Chapter 7
Boundary Points and Resolution

Eugene Goldberg, Panagiotis Manolios

7.1 Chapter Overview

We use the notion of boundary points to study resolution proofs. Given a CNF formula F, an $l(x)$-boundary point is a complete assignment falsifying only clauses of F having the same literal $l(x)$ of variable x. An $l(x)$-boundary point $\boldsymbol{p}$ mandates a resolution on variable x. Adding the resolvent of this resolution to F eliminates $\boldsymbol{p}$ as an $l(x)$-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 of proof quality. We argue that obtaining proofs with a high value of this metric requires taking into account the formula structure.

We show that for any unsatisfiable CNF formula there always exists a proof consisting only of resolutions eliminating cut boundary points (which are a relaxation of the notion of boundary points). This result enables building resolution SAT-solvers that are driven by elimination of cut boundary points.

This chapter is an extended version of the conference paper [9].

7.2 Introduction

Resolution-based SAT-solvers [3, 6, 10, 13, 16, 12, 15] have achieved great success in numerous applications. However, the reason for this success and, more generally, the semantics of resolution is not well understood yet. This

obviously impedes progress in SAT-solving. In this chapter, we study the relation between the resolution proof system [2] and boundary points [11]. The most important property of boundary points is that they mandate particular resolutions of a proof. So by studying the relation between resolution and boundary points one gets a deeper understanding of resolutions proofs, which should lead to building better SAT-solvers.

Given a CNF formula F , a non-satisfying complete assignment $\boldsymbol{p}$ is called an $l(x)$-boundary point, if it falsifies only the clauses of F that have the same literal $l(x)$ of variable x. The name is due to the fact that for satisfiable formulas the set of such points contains the boundary between satisfying and unsatisfying assignments. If F is unsatisfiable, for every $l(x)$-boundary point $\boldsymbol{p}$ there is a resolvent of two clauses of F on variable x that eliminates $\boldsymbol{p}$. (That is after adding such a resolvent to F, $\boldsymbol{p}$ is not an $l(x)$-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, one reaches this goal by adding to F resolvents. If formula F has an $l(x)$-boundary point, a resolution proof has to have a resolution operation on variable x. The resolvents of a resolution proof eventually eliminate all boundary points. We will call a resolution mandatory if it eliminates a boundary point of the initial formula F that has not been eliminated by adding the previous resolvents. (In [9] such a resolution was called boundary.)

Intuitively, one can use the Share of Mandatory Resolutions (SMR) of a proof as a metric of proof quality. The reason is that finding mandatory resolutions is not an easy task. (Identification of a boundary point is computationally hard, which implies that finding a mandatory resolution eliminating this point is not easy either.) However, finding mandatory resolutions becomes much simpler if one knows subsets of clauses of F such that resolving clauses of these subsets eliminate boundary points. (An alternative is to try to guess these subsets heuristically.) Intuitively, such subsets have a lot to do with the structure of the formula. So the value of SMR may be used to gauge how well the resolution proof built by a SAT-solver follows the structure of the formula.

We substantiate the intuition above experimentally by comparing two kinds of proofs for equivalence checking formulas. (These formulas describe equivalence checking of two copies of a combinational circuit.) Namely, we consider 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 show experimentally that the share of boundary resolution operations in high-quality specialized proofs is much greater than in proofs generated by the SAT-solver.

Generally speaking, it is not clear yet if for any irredundant unsatisfiable formula there is a proof consisting only of mandatory resolutions. However,

as we show in this chapter, for any unsatisfiable formula F there always exists a proof where each resolution eliminates a *cut boundary point*. The latter is computed with respect to the CNF formula F_T consisting of clauses of F and their resolvents that specify a cut T of a resolution graph describing a proof. (Formula F_T is unsatisfiable for any cut T). The notion of a cut boundary point is a relaxation of that of a boundary point computed with respect to the initial formula F. Point $\boldsymbol{p}$ that is boundary for F_T may not be boundary for F. Proving that resolution is complete with respect to elimination of cut boundary points enables building SAT-solvers that are driven by cut boundary point elimination. Another important observation is that the metric that computes the share of resolutions eliminating cut boundary points is more robust than SMR metric above. Namely, it can be also applied to the formulas that do not have proofs with a 100 % value of the SMR metric.

The contributions of this chapter are as follows. First, we show that one can view resolution as elimination of boundary points. Second, we introduce the SMR metric that can be potentially used as a measure of proof quality. Third, we give some experimental results about the relation between SMR-metric and proof quality. Fourth, we show that resolution remains complete even when it is restricted to resolutions eliminating cut boundary points.

This chapter is structured as follows. Section 7.3 introduces main definitions. Some properties of boundary points are given in Section 7.4. Section 7.5 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 7.6. Experimental results are given in Section 7.7. Some relevant background is recalled in Section 7.8. In Section 7.9 we show that for any unsatisfiable formula there is proof consisting only of resolutions eliminating cut boundary points. Conclusions and directions for future research are listed in Section 7.10.

