The Cook-Levin Theorem

Mark Greenstreet, CpSc 421, Term 1, 2008/09

- The Cook-Levin Theorem
  - Define NP Completeness
  - Satisfiability is NP complete
- NP complete problems

NP Completeness

- Let A and B be algorithms. If A is reducible to B in deterministic polynomial time, we write $A \leq_P B$.
- If a problem language is decidable in non-deterministic polynomial time, we say that it is in NP.
- If B is a language such that for any language $A \in NP$, $A \leq_P B$ we say that B is NP-hard.
- If B is a language such that $B \in NP$ and B is NP hard, then we say that B is NP complete.
- NP complete problems are interesting because:
  - They are, intuitively, the hardest problems in NP.
  - Many commonly occuring intractable problems are NP complete.
  - If we could find an efficient (i.e. polynomial time) algorithm for one of them, we would have a polynomial time algorithm for all of them.

Satisfiability

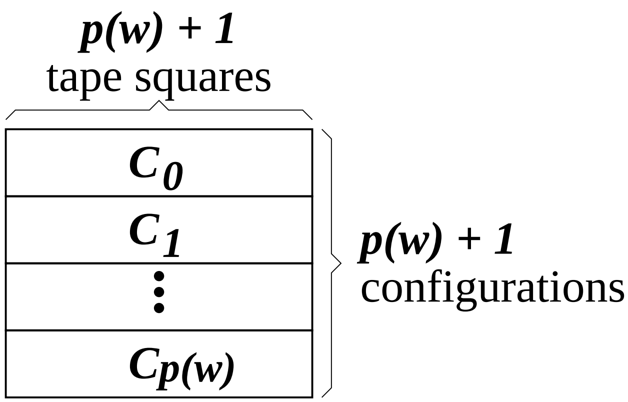
- A boolean formula is:
  - Variables: x, y, p_3, $\ldots$ are boolean formulas.
  - Conjunction: $\phi_1 \wedge \phi_2$, where ϕ_1 and ϕ_2 are boolean formulas.
  - Disjunction: $\phi_1 \vee \phi_2$, where ϕ_1 and ϕ_2 are boolean formulas.
  - Negation: $\neg\phi$, where ϕ is a boolean formula.
  - Parentheses: (ϕ), where ϕ is a boolean formula.
- Satisfiability
  - Let ϕ be a boolean formula and let x_1, x_2, $\ldots$, x_k be the variables that appear in ϕ.
  - We say that Φ is satisfiable iff $\exists x_1, x_2, \ldots, x_k.\ \phi$.
  - Formulas can be represented by strings. Thus, we can talk about a language of satisifiable formulas:

$$SAT \quad = \quad \{\langle\phi\rangle \mid \phi \text{ is satisfiable}\}$$

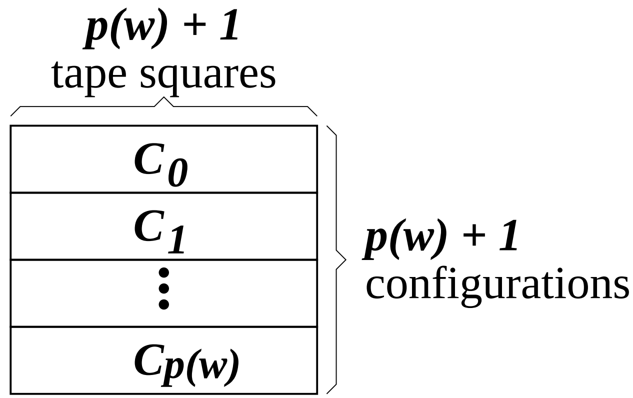
The Cook-Levin Theorem

SAT is NP complete.

- Proof: By computational histories.
  - Let A be any language in NP.
    - There is an NTM, N_A that decides A in polynomial time.
    - Let p be a polynomial such that for any string w, N_A decides w after at most $p(|w|)$ steps.
    - Thus, N_A has a computational history of at most $p(|w|) + 1$ configurations when deciding w.
    - N_A can visit at most $p(w) + 1$ tape squares in these $p(|w|)$ steps.



Checking Successive Configurations



- Encode tape symbols and states with binary strings.
- If a tape square does not have the tape-head marker and is not next to the tape head marker
  - make sure that the symbol is unchanged.
  - This can be written as a boolean formula.
- For the three squares centered on the tape head
  - make sure that the successor has the correct next state.
  - This also can be written as a boolean formula.

The reduction is polynomial time.

