

# COMPLEXITY THEORY

## Lecture 10: Polynomial Space

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 13th Nov 2018

# Review

# The Class PSpace

We defined PSpace as:

$$\text{PSpace} = \bigcup_{d \geq 1} \text{DSpace}(n^d)$$

and we observed that

$$P \subseteq NP \subseteq \text{PSpace} = \text{NPSpace} \subseteq \text{ExpTime}.$$

We can also define a corresponding notion of PSpace-hardness:

## Definition 10.1:

- A language **H** is **PSpace-hard**, if  $L \leq_p H$  for every language  $L \in \text{PSpace}$ .
- A language **C** is **PSpace-complete**, if **C** is PSpace-hard and  $C \in \text{PSpace}$ .

# Quantified Boolean Formulae (QBF)

A **QBF** is a formula of the following form:

$$Q_1 X_1. Q_2 X_2. \dots Q_\ell X_\ell. \varphi[X_1, \dots, X_\ell]$$

where  $Q_i \in \{\exists, \forall\}$  are quantifiers,  $X_i$  are propositional logic variables, and  $\varphi$  is a propositional logic formula with variables  $X_1, \dots, X_\ell$  and constants  $\top$  (true) and  $\perp$  (false)

## Semantics:

- Propositional formulae without variables (only constants  $\top$  and  $\perp$ ) are evaluated as usual
  - $\exists X. \varphi[X]$  is true if either  $\varphi[X/\top]$  or  $\varphi[X/\perp]$  are true
  - $\forall X. \varphi[X]$  is true if both  $\varphi[X/\top]$  and  $\varphi[X/\perp]$  are true
- (where  $\varphi[X/\top]$  is “ $\varphi$  with  $X$  replaced by  $\top$ , and similar for  $\perp$ )

# Deciding QBF Validity

## **TRUE QBF**

Input: A quantified Boolean formula  $\varphi$ .

Problem: Is  $\varphi$  true (valid)?

**Observation:** We can assume that the quantified formula is in CNF or 3-CNF (same transformations possible as for propositional logic formulae)

# Deciding QBF Validity

## TRUE QBF

Input: A quantified Boolean formula  $\varphi$ .

Problem: Is  $\varphi$  true (valid)?

**Observation:** We can assume that the quantified formula is in CNF or 3-CNF (same transformations possible as for propositional logic formulae)

Consider a propositional logic formula  $\varphi$  with variables  $X_1, \dots, X_\ell$ :

**Example 10.2:** The QBF  $\exists X_1. \dots \exists X_\ell. \varphi$  is true if and only if  $\varphi$  is satisfiable.

**Example 10.3:** The QBF  $\forall X_1. \dots \forall X_\ell. \varphi$  is true if and only if  $\varphi$  is a tautology.

# The Power of QBF

**Theorem 10.4:** **TRUE QBF** is PSpace-complete.

## Proof:

(1) **TRUE QBF**  $\in$  PSpace:

Give an algorithm that runs in polynomial space.

(2) **TRUE QBF** is PSpace-hard:

Proof by reduction from the word problem for polynomially space-bounded TMs.

# Solving **TRUE QBF** in PSpace

```
01 TRUEQBF( $\varphi$ ) {  
02   if  $\varphi$  has no quantifiers :  
03     return "evaluation of  $\varphi$ "  
04   else if  $\varphi = \exists X.\psi$  :  
05     return (TRUEQBF( $\psi[X/\top]$ ) OR TRUEQBF( $\psi[X/\perp]$ ))  
06   else if  $\varphi = \forall X.\psi$  :  
07     return (TRUEQBF( $\psi[X/\top]$ ) AND TRUEQBF( $\psi[X/\perp]$ ))  
08 }
```

- Evaluation in line 03 can be done in polynomial space
- Recursions in lines 05 and 07 can be executed one after the other, reusing space
- Maximum depth of recursion = number of variables (linear)
- Store one variable assignment per recursive call

↪ polynomial space algorithm



# PSPACE-Hardness of TRUE QBF

Express TM computation in logic, similar to Cook-Levin

Given:

- a polynomial  $p$
- a  $p$ -space bounded 1-tape NTM  $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}})$
- a word  $w$

Intended reduction

Define a QBF  $\varphi_{p, \mathcal{M}, w}$  such that

$\varphi_{p, \mathcal{M}, w}$  is true if and only if  $\mathcal{M}$  accepts  $w$  in space  $p(|w|)$ .

Note

We show the reduction for NTMs, which is more than needed, but makes little difference in logic and allows us to reuse our previous formulae from Cook-Levin

# Review: Encoding Configurations

Use propositional variables for describing configurations:

$Q_q$  for each  $q \in Q$  means “ $\mathcal{M}$  is in state  $q \in Q$ ”

$P_i$  for each  $0 \leq i < p(n)$  means “the head is at Position  $i$ ”

$S_{a,i}$  for each  $a \in \Gamma$  and  $0 \leq i < p(n)$  means “tape cell  $i$  contains Symbol  $a$ ”

Represent configuration  $(q, p, a_0 \dots a_{p(n)})$

by assigning truth values to variables from the set

$$\bar{C} := \{Q_q, P_i, S_{a,i} \mid q \in Q, \quad a \in \Gamma, \quad 0 \leq i < p(n)\}$$

using the truth assignment  $\beta$  defined as

$$\beta(Q_s) := \begin{cases} 1 & s = q \\ 0 & s \neq q \end{cases} \quad \beta(P_i) := \begin{cases} 1 & i = p \\ 0 & i \neq p \end{cases} \quad \beta(S_{a,i}) := \begin{cases} 1 & a = a_i \\ 0 & a \neq a_i \end{cases}$$