7.3 Basic Definitions

Definition 7.1. ***A literal*** of a Boolean variable x (denoted as $l(x)$ is a Boolean function of x. The identity and negation functions (denoted as x and $\overline{x}$ respectively) are called the ***positive literal*** and ***negative literal*** of x respectively. We will denote $l(x)$ as just l if the identity of variable x is not important.

Definition 7.2. ***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***. Denote by ***Vars(F)*** (respectively ***Vars(C)***) the set of variables of CNF formula F (respectively clause C).

Definition 7.3. Given a CNF formula $F(x_1, \ldots, x_n)$, ***a complete assignment p*** (also called ***a point***) is a mapping $\{x_1, \ldots, x_n\} \to \{0, 1\}$. 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 $\boldsymbol{p}$ if $C(\boldsymbol{p}) = 1$ (respectively $C(\boldsymbol{p}) = 0$).

Definition 7.4. Given a CNF formula F, ***a satisfying assignment 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 7.5. Let F be a CNF formula and $\boldsymbol{p}$ be a complete assignment. Denote by ***Unsat(p,F)*** the set of all clauses of F falsified by $\boldsymbol{p}$.

Definition 7.6. Given a CNF formula F, a complete assignment $\boldsymbol{p}$ is called an ***$l(x_i)$-boundary point***, if $Unsat(\boldsymbol{p},F) \neq \emptyset$ and every clause of $Unsat(\boldsymbol{p},F)$ contains literal $l(x_i)$.

Example 7.1. Let F consist of 5 clauses: $C_1 = x_2$, $C_2 = \overline{x}_2 \vee x_3$, $C_3 = \overline{x}_1 \vee \overline{x}_3$, $C_4 = x_1 \vee \overline{x}_3, C_5 = \overline{x}_2 \vee \overline{x}_3$. Complete assignment $\boldsymbol{p_1}$=(x_1=0,x_2=0,x_3=1) falsifies only clauses C_1,C_4. So $Unsat(\boldsymbol{p_1},F)$=$\{C_1,C_4\}$. There is no literal shared by all 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 only clauses C_4,C_5 that share literal $\overline{x}_3$. So $\boldsymbol{p_2}$ is a $\overline{x}_3$-boundary point.

7.4 Properties

In this section, we give some properties of boundary points.

7.4.1 Basic Propositions

In this subsection, we prove the following propositions. The set of boundary points contains the boundary between satisfying and unsatisfying assignments (Proposition 7.1). A CNF formula without boundary points is unsatisfiable (Proposition 7.2). Boundary points come in pairs (Proposition 7.3).

Definition 7.7. Denote by ***Bnd_pnts(F)*** the set of all boundary points of a CNF formula F. We assume that an $l(x_i)$-boundary point $\boldsymbol{p}$ is specified in $Bnd\_pnts(F)$ as the pair $(l(x_i),\boldsymbol{p})$. So the same point $\boldsymbol{p}$ may be present in $Bnd\_pnts(F)$ more than once (e.g. if $\boldsymbol{p}$ is an $l(x_i)$-boundary point and an $l(x_j)$-boundary point at the same time).

Proposition 7.1. *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 an $l(x_i)$-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 7.1 means $Bnd\_pnts(F)$ contains the boundary between satisfying and unsatisfying assignments of a satisfiable CNF formula F.

Proposition 7.2. *Let F be a CNF formula that has at least one clause. If $Bnd\_pnts(F) = \emptyset$, then F is unsatisfiable.*

Proof. Assume the contrary i.e. $Bnd\_pnts(F) = \emptyset$ 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 7.1, $\boldsymbol{p_1}$ is an $l(x_i)$-boundary point. A contradiction.

Proposition 7.3. *Let $\boldsymbol{p_1}$ be an $l(x_i)$-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 $\overline{l(x_i)}$-boundary point.*

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

Definition 7.8. Proposition 7.3 means that for unsatisfiable formulas every x_i-boundary point has the corresponding $\overline{x}_i$-boundary point (and vice versa). We will call such a pair of points ***twin boundary points in variable $\boldsymbol{x_i}$***.

Example 7.2. The point $\boldsymbol{p_2}= (x_1 = 0, x_2 = 1, x_3 = 1)$ of Example 7.1 is a $\overline{x}_3$-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 $C_2= \overline{x}_2 \vee x_3$. So $\boldsymbol{p_3}$ is an x_3-boundary point.

7.4.2 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 7.4.

Adding such a resolvent to F eliminates these boundary points (Proposition 7.5). Adding the resolvents of a resolution proof eventually eliminates all boundary points (Proposition 7.6). An $l(x_i)$-boundary point can be eliminated only by a resolution on variable x_i (Proposition 7.7). If formula F has an $l(x_i)$-boundary point, any resolution proof that F is unsatisfiable has a resolution on variable x_i (Proposition 7.8).

Definition 7.9. 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*** of C.

Proposition 7.4. *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 7.3. 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 7.1 and 7.2 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, $C_4 = x_1 \vee \overline{x}_3$, can be resolved with $C_2 = \overline{x}_2 \vee x_3$ on variable x_3. Their resolvent $C = x_1 \vee \overline{x}_2$ is falsified by both $\boldsymbol{p_2}$ and $\boldsymbol{p_3}$.