- How big is the formula?
  - Each tape square requires $O(1)$ terms.
  - There are $p(|w|) + 1$ squares per configuration.
  - There are $p(|w|)$ successive configurations.
  - Plus a few more to check that the intial configuration is correct and that the final configuration is accepting, but these are small (i.e. $O(|w|)$).
- The formula:

$$
\begin{array}{ll}
& InitialConfigurationCorrect \\
\wedge & \displaystyle\bigwedge_{i=0}^{p(|w|)-1} \bigwedge_{j=1}^{p(|w|)} \\
& \qquad (c_{i,j-1} \notin Q) \wedge (c_{i,j} \notin Q) \wedge (c_{i,j+1} \notin Q) \wedge (c_{i+1,j} = c_{i,j}) \\
& \quad \vee \quad (c_{i,j-1} \in Q) \vee (c_{i,j+1} \in Q) \\
& \quad \vee \quad (c_{i,j} \in Q) \wedge c_{i+1,j-1}c_{i+1,j}c_{i+1,j+1} = \Delta(c_{i,j-1}c_{i,j}c_{i,j+1})
\end{array}
$$

  - $c_{i,j}$ denotes the symbol in the j^{th} square of the i^{th} configuration.
  - I've ignored a few special cases at the ends of the tape.

SAT is NP complete

- SAT is in NP:
  - Just guess a satisfying assignment and check it.
- SAT is in NP hard: shown above by the reduction from any problem in NP to SAT.
  - The formula can be generated in time proportional to its length.
  - This is a polynomial time reduction from any problem in NP to SAT.
- $\therefore SAT$ is NP-complete.

$\mathit{3SAT}$

- 3-CNF
  - A “literal” is a variable, v, or its negation, $\overline{v}$.
  - A “clause” is a disjunction (“or”) of variables e.g. $v_3 \vee \overline{v_5} \vee \overline{v_8} \vee v_{13}$.
  - A formula is in conjunctive normal form (CNF) if it is the can be written as a conjunction (“and”) of clauses, e.g.
    $(v_3 \vee \overline{v_5} \vee \overline{v_8} \vee v_{13}) \wedge (v_1 \vee v_2 \vee v_3) \wedge (\overline{v_{13}} \vee v_{27})$
  - A formula is in 3-CNF if it is a CNF formula where each clause consists of three variables.
- $\mathit{3SAT}$ is a restricted version of SAT where the formula is in 3-CNF.
- By restricting the class of formulas, it can be easier to show a reduction from 3-CNF to some other problem, X, than it is to reduce arbitrary boolean formulas.

$3SAT$ is NP-complete

- I'll show that any boolean formula can be transformed into an equivalent 3-CNF formula.
  - We can push all negations down to the literals (just use DeMorgan's laws). This at most doubles the length of the formula.
  - If our top-level formula is of the form $\Phi_1 \vee \Phi_2$ where neither Φ_1 nor Phi_2 are literals:
    - We rewrite it as $(\Phi_1 \vee x)(\Phi_2 \vee \overline{x})$
    - x is a "new" variable (doesn't appear in Φ_1 or Φ_2).
    - x "selects" which of Φ_1 or Φ_2 must be true.
    - For example, $(a \wedge b) \vee (c \wedge d)$ is satisfiable iff $(a \wedge b \vee x) \vee (c \wedge d \vee \overline{x})$ is satisfiable.
- This shows that $SAT \leq_P 3SAT$.
  - We know that SAT is NP complete.
  - Therefore, $3SAT$ is NP hard.
- Obviously, $3SAT \leq_P SAT$.
  - Therefore, $3SAT$ is in NP.
- $\therefore 3SAT \leq_P SAT$.

$CLIQUE$

$CLIQUE$ is NP-complete

Karp's 21 NP-complete problems

- See: http://en.wikipedia.org/wiki/Karp's_21_NP-complete_problems.

Not all hard problems are NP-complete

- Some algorithms that are polynomial time, but can look hard:
  - Dynamic programming
  - Linear programming
  - Bipartite matching
- Harder than P but easier than NP?
  - If $P \neq NP$, then there must be an infinite number of problems in between the two. Here are candidates for "gap" problems:
  - Factoring
  - Graph isomorphism
- Harder than NP
  - There are problems that are known to require exponential time or more.
  - Note that even these are "easier" than undecidable problems, but they are harder than NP complete.

This coming week (and beyond)

- Reading
  - Nov. 17 (Today): Sipser 7.4
  - Nov. 19 (Wednesday): class cancelled
  - Nov. 21 (Friday): Sipser 7.5
  - Nov. 24 (A week from today): Sipser 10.6
- Homework
  - Nov. 14 (Friday): HW 10 goes out.
  - Nov. 17 (Monday): HW 9 due.