The Complexity of Satisfiability Problems with Two Occurrences

Jan Johannsen
LMU München

Let CNF(2) be the class of formulas $F \in$ CNF such that every variable occurs at most twice in F, and CNF(E2) the class of formulas in CNF in which every variable occurs *exactly* twice.

We study the complexity of variants of the satisfiability problem for formulas in CNF(2). In a previous paper [1], we have shown that SAT(2), i.e. SAT restricted to instances in CNF(2), is complete for deterministic logspace. The same holds for the problem NAE-SAT(2), not-all-equal satisfiability for formulas in CNF(2).

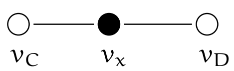
In this note we study the complexity of $\oplus$SAT(2), i.e., XOR-satisfiability and XSAT(2), i.e., exact satisfiability (XSAT), for formulas in CNF(2). A formula in CNF is XOR-satisfiable (resp. exact satisfiable), if there is an assignment that sets an odd number of literals (resp. exactly one literal) in each clause to true.

We shall show that $\oplus$SAT(2) is complete for symmetric logspace **SL**, and XSAT(2) is equivalent to the problem PM of deciding whether a graph contains a perfect matching.

A *tagged graph* $G = (V, E, T)$ is an undirected multigraph (V, E) with a distinguished set $T \subseteq V$ of vertices. We refer to the vertices in T as the *tagged* vertices.

For a formula $F \in$ CNF(2), we define the tagged graph $G(F)$ by

- $G(F)$ has a vertex v_C for every clause C in F.
- If clauses C and D contain the same literal a, then there is an edge e_a between v_C and v_D.
- if C contains a literal a, and D contains the complementary literals $\bar{a}$, then we add a new vertex v_a and connect it to v_C by an edge e_a and to v_D by an edge $e_{\bar{a}}$, as shown below.



- If C contains a literal, that does not occur in another clause, then v_C is tagged.

SL-completeness of $\oplus$SAT(2)

If G is a (tagged) graph, then we call a coloring of the edges by two colors 0, 1 admissible if every (untagged) vertex has an odd number of incident edges colored by 1. Obviously, for $F \in \mathrm{CNF}(2)$, we have that $G(F)$ has an admissible coloring iff F is in $\oplus$SAT(2).

Define the problem EvenCC (resp. TEvenCC) as the problem to determine for a given (tagged) graph G, whether every (untagged) connected component has an even number of vertices.

Proposition 1. *A tagged graph* G *has an admissible coloring iff it is in* TEvenCC.

Proof. Let G have an admissible coloring, and let C be an untagged component of odd size. Since C has even number of vertices of odd degree, the number of vertices of even degree is odd. Therefore, in the edge subgraph consisting of the edges colored 0, the component C has an odd number of vertices of odd degree, which is impossible. Hence every untagged component is of even size, and G is in TEvenCC.

For the other direction, we let G be in TEvenCC, and construct an admissible coloring of G. First, it is easy to see, analogous to the proof of Lemma 9 in [1], that every tagged component has an admissible coloring.

Note that if there is an admissible coloring for a spanning forest of G, then it can be extended to an admissible coloring of G by giving all the missing edges the color 0. Therefore, it suffices to give an admissible coloring for a tree T of even size, which is done by induction on the size of T. We distinguish two cases.

If all vertices in T have odd degree, then all edges can be colored by 1.

Otherwise, we show that there is an edge e such that deleting e leaves two trees of even size, which have admissible colorings by the induction hypothesis. These can be extended to T by coloring e with 0.

To see that the edge e exists, let v be a vertex of even degree, and let $e_1, \ldots, e_k$ be the incident edges, and let T_i be the subtree reached by following e_i from v. Since $|T_1| + \ldots + |T_k| = |T| - 1$ is odd, and k is even, there must be some i such that $|T_i|$ is even. Thus deleting e_i cuts T into two trees of even size. □

Corollary 2.

- $\oplus$SAT(E2) *is equivalent to* EvenCC *under* FO*-reductions.*
- $\oplus$SAT(2) *is equivalent to* TEvenCC *under* FO*-reductions.*

Proposition 3. TEvenCC *is complete for* **SL**.

