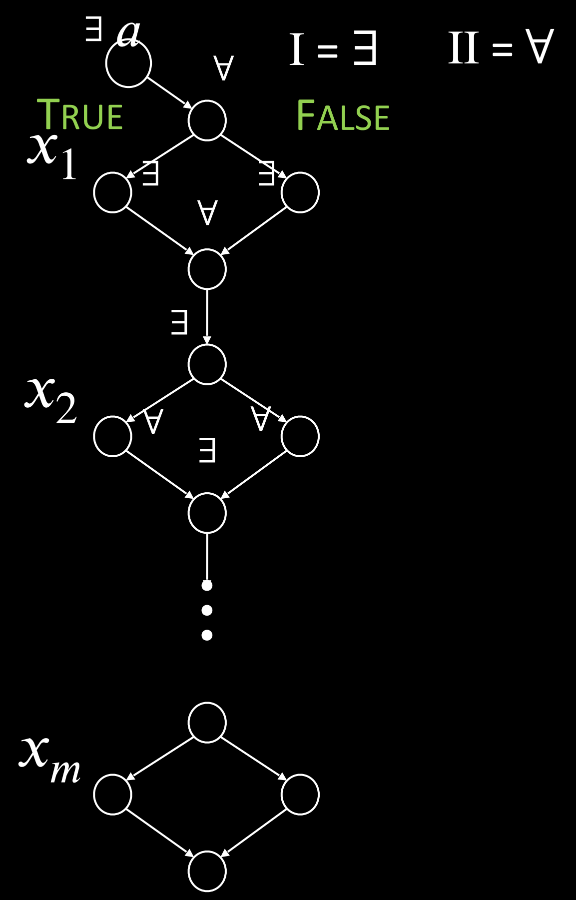


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3 \,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k} \,]$

$G =$

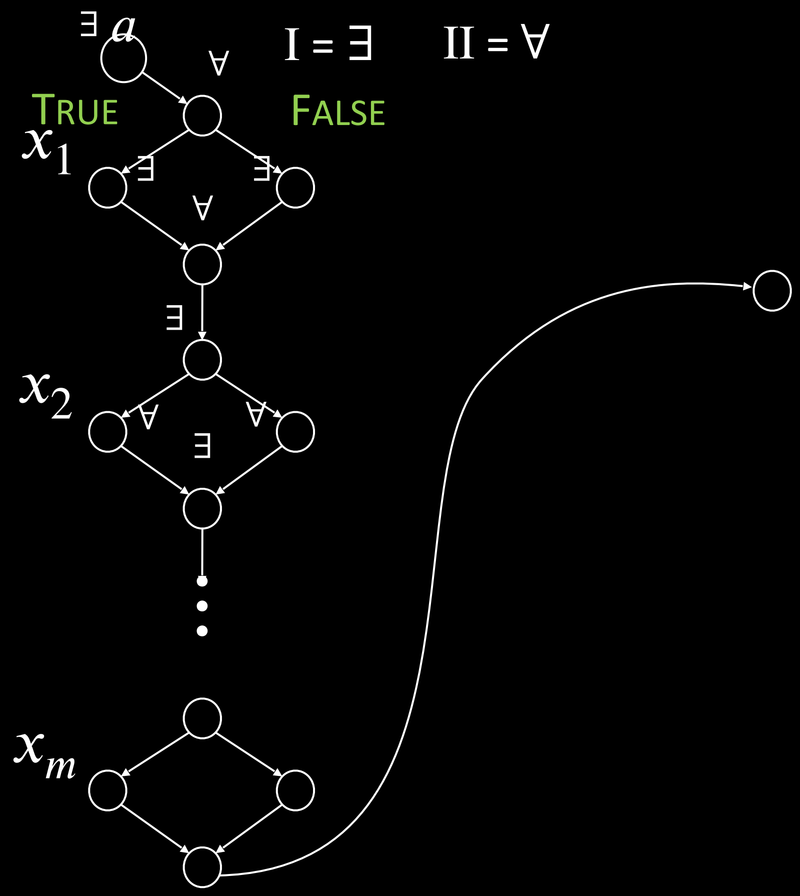


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$

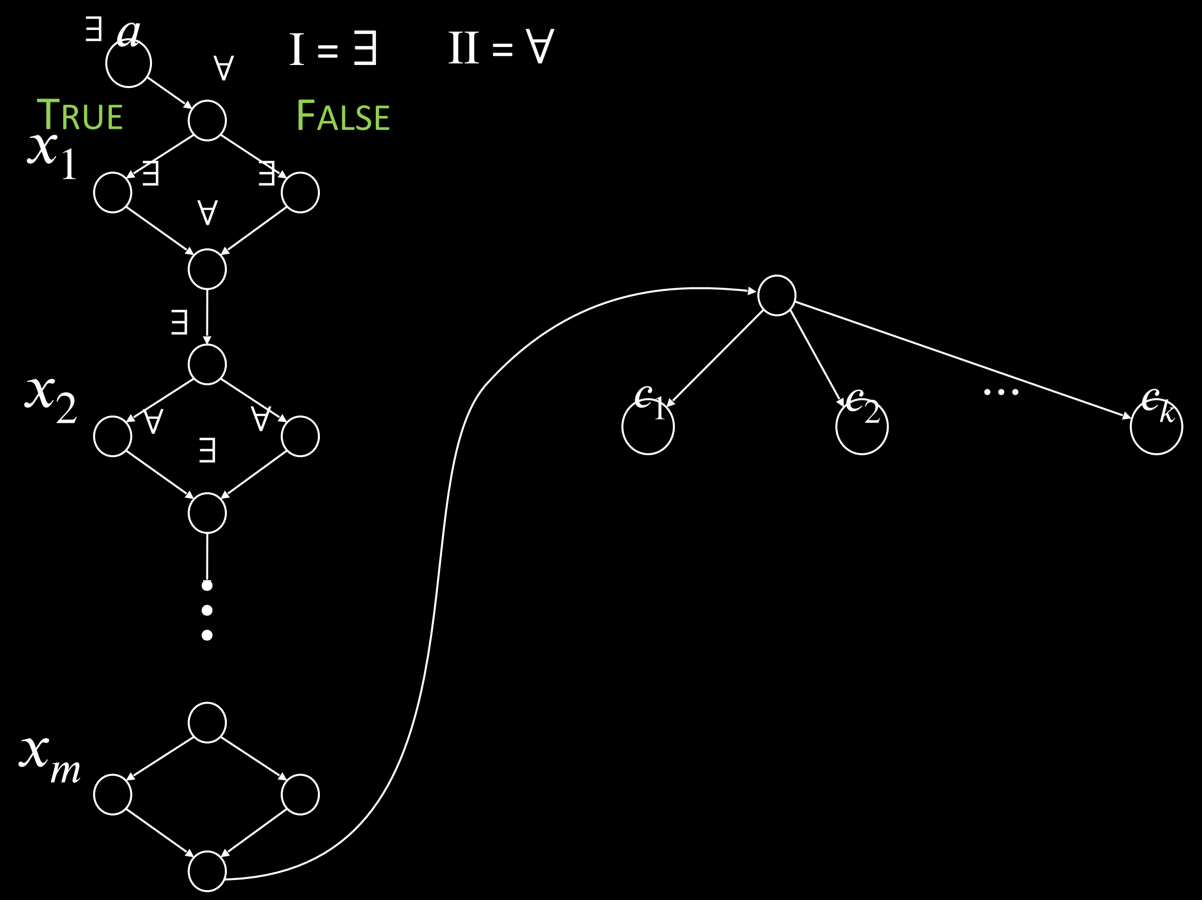


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\,]$

$G =$

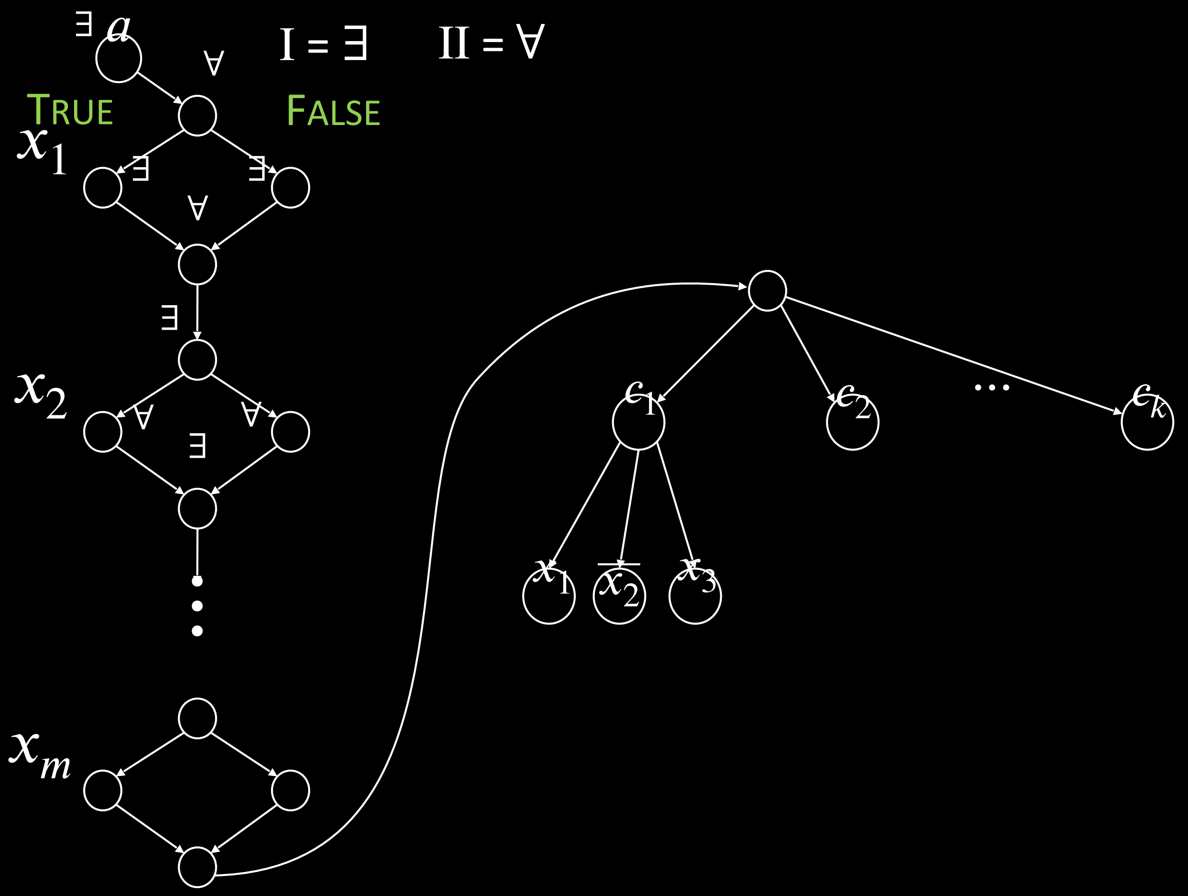


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$

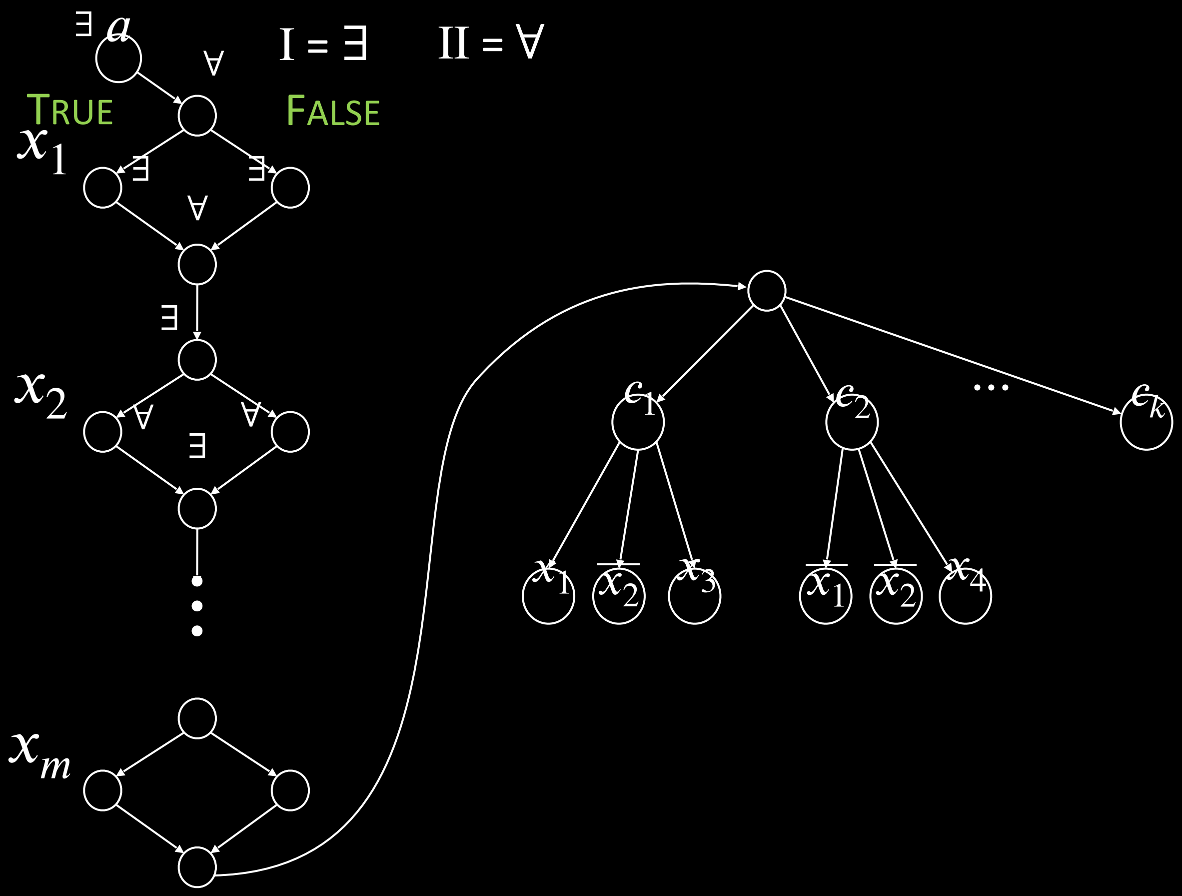


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

$G =$

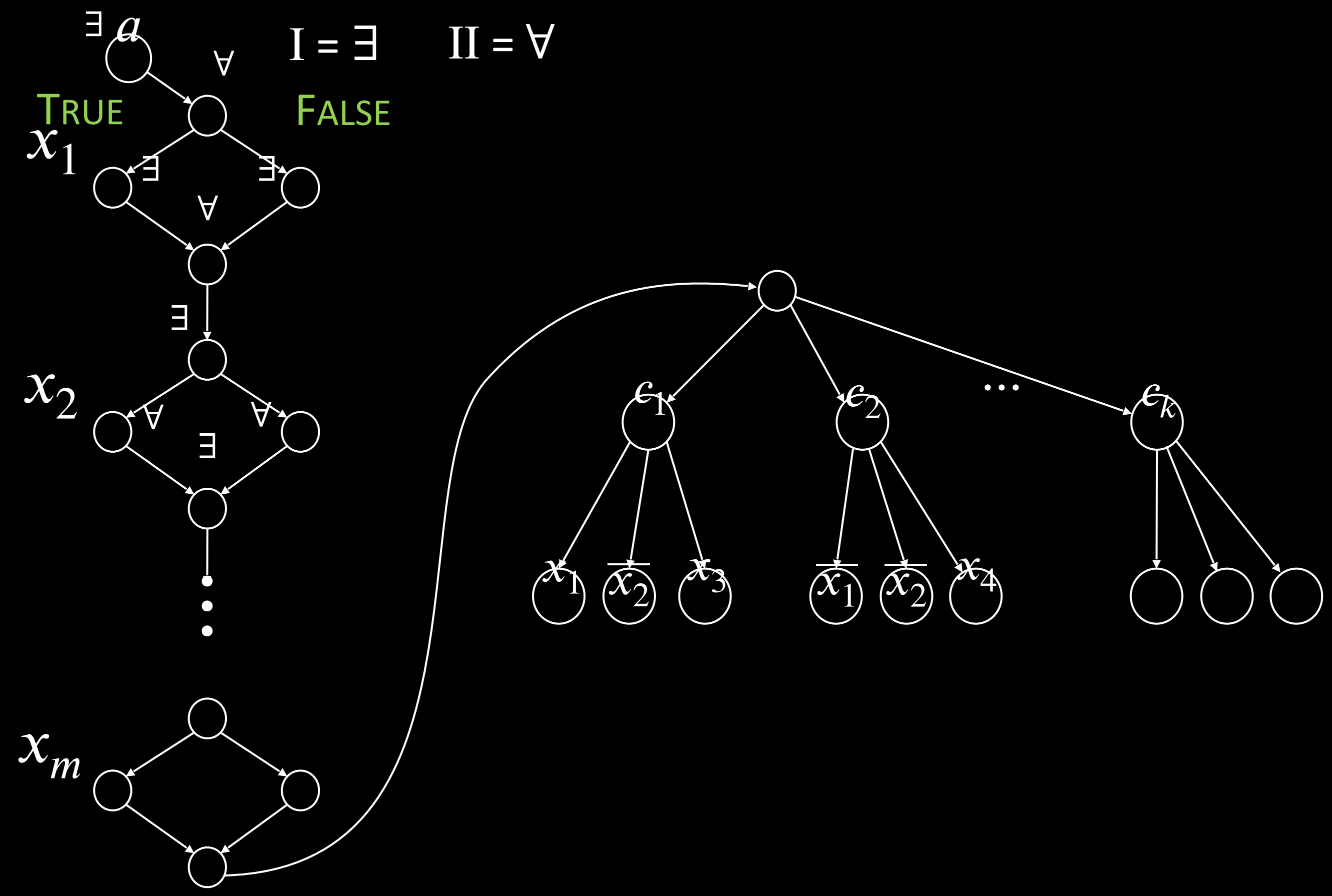


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\,[\,\underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\cdots)}_{c_k}\,]$

