Analysis of Search Based Algorithms for Satisfiability of Quantified Boolean Formulas Arising from Circuit State Space Diameter Problems

Daijue Tang, Yinlei Yu, Darsh Ranjan, and Sharad Malik

Princeton University
{dtang,yyu,dranjan,malik}@Princeton.EDU

Abstract. The sequential circuit state space diameter problem is an important problem in sequential verification. Bounded model checking is complete if the state space diameter of the system is known. By unrolling the transition relation, the sequential circuit state space diameter problem can be formulated as an evaluation for satisfiability of a Quantified Boolean Formula (QBF). This has prompted research in QBFs in the verification community. Most of existing QBF algorithms, such as those based on the DPLL SAT algorithm, are search based. We show that using search based QBF algorithms to calculate the state space diameter of sequential circuits with existing problem formulations is no better than an explicit state space enumeration method. This result holds independent of the representation of the QBF formula. This result is important as it highlights the need to explore non-search based or hybrid of search and non-search based QBF algorithms for the sequential circuit state space diameter problem.

1 Introduction

A quantified Boolean formula (QBF) is a Boolean formula with its variables quantified by either universal $\forall$ or existential $\exists$ quantifiers. The problem of deciding whether a quantified Boolean formula evaluates to true or false is also referred to as the QBF problem. Theoretically, QBF problems belong to the class of P-SPACE complete problems, widely considered harder than NP-complete problems like Boolean Satisfiability (SAT) Problems.

Many problems in AI planning [1] and sequential circuit verification [2] [3] can be formulated as QBF instances. In recent years, there has been an increasing interest within the verification community in exploring QBF based sequential verification as an alternative to Binary Decision Diagram (BDD) based techniques. Therefore, finding efficient QBF evaluation algorithms is gaining interest in sequential verification. Like SAT evaluation, QBF evaluation can be search based and does not suffer from the potential space explosion problem of BDDs. This makes it attractive to use QBF over BDD based algorithms since the problem of QBF evaluation is known to have polynomial space complexity. An obvious linear space algorithm to decide QBF assigns Boolean values to the variables and recursively evaluates the truth of the formula. The recursion depth is at most the number of variables. Since we need to store only one value of the variable at each level, the total space required is $O(n)$ where n is the number of variables.

Many complete QBF evaluation algorithms have been developed and several complete QBF solvers have been implemented. In [4], a resolution based QBF evaluation algorithm is presented. But like most resolution based decision methods, this algorithm has a potential memory blow up problem. In [5], the authors propose a decision procedure that achieves the effect of resolving multiple variables at one time by enumerating conflicts of cut variables. Besides the potential memory blow up problem, this method is likely to be inefficient for problems without small cuts. In practice, for many QBF instances, during the execution of this method, the variables will be so much interleaved that it is impossible to find a small cut. Partly due to its success in SAT solvers, the Davis Logemann Loveland (DPLL) algorithm [6] has been adapted to many QBF evaluation procedures [7][8][9] [10][11] [12][13][14]. Although DPLL based QBF solvers do not blow up in space, they consume significant CPU time and are unable to handle practical sized problems as of now.

In sequential verification, symbolic model checking is a powerful technique and has been used widely. Traditional symbolic model checking uses BDDs to represent sequential systems. With the development of many efficient SAT solvers, bounded model checking (BMC) [3] has emerged as an alternative approach to perform model checking. Although BMC uses fast SAT solvers and may be able to quickly find counter examples, it is incomplete in the sense that it can not determine when to stop the incremental unrolling of

the transition relation. The maximum number of unrollings needed to complete BMC is the diameter of the corresponding sequential circuit state space. Therefore, determining sequential circuit state space diameters is crucial for the completeness of BMC and its solution will have practical significance. Existing methods for computing the sequential circuit state space diameters [15] [16] search for new states at every step of time frame expansion. The search can be done by multiple calls to the SAT procedure or with a combination of BDD methods.

The sequential circuit state space diameter problem can also be tackled by formulating it as instances of QBFs and using QBF solvers to solve them. But currently this approach lags behind other methods. The immaturity of QBF evaluation techniques is often considered as the major reason for this. In this paper, we show that for the existing QBF formulations of the circuit diameter problem, search based QBF algorithms (this includes all DPLL based solvers) have no hope to outperform algorithms based on explicit reachable state space enumeration. This result is important as it underscores the need to explore non-search based or possibly hybrids of search and non-search based techniques.

2 The Sequential Circuit State Space Diameter Problem and Its QBF Formulations

A QBF is of the form $Q_1 x_1 \cdots Q_n x_n\ \varphi$, where $Q_i (i = 1 \cdots n)$ is either an existential quantifier $\exists$ or a universal quantifier $\forall$. φ is a propositional formula with $x_1 \cdots x_n$ as its variables. Adjacent variables quantified by the same quantifier in the prefix can be grouped together to form a quantification set. The order of the variables in the same quantification set can be exchanged without changing the QBF evaluation result (true or false). Variables in the outermost quantification set are said to have quantification level 1, and so on.