Proof. The obvious algorithm for TEvenCC tests for every vertex v, whether the number of vertices reachable from v is even, or whether there is a tagged vertex among them. If for some v neither holds then reject, otherwise accept. This can be done in logarithmic space with an oracle for UGAP, thus TEvenCC $\in \mathbf{L}^{\mathbf{SL}}$, and by the result of Nisan and Ta-Shma [2], $\mathbf{L}^{\mathbf{SL}} = \mathbf{SL}$.

For hardness, we reduce UGAP to EvenCC as follows: Given a graph G and vertices s and t, we construct a graph G' as follows: we take two copies G_0 and G_1 of G, and for each vertex v in G, we put an additional edge between the two copies v_0 and v_1 of v. Then we add two new vertices s^* and t^*, and edges between s^* and s_0 and s_1, as well as between t^* and t_0 and t_1.

If t is reachable from s, then every connected component of G' is of even size, otherwise the connected components containing s^* and t^* are of odd size. Thus the construction reduces UGAP to EvenCC. □

Corollary 4. $\oplus$SAT(2) *is complete for* **SL**.

Equivalence of XSAT(2) to Perfect matching

For an assignment α to the variables of F, consider the edge subgraph of $G(F)$ containing those edges e_a for which the literal a is set to true by α. If α satisfies F exactly, then this edge subgraph is a matching in $G(F)$ that matches every untagged vertex.

Thus we define the following variant of the perfect matching problem for tagged graphs:

> TPM: Given a tagged graph G, is there a matching in G that matches every untagged vertex?

Proposition 5.

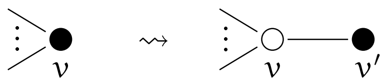
- XSAT(E2) *is equivalent to* PM *under* FO*-reductions.*
- XSAT(2) *is equivalent to* TPM *under* FO*-reductions.*

The construction of $G(F)$ from F gives the reduction in one direction for both statements, since for $F \in$ CNF(E2), the graph $G(F)$ contains no tagged vertices. For the other direction, given a tagged graph $G = (V, E, T)$, we define a formula $F(G) \in \mathrm{CNF}(2)$ as follows: for every edge $e \in E$, there is a variable x_e. For every vertex we form a clause C_v containing the variables x_e for the edges e incident on v. Finally, for every tagged vertex $v \in T$, we add a variable x_v to the clause C_v. It is easily seen that $G(F(G)) = G$, and hence the construction gives the opposite reductions. Note that the reduction produces only formulas with only positive literals.

We now show that XSAT(2) is equivalent to the more natural problem PM as well, in two steps. Unfortunately, we need slightly more complex reductions.

Proposition 6. TPM *is equivalent to* rTPM *under* FO*-reductions.*

We only need to reduce TPM to rTPM, the other direction is trivial. Given a tagged graph G, construct a graph G' by untagging every tagged vertex v and connecting it by an edge to a new tagged vertex v', as shown below.



A tagged perfect matching in G exactly corresponds to a tagged perfect matching in G', where each tagged vertex v unmatched in G is matched to the corresponding vertex v' in G'. Thus the construction reduces TPM to rTPM.

Proposition 7. rTPM *is equivalent to* PM *under* $\mathrm{FO}(\mathrm{Mod}_2)$*-reductions.*

Given an instance G of rTPM, we construct G' as follows: if $|V|$ is even, we connect all tagged vertices in a large clique, otherwise, we add a new vertex, and connect this new vertex together with the tagged vertices in a large clique. If $|V|$ is even, then T and $|V \setminus T|$ will have the same parity, so a tagged perfect matching will leave an even number of tagged vertices unmatched. Otherwise, it will leave an odd number of tagged vertices unmatched. In either case, a tagged perfect matching in G can be extended to a perfect matching in G'. Thus the construction reduces rTPM to PM. The other direction is trivial.

Corollary 8. XSAT(2) *is equivalent to* PM *under* $\mathrm{FO}(\mathrm{Mod}_2)$*-reductions.*

References

[1] J. Johannsen. Satisfiability problems complete for deterministic logarithmic space. Accepted for the 21st International Symposium on Theoretical Aspects of Computer Science (STACS 2004), 2004.

[2] N. Nisan and A. Ta-Shma. Symmetric Logspace is closed under complement. *Chicago Journal of Theoretical Computer Science*, 1995.