$G =$

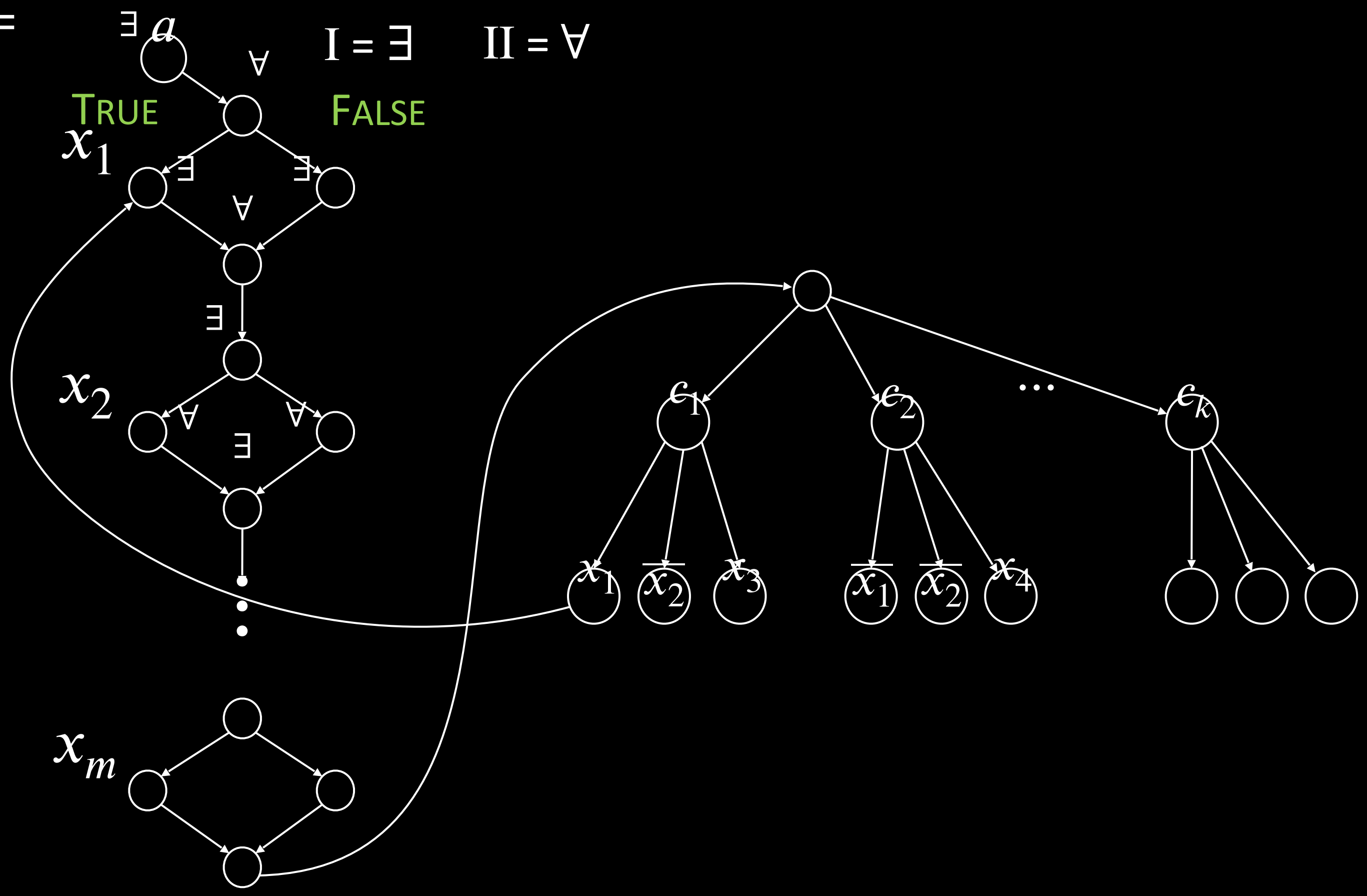


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3 \,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k} \,]$

$G =$

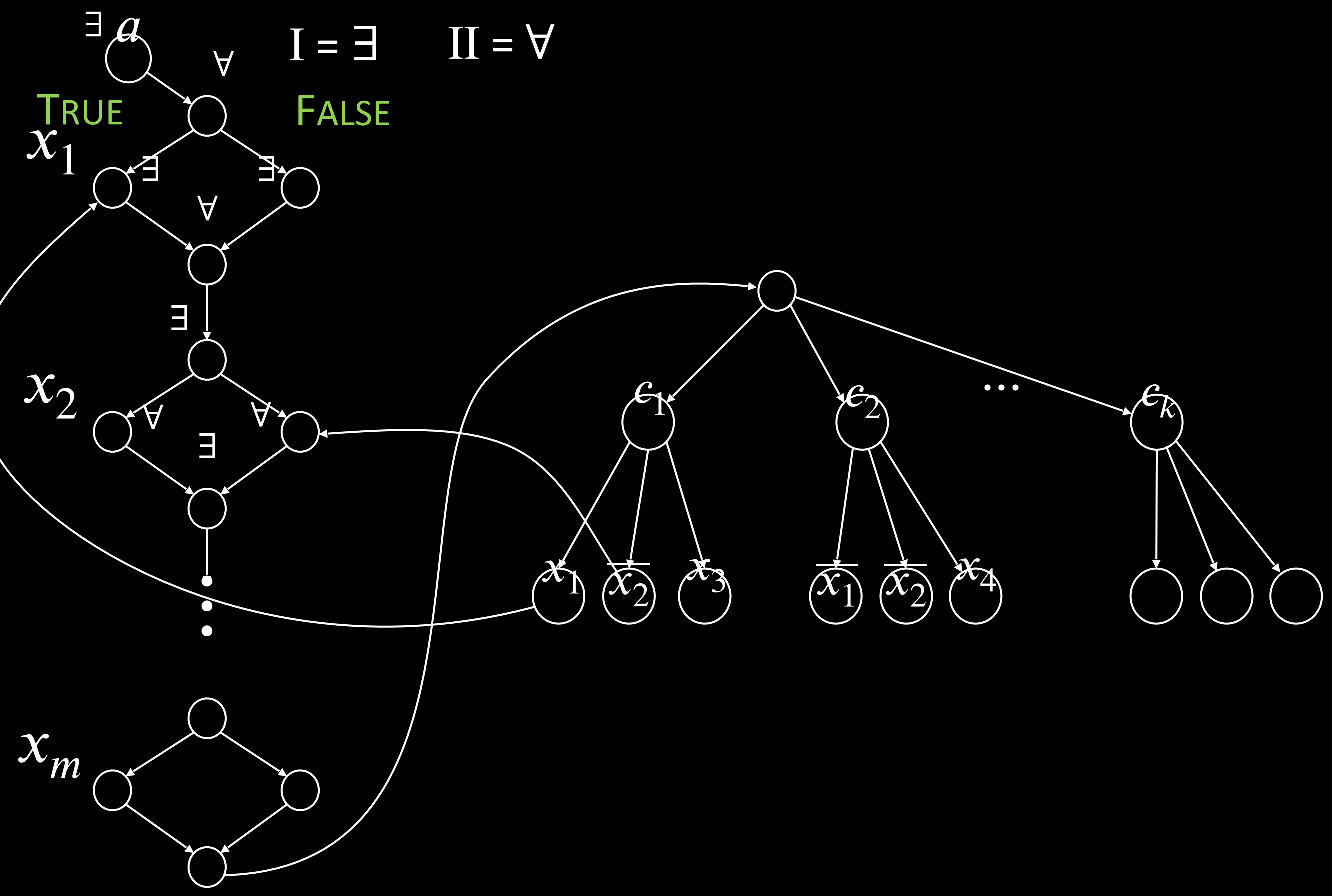


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\,[\,\underbrace{(\,x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\,\cdots)}_{c_k}\,]$

