Computability and Complexity
COSC 4200

PSPACE-Completeness

PSPACE-Completeness

Definition

A problem B is PSPACE-*complete* if

1. B is in PSPACE, and
2. every A in PSPACE is $\leq_{\mathrm{P}}$-reducible to B.

PSPACE-Completeness

Definition

A problem B is PSPACE-*complete* if

1. B is in PSPACE, and
2. every A in PSPACE is $\leq_{\mathrm{P}}$-reducible to B.

Theorem

The following are equivalent:

1. *Some* PSPACE-*complete problem is in* P.
2. *Every* PSPACE-*complete problem is in* P.
3. $\mathrm{P} = \mathrm{PSPACE}$.

Quantified Boolean Formulae

A *quantified Boolean formula* (QBF) is a propositional formula preceded by quantifiers over the variables. There are no free variables. An example is

$$\phi = (\exists x_1)(\forall x_2)(\exists x_3)(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_1).$$

Quantified Boolean Formulae

A *quantified Boolean formula* (QBF) is a propositional formula preceded by quantifiers over the variables. There are no free variables. An example is

$$\phi = (\exists x_1)(\forall x_2)(\exists x_3)(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_1).$$

The order in which the variables are quantified matters. You can think of it terms of a game (FORMULA-GAME in the textbook).

- Player I picks the values for the existentially quantified variables, and Player II picks the values for the universally quantified variables.

Quantified Boolean Formulae

A *quantified Boolean formula* (QBF) is a propositional formula preceded by quantifiers over the variables. There are no free variables. An example is

$$\phi = (\exists x_1)(\forall x_2)(\exists x_3)(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_1).$$

The order in which the variables are quantified matters. You can think of it terms of a game (FORMULA-GAME in the textbook).

- Player I picks the values for the existentially quantified variables, and Player II picks the values for the universally quantified variables.
- Player I's goal is to satisfy the formula; Player II's goal is to avoid satisfying the formula.

Quantified Boolean Formulae

A *quantified Boolean formula* (QBF) is a propositional formula preceded by quantifiers over the variables. There are no free variables. An example is

$$\phi = (\exists x_1)(\forall x_2)(\exists x_3)(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_1).$$

The order in which the variables are quantified matters. You can think of it terms of a game (FORMULA-GAME in the textbook).

- Player I picks the values for the existentially quantified variables, and Player II picks the values for the universally quantified variables.
- Player I's goal is to satisfy the formula; Player II's goal is to avoid satisfying the formula.
- In our example, Player I picks the value for x_1. Then Player II picks a value for x_2. Lastly, Player I picks a value for x_3.

Quantified Boolean Formulae

A *quantified Boolean formula* (QBF) is a propositional formula preceded by quantifiers over the variables. There are no free variables. An example is

$$\phi = (\exists x_1)(\forall x_2)(\exists x_3)(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_1).$$

The order in which the variables are quantified matters. You can think of it terms of a game (FORMULA-GAME in the textbook).

- Player I picks the values for the existentially quantified variables, and Player II picks the values for the universally quantified variables.
- Player I's goal is to satisfy the formula; Player II's goal is to avoid satisfying the formula.
- In our example, Player I picks the value for x_1. Then Player II picks a value for x_2. Lastly, Player I picks a value for x_3.
- Player I wins when he picks an x_1 such that no matter what x_2 Player II picks, he can always pick a value for x_3 that satisfies the formula.

The TQBF Problem

Consider the problem

$$\mathrm{TQBF} = \{\phi \mid \phi \text{ is a true QBF}\}.$$

Since TQBF involves both $\exists$ and $\forall$ quantifiers, it does not appear that $\mathrm{TQBF} \in \mathrm{NP}$ or $\mathrm{TQBF} \in \mathrm{coNP}$.

Theorem

TQBF *is* PSPACE*-complete.*

Proof. We show that $\mathrm{TQBF} \in \mathrm{PSPACE}$ using the following recursive algorithm.

truth(Ψ)
    if Ψ has no quantifiers,
        then return its truth value

Proof. We show that $\mathrm{TQBF} \in \mathrm{PSPACE}$ using the following recursive algorithm.

truth(Ψ)

if Ψ has no quantifiers,

then return its truth value

// *otherwise* $\Psi = Q_1 x_{i_1} \ldots Q_k x_{i_k} \phi(x_1, \ldots, x_n)$

// *where each* $Q_j \in \{\exists, \forall\}$ *and each* $i_j \in \{1, \ldots, n\}$

Proof. We show that $\mathrm{TQBF} \in \mathrm{PSPACE}$ using the following recursive algorithm.

truth(Ψ)
    if Ψ has no quantifiers,
        then return its truth value

    // *otherwise* $\Psi = Q_1 x_{i_1} \ldots Q_k x_{i_k} \phi(x_1, \ldots, x_n)$
    // *where each* $Q_j \in \{\exists, \forall\}$ *and each* $i_j \in \{1, \ldots, n\}$

    $b_0 = truth(Q_2 x_{i_2} \ldots Q_k x_{i_k} \phi(x_1, \ldots, x_{i_1} = F, \ldots, x_n))$
        // *recursively evaluate substituting* $x_{i_1} = F$

Proof. We show that $\mathrm{TQBF} \in \mathrm{PSPACE}$ using the following recursive algorithm.

truth(Ψ)

if Ψ has no quantifiers,

then return its truth value

// otherwise $\Psi = Q_1 x_{i_1} \ldots Q_k x_{i_k} \phi(x_1, \ldots, x_n)$

// where each $Q_j \in \{\exists, \forall\}$ *and each* $i_j \in \{1, \ldots, n\}$

$b_0 = \mathit{truth}(Q_2 x_{i_2} \ldots Q_k x_{i_k} \phi(x_1, \ldots, x_{i_1} = F, \ldots, x_n))$

// recursively evaluate substituting $x_{i_1} = F$

$b_1 = \mathit{truth}(Q_2 x_{i_2} \ldots Q_k x_{i_k} \phi(x_1, \ldots, x_{i_1} = T, \ldots, x_n))$

// recursively evaluate substituting $x_{i_1} = T$

Proof. We show that $\mathrm{TQBF} \in \mathrm{PSPACE}$ using the following recursive algorithm.

truth(Ψ)

if Ψ has no quantifiers,

then return its truth value

// otherwise $\Psi = Q_1 x_{i_1} \dots Q_k x_{i_k} \phi(x_1, \dots, x_n)$

// where each $Q_j \in \{\exists, \forall\}$ *and each* $i_j \in \{1, \dots, n\}$

$b_0 = \mathit{truth}(Q_2 x_{i_2} \dots Q_k x_{i_k} \phi(x_1, \dots, x_{i_1} = F, \dots, x_n))$

// recursively evaluate substituting $x_{i_1} = F$

$b_1 = \mathit{truth}(Q_2 x_{i_2} \dots Q_k x_{i_k} \phi(x_1, \dots, x_{i_1} = T, \dots, x_n))$

