On the Parallel Complexity of Quantified 2SAT

K. Subramani
Department of Computer Science and Electrical Engineering,
West Virginia University,
Morgantown, WV
ksmani@csee.wvu.edu

Abstract

Quantified Boolean Formulae (QBF) have been used to model problems in Constraint Logic Programming and Robotics and AI search domains. It is known that the problem of evaluating a QBF is `PSPACE-complete`, even when each clause represents the disjunction of at most 3 literals. However, polynomial time algorithms exist for the case in which each clause has at most 2 literals i.e. Q2SAT. In this paper, we explore the parallel complexity of the evaluating a Quantified 2SAT formula and present an `NC` algorithm for the same. To the best of our knowledge, our results are the first of their kind.

1 Introduction

Quantified Boolean formulae (QBF) are used to model a number of problems in Constraint Logic Programming [CH99], Artificial Intelligence [CGS98] and Real-Time scheduling [Ost89]. The decision problem of evaluating a QBF is known to be `PSPACE-complete` even when the propositional formula is an instance of 3-SAT. If each clause is restricted to have at most two literals (Q2SAT), then polynomial time algorithms are known [APT79, Sch78]. The parallel complexity of Q2SAT has not been explored in the literature. In this paper, we show that Q2SAT is in the parallel complexity class `NC`; in particular we show that the evaluation problem can be decided in $(\log^2 n)$ time, using at most $O(\frac{n^3}{\log n})$ processors i.e. Q2SAT $\in$ $\mathtt{NC}_2$.

The rest of this paper is organized as follows: Section §2 formally presents the Q2SAT problem. The motivation underlying our approach as well as related work are discussed in §3. We present our algorithm and analyze its complexity in §4. A summary of our work and a discussion of open problems is presented in §5.

2 Statement of Problem

Let

$$F = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \;\; C \qquad (1)$$

be a quantified boolean formula, where

- Each Q_i is either a $\exists$ or a $\forall$,
- Each variable x_i is bound to some quantifier,
- C is a conjunction of clauses, such that each clause is a disjunction of at most two literals [1]

The evaluation problem is concerned with the following question : Given F, is F true ?

[1] A variable x_i gives rise to 2 literals, x_i and $\overline{x_i}$.

2.1 Graph Representation for 2SAT formuale

[APT79] and [Pap94] provide a directed graph representation for 2SAT formulae. We present their construction here. Given a collection of m 2SAT clauses $C_i, i = 1, \ldots, m$, over the n boolean variables $x_1, x_2, \ldots, x_n$, construct graph $\mathbf{G}$ with vertex set $x_1, x_2, \ldots, x_n, \overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$. Corresponding to each clause C_i of the form (x_i, x_j), add edges $(\overline{x_i}, x_j)$ and $(\overline{x_j}, x_i)$. Thus $\mathbf{G}$ has $2n$ vertices and $2.m$ edges. Additionally, each vertex contains a bit which represents its quantifying type, i.e. universal or existential.

For instance, the formula

$$\begin{aligned} F \;=\; & \exists\, x_1 \,\forall x_2 \,\exists x_3 \\ & (\overline{x_1}, x_2) \\ & (\overline{x_1}, x_3) \\ & (\overline{x_2}, \overline{x_3}) \end{aligned}$$

is represented by the graph in Figure (1).

Observation: 2.1 *Each edge of the graph represents an implication; the clause (x_i, x_j) represents the conjunction of the two implications $(\overline{x_i} \rightarrow x_j)$ and $(\overline{x_j} \rightarrow x_i)$. The graph $\mathbf{G}$ is called the implication graph corresponding to the formula F.*

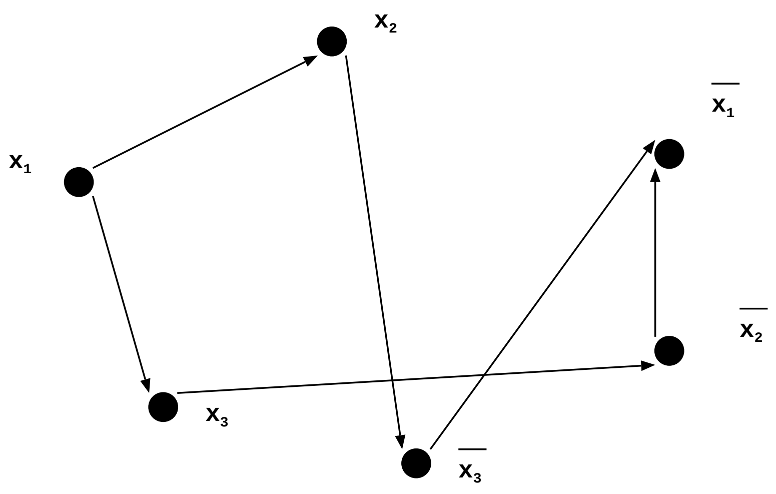

Figure 1: Graph representation of a Q2SAT formula

Theorem: 2.1 <u>*Aspvall,et.al:*</u> *The Q2SAT formula F is true if and only if none of the followning three conditions hold:*

1. *There is a path from an existential vertex u_i to its complement $\overline{u_i}$ and vice versa;*
2. *There is a path from a universal vertex u_i to an existential vertex u_j and vice versa, with $j < i$ i.e. x_j is not within the quantified scope of Q_i;*
3. *There is a path from a universal vertex u to another universal vertex v.*

3 Motivation and Related Work

Computational approaches for the general QBF evaluation problem have been discussed extensively in [CGS98] and [CG$^+$97]. Their work emphasizes the necessity for faster sequential and parallel algorithms for this problem. While it is unlikely that there exist efficient parallel algorithms for the general case, it is worthwhile to explore the possibility for restricted subclasses, such as Q2SAT.

