Removal of Quantifiers by Elimination of Boundary Points

Eugene Goldberg and Panagiotis Manolios

College of Computer and Information Science
Northeastern University
360 Huntington Ave., Boston MA 02115, USA
Email: {eigold,pete}@ccs.neu.edu

***Abstract*—We consider the problem of elimination of existential quantifiers from a Boolean CNF formula. Our approach is based on the following observation. One can get rid of dependency on a set of variables of a quantified CNF formula F by adding resolvent clauses of F eliminating boundary points. This approach is similar to the method of quantifier elimination described in [9]. The difference of the method described in the present paper is twofold:**

- **branching is performed only on quantified variables,**
- **an explicit search for boundary points is performed by calls to a SAT-solver**

Although we published the paper [9] before this one, chronologically the method of the present report was developed first. Preliminary presentations of this method were made in [10], [11]. We postponed a publication of this method due to preparation of a patent application [8].

I. Introduction

In this paper, we are concerned with the problem of elimination of quantified variables from a Boolean CNF formula. (Since we consider only existential quantifiers, further on we omit the word "existential".) Namely, we solve the following problem: given a Boolean CNF formula $\exists X.F(X,Y)$, find a Boolean CNF formula $F^*(Y)$ such that $F^*(Y) \equiv \exists X.F(X,Y)$. We will refer to this problem as QEP (Quantifier Elimination Problem). Since QEP is to find a formula, it is not a decision problem as opposed to the problem of solving a Quantified Boolean Formula (QBF). QEP occurs in numerous areas of hardware/software design and verification, model checking [4], [18] being one of the most prominent applications of QEP.

A straightforward method of solving QEP for CNF formula $\exists X.F(X,Y)$ is to eliminate the variables of X one by one, in the way it is done in the DP procedure [5]. To delete a variable x_i of X, the DP procedure produces all possible resolvents on variable x_i and adds them to F. An obvious drawback of such a method is that it generates a prohibitively large number of clauses. Another set of QEP-solvers employ the idea of enumerating satisfying assignments of formula $F(X,Y)$. Here is how a typical method of this kind works. First, a CNF formula $F^+(Y)$ is built such that each clause C of F^+ (called a blocking clause [17]) eliminates a set of assignments satisfying $F(X,Y)$. By negating $F^+(Y)$ one obtains a CNF formula $F^*(Y)$ that is a solution to QEP.

Unfortunately, F^+ may be exponentially larger than F^*. This occurs, for instance, when $F(X,Y) = F_1(X_1,Y_1) \wedge \ldots \wedge F_k(X_k,Y_k)$ and $(X_i \cup Y_i) \cap (X_j \cup Y_j) = \emptyset$, $i \neq j$ that is when F is the conjunction of independent CNF formulas F_i. In this case, one can build $F^*(Y)$ as $F^*_1 \wedge \ldots \wedge F^*_k$, where $F^*_i(Y_i) \equiv \exists X_i.F_i(X_i,Y_i)$,$i = 1,\ldots,k$. So the size of F^* is linear in k whereas that of F^+ is *exponential* in k. This fact implies that QEP-solvers based on enumeration of satisfying assignments *are not compositional*. (We say that a QEP-solver is compositional if it reduces the problem of finding $F^*(Y)$ to k independent subproblems of finding $F^*_i(Y_i)$,$i = 1,\ldots,k$.) Note that in practical applications, it is very important for a QEP-solver to be compositional. Even if F does not break down into independent subformulas, there may be numerous branches of the search tree where such subformulas appear.

Both kinds of QEP-solvers mentioned above have the same drawback. A resolution-based QEP-solver can only efficiently check if a clause C of $F^*(Y)$ is correct i.e. whether it is implied by $F(X,Y)$. But how does one know if F^* contains a *sufficient* set of correct clauses i.e. whether every assignment $\boldsymbol{y}$ satisfying F^* can be extended to ($\boldsymbol{x}$,$\boldsymbol{y}$) satisfying F? A *non-deterministic* algorithm does not have to answer this question. Once a sufficient set of clauses is derived, an oracle stops this algorithm. But a *deterministic* algorithm has no oracle and so has to decide for itself when it is the right time to terminate. One way to guarantee the correctness of termination is to enumerate the satisfying assignments of F. The problem here is that then, the size of a deterministic derivation of F^* may be exponentially larger than that of a non-deterministic one. (Non-compositionality of QEP-solvers based on enumeration of satisfying assignments is just a special case of this problem.)

In this paper, we introduce a new termination condition for QEP that is based on the notion of boundary points. A complete assignment $\boldsymbol{p}$ falsifying $F(X,Y)$ is an X'-boundary point where $X' \subseteq X$ if a) every clause of F falsified by $\boldsymbol{p}$ has a variable of X' and b) first condition breaks for every proper subset of X'. An X'-boundary point $\boldsymbol{p}$ is called removable if no satisfying assignment of F can be obtained from $\boldsymbol{p}$ by changing values of variables of X. One can eliminate a removable X'-boundary point by adding to F a clause C that is implied by F and does not have a variable of X'. If for a set of variables X'' where $X'' \subseteq X$, formula $F(X,Y)$ does not have a removable X'-boundary point where $X' \subseteq X''$, the variables of X'' are *redundant* in formula $\exists X.F(X,Y)$. This means that every clause with a variable of X'' can be removed from $F(X,Y)$. QEP-solving terminates when the current formula $F(X,Y)$ (consisting of the initial clauses and resolvents) has no removable boundary points. A solution $F^*(Y)$ to QEP is formed from $F(X,Y)$ by discarding every clause that has a variable of X.

The new termination condition allows one to address drawbacks of the QEP-solvers mentioned above. In contrast to the DP procedure, *only* resolvents eliminating a boundary point need to be added. This dramatically reduces the number of resolvents one has to generate. On the other hand, a solution F^* can be derived *directly* without enumerating satisfying assignments of F. In particular, using the new termination condition makes a QEP-solver compositional.

To record the fact that all boundary removable points have been removed from a subspace of the search space, we introduce the notion of a dependency sequent (D-sequent for short). Given a CNF formula $F(X,Y)$, a D-sequent has the form $(F, X', \boldsymbol{q}) \rightarrow X''$ where $\boldsymbol{q}$ is a partial assignment to variables of X, $X' \subseteq X$, $X'' \subseteq X$. Let $F_{\boldsymbol{q}}$ denote formula F after assignments $\boldsymbol{q}$ are made. We say that the D-sequent above holds if

- the variables of X' are redundant in $F_{\boldsymbol{q}}$,
- the variables of X'' are redundant in the formula obtained from $F_{\boldsymbol{q}}$ by discarding every clause containing a variable of X'.

The fact that the variables of X' (respectively X'') are redundant in F means that F has no removable X^*-boundary point where $X^* \subseteq X'$ (respectively $X^* \subseteq X''$). The reason for using name "D-sequent" is that the validity of $(F, X', \boldsymbol{q}) \rightarrow X''$ suggests interdependency of variables of $\boldsymbol{q}$, X' and X''.

In a sense, the notion of a D-sequent generalizes that of an implicate of formula $F(X,Y)$. Suppose, for instance, that $F \rightarrow C$ where $C = x_1 \vee x_2$, $x_1 \in X$, $x_2 \in X$. After adding C to F, the D-sequent $(F, \emptyset, \boldsymbol{q}) \rightarrow X'$ where $\boldsymbol{q}$=$(x_1 = 0,\ x_2 = 0)$, $X' = X \setminus \{x_1, x_2\}$ becomes true. (An assignment falsifying C makes the unassigned variables of F redundant.) But the opposite is not true. The D-sequent above may hold even if $F \rightarrow C$ does not. (The latter means that $\boldsymbol{q}$ can be extended to an assignment satisfying F).

We will refer to the method of QEP-solving based on elimination of boundary points as DDS (Derivation of D-Sequents). We will refer to the QEP-solver based on the DDS method we describe in this paper as *DDS_impl* (DDS implementation). To reflect the progress in elimination of boundary points of F, *DDS_impl* uses resolution of D-sequents. Suppose D-sequents $(F, \emptyset, \boldsymbol{q_1}) \rightarrow \{x_{10}\}$ and $(F, \emptyset, \boldsymbol{q_2}) \rightarrow \{x_{10}\}$ have been derived where $\boldsymbol{q_1}$=$(x_1 = 0,\ x_3 = 0)$ and $\boldsymbol{q_2}$=$(x_1 = 1,\ x_4 = 0)$. Then a new D-sequent $(F, \emptyset, \boldsymbol{q}) \rightarrow \{x_{10}\}$ where $\boldsymbol{q} = (x_3 = 0, x_4 = 0)$ can be produced from them by resolution on variable x_1. *DDS_impl* terminates as soon as D-sequent $(F, \emptyset, \emptyset) \rightarrow X$ is derived, which means that the variables of X are redundant in F (because every removable X'-boundary point where $X' \subseteq X$ has been eliminated from F due to adding resolvent-clauses).

Our contribution is threefold. First, we formulate a new method of quantifier elimination based on the notion of X-removable boundary points which are a generalization of those introduced in [14]. One of the advantages of this method is that it uses a new termination condition. Second, we introduce the notion of D-sequents and the operation of resolution of D-sequents. The calculus of D-sequents is meant for building QEP-solvers based on the semantics of boundary point elimination. Third, we describe a QEP-solver called *DDS_impl* and prove its compositionality. We show that in contrast to a BDD-based QEP-solver that is compositional only for particular variable orderings, *DDS_impl* is compositional regardless of how branching variables are chosen. We give preliminary experimental results that show the promise of DDS.

This paper is structured as follows. In Section II, we define the notions related to boundary points. The relation between boundary points and QEP is discussed in Section III. Section IV describes how adding/removing clauses affects the set of boundary points of a formula. D-sequents are introduced in Section V. Section VI describes *DDS_impl*. The compositionality of *DDS_impl* is discussed in Section VII. Section VIII describes experimental results. Some background in given in Section IX. Section X summarizes this paper.