Proposition 7.5. *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 ($\overline{x}_i$,$\boldsymbol{p_2}$) 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 $\overline{x}_i$-boundary point of F).*

Proof. According to Proposition 7.4, any clauses C_1 and C_2 falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ respectively can be resolved on 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 $\overline{x}_i$-boundary point of $F \wedge C$.

Proposition 7.6. *If a CNF formula F contains an empty clause, then Bnd_pnts(F) = $\emptyset$.*

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 an l-boundary point.

Proposition 7.6 works only in one direction, i.e. if $Bnd\_pnts(F) = \emptyset$, it does not mean that F contains an empty clause. Proposition 7.6 only implies that, given an unsatisfiable formula F for which $Bnd\_pnts(F)$ is not empty, the resolvents of any resolution proof of unsatisfiability of F eventually eliminate all the boundary points.

Proposition 7.7. *Let F be a CNF formula and $\boldsymbol{p}$ be an $l(x_i)$-boundary point of F. Let C be the resolvent of clauses C_1 and C_2 of F that eliminates $\boldsymbol{p}$ (i.e. $(l(x_i),\boldsymbol{p})$ is not in $Bnd\_pnts(F \wedge C)$). Then C is obtained by resolution on variable x_i. In other words, an $l(x_i)$-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 an $l(x_i)$-boundary point, it is falsified by $\boldsymbol{p}$ and does not contain $l(x_i)$. 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 $l(x_i)$, $\boldsymbol{p}$ is not an $l(x_i)$- boundary point of F. A contradiction.

Proposition 7.8. *Let $\boldsymbol{p}$ be an $l(x_i)$-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 7.6, every boundary point of F is eventually eliminated in a resolution proof. According to Proposition 7.7, an $l(x_i)$-boundary point can be eliminated only by adding to F a clause produced by resolution on variable x_i.

7.4.3 Boundary Points and Redundant Formulas

In this subsection, we prove the following propositions. A clause C of a CNF formula F that has a literal l and is not falsified by an l-boundary point of F is redundant in F (Proposition 7.9). If F does not have any $l(x_i)$-boundary points, all clauses depending on variable x_i can be removed from F (Proposition 7.10). If $\boldsymbol{p}$ is an $l(x_i)$-boundary point of F, it is also an $l(x_i)$-boundary point of every unsatisfiable subset of clauses of F.

Definition 7.10. A clause C of a CNF formula F is called ***redundant*** if $F \setminus \{C\} \rightarrow C$.

Proposition 7.9. *Let C be a clause of a CNF formula F. Let l be a literal of C. If no l-boundary point of F falsifies 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 an l-boundary point. A contradiction.

Importantly, Proposition 7.9 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: $\overline{x}_1$, x_1, $x_1 \vee \overline{x}_2$, $x_1 \vee x_2$. 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 $\overline{x}_1$ eliminates $\boldsymbol{p}$ as a boundary point.

Proposition 7.10. *Let a CNF formula F have no $l(x_i)$-boundary points. Then removing the clauses containing x_i or $\overline{x}_i$ from F does not change F (functionally).*

Proof. Let C be a clause of F with a literal $l(x_i)$. Then according to Proposition 7.9 C is redundant in F and so its removal does not change the Boolean function specified by F. Removing C from F, cannot produce an $l(x_i)$-boundary point in $F \setminus \{C\}$. So Proposition 7.9 can be applied again to any of the remaining clauses with x_i or $\overline{x}_i$ (and so on).

Proposition 7.11. *Let F be an unsatisfiable formula. Let $\boldsymbol{p}$ be an $l(x_i)$-boundary point of F and F' be an unsatisfiable subset of clauses of F. Then $\boldsymbol{p}$ is an $l(x_i)$-boundary point of F'.*

Proof. Since $F' \subseteq F$ then $Unsat(\boldsymbol{p},F') \subseteq Unsat(\boldsymbol{p},F)$. Since F' is unsatisfiable, $Unsat(\boldsymbol{p},F') \neq \emptyset$.

7.5 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 proof quality.

7.5.1 Resolution Proof as Boundary Point Elimination

First, we define the notion of a resolution proof [2] and a boundary resolution.

Definition 7.11. 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 $R_1, \ldots, R_k$ 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 7.12. Let $\{R_1, \ldots, R_k\}$ 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 $F \cup \{R_1, \ldots, R_{i-1}\}$ for $i = 2, \ldots, k$. We will say that the i-th resolution (i.e. one that produces resolvent R_i) is ***non-mandatory*** if $Bnd\_pnts(F_i) = Bnd\_pnts(F_{i+1})$. Otherwise (i.e. if $Bnd\_pnts(F_i) \subset Bnd\_pnts(F_{i+1})$, because adding a clause can not create a boundary point) , i-th resolution is called ***mandatory***. So a resolution operation is mandatory if adding R_i to F_i eliminates a boundary point.