Schaefer indicates in [Sch78] that the Q2SAT evaluation problem is solvable in polynomial time; however no proof is given. [APT79] presents a linear-time algorithm for Q2SAT; in this paper, we shall show how their approach can be parallelized.

4 A Parallel Algorithm

We first review the requirements for a 2SAT formula to be true.

Lemma: 4.1 *A 2SAT formula i.e. a Q2SAT formula in which every $Q_i = \exists$ is true under an assignment if and only if:*

1. *For all i, vertices x_i and $\overline{x_i}$ receive complementary values, and*
2. *There is no path from a true vertex to a false vertex.*

The above two conditions can be unified as follows: The formula is satisfiable, if and only if there do not exist paths between a vertex x_i and its complement $\overline{x_i}$ and vice versa in the implication graph $\mathbf{G}$.

Proof: *Obvious, from the definition of the implication graph $\mathbf{G}$.* □
Prior to presenting our algorithm, we discuss the complexity of Q2SAT.

Lemma: 4.2 *Q2SAT $\in$ `AL`, i.e. Q2SAT can be solved by an Alternating Turing Machine in logarithmic space.*

Definition: 4.1 *NQ2SAT denotes the set of Quantified 2SAT formulae, which evaluate to* **false**.

Observation: 4.1 *The "yes" instances of NQ2SAT are the "no" instances of Q2SAT and vice versa, i.e. NQ2SAT is the complement of Q2SAT.*

Proof: *We show that NQ2SAT can be decided in $O(\log n)$ space using an Alternating Turing Machine. Since `AL` is closed under complementation, the lemma follows. It is clear that Algorithm (4.1) decides NQ2SAT. The input to the algorithm is the implication graph discussed in Section §2. The key issue is that we do not have to store all the guessed moves; clearly that would take $O(n)$ space. Instead, we store the indices of the current two vertices under consideration and process one edge at a time. Clearly this can be achieved in $O(\log n)$ space.* □

The above result is not surprising, since we know from [Pap94], that `AL = P`, i.e. all languages that can be decided in polynomial time by a deterministic Turing Machine, can be decided by an Alternating Turing Machine in logarithmic space and vice versa. The interesting question is whether an Alternating Turing Machine is *required* to evaluate a Q2SAT formula in logarithmic space. For instance, proving that the problem of deciding a language is `P-complete` assures us that an alternating machine is required, unless `NC=P`, an event which is considered unlikely. However, we show that a Nondeterministic Turing Machine suffices to decide the evaluation problem in logarithmic space.

Observation: 4.2 *The above technique expectedly breaks down for Quantified 3SAT formulae; in fact a Quantified 3SAT formula cannot be decided in $O(\log n)$ space using an Alternating Turing Machine, unless* `P=PSPACE`.

Lemma: 4.3 *Q2SAT is* `NL-Hard`.

Proof: *Even the restriction of Q2SAT to the case in which each quantifier is existential is* `NL-Hard`. *See [GJ79, Pap94].* □

Lemma: 4.4 *Q2SAT is in* `NL`.

```
Function NQ2SAT Decision Procedure using alternation( G )
1: Let $A$ and $B$ denote 2 players, with $A$ being the existential player and $B$ being the universal player.
2: $A's$ strategy is to guess values for the existential variables so that there is never a path from a true vertex
   to a false vertex.
3: $B's$ strategy is to guess values for the universal variables so that a path from a true vertex to a false vertex
   is forced.
4: Guess a path from a vertex $x_i$ to another vertex $x_j$ as follows:
5: if ($x_i$ is an existential variable) then
6:    Enter existential state and make a guess on $x_i$.
7: else
8:    Enter universal state and make a guess on $x_i$.
9: end if
10: if ( there is a path from a true vertex to a false vertex ) then
11:    return( true )
12: else
13:    return( false )
14: end if
```

Algorithm 4.1: An alternating Turing Machine to decide NQ2SAT

Proof: *We argue that the complement of Q2SAT i.e. NQ2SAT is in* NL. *Since* NL *is closed under complementation, the claim follows.*

From the discussion in Section §2, we know that the nondeterministic algorithm (4.2) provides the required decision procedure. Further, it requires space at most $3.\log n$ *i.e.* $O(\log n)$. *We point out that the entire sequence of paths is never guessed; that would entail space more than* $O(\log n)$. *Instead succeeding vertices on the path are guessed and verified using the input adjacency matrix. For details of implementing graph reachability, in space* $O(\log n)$, *see [PS82].*
□

Lemma: 4.5 *Q2SAT is* NL-complete.