II. Basic Definitions

Notation: Let F be a CNF formula and C be a clause. We denote by $\mathit{Vars}(F)$ (respectively $\mathit{Vars}(C)$) the set of variables of F (respectively of C). If $\boldsymbol{q}$ is a partial assignment to $\mathit{Vars}(F)$, $\mathit{Vars}(\boldsymbol{q})$ denotes the variables assigned in $\boldsymbol{q}$.

Notation: In this paper, we consider a quantified CNF formula $\exists X.F(X,Y)$ where $X \cup Y = \mathit{Vars}(F)$ and $X \cap Y = \emptyset$.

Definition 1: A CNF formula $F^*(Y)$ is a solution to the Quantifier Elimination Problem (**QEP**) if $F^*(Y) \equiv \exists X.F(X,Y)$.

Definition 2: Given a CNF formula $G(Z)$, a complete assignment to the variables of Z is called **a point**.

Definition 3: Let $G(Z)$ be a CNF formula and $Z' \subseteq Z$. A clause C of G is called a **Z'-clause** if $\mathit{Vars}(C) \cap Z' \neq \emptyset$. Otherwise, C is called a **non-Z'-clause**.

Definition 4: Let $G(Z)$ be a CNF formula and $Z' \subseteq Z$. A point $\boldsymbol{p}$ is called a **Z'-boundary point** of G if $G(\boldsymbol{p}) = 0$ and

1) Every clause of G falsified by $\boldsymbol{p}$ is a Z'-clause.
2) Condition 1 breaks for every proper subset of Z'.

A Z'-boundary point $\boldsymbol{p}$ is at least $|Z'|$ flips away from a point $\boldsymbol{p^*}$, $G(\boldsymbol{p^*}) = 1$ (if $\boldsymbol{p^*}$ exists and only variables of Z' are allowed to be changed), hence the name "boundary".

Let $\boldsymbol{p}$ be a Z'-boundary point of $G(Z)$ where $Z' = \{z\}$. Then every clause of G falsified by $\boldsymbol{p}$ contains variable z. This special class of boundary points was introduced in [13], [14].

Definition 5: Point $\boldsymbol{p}$ is called a **Z'-removable boundary point** of $G(Z)$ where $Z' \subseteq Z$ if $\boldsymbol{p}$ is a Z''-boundary point where $Z'' \subseteq Z'$ and there is a clause C such that

- $\boldsymbol{p}$ falsifies C;

- C is a non-Z'-clause;
- C is implied by the conjunction of Z'-clauses of G.

Adding clause C to G eliminates $\boldsymbol{p}$ as a Z''-boundary point ($\boldsymbol{p}$ falsifies clause C and C has no variables of Z'').

Proposition 1: Point $\boldsymbol{p}$ is a Z'-removable boundary point of a CNF formula $G(Z)$ iff no point $\boldsymbol{p}^*$ obtained from $\boldsymbol{p}$ by changing values of (some) variables of Z' satisfies G.

The proofs are given in the Appendix.

Example 1: Let CNF formula G consist of four clauses: $C_1 = z_1 \vee z_2$, $C_2 = z_3 \vee z_4$, $C_3 = \overline{z_1} \vee z_5$, $C_4 = \overline{z_3} \vee z_5$. Let $\boldsymbol{p}$=$(z_1=0, z_2=0, z_3=0, z_4=0, z_5=0)$. Point $\boldsymbol{p}$ falsifies only C_1 and C_2. Since both C_1 and C_2 contain a variable of $Z'' = \{z_1, z_3\}$, $\boldsymbol{p}$ is a Z''-boundary point. (Note that $\boldsymbol{p}$ is also, for instance, a $\{z_2, z_4\}$-boundary point.) Let us check if point $\boldsymbol{p}$ is a Z'-removable boundary point where $Z' = \{z_1, z_3, z_5\}$. One condition of Definition 5 is met: $\boldsymbol{p}$ is a Z''-boundary point, $Z'' \subseteq Z'$. However, the point $\boldsymbol{p}^*$ obtained from $\boldsymbol{p}$ by flipping the values of z_1,z_3,z_5 satisfies G. So, according to Proposition 1, $\boldsymbol{p}$ is not a Z'-removable boundary point (i.e. the clause C of Definition 5 does not exist for $\boldsymbol{p}$).

Definition 6: We will say that a boundary point $\boldsymbol{p}$ of $F(X,Y)$ is just **removable** if it is X-removable.

Remark 1: Informally, a boundary point $\boldsymbol{p}$ of $F(X,Y)$ is removable only if there exists a clause C implied by F and falsified by $\boldsymbol{p}$ such that $Vars(C) \subseteq Y$. The fact that an X''-boundary point $\boldsymbol{p}$ is not X'-removable (where $X'' \subseteq X'$) also means that $\boldsymbol{p}$ is not removable. The opposite is not true.

III. X-Boundary Points and Quantifier Elimination

In this section, we relate QEP-solving and boundary points. First we define the notion of redundant variables in the context of boundary point elimination (Definition 7). Then we show that monotone variables are redundant (Proposition 2). Then we prove that clauses containing variables of X', $X' \subseteq X$ can be removed from formula $\exists X.F(X,Y)$ if and only if the variables of X' are redundant in F (Proposition 3).

Definition 7: Let $F(X,Y)$ be a CNF formula and $X' \subseteq X$. We will say that the variables of X' are **redundant** in F if F has no removable X''-boundary point where $X'' \subseteq X$.

Proposition 2: Let $G(Z)$ be a CNF formula and z be a monotone variable of F. (That is clauses of G contain the literal of z of only one polarity.) Then z is redundant in G.

Definition 8: Let $F(X,Y)$ be a CNF formula. Denote by $\boldsymbol{Dis(F,X')}$ where $X' \subseteq X$ the CNF formula obtained from $F(X,Y)$ by discarding all X'-clauses.

Proposition 3: Let $F(X,Y)$ be a CNF formula and X' be a subset of X. Then $\exists X.F(X,Y) \equiv \exists (X \setminus X').Dis(F,X')$ iff the variables of X' are redundant in F.

Corollary 1: Let $F(X,Y)$ be a CNF formula. Let $F^*(Y) = Dis(F,X)$. Then $F^*(Y) \equiv \exists X.F(X,Y)$ holds iff the variables of X are redundant in F.

IV. Appearance of Boundary Points When Adding/Removing Clauses

In this section, we give two theorems later used in Proposition 8 (about D-sequents built by *DDS_impl*). They describe the type of clauses one can add to (or remove from) $G(Z)$ without creating a new $\{z\}$-removable boundary point where $z \in Z$.

Proposition 4: Let $G(Z)$ be a CNF formula. Let G have no $\{z\}$-removable boundary points. Let C be a clause. Then the formula $G \wedge C$ does not have a $\{z\}$-removable boundary point if at least one of the following conditions hold: a) C is implied by G; b) $z \notin Vars(C)$.

Proposition 5: Let $G(Z)$ be a CNF formula. Let G have no $\{z\}$-removable boundary points. Let C be a $\{z\}$-clause of G. Then the CNF formula G' where $G' = G \setminus \{C\}$ does not have a $\{z\}$-removable boundary point.

Remark 2: According to Propositions 4 and 5, adding clause C to a CNF formula G or removing C from G may produce a new $\{z\}$-removable boundary point only if:

- one adds to G a $\{z\}$-clause C that is not implied by G or
- one removes from G a clause C that is not a $\{z\}$-clause.

V. Dependency Sequents (D-sequents)

A. General Definitions and Properties

In this subsection, we introduce D-sequents (Definition 10) and resolution of D-sequents (Definition 12). Proposition 6 states that a D-sequent remains true if resolvent-clauses are added to F. The soundness of resolving D-sequents is shown in Proposition 7.

Definition 9: Let F be a CNF formula and $\boldsymbol{q}$ be a partial assignment to $Vars(F)$. Denote by $F_{\boldsymbol{q}}$ the CNF formula obtained from F by

- removing the literals of (unsatisfied) clauses of F that are set to 0 by $\boldsymbol{q}$,
- removing the clauses of F satisfied by $\boldsymbol{q}$,

Definition 10: Let $F(X,Y)$ be a CNF formula. Let $\boldsymbol{q}$ be a partial assignment to variables of X and X' and X'' be subsets of X such that $Vars(\boldsymbol{q})$, X', X'' do not overlap. A dependency sequent (**D-sequent**) S has the form $(F,X',\boldsymbol{q}) \rightarrow X''$. We will say that S holds if

- the variables of X' are redundant in $F_{\boldsymbol{q}}$ (see Definition 9),
- the variables of X'' are redundant in $Dis(F_{\boldsymbol{q}},X')$ (see Definition 8).

Example 2: Let CNF formula $F(X,Y)$ where $X = \{x_1,x_2\}$, $Y = \{y_1,y_2\}$ consist of two clauses: $C_1 = x_1 \vee y_1$ and $C_2 = \overline{x}_1 \vee x_2 \vee y_2$. Note that variable x_2 is monotone and hence redundant in F (due to Proposition 2). After discarding the clause C_2 (containing the redundant variable x_2), variable x_1 becomes redundant. Hence, the D-sequent $(F,\{x_2\},\emptyset) \rightarrow \{x_1\}$ holds.

Proposition 6: Let $F^+(X,Y)$ be a CNF formula obtained from $F(X,Y)$ by adding some resolvents of clauses of F. Let $\boldsymbol{q}$ be a partial assignment to variables of X and $X' \subseteq X$. Then the fact that D-sequent $(F,X',\boldsymbol{q}) \rightarrow X''$ holds implies that $(F^+,X',\boldsymbol{q}) \rightarrow X''$ holds too. The opposite is not true.