// recursively evaluate substituting $x_{i_1} = T$

if $Q_1 = \exists$,

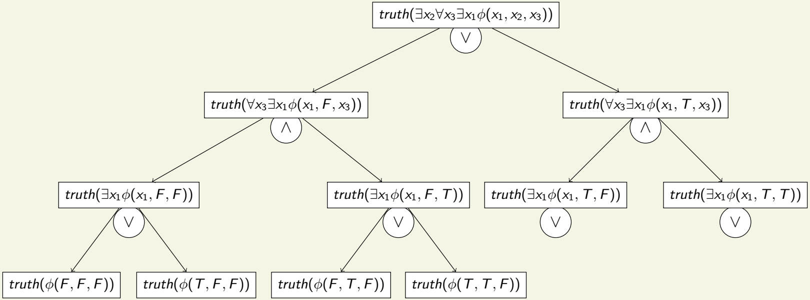
then return $b_0 \vee b_1$

Proof. We show that $\mathrm{TQBF} \in \mathrm{PSPACE}$ using the following recursive algorithm.

```
truth(Ψ)
    if Ψ has no quantifiers,
        then return its truth value

    // otherwise Ψ = Q_1 x_{i_1} ... Q_k x_{i_k} φ(x_1, ..., x_n)
    // where each Q_j ∈ {∃, ∀} and each i_j ∈ {1, ..., n}

    b_0 = truth(Q_2 x_{i_2} ... Q_k x_{i_k} φ(x_1, ..., x_{i_1} = F, ..., x_n))
        // recursively evaluate substituting x_{i_1} = F
    b_1 = truth(Q_2 x_{i_2} ... Q_k x_{i_k} φ(x_1, ..., x_{i_1} = T, ..., x_n))
        // recursively evaluate substituting x_{i_1} = T

    if Q_1 = ∃,
        then return b_0 ∨ b_1
    if Q_1 = ∀,
        then return b_0 ∧ b_1
```

Example

As an example, let us consider $\Psi = \exists x_2 \forall x_3 \exists x_1 \phi(x_1, x_2, x_3)$.



In the end, the algorithm has evaluated

$$((\phi(FFF) \vee \phi(TFF)) \wedge (\phi(FFT) \vee \phi(TFT))) \vee ((\phi(FTF) \vee \phi(TTF)) \wedge (\phi(FTT) \vee \phi(TTT))).$$

TQBF is in PSPACE

We have $\Psi \in \mathrm{TQBF} \iff \mathit{truth}(\Psi) = T$.

This algorithm produces an exponential-size tree: if ϕ has n variables, the depth is n and there are 2^n leaves. This means that the algorithm takes exponential time.

However, the algorithm uses only polynomial space because it traverses the tree using depth-first search. Therefore

$$\mathrm{TQBF} \in \mathrm{PSPACE}.$$

To show that TQBF is PSPACE-complete, let $A \in \mathrm{PSPACE}$ be arbitrary. Let M be a polynomial-space bounded TM that decides A, say in space $p(n)$. We need to show that $A \leq_{\mathrm{P}} \mathrm{TQBF}$.

To show that TQBF is PSPACE-complete, let $A \in \mathrm{PSPACE}$ be arbitrary. Let M be a polynomial-space bounded TM that decides A, say in space $p(n)$. We need to show that $A \leq_{\mathrm{P}} \mathrm{TQBF}$.

For convenience, we can assume that M has a unique accepting configuration F. (Recall that if it doesn't have this property, we can modify M so it does. We change M to clear its work tapes and put all tape heads on the leftmost cells before accepting.)

For two configurations a and b of M and a number t, let's define the predicate

$$reach(a, b, t) = \begin{cases} T & \text{if } M \text{ can go from } a \text{ to } b \text{ in at most } 2^t \text{ steps} \\ F & \text{otherwise.} \end{cases}$$

Let $x \in \{0, 1\}^*$ and let C_x be the initial configuration of M on input x.

For two configurations a and b of M and a number t, let's define the predicate

$$reach(a, b, t) = \begin{cases} T & \text{if } M \text{ can go from } a \text{ to } b \text{ in at most } 2^t \text{ steps} \\ F & \text{otherwise.} \end{cases}$$

Let $x \in \{0, 1\}^*$ and let C_x be the initial configuration of M on input x.

- Then
$$x \in A \iff reach(C_x, F, m),$$
where 2^m is a bound on the number of configurations M can use.

For two configurations a and b of M and a number t, let's define the predicate

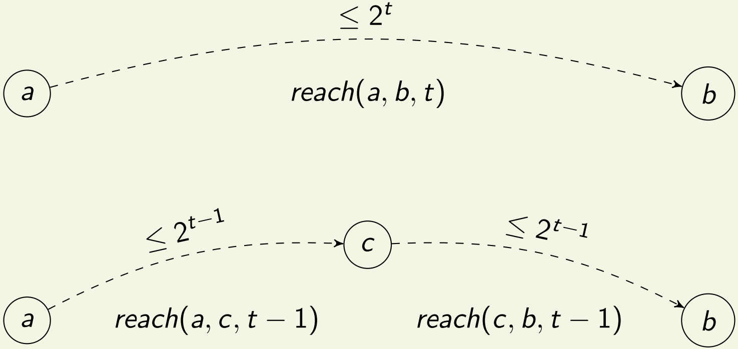
$$reach(a, b, t) = \begin{cases} T & \text{if } M \text{ can go from } a \text{ to } b \text{ in at most } 2^t \text{ steps} \\ F & \text{otherwise.} \end{cases}$$

Let $x \in \{0, 1\}^*$ and let C_x be the initial configuration of M on input x.

- Then

  $$x \in A \iff reach(C_x, F, m),$$

  where 2^m is a bound on the number of configurations M can use.
- The total number of configurations is $O(2^{cp(|x|)})$ for some constant c, so we can set $m = dp(|x|)$ for some constant d.



Observe that

$$\mathit{reach}(a, b, t) \iff (\exists c)\ \mathit{reach}(a, c, t-1) \wedge \mathit{reach}(c, b, t-1).$$

If M goes from a to b in at most 2^t steps, then there is a midpoint c such that M goes

- from a to c in at most 2^{t-1} steps and
- from c to b in at most 2^{t-1} steps.

For each i, $i = 0, \ldots, m$, we define a formula $\phi_i(X, Y)$ for *reach*(X, Y, i) as follows. We use capital letters denote a sequence of variables that encode configurations of M.

For each i, $i = 0, \ldots, m$, we define a formula $\phi_i(X, Y)$ for *reach*(X, Y, i) as follows. We use capital letters denote a sequence of variables that encode configurations of M.

First, we consider $i = 0$, the base case.

$\phi_0(X, Y)$ is true if X and Y encode configurations, and either $X = Y$ or Y can be reached from X in one step. In other words,

$$\phi_0(X, Y) \iff \mathit{reach}(X, Y, 0).$$

For each i, $i = 0, \ldots, m$, we define a formula $\phi_i(X, Y)$ for *reach*(X, Y, i) as follows. We use capital letters denote a sequence of variables that encode configurations of M.

First, we consider $i = 0$, the base case.

$\phi_0(X, Y)$ is true if X and Y encode configurations, and either $X = Y$ or Y can be reached from X in one step. In other words,

$$\phi_0(X, Y) \iff \textit{reach}(X, Y, 0).$$

A configuration of M takes m bits to encode. X and Y can be considered sequences of m variables, $x_1, \ldots, x_m$ and $y_1, \ldots, y_m$. Checking whether $X = Y$ or whether Y can be reached from X in one step can be done with a Boolean formula. This allows us to do is to express the base case of *reach*.