$G =$

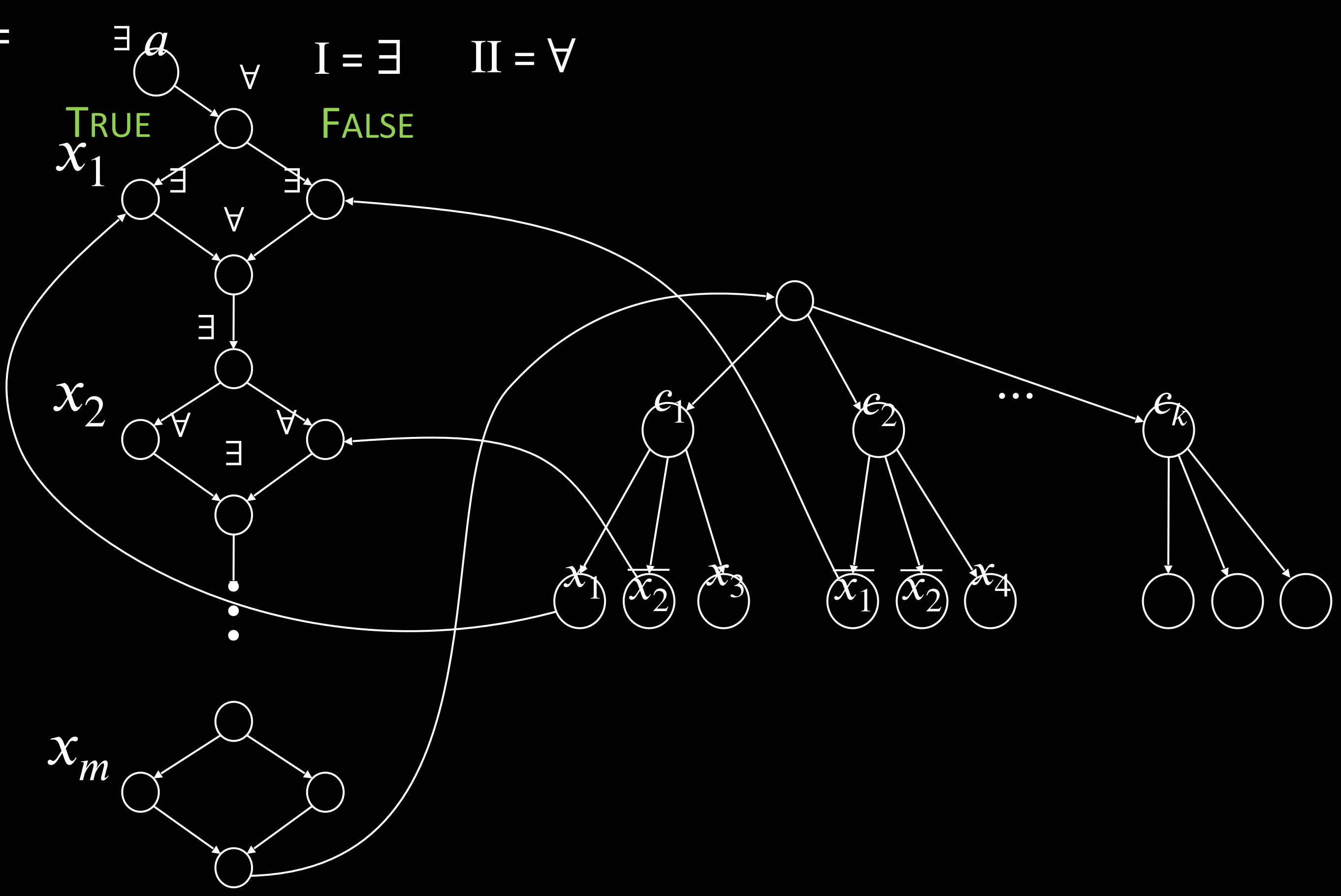


Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\cdots)}_{c_k} \,]$

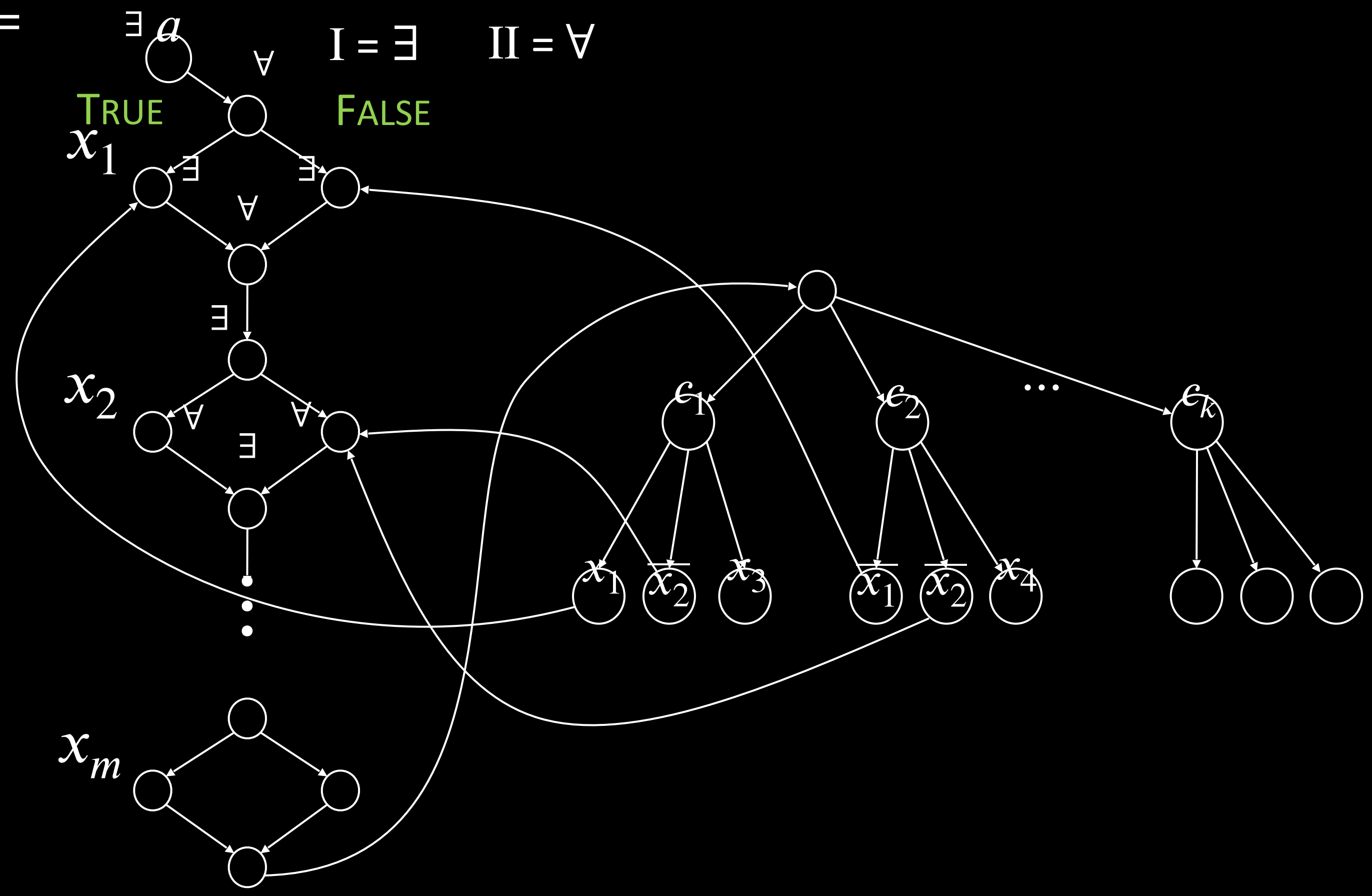
$G =$



Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

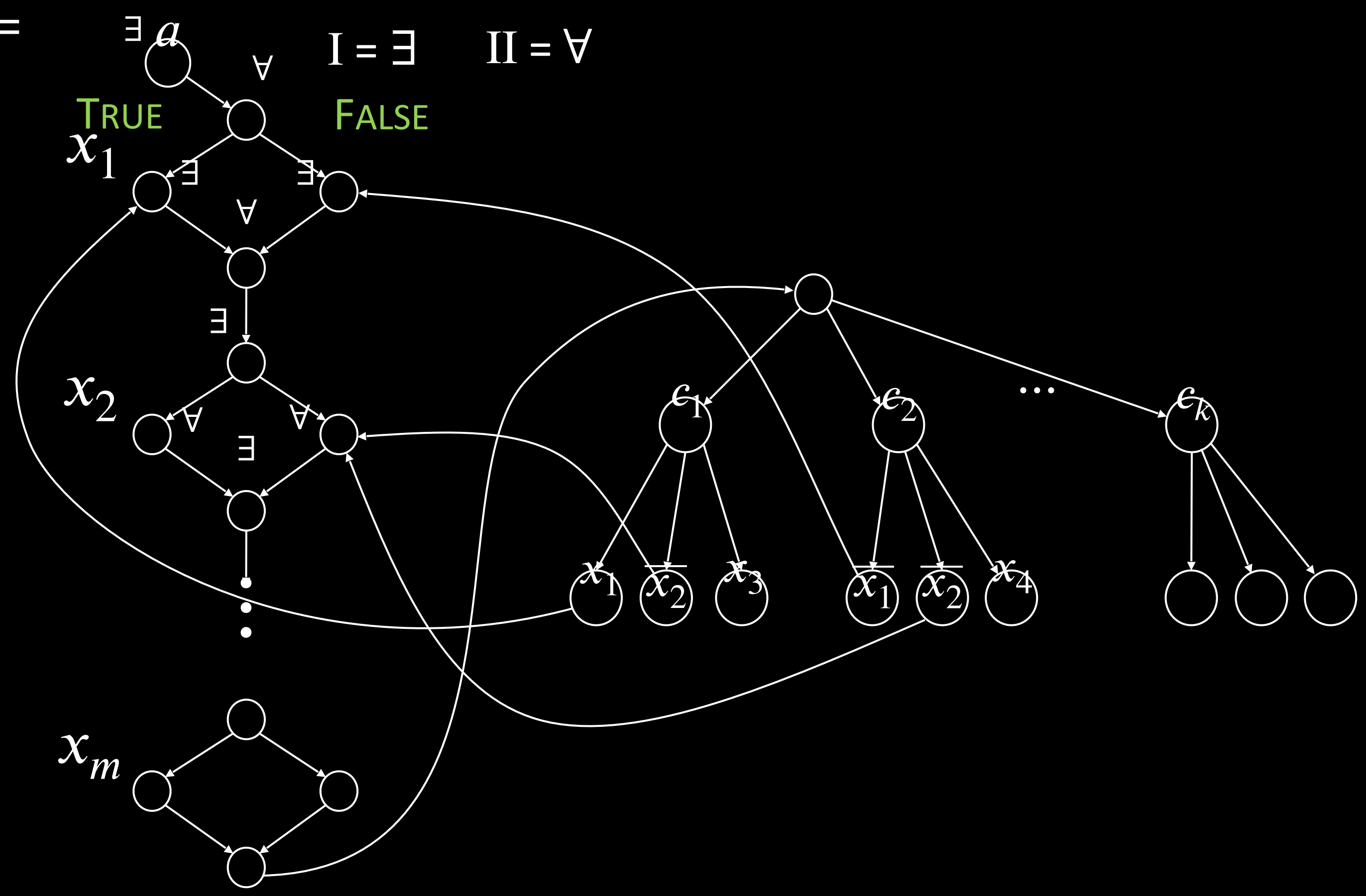
$G =$ 

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

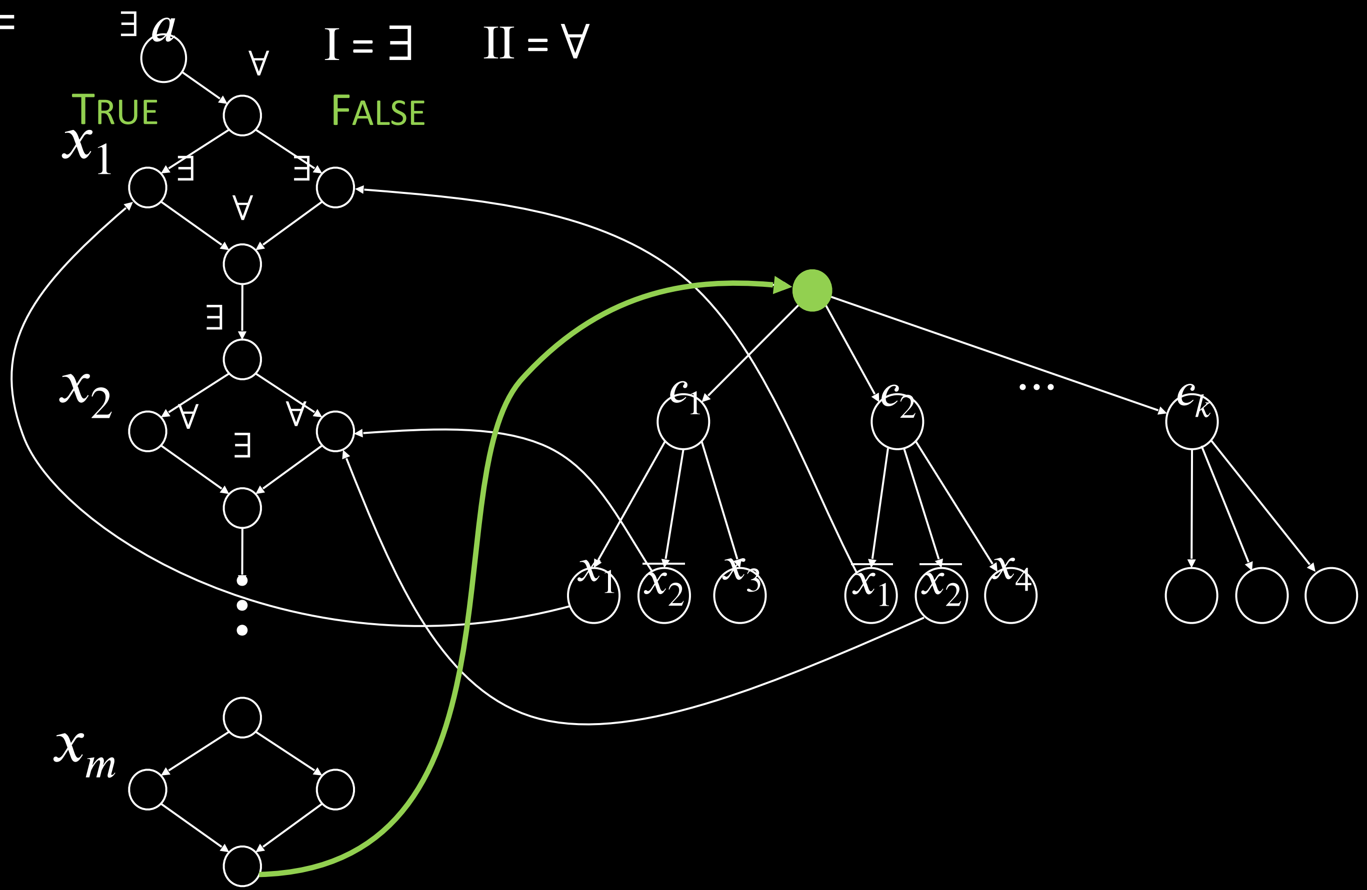
∀ should win if some unsatisfied clause

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

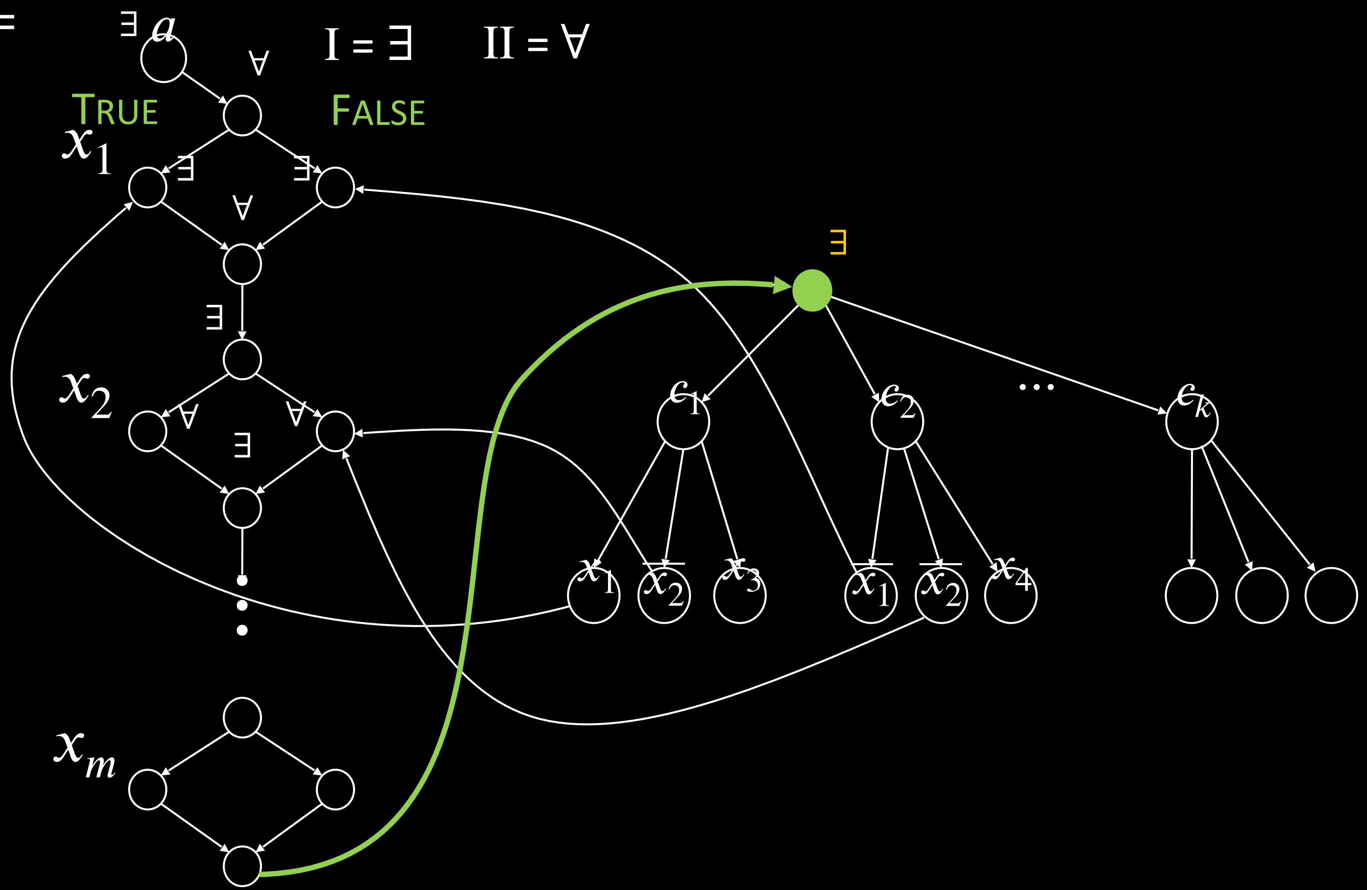
∀ should win if some unsatisfied clause

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\ ]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

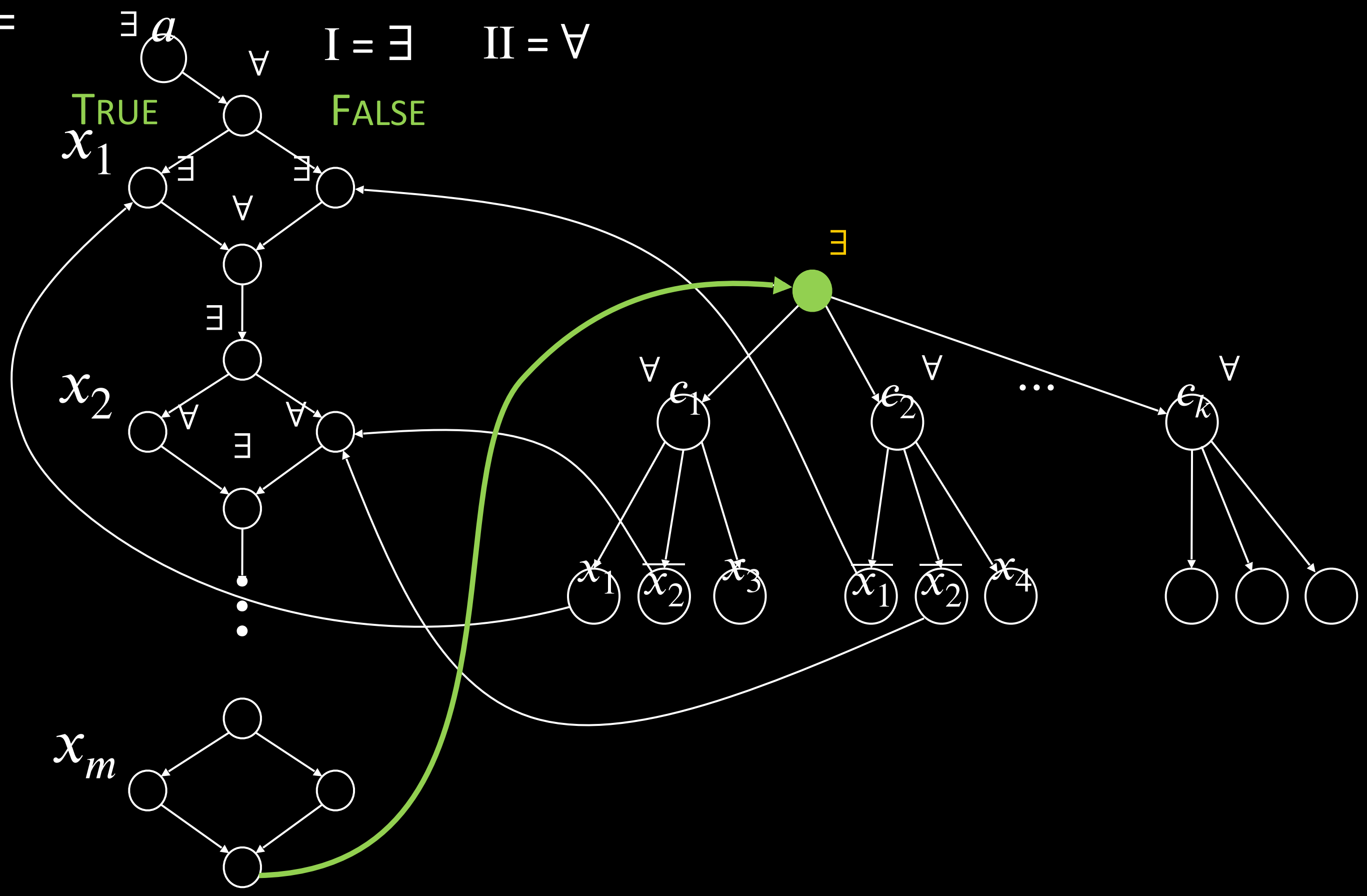
∀ should win if some unsatisfied clause

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\,\cdots)}_{c_k}\,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

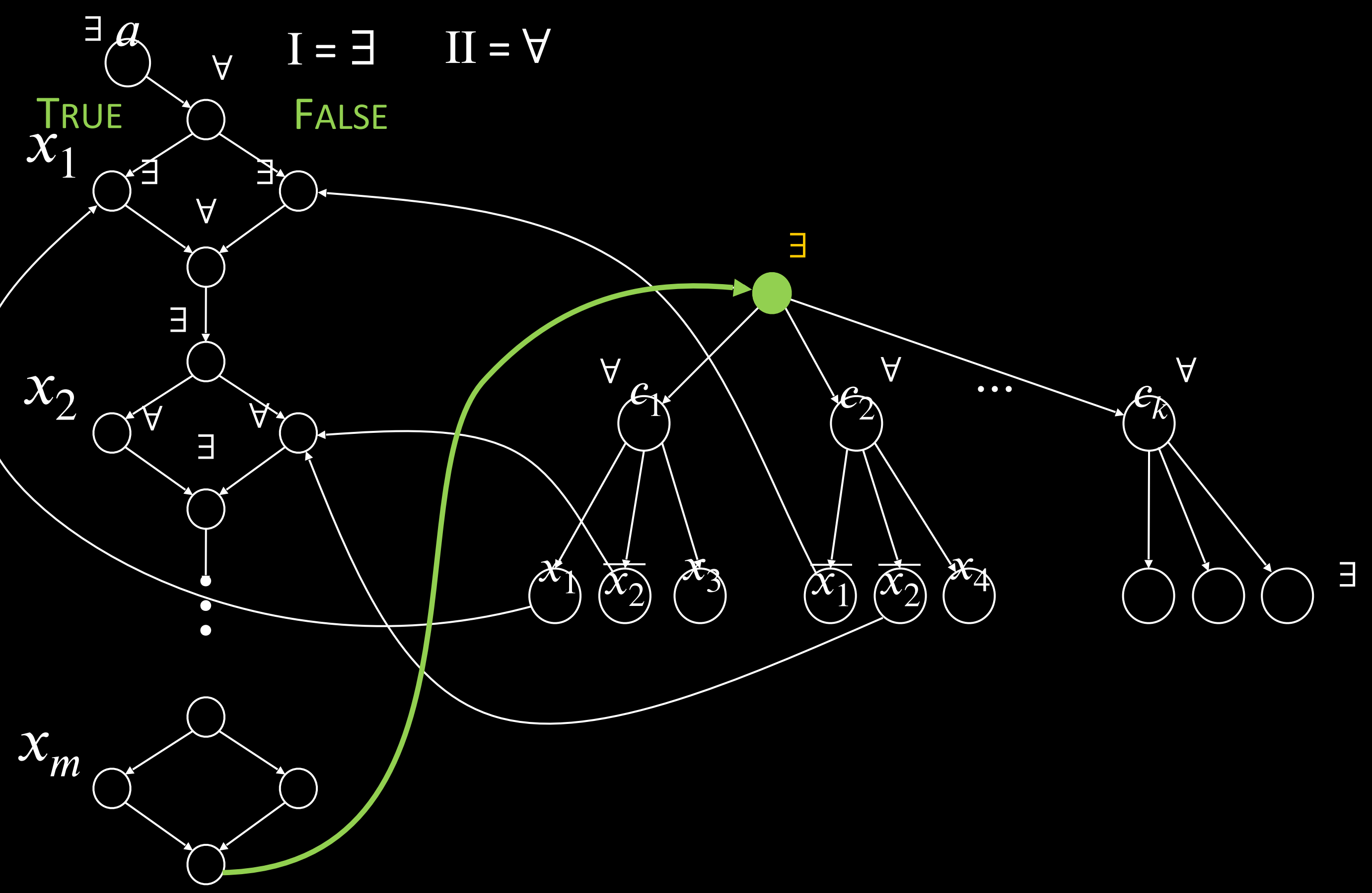
∀ should win if some unsatisfied clause

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k})\,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

