Boundary Points and Resolution

Eugene Goldberg

eu.goldberg@gmail.com

Abstract. We use the notion of boundary points to study resolution proofs. Given a CNF formula F, a $lit(x)$-boundary point is a complete assignment falsifying only clauses of F having the same literal $lit(x)$ of variable x. A $lit(x)$-boundary point mandates a resolution on variable x. Adding the resolvent of this resolution to F eliminates this boundary point. Any resolution proof has to eventually eliminate all boundary points of F. Hence one can study resolution proofs from the viewpoint of boundary point elimination. We use equivalence checking formulas to compare proofs of their unsatisfiability built by a conflict driven SAT-solver and very short proofs tailored to these formulas. We show experimentally that in contrast to proofs generated by this SAT-solver, almost every resolution of a specialized proof eliminates a boundary point. This implies that one may use the share of resolutions eliminating boundary points as a metric for proof quality.

Keywords: SAT-solver, boundary points, resolution, proof quality

1 Introduction

Resolution-based SAT-solvers [4,9,13,16,18,19,20] have achieved great success in numerous applications. Importantly, in many cases, the quality of resolution proofs generated by a SAT-solver matters as much as its performance. For example, in bounded model checking [5], a resolution proof is used to identify the part of the circuit relevant to a particular property [7]. In interpolation based model checking [15], the size of interpolant strongly depends on that of the resolution proof it is extracted from. In [10], we showed that high-quality tests can be obtained from a resolution proof. The number of these tests is proportional to the number of resolutions in the proof. So proof reduction means getting a more compact test set.

In this paper, we study the relation between resolution and boundary points [12]. The motivation here is as follows. To show that a CNF formula is unsatisfiable it is sufficient to eliminate all its boundary points. Resolution can be viewed as a method of boundary point elimination. Relating resolution proofs with boundary point elimination, may lead to a) better understanding of resolution proofs; b) identifying proof redundancies; c) designing SAT-solvers that generate smaller proofs.

Given a CNF formula $F(x_1,..,x_n)$, a non-satisfying complete assignment $\boldsymbol{p}$ is called a $lit(x_i)$-*boundary* point, if it falsifies only the clauses of F that have literal $lit(x_i)$. The name is due to the fact that for satisfiable formulas the set of such points contain the boundary between satisfying and unsatisfying assignments. If F is unsatisfiable, for

every $lit(x_i)$-boundary point $\boldsymbol{p}$ there is a resolvent of two clauses of F on variable x_i that eliminates $\boldsymbol{p}$ (i.e. after adding such a resolvent to F, $\boldsymbol{p}$ is not a boundary point anymore). On the contrary, for a non-empty satisfiable formula F, there is always a boundary point that can not be eliminated by adding a clause implied by F.

To prove that a CNF formula F is unsatisfiable it is sufficient to eliminate all its boundary points. In the resolution proof system [3], one reaches this goal by adding to F resolvents. If formula F has a $lit(x_i)$-boundary point, a resolution proof has to have a resolution operation on variable x_i. The resolvents of a resolution proof eventually eliminate all boundary points. (A formula with an empty clause does not have any boundary points.) However, as we show experimentally many resolution operations of proofs generated by a conflict driven SAT-solver do not eliminate any boundary points (non-boundary resolutions). We use the ratio of boundary and non-boundary resolutions of a proof as a redundancy measure called SBR metric (Share of Boundary Resolutions).

To check if there is a relation between the value of SBR metric and proof quality we computed the values of this metric for two kinds of proofs for equivalence checking formulas. (These formulas describe equivalence checking of two copies of a combinational circuit.) Namely, we considered short proofs of linear size particularly tailored for equivalence checking formulas and much longer proofs generated by a SAT-solver with conflict driven learning. We showed experimentally that the share of boundary resolution operations in high-quality specialized proofs is much greater than in proofs generated by the SAT-solver. This implies that the SAT-solver's proofs may have some redundancies.

The contribution of this paper is threefold. First, we show that one can view resolution as elimination of boundary points. Second, we introduce the SBR metric that can be potentially used as a measure of proof redundancy. Third, we give some experimental results about the relation between SBR -metric and proof quality.

This paper is structured as follows. Section 2 introduces main definitions. Some properties of boundary points are given in Section 3. Section 4 views a resolution proof as a process of boundary point elimination. A class of equivalence checking formulas and their short resolution proofs are described in Section 5. Experimental results are given in Section 6. Some relevant background is recalled in Section 7. Conclusions and directions for future research are listed in Section 8.

2 Basic Definitions

Definition 1. A *literal* of a Boolean variable x_i (denoted as $lit(x_i)$) is a Boolean function of x_i. The identity function (denoted just as x_i) is called the *positive literal* of x_i. The negation function (denoted as $\sim x_i$) is called the *negative literal* of x_i.

Definition 2. A *clause* is the disjunction of literals where no two (or more) literals of the same variable can appear. A *CNF formula* is the conjunction of clauses. We will also view a CNF formula as a *set of clauses*.

Definition 3. Given a CNF formula $F(x_1,..,x_n)$, a *complete assignment* (also called a *point*) is a mapping $\{x_1,..,x_n\} \rightarrow \{0,1\}$. Given a complete assignment $\boldsymbol{p}$ and a clause C, denote by $C(\boldsymbol{p})$ the value of C when its variables are assigned by $\boldsymbol{p}$. A clause C is *satisfied* (respectively *falsified*) by a complete assignment $\boldsymbol{p}$, if $C(\boldsymbol{p}) = 1$ (respectively $C(\boldsymbol{p}) = 0$).