# Review: Validating Configurations

We define a formula  $\text{Conf}(\bar{C})$  for a set of configuration variables

$$\bar{C} = \{Q_q, P_i, S_{a,i} \mid q \in Q, \quad a \in \Gamma, \quad 0 \leq i < p(n)\}$$

as follows:

$\text{Conf}(\bar{C}) :=$

“the assignment is a valid configuration”:

$$\bigvee_{q \in Q} (Q_q \wedge \bigwedge_{q' \neq q} \neg Q_{q'})$$

“TM in exactly one state  $q \in Q$ ”

$$\wedge \bigvee_{p < p(n)} (P_p \wedge \bigwedge_{p' \neq p} \neg P_{p'})$$

“head in exactly one position  $p < p(n)$ ”

$$\wedge \bigwedge_{0 \leq i < p(n)} \bigvee_{a \in \Gamma} (S_{a,i} \wedge \bigwedge_{b \neq a \in \Gamma} \neg S_{b,i})$$

“exactly one  $a \in \Gamma$  in each cell”

# Review: Validating Configurations

For an assignment  $\beta$  defined on variables in  $\overline{C}$  define

$$\text{conf}(\overline{C}, \beta) := \left\{ \begin{array}{l} \beta(Q_q) = 1, \\ (q, p, w_0 \dots w_{p(n)}) \mid \beta(P_p) = 1, \\ \beta(S_{w_i, i}) = 1 \text{ for all } 0 \leq i < p(n) \end{array} \right\}$$

Note:  $\beta$  may be defined on other variables besides those in  $\overline{C}$ .

**Lemma 10.5:** If  $\beta$  satisfies  $\text{Conf}(\overline{C})$  then  $|\text{conf}(\overline{C}, \beta)| = 1$ .

We can therefore write  $\text{conf}(\overline{C}, \beta) = (q, p, w)$  to simplify notation.

Observations:

- $\text{conf}(\overline{C}, \beta)$  is a potential configuration of  $\mathcal{M}$ , but it may not be reachable from the start configuration of  $\mathcal{M}$  on input  $w$ .
- Conversely, every configuration  $(q, p, w_1 \dots w_{p(n)})$  induces a satisfying assignment  $\beta$  or which  $\text{conf}(\overline{C}, \beta) = (q, p, w_1 \dots w_{p(n)})$ .

# Review: Transitions Between Configurations

Consider the following formula  $\text{Next}(\bar{C}, \bar{C}')$  defined as

$$\text{Conf}(\bar{C}) \wedge \text{Conf}(\bar{C}') \wedge \text{NoChange}(\bar{C}, \bar{C}') \wedge \text{Change}(\bar{C}, \bar{C}').$$

$$\text{NoChange} := \bigvee_{0 \leq p < p(n)} \left( P_p \wedge \bigwedge_{i \neq p, a \in \Gamma} (S_{a,i} \rightarrow S'_{a,i}) \right)$$

$$\text{Change} := \bigvee_{0 \leq p < p(n)} \left( P_p \wedge \bigvee_{\substack{q \in Q \\ a \in \Gamma}} (Q_q \wedge S_{a,p} \wedge \bigvee_{(q', b, D) \in \delta(q, a)} (Q'_{q'} \wedge S'_{b,p} \wedge P'_{D(p)})) \right)$$

where  $D(p)$  is the position reached by moving in direction  $D$  from  $p$ .

**Lemma 10.6:** For any assignment  $\beta$  defined on  $\bar{C} \cup \bar{C}'$ :

$\beta$  satisfies  $\text{Next}(\bar{C}, \bar{C}')$  if and only if  $\text{conf}(\bar{C}, \beta) \vdash_{\mathcal{M}} \text{conf}(\bar{C}', \beta)$

# Review: Start and End

Defined so far:

- $\text{Conf}(\bar{C})$ :  $\bar{C}$  describes a potential configuration
- $\text{Next}(\bar{C}, \bar{C}')$ :  $\text{conf}(\bar{C}, \beta) \vdash_{\mathcal{M}} \text{conf}(\bar{C}', \beta)$

**Start configuration:** Let  $w = w_0 \cdots w_{n-1} \in \Sigma^*$  be the input word

$$\text{Start}_{\mathcal{M}, w}(\bar{C}) := \text{Conf}(\bar{C}) \wedge Q_{q_0} \wedge P_0 \wedge \bigwedge_{i=0}^{n-1} S_{w_i, i} \wedge \bigwedge_{i=n}^{p(n)-1} S_{\perp, i}$$

Then an assignment  $\beta$  satisfies  $\text{Start}_{\mathcal{M}, w}(\bar{C})$  if and only if  $\bar{C}$  represents the start configuration of  $\mathcal{M}$  on input  $w$ .

**Accepting stop configuration:**

$$\text{Acc-Conf}(\bar{C}) := \text{Conf}(\bar{C}) \wedge Q_{q_{\text{accept}}}$$

Then an assignment  $\beta$  satisfies  $\text{Acc-Conf}(\bar{C})$  if and only if  $\bar{C}$  represents an accepting configuration of  $\mathcal{M}$ .