Let $i \geq 0$. Given ϕ_i, we'd like to define ϕ_{i+1} so that

$$\phi_{i+1}(X, Y) \iff \mathit{reach}(X, Y, i+1).$$

Let $i \geq 0$. Given ϕ_i, we'd like to define ϕ_{i+1} so that

$$\phi_{i+1}(X, Y) \iff \mathit{reach}(X, Y, i+1).$$

We're trying to express the reachability of Y from X in 2^{i+1} steps.

Let $i \geq 0$. Given ϕ_i, we'd like to define ϕ_{i+1} so that

$$\phi_{i+1}(X, Y) \iff \mathit{reach}(X, Y, i+1).$$

We're trying to express the reachability of Y from X in 2^{i+1} steps.

A first attempt is the following:

$$\phi_{i+1}(X, Y) = \exists Z[\phi_i(X, Z) \wedge \phi_i(Z, Y)]$$

This corresponds directly to *reach*, and is therefore a correct formula, but there is a problem.

Let $i \geq 0$. Given ϕ_i, we'd like to define ϕ_{i+1} so that

$$\phi_{i+1}(X, Y) \iff \mathit{reach}(X, Y, i+1).$$

We're trying to express the reachability of Y from X in 2^{i+1} steps.

A first attempt is the following:

$$\phi_{i+1}(X, Y) = \exists Z[\phi_i(X, Z) \wedge \phi_i(Z, Y)]$$

This corresponds directly to *reach*, and is therefore a correct formula, but there is a problem.

With each step, we double the size of the formula. Ultimately, we want to output ϕ_m, but with this doubling, ϕ_m will have exponential size, which is too large to output in polynomial time.

Here is a better way to accomplish this task. Notice that thus far we have only used existential quantifiers. The next idea is to use universal quantifiers to help create a more succinct formula. Each ϕ_{i+1} includes only one copy of ϕ_i.

$$
\begin{aligned}
&\phi_{i+1}(X, Y) \\
&= (\exists Z_{i+1})(\forall A_{i+1})(\forall B_{i+1}) \\
&\quad [(A_{i+1} = X \wedge B_{i+1} = Z_{i+1}) \vee (A_{i+1} = Z_{i+1} \wedge B_{i+1} = Y)] \implies \phi_i(A_{i+1}, B_{i+1})
\end{aligned}
$$

Here is a better way to accomplish this task. Notice that thus far we have only used existential quantifiers. The next idea is to use universal quantifiers to help create a more succinct formula. Each ϕ_{i+1} includes only one copy of ϕ_i.

$$\begin{aligned}
&\phi_{i+1}(X, Y) \\
&= (\exists Z_{i+1})(\forall A_{i+1})(\forall B_{i+1}) \\
&\quad [(A_{i+1} = X \wedge B_{i+1} = Z_{i+1}) \vee (A_{i+1} = Z_{i+1} \wedge B_{i+1} = Y)] \implies \phi_i(A_{i+1}, B_{i+1})
\end{aligned}$$

- We use an implication here, but it can be rewritten using negation and disjunction. The implication helps with the intuition.

Here is a better way to accomplish this task. Notice that thus far we have only used existential quantifiers. The next idea is to use universal quantifiers to help create a more succinct formula. Each ϕ_{i+1} includes only one copy of ϕ_i.

$$
\begin{aligned}
&\phi_{i+1}(X, Y) \\
&= (\exists Z_{i+1})(\forall A_{i+1})(\forall B_{i+1}) \\
&\quad [(A_{i+1} = X \wedge B_{i+1} = Z_{i+1}) \vee (A_{i+1} = Z_{i+1} \wedge B_{i+1} = Y)] \implies \phi_i(A_{i+1}, B_{i+1})
\end{aligned}
$$

- We use an implication here, but it can be rewritten using negation and disjunction. The implication helps with the intuition.
- The idea is, looking over all possible configurations A and B:
  - if $A = X$ and $B = Z$, then we can reach Z from X in 2^i steps, or
  - if $A = Z$ and $B = Y$, then we can reach Y from Z in 2^i steps.

Here is a better way to accomplish this task. Notice that thus far we have only used existential quantifiers. The next idea is to use universal quantifiers to help create a more succinct formula. Each ϕ_{i+1} includes only one copy of ϕ_i.

$$\begin{aligned}
&\phi_{i+1}(X, Y) \\
&= (\exists Z_{i+1})(\forall A_{i+1})(\forall B_{i+1}) \\
&\quad [(A_{i+1} = X \wedge B_{i+1} = Z_{i+1}) \vee (A_{i+1} = Z_{i+1} \wedge B_{i+1} = Y)] \implies \phi_i(A_{i+1}, B_{i+1})
\end{aligned}$$

- We use an implication here, but it can be rewritten using negation and disjunction. The implication helps with the intuition.
- The idea is, looking over all possible configurations A and B:
  - if $A = X$ and $B = Z$, then we can reach Z from X in 2^i steps, or
  - if $A = Z$ and $B = Y$, then we can reach Y from Z in 2^i steps.
- This is equivalent to our first attempt, but produces a much smaller formula. Note that subscripts are present to rename variables so we are not using the same variable multiple times.

Once we have completed the construction of ϕ_m, we can rewrite it so that it has all of its quantifiers up front. We can do this because the variables bound by a quantifier are never reused.

Once we have completed the construction of ϕ_m, we can rewrite it so that it has all of its quantifiers up front. We can do this because the variables bound by a quantifier are never reused.

We have

$$\begin{aligned} x \in A \quad &\iff \quad \mathit{reach}(C_x, F, m) \\ &\iff \quad \phi_m(C_x, F) \in \mathrm{TQBF}. \end{aligned}$$

Once we have completed the construction of ϕ_m, we can rewrite it so that it has all of its quantifiers up front. We can do this because the variables bound by a quantifier are never reused.

We have

$$\begin{aligned} x \in A &\iff \mathit{reach}(C_x, F, m) \\ &\iff \phi_m(C_x, F) \in \mathrm{TQBF}. \end{aligned}$$

We can compute $f(x) = \phi_m(C_x, F)$ in polynomial time, so $A \leq_{\mathrm{P}} \mathrm{TQBF}$ via f.

Once we have completed the construction of ϕ_m, we can rewrite it so that it has all of its quantifiers up front. We can do this because the variables bound by a quantifier are never reused.

We have

$$\begin{aligned} x \in A \quad &\iff \quad \mathit{reach}(C_x, F, m) \\ &\iff \quad \phi_m(C_x, F) \in \mathrm{TQBF}. \end{aligned}$$

We can compute $f(x) = \phi_m(C_x, F)$ in polynomial time, so $A \leq_{\mathrm{P}} \mathrm{TQBF}$ via f.

Since A was an arbitrary member of PSPACE, we have shown that TQBF is PSPACE-complete. $\square$

We can use transitivity of $\leq_{\mathrm{P}}$ to show that more problems are PSPACE-complete, just like we did for NP-completeness.

