Today’s NP-Completeness Example: One-in-three 3SAT

Research Advertisement
- Hybrid Automata
- Circuit Verification
- Parallel Computing
One-In-Three 3SAT

- Let $f$ be a 3cnf formula. Does there exist a satisfying assignment where exactly one literal in each clause is satisfied?

- One-in-three 3SAT is NP complete.
  - It is easy to see that one-in-three 3SAT is in NP, an assignment of truth values to variables suffices as a certificate.
    - Such a list is shorter than the original input, thus its size is polynomial in the length of the input.
    - Checking that each clause has exactly one satisfied literal for the given assignment is straightforward and polynomial time.
  - To show that one-in-three 3SAT is NP hard, we show that we can reduce 3SAT to one-in-three 3SAT.
    - We add variables and rewrite each clause to produce a modified formula that is one-in-three satisfiable iff the original formula had any satisfying assignment.
One-In-Three 3SAT: Details
Verifying the Reduction
Monotone One-In-Three 3SAT

One-in-three 3SAT remains NP complete even if we only consider 3cnf formulas where no literals are negated.

- Construct a clause that forces a particular variable, $t$, to be true, and another variable, $f$, to be false:

  $$(t \lor f \lor f)$$

  If you think it was cheating to use the same variable twice in the same clause, we could use the clauses:

  $$(f \lor b \lor c)(f \lor d \lor e)(f \lor g \lor h)$$
  $$\land (b \lor d \lor g)(c \lor e \lor h)$$

  We could make $f_1$ and $f_2$ in this fashion, and then use the clause $(t \lor f_1 \lor f_2)$ to create a variable that must be true.

- Now, anytime we need the inverse of some variable, $v$, we just add the clause $(v \lor vB \lor f)$. Any assignment that satisfies one-in-three 3SAT will assign opposite values to $v$ and $vB$. 
Let $U$ be a set. Let $C = \{C_\infty, C_\in, \ldots C_\uparrow\}$ be a collection of subsets of $U$. Is there a set of disjoint sets, $\{S_1, S_2, \ldots S_k\}$ with each $S_i \in C$ such that the $\bigcup_{i=1}^k S_i = U$?
Sipser’s Minesweeper Problem

See Sipser problem 7.30.
The Rambus Oscillator Challenge
Post-Silicon Debug
Interested?
This coming week (and beyond)

- **Reading**
  - Nov. 26 (Today): no reading
  - Nov. 28 (Friday): Sipser 6.1 and 6.2

- **Homework**

- **Final Exam:**
  - Dec. 6: 3:30-6:30pm
  - CHBE 103