Definition 11: Let $F(X,Y)$ be a CNF formula and $\boldsymbol{q'}$, $\boldsymbol{q''}$ be partial assignments to X. Let $Vars(\boldsymbol{q'}) \cap Vars(\boldsymbol{q''})$ contain exactly one variable x for which $\boldsymbol{q'}$ and $\boldsymbol{q''}$ have the opposite values. Then the partial assignment $\boldsymbol{q}$ such that

- $Vars(\boldsymbol{q})$ = $((Vars(\boldsymbol{q'}) \cup Vars(\boldsymbol{q''})) \setminus \{x\}$,
- the value of each variable x^* of $Vars(\boldsymbol{q})$ is equal to that of x^* in $Vars(\boldsymbol{q'}) \cup Vars(\boldsymbol{q''})$.

is denoted as $Res(\boldsymbol{q'},\boldsymbol{q''},x)$ and called **the resolvent of $\boldsymbol{q'}$,$\boldsymbol{q''}$** on x. Assignments $\boldsymbol{q'}$ and $\boldsymbol{q''}$ are called resolvable on x.

Proposition 7: Let $F(X,Y)$ be a CNF formula. Let D-sequents S_1 and S_2 be equal to $(F,X_1,\boldsymbol{q_1}) \rightarrow X'$ and $(F,X_2,\boldsymbol{q_2}) \rightarrow X'$ respectively. Let $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ be resolvable on variable x. Denote by $\boldsymbol{q}$ the partial assignment $Res(\boldsymbol{q_1},\boldsymbol{q_2},x)$ and by X^* the set $X_1 \cap X_2$. Then, if S_1 and S_2 hold, the D-sequent S equal to $(F,X^*,\boldsymbol{q}) \rightarrow X'$ holds too.

Definition 12: We will say that the D-sequent S of Proposition 7 is produced by **resolving D-sequents S_1 and S_2** on variable x. S is called the resolvent of S_1 and S_2 on x.

B. Derivation of D-sequents in DDS_impl

In this subsection, we discuss generation of D-sequents in *DDS_impl* (see Section VI). *DDS_impl* builds a search tree by branching on variables of X of $F(X,Y)$.

Definition 13: Let $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ be partial assignments to variables of X. We will denote by $\boldsymbol{q_1} \leq \boldsymbol{q_2}$ the fact that a) $Vars(\boldsymbol{q_1}) \subseteq Vars(\boldsymbol{q_2})$ and b) every variable of $Vars(\boldsymbol{q_1})$ is assigned in $\boldsymbol{q_1}$ exactly as in $\boldsymbol{q_2}$.

Let $\boldsymbol{q}$ be the current partial assignment to variables of X and X_{red} be the unassigned variables proved redundant in $F_{\boldsymbol{q}}$. *DDS_impl* generates a new D-sequent a) by resolving two existing D-sequents or b) if one of the conditions below is true.

1) *A (locally) empty clause appears in* $Dis(F_{\boldsymbol{q}},X_{red})$. Suppose, for example, that F contains clause $C = x_1 \vee \overline{x_5} \vee x_7$. Assume that assignments $(x_1 = 0, x_5 = 1)$ are made turning C into the unit clause x_7. Assignment $x_7 = 0$ makes C an empty clause and so eliminates all boundary points of $Dis(F_{\boldsymbol{q}},X_{red})$. So *DDS_impl* builds D-sequent $(F,\emptyset,\boldsymbol{g}) \rightarrow X'$ where $\boldsymbol{g}$ = $(x_1 = 0, x_5 = 1, x_7 = 0)$ and X' is the set of unassigned variables of $Dis(F_{\boldsymbol{q}},X_{red})$ that are not in X_{red}.

2) $Dis(F_{\boldsymbol{q}},X_{red})$ *has only one variable* x of X that is not assigned and is not redundant. In this case, *DDS_impl* makes x redundant by adding resolvents on variable x and then builds D-sequent $(F,X'_{red},\boldsymbol{g}) \rightarrow \{x\}$ where $X'_{red} \subseteq X_{red}$, $\boldsymbol{g} \leq \boldsymbol{q}$ and X'_{red} and $\boldsymbol{g}$ are defined in Proposition 8 below (see also Remark 3).

3) *A monotone variable x appears in formula* $Dis(F_{\boldsymbol{q}},X_{red})$. Then *DDS_impl* builds D-sequent $(F,X'_{red},\boldsymbol{g}) \rightarrow \{x\}$ where $X'_{red} \subseteq X_{red}$, $\boldsymbol{g} \leq \boldsymbol{q}$ and X'_{red} and $\boldsymbol{g}$ are defined in Proposition 8 (see Remark 4).

Proposition 8 and Remark 3 below explain how to pick a subset of assignments of the current partial assignment $\boldsymbol{q}$ responsible for the fact that a variable x is redundant in branch $\boldsymbol{q}$. This is similar to picking a subset of assignments responsible for a conflict in SAT-solving.

Proposition 8: Let $F(X,Y)$ be a CNF formula and $\boldsymbol{q}$ be a partial assignment to variables of X. Let X_{red} be the variables proved redundant in $F_{\boldsymbol{q}}$. Let x be the only variable of X that is not in $Vars(\boldsymbol{q}) \cup X_{red}$. Let D-sequent $(F,X_{red},\boldsymbol{q}) \rightarrow \{x\}$ hold. Then D-sequent $(F,X'_{red},\boldsymbol{g}) \rightarrow \{x\}$ holds where $\boldsymbol{g}$ and X'_{red} are defined as follows. Partial assignment $\boldsymbol{g}$ to variables of X satisfies the two conditions below (implying that $\boldsymbol{g} \leq \boldsymbol{q}$):

1) Let C be a $\{x\}$-clause of F that is not in $Dis(F_{\boldsymbol{q}},X_{red})$. Then either
   - $\boldsymbol{g}$ contains an assignment satisfying C or
   - D-sequent $(F,X^*_{red},\boldsymbol{g^*}) \rightarrow \{x^*\}$ holds where $\boldsymbol{g^*} \leq \boldsymbol{g}$, $X^*_{red} \subset X_{red}$, $x^* \in (X_{red} \cap Vars(C))$.
2) Let $\boldsymbol{p_1}$ be a point such that $\boldsymbol{q} \leq \boldsymbol{p_1}$. Let $\boldsymbol{p_1}$ falsify a clause of F with literal x. Let $\boldsymbol{p_2}$ be obtained from $\boldsymbol{p_1}$ by flipping the value of x and falsify a clause of F with literal $\overline{x}$. Then there is a non-$\{x\}$-clause C of F falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ such that $(Vars(C) \cap X) \subseteq Vars(\boldsymbol{g})$.

The set X'_{red} consists of all the variables already proved redundant in $F_{\boldsymbol{g}}$. That is every redundant variable x^* of X_{red} with D-sequent $(F,X^*_{red},\boldsymbol{g^*}) \rightarrow \{x^*\}$ such that $\boldsymbol{g^*} \leq \boldsymbol{g}$, $X^*_{red} \subset X_{red}$ is in X'_{red}.

Remark 3: When backtracking (and making new assignments) formula $Dis(F_{\boldsymbol{q}},X_{red})$ changes. Partial assignment $\boldsymbol{g}$ is formed so as to prevent the changes that may produce new $\{x\}$-boundary points. According to Remark 2, this may occur only in two cases.

The first case is adding an $\{x\}$-clause C to $Dis(F_{\boldsymbol{q}},X_{red})$. This may happen after backtracking if C was satisfied or contained a redundant variable. Condition 1 of Proposition 8 makes $\boldsymbol{g}$ contain assignments that prevent C from appearing.

The second case is removing a non-$\{x\}$-clause C from $Dis(F_{\boldsymbol{q}},X_{red})$. This may happen if C contains a literal falsified by an assignment in $\boldsymbol{q}$ and then this assignment is flipped. Condition 2 of Proposition 8 makes $\boldsymbol{g}$ contain assignments guaranteeing that a "mandatory" set of clauses preventing appearance of new $\{x\}$-boundary points is present when D-sequent $(F,X'_{red},\boldsymbol{g}) \rightarrow \{x\}$ is used.

Remark 4: If x is monotone, Condition 2 of Proposition 8 is vacuously true because $\boldsymbol{p_1}$ or $\boldsymbol{p_2}$ does not exist. So one can drop the requirement of Proposition 8 about x being the only variable of X that is not in $Vars(\boldsymbol{q}) \cup X_{red}$. (It is used only when proving that the contribution of non-$\{x\}$-clauses into $\boldsymbol{g}$ specified by Condition 2 is correct. But if x is monotone non-$\{x\}$-clauses are not used when forming $\boldsymbol{g}$.)

C. Notation Simplification for D-sequents of DDS_impl

In the description of *DDS_impl* we will use the notation $\boldsymbol{g} \rightarrow X''$ instead of $(F,X',\boldsymbol{g}) \rightarrow X''$. We do this for two reasons. First, according to Proposition 6, in any D-sequent $(F_{earlier},X',\boldsymbol{g}) \rightarrow X''$, one can replace $F_{earlier}$ with $F_{current}$ where the latter is obtained from the former by adding some resolvent-clauses. Second, whenever *DDS_impl* derives a new D-sequent, X' is the set X_{red} of all unassigned variables of $F_{\boldsymbol{q}}$ already proved redundant. So when we say that $\boldsymbol{g} \rightarrow X''$ holds we mean that $(F,X',\boldsymbol{g}) \rightarrow X''$ does where F is the current formula (i.e. the latest version of F) and X' is X_{red}.

VI. Description of *DDS_impl*

A. Search tree

DDS_impl branches on variables of X of $F(X,Y)$ building a search tree. The current path of the search tree is specified by partial assignment $\boldsymbol{q}$. *DDS_impl* does not branch on variables proved redundant for current $\boldsymbol{q}$. Backtracking to the root of the search tree means derivation of D-sequent $\emptyset \rightarrow X$ (here we use the simplified notation of D-sequents, see Subsection V-C). At this point, *DDS_impl* terminates. We will denote the last variable assigned in $\boldsymbol{q}$ as $Last(\boldsymbol{q})$.