Definition 4. Given a CNF formula F, a *satisfying assignment* $\boldsymbol{p}$ is a complete assignment satisfying every clause of F. *The satisfiability problem* (SAT) is to find a satisfying assignment for F or to prove that such an assignment does not exist.

Definition 5. Let F be a CNF formula and $\boldsymbol{p}$ be a complete assignment. Denote by $Unsat(\boldsymbol{p},F)$ the set of all clauses of F falsified by $\boldsymbol{p}$.

Definition 6. Given a CNF formula $F(x_1,..,x_n)$, a complete assignment $\boldsymbol{p}$ is called a $lit(x_i)$-*boundary point* if $Unsat(\boldsymbol{p},F)$ is not empty and every clause of $Unsat(\boldsymbol{p},F)$ contains literal $lit(x_i)$.

Example 1. Let F consist of 5 clauses: $C_1= x_2$, $C_2= \sim x_2 \vee x_3$, $C_3 =\sim x_1 \vee \sim x_3$, $C_4 = x_1 \vee \sim x_3$, $C_5 = \sim x_2 \vee \sim x_3$. Complete assignment $\boldsymbol{p_1}=(x_1=0,x_2=0,x_3=1)$ falsifies clauses C_1,C_4. So $Unsat(\boldsymbol{p}_1,F)=\{C_1,C_4\}$. There is no literal shared by the clauses of $Unsat(\boldsymbol{p}_1,F)$. Hence $\boldsymbol{p_1}$ is not a boundary point. On the other hand, $\boldsymbol{p_2}=(x_1=0,x_2=1,x_3=1)$ falsifies the clauses C_4,C_5 sharing literal $\sim x_3$. So $\boldsymbol{p_2}$ is a $\sim x_3$-boundary point

3. Basic Properties of Boundary Points

In this section we give some properties of boundary points.

3.1. Basic Propositions

In this subsection, we prove the following propositions. The set of boundary points contains the boundary between satisfying and unsatisfying assignments (Proposition 1). A CNF formula without boundary points is unsatisfiable (Proposition 2). Boundary points come in pairs (Proposition 3).

Definition 7. Denote by $Bnd\_pnts(F)$ the set of all boundary points of a CNF formula F. We assume that a $lit(x_i)$-boundary point $\boldsymbol{p}$ is specified in $Bnd\_pnts(F)$ as the pair $(lit(x_i), \boldsymbol{p})$. So the same point $\boldsymbol{p}$ may be present in $Bnd\_pnts(F)$ more than once (e.g. if $\boldsymbol{p}$ is a $lit(x_i)$-boundary point and a $lit(x_j)$-boundary point at the same time).

Proposition 1. Let F be a satisfiable formula whose set of clauses is not empty. Let $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ be two complete assignments such that a) $F(\boldsymbol{p_1})=0$, $F(\boldsymbol{p_2})=1$; b) $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ are different only in the value of variable x_i. Then $\boldsymbol{p_1}$ is a $lit(x_i)$-boundary point.

Proof. Assume the contrary i.e. $Unsat(\boldsymbol{p_1},F)$ contains a clause C of F that does not have variable x_i. Then $\boldsymbol{p_2}$ falsifies C too and so $\boldsymbol{p_2}$ cannot be a satisfying assignment. A contradiction.

Proposition 1 means that the set $Bnd\_pnts(F)$ contains the boundary between satisfying and unsatisfying assignments of a satisfiable CNF formula F.

Proposition 2. Let F be a CNF formula that has at least one clause. If $Bnd\_pnts(F) = \varnothing$, then F is unsatisfiable.

Proof. Assume the contrary i.e. $Bnd\_pnts(F) = \varnothing$ and F is satisfiable. Since F is not empty, one can always find two points $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ such that $F(\boldsymbol{p_1})=0$ and $F(\boldsymbol{p_2})=1$ and that are different only in the value of one variable x_i of F. Then according to Proposition 1, $\boldsymbol{p_1}$ is a $lit(x_i)$-boundary point. A contradiction.

Proposition 3. Let $\boldsymbol{p_1}$ be a $lit(x_i)$-boundary point for a CNF formula F. Let $\boldsymbol{p_2}$ be the point obtained from $\boldsymbol{p_1}$ by changing the value of x_i. Then $\boldsymbol{p_2}$ is either a satisfying assignment or a $\sim lit(x_i)$-boundary point.

Proof. Reformulating the proposition, one needs to show that $Unsat(\boldsymbol{p_2},F)$ is either empty or contains only clauses with $\sim lit(x_i)$. Assume that contrary, i.e. $Unsat(\boldsymbol{p_2},F)$ contains a clause C with no literal of x_i. (All clauses with $lit(x_i)$ are satisfied by $\boldsymbol{p_2}$.) Then C is falsified by $\boldsymbol{p_1}$ too and so $\boldsymbol{p_1}$ is not a $lit(x_i)$-boundary point. A contradiction.

Definition 8. Proposition 3 means that for unsatisfiable formulas every x_i-boundary point has the corresponding $\sim x_i$-boundary point (and vice versa). We will call such a pair of points *twin boundary points in variable* x_i.