The propositional part φ in a QBF is usually expressed in the Conjunctive Normal Form (CNF). If φ of a QBF is in CNF, the innermost quantifier of this QBF is existential because the innermost universal quantifier can always be dropped by removing all the occurrences of the variables quantified by this universal quantifier in the CNF. The innermost quantifier can be a universal quantifier if φ is not in CNF. Converting φ to CNF needs to introduce new variables and quantify these new variables with existential quantifiers put inside the originally innermost quantifier. The number of quantification levels of a QBF may change if the representation of φ of this QBF changes. When φ is in CNF, the QBF having k levels of quantification is called kQBF. Most practical QBF instances are 2QBF or 3QBF. In the rest of the paper, when we talk about kQBF, k is the number of quantification levels when φ is in CNF.

Many problems in hardware verification concern verifying certain properties of logic circuits. For combinational circuits, we can formulate such problems as QBF instances by introducing one variable for each circuit node. A circuit node is either a primary input or a gate output. The propositional part is usually written as $\varphi = S \cdot P$, where S represents the consistency condition of combinational circuit C and P is the conjunction of the properties of C that need to be verified. Each logic gate in C can be represented as either a Boolean formula directly translated from the gate operation or a set of clauses that capture the consistent logic conditions associated with that gate. S is the conjunction of the Boolean representation for each logic gate.

The behavior of a sequential circuit over a number of time frames can be modeled using the conventional time frame expansion approach, which creates a combinational circuit by unrolling the next state function of the sequential circuit. The sequential circuit state space diameter problem can be formulated as a QBF by unrolling the next state function. The shortest path from one state s_i of a sequential circuit to another state s_j is defined as the minimum number of steps to go from s_i to s_j in the corresponding state transition graph. Clearly, every state on a shortest path appears at most once, which means that a shortest path has no loop. The state space diameter of a sequential circuit is the longest shortest path from one of the original states of this sequential circuit to any other reachable state. Figure 1 shows the combinational circuit constructed at each step of the circuit state space diameter calculation. We have two expansions of the combinational logic, one for $n + 1$ time frames and the other for n time frames. I_i and $I'_i (i = 1, 2, \cdots)$ are sets of primary inputs, O_i and $O'_i (i = 1, 2, \cdots)$ are sets of primary outputs and S_i and $S'_i (i = 0, 1, \cdots)$ are sets of state variables. Let C_1 denote the $n + 1$ time frame expansion part and C_2 denote the n time frame expansion part. Let $F(C_1)$ and $F(C_2)$ be the Boolean functions representing the logic consistencies of C_1 and C_2 respectively. If $I(S)$ is the characteristic function of the initial states, $T(I, S, S')$ is the characteristic function of the state transition relation, then $F(C_1) = I(S_0) \wedge \bigwedge_{i=0}^{n} T(I_i, S_i, S_{i+1})$ and

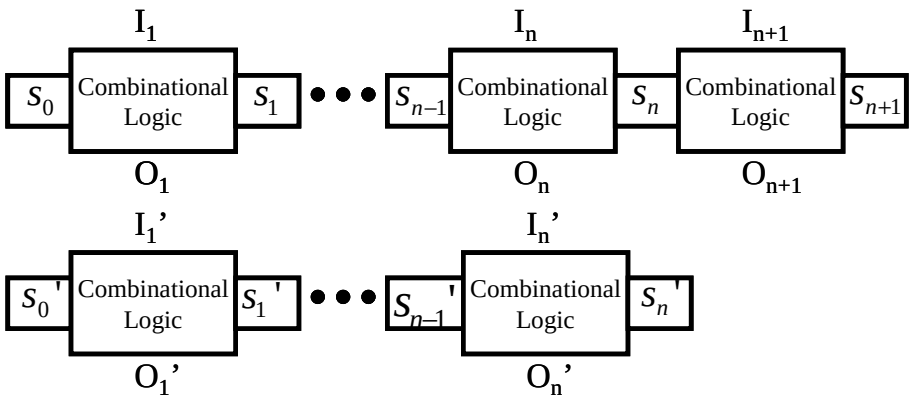

Fig. 1. time frame expansions for state space diameter calculation

$F(C_2) = I(S'_0) \wedge \bigwedge_{i=0}^{n-1} T(I'_i, S'_i, S'_{i+1})$. If $E1$ is the set of variables of $F(C_1)$ and $E2$ is the set of variables of $F(C_2)$, $E1$ and $E2$ are also sets of circuit node variables of C_1 and C_2 respectively. If for all possible input sequences of C_1, the state reached at the $(n+1)$th time frame can be reached at one or more of the first n time frames for some input sequence of C_2, then $n+1$ is greater than the state space diameter of the sequential circuit. A straightforward translation of the above sentence gives us the 2QBF formulation of the circuit state space diameter problem:

$$\forall I_1 I_2 \cdots I_{n+1} \exists((E1 \setminus \bigcup_{i=1\cdots n+1} I_i) \cup E2)\ F(C_1) \wedge F(C_2) \wedge (\bigvee_{i=0\cdots n} S_{n+1} = S'_i) \tag{1}$$