In Section 7.4, we showed that eventually all the boundary points of a CNF formula F are removed by resolvents. Importantly, an $l(x_i)$-boundary point mandates a resolution on x_i. Besides, as we showed in Subsection 7.4.3, even redundant clauses can be used to produce new resolvents eliminating boundary points. It is important because all clauses derived by resolution (e.g. conflict clauses generated by modern SAT-solvers) are redundant. So the derived clauses are as good as the original ones for boundary point elimination.

A natural question arises about the role of non-mandatory 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 7.10.) For a redundant formula, one may have to use non-mandatory resolutions. (In particular, a heavily redundant formula may not have boundary points at all. Then every resolution operation is non-mandatory.) For irredundant formulas the situation is different.

Proposition 7.12. *Let F be an irredundant formula of m clauses. Then F has at least d boundary points where d is the number of literals in 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 an l-boundary point where l is a literal of C.

7.5.2 SMR metric and Proof Quality

Intuitively, to efficiently build a short proof for an unsatisfiable CNF formula F, a resolution based SAT-solver has to find mandatory resolutions as soon as possible. Otherwise, a lot of non-mandatory resolutions may be generated that would not have been necessary, had mandatory resolutions been derived early. (In particular, as we show in experiments, an entire proof may consist only of mandatory resolutions.)

This implies that the Share of Mandatory Resolutions (SMR) of a proof can be used as a proof quality metric. Generation of proofs with a high value of SMR most likely requires a good knowledge of the formula structure. The reason is as follows. Finding a mandatory resolution suggests identification of at least one boundary point this resolution eliminates. But detection of boundary points is hard. (Finding an $l(x_i)$-boundary point of formula F reduces to checking the satisfiability of the set of clauses $F \setminus \{$ the clauses of F

with $l(x_i)\}$, see Section 7.7.) Identification of mandatory resolutions without looking for boundary points of F, requires the knowledge of "special" subsets of clauses of the current formula F. (These special subsets should contain clauses whose resolutions produce resolvents eliminating boundary points). Intuitively, such subsets can be identified if the formula structure is known. This intuition is substantiated experimentally in Section 7.7.

The simplest example of information about the formula structure is to identify a small unsatisfiable core. 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 the smallest unsatisfiable subformula of the final CNF formula.) Let F_1 be an unsatisfiable subformula of F. Then no $l(x_i)$-boundary point exists if x_i is in $\mathit{Vars}(F) \setminus \mathit{Vars}(F_1)$. (The set of clauses falsified by any point $\boldsymbol{p}$ contains at least one clause C of F_1 and $x_i \notin \mathit{Vars}(C)$.) So any resolution on a variable of $\mathit{Vars}(F) \setminus \mathit{Vars}(F_1)$ is non-mandatory.

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

Unfortunately, we do not know yet if, given an unsatisfiable irredundant CNF formula, there is always a proof consisting only of mandatory resolutions (and so having a 100% value of SMR metric). Hence a low value of the SMR metric for a CNF formula F may mean that the latter does not have a "natural" proof in the resolution proof system (see Section 7.9). However, as we show in Section 7.9, for any unsatisfiable formula there always exists a proof where each resolution eliminates a *cut boundary point*. So, for measuring proof quality one can also use the share of cut mandatory resolutions (i.e. resolutions eliminating cut boundary points). This metric should be more robust than SMR in the sense that it should work even for formulas that do not have proofs with a 100% value of SMR metric.

7.6 Equivalence Checking Formulas

In this section, we introduce the formulas we use in the experimental part of this chapter. These are the formulas that describe equivalence checking of two copies of a combinational circuit. In Subsection 7.6.1 we show how such formulas are constructed. In Subsection 7.6.2, we build short proofs of unsatisfiability particularly tailored for equivalence checking formulas.

7.6.1 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 use a miter of two identical copies of the same circuit. Then $Miter(N,N^*)$ is always unsatisfiable and so is CNF formula F_{Miter}.

Example 7.4. Figure 7.1 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 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 $F_N \wedge F_N^* \wedge F_{xor} \wedge h$. Here F_N and F_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_{Miter} is unsatisfiable.

Formulas F_N and F_N^* are formed as the conjunction of subformulas describing the gates of N and N^*. For instance, $F_N = F_{g_1} \wedge F_{g_2}$ where, for example, $F_{g_1} = (x_1 \vee x_2 \vee \overline{g}_1) \wedge (\overline{x}_1 \vee g_1) \wedge (\overline{x}_2 \vee g_1)$ specifies the functionality of an OR gate. Each clause of F_{g_1} rules out some inconsistent assignments to the variables of gate g_1. For example, the clause $(x_1 \vee x_2 \vee \overline{g}_1)$ rules out the assignment $x_1 = 0, x_2 = 0, g_1 = 1$.

7.6.2 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 and g_i^* are derived where $Eq(g_i,g_i^*) = (\overline{g}_i \vee g_i^*) \wedge (g_i \vee \overline{g}_i^*)$. 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 $\overline{h}$ is derived. Resolution of $\overline{h}$ and the unit clause h of F_{Miter} produces an empty clause.