Proposition

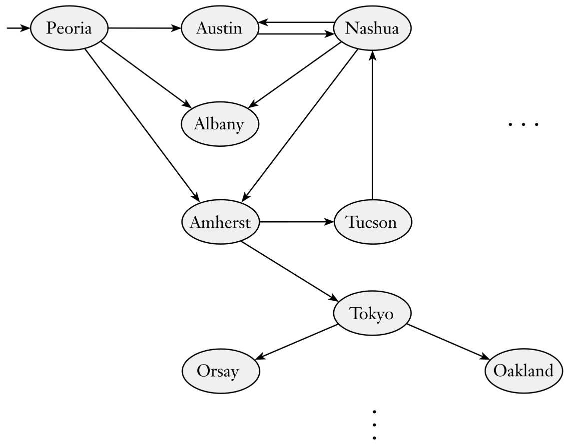
Let $B, C \in \mathrm{PSPACE}$. If B is PSPACE-complete and $B \leq_{\mathrm{P}} C$, then C is also PSPACE-complete.

Proof. Because B is PSPACE-complete, $A \leq_{\mathrm{P}} B$ for every $A \in \mathrm{PSPACE}$. By transitivity and $B \leq_{\mathrm{P}} C$, we have $A \leq_{\mathrm{P}} C$ for every $A \in \mathrm{PSPACE}$. $\square$

To prove that a problem $B \in \mathrm{PSPACE}$ is PSPACE-complete, we don't have to reduce every problem in PSPACE to it. We may select any PSPACE-complete problem A and show that $A \leq_{\mathrm{P}} B$.

Geography Game

Two players take turns naming cities. Each city chosen must begin with the same letter that ended the previous city's name.



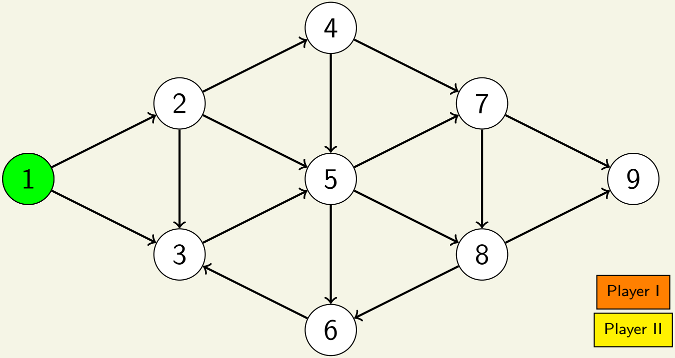
For example:

- I: Peoria
- II: Amherst
- I: Tucson
- II: Nashua
- ⋮

The game ends when one player gets stuck.

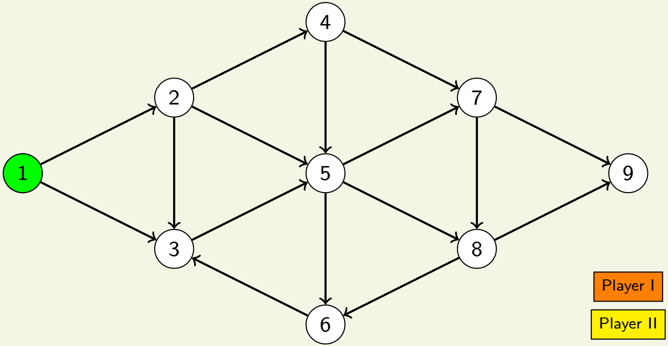
Generalized Geography

Use an arbitrary directed graph and start vertex.



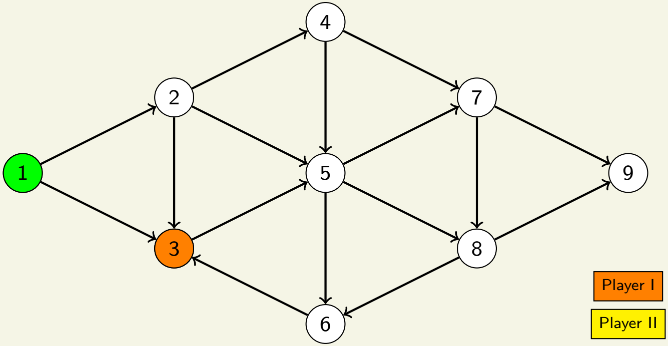
Player I has a winning strategy:

- The game begins at node 1, which points to nodes 2 and 3. Player I chooses 3.
- Then Player II must move, but 3 only points to 5. Player II chooses 5.
- Player I now has 6, 7, and 8 available. Player I chooses 6.
- Now player II is stuck because 6 points only to 3, and 3 was already played.



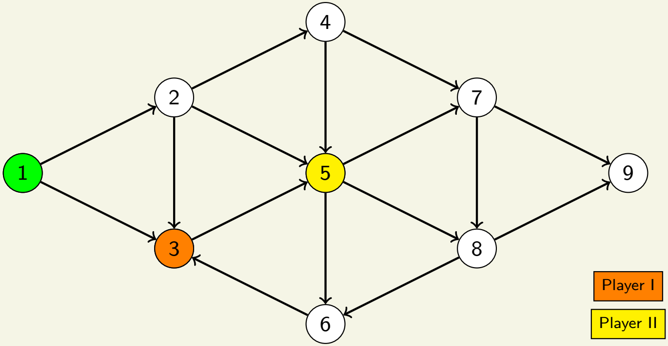
Player I has a winning strategy:

- The game begins at node 1, which points to nodes 2 and 3. Player I chooses 3.
- Then Player II must move, but 3 only points to 5. Player II chooses 5.
- Player I now has 6, 7, and 8 available. Player I chooses 6.
- Now player II is stuck because 6 points only to 3, and 3 was already played.



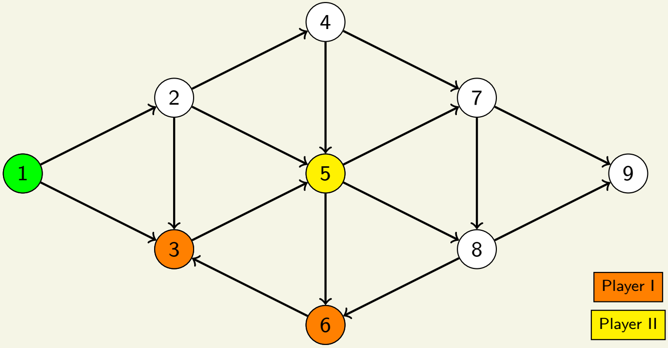
Player I has a winning strategy:

- The game begins at node 1, which points to nodes 2 and 3. Player I chooses 3.
- Then Player II must move, but 3 only points to 5. Player II chooses 5.
- Player I now has 6, 7, and 8 available. Player I chooses 6.
- Now player II is stuck because 6 points only to 3, and 3 was already played.



