COMP4141 Theory of Computation

Lecture 13 NP-Complete Problems

Ron van der Meyden

CSE, UNSW

Revision: 2015/04/23

(Credits: D Dill, K Engelhardt, M Sipser, W Thomas, T Wilke, R van Glabbeek, P Hofner)

Cook's Theorem (SAT is NP-Complete)

Cook's theorem gives a "generic reduction" for every problem in **NP** to *SAT*. So SAT is as hard as any other problem in **NP**—it's **NP**-complete.

So, *SAT* is the granddaddy of all **NP**-complete problems.

Many people have worked on the *SAT* problem, and there are now solvers (SAT solvers) for it that can solve problems up to thousands of variables in practice (though not polynomial time in theory).

People frequently translate **NP**-complete problems to propositional logic, and then attack them with these general solvers!

CSAT

CSAT is a special case of *SAT*.

$$CSAT = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable cnf formula } \}$$

where a Boolean formula is in *cnf* (for *conjunctive normal form*) if it is (also) generated by the grammar

$$\phi \to (c) \mid (c) \land \phi \qquad c \to \ell \mid \ell \lor c$$
$$\ell \to p \mid \neg p \qquad p \to x \mid y \mid \ldots$$

We call cs *clauses*, ℓs *literals*, and ps *propositions*.

Example

$(x \lor z) \land (y \lor z)$ is a cnf for the Boolean formula $(x \land y) \lor z$.

CSAT is NP-Complete

Clearly *CSAT* is in **NP** because we can use the same certificate for ϕ in cnf as we would for the same ϕ in *SAT*.

Giving a **P** reduction from *SAT* to *CSAT* is tricky.

A straight-forward translation of Boolean formulae into equivalent cnf may result in an exponential blow-up, meaning that this approach is useless.

Instead, we recall a reduction f won't have to preserve *satisfaction*:

$$\forall\pi\,(\pi \models \phi \quad \Leftrightarrow \quad \pi \models f(\phi))$$

but merely *satisfiability*

$$\exists\pi\,(\pi \models \phi) \quad \Leftrightarrow \quad \exists\pi\,(\pi \models f(\phi))$$

meaning that we're free to choose different πs for the two sides.

CSAT is NP-Hard

The translation from Boolean formulae to cnf proceeds in two steps which are both in **P**.

1. Translate to *nnf* (*negation normal form*) by pushing all negation symbols down to propositions. (This is still satisfaction-preserving.)
2. Translate from nnf to cnf. (This merely preserves satisfiability.)

nnf formulas are those that have all negations applied only to atomic propositions.

Pushing Down $\neg$

We use de Morgan's laws and the law of double negation to rewrite left-hand-sides to right-hand-sides:

$$
\begin{aligned}
\neg(\phi \wedge \psi) &\Leftrightarrow \neg(\phi) \vee \neg(\psi) \\
\neg(\phi \vee \psi) &\Leftrightarrow \neg(\phi) \wedge \neg(\psi) \\
\neg(\neg(\phi)) &\Leftrightarrow \phi
\end{aligned}
$$

Example

$$
\begin{aligned}
\neg((\neg(x \vee y)) \wedge (\neg x \vee y)) &\Leftrightarrow \neg(\neg(x \vee y)) \vee \neg(\neg x \vee y) \\
&\Leftrightarrow x \vee y \vee \neg(\neg x \vee y) \\
&\Leftrightarrow x \vee y \vee \neg(\neg x) \wedge \neg y \\
&\Leftrightarrow x \vee y \vee x \wedge \neg y
\end{aligned}
$$

Pushing Down $\neg$ cont.

Theorem

Every Boolean formula ϕ is equivalent to a Boolean formula ψ in nnf. Moreover, $|\psi|$ is linear in $|\phi|$ and ψ can be constructed from ϕ in **P**.

Proof.

by induction on the number n of Boolean operators ($\wedge$, $\vee$, $\neg$) in ϕ we may show that there is an equivalent ψ in nnf with at most $2n - 1$ operators. $\square$

nnf $\longrightarrow$ cnf

Theorem

There is a constant c such that every nnf ϕ has a cnf ψ such that:

1. ψ *consists of at most* $|\phi|$ *clauses.*
2. ψ *is constructable from* ϕ *in time at most* $c|\phi|^2$*.*
3. $\pi \models \phi$ *iff there exists an extension* π' *of* π *satisfying* $\pi' \models \psi$*, for all interpretations* π *of the propositions in* ϕ*.*

Proof.

by induction on $|\phi|$. $\square$

nnf $\longrightarrow$ cnf cont.