Example 2. The point $\boldsymbol{p_2}= (x_1=0,x_2=1,x_3=1)$ of Example 1 is an $\sim x_3$-boundary point. The point $\boldsymbol{p_3}=(x_1=0,x_2=1, x_3=0)$ obtained from $\boldsymbol{p_2}$ by flipping the value of x_3 falsifies only clause $C_2= \sim x_2 \vee x_3$. So $\boldsymbol{p_3}$ is an x_3-boundary point.

3.2. Elimination of Boundary Points by Adding Resolvents

In this subsection, we prove the following propositions. Clauses of a CNF formula F falsified by twin boundary points can be resolved (Proposition 4). Adding such a resolvent to F eliminates these boundary points (Proposition 5). Adding the resolvents of a resolution proof eventually eliminates all boundary points (Proposition 6). A $lit(x_i)$-boundary point can be eliminated only by a resolution on variable x_i (Proposition 7). If formula F has a $lit(x_i)$-boundary point, any resolution proof that F is unsatisfiable has a resolution on variable x_i. (Proposition 8).

Definition 9. Let C_1 and C_2 be two clauses that have opposite literals of variable x_i (and no opposite literals of any other variable). The *resolvent* C of C_1 and C_2 is the clause consisting of all the literals of C_1 and C_2 but the literals of x_i. The clause C is said to be obtained by *a resolution operation* on variable x_i. C_1 and C_2 are called the *parent clauses*.

Proposition 4. Let $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ be twin boundary points of a CNF formula F in variable x_i. Let C_1 and C_2 be two arbitrary clauses falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ respectively. Then a) C_1 , C_2 can be resolved on variable x_i; b) $C(\boldsymbol{p_1}) = 0$, $C(\boldsymbol{p_2}) = 0$ where C is the resolvent of C_1 and C_2.
Proof. Since $C_1(\boldsymbol{p_1})=0$, $C_2(\boldsymbol{p_2})=0$ and $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ are twin boundary points in x_i, C_1 and C_2 have opposite literals of variable x_i. Since $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ are different only in the value of x_i, clauses C_1 and C_2 can not contain opposite literals of a variable other than x_i. (Otherwise, $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ had to be different in values of at least 2 variables.) Since $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ are different only in the value of x_i, they both set to 0 all the literals of C_1 and C_2 but literals of x_i. So the resolvent C of C_1 and C_2 is falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$.

Example 3. Points $\boldsymbol{p_2}$= (x_1=0, x_2=1, x_3=1) and $\boldsymbol{p_3}$=(x_1=0, x_2=1, x_3=0) from Examples 1 and 2 are twin boundary points in variable x_3. $Unsat(\boldsymbol{p_2}, F)=\{C_4,C_5\}$ and $Unsat(\boldsymbol{p_3}, F) =\{C_2\}$. For example, $C_4 = x_1 \vee \sim x_3$, can be resolved with $C_2=\sim x_2 \vee x_3$ on variable x_3. Their resolvent $C= x_1 \vee \sim x_2$ is falsified by both $\boldsymbol{p_2}$ and $\boldsymbol{p_3}$.

Proposition 5. Let $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ be twin boundary points in variable x_i and C_1 and C_2 be clauses falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ respectively Then adding the resolvent C of C_1 and C_2 to F *eliminates* the boundary points $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$. That is pairs $(x_i,\boldsymbol{p_1})$ and $(\sim x_i,\boldsymbol{p_2})$ are not in the set $Bnd\_pnts(F \wedge C)$ (here we assume that $\boldsymbol{p_1}$ is an x_i-boundary point and $\boldsymbol{p_2}$ is a $\sim x_i$-boundary point of F).
Proof. According to Proposition 4, any clauses C_1 and C_2 falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ respectively can be resolved in x_i and $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ falsify the resolvent C of C_1 and C_2. Since clause C does not have a literal of x_i , $\boldsymbol{p_1}$ is not an x_i-boundary point and $\boldsymbol{p_2}$ is not a $\sim x_i$-boundary point of $F \wedge C$.

Proposition 6. If a CNF formula F contains an empty clause, then $Bnd\_pnts(F) = \varnothing$.
Proof. For any complete assignment $\boldsymbol{p}$, the set $Unsat(\boldsymbol{p},F)$ contains the empty clause of F. So $\boldsymbol{p}$ can not be a $lit(x_i)$-boundary point.

Proposition 6 works only in one direction, i.e. if $Bnd\_pnts(F) = \varnothing$, it does not mean that F contains an empty clause. Proposition 6 implies that, given an unsatisfiable formula F for which $Bnd\_pnts(F)$ is not empty , the resovlents of any resolution proof of unsatisfiability of F eventually eliminate all the boundary points.

Proposition 7. Let F be a CNF formula and $\boldsymbol{p}$ be a $lit(x_i)$-boundary point of F. Let C be the resolvent of clauses C_1 and C_2 of F that eliminates $\boldsymbol{p}$ (i.e. $(lit(x_i),\boldsymbol{p})$ is not in $Bnd\_pnts(F \wedge C)$). Then C is obtained by resolution on variable x_i. In other words, a $lit(x_i)$-boundary point can be eliminated only by adding to F a resolvent on variable x_i.
Proof. Assume the contrary i.e. adding C to F eliminates $\boldsymbol{p}$ and C is obtained by resolving C_1 and C_2 on variable x_j, $j \neq i$. Since C eliminates $\boldsymbol{p}$ as a $lit(x_i)$-boundary point, it is falsified by $\boldsymbol{p}$ and does not contain $lit(x_i)$. This means that neither C_1 nor C_2 contain variable x_i. Since C is falsified by $\boldsymbol{p}$, one of the parent clauses, say clause C_1, is falsified by $\boldsymbol{p}$ too. Since C_1 does not contain literal $lit(x_i)$, $\boldsymbol{p}$ is not a $lit(x_i)$-boundary point of F. A contradiction.