Let x be a branching variable. *DDS_impl* maintains the notion of left and right branches corresponding to the first and second assignment to x respectively. (In the modern SAT-solvers, the second assignment to a branching variable x is implied by a clause C derived in the left branch of x where C is empty in the left branch. A QEP-solver usually deals with satisfiable formulas. If the left branch of x contains a satisfying assignment, clause C above does not exist.)

Although *DDS_impl* distinguishes between decision and implied assignments (and employs BCP procedure), no notion of decision levels is used. When an assignment (decision or implied) is made to a variable, the depth of the current path increases by one and a new node of the search tree is created at the new depth. The current version of *DDS_impl* maintains a single search tree (no restarts are used).

B. Leaf Condition, Active D-sequents, Branch Flipping

Every assignment made by *DDS_impl* is added to $\boldsymbol{q}$. The formula *DDS_impl* operates on is $Dis(F_{\boldsymbol{q}}, X_{red})$. When a monotone variable x appears in $Dis(F_{\boldsymbol{q}}, X_{red})$, it is added to the set X_{red} of redundant variables of $F_{\boldsymbol{q}}$ and the $\{x\}$-clauses are removed from $Dis(F_{\boldsymbol{q}}, X_{red})$. For every variable x' of X_{red} there is one D-sequent $\boldsymbol{g} \rightarrow \{x'\}$ where $\boldsymbol{g} \leq \boldsymbol{q}$. We will call such a D-sequent **active**. (Partial assignment $\boldsymbol{g}$ is in general different for different variables of X_{red}.) Let $\boldsymbol{D^{act}_{seq}}$ denote the current set of active D-sequents.

DDS_impl keeps adding assignments to $\boldsymbol{q}$ until every variable of F is either assigned (i.e. in $Vars(\boldsymbol{q})$) or redundant (i.e. in X_{red}). We will refer to this situation as **the leaf condition**. The appearance of an empty clause in $Dis(F_{\boldsymbol{q}}, X_{red})$ is one of the cases where the leaf condition holds.

If *DDS_impl* is in the left branch of x (where $x = Last(\boldsymbol{q})$) when the leaf condition occurs, *DDS_impl* starts the right branch by flipping the value of x. For every variable x' of X_{red}, *DDS_impl* checks if $\boldsymbol{g}$ of D-sequent $\boldsymbol{g} \rightarrow \{x'\}$ contains an assignment to x. If it does, then this D-sequent is not true any more. Variable x' is removed from X_{red} and $\boldsymbol{g} \rightarrow \{x'\}$ is removed from D^{act}_{seq} and added to the set $\boldsymbol{D^{inact}_{seq}}$ of inactive D-sequents. Every $\{x'\}$-clause C discarded from $Dis(F_{\boldsymbol{q}}, X_{red})$ due to redundancy of x' is recovered (unless C contains a variable that is still in X_{red}).

C. Merging Results of Left and Right Branches

If *DDS_impl* is in the right branch of x (where $x = Last(\boldsymbol{q})$) when the leaf condition occurs, then *DDS_impl* does the following. First *DDS_impl* unassigns x. Then *DDS_impl* examines the list of variables removed from X_{red} after flipping the value of x. Let x' be such a variable and S_{left} and S_{right} be the D-sequents of x' that were active in the left and right branch respectively. (Currently S_{left} is in D^{inact}_{seq}). If S_{right} does not depend on x, then S_{left} is just removed from D^{inact}_{seq} and S_{right} remains in the set of active D-sequents D^{act}_{seq} . Otherwise, S_{left} is resolved with S_{right} on x. Then S_{left} and S_{right} are removed from D^{inact}_{seq} and D^{act}_{seq} respectively, and the resolvent is added to D^{act}_{seq} and becomes a new active D-sequent of x'.

Then *DDS_impl* makes variable x itself redundant. (At this point every variable of X but x is either assigned or redundant.) To this end, *DDS_impl* eliminates all $\{x\}$-removable boundary points from $Dis(F_{\boldsymbol{q}}, X_{red})$ by adding some resolvents on variable x. This is done as follows. First, a CNF H is formed from $Dis(F_{\boldsymbol{q}}, X_{red})$ by removing all the $\{x\}$-clauses and adding a set of "directing" clauses H_{dir}. The latter is satisfied by an assignment $\boldsymbol{p}$ iff at least one clause C' of $Dis(F_{\boldsymbol{q}}, X_{red})$ with literal x and one clause C'' with literal $\overline{x}$ is falsified by $\boldsymbol{p}$. (How H_{dir} is built is described in [12].) The satisfiability of H is checked by calling a SAT-solver. If H is satisfied by an assignment $\boldsymbol{p}$, then the latter is an $\{x\}$-removable boundary point of $Dis(F_{\boldsymbol{q}}, X_{red})$. It is eliminated by adding a resolvent C on x to $Dis(F_{\boldsymbol{q}}, X_{red})$. (Clause C is also added to H). Otherwise, the SAT-solver returns a proof *Proof* that H is unsatisfiable.

Finally, a D-sequent $\boldsymbol{g} \rightarrow \{x'\}$ is generated satisfying the conditions of Proposition 8. To make $\boldsymbol{g}$ satisfy the second condition of Proposition 8, *DDS_impl* uses *Proof* above. Namely, every assignment falsifying a literal of a clause of $Dis(F_{\boldsymbol{q}}, X_{red})$ used in *Proof* is included in $\boldsymbol{g}$.

D. Pseudocode of DDS_impl

The main loop of *DDS_impl* is shown in Figure 1. *DDS_impl* can be in one of the six states listed in Figure 1. *DDS_impl* terminates when it reaches the state *Finish*. Otherwise, *DDS_impl* calls the procedure corresponding to the current state. This procedure performs some actions and returns the next state of *DDS_impl*.

DDS_impl starts in the *BCP* state in which it runs the *bcp* procedure (Figure 3). Let C be a unit clause of $Dis(F_{\boldsymbol{q}}, X_{red})$ where $Vars(C) \subseteq X$. As we mentioned in Subsection V-B, *DDS_impl* adds D-sequent $\boldsymbol{g} \rightarrow X''$ to D^{act}_{seq} where $X'' = X \setminus (Vars(\boldsymbol{q}) \ \cup \ X_{red})$ and $\boldsymbol{g}$ is the minimal assignment falsifying C. This D-sequent corresponds to the (left) branch of the search tree. In this branch, the only literal of C is falsified, which makes the leaf condition true.

If a conflict occurs during BCP, *DDS_impl* switches to the state *Conflict* and calls a procedure that generates a conflict clause C_{cnfl} (Figure 5). Then *DDS_impl* backtracks to the first node of the search tree at which C_{cnfl} becomes unit.

If BCP does not lead to a conflict, *DDS_impl* switches to the state *Decision_Making* and calls a decision making procedure (Figure 2). This procedure first looks for monotone variables. (X_{mon} of Figure 2 denotes the set of new monotone variables.)

If after processing monotone variables every unassigned variable is redundant *DDS_impl* switches to the *Backtracking* state (and calls the *backtrack* procedure, see Figure 6). Otherwise, a new assignment is made and added to $\boldsymbol{q}$.

If *DDS_impl* backtracks to the right branch of x (where x may be an implied or a decision variable), it switches to the state BPE (Boundary Point Elimination) and calls the *bpe* procedure (Figure 4). This procedure merges results of left and right branches as described in Subsection VI-C.

E. Example

Example 3: Let $F(X,Y)$ consist of clauses: $C_1 = x_1 \vee y_1$, $C_2 = \overline{x}_1 \vee \overline{x}_2 \vee y_2$, $C_3 = x_1 \vee x_2 \vee \overline{y}_3$. Let us consider how *DDS_impl* builds formula $F^*(Y)$ equivalent to $\exists X.F(X,Y)$. Originally, $\boldsymbol{q}$, X_{red}, D^{act}_{seq}, D^{inact}_{seq} are empty. Since F does not have a unit clause, *DDS_impl* switches to the state *Decision_Making*. Suppose *DDS_impl* picks x_1 for branching and first makes assignment $x_1 = 0$. At this point, $\boldsymbol{q} = (x_1 = 0)$, clause C_2 is satisfied and $F_{\boldsymbol{q}} = y_1 \wedge (x_2 \vee \overline{y}_3)$.

Before making next decision, *DDS_impl* processes the monotone variable x_2. First the D-sequent $\boldsymbol{g} \rightarrow \{x_2\}$ is derived and added to D^{act}_{seq} where $\boldsymbol{g} = (x_1 = 0)$. (The appearance of the assignment $(x_1 = 0)$ in $\boldsymbol{g}$ is due to Proposition 8. According to Condition 1, $\boldsymbol{g}$ has to contain assignments that keep satisfied or redundant the $\{x_2\}$- clauses that are not currently in $F_{\boldsymbol{q}}$. The only $\{x_2\}$-clause that is not in $F_{\boldsymbol{q}}$ is C_2. It is satisfied by $(x_1 = 0)$.) Variable x_2 is added to X_{red} and clause $x_2 \vee \overline{y}_3$ is removed from $F_{\boldsymbol{q}}$ as containing redundant variable x_2. So $Dis(F_{\boldsymbol{q}}, X_{red}) = y_1$.

Since X has no variables to branch on (the leaf condition), *DDS_impl* backtracks to the last assignment $x_1 = 0$ and starts the right branch of x_1. So $\boldsymbol{q} = (x_1 = 1)$. Since the D-sequent $(x_1 = 0) \rightarrow \{x_2\}$ is not valid now, it is moved from D^{act}_{seq} to D^{inact}_{seq}. Since x_2 is not redundant anymore it is removed from X_{red} and the clause C_2 is recovered in $F_{\boldsymbol{q}}$ which is currently equal to $\overline{x}_2 \vee y_2$ (because C_1 and C_3 are satisfied by $\boldsymbol{q}$).

Since x_2 is monotone again, D-sequent $(x_1 = 1) \rightarrow \{x_2\}$ is derived, x_2 is added to X_{red} and C_2 is removed from $F_{\boldsymbol{q}}$. So $Dis(F_{\boldsymbol{q}}, X_{red}) = \emptyset$. At this point *DDS_impl* backtracks to the right branch of x_1 and switches to the state *BPE*.