Let the state space diameter of the circuit be d. If $n < d$, (1) evaluates to false; if $n \geq d$, (1) evaluates to true. A very similar formulation is:

$$\forall E1 \exists E2\ \neg F(C_1) \vee (F(C_2) \wedge (\bigvee_{i=0\cdots n} S_{n+1} = S'_i)) \tag{2}$$

(2) is correct according to the previous statement of the circuit state space diameter problem.

Another way to state the state space diameter problem is: if $n < d$, then there exists S_{n+1} that cannot be reached from S_0 in less than $n+1$ steps; if $n \geq d$, such S_{n+1} does not exist. A direct translation of the above statement gives us the following QBF:

$$\exists E1 \forall E2\ F(C_1) \wedge \neg(F(C_2) \wedge (\bigvee_{i=0\cdots n} S_{n+1} = S'_i)) \tag{3}$$

(3) is actually the dual of (2). When $n < d$, (3) is true; when $n \geq d$, (3) is false. If the sequence of states $S_0 S_1 \cdots S_{n+1}$ has one or more loops in the state transition graph, then a sequence of states starting from S_0 and ending in S_{n+1} with less than $n+1$ state transitions must exist because we can always take the path without looping. Therefore, (3) is equivalent to:

$$\exists E1 \forall E2\ F(C_1) \wedge (\bigwedge_{i \neq j} S_i \neq S_j) \wedge \neg(F(C_2) \wedge (\bigvee_{i=0\cdots n} S_{n+1} = S'_i)) \tag{4}$$

The constraint $(\bigwedge_{i \neq j} S_i \neq S_j)$ in the propositional part of (4) is actually the condition to determine the recurrence diameter [3], adding this constraint does not change the QBF evaluation since the recurrence diameter is an upper bound of the state space diameter. In [16], the circuit netlist is modified to get a simpler QBF formulation than (4). They add a new auxiliary primary input to the sequential circuit netlist and the logic gates that have the following effect: when the auxiliary input is 0, the circuit operation is not changed; when it is 1, the present state makes a transition to the initial state. With this circuit modification, the formulation becomes:

$$\exists E1 \forall E2\ F(C_1) \wedge (\bigwedge_{i \neq j} S_i \neq S_j) \wedge \neg(F(C_2) \wedge (S_{n+1} = S'_n)) \tag{5}$$

The propositional part of (1)-(5) can be transformed to CNF by introducing new variables. This makes (3), (4) and (5) become 3QBFs while (1) and (2) are 2QBFs. In all cases, n is incrementally tested from 1 until $n = d$. d is the minimum number for n that makes (1), (2) true and (3), (4), (5) false. The drawback of the last four formulations is that when represented in CNF they have a large set of universal variables and introduce too many new variables.

3 QBF Algorithms

3.1 Overview

Just as in SAT, the propositional part φ of a QBF is often in CNF. Most existing QBF algorithms are complete and can be roughly divided into two categories: resolution based and search based.

QBF algorithms based on resolution use Q-resolution to eliminate variables until there is no more variable to eliminate or an empty clause is generated [4]. Only an existential variable can be resolved out in a Q-resolution. A universal variable in a Q-resolution generated clause can be eliminated when there are no existential variable having higher quantification level than this universal variable. Resolution based QBF algorithms have the potential memory blow up problem. Therefore they are seldom used in practice.

The majority of recent QBF solvers are search based. A search based algorithm tries to evaluate QBF by branching on variables to determine the value of φ at certain branches in the search tree. Note that we may not need to go all the way to the leaves of the search tree to determine the value of φ. A partial assignment to the variables may be enough for φ to be 0 or 1. Also we do not limit the search based algorithms to any particular search method like depth-first search or breadth-first search. Nor do we have any limitation on the ordering of the nodes in the search tree. The well-known DPLL algorithm, which is a depth-first search procedure, is just one example of the search algorithms.