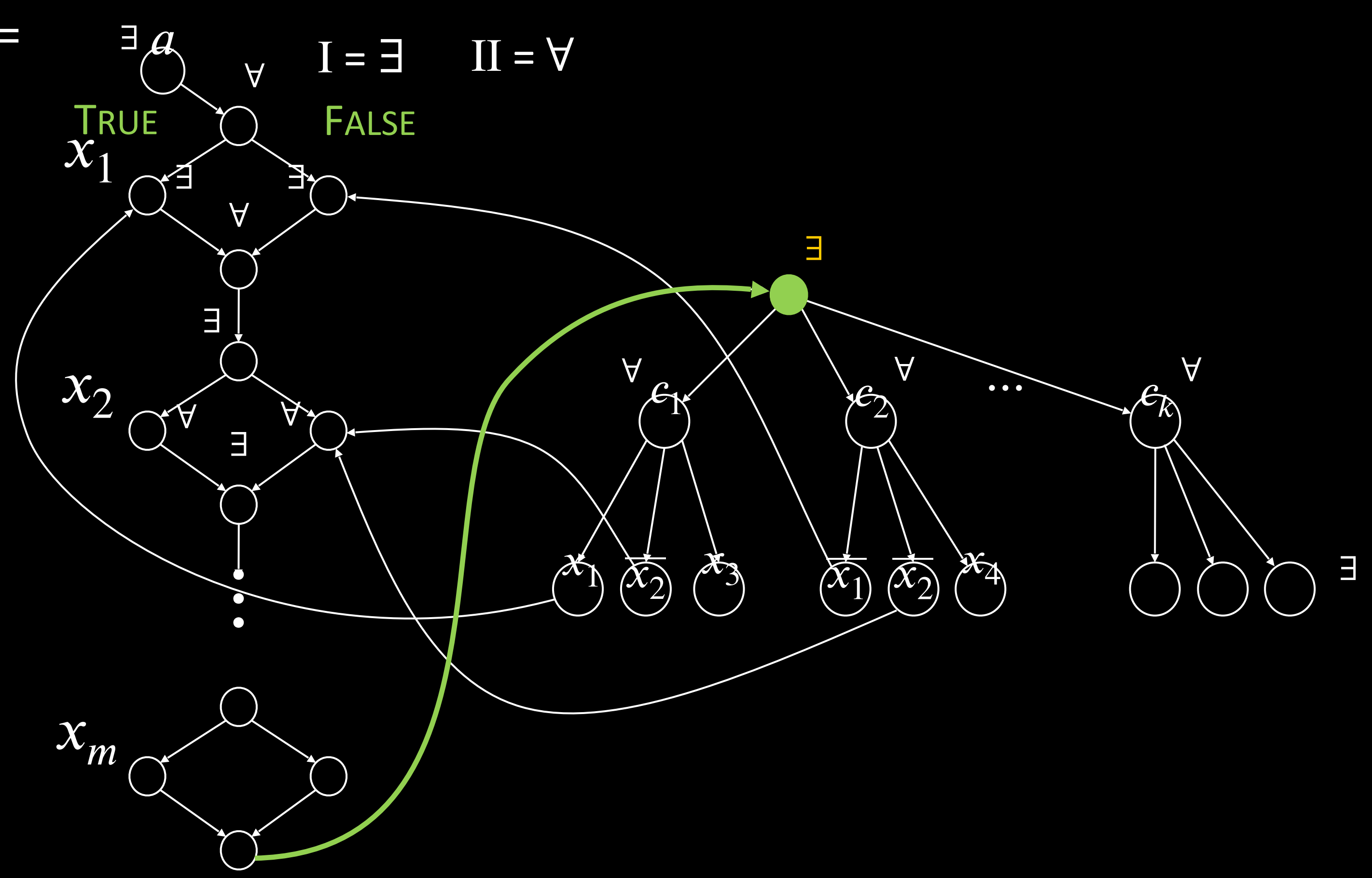
∀ should win if some unsatisfied clause

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k}\,]$

$G =$



Endgame

$\exists$ should win if assignment satisfied all clauses

$\forall$ should win if some unsatisfied clause

Implementation

$\forall$ picks clause node claimed unsatisfied

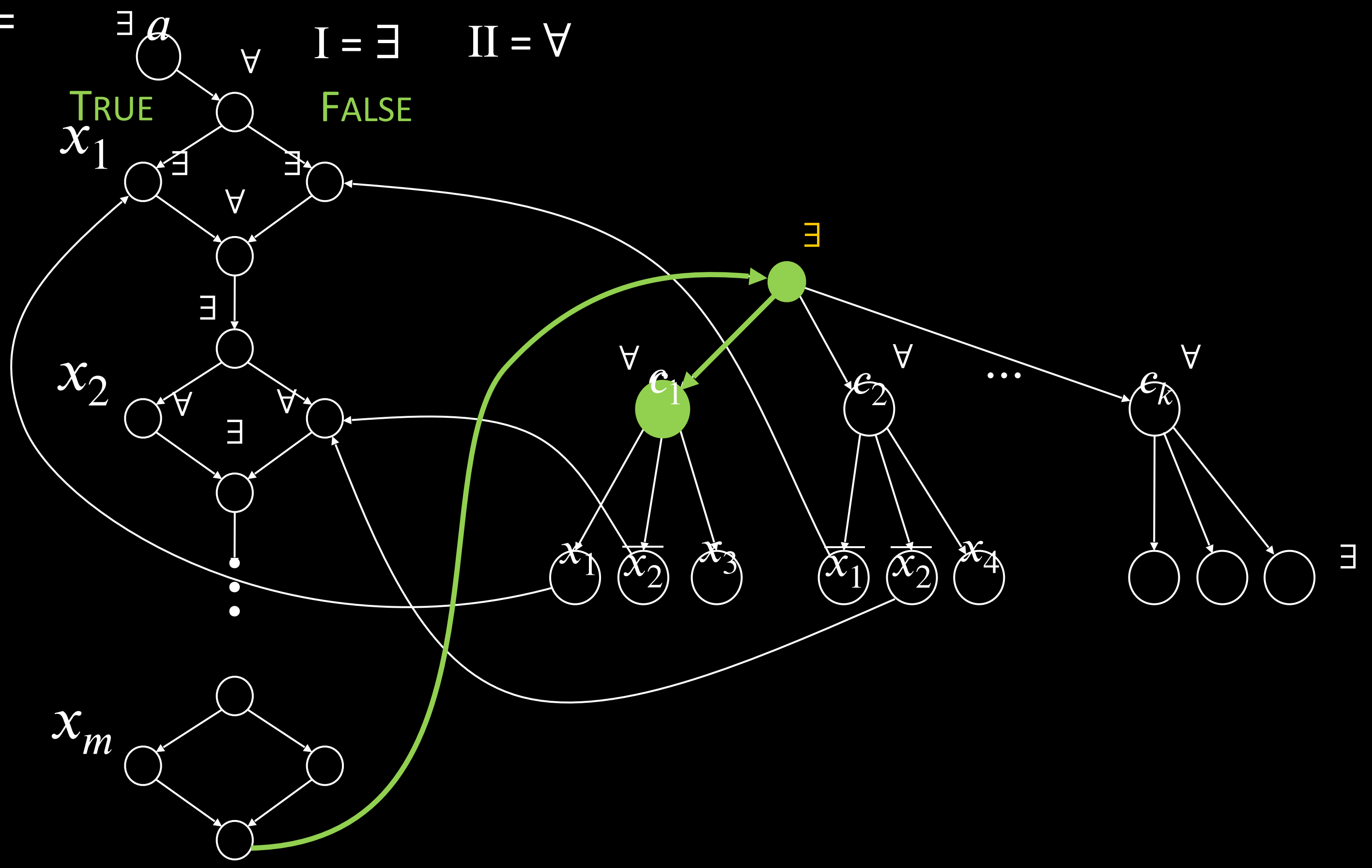
$\exists$ picks literal node claimed to satisfy the clause

liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$ 

Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

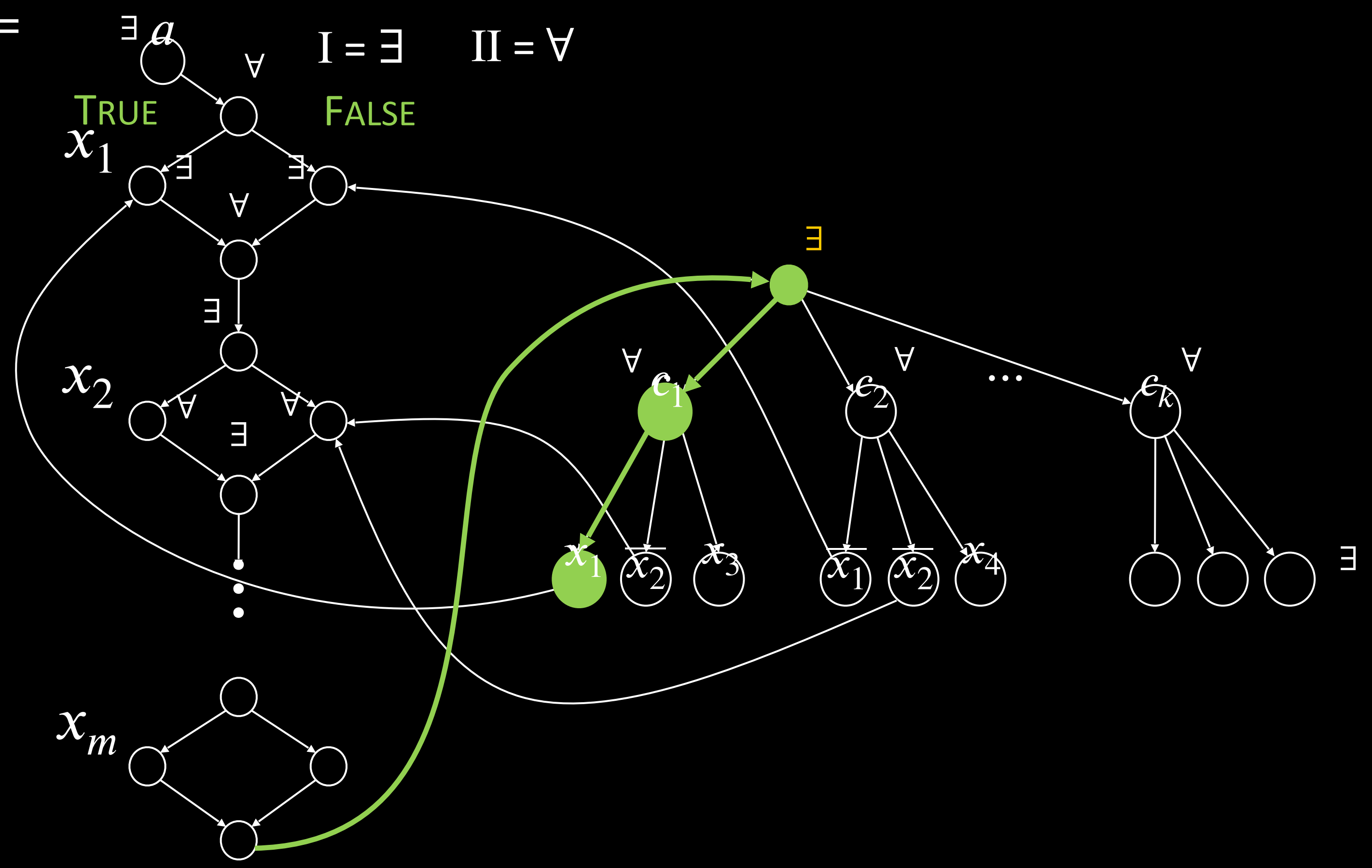
liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

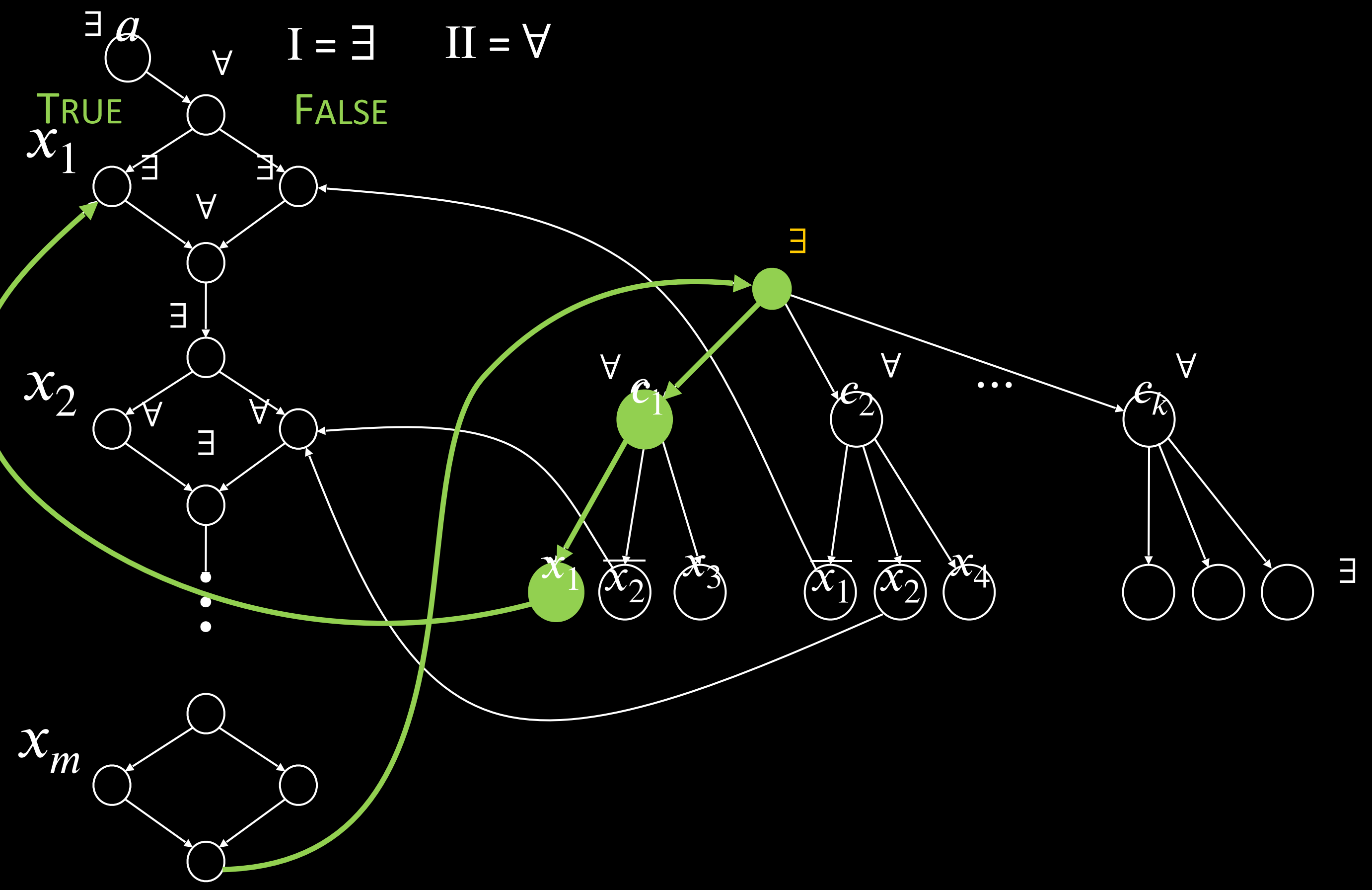
liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\,\underbrace{(\,x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\,\cdots)}_{c_k}\,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