In the *BPE* state, x_1 is unassigned. C_1 satisfied by assignment $x_1 = 1$ is recovered. C_2 and C_3 (removed due to redundancy of x_2) are not recovered. The reason is that redundancy of x_2 has been proved in both branches of x_1. So x_2 stays redundant due to generation of D-sequent $\emptyset \rightarrow \{x_2\}$ obtained by resolving D-sequents $(x_1 = 0) \rightarrow \{x_2\}$ and $(x_1 = 1) \rightarrow \{x_2\}$ on x_1. So $Dis(F_{\boldsymbol{q}}, X_{red}) = C_1$. D-sequent $\emptyset \rightarrow \{x_2\}$ replaces $(x_1 = 1) \rightarrow \{x_2\}$ in D^{act}_{seq}. D-sequent $(x_1 = 0) \rightarrow \{x_2\}$ is removed from D^{inact}_{seq}.

Then *DDS_impl* is supposed to make x_1 redundant by adding resolvents on x_1 that eliminate $\{x_1\}$-removable boundary points of $Dis(F_{\boldsymbol{q}}, X_{red})$. Since x_1 is monotone in $Dis(F_{\boldsymbol{q}}, X_{red})$ it is already redundant. So D-sequent $\emptyset \rightarrow \{x_1\}$ is derived and x_1 is added to X_{red}. Since $\boldsymbol{q}$ is currently empty, *DDS_impl* terminates returning an empty set of clauses as a CNF formula $F^*(Y)$ equivalent to $\exists X.F(X,Y)$.

```
// Given F(X,Y), DDS_impl returns F*(Y)
// such that F*(Y) ≡ ∃X.F(X,Y)
// q is a partial assignment to vars of X
// States of DDS_impl are Finish, BCP, BPE,
// Decision_Making, Conflict, Backtracking

DDS_impl (F,X,Y)
 {while (True)
   if (state == Finish)
    return(Dis(F,X));
   if (state == Non_Finish_State)
    {state = state_procedure(q,other_params);
    continue;}}
```

Fig. 1. *Main loop of DDS_impl*

```
decision_making(q, F, X, X_red, D^act_seq)
 {(X_mon, D^act_seq (X_mon)) ← find_monot_vars(F, X);
 D^act_seq = D^act_seq (X_red) ∪ D^act_seq (X_mon);
 X_red = X_red ∪ X_mon;
 if (X == X_red ∪ Vars(q))
   if (Vars(q) == ∅) return(Finish);
   else return(Backtracking);
 F = Dis(F, X_mon);
 assgn(x) ← pick_assgn(F, X);
 q' = q ∪ assgn(x);
 return(BCP);}
```

Fig. 2. *Pseudocode of the decision_making procedure*

Proposition 9: *DDS_impl* is sound and complete.

VII. Compositionality of *DDS_impl*

Let $F(X,Y) = F_1(X_1,Y_1) \wedge \ldots \wedge F_k(X_k,Y_k)$ where $(X_i \cup Y_i) \cap (X_j \cup Y_j) = \emptyset$, $i \neq j$. As we mentioned in the introduction, the formula $F^*(Y)$ equivalent to $\exists X.F(X,Y)$ can be built as $F^*_1 \wedge \ldots \wedge F^*_k$ where $F^*_i(Y_i) \equiv \exists X_i.F_i(X_i,Y_i)$.

We will say that a QEP-solver is **compositional** if it reduces the problem of finding F^* to k independent subproblems of

```
bcp(q, F, C_unsat)
 {(answer, F, q, C_unsat, D^act_seq) ← run_bcp(q, F);
 if (answer == unsat_clause) return(Conflict);
 else return(Decision_Making);}
```

Fig. 3. *Pseudocode of the bcp procedure*

```
bpe(q, F, X_red, D^act_seq, D^inact_seq)
 {x = Last(q);
 (q, F) ← unassign(q, F, x);
 (F, Proof) ← elim_bnd_pnts(F, x);
 optimize(Proof);
 (D^act_seq, D^inact_seq) ← resolve(D^act_seq, D^inact_seq, x);
 D^act_seq ({x}) = gen_Dsequent(q, Proof);
 D^act_seq = D^act_seq (X_red) ∪ D^act_seq ({x});
 X_red = X_red ∪ {x};
 F = Dis(F, {x});
 if (Vars(q) == ∅) return(Finish);
 else return(Backtracking);}
```

Fig. 4. *Pseudocode of the bpe procedure*

cnfl_processing$(\boldsymbol{q}, F, C_{unsat})$
$\{(\boldsymbol{q}, F, C_{cnfl}) \leftarrow gen\_cnfl\_clause(\boldsymbol{q}, F, C_{unsat})$;
$F = F \cup C_{cnfl}$;
if ($C_{cnfl} == \emptyset$) return(*Finish*);
$x = Last(\boldsymbol{q})$;
if ($left\_branch(x)$) return(*BCP*);
else return(*BPE*);}

Fig. 5. *Pseudocode of the cnfl_processing procedure*

backtrack$(\boldsymbol{q}, F, X_{red}, D^{act}_{seq}, D^{inact}_{seq})$
$\{x = Last(\boldsymbol{q})$;
if ($right\_branch(x)$) return(*BPE*);
$\boldsymbol{q} = flip\_assignment(\boldsymbol{q}, x)$;
$X' = find\_affected\_red\_vars(D^{act}_{seq}(X_{red}), x)$;
$D^{act}_{seq} = D^{act}_{seq}(X_{red}) \setminus D^{act}_{seq}(X')$;
$D^{inact}_{seq} = D^{inact}_{seq}(X_{red}) \cup D^{act}_{seq}(X')$;
$X_{red} = X_{red} \setminus X'$;
$F = recover\_clauses(F, X')$;
return(*BCP*);}

Fig. 6. *Pseudocode of the backtrack procedure*

building F^*_i. The DP-procedure [5] is compositional (clauses of F_i and F_j, $i \neq j$ cannot be resolved with each other). However, it may generate a huge number of redundant clauses. A QEP-solver based on enumeration of satisfying assignments is not compositional. (The number of blocking clauses, i.e. clauses eliminating satisfying assignments of F, is exponential in k). A QEP-solver based on BDDs [3] is compositional but only for variable orderings where variables of F_i and F_j, $i \neq j$ do not interleave.

Proposition 10: DDS_impl is compositional *regardless* of how branching variables are chosen.

The fact that *DDS_impl* is compositional regardless of branching choices is important in practice. Suppose $F(X, Y)$ does not have independent subformulas but such subformulas appear in branches of the search tree. A BDD-based QEP-solver may not be able to handle this case because a BDD maintains one *global* variable order (and different branches may require different variable orders). *DDS_impl* does not have such a limitation. It will *automatically* use its compositionality whenever independent subformulas appear.

VIII. Experimental Results

TABLE I
Results for the sum-of-counters experiment

#bits	#coun-ters	#state vars	*Enum-SA* (s.)	Inter-pol. *Pico.* (s.)	Inter-pol. *Mini.* (s.)	*DDS_impl* rand. (s.)	*DDS_impl* (s.)
3	5	15	12.1	0.0	0.0	0.0	0.0
4	20	80	$*$	0.4	0.1	0.5	0.4
5	40	200	$*$	42	26	7	5
6	80	480	$*$	$*$	$*$	101	67

Instances marked with '$*$' exceeded the time limit (2 hours).

In this section, we give results of some experiments with an implementation of *DDS_impl*. The objectives of our experiments were a) to emphasize the compositionality of *DDS_impl*; b) to compare *DDS_impl* with a QEP-solver based on enumeration of satisfying assignments. As a such QEP-solver we used an implementation of the algorithm recently introduced at CAV-11 [2] (courtesy of Andy King). (We will refer to this QEP-solver as *EnumSA*). For the sake of completeness we also compared *DDS_impl* and *EnumSA* with our implementation of the DP procedure.

TABLE II
Experiments with model checking formulas

model che-king mode	*DP*		*EnumSA*		*DDS_impl*	
	solved (%)	time (s.)	solved (%)	time (s.)	solved (%)	time (s.)
forward	416 (54%)	664	425 (56%)	466	531 (70%)	3,143
backward	47 (6%)	13	97 (12%)	143	559 (73%)	690

The time limit is 1 min.

Our current implementation of *DDS_impl* is not particularly well optimized yet and written just to satisfy the two objectives above. For example, to simplify the code, the SAT-solver employed to find boundary points does not use fast BCP (watched literals). More importantly, the current version of *DDS_impl* lacks important features that should have a dramatic impact on its performance. For example, to simplify memory management, *DDS_impl* does not currently *reuse* D-sequents. As soon as two D-sequents are resolved (to produce a new D-sequent) they are discarded.

To verify the correctness of results of *DDS_impl* we used two approaches. If an instance $\exists X.F(X, Y)$ was solved by *EnumSA* we simply checked the CNF formulas $F^*(Y)$ produced by *DDS_impl* and *EnumSA* for equivalence. Otherwise, we applied a two-step procedure. First, we checked that every clause of F^* was implied by F. Second, we did random testing to see if F^* missed some clauses. Namely, we randomly generated assignments $\boldsymbol{y}$ satisfying F^*. For every $\boldsymbol{y}$ we checked if it could be extended to ($\boldsymbol{x}$,$\boldsymbol{y}$) satisfying F. (If no such extension exists, then F^* is incorrect.)

In the first experiment (Table I), we considered a circuit N of k independent m-bit counters. Each counter had an independent input variable. The property we checked (further referred to as ξ) was $Num(Cnt_1) + \ldots + Num(Cnt_k) < R$. Here $Num(Cnt_i)$ is the number specified by the outputs of i-th counter and R is a constant equal to $k * (2^m - 1) + 1$. Since, the maximum number that appears at the outputs of a counter is $2^m - 1$, property ξ holds. Since the counters are independent of each other, the state space of N is the Cartesian product of the k state spaces of individual counters. However, property ξ itself is not compositional (one cannot verify it by solving k-independent subproblems), which makes verification harder.