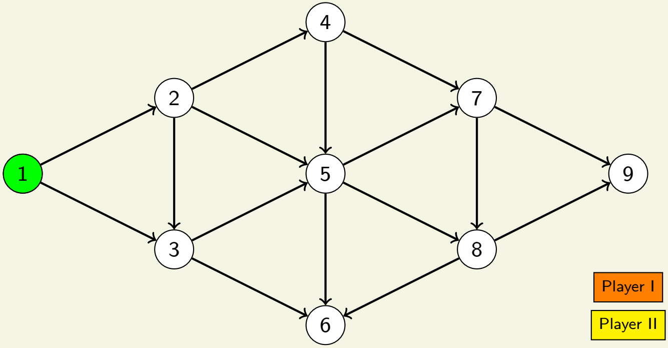
Player I has a winning strategy:

- The game begins at node 1, which points to nodes 2 and 3. Player I chooses 3.
- Then Player II must move, but 3 only points to 5. Player II chooses 5.
- Player I now has 6, 7, and 8 available. Player I chooses 6.
- Now player II is stuck because 6 points only to 3, and 3 was already played.



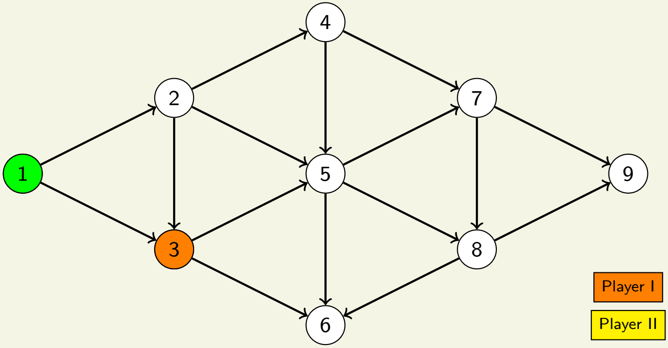
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
  - If Player I chooses 3, then Player II chooses 6 and wins.
  - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
  - If Player I chooses 5, then Player II chooses 6 and wins.
  - If Player I chooses 7, then Player II chooses 9 and wins.



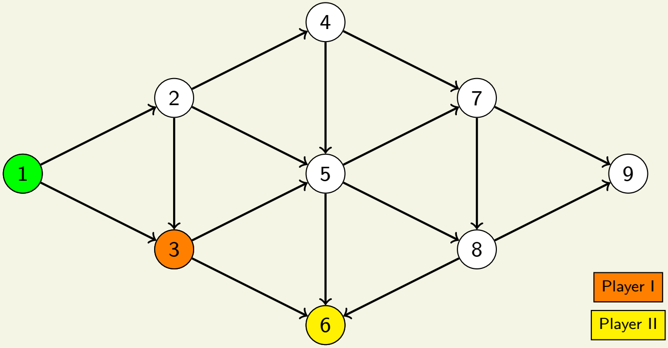
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
  - If Player I chooses 3, then Player II chooses 6 and wins.
  - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
  - If Player I chooses 5, then Player II chooses 6 and wins.
  - If Player I chooses 7, then Player II chooses 9 and wins.



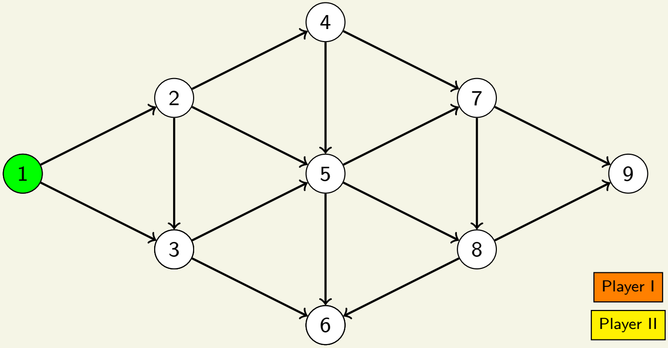
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
  - If Player I chooses 3, then Player II chooses 6 and wins.
  - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
  - If Player I chooses 5, then Player II chooses 6 and wins.
  - If Player I chooses 7, then Player II chooses 9 and wins.



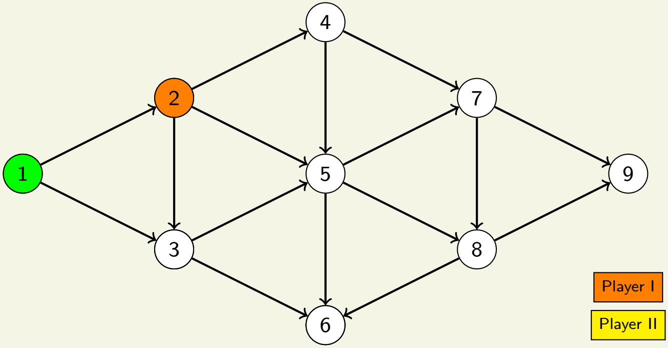
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
    - If Player I chooses 3, then Player II chooses 6 and wins.
    - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
    - If Player I chooses 5, then Player II chooses 6 and wins.
    - If Player I chooses 7, then Player II chooses 9 and wins.



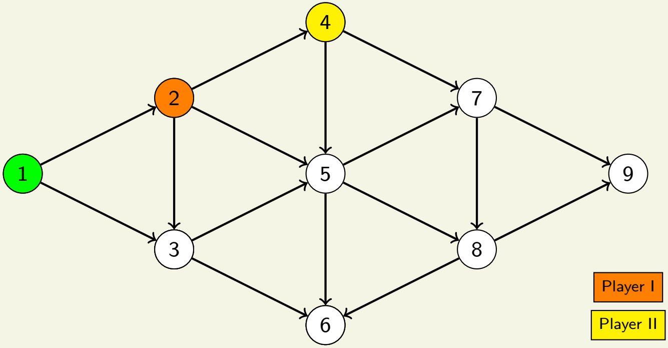
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
  - If Player I chooses 3, then Player II chooses 6 and wins.
  - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
  - If Player I chooses 5, then Player II chooses 6 and wins.
  - If Player I chooses 7, then Player II chooses 9 and wins.



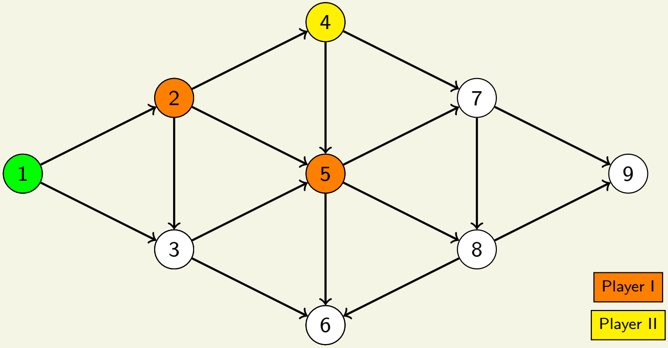
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
  - If Player I chooses 3, then Player II chooses 6 and wins.
  - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
  - If Player I chooses 5, then Player II chooses 6 and wins.
  - If Player I chooses 7, then Player II chooses 9 and wins.



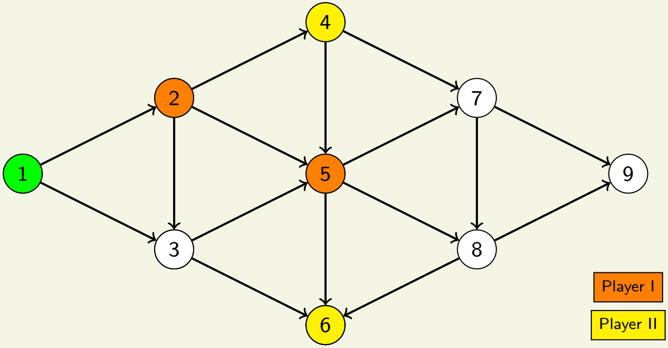
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
  - If Player I chooses 3, then Player II chooses 6 and wins.
  - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
  - If Player I chooses 5, then Player II chooses 6 and wins.
  - If Player I chooses 7, then Player II chooses 9 and wins.



