CSC363 Tutorial #9

NP completeness, Boolean Satisfiability Problem

March 22, 2023

Things covered in this tutorial

- ⋆ What is NP-completeness?
- ⋆ What's the Boolean Satisfiability Problem (SAT)?
- ⋆ Why is SAT NP-complete?

NP-completeness

These problems are NP-complete! They are the *hardest NP-problems*.

- ⋆ Subset Sum.
- ⋆ Boolean Satisfiability Problem.
- ⋆ Graph Isomorphism Problem.
- ⋆ Vertex Cover Problem.
- ⋆ Knapsack Problem.
- ⋆ Hamiltonian Path Problem.
- ⋆ Generalized Sudoku.
- ⋆ ...

Showing that any of these problems is $\notin \mathrm{P}$ will net you $1 million USD.

NP-completeness

These problems are NP-complete! They are the *hardest NP-problems*.

- ⋆ Subset Sum.
- ⋆ Boolean Satisfiability Problem.
- ⋆ Graph Isomorphism Problem.
- ⋆ Vertex Cover Problem.
- ⋆ Knapsack Problem.
- ⋆ Hamiltonian Path Problem.
- ⋆ Generalized Sudoku.
- ⋆ ...

Showing that any of these problems is $\notin \mathrm{P}$ will net you $1 million USD.

NP-completeness

A langague L is NP-complete if both of the following are satisfied:

1.
2.

Task: Fill out the above!

NP-completeness

A langague L is NP-complete if both of the following are satisfied:

1. $L \in \mathrm{NP}$.
2. For any language $M \in \mathrm{NP}$, we have $M \leq_p L$.

Task: Fill out the above!

NP-complete problems are the *hardest NP problems*.

Boolean Formulae

Question: What's a *boolean formula*?

Boolean Formulae

Question: What's a *boolean formula*?

Ans: A *boolean formula* is a "well-formed" logical expression consisting of symbols from

$$\{(,), \neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$$

and one or more "variables". A boolean formula's truth value can be evaluated once all variables are assigned to T or F.

Boolean Formulae

Question: What's a *boolean formula*?

Ans: A *boolean formula* is a "well-formed" logical expression consisting of symbols from

$$\{(,), \neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$$

and one or more "variables". A boolean formula's truth value can be evaluated once all variables are assigned to T or F.

The following are boolean formulas:

- ⋆ $(x_1 \vee x_2) \rightarrow x_3$ (with x_1, x_2, x_3 being the variables).
- ⋆ $(x \wedge \neg y) \wedge x$ (with x, y, z being the variables).
- ⋆ $x \wedge x \wedge x \wedge \neg x$ (with x being the only variable).

Boolean Formulae

Question: What's a *boolean formula*?
Ans: A *boolean formula* is a "well-formed" logical expression consisting of symbols from

$$\{(,), \neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$$

and one or more "variables". A boolean formula's truth value can be evaluated once all variables are assigned to T or F.

The following are boolean formulas:

- ⋆ $(x_1 \vee x_2) \rightarrow x_3$ (with x_1, x_2, x_3 being the variables).
- ⋆ $(x \wedge \neg y) \wedge x$ (with x, y, z being the variables).
- ⋆ $x \wedge x \wedge x \wedge \neg x$ (with x being the only variable).

The following are not boolean formulas:

- ⋆ $x_1 x_2^2 = x_3$.
- ⋆ $()()()x_1 \rightarrow \neg$

Boolean Formulae

Task: Evaluate the boolean formula

$$(x_1 \vee x_2 \vee x_3) \rightarrow ((x_2 \wedge x_3) \rightarrow \neg x_1)$$

with the assignments $x_1 = T, x_2 = F, x_3 = F$

Boolean Formulae

Task: Evaluate the boolean formula

$$(x_1 \vee x_2 \vee x_3) \to ((x_2 \wedge x_3) \to \neg x_1)$$

with the assignments $x_1 = T, x_2 = F, x_3 = F$

Ans: We have

$$(T \vee F \vee F) \to ((F \wedge F) \to \neg T).$$

Since $(T \vee F \vee F)$ is true, the above reduces to

$$T \to ((F \wedge F) \to \neg T).$$

Since $T \to X$ has the same truth value as X, we get

$$(F \wedge F) \to \neg T.$$

Boolean Formulae

$$(F \wedge F) \rightarrow \neg T.$$

Continuing, $(F \wedge F)$ is false, so we reduce to

$$F \rightarrow \neg T$$

and since $F \rightarrow$ something is always true, we reduce to

$$T.$$

Thus, we conclude that

$$(x_1 \vee x_2 \vee x_3) \rightarrow ((x_2 \wedge x_3) \rightarrow \neg x_1)$$

with the assignments $x_1 = T, x_2 = F, x_3 = F$ evaluates to true.

SAT

Task: Come up with a boolean formula such that *no assignment of its variables* makes the formula evaluate to true.

SAT

Task: Come up with a boolean formula such that *no assignment of its variables* makes the formula evaluate to true.

Answer: Something like

$$x \wedge \neg x$$

should work.

SAT

Task: Come up with a boolean formula such that *no assignment of its variables* makes the formula evaluate to true.

Answer: Something like

$$x \wedge \neg x$$

should work.

Definition: A boolean formula ϕ is **satisfiable** if *some* assignment of its variables makes ϕ evaluate to true. We let SAT be the set of all satisfiable boolean formulas:

$$\mathrm{SAT} = \{\phi : \phi \text{ is a satisfiable boolean formula.}\}$$

SAT

Task: Determine which of the following formulas are satisfiable.