Plaisted *et al.* proposed an algorithm for QBF that belongs to neither of the above categories [5]. This algorithm iteratively eliminates a subset of variables in the innermost quantification level. This is done by partitioning the propositional formula using a set of cut variables and substituting one partition with a CNF of only the cut variables. The conflicting assignments of the cut variables are enumerated and the negations of the conflicting assignments are conjuncted to form the new CNF part. Unlike Q-resolution which can only eliminate one variable at a time, this algorithm can eliminate multiple variables simultaneously. However, enumerating conflicts may take exponential time in the number of cut variables therefore is very expensive for formulas without a small cut. In fact, variables of many practical QBF instances are so much interleaved that it is impossible to find a small cut. From another point of view, the process of searching for conflicts is similar to the search in search based algorithms. Particularly, for a 2QBF instance, if the cut set is chosen to be the universal variables, then this algorithm is essentially a DPLL search algorithm.

3.2 The DPLL Algorithm

The DPLL algorithm is the most widely used search based algorithm for QBF as well as SAT evaluation. It only requires polynomial space during execution. The original DPLL algorithm is a recursive procedure for SAT and is not very efficient. Modern SAT solvers enhance the original DPLL algorithm with techniques like non-chronological backtracking and conflict-driven learning[17][18], which greatly accelerate the SAT solvers. Some of the most efficient SAT solvers today [19] [17][20] are based on the DPLL framework. Because SAT is a restricted form of QBF in the sense that it only has existential quantifiers, most existing QBF solvers incorporate variations of the DPLL procedure and many of the techniques that work well on SAT can also be used in QBF evaluation with some modifications. [7] is probably the first paper that extends the DPLL algorithm for SAT to QBF evaluation. It gives the basic rules for formula simplification like rules for monotone literals and unit propagation for existential variables. Conflict driven learning and non-chronological backtracking are adapted to later DPLL based QBF solvers [21] [14][13]. Also, the idea of satisfiability directed learning, which is a dual form of conflict driven learning and is specifically for QBF, is introduced and incorporated in these solvers. Deduction techniques such as inverting quantifiers [8] and partial implicit unfolding [9] were proposed and implemented by Rintanen. These deduction rules deduce forced assignments to existential variables by assigning truth values to universal variables having higher quantification levels.

Note that DPLL based QBF evaluation requires the branching order obey the quantification order, which corresponds to the semantics of the formula. Other decision orderings may require exponential memory to

store the already searched space. They also make search hard to control and result in many fruitless searches. One exception is in 2QBF evaluation where no useless enumeration of the existential variables occurs since all existential variables do not precede universal variables in the prefix. Decision strategies with and without restriction to quantification order for 2QBF are described and compared in [22].

4 Previous Algorithms for the Circuit Diameter Problem

Most of existing algorithms calculate the sequential diameter as a byproduct of the sequential reachability analysis [15]. In these works, image computation is repeated from the set of initial states until the least fixed point is reached. The state sets are enumerated implicitly and stored in the form of either BDDs or Boolean formulas. Images are calculated using either BDD operations or SAT evaluations or a combination of these two methods.

Circuit diameter computation in [16] is purely SAT based. It does not calculate reachable states. Like in BMC, it unrolls the transition relation. The circuit unrolling is the same as the QBF formulations of the diameter problem as illustrated in figure 1. For each n, every end state S_{n+1} that is different from S_i $(i = 1 \cdots n)$ of the first expansion is enumerated using SAT solvers. Every enumerated S_{n+1} is tested by satisfiability evaluation to see if it is reachable in less than $n + 1$ steps from the initial state.

In [23], search based SAT procedure is used in model checking algorithms. However, that work mainly concerns the method for preimage computation. It is not clear how to use preimage computation in the calculation of the circuit state space diameter.

5 Analysis of Search Based QBF Algorithms for the Circuit Diameter Problem

5.1 Handling Conflicts and Satisfying Assignments

Search based algorithms evaluate QBF by assigning Boolean values to variables. The propositional part φ of a QBF has three possible evaluations under a partial assignment: false, true and undetermined. If the value of φ is false, the partial assignment is called a conflict; if the value of φ is true, the partial assignment is called a satisfying partial assignment. In these two cases, the search procedure will backtrack and may do some learning. The search procedure will continue assigning unassigned variables if the value of φ is undetermined. Search based QBF algorithms often learn from the result of φ being true or false to prevent getting into the same conflicting or satisfying space again and again. Learning can be considered as choosing a subset of the current partial assignment such that this subset can still result in φ being true or false. These subsets of partial assignments are usually cached for future search space pruning.

Sometimes a conflict of a Boolean formula F is also a conflict of a subformula F_{sub} of F. In this case, we say this conflict is local to F_{sub}. For example, consider the Boolean formula $(a + b)(b' + c)$. Partial assignment $a = 0, b = 0$ is a conflict. This conflict is actually local to $(a + b)$. If a QBF evaluates to false, we need to show that conflicts at certain branches cover the entire Boolean space of some existential variables. Conflict driven learning performs selected resolution, including long distance resolution in QBF [13], to get the actual reason for a conflict.