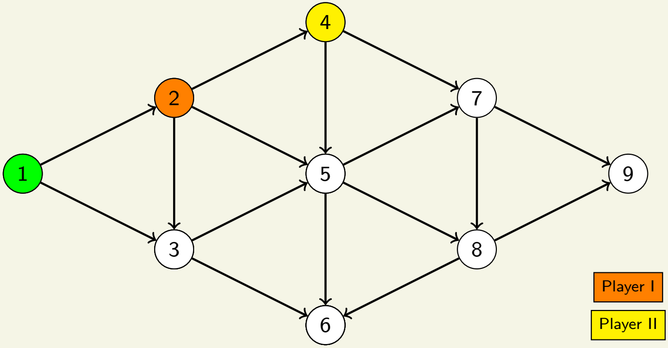
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
    - If Player I chooses 3, then Player II chooses 6 and wins.
    - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
    - If Player I chooses 5, then Player II chooses 6 and wins.
    - If Player I chooses 7, then Player II chooses 9 and wins.



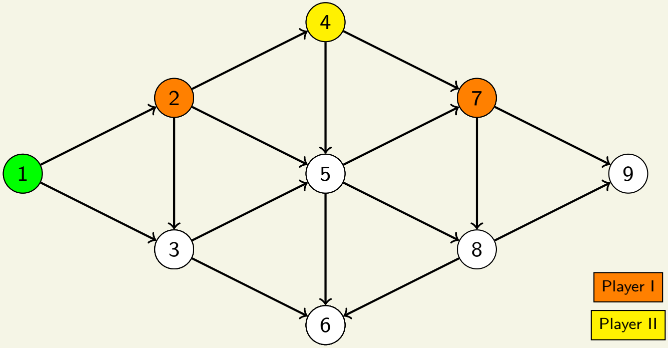
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
  - If Player I chooses 3, then Player II chooses 6 and wins.
  - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
  - If Player I chooses 5, then Player II chooses 6 and wins.
  - If Player I chooses 7, then Player II chooses 9 and wins.



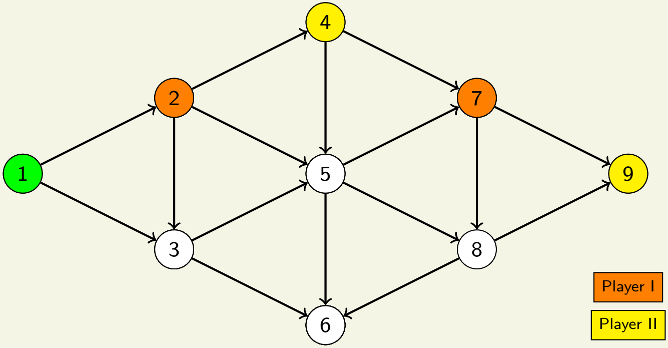
In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
    - If Player I chooses 3, then Player II chooses 6 and wins.
    - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
    - If Player I chooses 5, then Player II chooses 6 and wins.
    - If Player I chooses 7, then Player II chooses 9 and wins.



In the instance below, the edge from 6 to 3 has been reversed.
Player II now has a winning strategy:

- The game begins at node 1, which points to 2 and 3.
    - If Player I chooses 3, then Player II chooses 6 and wins.
    - Suppose Player I chooses 2.
- Now Player II may choose from 3, 4, and 5. Player II chooses 4.
- Now Player I may choose from 5 or 7.
    - If Player I chooses 5, then Player II chooses 6 and wins.
    - If Player I chooses 7, then Player II chooses 9 and wins.



Generalized Geography

In the Generalized Geography decision problem, the goal is to determine whether Player I has a winning strategy.

$$\mathrm{GG} = \left\{ \langle G, b \rangle \;\middle|\; \begin{array}{l} \text{Player I has a winning strategy for the} \\ \text{generalized geography game played on} \\ \text{graph } G \text{ starting at node } b \end{array} \right\}$$

Theorem

GG *is* PSPACE*-complete.*

First, we show that $\mathrm{GG} \in \mathrm{PSPACE}$.

M: on input $\langle G, b \rangle$, where G is a directed graph and b is a node of G:
    if b has outdegree 0
        reject *// Player I loses immediately*

First, we show that $\mathrm{GG} \in \mathrm{PSPACE}$.

M: on input $\langle G, b\rangle$, where G is a directed graph and b is a node of G:
  if b has outdegree 0
    reject *// Player I loses immediately*

  remove node b and all connected edges to get a new graph G'
  for each node $b_1, \ldots, b_k$ that b originally pointed at
    recursively call $M(\langle G', b_i\rangle)$

First, we show that $\mathrm{GG} \in \mathrm{PSPACE}$.

M: on input $\langle G, b\rangle$, where G is a directed graph and b is a node of G:

if b has outdegree 0

reject *// Player I loses immediately*

remove node b and all connected edges to get a new graph G'

for each node $b_1, \ldots, b_k$ that b originally pointed at

recursively call $M(\langle G', b_i\rangle)$

if all of these recursive calls accept

reject *// Player II has a winning strategy*

else

accept *// Player I has a winning strategy*

First, we show that $\mathrm{GG} \in \mathrm{PSPACE}$.

```
M: on input ⟨G, b⟩, where G is a directed graph and b is a node of G:
    if b has outdegree 0
        reject // Player I loses immediately

    remove node b and all connected edges to get a new graph G′
    for each node b_1, ..., b_k that b originally pointed at
        recursively call M(⟨G′, b_i⟩)

    if all of these recursive calls accept
        reject // Player II has a winning strategy
    else
        accept // Player I has a winning strategy
```

If there are n vertices in G, the recursion depth is at most n. Each recursive call uses polynomial space. Thus the total space used is polynomial.

To show GG is PSPACE-complete, we will reduce $\mathrm{TQBF} \leq_{\mathrm{P}} \mathrm{GG}$.

To show GG is PSPACE-complete, we will reduce $\mathrm{TQBF} \leq_{\mathrm{P}} \mathrm{GG}$.

Let

$$\phi = (\exists x_1)(\forall x_2)(\exists x_3) \cdots (\exists x_k)\ \psi$$

be a QBF.

- We assume for simplicity that ϕ's quantifiers begin and ends with $\exists$, and that they strictly alternate between $\exists$ and $\forall$. Thus k is odd.
- We also assume that ψ is in CNF.
- If a formula doesn't meet these conditions, it can be modified.

To show GG is PSPACE-complete, we will reduce $\mathrm{TQBF} \leq_{\mathrm{P}} \mathrm{GG}$.