Proposition 8. Let $\boldsymbol{p}$ be a $lit(x_i)$-boundary point of a CNF formula F. Then any resolution derivation of an empty clause from F has to contain a resolution operation on variable x_i.
Proof. According to Proposition 6, every boundary point of F is eventually eliminated in a resolution proof. According to Proposition 7, a $lit(x_i)$-boundary point can be eliminated only by adding to F a clause produced by resolution on variable x_i.

3.3 Boundary Points and Clause Redundancy

In this subsection, we show that redundant clauses (e.g. conflict clauses) can be used in resolutions as parent clauses to eliminate boundary points.
Definition 10. A clause C of a CNF formula F is called *redundant* if $F \setminus \{C\} \rightarrow C$.
Proposition 9. Let C be a clause of a CNF formula F. Let $lit(x_i)$ be a literal of C. Suppose that no $lit(x_i)$-boundary point of F falsifies clause C. Then C is redundant.
Proof. Assume the contrary, i.e. C is not redundant. Then there is an assignment $\boldsymbol{p}$ such that C is falsified and all the other clauses of F are satisfied. Then $\boldsymbol{p}$ is a $lit(x_i)$-boundary point. A contradiction.

Importantly, Proposition 9 works only in one direction. That is the fact that a clause C is redundant in F does not mean that no boundary point of F falsifies C. Let CNF formula $F(x_1,x_2)$ consist of four clauses: $\sim x_1$, x_1, $x_1 \vee \sim x_2$, $x_1 \vee x_2$. Although the clause x_1 is redundant in F, $\boldsymbol{p} = (x_1=0, x_2=0)$ is an x_1-boundary point falsifying x_1 (and $x_1 \vee x_2$). The resolvent of clauses x_1 and $\sim x_1$ eliminates $\boldsymbol{p}$ as a boundary point.

4. Resolution Proofs and Boundary Points

In this section, we view construction of a resolution proof as a process of boundary point elimination and give a metric for measuring potential proof redundancy.

4.1 Resolution Proof as Boundary Point Elimination

First, we define the notion of a resolution proof [3] and a boundary resolution.
Definition 11. Let F be an unsatisfiable formula. Let $R_1,\ldots,R_k$ be a set of clauses such that a) each clause R_i is obtained by resolution operation where a parent clause is either a clause of F or the resolvent of a previous resolution operation; b) clauses R_i are numbered in the order they are derived; c) R_k is an empty clause; Then the set of resolutions that produced the resolvents $R_1,..,R_k$ is called a *resolution proof*. We assume that this proof is *irredundant* i.e. removal of any non-empty subset of these k resolvents breaks condition a).
Definition 12. Let $R_1,..,R_k$ be the set of resolvents forming a resolution proof that a CNF formula F is unsatisfiable. Denote by F_i the CNF formula that is equal to F for i=1 and to $F \cup \{R_1,..,R_{i-1}\}$ for $i = 2,\ldots,k$.. In other words, F_i consists of the initial clauses of F and first i-1 resolvents. We will say that the i-th resolution (i.e. one that produces resolvent R_i) is *non-boundary* if $Bnd\_pnts(F_i) = Bnd\_pnts(F_{i+1})$. Otherwise

(i.e. if $Bnd\_pnts(F_i) \supset Bnd\_pnts(F_{i+1})$, because adding a clause can not *create* a boundary point) , *i*-th resolution is called *boundary*. So a resolution operation is boundary if adding R_i to F_i eliminates a boundary point.

In the previous section, we showed that eventually all the boundary points of a CNF formula F are removed by resolvents. Importantly, a $lit(x_i)$-boundary point mandates a resolution on variable x_i. Besides, as we showed in Section 3.3. even redundant clauses can be used to produce new resolvents eliminating boundary points. It is important because, all clauses derived by resolution (in particular conflict clauses generated by modern SAT-solvers) are redundant. So the derived clauses are as good for boundary point elimination as the original ones.

A natural question arises about the role of non-boundary resolutions. When answering this question it makes sense to separate redundant and irredundant formulas. (A CNF formula F is said to be *irredundant* if no clause of F is redundant, see Definition 10) For a redundant formula, one may have to use non-boundary resolutions. (In particular, a heavily redundant formula may not have boundary points at all. In such a case, every resolution operation is non-boundary.) For irredundant formulas the situation is different.

Proposition 10. Let F be an irredundant formula of m clauses. Then F has at least d boundary points where d is the total number of literals in the clauses of F.

Proof. Let C be a clause of F. Then there is a complete assignment $\boldsymbol{p}$ falsifying C and satisfying the clauses of $F \setminus \{C\}$. This assignment is a $lit(x_i)$-boundary point where $lit(x_i)$ is a literal of C.

As far as irredundant formulas are concerned some natural questions arise. Given an irredundant formula, can one always build a resolution proof using only boundary resolutions? If so, what is the relation between such a limited proof system and general resolution? Can general resolution produce proofs shorter than in this limited proof system?