The first two columns of Table I give the value of m and k of four circuits N. The third column specifies the number of state variables (equal to $m * k$). In this experiment, we applied *EnumSA* and *DDS_impl* to verify property ξ using forward model checking. In either case, the QEP-solver was used to compute CNF formula $RS^*(S_{next})$ specifying the next set of reachable states. It was obtained from formula $\exists S_{curr} \exists X.Tr(S_{curr}, S_{next}, X) \wedge RS_p(S_{curr})$ by quantifier elimination. Here *Tr* is a CNF formula representing the transition relation and $RS_p(S_{curr})$ specifies the set of states reached in p iterations. $RS_{p+1}(S_{curr})$ was computed as a CNF formula equivalent to $RS_p(S_{curr}) \vee RS^*(S_{curr})$.

We also estimated the complexity of verifying the examples of Table I by interpolation [16]. Namely, we used *Picosat 913* and *Minisat 2.0* for finding a proof that ξ holds for 2^{m-1} iterations (the diameter of circuits N of Table I is 2^m, $m = 3, 4, 5, 6$). Such a proof is used in the method of [16] to extract an interpolant. So, in Table I, we give only the time necessary to find the first interpolant.

Table I shows that *EnumSA* does not scale well (the number of blocking clauses one has to generate for the formulas of Table I is exponential in the number of counters). Computation of interpolants scales much better, but *Picosat* and *Minisat* failed to compute a proof for the largest example in 2 hours.

The last two columns of Table I give the performance of *DDS_impl* when branching variables were chosen randomly (next to last column) and heuristically (last column). In either case, *DDS_impl* shows good scalability explained by the fact that *DDS_impl* is compositional. Moreover, the fact that the choice of branching variables is not particularly important means that *DDS_impl* has a "stronger" compositionality than BDD-based QEP-solvers. The latter are compositional only for particular variable orderings.

In second and third experiments (Table II) we used the 758 model checking benchmarks of HWMCC'10 competition [19]. In the second experiment, (the first line of Table II) we used *DP*, *EnumSA* and *DDS_impl* to compute the set of states reachable in the first transition. In this case one needs to find CNF formula $F^*(Y)$ equivalent to $\exists X.F(X,Y)$ where $F(X,Y)$ specifies the transition relation and the initial state. Then $F^*(Y)$ gives the set of states reachable in one transition.

In the third experiment, (the second line of Table II) we used the same benchmarks to compute the set of bad states in backward model checking. In this case, formula $F(X,Y)$ specifies the output function and the property (where Y is the set of variables describing the current state). If $F(X,Y)$ evaluates to 1 for some assignment $(\boldsymbol{x},\boldsymbol{y})$ to $X \cup Y$, the property is broken and the state specified by $\boldsymbol{y}$ is "bad". The formula $F^*(Y)$ equivalent to $\exists X.F(X,Y)$ specifies the set of bad states.

Table II shows the number of benchmarks solved by each program and the percentage of this number to 758. Besides the time taken by each program for the *solved* benchmarks is shown. *DDS_impl* solved more benchmarks than *EnumSA* and *DP* in forward model checking and dramatically more benchmarks in the backward model checking. *DDS_impl* needed more time than *DP* and *EnumSA* because typically the benchmarks solved only by *DDS_impl* were the most time consuming.

IX. Background

The notion of boundary points was introduced in [13]. for pruning the search tree (in the context of SAT-solving). The relation between a resolution proof and the process of elimination of boundary points was discussed in [14], [12]. The previous papers considered only the notion of $\{z\}$-boundary of formula $G(Z)$ where z is a variable of Z. In the present paper, we consider Z'-boundary points where Z' is an arbitrary subset of Z. (This extension is not trivial and at the same time crucial for the introduction of D-sequents.)

The idea of a QEP-solver based on enumerating satisfying assignments was introduced in [17]. It has been further developed in [15], [7], [2]. In [16] it was shown how one can avoid QEP-solving in reachability analysis by building interpolants. Although, this direction is very promising, interpolation based methods have to overcome the following problem. In the current implementations, interpolants are extracted from resolution proofs. Unfortunately, modern SAT-solvers are still not good enough to take into account the high-level structure of a formula. (An example of that is given in Section VIII.) So proofs they find and the interpolants extracted from those proofs may have poor quality.

Note that our notion of redundancy of variables is different from observability related notions of redundancy. For instance, in contrast to the notion of careset [6], if a CNF formula $G(Z)$ is satisfiable, *all* the variables of Z are redundant in the formula $\exists Z.G(Z)$ according to our definition. (G may have a lot of boundary points, but none of them is removable. So $\exists Z.G(Z)$ is equivalent to an empty CNF formula. Of course, to prove the variables of Z redundant, one has to derive D-sequent $\emptyset \rightarrow Z$.)

X. Conclusion

We present a new method for eliminating existential quantifiers from a Boolean CNF formula $\exists X.F(X,Y)$. The essence of this method is to add resolvent clauses to F and record the decreasing dependency on variables of X by dependency sequents (D-sequents). An algorithm based on this method (called DDS, Derivation of D-Sequents) terminates when it derives the D-sequent saying that the variables of X are redundant. Using this termination condition may lead to a significant performance improvement in comparison to the algorithms based on enumerating satisfying assignments. This improvement may be even exponential (e.g. if a CNF formula is composed of independent subformulas.)

Our preliminary experiments with a very simple implementation show the promise of DDS. At the same time, DDS needs further study. Here are some directions for future research: a) decision making heuristics; b) reusing D-sequents; c) efficient data structures; d) getting information about the structure of the formula (specified as a sequence of D-sequents to derive).

XI. Acknowledgment

This work was funded in part by NSF grant CCF-1117184 and SRC contract 2008-TJ-1852.

References

[1] A.Biere,"PicoSAT Essentials", *JSAT*, vol.4, no.2-4, pp.75-97, 2008.
[2] J.Brauer, A. King, and J. Kriener, "Existential Quantification as Incremental SAT", to appear in *Proc. CAV-2011*.

[3] R.Bryant,"Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. on Computers*, vol.C-35, no.8, pp.677-691, 1986.
[4] E.Clarke,O. Grumberg,and D. Peled. *Model Checking*, MIT Press, 2000.
[5] M.Davis, and H.Putnam, "A Computing Procedure for Quantification Theory", *J. ACM*, vol.7, no.3, pp.201-215, July, 1960.
[6] M.Ganai,"Propelling SAT and SAT-based BMC using Careset", in *Proc. FMCAD-2010*, pp.231-238.
[7] M.Ganai, A.Gupta, and P. Ashar,"Efficient SAT-based unbounded symbolic model checking using circuit cofactoring", in *Proc. ICCAD-2004*, pp.510-517.
[8] E.Goldberg, P.Manolios. "Quantifier Elimination by Dependency Sequents", Patent application no. 13/341,564, filing date 30 Dec., 2011.
[9] E.Goldberg, P.Manolios. "Quantifier Elimination by Dependency Sequents", arXiv:1201.5653v1 [cs.LO]. (A technical report archived at Cornell University Library.)
[10] E.Goldberg, P.Manolios. "Decision Procedures for System-Level Verification", Annual (SRC) review presentation 2011 (P059425), March 24, 2011.
[11] E.Goldberg, P.Manolios. "Quantifier Elimination by Dependency Sequents", SRC, Pub. P060164, June 4, 2011.
[12] E.Goldberg, and P.Manolios, "SAT-solving Based on Boundary Point Elimination", in *Proc. HVC-2010*, LNCS vol.6504, pp.93-111.
[13] E.Goldberg, M.Prasad, and R.Brayton, "Using Problem Symmetry in Search Based Satisfiability Algorithms", in *DATE-2002*, pp. 134-141.
[14] E.Goldberg,"Boundary points and resolution", in *Proc. SAT-2009*, LNCS vol.5584, pp.147-160.
[15] H.Jin, and F.Somenzi,"Prime clauses for fast enumeration of satisfying assignments to Boolean circuits", in *Proc. DAC-2005*, pp. 750-753.
[16] K.McMillan,"Interpolation and SAT-Based Model Checking", in *Proc. CAV-2003*, LNCS, vol. 2725, pp.1-13.
[17] K.McMillan,"Applying SAT Methods in Unbounded Symbolic Model Checking", in *Proc. CAV-2002*, pp.250-264.
[18] K.McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.
[19] HWMCC-2010 benchmarks, http://fmv.jku.at/hwmcc10/benchmarks.html

APPENDIX

PROOFS OF SECTION II

Proposition 1: Point $\boldsymbol{p}$ is a Z'-removable boundary point of a CNF formula $G(Z)$ iff no point $\boldsymbol{p}^*$ obtained from $\boldsymbol{p}$ by changing values of (some) variables of Z' satisfies G.

Proof: If part. Let us partition G into G_1 and G_2 where G_1 is the set of Z'-clauses and G_2 is the set of of non-Z'-clauses. By definition, $\boldsymbol{p}$ is a Z''-boundary point where $Z'' \subseteq Z'$. So $\boldsymbol{p}$ satisfies G_2.

Let C be the clause such that

- $Vars(C) = Z \setminus Z'$,
- C is falsified by $\boldsymbol{p}$.

Clause C is implied by G_1. Indeed, assume the contrary i.e. there exists $\boldsymbol{p}^*$ for which $G_1(\boldsymbol{p}^*)$=1 and $C(\boldsymbol{p}^*)$=0. Note that since $\boldsymbol{p}^*$ falsifies C, it can be different from $\boldsymbol{p}$ only in assignments to $Z \setminus Z'$. Then, there is a point $\boldsymbol{p}^*$ obtained by flipping values of Z' that satisfies G_1. But since $\boldsymbol{p}^*$ has the same assignments to variables of $Z \setminus Z'$ as $\boldsymbol{p}$, it satisfies G_2 too. So $\boldsymbol{p}^*$ is obtained by flipping assignments of Z' and satisfies G, which contradicts the assumption of the proposition at hand. So C is implied by G_1. Since C satisfies the conditions of Definition 5, $\boldsymbol{p}$ is a Z'-removable boundary point.