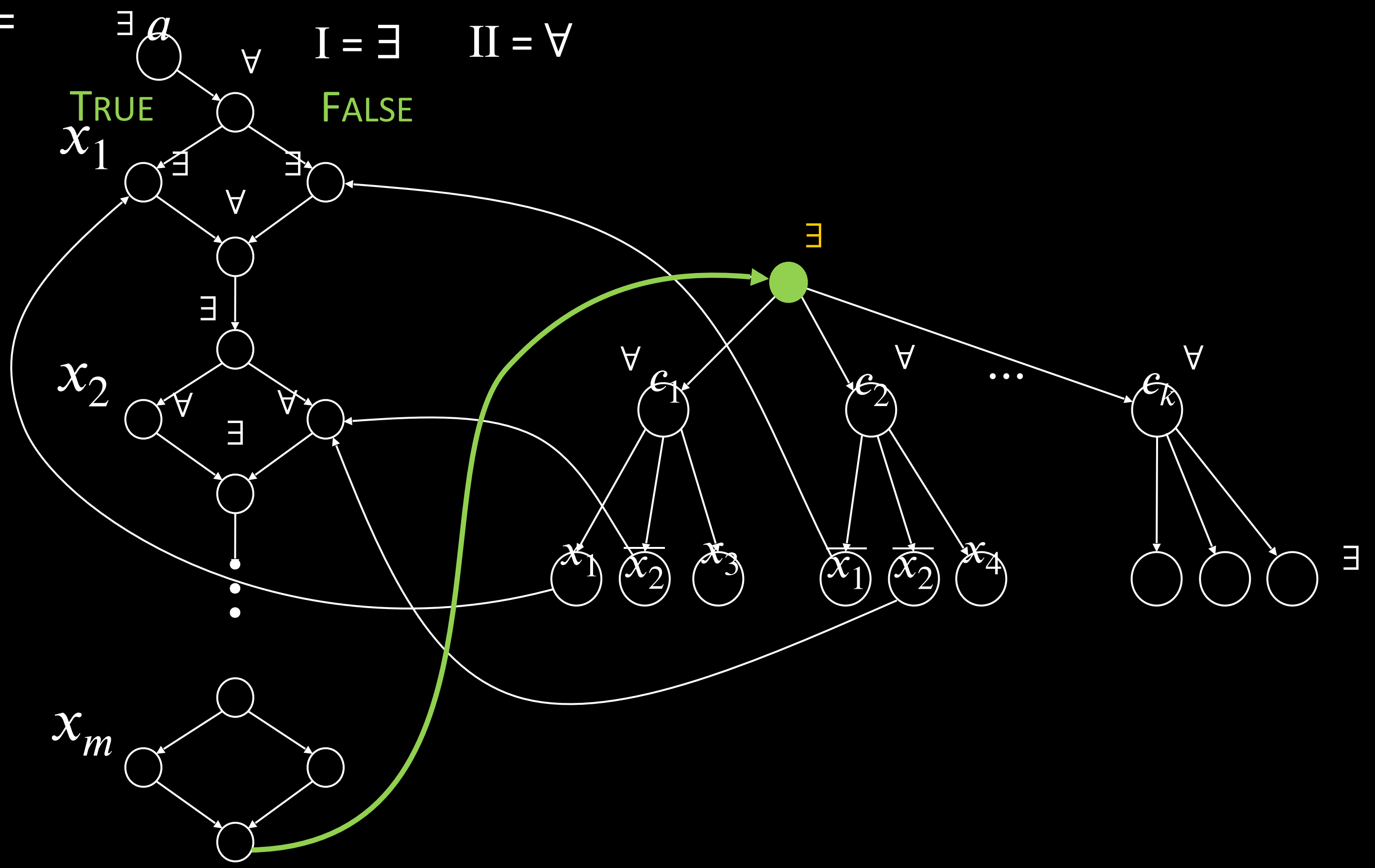
liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, (\underbrace{x_1 \vee \overline{x_2} \vee x_3}_{c_1}) \wedge (\underbrace{\overline{x_1} \vee \overline{x_2} \vee x_4}_{c_2}) \wedge \cdots \wedge (\underbrace{\cdots}_{c_k}) \,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

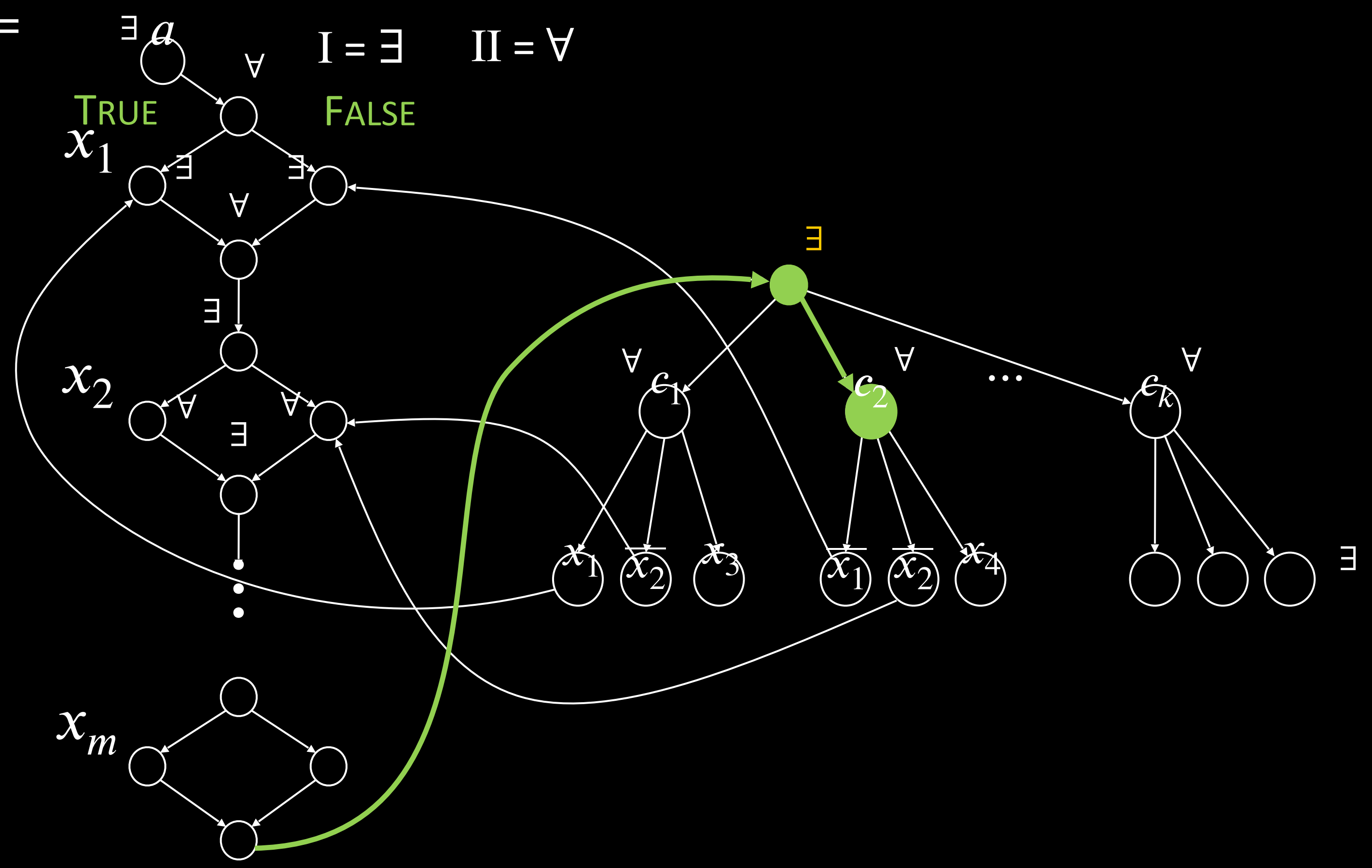
liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\cdots)}_{c_k} \,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

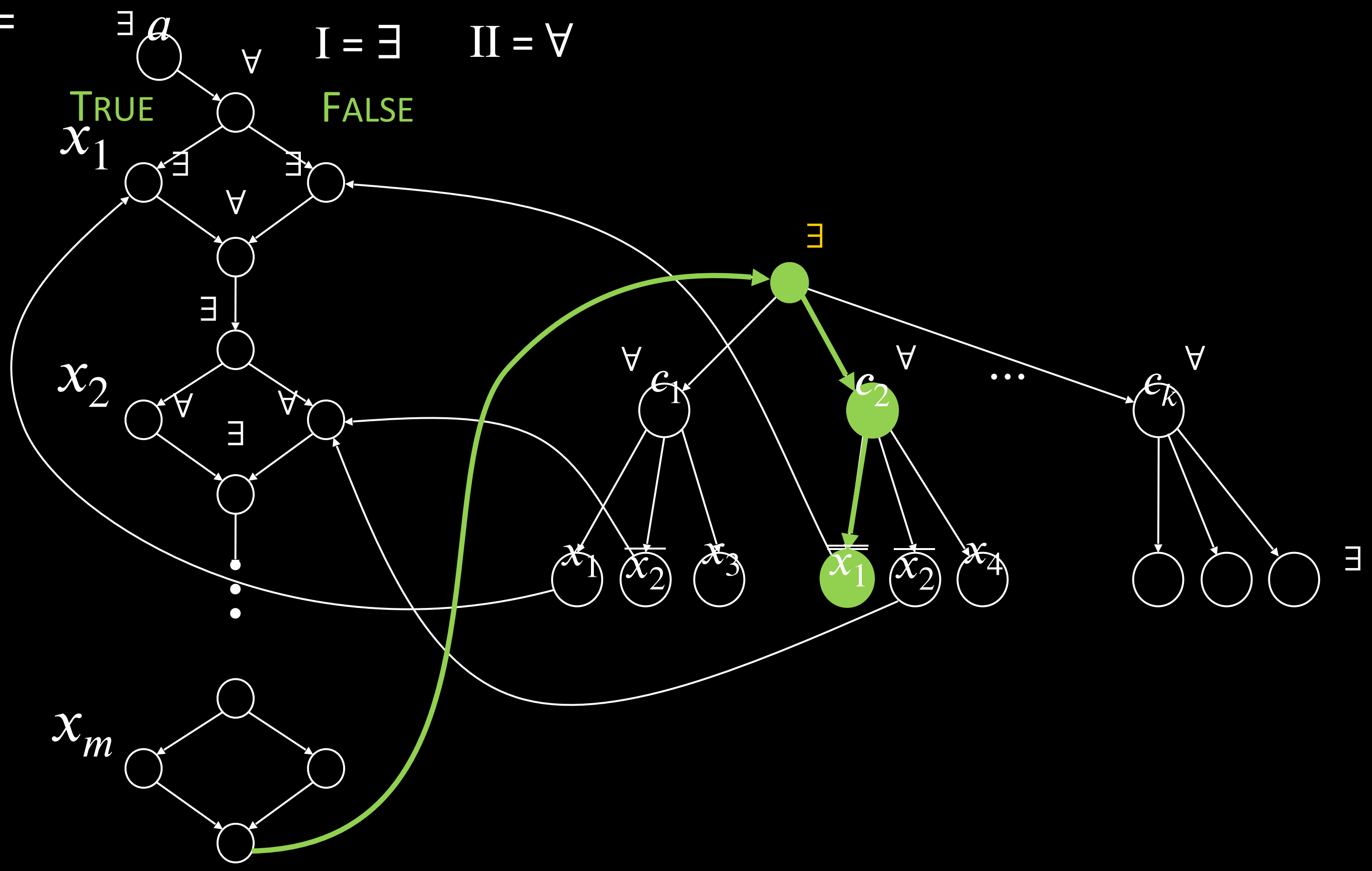
liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\, [\, \underbrace{(\, x_1 \vee \overline{x_2} \vee x_3\,)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\, \cdots)}_{c_k}\,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

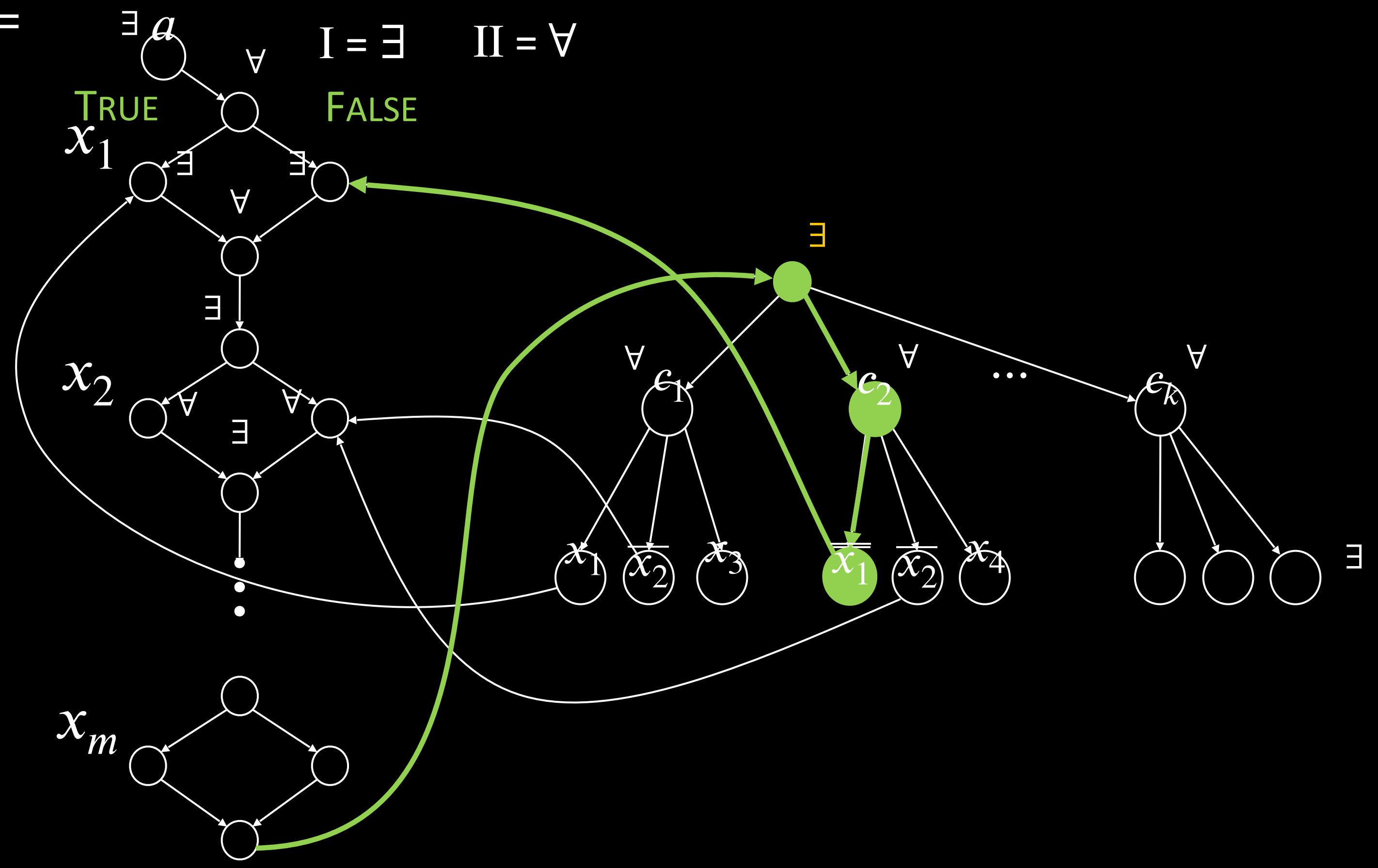
liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1\ \forall x_2\ \exists x_3 \cdots\ \forall x_k\ [\ \underbrace{(\ x_1 \vee \overline{x_2} \vee x_3\ )}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\ \cdots)}_{c_k}\ ]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