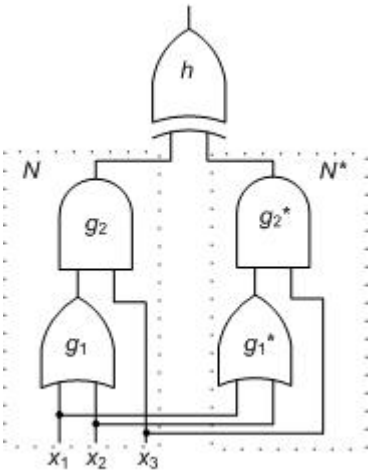

Fig. 7.1 Circuit $Miter(N, N^*)$

Example 7.5. Let us explain the construction of the proof using the CNF F_{Miter} from Example 7.4. 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_{g_1} and $F_{g_1^*}$ describing gates g_1 and g_1^*. That the clauses of $Eq(g_1,g_1^*)$ can be derived from $F_{g_1} \wedge F_{g_1^*}$ just follows from the completeness of resolution and the fact that $Eq(g_1,g_1^*)$ is implied by the CNF formula $F_{g_1} \wedge F_{g_1^*}$. (This implication is due to the fact that F_{g_1} and $F_{g_1^*}$ describe two functionally equivalent gates with the same set of input variables). More specifically, the clause $\overline{g}_1 \vee g_1^*$ is obtained by resolving the clause $x_1 \vee x_2 \vee \overline{g}_1$ of F_{g_1} with the clause $\overline{x}_1 \vee g_1^*$ of $F_{g_1^*}$ and then resolving the resolvent with the clause $\overline{x}_2 \vee g_1^*$ of $F_{g_1^*}$. In a similar manner, the clause $g_1 \vee \overline{g}_1^*$ is derived by resolving the clause $x_1 \vee x_2 \vee \overline{g}_1^*$ of $F_{g_1^*}$ with clauses $\overline{x}_1 \vee g_1$ and $\overline{x}_2 \vee g_1$ of F_{g_1}.

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 $F_{g_2} \wedge F_{g_2^*} \wedge Eq(g_1,g_1^*)$. 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 $F_{g_2} \wedge F_{g_2^*} \wedge Eq(g_1,g_1^*)$ by resolution. Then the clause $\overline{h}$ 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 $\overline{h}$ produces an empty clause.

7.7 Experimental Results

The goal of experiments was to compare the values of SMR metric (see Subsection 7.5.2) for two kinds of proofs of different quality. In the experiments we used formulas describing the equivalence checking of two copies of combina-

tional circuits. The reason for using such formulas is that one can easily generate high-quality specialized proofs of their unsatisfiability (see Section 7.6). In the experiments we compared these short proofs with ones generated by a well known SAT-solver Picosat, version 913 [3].

We performed the experiments on a Linux machine with Intel Core 2 Duo CPU with 3.16GHz clock frequency. The time limit in all experiments was set to 1 hour. The formulas and specialized proofs we used in the experiments can be downloaded from [17].

Table 7.1 Computing value of SMR metric for short specialized proofs

Name	#vars	#clauses	#resolutions	SMR %	time (s.)
c432	480	1,333	1,569	95	1.4
9symml	480	1,413	1,436	100	0.6
mlp7	745	2,216	2,713	100	1.3
c880	807	2,264	2,469	100	3.8
alu4	2,369	7,066	8,229	96	43
c3540	2,625	7,746	9,241	97	137
x1	4,381	12,991	12,885	97	351
dalu	4,714	13,916	15,593	84	286

Given a resolution proof R of k resolutions that a CNF formula F is unsatisfiable, computing the value of SMR metric of R reduces to k SAT-checks. In our experiments, these SAT-checks were performed by a version of DMRP-SAT [8]. Let F_i be the CNF formula $F \cup \{R_1, \ldots, R_{i-1}\}$ where $\{R_1, \ldots, R_{i-1}\}$ 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 $\overline{x}_j$ 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 $\overline{x}_j$, 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 mandatory. 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 7.3). So no $\overline{x}_j$-boundary point is eliminated by i-th resolution either. Hence the unsatisfiability of G_i means that the i-th resolution is non-mandatory.

Table 7.1 shows the value of SMR 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 7.1 gives the size of the proof (in the number of resolutions). The fifth column shows the value of SMR 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 the problem of computing the SMR metric. (For example, one can try to share conflict clauses learned in different SAT-checks.)

Looking at Table 7.1 one can conclude that the specialized proofs have a very high value of SMR-metric (almost every resolution operation eliminates a boundary point). The only exception is the dalu formula (84%). The fact that the value of SMR 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 SMR metric.