Proof: *Follows from Lemma (4.3) and Lemma (4.4).* □

Lemma: 4.6 *Q2SAT* $\in \mathtt{NC}_2$.

Proof: *Any language in* NL *can be decided in* $O(\log^2 n)$ *time with a polynomial number of processors.* □

We now present an algorithm which evaluates a Q2SAT formula in $O(\log^2 n)$ time, using $O(\frac{n^3}{\log n})$ processors. The input to the algorithm is the implication graph **G**, discussed in Section §2.

4.1 Analysis

Computing the reachability matrix can be accomplished in $\lceil \log 2n \rceil$ time i.e. $O(\log n)$ time, using $O(\frac{n^3}{\log n})$ procesors, using the matrix multiplication method, outlined in [Pap94]. The key idea is to use repeated squaring and compute the matrices, $\mathbf{G^2}$, $\mathbf{G^4}$ and so on.

Checking the conditions in the second step can be implemented in $O(1)$ time using $O(n^2)$ processors [Rei93].

Corollary: 4.1 *The quantified version of any problem in* NL *can be decided in* NL.

Proof: *Any problem in* NL *can be reduced to 2SAT.* □

```
Function NQ2SAT Decision Procedure
1: Guess the sequence of vertices on a path from an existential vertex $u$ to its complement $\overline{u}$ and vice-versa.
2: if (both paths exist) then
3:   return( true)
4: else
5:   return( false)
6: end if
7: Guess the path between a universal vertex $u_j$ and an existential vertex $u_i$ and vice-versa, where $j < i$.
8: if (both paths exist) then
9:   return( true)
10: else
11:   return( false)
12: end if
13: Guess the path from a universal vertex $u$ to another universal vertex $v$.
14: if ( such a path exists ) then
15:   return( true)
16: else
17:   return( false)
18: end if
```

Algorithm 4.2: A nondeterministic algorithm for deciding NQ2SAT

```
Function Q2SAT Decision Procedure(G)
1: Compute $\mathbf{G^{2n}}$, where $\mathbf{G^{2n}}$ is the reachability matrix of the graph $\mathbf{G}$.
2: if ( any of the conditions in Theorem (2.1) holds ) then
3:   return ( false )
4: else
5:   return ( true )
6: end if
```

Algorithm 4.3: An $\mathtt{NC}_2$ algorithm to decide Q2SAT

5 Conclusions and Future research

In this paper, we have been able to show that the quantified version of the 2SAT problem is in NC; in particular we designed a parallel algorithm that decides the evaluation problem in $O(\log^2 n)$ parallel time, using $O(\frac{n^3}{\log n})$ processors. We point out that our goal was not to design an optimal parallel algorithm, but to provide proof of the existence of parallelizability. In the full paper, we hope to show that the processor cost can be reduced to n^2.

A natural consequece of our result is that the quantified version of any `NL-complete` problem is in NC. Contrast this result with the AL=P result discussed in [Pap94].

Our interest in Q2SAT stems from our focus on the problem of parametric scheduling [Sub00]. We suspect that the parametric scheduling problem with "standard constraints" is `P-complete`; however we have been unable to prove the same. We intend to use the reachability method outlined in this paper, to show that restricted versions of the Parametric Scheduling problem can be decided in NC.

A Complexity Classes

1. P - The set of languages that can be decided in polynomial time by a deterministic Turing Machine;
2. NP - The set of languages that can be decided in polynomial time by a nondeterministic Turing Machine;
3. AL - The set of languages that can be decided in *logarithmic space* by an alternating Turing Machine. It has been shown that AL = P;
4. NC - The set of languages that can be decided in polylogarithmic space, using a polynomial number of processors;
5. PSPACE - The set of languages that can be decided by a deterministic Turing Machine in space that is polynomial in the size of the input.

References

[APT79] Bengt Aspvall, Michael F. Plass, and Robert Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, (3), 1979.

[CG+97] M. Cadoli, , M. Giovanardi, A. Giovanardi, and M. Schaerf. Experimental analysis of the computational cost of evaluating quantified boolean formulae. In *Lecture Notes in Artificial Intelligence*, 1997.

[CGS98] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *AAAI-98*, July 1998.

[CH99] V. Chandru and J. N. Hooker. *Optimization Methods for Logical Inference.* Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., 1999.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness.* W. H. Freeman Company, San Francisco, 1979.

[Ost89] Jonathan S. Ostroff. *Temporal Logic for Real-Time Systems.* Research Studies Press Ltd., England, 1989.

[Pap94] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, New York, 1994.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Prentice Hall, 1982.

[Rei93] John Reif, editor. *Synthesis of Parallel Algorithms.* Morgan Kaufmann, September 1993.

[Sch78] T.J. Schaefer. The complexity of satisfiability problems. In Alfred Aho, editor, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, New York City, NY, 1978. ACM Press.

[Sub00] K. Subramani. *Duality in the Parametric Polytope and its Applications to a Scheduling Problem.* PhD thesis, University of Maryland, College Park, July 2000.