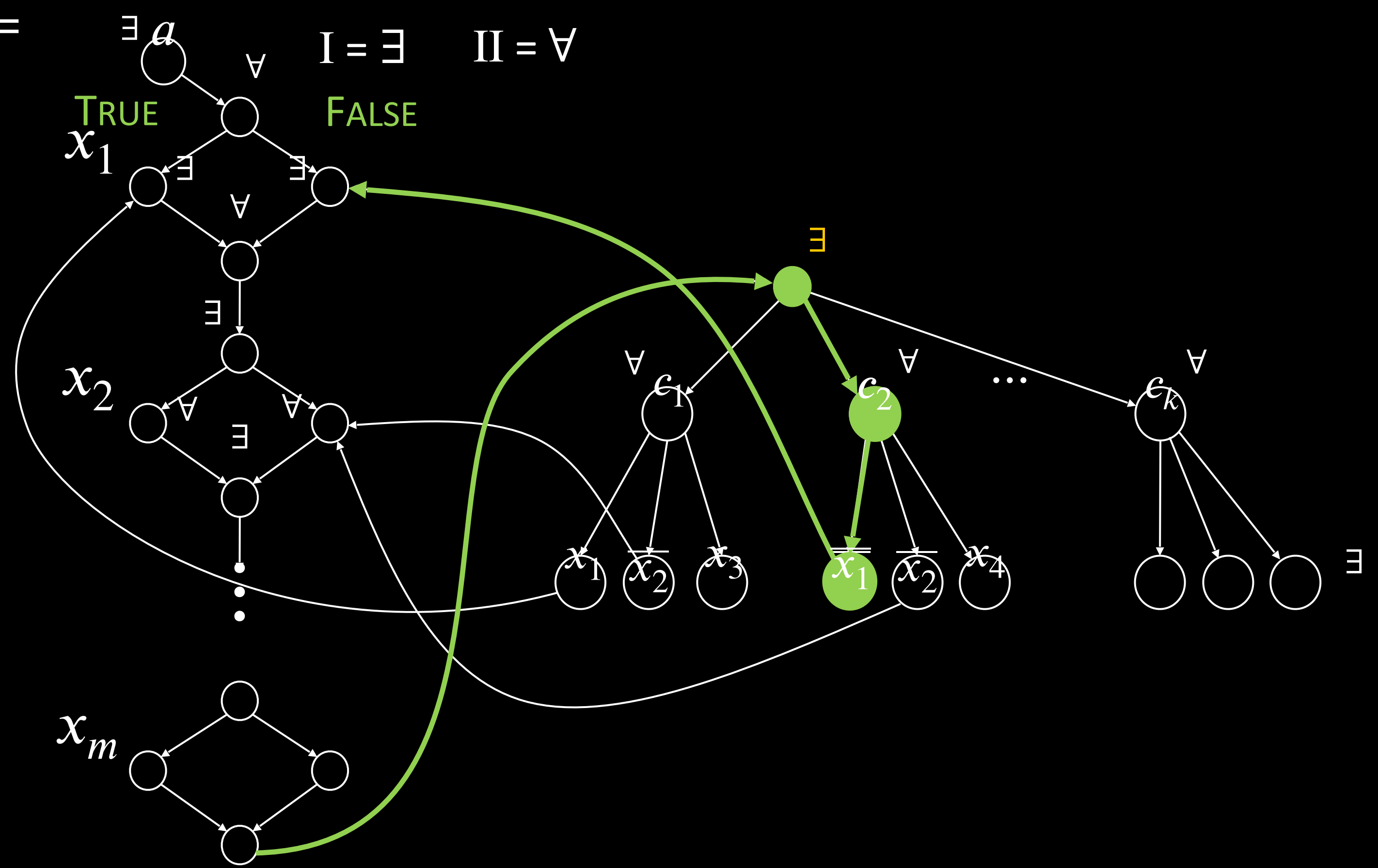
liar will be stuck

Constructing the GG graph G

Illustrate construction by example

Say $\phi = \exists x_1 \; \forall x_2 \; \exists x_3 \cdots \; \forall x_k \, [\, \underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\cdots)}_{c_k} \,]$

$G =$



Endgame

∃ should win if assignment satisfied all clauses

∀ should win if some unsatisfied clause

Implementation

∀ picks clause node claimed unsatisfied

∃ picks literal node claimed to satisfy the clause

liar will be stuck

Log space

Log space

To define sublinear space computation, do not count input as part of space used.

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$\left(\log n\right)$

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$(\log n)$

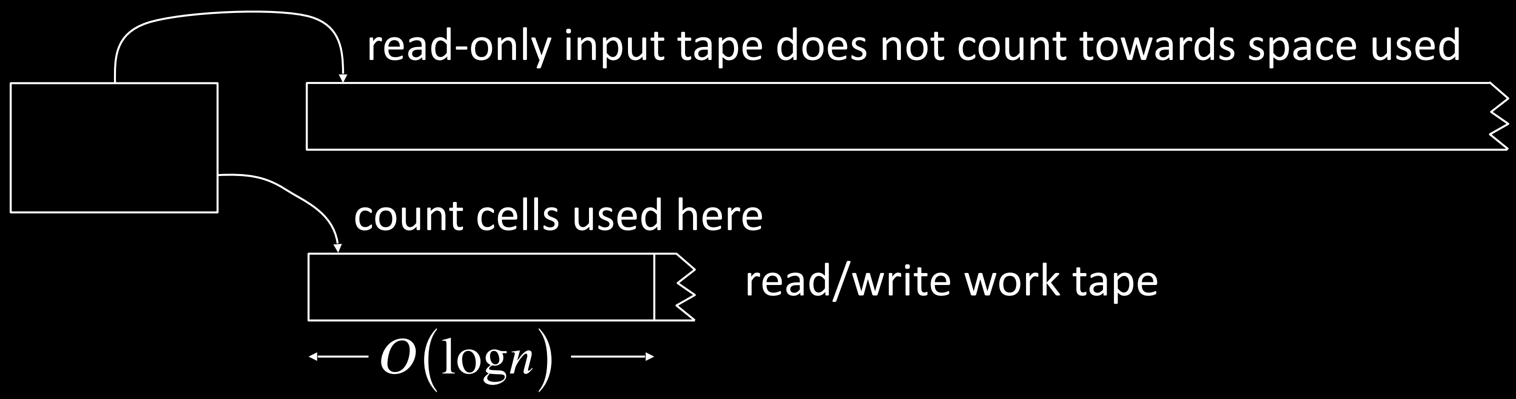
NL = NSPACE$(\log n)$

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$(\log n)$

NL = NSPACE$(\log n)$



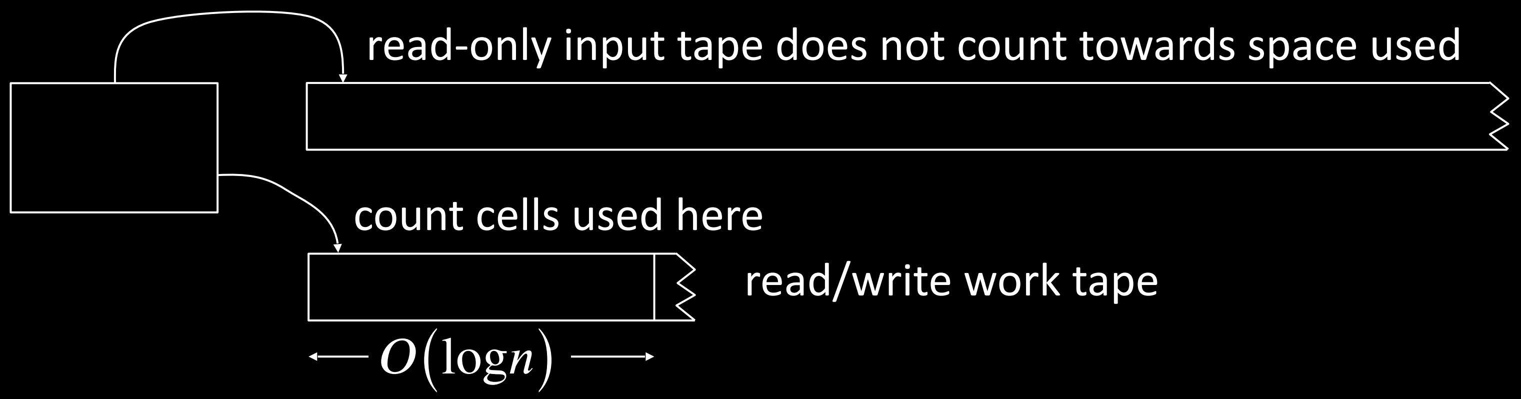
Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$\left(\log n\right)$

NL = NSPACE$\left(\log n\right)$

Log space can represent a constant number of pointers into the input.

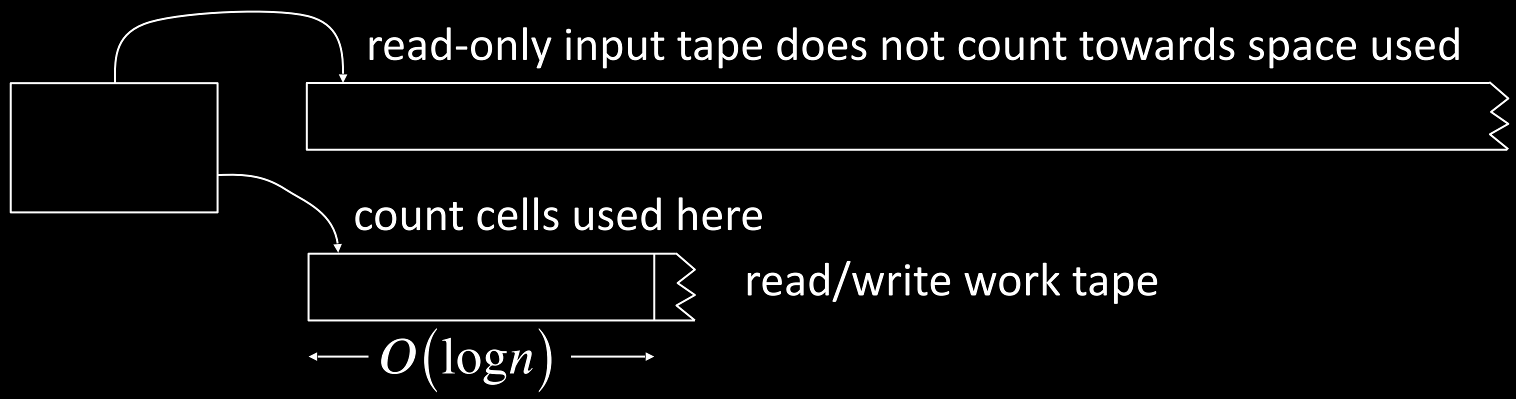


Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$\left(\log n\right)$

NL = NSPACE$\left(\log n\right)$



Log space can represent a constant number of pointers into the input.

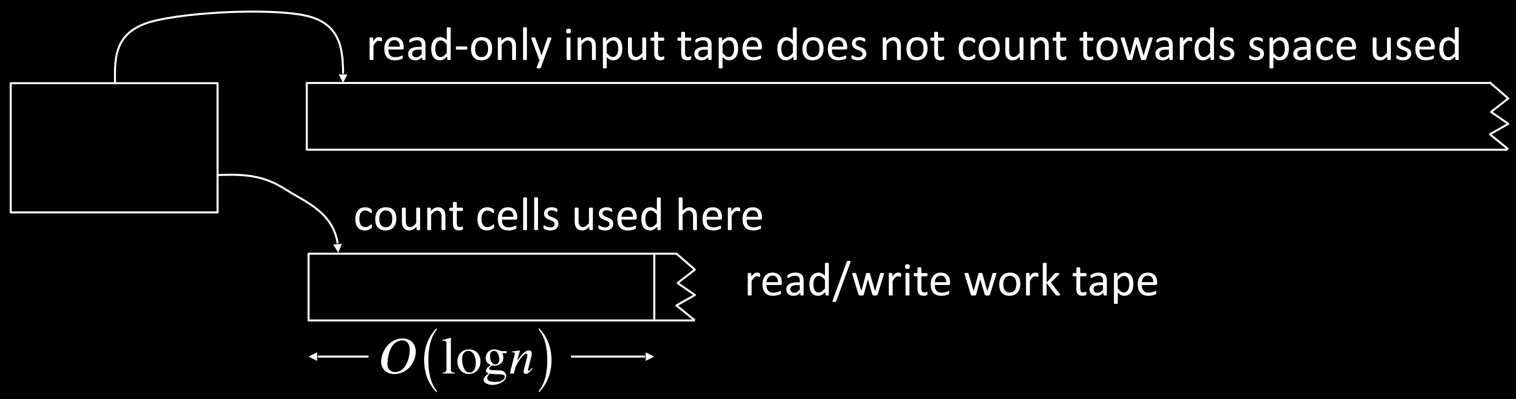
Examples

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$\left(\log n\right)$

NL = NSPACE$\left(\log n\right)$



Log space can represent a constant number of pointers into the input.

Examples

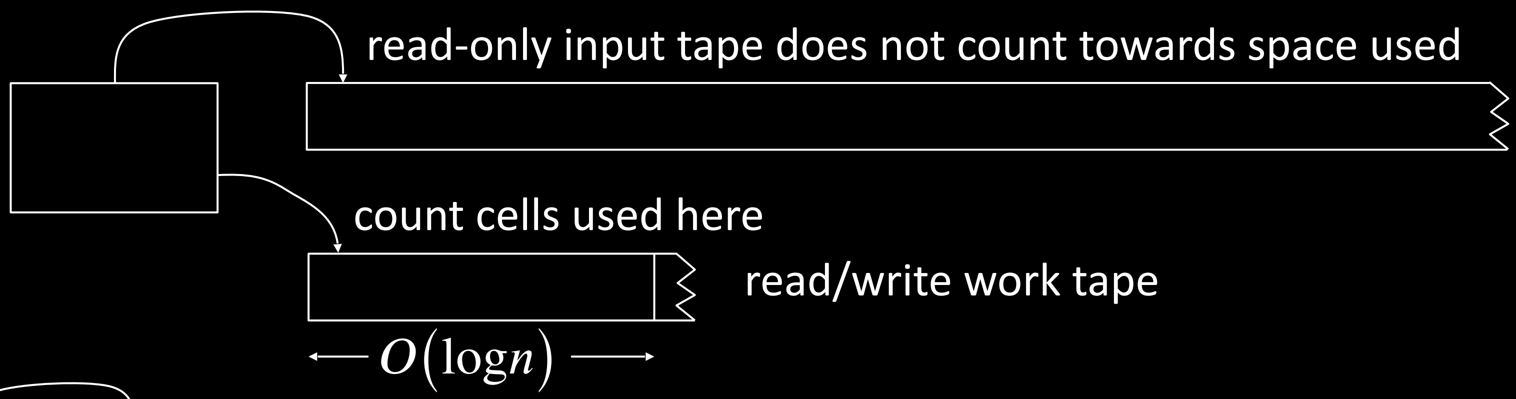
1. $\left\{ww^{\mathcal{R}} \;\middle|\; w \in \Sigma^*\right\} \in$ L

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$(\log n)$

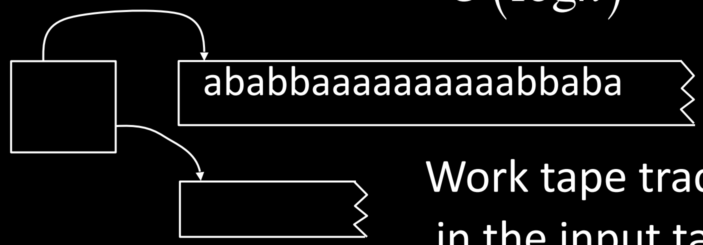
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



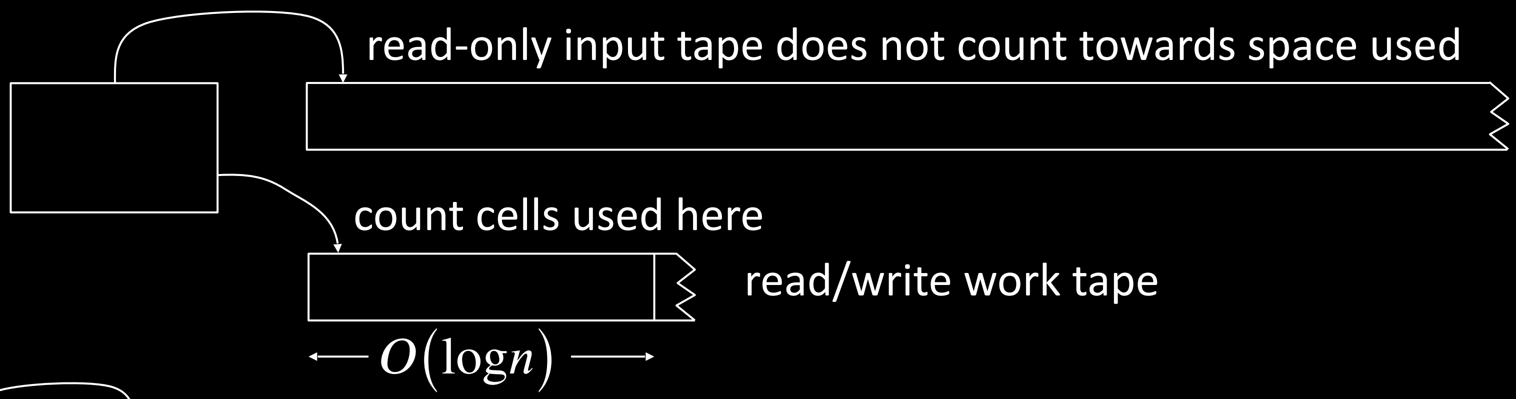
Work tape tracks corresponding locations in the input tape.