Let

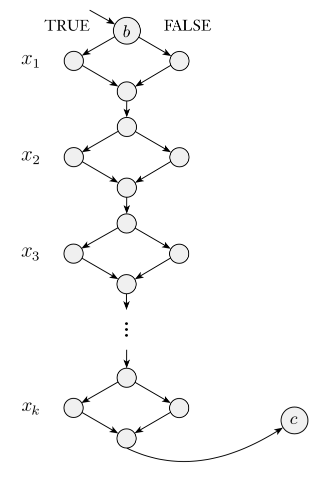
$$\phi = (\exists x_1)(\forall x_2)(\exists x_3) \cdots (\exists x_k)\ \psi$$

be a QBF.

- We assume for simplicity that ϕ's quantifiers begin and ends with $\exists$, and that they strictly alternate between $\exists$ and $\forall$. Thus k is odd.
- We also assume that ψ is in CNF.
- If a formula doesn't meet these conditions, it can be modified.

We need to convert ψ into an instance $\langle G, b\rangle$ of GG such that

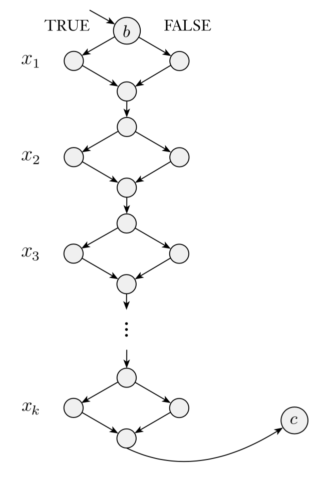
- If Player I wins the formula game for ϕ, then Player I wins the geography game $\langle G, b\rangle$.
- If Player II wins the formula game for ϕ, then Player II wins the geography game $\langle G, b\rangle$.



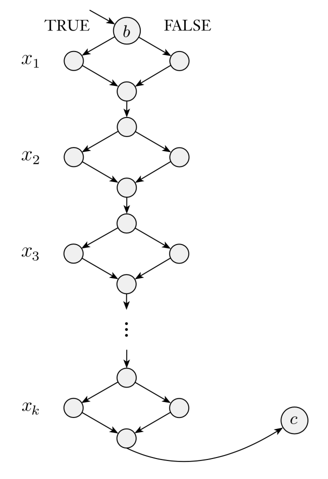
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.

Here is the first part of G:

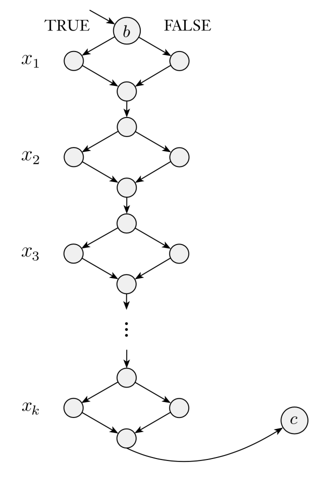


- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.



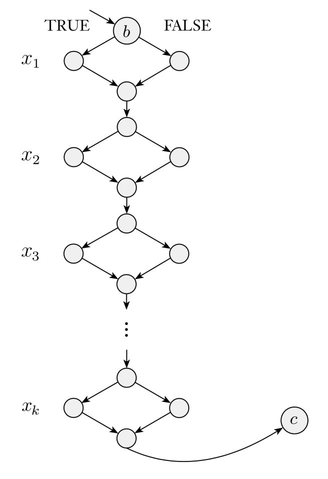
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.



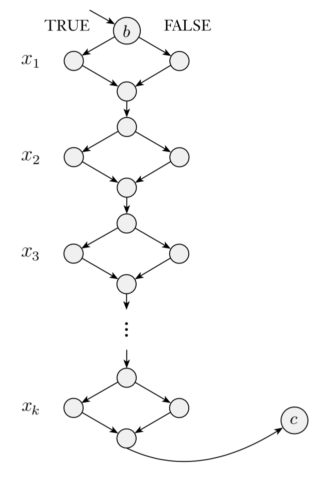
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.



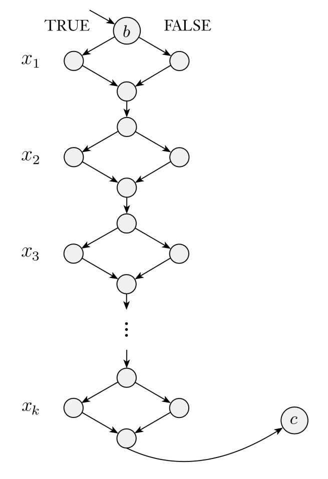
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.
- Player II chooses left or right in the 2^{nd} diamond, corresponding to $x_2 = T$ or $x_2 = F$.



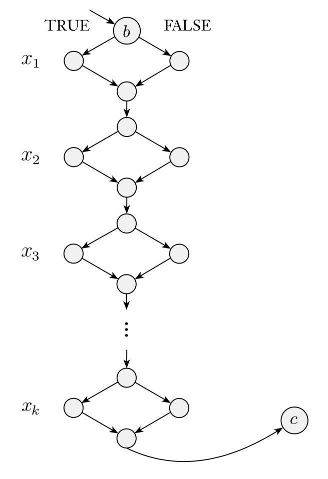
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.
- Player II chooses left or right in the 2^{nd} diamond, corresponding to $x_2 = T$ or $x_2 = F$.
- Player I chooses left or right in the 3^{rd} diamond.



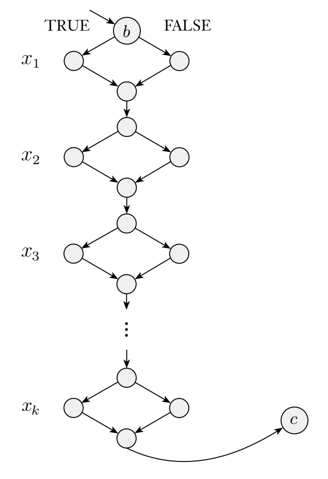
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.
- Player II chooses left or right in the 2^{nd} diamond, corresponding to $x_2 = T$ or $x_2 = F$.
- Player I chooses left or right in the 3^{rd} diamond.
- Player II chooses left or right in the 4^{th} diamond.



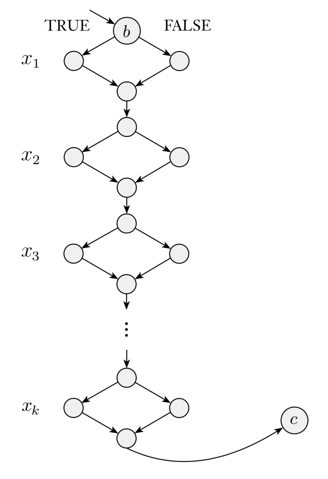
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.
- Player II chooses left or right in the 2^{nd} diamond, corresponding to $x_2 = T$ or $x_2 = F$.
- Player I chooses left or right in the 3^{rd} diamond.
- Player II chooses left or right in the 4^{th} diamond.
- $\ldots$



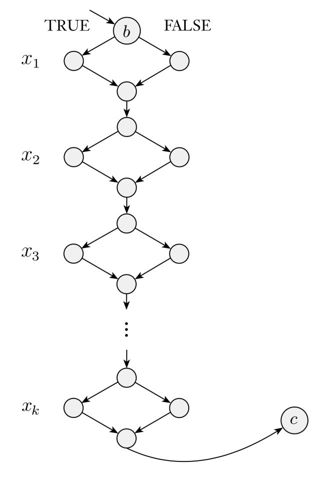
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.
- Player II chooses left or right in the 2^{nd} diamond, corresponding to $x_2 = T$ or $x_2 = F$.
- Player I chooses left or right in the 3^{rd} diamond.
- Player II chooses left or right in the 4^{th} diamond.
- $\ldots$
- Player I chooses left or right in the k^{th} diamond.



Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.
- Player II chooses left or right in the 2^{nd} diamond, corresponding to $x_2 = T$ or $x_2 = F$.
- Player I chooses left or right in the 3^{rd} diamond.
- Player II chooses left or right in the 4^{th} diamond.
- ...
- Player I chooses left or right in the k^{th} diamond.
- Player II chooses the bottom node in the k^{th} diamond.



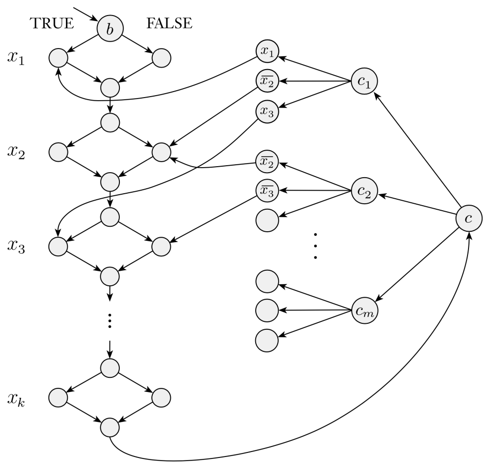
Here is the first part of G:

- Play starts at b. We have a diamond of vertices corresponding to each variable.
- Player I chooses one of the two nodes below b in the 1^{st} diamond. Left corresponds to setting $x_1 = T$ and right corresponds to setting $x_1 = F$.
- Player II has only one choice, the bottom node in the 1^{st} diamond.
- Then Player I also has only one choice, the top node in the 2^{nd} diamond.
- Player II chooses left or right in the 2^{nd} diamond, corresponding to $x_2 = T$ or $x_2 = F$.
- Player I chooses left or right in the 3^{rd} diamond.
- Player II chooses left or right in the 4^{th} diamond.
- ...
- Player I chooses left or right in the k^{th} diamond.
- Player II chooses the bottom node in the k^{th} diamond.
- Player I chooses c.

Suppose

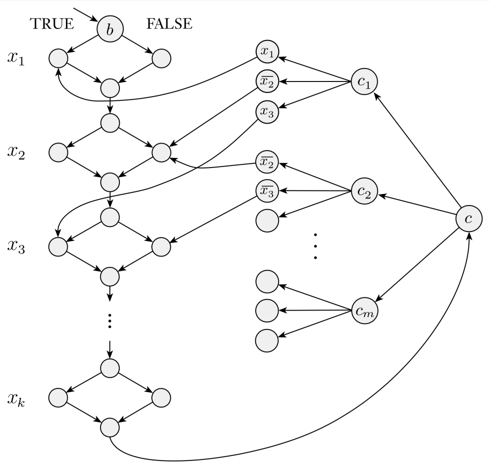
$$\phi = \exists x_1 \forall x_2 \cdots \exists x_k [(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \cdots) \wedge \cdots \wedge (\quad)]$$

Here is the construction for the rest of G:



- nodes $c_1, \ldots, c_m$ for each clause
- each c_i is connected to nodes named by its literals
- each literal node is connected to the diamonds:
  - x_i is connected to the left node in the i^{th} diamond
  - $\overline{x_i}$ is connected to the right node in the i^{th} diamond

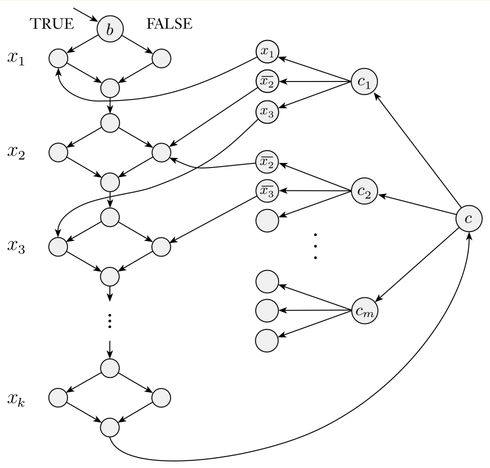
$$\phi = \exists x_1 \forall x_2 \cdots \exists x_k [(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \cdots) \wedge \cdots \wedge (\quad)]$$



Suppose $\phi \in \mathrm{TQBF}$.

- Playing from node c, Player II chooses one of $c_1, \ldots, c_m$.
- Then Player I picks a literal from that clause that is satisfied.
- Then Player II is stuck because that literal is only connected to one node: a left/right diamond node that was previously played. Thus Player I wins.

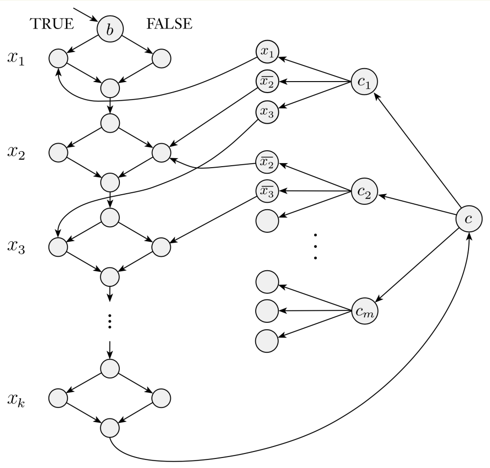
$$\phi = \exists x_1 \forall x_2 \cdots \exists x_k [(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \cdots) \wedge \cdots \wedge (\quad)]$$



Now suppose $\phi \notin \mathrm{TQBF}$.

- Playing from node c, Player II can win by choosing the unsatisfied clause.
- Any literal that Player I picks is false and connected to the side of a diamond that wasn't played.
- Player II plays that node in the diamond, leaving Player I stuck. Thus Player II wins.

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



Thus Player I wins the formula game if and only if Player I wins the geography game. This is a polynomial-time reduction, so $\mathrm{TQBF} \leq_{\mathrm{P}} \mathrm{GG}$.

Since TQBF is PSPACE-complete, GG is also PSPACE-complete. $\square$

PSPACE-Complete Problems

- TQBF
- Generalized Geography (GG)
- Many other generalized games
  - Tic-tac-toe
  - Othello
  - Chess
  - Go
- Problems about NFAs and regular expressions:
  - $ALL_{\mathrm{NFA}} = \{\langle N\rangle \mid N \text{ is an NFA with } L(N) = \Sigma^*\}$
  - $EQ_{\mathrm{NFA}} = \{\langle M, N\rangle \mid M, N \text{ are NFAs with } L(R) = L(S)\}$
  - $ALL_{\mathrm{REGEX}} = \{\langle R\rangle \mid R \text{ is a regular expression with } L(R) = \Sigma^*\}$
  - $EQ_{\mathrm{REGEX}} = \{\langle R, S\rangle \mid R, S \text{ are regular expressions with } L(R) = L(S)\}$