Table 7.2 Computing value of SMR metric for proofs generated by Picosat

Name	#resolutions	SMR %	run time (s) (% of proof finished)
c432	19,274	41	75
9symml	12,198	47	28
mlp7	7,253,842	60	$> 1h$ (1.2%)
c880	163,655	17	$> 1h$ (72%)
alu4	672,293	41	$> 1h$ (12%)
c3540	3,283,170	29	$> 1h$ (2.7%)
x1	92,486	45	$> 1h$ (84%)
dalu	641,714	33	$> 1h$ (6.9%)

The values of SMR-metric for the proofs generated by Picosat are given in Table 7.2. The second column gives the size of resolution proofs generated by Picosat. 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 SMR 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 7.3 Using sampling to compute SMR metric for Picosat proofs

Name	sampling rate	SMR %	proof processed %
mlp7	100	18	11
alu4	10	29	36
c3540	100	11	26
dalu	10	37	26

Table 7.2 shows that the size of the proofs generated by Picosat is much larger than that of specialized proofs (Table 7.1, fourth column). Importantly, the value of SMR 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-mandatory. So the early termination of SMR metric computation ignored resolutions with the highest chances to be non-mandatory.

The intuition above is confirmed by the results of Table 7.3. To reach later resolutions we sampled four largest resolution proofs that is we checked only every k-th resolution whether it was mandatory. The value of k is shown in

the second column. The next column gives the value of SMR metric computed over the set of sampled resolutions. The part of the proof covered by sampling within the 1-hour time limit is shown in the last column. It is not hard to see that taking into account later resolutions significantly reduced the value of SMR metric for 3 proofs out of 4.

Summing up, one can conclude that for the formulas we considered in experiments, the proofs of poorer quality (generated by Picosat) have lower values of SMR metric. This substantiates the reasoning of Section 7.5 that not taking into account the formula structure leads to generation of proofs with a low value of SMR metric. On the contrary, picking "right" subsets of clauses to be resolved with each other i.e. closely following the formula structure leads to generation of proofs with a high value of SMR metric.

7.8 Some Background

The notion of boundary points was introduced in [11] where they were called essential points. (We decided to switch to the term "boundary point" as more precise.) Boundary points were used in [11] 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 7.3). The same idea of search pruning was independently described in [14] and implemented in the SAT-solver Jerusat. The ideas of search pruning introduced in [11] were further developed in [5].

In [7], we formulate two proof systems meant for exploring the 1-neighborhood of clauses of the formula to be solved. The union of the 1-neighborhoods 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 7.2). The proof systems of [7] 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 this chapter can also be considered as an approach to improving automatizability of resolution [4]. 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, mandatory resolutions is a tiny part of the set of all possible resolutions. So the restriction of resolutions to mandatory ones (or cut mandatory ones, see Section 7.9) can be viewed as a way to make it easier to find good proofs.

7.9 Completeness of Resolution Restricted to Boundary Point Elimination

In this section, we show that, given a CNF formula (irredundant or not), there always exists a proof consisting only of resolutions that eliminate so called cut boundary points. (This result was not published in [9]). The importance of this result is that it enables SAT-solvers to use (cut) boundary point elimination for building resolution proofs.

7.9.1 Cut Boundary Points

Let F be a CNF formula and resolvents $R_1, ..., R_k$ form a proof R that F is unsatisfiable. So far we considered elimination of boundary points of the original CNF formula F (by adding resolvents R_i). Now we introduce the notion of a cut boundary point.

Denote by G_R a DAG (called ***a resolution graph***) specified by proof R. The nodes of G_R correspond to the clauses of F (the sources of G_R) and resolvents $R_1, ..., R_k$ (the empty clause R_k being the sink of G_R). Graph G_R has an edge directed from n_1 to n_2 if and only if n_2 corresponds to a resolvent and n_1 corresponds to a parent clause of this resolvent.

Denote by T ***a cut*** of G_R i.e. a set of nodes such that every path from a source to the sink of G_R has to go through a node of T. Denote by F_T the CNF formula that consists of the clauses corresponding to the nodes of cut T. (If cut T consists of the sources of G_R, then F_T is the initial formula F.) Formula F_T is unsatisfiable for any cut T. Indeed, the resolutions corresponding to the nodes located between the nodes of T and this sink of G_R form a proof that F_T is unsatisfiable. In terms of Definition 7.12, F_T is a subset of F_i for some $i \leq k$ where $F_i = F \cup R_1 \cup \ldots \cup R_i$. (Since F_i is redundant, one can remove some clauses of F_i without breaking its unsatisfiability.)

Definition 7.13. Let T be a cut of a proof G_R that a CNF formula F is unsatisfiable. Let $\boldsymbol{p}$ be an l-boundary point for F_T (i.e. $\boldsymbol{p} \in \mathit{Bnd\_pnts}(F_T)$). We will call $\boldsymbol{p}$ an l-boundary point with respect to cut T or just ***cut boundary point***.

Note that $\mathit{Vars}(F_i) = \mathit{Vars}(F)$ and $\mathit{Vars}(F_T) \subseteq \mathit{Vars}(F)$. For the sake of simplicity, we will assume that if $\boldsymbol{p}$ is a boundary point for F_T, the variables of $\mathit{Vars}(F) \setminus \mathit{Vars}(F_T)$ are assigned in $\boldsymbol{p}$ (even though these assignments cannot affect satisfying or falsifying a clause of F_T). Importantly, since F_T is only a subset of F_i, a point $\boldsymbol{p}$ that is l-boundary for F_T may not be such for F_i.