- $(x_1 \vee x_2) \rightarrow x_3$.
- $(x \wedge \neg y) \wedge x$.
- $x \wedge x \wedge x \wedge \neg x$.
- $$\begin{aligned} &(T_{0,0,0} \wedge T_{1,0,0}) \\ &\wedge (Q_{0,0}) \wedge (H_{0,0}) \\ &\wedge (\neg(T_{0,0,0}) \vee \neg(T_{0,1,0})) \wedge (\neg(T_{1,0,0}) \vee \neg(T_{1,1,0})) \\ &\wedge (\neg(T_{0,0,1}) \vee \neg(T_{0,1,1})) \wedge (\neg(T_{1,0,1}) \vee \neg(T_{1,1,1})) \\ &\wedge (T_{0,0,0} \wedge T_{0,1,1} \rightarrow H_{0,0}) \wedge \wedge (T_{0,1,0} \wedge T_{0,0,1} \rightarrow H_{0,0}) \\ &\wedge (\neg Q_{0,0} \vee \neg Q_{1,0}) \wedge (\neg Q_{0,1} \vee \neg Q_{1,1}) \\ &\wedge (\neg H_{0,0} \vee \neg H_{1,0}) \wedge (\neg H_{0,1} \vee \neg H_{1,1}) \\ &\wedge ((H_{0,0} \wedge Q_{0,0} \wedge T_{0,0,0}) \rightarrow (H_{1,1} \wedge Q_{1,1} \wedge T_{0,1,1})) \\ &\wedge Q_{1,0} \vee Q_{1,1} \end{aligned}$$

SAD SAT

Question: How do we know that a boolean formula is not satisfiable?

SAD SAT

Question: How do we know that a boolean formula is not satisfiable?
Answer: Check through all combinations of variable assignments... 💀

SAD SAT

Question: How do we know that a boolean formula is not satisfiable?
Answer: Check through all combinations of variable assignments... 💀

Question: How long does it take to check through all combinations?

SAD SAT

Question: How do we know that a boolean formula is not satisfiable?
Answer: Check through all combinations of variable assignments... 💀

Question: How long does it take to check through all combinations?
Answer: $O(2^n)$... 💀

SAD SAT

SAT is another easy-to-verify, hard-to-solve[1] problem.

[1]We haven't proven that it's hard-to-solve though, since we haven't yet proven $\mathrm{SAT} \notin P$.

SAT is NP-complete

$$\text{SAT} = \{\phi : \phi \text{ is a satisfiable boolean formula.}\}$$

Very hard problem! SAT is NP-complete.

Question: which two statements do we need to show, in order to prove SAT is NP-complete?

SAT is NP-complete

$$\mathrm{SAT} = \{\phi : \phi \text{ is a satisfiable boolean formula.}\}$$

Very hard problem! SAT is NP-complete.

Question: which two statements do we need to show, in order to prove SAT is NP-complete?

Answer:

1. $\mathrm{SAT} \in \mathrm{NP}$.
2. For any language $L \in \mathrm{NP}$, we have $L \leq_p \mathrm{SAT}$.

SAT is NP-complete

$$\text{SAT} = \{\phi : \phi \text{ is a satisfiable boolean formula.}\}$$

Question: How do we show a language L is in NP?

SAT is NP-complete

$$\text{SAT} = \{\phi : \phi \text{ is a satisfiable boolean formula.}\}$$

Question: How do we show a language L is in NP?
Answer: Build a poly-time NTM that decides L!

SAT is NP-complete

$$\text{SAT} = \{\phi : \phi \text{ is a satisfiable boolean formula.}\}$$

Question: How do we show a language *L* is in NP?
Answer: Build a poly-time NTM that decides *L*!

Task: Show that $\text{SAT} \in \text{NP}$.

```
SAT(phi):
  if phi is not a boolean formula:
    reject
  let x1, ..., xn be the variables of phi
```

SAT is NP-complete

Task: Show that $\mathrm{SAT} \in \mathrm{NP}$.

```
SAT(phi):
  if phi is not a boolean formula:
    reject
  let x1, ..., xn be the variables of phi
  for i in 1..n:
    nondeterminstically assign xi to T/F
  if our assignment of x1..xn satsifies phi:
    accept
  else:
    reject
```

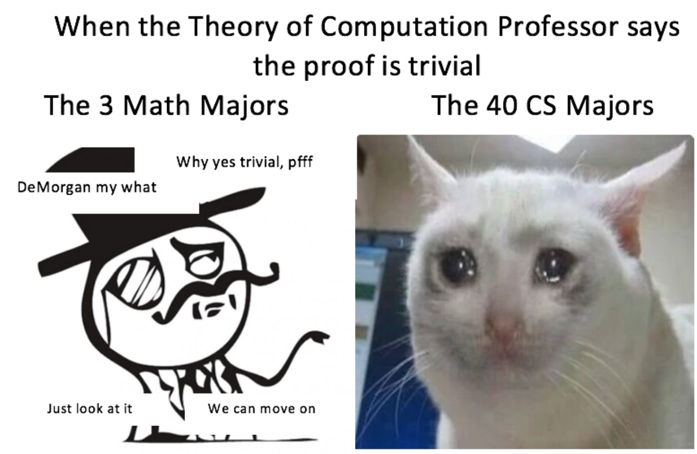
SAT is NP-complete

Task: Let $L \in \mathrm{NP}$. Show that $L \leq_p \mathrm{SAT}$.

SAT is NP-complete

Task: Let $L \in \mathrm{NP}$. Show that $L \leq_p \mathrm{SAT}$.

Answer:



Worksheet time!

How to show a language L is NP-complete

1. Show $L \in \mathrm{NP}$.
2. Show that for any language $M \in \mathrm{NP}$, we have $M \leq_p L$.

2. is hard! Usually, we can just do the following instead:

1. Show $L \in \mathrm{NP}$.
2. Find a known NP-complete language M (such as SAT), and show that $\leq_p L$.