Unlike in SAT, a satisfying assignment in QBF does not mean the end of the search. The search algorithms need to see if for all combinations of universal variables φ is satisfiable. The pruning of the satisfying space is usually done by constructing a partial assignment that is sufficient for φ to be true. When φ is in CNF, this partial assignment is constructed by choosing from every clause at least one of the value 1 literals. We call this partial assignment a *cover set* of the satisfying assignment [12]. The idea of using cover set for satisfying space pruning is incorporated in many QBF solvers. It is also called good learning in [21] and model caching in [14]. For a QBF instance of n variables, a cover set with m literals implies 2^{n-m} satisfying assignments. In fact, when a cover set is stored in the database for future pruning, existential variables belonging to the highest quantification level can be eliminated from the cover set due to the semantics of QBF. Thus the cover sets for a 2QBF instance consist only of universal variables. The conjunction of a set of literals is called a cube. A cover set is a cube.

When a QBF is derived from a circuit netlist and the value of this QBF denotes whether or not certain property of this circuit holds, a conflict in the QBF is either an inconsistent assignment to gate inputs and outputs or a consistent assignment to the circuit nodes that does not satisfy the property. A satisfying assignment to the propositional part F of the QBF is a consistent assignment to the circuit nodes that satisfy

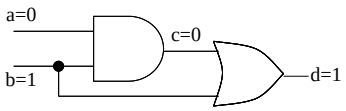

Fig. 2. example for critical and non-critical signals

the property. For some circuit node variables, their values do not affect the satisfiability of the property. This is because some logic gate output does not depend on the values of all the gate inputs. Therefore, some input is unobservable at the gate output. Any signal which fans out only to the transitive fanin cone of this input also becomes unobservable at this output. Suppose a circuit property is represented as a variable p, then p is the primary output of the property testing circuit C. Consistent assignments to other circuit nodes can either satisfy p or violate p, but some circuit nodes might become unobservable at p. For example, in figure 2, d represents the circuit property. If $b = 1$, then $d = 1$ which makes signals a and c unobservable at d. Such unobservable signals are called *non-critical* signals. The set of *critical* signals S_c is both sufficient and necessary as the reason of the satisfiability of the property. Removing any signal from S_c results in p being undetermined. So caching the satisfiability result of p should include all the variables in S_c. If F is in CNF and is satisfied, S_c may not satisfy every clause of the CNF. Critical signals for a consistent assignment to circuit signals that unsatisfies the property are both sufficient and necessary for the result of p being violated. The selection of S_c does not require the knowledge of F, the information of circuit structure is enough. S_c is generally much smaller than cover sets for CNF clauses. The above idea is very similar to the dynamic removal of inactive clauses in SAT proposed in [24].

5.2 Explicit State Enumeration vs. QBF Evaluation with Satisfiability Driven Learning

We now demonstrate that using the search based QBF algorithms to solve the circuit diameter problem with existing QBF formulations is no better than the previous explicit state enumeration methods.

We first analyze the 2QBF formulation of the diameter problem which is shown in (1). The property P in this case is:

$$(S_{n+1} = S_0 \vee S_{n+1} = S'_1 \vee \cdots \vee S_{n+1} = S'_n) \tag{6}$$

Since every equality in the disjunction of (6) has S_{n+1}, P can not be evaluated if the value of any state variable in S_{n+1} is unknown. This means any state variable in S_{n+1} is critical. In another words, any partial assignment that is satisfying for the propositional part of (1) must be a complete assignment for the state variables of S_{n+1}. Consider the Boolean space of the universal variables of (1) and the reachable state space at the $(n+1)$th time frame S_{n+1} as shown in figure 3. When using search based QBF algorithms to evaluate (1), we want to derive satisfying cubes to cover as much of the Boolean space of $I_1 \times I_2 \times \cdots \times I_{n+1}$ as possible for every satisfying assignment. Among the existing methods for deriving satisfying cubes for Boolean formulas arising from circuits, satisfying cubes consisting of all critical signals are the most effective in pruning the satisfying space of the search tree. Since the universal variables in (1) are all primary inputs, any set of critical signals must contain at least one universal variable. For one satisfying assignment, since there are one or multiple sets of critical signals, one or multiple cubes in the Boolean space of universal variables $I_1 \times I_2 \times \cdots \times I_{n+1}$ are pruned away. However, in the state space reachable at the $(n+1)$th time frame, only one minterm is pruned away for every satisfying assignment because all the state variables of S_{n+1} are critical. Due to the circuit structure, once the set of critical primary inputs are determined, all other critical signals can be computed as well. This means cubes in the universal space derived from one satisfying assignment corresponds to a single state of S_{n+1}. Thus minterms in the reachable state space of S_{n+1} are covered by non-overlapping sets of cubes in the universal Boolean space as illustrated in figure 3. If (1) is true, the entire Boolean space of universal variables needs to be covered by satisfying cubes. In this case, search procedures also need to cover the entire reachable state space of S_{n+1} because every reachable state results from at least one assignment to the universal primary inputs. The number of satisfying assignments that needs to be searched is at least the number of reachable states at the $(n+1)$th time frame. Therefore, when applying search based algorithms to the evaluation of (1), although we might get some pruning of the satisfying space of the search tree, we can not prune any part of the reachable state space. An explicit state enumeration algorithm can enumerate the states reachable at the $(n+1)$th time frame by calling the SAT procedure. The number of times to call SAT is exactly the number of states reachable at the $(n+1)$th time