Note that if F is an irredundant CNF formula, all resolution graphs G_R have the same cut T consisting only of the nodes that are sources of G_R. (For such a cut, $F_T = F$).

7.9.2 The Completeness Result

In this subsection, we describe the procedure *gen_proof* that, given an unsatisfiable CNF formula F, builds a resolution proof where every resolution eliminates a cut boundary point. This means that the resolution proof system remains complete even if it is restricted only to the resolution operations that eliminate cut boundary points.

```
gen_proof(F)
{/* F_{x_i} - the clauses of F with x_i
  F_{~x_i} - the clauses of F with ~x_i */
while (true)
  {x_i = pick_variable(F);
  while (true)
    {(ans,p) ← find_bp(F,x_i);
    if (ans == failure)
      {F = F \ (F_{x_i} ∪ F_{~x_i});
      break;} /* leave the inner loop */
    (C, C_{x_i}, C_{~x_i}) ← elim_bp(F_{x_i},F_{~x_i},p);
    update_proof(G_R, C, C_{x_i}, C_{~x_i});
    if (empty_clause(C))
      return(rem_redundant(G_R));
    F = F ∪ C; }}}
```

Fig. 7.2 *gen_proof* generates a proof where every resolution eliminates a cut boundary point

The pseudocode of *gen_proof* is shown in Figure 7.2. In the outer *while* loop, *gen_proof* processes variables of F one by one. First, a variable x_i to be processed is chosen by the function *pick_variable*. Then all $l(x_i)$-boundary points are eliminated by *gen_proof* in the inner *while* loop. (Finding and elimination of $l(x_i)$-boundary points is performed by procedures *find_bp* and *elim_bp* respectively.)

To eliminate an $l(x_i)$-boundary point, the resolvent C of parent clauses C_{x_i} and $C_{\overline{x}_i}$ is generated. Here $C_{x_i} \in F_{x_i}$ and $C_{\overline{x}_i} \in F_{\overline{x}_i}$ where F_{x_i} and $F_{\overline{x}_i}$ are the clauses of F having literals x_i and $x_{\overline{x}_i}$ respectively. If C is an empty clause, *gen_proof* stops and returns the resolution graph G_R specifying a proof that F is unsatisfiable. (G_R may contain nodes corresponding to resolvents that are not on a path leading from a source to the empty clause. So, the procedure *rem_redundant* is called to remove the parts of G_R corresponding to redundant resolution operations.) Otherwise, a new $l(x_i)$-boundary point is looked for. When all $l(x_i)$-boundary points are eliminated, the clauses F_{x_i} and $F_{\overline{x}_i}$ are removed from F and *proof_gen* leaves the inner loop.

That the *proof_gen* procedure is sound follows from the soundness of the resolution proof system. The *proof_gen* procedure is also complete. The number of $l(x_i)$-boundary points is finite and monotonically decreases in the process of adding resolvents C to F. So after a finite number of steps, all $l(x_i)$-boundary points are eliminated from the current formula F. At this point, the clauses of F containing x_i or $\overline{x}_i$ are removed from F, which does affect its unsatisfiability (see Proposition 7.10). After processing all the variables, an empty clause is inevitably derived (because the resulting formula has no variables and has to be functionally equivalent to the original formula F that is unsatisfiable).

Let us show that every resolution operation of the proof G_R produced by *gen_proof* eliminates a cut boundary point. Let C be a resolvent on variable

x_i produced from parent clauses C_{x_i} and $C_{\overline{x}_i}$ in the inner loop of the *gen_proof* procedure. By construction, C eliminates an $l(x_i)$-boundary point $\boldsymbol{p}$ of the current formula F. The nodes corresponding to F form a cut of G_R (because every resolvent produced in the future is a descendent of clauses of F). The elimination of redundant nodes of G_R by the procedure *rem_redundant* of *gen_proof* does not change much. If the resolvent C turns out to be redundant, then whether or not adding C to F eliminates a cut boundary point is irrelevant. If the resolvent C stays in the proof, it still eliminates the same $l(x_i)$ boundary $\boldsymbol{p}$ (according to Proposition 7.11).

Summarizing, each resolution of the proof specified by the graph G_R built by *gen_proof* eliminates a cut boundary point. (The cut corresponding to a resolution is specified by the formula F at the time this resolution was performed minus some redundant clauses identified by *rem_redundant*).

7.9.3 Boundary Points as Complexity Measure

The existence of an $l(x_i)$-boundary point of a CNF formula F implies dependency of F on variable x_i. In this subsection, we argue that in the context of resolution proofs, $Bnd\_pnts(F)$ is a more precise complexity measure than $Vars(F)$. Besides, we introduce the notion of natural resolution proofs.

Let resolution graph G_R specify a proof that a CNF formula F is unsatisfiable. Such a proof can be represented as a sequence of unsatisfiable formulas $F_{T_1}, \ldots, F_{T_m}$ corresponding to cuts T_1,..,T_m of G_R. We assume that $F_{T_1} = F$ and F_{T_m} consists only of an empty clause and that $T_i \leq T_j$ if $i \leq j$. (The partial order $T_i \leq T_j$ holds iff there is no path from a source to the sink of G_R such that a node of T_j appears on this path before a node of T_i.)