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$\left(\log n\right)$

NL = NSPACE$\left(\log n\right)$

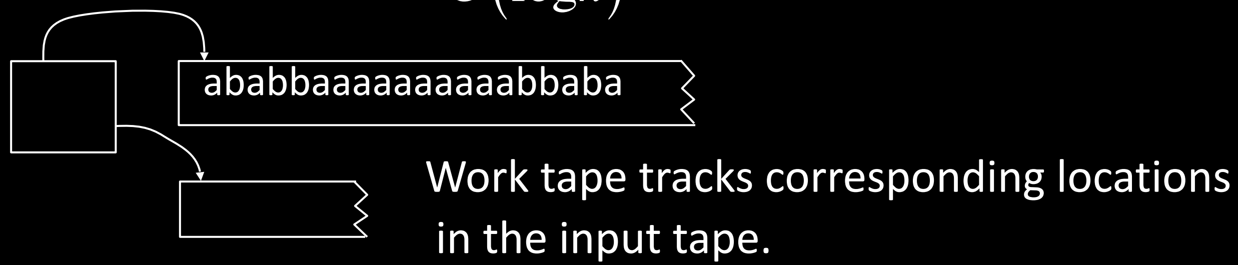


Log space can represent a constant number of pointers into the input.

Examples

1. $\left\{ww^{\mathscr{R}} \;\middle|\; w \in \Sigma^*\right\} \in$ L

2. $PATH \in$ NL

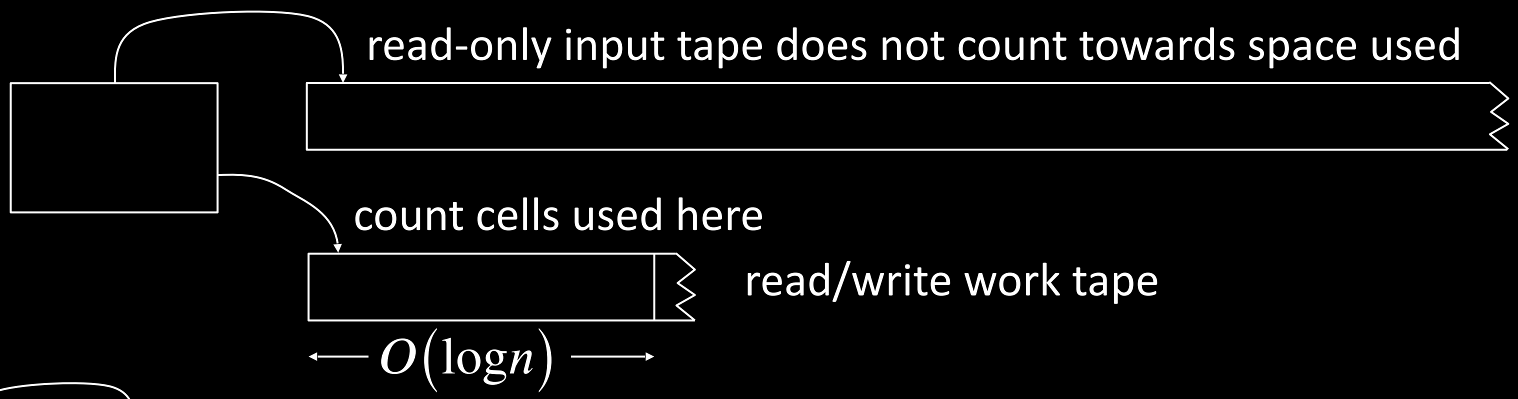


Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$(\log n)$

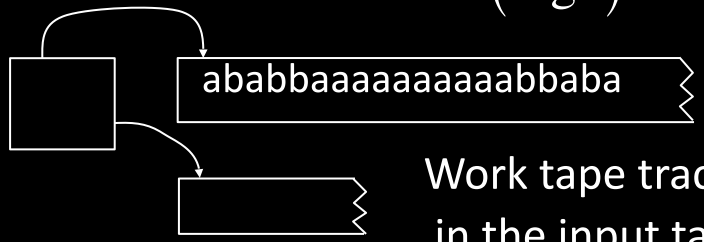
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



Work tape tracks corresponding locations in the input tape.

2. $PATH \in$ NL

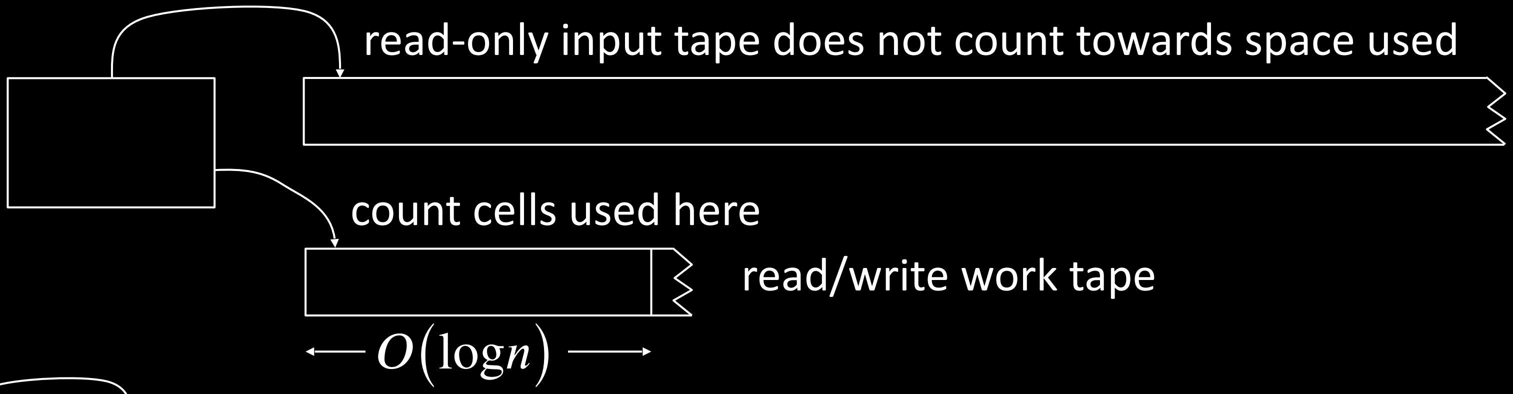
Nondeterministically select the nodes of a path connecting s to t.

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$(\log n)$

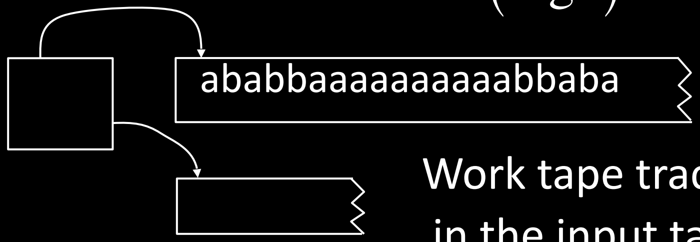
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

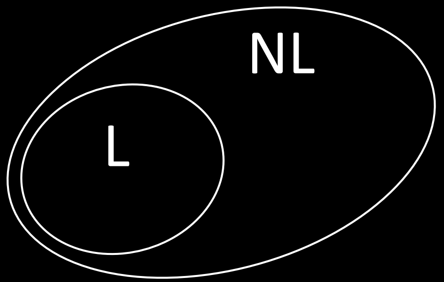
1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



Work tape tracks corresponding locations in the input tape.

2. $PATH \in$ NL

Nondeterministically select the nodes of a path connecting s to t.

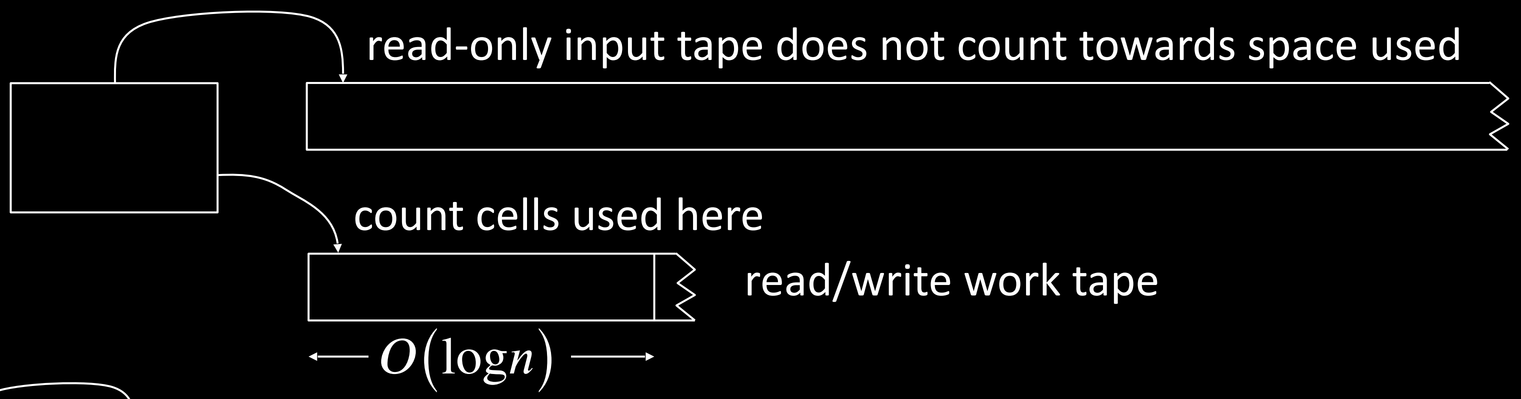


Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$(\log n)$

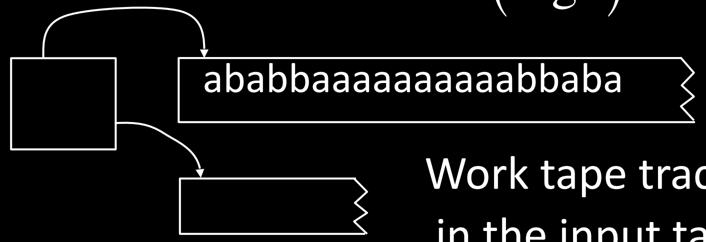
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

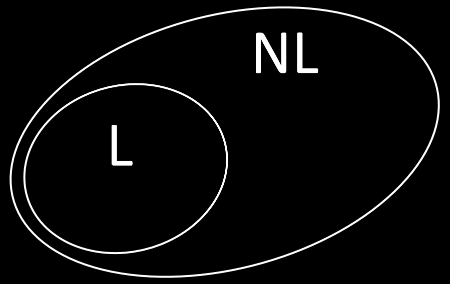
1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



Work tape tracks corresponding locations in the input tape.

2. $PATH \in$ NL

Nondeterministically select the nodes of a path connecting s to t.



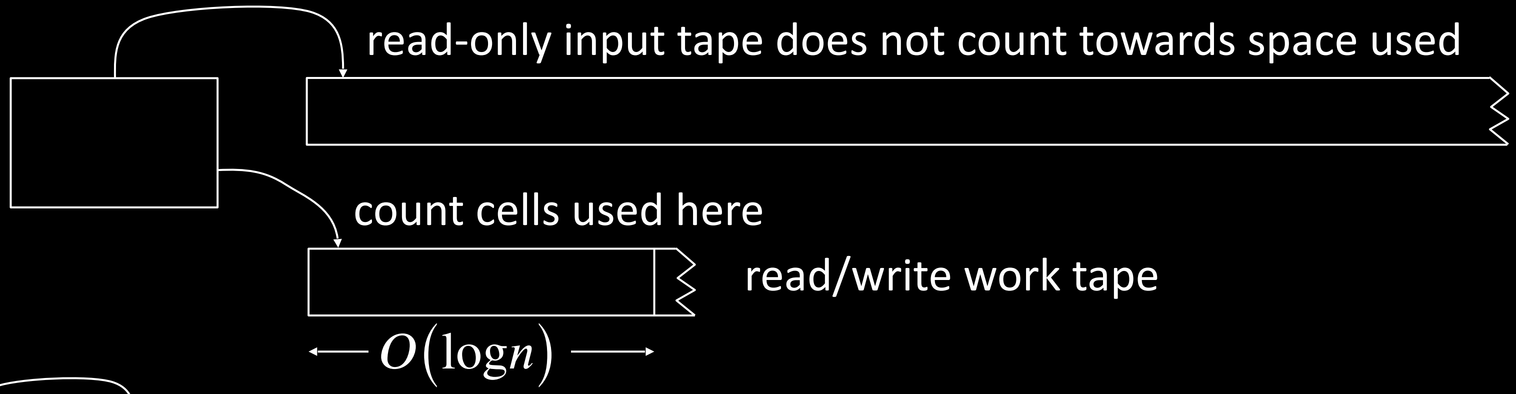
L = NL? Unsolved

Log space

To define sublinear space computation, do not count input as part of space used.
Use 2-tape TM model with read-only input tape.

Defn: L = SPACE$(\log n)$

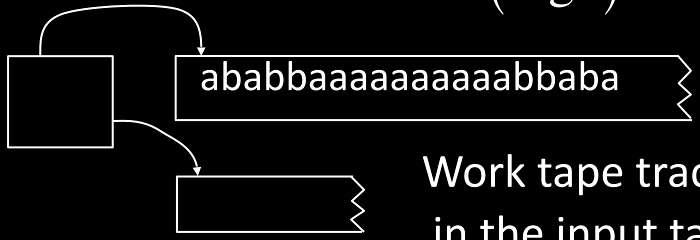
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

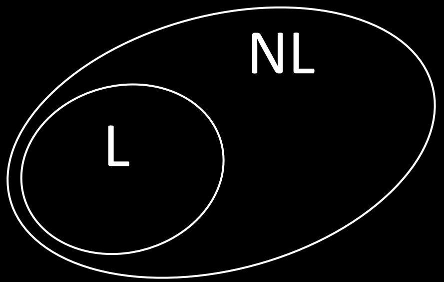
1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



Work tape tracks corresponding locations in the input tape.

2. $PATH \in$ NL

Nondeterministically select the nodes of a path connecting s to t.



L = NL? Unsolved

Log space properties

Theorem: $L \subseteq P$

Log space properties

Theorem: L $\subseteq$ P

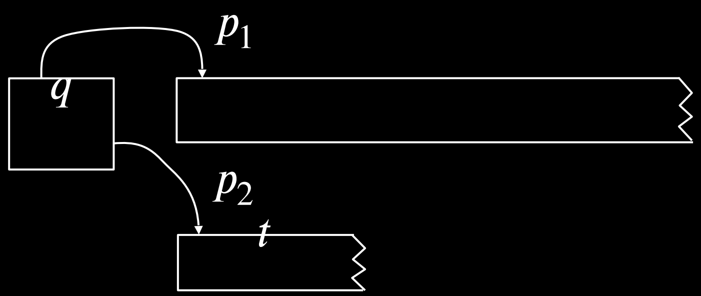
Proof: Say M decides A in space $O(\log n)$.

Log space properties

Theorem: L ⊆ P

Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the tape contents.



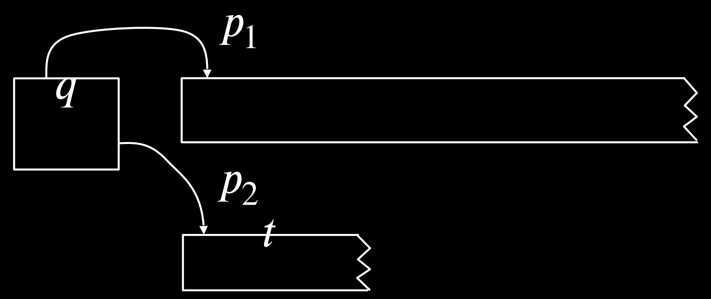
Log space properties

Theorem: L ⊆ P

Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.



Log space properties

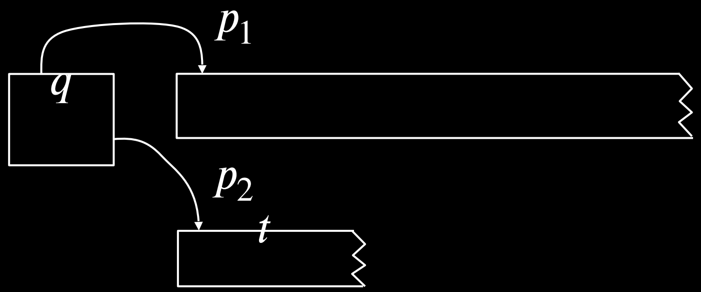
Theorem: L ⊆ P

Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.



Log space properties

Theorem: L $\subseteq$ P

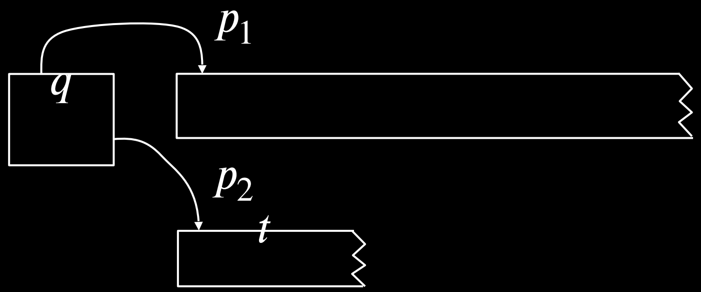
Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.

