CSI - 3105 Design & Analysis of Algorithms
Course 21

Jean-Lou De Carufel

Fall 2019

Remember the following theorem (refer to Course 20).

Theorem

$$\left.\begin{array}{r} L \text{ is } NP\text{-Complete} \\ L \leq_P L' \\ L' \in NP \end{array}\right\} \implies L' \text{ is } NP\text{-Complete}$$

Remember the following theorem (refer to Course 20).

Theorem

$$\left.\begin{array}{l} L \text{ is } NP\text{-Complete} \\ L \leq_P L' \\ L' \in NP \end{array}\right\} \Longrightarrow L' \text{ is } NP\text{-Complete}$$

Now we can start using this theorem to prove that other problems are NP-Complete.

Remember the following theorem (refer to Course 20).

Theorem

$$\left.\begin{array}{r} L \text{ is } NP\text{-Complete} \\ L \leq_P L' \\ L' \in NP \end{array}\right\} \implies L' \text{ is } NP\text{-Complete}$$

Now we can start using this theorem to prove that other problems are NP-Complete.

At this moment, we know that $CIRCUIT - SAT$ is NP-Complete.

Remember the following theorem (refer to Course 20).

Theorem

$$\left.\begin{array}{r} L \text{ is } NP\text{-Complete} \\ L \leq_P L' \\ L' \in NP \end{array}\right\} \implies L' \text{ is } NP\text{-Complete}$$

Now we can start using this theorem to prove that other problems are NP-Complete.

At this moment, we know that $CIRCUIT - SAT$ is NP-Complete. We will prove that 3SAT is NP-Complete.

Remember the following theorem (refer to Course 20).

Theorem

$$\left.\begin{array}{r} L \text{ is } NP\text{-Complete} \\ L \leq_P L' \\ L' \in NP \end{array}\right\} \implies L' \text{ is } NP\text{-Complete}$$

Now we can start using this theorem to prove that other problems are NP-Complete.

At this moment, we know that $CIRCUIT - SAT$ is NP-Complete. We will prove that 3SAT is NP-Complete. It is sufficient to show that

1. 3SAT is in NP.
2. $CIRCUIT - SAT \leq_P 3SAT$

Remember the following theorem (refer to Course 20).

Theorem

$$\left.\begin{array}{l} L \text{ is } NP\text{-Complete} \\ L \leq_P L' \\ L' \in NP \end{array}\right\} \implies L' \text{ is } NP\text{-Complete}$$

Now we can start using this theorem to prove that other problems are *NP*-Complete.

At this moment, we know that $CIRCUIT - SAT$ is *NP*-Complete. We will prove that 3SAT is *NP*-Complete. It is sufficient to show that

1. 3SAT is in *NP*.
2. $CIRCUIT - SAT \leq_P 3SAT$

The first item is easy: for a given truth-assignment of the variables, we can verify in polynomial time if the Boolean formula is true.

It remains to show that $CIRCUIT - SAT \leq_P 3SAT$.

It remains to show that $CIRCUIT - SAT \leq_P 3SAT$. We need a function f such that

1. f transforms any input (a Boolean circuit) B for $CIRCUIT - SAT$ and produces an input $\phi = f(B)$ (a Boolean formula) for 3SAT.
2. There exist truth-values for the unknown input gates such that B's output is true

   $$\Longleftrightarrow$$

   There exist truth-values for the variables such that ϕ is true
3. $\phi = f(B)$ can be computed in time that is polynomial in the size of B.

Consider a Boolean circuit B.

- one variable for each gate
- describe the effect of each gate using a few clauses
- connect all clauses with $\bigwedge$'s

Consider a Boolean circuit B.

- one variable for each gate
- describe the effect of each gate using a few clauses
- connect all clauses with $\bigwedge$'s

Known input gates:

$x\ \boxed{\text{true}} \longleftrightarrow$ clause: x

$x\ \boxed{\text{false}} \longleftrightarrow$ clause: $\neg x$

Consider a Boolean circuit B.

- one variable for each gate
- describe the effect of each gate using a few clauses
- connect all clauses with $\bigwedge$'s

Known input gates:

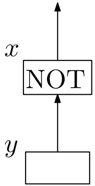
$$x\ \boxed{\text{true}} \longleftrightarrow \text{clause: } x$$

$$x\ \boxed{\text{false}} \longleftrightarrow \text{clause: } \neg x$$

Unknown input gates:

$$x\ \boxed{?} \longleftrightarrow \text{clause: } x \vee \neg x$$

NOT-gates:

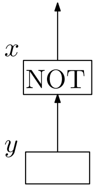


$\longleftrightarrow$

clauses: $(y \implies \neg x) \wedge (\neg y \implies x)$
same as

$$(\neg y \vee \neg x) \wedge (y \vee x)$$

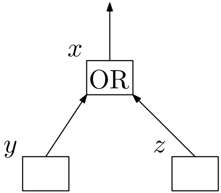
NOT-gates:



$\longleftrightarrow$

clauses: $(y \implies \neg x) \wedge (\neg y \implies x)$
same as
$$(\neg y \vee \neg x) \wedge (y \vee x)$$

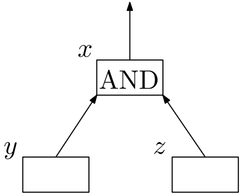
OR-gates:



$\longleftrightarrow$

clauses: $(x \implies (y \vee z)) \wedge ((y \vee z) \implies x)$
same as
$$(\neg x \vee y \vee z) \wedge (\neg y \vee x) \wedge (\neg z \vee x)$$

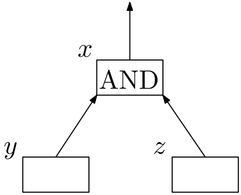
AND-gates:



clauses: $(x \Longrightarrow (y \wedge z)) \wedge ((y \wedge z) \Longrightarrow x)$
same as

$$(\neg x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z \vee x)$$

AND-gates:

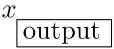


$\longleftrightarrow$

clauses: $(x \Longrightarrow (y \wedge z)) \wedge ((y \wedge z) \Longrightarrow x)$
same as

$$(\neg x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z \vee x)$$

Output-gates:



$\longleftrightarrow$

clause: x

Let ϕ be the conjunction of all these clauses.

By construction,

$$B \in Circuit - SAT \iff f(B) = \phi \in 3SAT$$

Let ϕ be the conjunction of all these clauses.

By construction,

$$B \in Circuit - SAT \iff f(B) = \phi \in 3SAT$$

Size of ϕ:

- number of variables in ϕ: number of gates in B.
- each clause has at most 3 literals
- number of clauses is at most 3 times the number of gates in B

Let ϕ be the conjunction of all these clauses.

By construction,

$$B \in Circuit - SAT \iff f(B) = \phi \in 3SAT$$

Size of ϕ:

- number of variables in ϕ: number of gates in B.
- each clause has at most 3 literals
- number of clauses is at most 3 times the number of gates in B

Therefore,

$$\text{size of } \phi = O(\text{size of } B)\text{: polynomial!}$$

Moreover, ϕ can be computed in time that is polynomial in the size of B.

Let ϕ be the conjunction of all these clauses.

By construction,

$$B \in Circuit - SAT \quad \Longleftrightarrow \quad f(B) = \phi \in 3SAT$$

Size of ϕ:

- number of variables in ϕ: number of gates in B.
- each clause has at most 3 literals
- number of clauses is at most 3 times the number of gates in B

Therefore,

$$\text{size of } \phi = O(\text{size of } B)\text{: polynomial!}$$

Moreover, ϕ can be computed in time that is polynomial in the size of B.

Conclusion: 3SAT is NP-Complete! $\square$

Putting it All Together

- 3SAT is NP-Complete
- $3SAT \leq_P INDEP-SET$ (refer to Course 19)
- $INDEP-SET \in NP$ (exercise)

Putting it All Together

- 3SAT is NP-Complete
- $3SAT \leq_P INDEP - SET$ (refer to Course 19)
- $INDEP - SET \in NP$ (exercise)

Therefore $INDEP - SET$ is NP-Complete!

Putting it All Together

- 3SAT is NP-Complete
- $3SAT \leq_P INDEP - SET$ (refer to Course 19)
- $INDEP - SET \in NP$ (exercise)

Therefore $INDEP - SET$ is NP-Complete!

- $CLIQUE \in NP$ (exercise)
- $INDEP - SET \leq_P CLIQUE$ (refer to Course 17)

Putting it All Together

- 3SAT is NP-Complete
- $3SAT \leq_P INDEP - SET$ (refer to Course 19)
- $INDEP - SET \in NP$ (exercise)

Therefore $INDEP - SET$ is NP-Complete!

- $CLIQUE \in NP$ (exercise)
- $INDEP - SET \leq_P CLIQUE$ (refer to Course 17)

Therefore $CLIQUE$ is NP-Complete!

Putting it All Together

- 3SAT is NP-Complete
- $3SAT \leq_P INDEP - SET$ (refer to Course 19)
- $INDEP - SET \in NP$ (exercise)

Therefore $INDEP - SET$ is NP-Complete!

- $CLIQUE \in NP$ (exercise)
- $INDEP - SET \leq_P CLIQUE$ (refer to Course 17)

Therefore $CLIQUE$ is NP-Complete!

- $VERTEX - COVER \in NP$ (exercise)
- $CLIQUE \leq_P VERTEX - COVER$ (refer to Course 18)

Putting it All Together

- 3SAT is NP-Complete
- $3SAT \leq_P INDEP - SET$ (refer to Course 19)
- $INDEP - SET \in NP$ (exercise)

Therefore $INDEP - SET$ is NP-Complete!

- $CLIQUE \in NP$ (exercise)
- $INDEP - SET \leq_P CLIQUE$ (refer to Course 17)

Therefore $CLIQUE$ is NP-Complete!

- $VERTEX - COVER \in NP$ (exercise)
- $CLIQUE \leq_P VERTEX - COVER$ (refer to Course 18)

Therefore $VERTEX - COVER$ is NP-Complete!

Exercise

How do we prove that $SUBSET - SUM \leq_P VERTEX - COVER$?

Exercise

How do we prove that $SUBSET - SUM \leq_P VERTEX - COVER$?

We could find a function f which satisfies the famous 3 properties...

Exercise

(Exercise 20) Is there a problem in *NP* that is not *NP*-Complete?