We do not answer the questions above theoretically. Instead we introduce a metric measuring the Share of Boundary Resolutions (SBR-metric) to check if proof quality depends on the value of SBR metric. (This value is computed as the percent of boundary resolutions of a proof). In Section 6, we give some experimental evidence that the value of SBR metric for short specialized proofs for equivalence checking formulas is much higher than for proofs generated by a SAT-solver with conflict driven learning.

4.2 SBR metric and Proof Redundancy

Although we do not know the nature of non-boundary resolutions we still can argue that the low value of SBR metric may mean some proof redundancy. The reason is that such a redundancy indeed leads to the appearance of non-boundary resolutions. We give two examples of that below.

Not sharing resolutions of conflict clause derivations. In a typical SAT-solver with conflict-driven learning, the only type of clauses learned are conflict clauses (this applies to the Sat-solver DMRP-SAT that we used in our experiments). On the other hand, a conflict clause is the result of many resolution operations. The intermediate resolvents of these operations are usually discarded. It may be the case though that for

two conflict clauses C_1 and C_2 resolution proofs of their derivation share some intermediate resolvents. When the value of SBR metric is computed, *all resolvents* are taken into consideration. This lack of sharing resolutions used in conflict clause derivations would lead to appearance of non-boundary resolutions (a resolution deriving a clause produced earlier by some other resolution can not eliminate a boundary point).

Appearance of unsatisfiable subformulas. Even if the initial unsatisfiable CNF formula F to be solved is irredundant, an unsatisfiable subformula of F inevitably appears due to the addition of new clauses. (In particular, one can view an empty clause as an unsatisfiable subformula of the final CNF formula.) Let F_1 be an unsatisfiable subformula of F. Let $Vars(F)$ and $Vars(F_1)$ denote the sets of variables of F and F_1. Then no $lit(x_i)$-boundary point exists if x_i is in $Vars(F) \setminus Vars(F_1)$. (The set of clauses falsified by $\boldsymbol{p}$ contains at least one clause of F_1 and such a clause does not have a literal of x_i.) So any resolution on a variable of $Vars(F) \setminus Vars(F_1)$ is non-boundary.

The appearance of unsatisfiable subformulas may lead to increasing the share of non-boundary resolutions in the final proof. For example, instead of deriving an empty clause from the clauses of F_1, the SAT-solver may first derive some clauses having variables of $Vars(F_1)$ from clauses of $F \setminus F_1$. It is possible since clauses of $F \setminus F_1$ may contain variables of $Vars(F_1)$. When deriving such clauses the SAT-solver may use (non-boundary) resolutions on variables of $Vars(F) \setminus Vars(F_1)$, which leads to redundancy of the final proof.

5 Equivalence Checking Formulas

In this section, we introduce the formulas we use in the experimental part of this paper. These are the formulas that describe equivalence checking of two copies of a combinational circuit. In Subsection 5.1 we show how such formulas are constructed. In Subsection 5.2. we describe how short proofs of unsatisfiability particularly tailored for equivalence checking formulas can be built.

5.1. Building Equivalence Checking Formulas

Let N and N^* be two single-output combinational circuits. To check their functional equivalence one constructs a circuit called a *miter* (we denote it as $Miter(N,N^*)$). It is a circuit that is satisfiable (i.e. its output can be set to 1) if and only if N and N^* are not functionally equivalent. (N and N^* are not functionally equivalent if there is an input assignment for which N and N^* produce different output values.) Then a CNF formula F_{Miter} is generated that is satisfiable if and only if $Miter(N,N^*)$ is satisfiable. In our experiments we built a miter of two *identical copies* of the same circuit. In such a case $Miter(N,N^*)$ is always unsatisfiable and so is CNF formula F_{Miter}.

Example 4. Figure 1 shows the miter of copies N and N^* of the same circuit. Here g_1, g_1^* are OR gates, g_2, g_2^* are AND gates and h is an XOR gate (implementing modulo-2 sum). Note that N and N^* have the same set of input variables but different

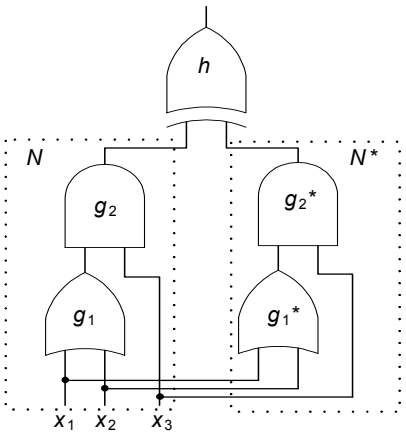

Figure 1. Circuit *Miter*(N, N*)

intermediate and output variables. Since $g_2 \oplus g_2$* evaluates to 1 if and only if $g_2 \neq g_2$*, and N and N* are functionally equivalent, the circuit *Miter*(N,N*) evaluates only to 0.

A CNF formula F_{Miter} whose satisfiability is equivalent to that of *Miter*(N,N*) is formed as $F_{\text{N}} \wedge F_{\text{N*}} \wedge F_{\text{xor}} \wedge h$. Here F_{N} and $F_{\text{N*}}$ are formulas specifying the functionality of N and N* respectively. The formula F_{xor} specifies the functionality of the XOR gate h and the unit clause h forces the output of *Miter*(N,N*) to be set to 1.

Since, in our case, the miter evaluates only to 0, the formula F_{M} is unsatisfiable.