# Simulating Polynomial Space Computations

For Cook-Levin, we used one set of configuration variables for every computing step:  
polynomially time  $\leadsto$  polynomially many variables

**Problem:** For polynomial space, we have  $2^{O(p(n))}$  possible steps . . .

# Simulating Polynomial Space Computations

For Cook-Levin, we used one set of configuration variables for every computing step:  
polynomially time  $\leadsto$  polynomially many variables

**Problem:** For polynomial space, we have  $2^{O(p(n))}$  possible steps . . .

**What would Savitch do?**



# Simulating Polynomial Space Computations

For Cook-Levin, we used one set of configuration variables for every computing step:  
polynomially time  $\leadsto$  polynomially many variables

**Problem:** For polynomial space, we have  $2^{O(p(n))}$  possible steps . . .

## What would Savitch do?

Define a formula  $\text{CanYield}_i(\bar{C}_1, \bar{C}_2)$  to state that  $\bar{C}_2$  is reachable from  $\bar{C}_1$  in at most  $2^i$  steps:

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

# Simulating Polynomial Space Computations

For Cook-Levin, we used one set of configuration variables for every computing step:  
polynomially time  $\leadsto$  polynomially many variables

**Problem:** For polynomial space, we have  $2^{O(p(n))}$  possible steps . . .

## What would Savitch do?

Define a formula  $\text{CanYield}_i(\bar{C}_1, \bar{C}_2)$  to state that  $\bar{C}_2$  is reachable from  $\bar{C}_1$  in at most  $2^i$  steps:

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

But what is  $\bar{C}_1 = \bar{C}_2$  supposed to mean here?

# Simulating Polynomial Space Computations

For Cook-Levin, we used one set of configuration variables for every computing step:  
polynomially time  $\leadsto$  polynomially many variables

**Problem:** For polynomial space, we have  $2^{O(p(n))}$  possible steps ...

## What would Savitch do?

Define a formula  $\text{CanYield}_i(\bar{C}_1, \bar{C}_2)$  to state that  $\bar{C}_2$  is reachable from  $\bar{C}_1$  in at most  $2^i$  steps:

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

But what is  $\bar{C}_1 = \bar{C}_2$  supposed to mean here? It is short for:

$$\bigwedge_{q \in Q} Q_q^1 \leftrightarrow Q_q^2 \wedge \bigwedge_{0 \leq i < p(n)} P_i^1 \leftrightarrow P_i^2 \wedge \bigwedge_{a \in \Gamma, 0 \leq i < p(n)} S_{a,i}^1 \leftrightarrow S_{a,i}^2$$

# Putting Everything Together

We define the formula  $\varphi_{p,\mathcal{M},w}$  as follows:

$$\varphi_{p,\mathcal{M},w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M},w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

where we select  $d$  to be the least number such that  $\mathcal{M}$  has less than  $2^{dp(n)}$  configurations in space  $p(n)$ .

**Lemma 10.7:**  $\varphi_{p,\mathcal{M},w}$  is satisfiable if and only if  $\mathcal{M}$  accepts  $w$  in space  $p(|w|)$ .

# Did we do it?

Note: we used only existential quantifiers when defining  $\varphi_{p, \mathcal{M}, w}$ :

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p, \mathcal{M}, w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M}, w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

# Did we do it?

Note: we used only existential quantifiers when defining  $\varphi_{p,\mathcal{M},w}$ :

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p,\mathcal{M},w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M},w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Now that's quite interesting . . .

# Did we do it?

Note: we used only existential quantifiers when defining  $\varphi_{p,\mathcal{M},w}$ :

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p,\mathcal{M},w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M},w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Now that's quite interesting . . .

- With only (non-negated)  $\exists$  quantifiers, **TRUE QBF** coincides with **SAT**

# Did we do it?

Note: we used only existential quantifiers when defining  $\varphi_{p,\mathcal{M},w}$ :

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p,\mathcal{M},w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M},w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Now that's quite interesting . . .

- With only (non-negated)  $\exists$  quantifiers, **TRUE QBF** coincides with **SAT**
- **SAT** is in NP



# Did we do it?

Note: we used only existential quantifiers when defining  $\varphi_{p, \mathcal{M}, w}$ :

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p, \mathcal{M}, w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M}, w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Now that's quite interesting . . .

- With only (non-negated)  $\exists$  quantifiers, **TRUE QBF** coincides with **SAT**
- **SAT** is in NP
- So we showed that the word problem for PSpace NTMs to be in NP

# Did we do it?

**Note:** we used only existential quantifiers when defining  $\varphi_{p, \mathcal{M}, w}$ :

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p, \mathcal{M}, w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M}, w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Now that's quite interesting . . .

- With only (non-negated)  $\exists$  quantifiers, **TRUE QBF** coincides with **SAT**
- **SAT** is in NP
- So we showed that the word problem for PSpace NTMs to be in NP

So we found that **NP = PSpace!**

# Did we do it?

**Note:** we used only existential quantifiers when defining  $\varphi_{p,\mathcal{M},w}$ :

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p,\mathcal{M},w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M},w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Now that's quite interesting . . .

- With only (non-negated)  $\exists$  quantifiers, **TRUE QBF** coincides with **SAT**
- **SAT** is in NP
- So we showed that the word problem for PSpace NTMs to be in NP

So we found that **NP = PSpace!**

Strangely, most textbooks claim that this is not known to be true . . .

Are we up for the next Turing Award, or did we make a [mistake](#)?

# Size

How big is  $\varphi_{p, \mathcal{M}, w}$ ?

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p, \mathcal{M}, w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M}, w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

# Size

How big is  $\varphi_{p, \mathcal{M}, w}$ ?

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p, \mathcal{M}, w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M}, w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Size of  $\text{CanYield}_{i+1}$  is more than twice the size of  $\text{CanYield}_i$

$\leadsto$  Size of  $\varphi_{p, \mathcal{M}, w}$  is in  $2^{O(p(n))}$ . Oops.