Only if part. Assume the contrary. That is there is clause C satisfying the conditions of Definition 5 and there is a point $\boldsymbol{p}^*$ obtained from $\boldsymbol{p}$ by flipping values of variables of Z' that satisfies G. Then $\boldsymbol{p}^*$ also satisfies the set G_1 of Z'-clauses of G. Since C is implied by G_1, then C is satisfied by $\boldsymbol{p}^*$ too. Since $\boldsymbol{p}$ and $\boldsymbol{p}^*$ have identical assignments to the variables of $Z \setminus Z'$, then C is also satisfied by $\boldsymbol{p}$. However this contradicts one of the conditions of Definition 5 assumed to be true.

PROOFS OF SECTION III

Lemma 1: Let $\boldsymbol{p'}$ be a $\{z\}$-boundary point of CNF formula $G(Z)$ where $z \in Z$. Let $\boldsymbol{p''}$ be obtained from $\boldsymbol{p'}$ by flipping the value of z. Then $\boldsymbol{p''}$ either satisfies F or it is also a $\{z\}$-boundary point.

Proof: Assume the contrary i.e. $\boldsymbol{p''}$ falsifies a clause C of G that does not have a variable of z. (And so $\boldsymbol{p''}$ is neither a satisfying assignment nor a $\{z\}$-boundary point of G.) Since $\boldsymbol{p'}$ is different from $\boldsymbol{p''}$ only in the value of z, it also falsifies C. Then $\boldsymbol{p'}$ is not a $\{z\}$-boundary point of G. Contradiction.

Proposition 2: Let $G(Z)$ be a CNF formula and z be a monotone variable of F. (That is clauses of G contain the literal of z of only one polarity.) Then z is redundant in G.

Proof: Let us consider the following two cases.

- $G(Z)$ does not have a $\{z\}$-boundary point. Then the proposition holds.
- $G(Z)$ has a $\{z\}$-boundary point $\boldsymbol{p'}$. Note that the clauses of G falsified by $\boldsymbol{p'}$ have the same literal $l(z)$ of variable z. Let $\boldsymbol{p''}$ be the point obtained from $\boldsymbol{p'}$ by flipping the value of z. According to Lemma 1, one needs to consider only the following two cases.
  - $\boldsymbol{p''}$ satisfies G. Then $\boldsymbol{p'}$ is not a $\{z\}$-removable boundary point. This implies that $\boldsymbol{p'}$ is not a removable boundary point of G either (see Remark 1). So the proposition holds.
  - $\boldsymbol{p''}$ falsifies only the clauses of G with literal $\overline{l(z)}$. (Point $\boldsymbol{p''}$ cannot falsify a clause with literal $l(z)$.) Then G has literals of z of both polarities and z is not a monotone variable. Contradiction.

Proposition 3: Let $F(X,Y)$ be a CNF formula and X' be a subset of X. Then $\exists X.F(X,Y) \equiv \exists (X \setminus X').Dis(F,X')$ iff the variables of X' are redundant in F.

Proof: Denote by X'' the set $X \setminus X'$ and by $F^*(X'',Y)$ the formula $Dis(F,X')$.

If part. Assume the contrary i.e. the variables of X' are redundant but $\exists X.F(X,Y) \not\equiv \exists X''.F^*(X'',Y)$. Let $\boldsymbol{y}$ be an assignment to Y such that $\exists X.F(X,\boldsymbol{y}) \neq \exists X''.F^*(X'',\boldsymbol{y})$. One has to consider the following two cases.

- $\exists X.F(X,\boldsymbol{y}) = 1$, $\exists X''.F^*(X'',\boldsymbol{y}) = 0$. Then there exists an assignment $\boldsymbol{x}$ to X such that ($\boldsymbol{x}$,$\boldsymbol{y}$) satisfies F. Since every clause of F^* is in F, formula F^* is also satisfied by ($\boldsymbol{x''}$,$\boldsymbol{y}$) where $\boldsymbol{x''}$ consists of the assignments of $\boldsymbol{x}$ to variables of X''. Contradiction.
- $\exists X.F(X,\boldsymbol{y}) = 0$, $\exists X''.F^*(X'',\boldsymbol{y}) = 1$. Then there exists an assignment $\boldsymbol{x''}$ to variables of X'' such that ($\boldsymbol{x''}$,$\boldsymbol{y}$) satisfies F^*. Let $\boldsymbol{x}$ be an assignment to X obtained from $\boldsymbol{x''}$ by arbitrarily assigning variables of X'. Since $F(X,\boldsymbol{y}) \equiv 0$, point ($\boldsymbol{x}$,$\boldsymbol{y}$) falsifies F. Since $F^*(\boldsymbol{x},\boldsymbol{y}) = 1$ and every clause of F that is not F^* is an X'-clause, ($\boldsymbol{x}$,$\boldsymbol{y}$) is an X'^*-boundary point of F. Since

$F(X,\boldsymbol{y}) \equiv 0$, ($\boldsymbol{x}$,$\boldsymbol{y}$) is removable. Hence the variables of X' are not redundant in F. Contradiction.

Only if part. Assume the contrary i.e. $\exists X.F(X,Y) \equiv \exists X''.F^*(X'',Y)$ but the variables of X' are not redundant in F. Then there is an X'^* boundary point $\boldsymbol{p}$=($\boldsymbol{x}$,$\boldsymbol{y}$) of F where $X'^* \subseteq X'$ that is removable in F. Since $\boldsymbol{p}$ is a boundary point, $F(\boldsymbol{p}) = 0$. Since $\boldsymbol{p}$ is removable, $\exists X.F(X,\boldsymbol{y}) = 0$. On the other hand, since $\boldsymbol{p}$ falsifies only X'-clauses of F, it satisfies F^*. Then the point ($\boldsymbol{x''}$,$\boldsymbol{y}$) obtained from $\boldsymbol{p}$ by dropping the assignments to X' satisfies F^*. Hence $\exists X''.F^*(X'',\boldsymbol{y}) = 1$ and so $\exists X.F(X,\boldsymbol{y}) \neq \exists X''.F^*(X'',\boldsymbol{y})$. Contradiction.

PROOFS OF SECTION IV

Definition 14: Point $\boldsymbol{p}$ is called a **Z'-unremovable boundary point** of $G(Z)$ where $Z' \subseteq Z$ if $\boldsymbol{p}$ is a Z''-boundary point where $Z'' \subseteq Z'$ and clause C of Definition 5 does not exist. (According to Proposition 1 this means that by flipping values of variables of Z' in $\boldsymbol{p}$ one can get a point satisfying G.)

Definition 15: Let $G(Z)$ be a CNF formula and $\boldsymbol{p}$ be an Z'-boundary point of G where $Z' \subseteq Z$. A point $\boldsymbol{p^*}$ is called a Z''-neighbor of $\boldsymbol{p}$ if

- $Z' \subseteq Z''$
- $\boldsymbol{p}$ and $\boldsymbol{p^*}$ are different only in (some) variables of Z''. In other words, $\boldsymbol{p}$ and $\boldsymbol{p^*}$ can be obtained from each other by flipping (some) variables of Z''.

Proposition 4: Let $G(Z)$ be a CNF formula. Let G have no $\{z\}$-removable boundary points. Let C be a clause. Then the formula $G \wedge C$ does not have a $\{z\}$-removable boundary point if at least one of the following conditions hold: a) C is implied by G; b) $z \notin \mathit{Vars}(C)$.

Proof: Let $\boldsymbol{p}$ be a complete assignment to the variables of G (a point) and C be a clause satisfying at least one of the two conditions of the proposition. Assume the contrary i.e. that $\boldsymbol{p}$ is a $\{z\}$-removable boundary point of $G \wedge C$.

Let us consider the following four cases.

1) G($\boldsymbol{p}$)=0, C($\boldsymbol{p}$)=0.
   - Suppose that $\boldsymbol{p}$ is not a $\{z\}$-boundary point of G. Then it falsifies a clause C' of G that is not a $\{z\}$-clause. Then $\boldsymbol{p}$ is not a $\{z\}$-boundary point of $G \wedge C$. Contradiction.
   - Suppose that $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of G. (According to the conditions of the proposition at hand, G cannot have a $\{z\}$-removable boundary point.) This means that the point $\boldsymbol{p'}$ that is the $\{z\}$-neighbor of $\boldsymbol{p}$ satisfies G.
     – Assume that C is not a $\{z\}$-clause. Then $\boldsymbol{p}$ is not a $\{z\}$-boundary point of $G \wedge C$. Contradiction.
     – Assume that C is implied by G. Then C($\boldsymbol{p'}$)=1 and so $\boldsymbol{p'}$ satisfies $G \wedge C$. Then $\boldsymbol{p}$ is still a $\{z\}$-unremovable boundary point of $G \wedge C$. Contradiction.
2) G($\boldsymbol{p}$)=0, C($\boldsymbol{p}$)=1.
   - Suppose that $\boldsymbol{p}$ is not a $\{z\}$-boundary point of G. Then it falsifies a clause C' of G that is not a $\{z\}$-clause. Then $\boldsymbol{p}$ is not a $\{z\}$-boundary point of $G \wedge C$. Contradiction.
   - Suppose that $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of G. This means that the point $\boldsymbol{p'}$ that is the $\{z\}$-neighbor of $\boldsymbol{p}$ satisfies G.
     – Assume that C is not a $\{z\}$-clause. Then C($\boldsymbol{p}$)=C($\boldsymbol{p'}$) and so C($\boldsymbol{p'}$)=1. Then $\boldsymbol{p'}$ satisfies $G \wedge C$ and so $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of $G \wedge C$. Contradiction.
     – Assume that C is implied by G and so C($\boldsymbol{p'}$)=1. Hence $\boldsymbol{p'}$ satisfies $G \wedge C$. Then $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of $G \wedge C$. Contradiction.
3) G($\boldsymbol{p}$)=1, C($\boldsymbol{p}$)=0.
   - If C is implied by G, then we immediately get a contradiction.
   - If C is not a $\{z\}$-clause, then $\boldsymbol{p}$ falsifies a non-$\{z\}$-clause of $G \wedge C$ and so $\boldsymbol{p}$ is not a $\{z\}$-boundary point of $G \wedge C$. Contradiction.
4) G($\boldsymbol{p}$)=1, C($\boldsymbol{p}$)=1. Point $\boldsymbol{p}$ satisfies $G \wedge C$ and so cannot be a $\{z\}$-boundary point of $G \wedge C$. Contradiction.