frame when (1) evaluates to true. In addition, for search based algorithms applied to (1), not every minterm in the Boolean space of the universal variables corresponding to a new reachable state gets pruned away for a new satisfying assignment. The minterms in the reachable state space might get enumerated again and again. For example, the two cubes c_1 and c_2 map to the same state s in figure 3, but two satisfying assignments are needed to derive c_1 and c_2 respectively.

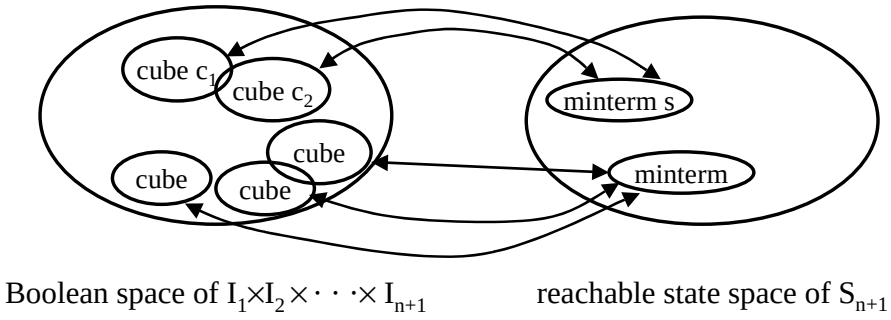

Fig. 3. space mapping for formula (1)

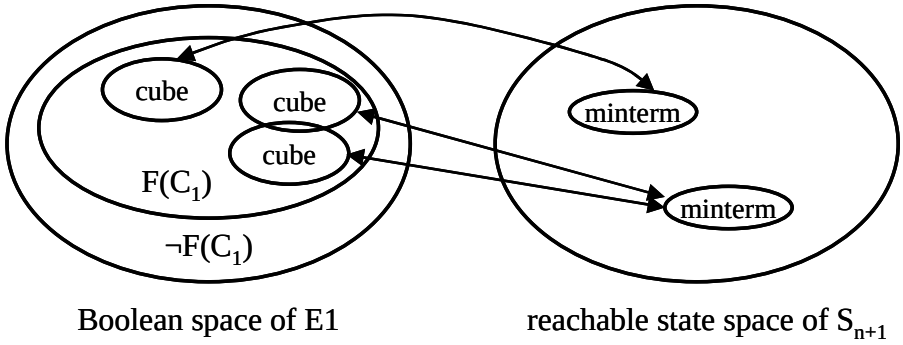

Fig. 4. space mapping for formula (2)

The set of universal variables in (1) is strictly contained in the set of universal variables in (2). The Boolean space of the universal variables in (2) can be divided into two parts: those that are consistent with the logic conditions of the expanded circuit C_1 and those that are inconsistent. Inconsistent assignments to $E1$ makes $\neg F(C_1)$ true thus the propositional part of (2) true. Cubes of consistent assignments to $E1$ map to the reachable states at the $(n+1)$th time frame as shown in figure 3. The mapping between the Boolean space of $E1$ and the reachable state space at the $(n+1)th$ time frame is shown in figure 4.

The analysis for the 3QBF formulations of the circuit state space diameter problem is a little more complicated but similar. The outermost quantification sets in the 3QBF formulations are all existentially quantified $E1$. If these 3QBFs are false, we need to prove that no subtree of the semantic tree below the branches of $E1$ is true. In these cases, we need to prune the search space that has conflicts. Let us first analyze the 3QBF of equation (5). The propositional part of (5) can be divided into two parts:

$$CNF(C1) \wedge (\bigwedge_{i \neq j} S_i \neq S_j) \tag{7}$$

and

$$\neg(CNF(C_2) \wedge (S'_n = S_{n+1})) \tag{8}$$

(7) has only variables in $E1$. Therefore, some assignments to $E1$ cause conflicts in (7). These conflicts are local to (7) and do not involve any variable in $E2$. They either represent inconsistent signal values in the first