# Size

How big is  $\varphi_{p, \mathcal{M}, w}$ ?

$$\text{CanYield}_0(\bar{C}_1, \bar{C}_2) := (\bar{C}_1 = \bar{C}_2) \vee \text{Next}(\bar{C}_1, \bar{C}_2)$$

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) := \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \text{CanYield}_i(\bar{C}_1, \bar{C}) \wedge \text{CanYield}_i(\bar{C}, \bar{C}_2)$$

$$\varphi_{p, \mathcal{M}, w} := \exists \bar{C}_1. \exists \bar{C}_2. \text{Start}_{\mathcal{M}, w}(\bar{C}_1) \wedge \text{Acc-Conf}(\bar{C}_2) \wedge \text{CanYield}_{dp(n)}(\bar{C}_1, \bar{C}_2)$$

Size of  $\text{CanYield}_{i+1}$  is more than twice the size of  $\text{CanYield}_i$

$\leadsto$  Size of  $\varphi_{p, \mathcal{M}, w}$  is in  $2^{O(p(n))}$ . Oops.

A correct reduction: We redefine  $\text{CanYield}$  by setting

$$\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) :=$$

$$\exists \bar{C}. \text{Conf}(\bar{C}) \wedge$$

$$\forall \bar{Z}_1. \forall \bar{Z}_2. (((\bar{Z}_1 = \bar{C}_1 \wedge \bar{Z}_2 = \bar{C}) \vee (\bar{Z}_1 = \bar{C} \wedge \bar{Z}_2 = \bar{C}_2)) \rightarrow \text{CanYield}_i(\bar{Z}_1, \bar{Z}_2))$$

# Size

Let's analyse the size more carefully this time:

$$\begin{aligned} \text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2) &:= \\ \exists \bar{C}. \text{Conf}(\bar{C}) \wedge \\ \forall \bar{Z}_1. \forall \bar{Z}_2. ((\bar{Z}_1 = \bar{C}_1 \wedge \bar{Z}_2 = \bar{C}) \vee (\bar{Z}_1 = \bar{C} \wedge \bar{Z}_2 = \bar{C}_2)) &\rightarrow \text{CanYield}_i(\bar{Z}_1, \bar{Z}_2) \end{aligned}$$

- $\text{CanYield}_{i+1}(\bar{C}_1, \bar{C}_2)$  extends  $\text{CanYield}_i(\bar{C}_1, \bar{C}_2)$  by parts that are linear in the size of configurations  $\leadsto$  growth in  $O(p(n))$
- Maximum index  $i$  used in  $\varphi_{p, \mathcal{M}, w}$  is  $dp(n)$ , that is in  $O(p(n))$
- Therefore:  $\varphi_{p, \mathcal{M}, w}$  has size  $O(p^2(n))$  – and thus can be computed in polynomial time

## Exercise:

Why can we just use  $dp(n)$  in the reduction? Don't we have to compute it somehow? Maybe even in polynomial time?

# The Power of QBF

**Theorem 10.4:** **TRUE QBF** is PSpace-complete.

## Proof:

(1) **TRUE QBF**  $\in$  PSpace:

Give an algorithm that runs in polynomial space.

(2) **TRUE QBF** is PSpace-hard:

Proof by reduction from the word problem for polynomially space-bounded TMs.

□



# A More Common Logical Problem in PSpace

Recall standard first-order logic:

- Instead of propositional variables, we have **atoms** (predicates with constants and variables)
- Instead of propositional evaluations we have **first-order structures** (or **interpretations**)
- First-order **quantifiers** can be used on variables
- **Sentences** are formulae where all variables are quantified
- A sentence can be **satisfied** or not by a given first-order structure

# A More Common Logical Problem in PSpace

Recall standard first-order logic:

- Instead of propositional variables, we have **atoms** (predicates with constants and variables)
- Instead of propositional evaluations we have **first-order structures** (or **interpretations**)
- First-order **quantifiers** can be used on variables
- **Sentences** are formulae where all variables are quantified
- A sentence can be **satisfied** or not by a given first-order structure

## **FOL MODEL CHECKING**

Input: A first-order sentence  $\varphi$  and a finite first-order structure  $\mathcal{I}$ .

Problem: Is  $\varphi$  satisfied by  $\mathcal{I}$ ?

# First-Order Logic is PSpace-complete

**Theorem 10.8: FOL MODEL CHECKING** is PSpace-complete.

## Proof:

- (1) **FOL MODEL CHECKING**  $\in$  PSpace:  
Give algorithm that runs in polynomial space.
- (2) **FOL MODEL CHECKING** is PSpace-hard:  
Proof by reduction **TRUE QBF**  $\leq_p$  **FOL MODEL CHECKING**.