Formulas F_{N} and $F_{\text{N*}}$ are formed as the conjunction of subformulas describing the gates of N and N*. For instance, $F_{\text{N}} = F_{\text{g1}} \wedge F_{\text{g2}}$ where, for example, $F_{\text{g1}} = (x_1 \vee x_2 \vee \sim g_1) \wedge (\sim x_1 \vee g_1) \wedge (\sim x_2 \vee g_1)$ specifies the functionality of an OR gate. Each clause of F_{g1} rules out some inconsistent assignments to the variables of gate g_1. For example, the clause $(x_1 \vee x_2 \vee \sim g_1)$ rules out the assignment x_1=0, x_2=0, g_1=1.

5.2. Short proofs for equivalence checking formulas

For a CNF formula F_{Miter} describing equivalence checking of two copies N,N* of the same circuit, there is a short resolution proof that F_{Miter} is unsatisfiable. This proof is linear in the number of gates in N and N*. The idea of this proof is as follows. For every pair g_i, g_i* of the corresponding gates of N and N* , the clauses of CNF formula *Eq*(g_i,g_i*) specifying the equivalence of variables g_i, g_i* are derived. Here *Eq*(g_i,g_i*) is equal to $(\sim g_i \vee g_i^*) \wedge (g_i \vee \sim g_i^*)$. These clauses are derived according to topological levels of gates g_i,g_i* in *Miter*(N,N*). (The topological level of a gate g_i is the length of the longest path from an input to gate g_i measured in the number of gates on this path.) First, clauses of *Eq*(g_i,g_i*) are derived for all pairs of gates g_i,g_i* of topological level 1. Then using previously derived *Eq*(g_i,g_i*), same clauses are derived for the pairs of gates g_j,g_j* of topological level 2 and so on.

Eventually, the clauses of *Eq*(g_s,g_s*) relating the output variables g_s, g_s* of N and N* are derived. Resolving the clauses of *Eq*(g_s,g_s*) and the clauses describing the XOR gate, the clause $\sim h$ is derived. Resolution of $\sim h$ and the unit clause h of F_{Miter} produces an empty clause.

Example 5. Let us explain the construction of the proof using the CNF F_{Miter} from Example 4. Gates g_1,g_1* have topological level 1 in *Miter*(N,N*). So first, the clauses of *Eq*(g_1,g_1*) are obtained. They are derived from the CNF formulas F_{g1} and $F_{\text{g1*}}$ describing gates g_1 and g_1*. That the clauses of *Eq*(g_1,g_1*) can be derived from $F_{\text{g1}} \wedge F_{\text{g1}}$* just follows from the completeness of resolution and the fact that *Eq*(g_1,g_1*) is

implied by the CNF formula $F_{g1} \wedge F_{g1*}$. (This implication is due to the fact that F_{g1} and F_{g1}* describe two functionally equivalent gates with the same set of input variables). More specifically, the clause $\sim g_1 \vee g_1$* is obtained by resolving the clause $x_1 \vee x_2 \vee \sim g_1$ of F_{g1} with the clause $\sim x_1 \vee g_{1*}$ of F_{g1}* and then resolving the resolvent with the clause $\sim x_2 \vee g_1$* of F_{g1*}. In a similar manner, the clause $g_1 \vee \sim g_1$* is derived by resolving the clause $x_1 \vee x_2 \vee \sim g_1$* of F_{g1*} with clauses $\sim x_1 \vee g_1$ and $\sim x_2 \vee g_1$ of F_{g1}.

Then the clauses of $Eq(g_2,g_2$*) are derived (gates g_2,g_2* have topological level 2). $Eq(g_2,g_2$*) is implied by $F_{g2} \wedge F_{g2}$* $\wedge Eq(g_1,g_1$*) . Indeed, g_2 and g_2* are functionally equivalent gates that have the same input variable x_3. The other input variables g_1 and g_1* are identical too due to the presence of $Eq(g_1,g_1$*). So the clauses of $Eq(g_2,g_2$*) can be derived from clauses of $F_{g2} \wedge F_{g2}$* $\wedge Eq(g_1,g_1$*) by resolution. Then the clause $\sim h$ is derived as implied by $F_{xor} \wedge Eq(g_2,g_{2*})$ (an XOR gate produces output 0 when its input variables have equal values). Resolution of h and $\sim h$ produces an empty clause.

6 Experimental Results

The goal of experiments was to compare the values of SBR metric for two kinds of proofs of different quality. In the experiments we used formulas describing the equivalence checking of two copies of combinational circuits. The reason for using such formulas is that one can easily generate high-quality specialized proofs of their unsatisfiability (see Section 5). In the experiments we compared these short proofs with ones generated by the SAT-solver DMRP-SAT [11].

We performed the experiments on a Pentium-4 PC with clock frequency of 3 GHz. The CNF formulas and proofs of both types we used in the experiments can be downloaded from http://eigold.tripod.com/exper_sat_2009.tar.gz. The time limit in all experiments was set to 1 hour.

Given a resolution proof R of k resolutions that a CNF formula F is unsatisfiable, computing the value of SBR metric of R reduces to k SAT-checks. In our experiments these SAT-checks were performed by a version of DMRP-SAT. Let F_i be the CNF formula $F \cup \{R_1,..,R_{i-1}\}$ where $R_1,..,R_{i-1}$ are the resolvents generated in the first i-1 resolutions. Let C_1 and C_2 be the clauses of F_i that are the parent clauses of the resolvent R_i. Let C_1 and C_2 be resolved on variable x_j. Assume that C_1 contains the positive literal of x_j. Checking if i-the resolution eliminates an x_j-boundary point can be performed as follows. First, all the clauses with a literal of x_j are removed from F_i. Then one adds to F_i the unit clauses that force the assignments setting all the literals of C_1 and all the literals of C_2 but the literal $\sim x_j$ to 0. Denote the resulting CNF formula by G_i.

