5CCS2FC2: Foundations of Computing II

P vs NP: The Million Dollar Problem

Week 2

Dr Christopher Hampson

Department of Informatics

King's College London

The Boolean Satisfiability Problem

The Boolean Satisfiability Problem SAT

Input) A propositional formula F

Output) **True** if and only if F is *satisfiable*

The Boolean Satisfiability Problem

- **Truth Tables and Satisfability**

P	$\neg P$
True	False
False	True

P	Q	$P \wedge Q$
True	True	True
True	False	False
False	True	False
False	False	False

P	Q	$P \vee Q$
True	True	True
True	False	True
False	True	True
False	False	False

P	Q	$P \rightarrow Q$
True	True	True
True	False	False
False	True	True
False	False	True

The Boolean Satisfiability Problem

P	Q	R	$(P \vee \neg R) \rightarrow \neg(\neg Q \vee R)$
True	True	True	False
True	True	False	True
True	False	True	False
True	False	False	False
False	True	True	True
False	True	False	True
False	False	True	True
False	False	False	False

The Boolean Satisfiability Problem

P	Q	R	$\neg(P \rightarrow Q) \wedge \neg(P \vee \neg R)$
True	True	True	False
True	True	False	False
True	False	True	False
True	False	False	False
False	True	True	False
False	True	False	False
False	False	True	False
False	False	False	False

Theorem Every formula is equivalent to a formula in DNF.

Proof:

Step 1) Rewrite any **implications** which do not appear in DNF.

$$(F \to G) \quad \rightsquigarrow \quad (\neg F \vee G)$$

Theorem Every formula is equivalent to a formula in DNF.

Proof:

Step 2) Use **De Morgan's Laws** to move the **negations** inside any brackets until they appear directly next to propositional variables,

$$\neg (F \wedge G) \quad \rightsquigarrow \quad (\neg F \vee \neg G)$$

$$\neg (F \vee G) \quad \rightsquigarrow \quad (\neg F \wedge \neg G)$$

Theorem Every formula is equivalent to a formula in DNF.

Proof:

Step 3) Eliminate any **double negations** with the **Cancellation Law**

$$\neg\neg F \rightsquigarrow F$$

Theorem Every formula is equivalent to a formula in DNF.

Proof:

Step 4) Apply the **Distributivity Laws** to bring any **disjunctions** to the 'surface'

$$\begin{array}{lcl} F \wedge (G \vee H) & \rightsquigarrow & (F \wedge G) \vee (F \wedge H) \\ (G \vee H) \wedge F & \rightsquigarrow & (G \wedge F) \vee (H \wedge F) \end{array}$$

The Boolean Satisfiability Problem

	P	Q	R	$(P \vee \neg R) \rightarrow \neg(\neg Q \vee R)$
1.	True	True	True	False
2.	True	True	False	True
3.	True	False	True	False
4.	True	False	False	False
5.	False	True	True	True
6.	False	True	False	True
7.	False	False	True	True
8.	False	False	False	False

(Row 2) $\vee$ (Row 5) $\vee$ (Row 6) $\vee$ (Row 7)

Theorem Every formula is equivalent to a formula in CNF.

Proof:

Step 1) Rewrite any **implications** which do not appear in DNF.

$$(F \to G) \quad \rightsquigarrow \quad (\neg F \vee G)$$

Theorem Every formula is equivalent to a formula in CNF.

Proof:

Step 2) Use **De Morgan's Laws** to move the **negations** inside any brackets until they appear directly next to propositional variables,

$$\neg (F \wedge G) \quad \rightsquigarrow \quad (\neg F \vee \neg G)$$

$$\neg (F \vee G) \quad \rightsquigarrow \quad (\neg F \wedge \neg G)$$

Theorem Every formula is equivalent to a formula in CNF.

Proof:

Step 3) Eliminate any **double negations** with the **Cancellation Law**

$$\neg\neg F \rightsquigarrow F$$

Theorem Every formula is equivalent to a formula in CNF.

Proof:

Step 4) Apply the **Distributivity Laws** to bring any **conjunctions** to the 'surface'

$$\begin{array}{lcl} F \vee (G \wedge H) & \rightsquigarrow & (F \vee G) \wedge (F \vee H) \\ (G \wedge H) \vee F & \rightsquigarrow & (G \vee F) \wedge (H \vee F) \end{array}$$

The Boolean Satisfiability Problem

	P	Q	R	$(P \vee \neg R) \rightarrow \neg(\neg Q \vee R)$
1.	True	True	True	False
2.	True	True	False	True
3.	True	False	True	False
4.	True	False	False	False
5.	False	True	True	True
6.	False	True	False	True
7.	False	False	True	True
8.	False	False	False	False

(Avoid 1) $\wedge$ (Avoid 3) $\wedge$ (Avoid 4) $\wedge$ (Avoid 8)

The Boolean Satisfiability Problem

The Boolean Satisfiability Problem SAT

Input) A propositional formula F in CNF

Output) **True** if and only if F is *satisfiable*

Proving NP-membership

(Upper Bound)

The Boolean Satisfiability Problem

Theorem The Boolean Satisfiability Problem **SAT** belongs to **NP**.

Proof: **Input)** Given formula F in Conjunctive Normal Form,

Step 1) We can decide whether F is **satisfiable** by computing its **truth table**.

However the truth table contains 2^n rows — **NOT** polynomial!

The Boolean Satisfiability Problem

Theorem The Boolean Satisfiability Problem **SAT** belongs to **NP**.

Step 2) However, a non-deterministic machine can evaluate each row in **parallel**,

Each row requires us to evaluate at most n boolean operators — **polynomial!**

Q.E.D.

Proving NP-hardness

(Lower Bound)

Theorem (The Cook-Levin Theorem) The Boolean Satisfiability problem **SAT** is **NP**-complete.

Proof:

Next time...

Q.E.D. (soon)

End of Slides!