expansion of the circuit or consistent signal values that have a loop in the path from S_0 to S_{n+1}. Conflict driven learning for conflicts local to (7) can only prune the conflicting spaces that are local to (7). If (5) is false, assignments to $E1$ that cause conflicts not local to (7) should also get pruned. Conflicts non-local to (7) may or may not exist. If for a certain n, every path of length $n+1$ has a loop in it, then all the local conflicts of (7) cover the entire Boolean space of $E1$. In this case, getting the local conflicts of (7) is enough to prove that (5) is false. When there exist non-looping paths of length $n+1$, conflicts involving (8) must be enumerated. Consider a conflict not local to (7). Since this conflict is not local to (7), it must make (8) false. Since (8) is the negation of

$$CNF(C_2) \wedge (S'_n = S_{n+1}) \tag{9}$$

conflicts not local to (7) must make (9) true. The only variables in (8) that are in the set $E1$ are variables in S_{n+1}. Assignments to variables in S_{n+1} alone cannot make (9) true. Thus critical signals of a non-local conflict to (7) must include variables in $E2$. On the other hand, every variable in S_{n+1} is critical because of the conjunction part $S'_n = S_{n+1}$ in (9). This means every conflict not local to (7) maps to exactly one minterm in the reachable state space. The set of states S_{nl} that are the end states of non-looping paths of length $n+1$ is a subset of the reachable states at the $(n+1)$th time frame. If (5) is false, every minterm in S_{nl} corresponds to a conflict of (5) not local to (7) and search algorithms need to enumerate minterms in S_{nl} one by one since conflict driven learning can only prune away one minterm in S_{nl} at a time. The above analysis is illustrated in figure 5.

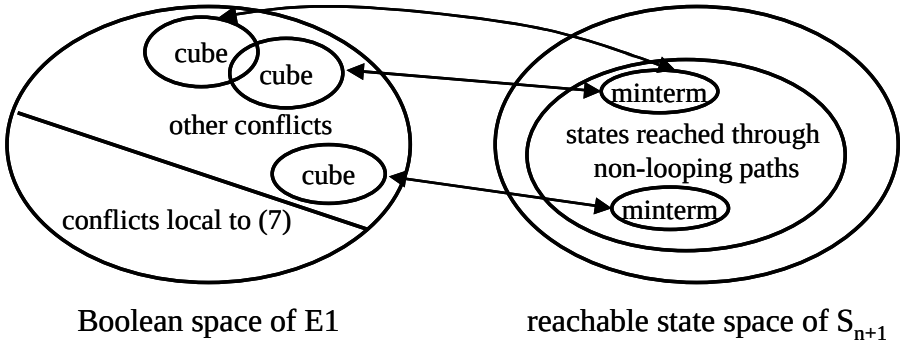

Fig. 5. space mapping for formula (4) and (5)

Note that in [16], states in S_{nl} are enumerated explicitly using the SAT procedure. Search algorithms for QBF evaluation can do no better than this since they too enumerate the states in S_{nl} explicitly. The results in [16] are in general not as good as the methods depending on calculating the least fixpoint of the reachable states from the set of initial states [15].

The mapping relation of (4) can be illustrated in figure 5 too. The illustration for formulation (3) is a little different from figure 5. For (3), the cubes in the left circle of figure 5 should map to a minterm in the whole reachable state space at the $(n+1)$th time frame, not just an end state of a non looping path with length $n+1$. The local conflicts should be in $CNF(C1)$ instead of formula (7).

It is worth emphasizing that although all the existing QBF solvers that we are aware of take CNF as their inputs, the above analysis holds independent of the representation of the formula. This is to say that if there was a QBF solver that worked directly off the circuit, the result would still hold since it depends only on the critical nature of the state variables in the formula.

6 Future Directions

From the analysis of last section, we can see that formulating the circuit diameter problem into QBF and using search based algorithms to evaluate the QBF currently is no better than the explicit state enumeration methods. However, there may be some room for optimism here. One possibility of making the QBF approach more efficient is changing the formulation of the diameter problem. The formulations described in section 2 are mainly based on unrolling the transition relations. Other formulations may require less universal variables thus greatly reduce the satisfying search space.

Another possible direction is devising more effective QBF algorithms. A partial search or a non-search based algorithm is definitely worth exploring. Current non-search based QBF algorithms include resolution based algorithms and Plaisted's algorithm proposed in [5]. Pure resolution based approach will likely blow up in space, but we could do some simplification of the resolved formula to alleviate the space explosion. It is also possible to combine the search based algorithms with resolution to get around the drawbacks of both approaches. Plaisted's algorithm in [5] is in some sense a hybrid of search and resolution. But this algorithm works well for circuits that are long and thin. The circuit constructed for calculating the sequential state space diameter in figure 1 is not long and thin. Thus Plaisted's algorithm is likely to be inefficient for the QBFs arising from the circuit state space diameter problem. Expanding the class of problems that Plaisted's algorithm work well on is another possibility of future research.