Conclusion: $A \in$ P



Log space properties

Theorem: L ⊆ P

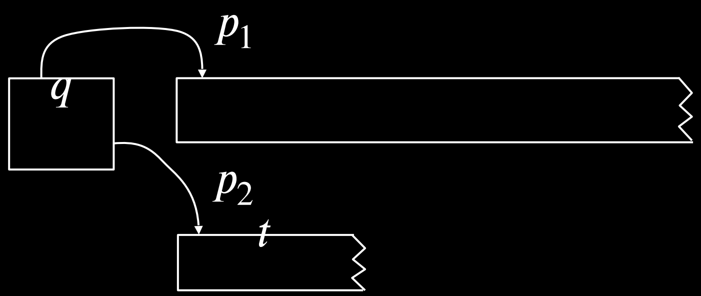
Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.

Conclusion: $A \in$ P



Theorem: NL ⊆ SPACE$(\log^2 n)$

Log space properties

Theorem: L $\subseteq$ P

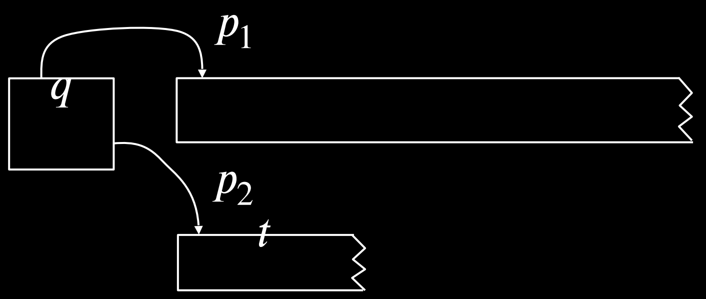
Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.

Conclusion: $A \in$ P



Theorem: NL $\subseteq$ SPACE$(\log^2 n)$

Proof: Savitch's theorem works for log space

Log space properties

Theorem: L $\subseteq$ P

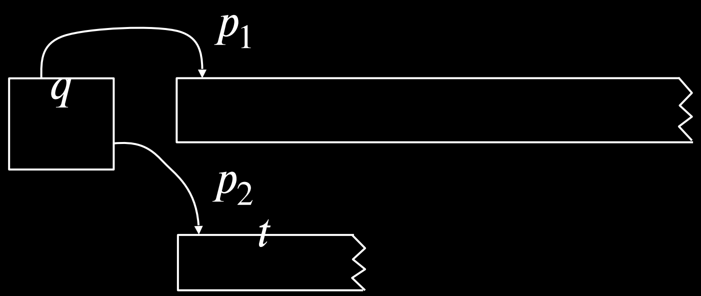
Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.

Conclusion: $A \in$ P



Theorem: NL $\subseteq$ SPACE$(\log^2 n)$

Proof: Savitch's theorem works for log space

NL properties

Theorem: NL $\subseteq$ P

NL properties

Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

NL properties

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The configuration graph $G_{M,w}$ for M on w has

NL properties

Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The configuration graph $G_{M,w}$ for M on w has

nodes: all configurations for M on w

NL properties

Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The configuration graph $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \rightarrow c_j$ if c_i can yield c_j in 1 step.

NL properties

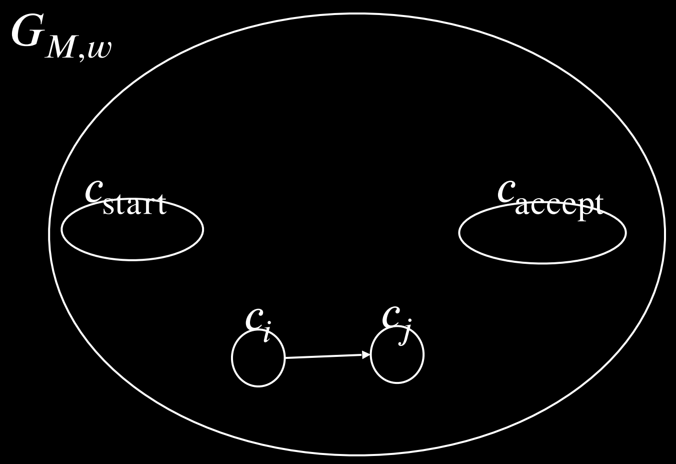
Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.



NL properties

Theorem: NL ⊆ P

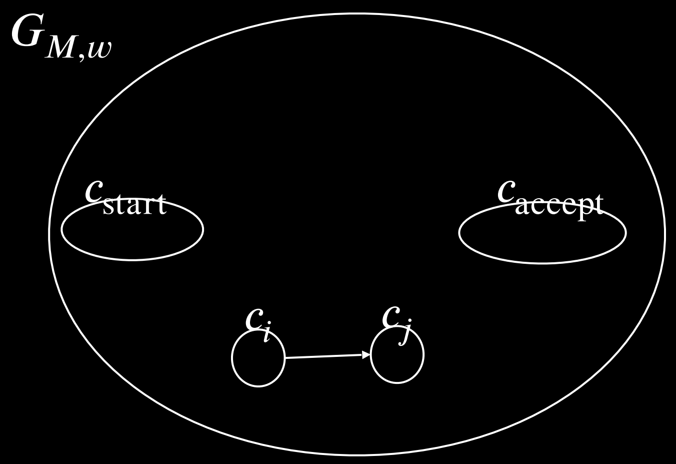
Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \rightarrow c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}



NL properties

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

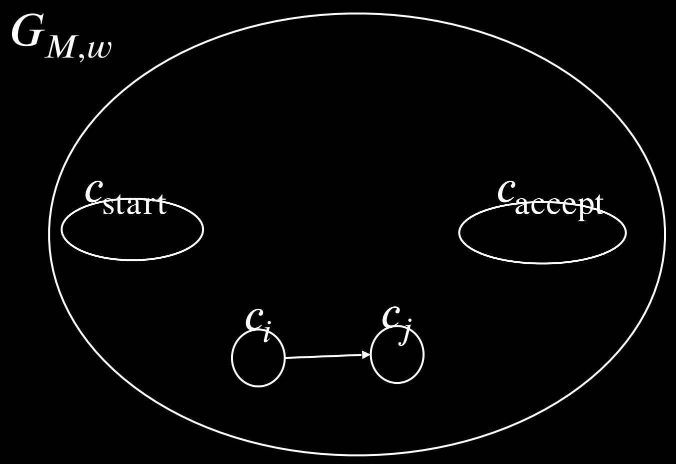
Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:



NL properties

Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The configuration graph $G_{M,w}$ for M on w has

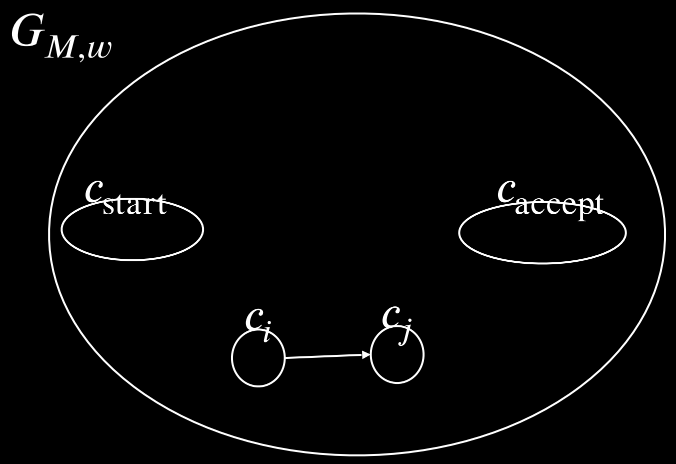
nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

$T =$ "On input w



NL properties

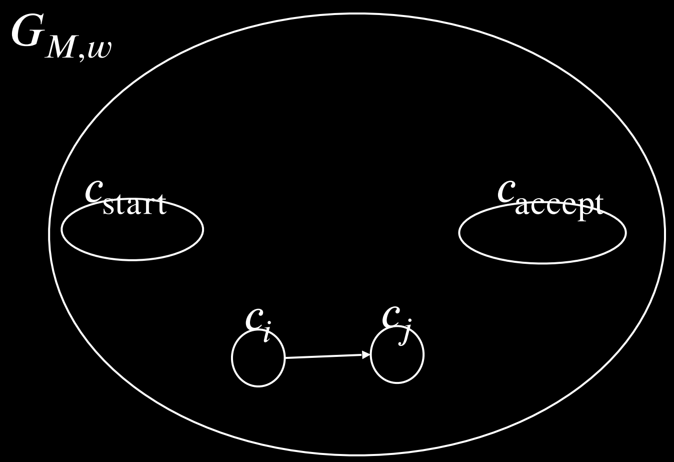
Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.



Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct the $G_{M,w}$.

NL properties

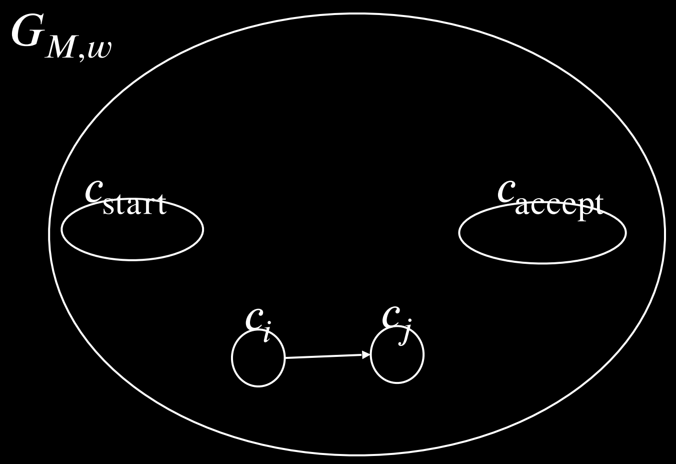
Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.



Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct the $G_{M,w}$.
2. *Accept* if there is a path from c_{start} to c_{accept}.
   Reject if not."

NL properties

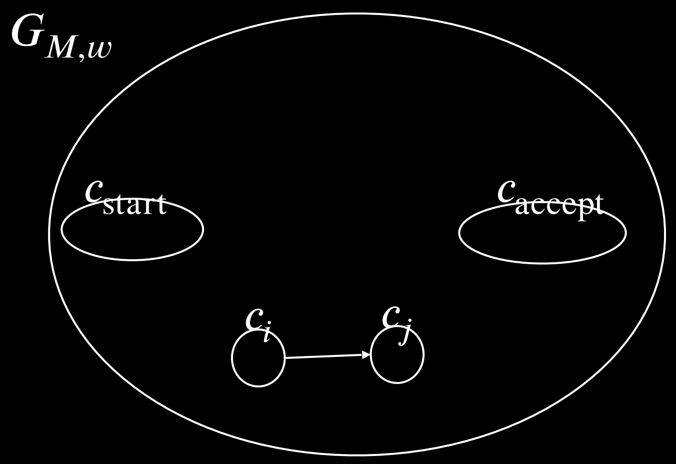
Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

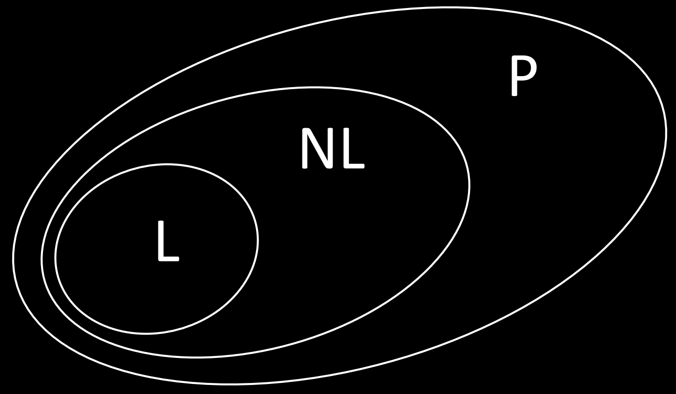


Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct the $G_{M,w}$.
2. *Accept* if there is a path from c_{start} to c_{accept}.
   Reject if not."



NL properties

Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

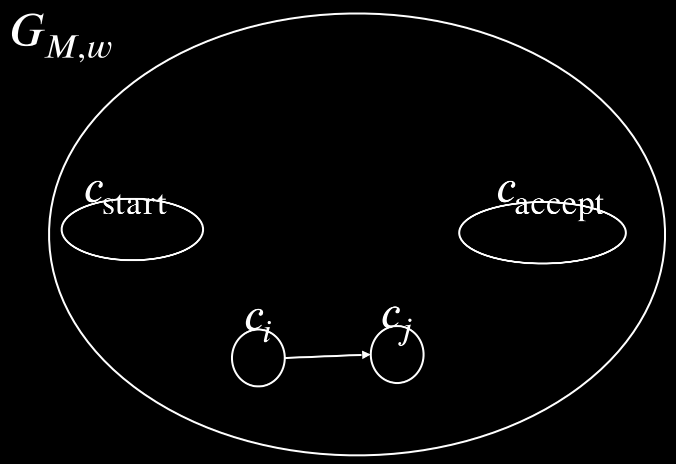
Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

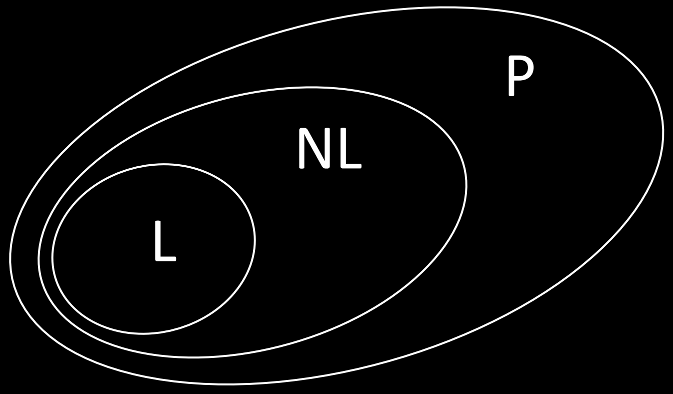
$T =$ "On input w

1. Construct the $G_{M,w}$.
2. *Accept* if there is a path from c_{start} to c_{accept}.

   Reject if not."



L = P? Unsolved



NL properties

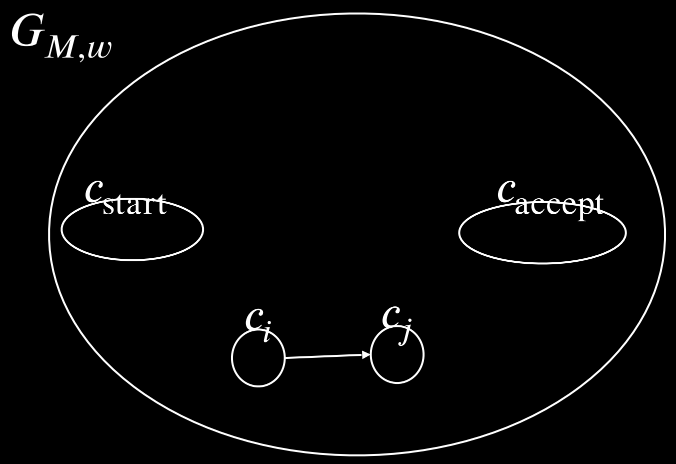
Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.



Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

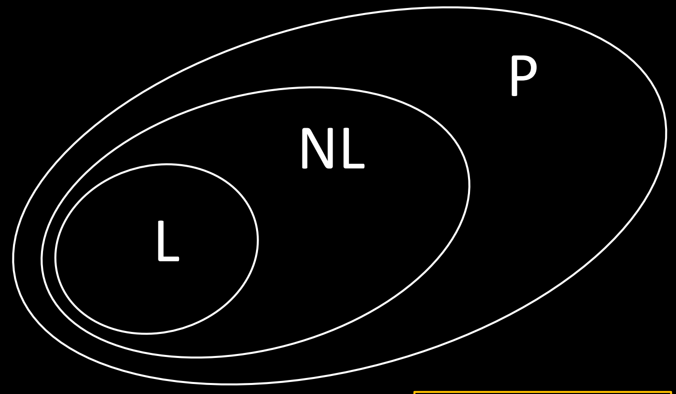
Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct the $G_{M,w}$.
2. *Accept* if there is a path from c_{start} to c_{accept}.

   Reject if not."

L = P? Unsolved



Check-in 19.3

NL properties

Theorem: NL $\subseteq$ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \rightarrow c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct the $G_{M,w}$.
2. *Accept* if there is a path from c_{start} to c_{accept}.
   Reject if not."

Check-in 19.3

We showed that $PATH \in$ NL.
What is the best we know about the deterministic space complexity of $PATH$?

(a) $PATH \in$ PSPACE

(b) $PATH \in$ SPACE(n)

(c) $PATH \in$ SPACE$\left(\log^2 n\right)$

(d) $PATH \in$ SPACE$\left(\log n\right)$

Quick review of today

1. The Formula Game
2. Generalized Geography is PSPACE-complete
3. Log space: L and NL
4. Configuration graph
5. NL $\subseteq$ P

Quick review of today

1. The Formula Game
2. Generalized Geography is PSPACE-complete
3. Log space: L and NL
4. Configuration graph
5. NL $\subseteq$ P