# Checking FOL Models in Polynomial Space (Sketch)

```
01 EVAL( $\varphi, \mathcal{I}$ ) {
02   switch ( $\varphi$ ) :
03     case  $p(c_1, \dots, c_n)$  : return  $\langle c_1, \dots, c_n \rangle \in p^{\mathcal{I}}$ 
04     case  $\neg\psi$  : return NOT EVAL( $\psi, \mathcal{I}$ )
05     case  $\psi_1 \wedge \psi_2$  : return EVAL( $\psi_1, \mathcal{I}$ ) AND EVAL( $\psi_2, \mathcal{I}$ )
06     case  $\exists x.\psi$  :
07       for  $c \in \Delta^{\mathcal{I}}$  :
08         if EVAL( $\psi[x \mapsto c], \mathcal{I}$ ) : return TRUE
09       // eventually, if no success:
10       return FALSE
11 }
```

- We can assume  $\varphi$  only uses  $\neg$ ,  $\wedge$  and  $\exists$  (easy to get)
- We use  $\Delta^{\mathcal{I}}$  to denote the (finite!) domain of  $\mathcal{I}$
- We allow domain elements to be used like constants in the formula

# Hardness of FOL MODEL CHECKING

Given: a QBF  $\varphi = Q_1 X_1 \cdot \dots \cdot Q_\ell X_\ell \cdot \psi$

FOL Model Checking Problem:

- Interpretation domain  $\Delta^{\mathcal{I}} := \{0, 1\}$
- Single predicate symbol true with interpretation  $\text{true}^{\mathcal{I}} = \{\langle 1 \rangle\}$
- FOL formula  $\varphi'$  is obtained by replacing variables in input QBF with corresponding first-order expressions:

$$Q_1 x_1 \cdot \dots \cdot Q_\ell x_\ell \cdot \psi[X_1 \mapsto \text{true}(x_1), \dots, X_\ell \mapsto \text{true}(x_\ell)]$$

**Lemma 10.9:**  $\langle \mathcal{I}, \varphi' \rangle \in \text{FOL MODEL CHECKING}$  if and only if  $\varphi \in \text{TRUE QBF}$ .

# First-Order Logic is PSpace-complete

**Theorem 10.8: FOL MODEL CHECKING** is PSpace-complete.

## Proof:

- (1) **FOL MODEL CHECKING**  $\in$  PSpace:  
Give algorithm that runs in polynomial space.
- (2) **FOL MODEL CHECKING** is PSpace-hard:  
Proof by reduction **TRUE QBF**  $\leq_p$  **FOL MODEL CHECKING**.

□

# FOL MODEL CHECKING: Practical Significance

Why is **FOL MODEL CHECKING** a relevant problem?

# FOL MODEL CHECKING: Practical Significance

Why is **FOL MODEL CHECKING** a relevant problem?

Correspondence with database query answering:

- Finite first-order interpretation = database
- First-order logic formula = database query
- Satisfying assignments (for non-sentences) = query results

Known correspondence:

As a query language, FOL has the same expressive power as (basic) SQL (relational algebra).

**Corollary 10.10:** Answering SQL queries over a given database is PSpace-complete.



# Games

# Games as Computational Problems

Many single-player games relate to NP-complete problems:

- Sudoku
- Minesweeper
- Tetris
- ...

Decision problem: *Is there a solution?*

(For Tetris: is it possible to clear all blocks?)

What about *two-player* games?

# Games as Computational Problems

Many single-player games relate to NP-complete problems:

- Sudoku
- Minesweeper
- Tetris
- ...

Decision problem: *Is there a solution?*

(For Tetris: is it possible to clear all blocks?)

What about *two-player games*?

- Two players take moves in turns
- The players have different goals
- The game ends if a player wins

Decision problem: *Does Player 1 have a winning strategy?*

In other words: can Player 1 enforce winning, whatever Player 2 does?

# Example: The Formula Game

A contrived game, to illustrate the idea:

- Given: a propositional logic formula  $\varphi$  with consecutively numbered variables  $X_1, \dots, X_\ell$ .
- Two players take turns in selecting values for the next variable:
  - Player 1 sets  $X_1$  to true or false
  - Player 2 sets  $X_2$  to true or false
  - Player 1 sets  $X_3$  to true or false
  - ...

until all variables are set.

- Player 1 wins if the assignment makes  $\varphi$  true.  
Otherwise, Player 2 wins.

# Deciding the Formula Game

## **FORMULA GAME**

Input: A formula  $\varphi$ .

Problem: Does Player 1 have a winning strategy on  $\varphi$ ?

**Theorem 10.11:** **FORMULA GAME** is PSpace-complete.

# Deciding the Formula Game

## **FORMULA GAME**

Input: A formula  $\varphi$ .

Problem: Does Player 1 have a winning strategy on  $\varphi$ ?

**Theorem 10.11:** **FORMULA GAME** is PSpace-complete.

**Proof sketch:** **FORMULA GAME** is essentially the same as **TRUE QBF**.

Having a winning strategy means: there is a truth value for  $X_1$ , such that, for all truth values of  $X_2$ , there is a truth value of  $X_3, \dots$  such that  $\varphi$  becomes true.

If we have a QBF where quantifiers do not alternate, we can add dummy quantifiers and variables that do not change the semantics to get the same alternating form as for the Formula Game.  $\square$

# Example: The Geography Game

## A children's game:

- Two players are taking turns naming cities.
- Each city must start with the last letter of the previous.
- Repetitions are not allowed.
- The first player who cannot name a new city loses.

# Example: The Geography Game

## A children's game:

- Two players are taking turns naming cities.
- Each city must start with the last letter of the previous.
- Repetitions are not allowed.
- The first player who cannot name a new city loses.

## A mathematicians' game:

- Two players are marking nodes on a directed graph.
- Each node must be a successor of the previous one.
- Repetitions are not allowed.
- The first player who cannot mark a new node loses.



# Example: The Geography Game

## A children's game:

- Two players are taking turns naming cities.
- Each city must start with the last letter of the previous.
- Repetitions are not allowed.
- The first player who cannot name a new city loses.

## A mathematicians' game:

- Two players are marking nodes on a directed graph.
- Each node must be a successor of the previous one.
- Repetitions are not allowed.
- The first player who cannot mark a new node loses.