If G_i is satisfiable then there is a complete assignment $\boldsymbol{p}$ that is falsified by C_1 and maybe by some other clauses with literal x_j. So $\boldsymbol{p}$ is an x_j-boundary point of F_i. Since $\boldsymbol{p}$ falsifies all the literals of C_2 but $\sim x_j$, it is falsified by the resolvent of C_1 and C_2. So the satisfiability of G_i means that i-th resolution eliminates $\boldsymbol{p}$ and so this resolution is boundary. If G_i is unsatisfiable, then no x_j-boundary point is eliminated by i-th resolution. All boundary points come in pairs (see Proposition 3). So no $\sim x_j$-boundary

point is eliminated by *i*-th resolution either. Hence the unsatisfiability of G_i means that the *i*-th resolution is non-boundary.

Table 1. shows the value of SBR metric for the short specialized proofs. The first column gives the name of the benchmark circuit whose self-equivalence is described by the corresponding CNF formula. The size of this CNF formula is given in the second and third columns. The fourth column of Table 1 gives the size of the proof (in the number of resolutions). The fifth column shows the value of SBR metric and the last column of Table 1 gives the run time of computing this value. These run times can be significantly improved if one uses a faster SAT-solver and tunes it to solving the particular problem of computing the value of SBR metric. (For example, one can try to share conflict clauses learned in different SAT-checks.)

Looking at Table 1 one can conclude that the specialized proofs have a very high value of SBR-metric (almost every resolution operation eliminates a boundary point). The only exception is the *dalu* formula (84%). The fact that the value of SBR metric for *dalu* and some other formulas is different from 100% is probably due to the fact that the corresponding circuits have some redundancies. Such redundancies would lead to redundancy of CNF formulas specifying the corresponding miters, which would lower the value of SBR metric.

Table 1. Computing value of SBR metric for short specialized proofs

Name	#vars	#clauses	#resolu-tions	#boundary resol. %	run time (s)
c432	480	1,333	1,569	95	10.2
9symml	480	1,413	1,436	100	4.5
c880	807	2,264	2,469	100	24.6
alu4	2,369	7,066	8,229	96	270
c3540	2,625	7,746	9,241	97	1,743
x1	4,381	12,991	12,885	97	2,890
dalu	4,714	13,916	15,593	84	2,202
c6288	4,771	14,278	17,925	100	2,462

Table 2. Indirect comparison of proofs generated by Minisat and DMRP-SAT

Name	Minisat (v1.14)		DMRP-SAT	
	#confl. clauses	run time (sec.)	#confl. clauses	run time (sec.)
c432	809	0.05	374	0.08
9symml	259	0.03	275	0.08
c880	6,000	0.29	1,309	0.27
alu4	2,355	0.33	1,320	1.20
c3540	22,214	2.65	10,021	7.75
x1	3,294	0.45	765	1.06
dalu	4,402	0.94	3,351	4.39
c6288	*	> 3,600	*	> 3,600

Table 2 is meant to show that the proofs generated by DMRP-SAT have high quality (for a SAT-solver with conflict driven learning). Here we compare the proofs generated by Minisat (version 1.14) and DMRP-SAT in the number of conflict clauses. (Minisat does not generate proofs so we could not compare the actual proof sizes). Although this is an indirect comparison, it gives an idea of the quality of proofs generated by DMRP-SAT. For self-equivalence of a 16-bit multiplier (instance C6288), neither SAT-solver finished the formula within the time limit.

The values of SBR-metric for the proofs generated by DMRP-SAT are given in Table 3. The second column gives the size of resolution proofs generated by DMRP-SAT. When computing the size of these proofs we removed the obvious redundancies. Namely, the derivation of the conflict clauses that did not contribute to the derivation of an empty clause was ignored. The third column shows the value of SBR metric and the last column gives the run time of computing this value. In the case the computation did not finish within the time limit, the number in parentheses shows the percent of the resolution operations processed before the computation terminated.

Table 3 shows that the size of the proofs generated by DMRP-SAT is much larger than that of specialized proofs (Table 1, fourth column). The only exception is the instance *x1* where the two kinds of proofs have comparable size. Interestingly, *x1* is the instance with the highest value of SBR metric (88%) among DMRP-SAT proofs. For the rest of the formulas the value of SBR metric is much smaller. Importantly, the value of SBR metric we give for the formulas for which computation was terminated due to exceeding the time limit is higher than it should be. Typically, the later a resolution occurs in a resolution proof, the more likely it is that this resolution is non-boundary. So the early termination of SBR metric computation ignored resolutions with the highest chances to be non-boundary.

Table 3. Computing value of SBR metric for proofs generated by DMRP-SAT

Name	#resolu-tions	#boundary resol. %	run time (s) (% of proof finished)
c432	7,655	37	48
9symml	5,317	57	23
c880	24,478	37	503
alu4	98,600	36	>3,600 (49%)
c3540	347,264	54	>3,600 (6%)
x1	16,259	88	864
dalu	119,553	40	>3,600 (33.4%)

Summing up, one can conclude that for the formulas we considered in experiments, the proofs of poorer quality (generated by DMRP-SAT) have lower values of SBR metric. It remains to be seen though whether it means that these proofs are redundant in some way and so can be optimized.

