Automata Theory & Formal Languages Final Project

CMSC 651

12/21/2010

Sandipan Dey
UMBC CSEE

Problem Statement

$X_5^{SAT}(\phi_1,\phi_2,\phi_3,\phi_4,\phi_5) = d_1 d_2 d_3 d_4 d_5$, where $d_i \in \{0,1\}$ and $d_i = 1 \Leftrightarrow \phi_i \in SAT$, with X_5^{SAT} 32-enumerable.

Theorem to Prove: If X_5^{SAT} is 5-enumerable, then P = NP.

Proof

Let's rewrite the function X_5^{SAT} as per **definition 1.1** in [1], **definition 2** in [2], **definition 4** in [3] s.t.,

$X_5^{SAT}(\phi_1,\phi_2,\phi_3,\phi_4,\phi_5) = SAT(\phi_1)SAT(\phi_2)SAT(\phi_3)SAT(\phi_4)SAT(\phi_5)$, where

$SAT(\phi_i) = \begin{cases} 1, & \phi_i \in SAT \\ 0, & \phi_i \notin SAT \end{cases}, \forall i = 1,\cdots,5$, a characteristic function.

The function X_5^{SAT} is easily seen to be computable with 5 || queries to SAT.

As argued in [1], if X_5^{SAT} can be computed with less (e.g., 4) queries to SAT, then SAT (NP complete) should be easy (in P?) in some sense $(\Rightarrow P = NP\,?)$

Fact 2.16 (ii) in [1] says that X_5^{SAT} is $2^5 = 32$ enumerable.
As in [1], we are interested to find out when the function X_5^{SAT} requires 5 queries to SAT.

But, given, X_5^{SAT} is 5-enumerable $\Leftrightarrow$ $X_{2^5}^{SAT}$ is 2^5 enumerable (replacing the constant 5 by 2^5)

$\Leftrightarrow$ SAT is 5 cheatable (by **Fact 2.16** (ii) and **definition 2.18** in [1])

(i.e., $\exists$ a polynomial time Oracle Turing machine with SAT as Oracle, such that $X_{2^5}^{SAT}$ can be computed by asking at most 5 queries to the Oracle SAT !).

$\Rightarrow$ SAT is cheatable (by **definition 2.18**, since $\exists k = 5$, s.t. SAT is k-cheatable)
$\Rightarrow$ P = NP (From theorem 3.20, corollary 3.21 (i) and 3.23 (iii) in [1], we have the result: if SAT is cheatable then P = NP)

Also, corollary 36 of [2] directly proves the result:

$P \neq NP \Rightarrow$ SAT is not cheatable
$\equiv$ SAT is cheatable $\Rightarrow$ P = NP (contra-positive)

Proof that SAT is not 5-cheatable (by contradiction, mimicking corollary 36 in [2])

Assume, to the contrary that SAT is 5-cheatble. Given an instance of SAT, divide into 2^5 sub-problems, (by trying all possible assignments to 5 variables $d_1, \cdots, d_5$): each sub-problem contains 5 variables smaller than the original problem. By theorem 30 in [2], one sub-problem can be eliminated. Again divide each of these $2^5 - 1$ sub-problems into two and use theorem 30 [2] to reduce $2^{10} - 2$ sub-problems to just $2^5 - 1$. Continuing this way, eliminate all the variables, SAT can be solved in polynomial time! A contradiction.

References

1. Amihood Amir, Richard Beigel, William I. Gasarch, Some Connections between bounded query classes and non-uniform complexity
2. Richard Beigel, Bounded queries to SAT and the Boolean hierarchy
3. Richard Beigel, When are k + 1 queries better than k queries
4. Robert Beals, Richard Chang, William I. Gasarch, Jacobo Toran, On finding number of Graph Automorphisms
5. Richard Chang, On the Query Complexity of Clique Size and Maximum Satisfiability
6. Richard Chang, William I. Gasarch, On Bounded Queries and Approximation