Overall for an algorithm to be successful, the reachable state space needs to be enumerated implicitly rather than explicitly by a QBF algorithm. In addition, a clever way of implicitly storing the already explored state space is critical as well.

7 Conclusions

In this paper we describe the QBF formulations of the circuit diameter problem. We prove that using search based QBF algorithms to determine the circuit state space diameter is no more efficient than previous explicit reachable state space enumeration algorithms. This result will direct the future QBF approaches for the circuit state space diameter problem away from pure search based algorithms. A non-search based algorithm or a hybrid of search based and non-search based methods are possible candidates for using QBF evaluation to solve the circuit state space diameter problem.

References

1. Rintanen, J.: Constructing conditional plans by a theorem prover. Journal of Artificial Intelligence Research **10** (1999) 323–352
2. Sheeran, M., Singh, S., Stålmark, G.: Checking safety properties using induction and a SAT-solver. In: Proceedings of the Third International Conference on Formal Methods in Computer-Aided Design (FMCAD 2000). (2000)
3. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without BDDs. In: Proceedings of Tools and Algorithms for the Analysis and Construction of Systems (TACAS'99). (1999)
4. Büning, H.K., Karpinski, M., Flögel, A.: Resolution for quantified Boolean formulas. Information and Computation **117** (1995) 12–18
5. Plaisted, D.A., Biere, A., Zhu, Y.: A satisfiability procedure for quantified boolean formulae. Discrete Appl. Math. **130** (2003) 291–328
6. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem proving. Communications of the ACM **5** (1962) 394–397
7. Cadoli, M., Schaerf, M., Giovanardi, A., Giovanardi, M.: An algorithm to evaluate quantified Boolean formulae and its experimental evaluation. Journal of Automated Reasoning **28** (2002) 101–142
8. Rintanen, J.: Improvements to the evaluation of quantified Boolean formulae. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (1999)
9. Rintanen, J.: Partial implicit unfolding in the Davis-Putnam procedure for quantified Boolean formulae. In: International Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR). (2001)
10. Giunchiglia, E., Narizzano, M., Tacchella, A.: Qube: a system for deciding quantified Boolean formulas satisfiability. In: Proceedings of International Joint Conference on Automated Reasoning (IJCAR). (2001)
11. Giunchiglia, E., Narizzano, M., Tacchella, A.: Backjumping for quantified Boolean logic satisfiability. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2001)
12. Zhang, L., Malik, S.: Towards a symmetric treatment of satisfaction and conflicts in quantified Boolean formula evaluation. In: Proceedings of 8th International Conference on Principles and Practice of Constraint Programming (CP2002). (2002)
13. Zhang, L., Malik, S.: Conflict driven learning in a quantified Boolean satisfiability solver. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD). (2002)
14. Letz, R.: Lemma, model caching in decision procedures for quantified Boolean formulas. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux2002). (2002)
15. Gupta, A., Yang, Z., Ashar, P., Gupta, A.: SAT-based image computation with application in reachability analysis. In: Proceedings of Third International Conference Formal Methods in Computer-Aided Design (FMCAD 2000). (2000)

16. Mneimneh, M., Sakallah, K.: Computing vertex eccentricity in exponentially large graphs: QBF formulation and solution. In: Sixth Intermational Conference on Theory and Application of Satisfiability Testing. (2003)
17. Marques-Silva, J.P., Sakallah, K.A.: GRASP - a search algorithm for propositional satisfiability. IEEE Transactions in Computers **48** (1999) 506–521
18. Zhang, L., Madigan, C.F., Moskewicz, M.W., Malik, S.: Efficient conflict driven learning in a Boolean satisfiability solver. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD). (2001)
19. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: Proceedings of the Design Automation Conference (DAC). (2001)
20. Goldberg, E., Novikov, Y.: BerkMin: A fast and robust SAT-solver. In: Proceedings of the IEEE/ACM Design, Automation, and Test in Europe (DATE). (2002)
21. Giunchiglia, E., Narizzano, M., Tacchella, A.: Learning for quantified Boolean logic satisfiability. In: Proceedings of the 18th National (US) Conference on Artificial Intelligence (AAAI). (2002)
22. Ranjan, D.P., Tang, D., Malik, S.: A comparative study of 2QBF algorithms. In: The Seventh International Conference on Theory and Applications of Satisfiability Testing. (2004)
23. McMillan, K.L.: Applying SAT methods in unbounded symbolic model checking. In: Proceedings of 14th Conference on Computer-Aided Verification (CAV 2002), Springer Verlag (2002)
24. Gupta, A., Gupta, A., Yang, Z., Ashar, P.: Dynamic detection and removal of inactive clauses in SAT with application in image computation. In: Design Automation Conference. (2001) 536–541