## Decision problem (**GENERALISED**) **GEOGRAPHY**:

given a graph and start node, does Player 1 have a winning strategy?

# GEOGRAPHY is PSpace-complete

**Theorem 10.12:** GENERALISED GEOGRAPHY is PSpace-complete.

## Proof:

(1) **GEOGRAPHY**  $\in$  PSpace:

Give algorithm that runs in polynomial space.

It is not difficult to provide a recursive algorithm similar to the one for **TRUE QBF** or **FOL MODEL CHECKING**.

(2) **GEOGRAPHY** is PSpace-hard:

Proof by reduction **FORMULA GAME**  $\leq_p$  **GEOGRAPHY**.

□

# GEOGRAPHY is PSpace-hard

Let  $\varphi$  with variables  $X_1, \dots, X_\ell$  be an instance of **FORMULA GAME**.

Without loss of generality, we assume:

- $\ell$  is odd (Player 1 gets the first and last turn)
- $\varphi$  is in CNF

We now build a graph that encodes **FORMULA GAME** in terms of **GEOGRAPHY**

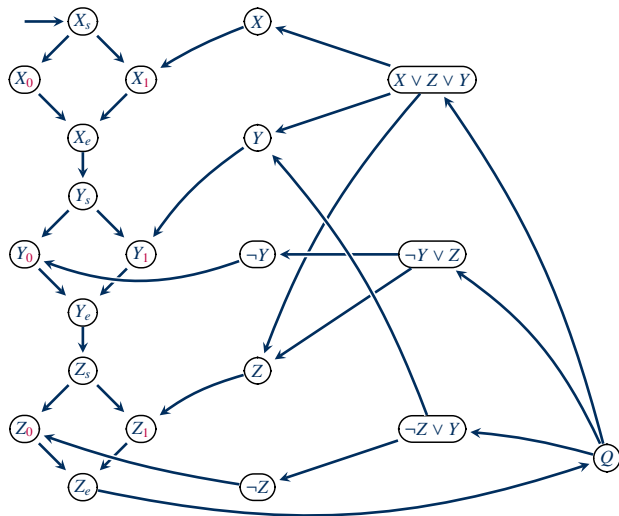
- The left-hand side of the graph is a chain of diamond structures that represent the choices that players have when assigning truth values
- The right-hand side of the graph encodes the structure of  $\varphi$ : Player 2 may choose a clause (trying to find one that is not true under the assignment); Player 1 may choose a literal (trying to find one that is true under the assignment).

(see board or [Sipser, Theorem 8.14])

□

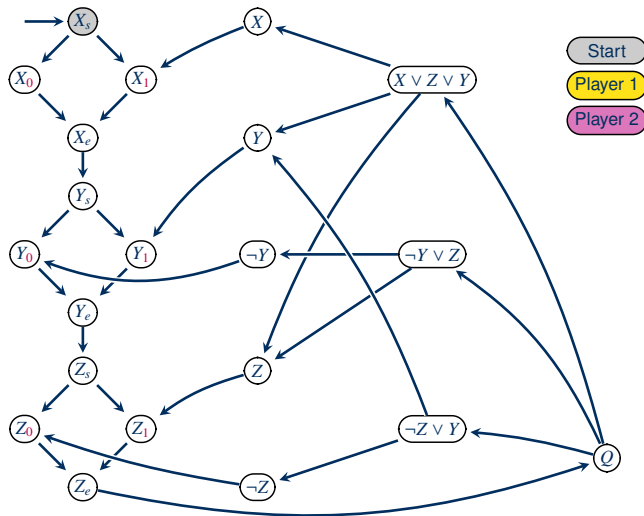
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



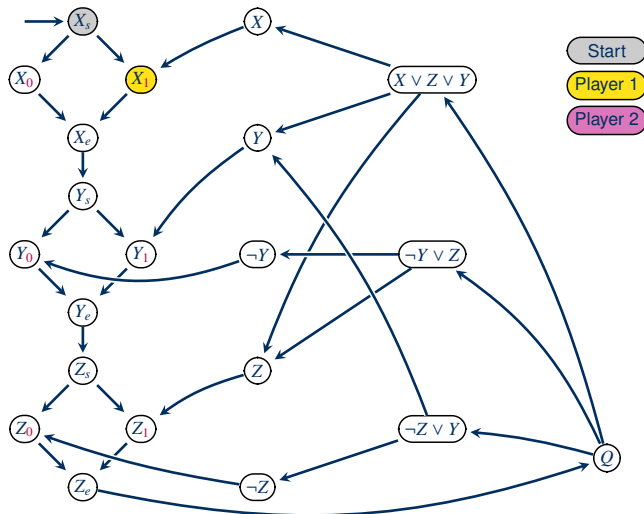
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



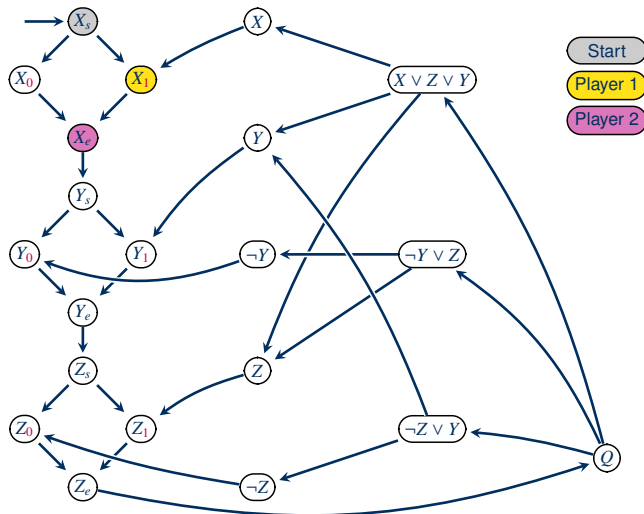
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