7 Some Background

The notion of boundary points was introduced in [12] where they were called essential points. (We decided to switch to the term "boundary point" as more precise.) Boundary points were used in [12] to help a SAT-solver prune the search space. If the subspace x_i=0 does not contain a satisfying assignment or an x_i-boundary point, one can claim that the symmetric subspace x_i=1 can not contain a satisfying assignment either (due to Proposition 3). The same idea of search pruning was independently described in [17] and implemented in the SAT-solver Jerusat. The ideas of search pruning introduced in [12] were further developed in [2].

In [14] we formulate two proof systems meant for exploring the 1-neighborhood of clauses of the formula to be solved. The union of the 1-neighborhhoods of these clauses is essentially a superset approximation of the set of boundary points. To prove that a formula is unsatisfiable it is sufficient to eliminate all boundary points (Proposition 2). The proof systems of [14] show that one can eliminate all boundary points without generation of an empty clause. So resolution can be viewed as a special case of boundary point elimination.

The results of the present paper can also be considered as an approach to improving automatizability of resolution [6]. General resolution is most likely non-automatizable [1]. This means that finding short proofs can not be done efficiently in general resolution. A natural way to mitigate this problem is to look for restricted versions of general resolution that are "more automatizable" i.e. that facilitate finding good proofs. Intuitively, boundary resolutions is a tiny part of the set of all possible resolutions. So the restriction of resolutions to boundary ones can be viewed as a way to make it easier to find good proofs (assuming that such a restriction does not kill *all* high-quality proofs.)

8 Conclusions and Directions for Future Research

We show that a resolution proof can be viewed as the process of boundary point elimination. We introduce the SBR metric that is the percent of resolutions of the proof that eliminate boundary points (boundary resolutions). This metric can be used for estimating proof redundancy. We experimentally show that short specialized proofs for equivalence checking formulas have high values of SBR metric. On the other hand, values of this metric for proofs generated by a SAT-solver with conflict driven learning are low. This implies that the proofs generated by this SAT-solver may have some redundancies.

The idea of treating resolution as boundary proof elimination has many interesting directions for research. Here are a few of them.

1) Testing further the conjecture that SBR metric relates to proof redundancy.
2) Proving completeness of resolution performing only boundary resolutions for irredundant CNF formulas
3) Answering the question about the nature of non-boundary resolutions. In particular, is the resolution proof system where only boundary resolutions are allowed less powerful than general resolution?

4) Designing SAT-solvers that generate proofs with high value of SBR metric.

References

1. Alekhnovich, M., Razborov, A.: Resolution Is Not Automatizable Unless W[P] Is Tractable. SIAM J. Comput., vol. 38, no.4 , pp. 1347--1363 (2008).
2. Babic, D., Bihgham, J.,Hu A.: Efficient SAT solving: beyond supercubes. In: DAC-2005, pp. 744--459.
3. Bachmair, L.,Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds). Handbook of automated reasoning. Chap. 2, vol. 1,pp. 19--99. Elsevier Science Publ. (2001).
4. Bierre, A.: PicoSAT essentials. JSAT 4, 75--97 (2008).
5. Bierre, A.,Cimatti, A.,Clarke, Fujita M., Zhu Y.: Symbolic model checking using SAT procedures instead of BDDs. In: DAC-1999, pp. 317--320.
6. Bonet, M. L., Pitassi, T., and Raz, R: On Interpolation and Automatization for Frege Systems. SIAM J. Comput. Vol. 29, no. 6 pp. 1939--1967 (2000).
7. Clarke, E., Gupta, A.,Strichman, O.: SAT-based counterexample-guided abstraction refinement, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, No. 7, pp. 1113--1123 (2004).
8. Davis, M., Longemann, G., Loveland, D.: A Machine program for theorem proving. Communications of the ACM 5, 394--397 (1962).
9. Een, N., Sorensson, N. : An extensible SAT-solver. In: SAT-2003, LNCS, vol. 2919, pp.503--518.
10. Goldberg, E.: On Bridging Simulation and Formal Verification. In: VMCAI-2008, pp.127--141.
11. Goldberg, E.: A Decision-Making Procedure for Resolution-Based SAT-Solvers. In: SAT-2008, pp. 119--132.
12. Goldberg, E., Prasad, M.,Brayton, R.: Using Problem Symmetry in Search Based Satisfiability Algorithms. In: DATE-2002, pp. 134--141.
13. Goldberg, E., Novikov, Y.: BerkMin: a Fast and Robust SAT-Solver. In: DATE-2002, pp. 142--149 .
14. Goldberg, E.: Proving unsatisfiability of CNFs locally. J. Autom. Reasoning, vol.28, no.5, pp. 417--434 (2002).
15. McMillan, K.: Interpolation and SAT-based model checking. In: CAV 03, LNCS, vol. 2725,pp. 1--13.
16. Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an Efficient SAT Solver. In: DAC-2001, pp. 530--535.
17. Nadel, A., Backtrack search algorithms for propositional logic satisfiability: review and innovations. Master Thesis, the Hebrew University (2002).
18. The siege sat-solver, http://www.cs.sfu.ca/~cl/software/siege/
19. Silva,J., Sakallah, K. GRASP: A Search Algorithm for Propositional Satisfiability. IEEE Transactions of Computers 48, 506--521 (1999).
20. Zhang,H.: SATO: An efficient propositional prover. CADE-1997, pp. 272--275.