It is natural to expect that formulas F_{T_i} are getting easier to solve as the value of i grows and eventually the trivial formula F_m is derived. Note that in terms of the number of variables this is true because $T_i \leq T_j \rightarrow Vars(T_i) \subseteq Vars(T_j)$. However, in terms of boundary points this is, in general, not true. For example, it is possible that $Bnd\_pnts(F_{T_i})$ is not a subset of $Bnd\_pnts(F_{T_j})$ even though $T_i \leq T_j$. The reason is that F_{T_j} is obtained from F_{T_i} by adding and *removing* some clauses. The removal of a clause C from F_{T_i} may lead to appearance of a new $l(x_s)$-boundary point $\boldsymbol{p}$ where $x_s \notin Vars(C)$. This means that F_{T_j} in some aspect depends on x_s stronger than F_{T_i} (because a resolution on variable x_s is mandated by $\boldsymbol{p}$).

The observation above implies that boundary points provide a more precise way to measure formula complexity than just computing the set of variables on which the formula depends. It would be interesting to find classes of formulas for which there exist resolution proofs where the complexity of formulas F_{T_i} monotonically reduces in terms of $Bnd\_pnts(F_{T_i})$. Intuitively, this is possible if the proof structure follows the natural structure of the formula. So such proofs can be called natural. On the other hand, there may be

a class of formulas for which natural resolution proofs do not exist (due to the fact that the structure of any resolution proof does not agree with that of a formula from this class).

7.10 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 SMR metric that is the percent of resolutions of the proof that eliminate boundary points (mandatory resolutions). This metric can be used for gauging proof quality. We experimentally show that short specialized proofs for equivalence checking formulas have high values of SMR metric. On the other hand, values of this metric for proofs generated by a SAT-solver with conflict driven learning are low. As we argue in Section 7.5, this may be attributed to not taking into account the formula structure.

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

1. Studying the quality of proofs consisting only of resolutions eliminating cut boundary points. (We showed the existence of such proofs for every CNF formula but have not made any claims about their quality.)
2. Studying further the relation between the value of SMR metric for resolution proofs obtained by SAT-solvers and their ability to take into account the formula structure.
3. Building SAT-solvers based on the idea of (cut) boundary point elimination.
4. Finding the classes of formulas for which "natural" proofs exist i.e. proofs for which the complexity of cut CNF formulas monotonically decreases (in terms of cut boundary points).

References

1. Alekhnovich, M., Razborov, A.: Resolution is not automatizable unless w[p] is tractable. SIAM J. Comput. **38**(4), 1347–1363 (2008)
2. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: A. Robinson, A. Voronkov (eds.) Handbook of Automated Reasoning, vol. I, chap. 2, pp. 19–99. North-Holland (2001)
3. Biere, A.: Picosat essentials. JSAT **4**(2-4), 75–97 (2008)
4. Bonet, L., Pitassi, T., Raz, R.: On interpolation and automatization for frege systems. SIAM J. Comput. **29**(6), 1939–1967 (2000)
5. D.Babic, Bingham, J., A.Hu: Efficient sat solving: beyond supercubes. In: DAC, pp. 744–749. Anaheim, California, USA (2005)
6. Eén, N., Sörensson, N.: An extensible sat-solver. In: SAT, pp. 502–518. Santa Margherita Ligure, Italy (2003)

7. Goldberg, E.: Proving unsatisfiability of cnfs locally. J. Autom. Reason. **28**(4), 417–434 (2002)
8. Goldberg, E.: A decision-making procedure for resolution-based sat-solvers. In: SAT-08, pp. 119–132. Guangzhou, China (2008)
9. Goldberg, E.: Boundary points and resolution. In: SAT-09, pp. 147–160. Springer-Verlag, Swansea, Wales, United Kingdom (2009)
10. Goldberg, E., Novikov, Y.: Berkmin: A fast and robust sat-solver. Discrete Appl. Math. **155**(12), 1549–1561 (2007)
11. Goldberg, E., Prasad, M., Brayton, R.: Using problem symmetry in search based satisfiability algorithms. In: DATE '02, pp. 134–141. Paris, France (2002)
12. Marques-Silva, J., Sakallah, K.: Grasp—a new search algorithm for satisfiability. In: ICCAD-96, pp. 220–227. Washington, DC, USA (1996)
13. Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient sat solver. In: DAC-01, pp. 530–535. New York, NY, USA (2001)
14. Nadel, A.: Backtrack search algorithms for propositional logic satisfiability: review and innovations. Master's thesis, The Hebrew University (2002)
15. Zhang, H.: Sato: An efficient propositional prover. In: CADE-97, pp. 272–275. Springer-Verlag, London, UK (1997)
16. The siege sat-solver, http://www.cs.sfu.ca/~cl/software/siege
17. Benchmarks, http://eigold.tripod.com/benchmarks/book2010.tar.gz