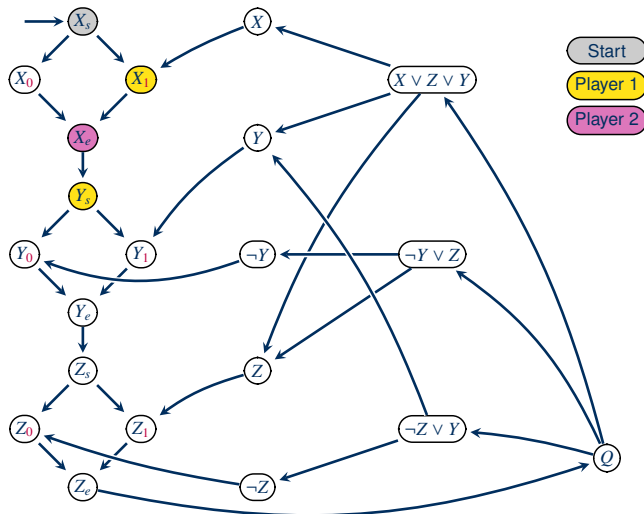
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



# GEOGRAPHY is PSpace-hard: Example

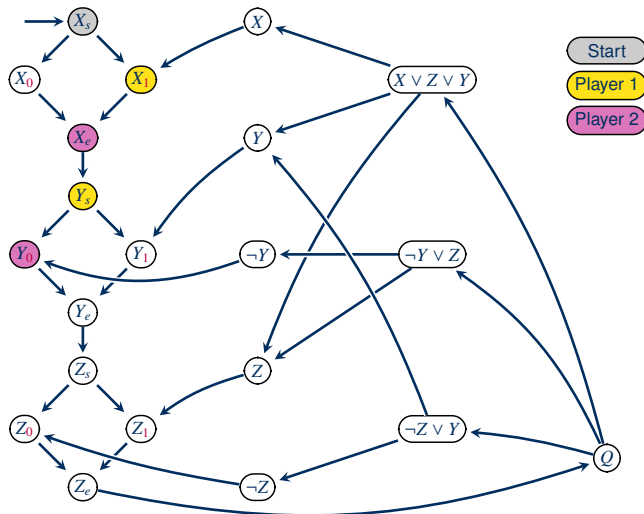
We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$





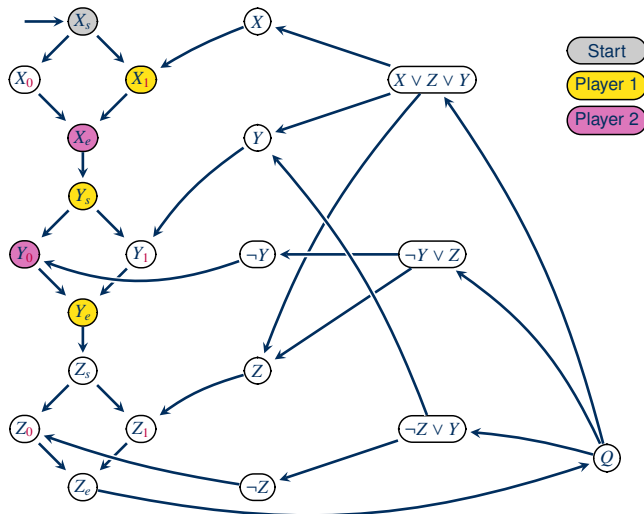
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



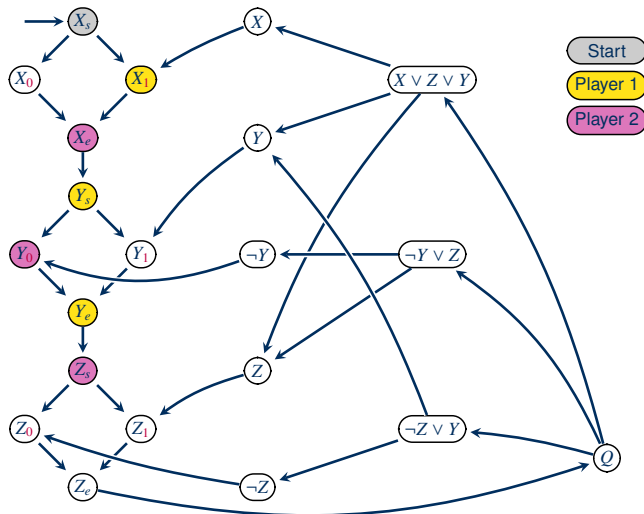
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X. \forall Y. \exists Z. (X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



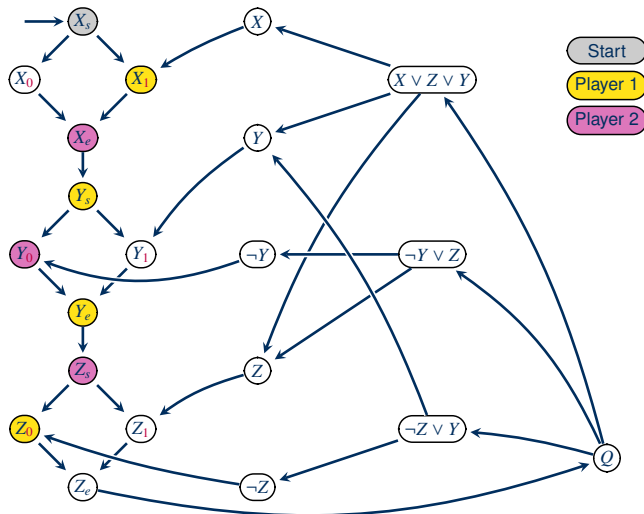
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X. \forall Y. \exists Z. (X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