Example

Consider

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

An equisatisfiable cnf is

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

General trick: $A \vee B$ is satisfiable iff $(A \vee p) \wedge (B \vee \neg p)$ is satisfiable, where p is a new atomic proposition.

3SAT

3SAT is a special case of *CSAT*.

$$\mathit{3SAT} = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable 3cnf formula } \}$$

where a Boolean formula is in *3cnf* (for *3 literal conjunctive normal form*) if it is (also) generated by the grammar

$$\phi \to (c) \mid (c) \wedge \phi \qquad\qquad c \to \ell \vee \ell \vee \ell$$
$$\ell \to p \mid \neg p \qquad\qquad p \to x \mid y \mid \ldots$$

Example

$(x \vee y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (x \vee \neg y \vee \neg z)$ is a 3cnf for the Boolean formula x.

3SAT is NP-Complete

Proof.

Clearly *3SAT* is in **NP** because we can use the same certificate for ϕ in 3cnf as we would for the same ϕ in *SAT* (or *CSAT*).

Sipser prefers to adapt his **NP**-hardness proof for *SAT* to *3SAT* over giving a **P** reduction from *SAT* to *3SAT*.

We **P** reduce from *CSAT* to *3SAT* instead, by translating arbitrary clauses into clauses with exactly three literals. □

Proof detail: how to transform a cnf $\phi = \bigwedge_{i=1}^{n} c_i$ into an equisatisfiable 3cnf. We transform each clause $c_i = \bigvee_{j=1}^{k_i} \ell_{i,j}$ depending on the number k_i of literals in it. (We omit subscript i.)

Case $k = 1$ (ℓ_1) is replaced by

$$(\ell_1 \vee u \vee v) \wedge (\ell_1 \vee u \vee \neg v) \wedge (\ell_1 \vee \neg u \vee v) \wedge (\ell_1 \vee \neg u \vee \neg v)$$

for some fresh propositions u, v.

Case $k = 2$ $(\ell_1 \vee \ell_2)$ is replaced by

$$(\ell_1 \vee \ell_2 \vee u) \wedge (\ell_1 \vee \ell_2 \vee \neg u)$$

for some fresh proposition u.

Case $k = 3$ is 3cnf already.

Case $k > 3$ $(\bigvee_{j=1}^{k} \ell_j)$ is replaced by

$$(\ell_1 \vee \ell_2 \vee u_1) \wedge \bigwedge_{j=1}^{k-4} (\ell_{j+2} \vee \neg u_j \vee u_{j+1}) \wedge (\neg u_{k-3} \vee \ell_{k-1} \vee \ell_k)$$

for some $k - 3$ fresh propositions $u_1, \ldots, u_{k-3}$.

For the correctness argument, note that to satisfy the formula in the case for $k > 3$ using only the u_i, we need the following formula to be satisfied:

$$u_1 \wedge (\neg u_1 \vee u_2) \wedge \ldots \wedge (\neg u_{k-4} \vee u_{k-3}) \wedge \neg u_{k-3}$$

or equivalently,

$$u_1 \wedge (u_1 \Rightarrow u_2) \wedge \ldots \wedge (u_{k-4} \Rightarrow u_{k-3}) \wedge \neg u_{k-3}$$

But this is easily seen to be *unsatisfiable*!

On the other hand if we drop any one of the conjuncts, it is satisfiable (all true to the left, all false to the right of dropped position).

CLIQUE is NP-Complete

A k-clique in an undirected graph is a set of k nodes such that there is an edge between each pair.

Let

$$CLIQUE = \left\{ \langle G, k \rangle \;\middle|\; \begin{array}{l} G \text{ is undirected graph} \\ \text{that has a } k\text{-clique} \end{array} \right\}$$

We show **NP**-completeness on the whiteboard.

HAMPATH is NP-Complete

A *Hamiltonian path* from node s to node t in a (directed) graph is a path starting at s and finishing at t that visits every node *exactly* once.

$$HAMPATH = \left\{ \langle G, s, t \rangle \;\middle|\; \begin{array}{l} \text{Directed graph } G \text{ has a} \\ \text{Hamiltonian path from } s \text{ to } t \end{array} \right\}$$

HAMPATH is in **NP**. We show **NP**-completeness by proving $3SAT \leq_{\mathbf{P}} HAMPATH$ on the whiteboard.

—The End—