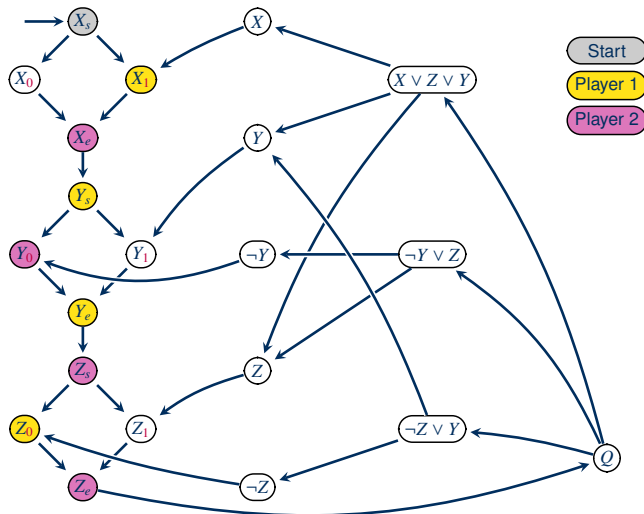
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



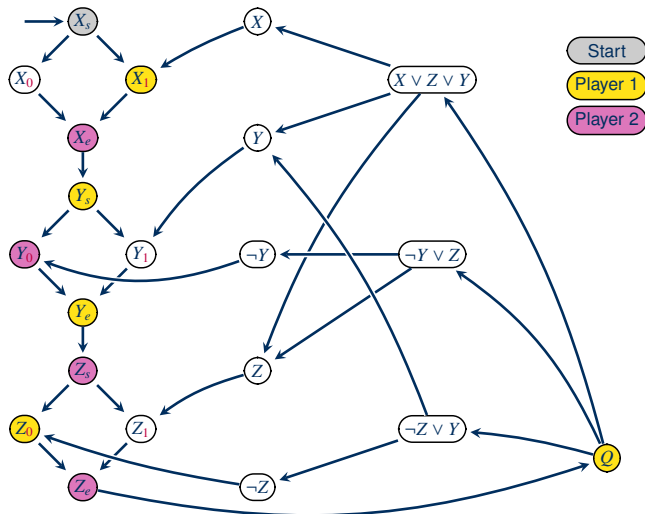
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



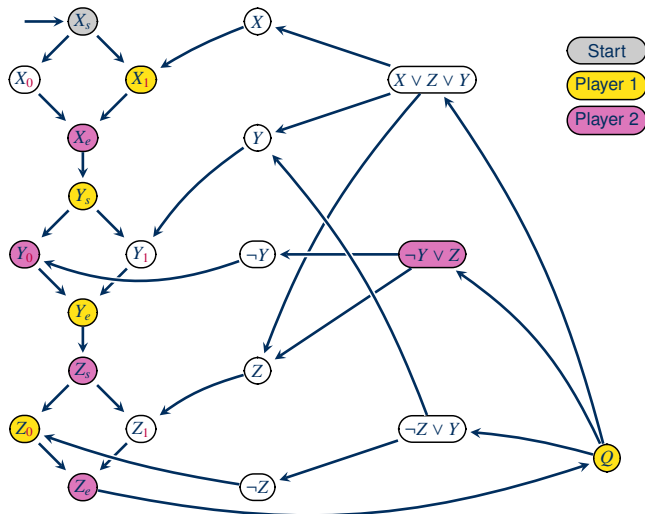
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



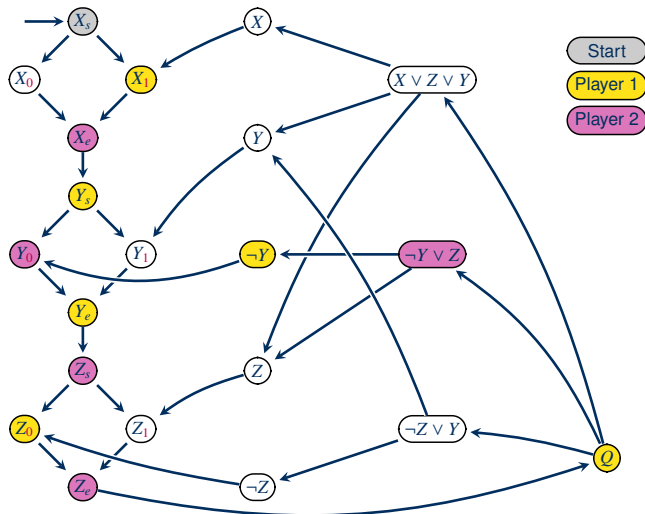
# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$



# GEOGRAPHY is PSpace-hard: Example

We consider the formula  $\exists X.\forall Y.\exists Z.(X \vee Z \vee Y) \wedge (\neg Y \vee Z) \wedge (\neg Z \vee Y)$





# Summary and Outlook

**TRUE QBF** is PSpace-complete

**FOL MODEL CHECKING** and the related problem of SQL query answering are PSpace-complete

Some games are PSpace-complete

## What's next?

- Some more remarks on games
- Logarithmic space
- Complements of space classes