Proposition 5: Let $G(Z)$ be a CNF formula. Let G have no $\{z\}$-removable boundary points. Let C be a $\{z\}$-clause of G. Then the formula $G' = G \setminus \{C\}$ does not have a $\{z\}$-removable boundary point.

Proof: Let $\boldsymbol{p}$ be a complete assignment to the variables of G (a point). Assume the contrary i.e. that $z \in \mathit{Vars}(C)$ and $\boldsymbol{p}$ is a $\{z\}$-removable boundary point of G'. Let us consider the following three cases.

1) G($\boldsymbol{p}$)=0, C($\boldsymbol{p}$)=0.
   - Suppose that $\boldsymbol{p}$ is not a $\{z\}$-boundary point of G. Then there is clause C' of G that is not a $\{z\}$-clause and that is falsified by $\boldsymbol{p}$. Since C' is different from C (because the former is not a $\{z\}$-clause) it remains in G'. Hence $\boldsymbol{p}$ is not a $\{z\}$-boundary point of G'. Contradiction.
   - Suppose that $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of G. Then its $\{z\}$-neighbor $\boldsymbol{p'}$ satisfies G and hence G'. Then $\boldsymbol{p}$ either satisfies G' (if C is the only $\{z\}$-clause of G falsified by $\boldsymbol{p}$) or $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of G'. In either case, we have a contradiction.
2) G($\boldsymbol{p}$)=0, C($\boldsymbol{p}$)=1.
   - Suppose that $\boldsymbol{p}$ is not a $\{z\}$-boundary point of G. Using the same reasoning as above we get a contradiction.
   - Suppose that $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of G. Then its $\{z\}$-neighbor $\boldsymbol{p'}$ satisfies G and hence G'. Let C' be a $\{z\}$-clause of G falsified by $\boldsymbol{p}$. Since C' is different from C (the latter being satisfied by $\boldsymbol{p}$), it is present in G'. Hence $\boldsymbol{p}$ falsifies G'. Then $\boldsymbol{p}$ is a $\{z\}$-unremovable boundary point of G'. We have a contradiction.

3) $G(\boldsymbol{p})$=1. Then $G'(\boldsymbol{p})$=1 too and so $\boldsymbol{p}$ cannot be a boundary point of G'. Contradiction.

PROOFS OF SECTION V

SUBSECTION: Formula Replacement in a D-sequent

Proposition 6: Let $F^+(X,Y)$ be a CNF formula obtained from $F(X,Y)$ by adding some resolvents of clauses of F. Let $\boldsymbol{q}$ be a partial assignment to variables of X and $X' \subseteq X$. Then the fact that D-sequent $(F,X',\boldsymbol{q}) \rightarrow X''$ holds implies that $(F^+,X',\boldsymbol{q}) \rightarrow X''$ holds too. The opposite is not true.

Proof: First, let us prove that if $(F,X',\boldsymbol{q}) \rightarrow X''$ holds, $(F^+,X',\boldsymbol{q}) \rightarrow X''$ holds too. Let us assume the contrary, i.e. $(F,X',\boldsymbol{q}) \rightarrow X''$ holds but $(F^+,X',\boldsymbol{q}) \rightarrow X''$ does not. According to Definition 10, this means that either

A) variables of X' are not redundant in $F^+_{\boldsymbol{q}}$ or
B) variables of X'' are not redundant in $Dis(F^+_{\boldsymbol{q}},X')$.

CASE A: The fact that the variables of X' are not redundant in $F^+_{\boldsymbol{q}}$ means that there is a removable X'^*-boundary point $\boldsymbol{p}$ of $F^+_{\boldsymbol{q}}$ where $X'^* \subseteq X'$. The fact that the variables of X' are redundant in $F_{\boldsymbol{q}}$ means that $\boldsymbol{p}$ is not a removable X'^*-boundary point of $F_{\boldsymbol{q}}$. Let us consider the three reasons for that.

- $\boldsymbol{p}$ satisfies $F_{\boldsymbol{q}}$. Then it also satisfies $F^+_{\boldsymbol{q}}$ and hence cannot be a boundary point of $F^+_{\boldsymbol{q}}$. Contradiction.
- $\boldsymbol{p}$ is not an X'^*-boundary point of $F_{\boldsymbol{q}}$. That is $\boldsymbol{p}$ falsifies a non-X'-clause C of $F_{\boldsymbol{q}}$. Since $F^+_{\boldsymbol{q}}$ also contains C, point $\boldsymbol{p}$ cannot be an X'^*-boundary point of $F^+_{\boldsymbol{q}}$ either. Contradiction.
- $\boldsymbol{p}$ is an X'^*-boundary point of $F_{\boldsymbol{q}}$ but it is not removable. This means that one can obtain a point $\boldsymbol{p^*}$ satisfying $F_{\boldsymbol{q}}$ by flipping the values of variables of $X \setminus Vars(\boldsymbol{q})$ in $\boldsymbol{p}$. Since $\boldsymbol{p^*}$ also satisfies $F^+_{\boldsymbol{q}}$, one has to conclude that $\boldsymbol{p}$ is not a removable point of $F^+_{\boldsymbol{q}}$. Contradiction.

CASE B: The fact that the variables of X'' are not redundant in $Dis(F^+_{\boldsymbol{q}},X')$ means that there is a removable X''^*-boundary point $\boldsymbol{p}$ of $Dis(F^+_{\boldsymbol{q}},X')$ where $X''^* \subseteq X''$. The fact that the variables of X'' are redundant in $Dis(F_{\boldsymbol{q}},X')$ means that $\boldsymbol{p}$ is not a removable X''^*-boundary point of $Dis(F_{\boldsymbol{q}},X')$.

Here one can reproduce the reasoning of case A). That is one can consider the three cases above describing why $\boldsymbol{p}$ is not an removable X''^*-boundary point of $Dis(F_{\boldsymbol{q}},X')$ and show that each case leads to a contradiction for the same reason as above.

Now we show that if $(F^+,X',\boldsymbol{q}) \rightarrow X''$ holds this does not mean that $(F,X',\boldsymbol{q}) \rightarrow X''$ holds too. Let $F(X,Y)$ be a CNF formula where $X = \{x\}, Y = \{y\}$. Let F consist of clauses C_1,C_2 where $C_1 = x \vee y$ and $C_2 = \overline{x} \vee y$. Let F^+ be obtained from F by adding the unit clause y (that is the resolvent of C_1 and C_2). It is not hard to see that the D-sequent $(F^+,\emptyset,\emptyset) \rightarrow \{x\}$ holds. (The latter does not have any $\{x\}$-boundary points. Hence it cannot have a removable $\{x\}$-boundary point.) At the same time, F has a removable $\{x\}$-boundary point $\boldsymbol{p}$=(x=0,y=0). So the D-sequent $(F,\emptyset,\emptyset) \rightarrow \{x\}$ does not hold.

SUBSECTION: Resolution of D-sequents

Definition 16: Let $F(X,Y)$ be a CNF formula and $X' \subseteq X$. We will say that the variables of X' are **locally redundant** in F if every X''-boundary point $\boldsymbol{p}$ of F where $X'' \subseteq X'$ is X'-removable.

Remark 5: We will call the variables of a set X' **globally redundant** in $F(X,Y)$ if they are redundant in the sense of Definition 7. The difference between locally and globally redundant variables is as follows. When testing if variables of X' are redundant, in either case one checks if every X''-boundary point $\boldsymbol{p}$ of F where $X'' \subseteq X'$ is removable. The difference is in the set variables one is allowed to change. In the case of locally redundant variables (respectively globally redundant variables) one checks if $\boldsymbol{p}$ is X'-removable (respectively X-removable). In other words, in the case of globally variables one is allowed to change variables that are not in X'.

Lemma 2: If variables of X' are locally redundant in a CNF formula $F(X,Y)$ they are also globally redundant there. The opposite is not true.

Proof: See Remark 5.

Lemma 3: Let z be a monotone variable of $G(Z)$. Then variable z is *locally* redundant.

Proof: Let us assume for the sake of clarity that only positive literals of z occur in clauses of G. Let us consider the following two cases:

- Let G have no any $\{z\}$-boundary points. Then the proposition is vacuously true.
- Let $\boldsymbol{p}$ be a $\{z\}$-boundary point. By flipping the value of z from 0 to 1, we obtain an assignment satisfying G. So $\boldsymbol{p}$ is not a removable $\{z\}$-boundary point and to prove that it is sufficient to flip the value of z. Hence z is locally redundant in G.

Lemma 4: Let $F(X,Y)$ be a CNF formula and X' be a subset of variables of X that are globally redundant in F. Let X'' be a non-empty subset of X'. Then the variables of X'' are also globally redundant in F.

Proof: Assume the contrary, i.e. the variables of X'' are not globally redundant in F. Then there is an X''^*-boundary point $\boldsymbol{p}$ where $X''^* \subseteq X''$ that is X-removable. Since X''^* is also a subset of X', the existence of point $\boldsymbol{p}$ means that the variables of X' are not globally redundant in F. Contradiction.

Remark 6: Note that Lemma 4 is not true for locally redundant variables. Let $F(X,Y)$ be a CNF formula and X' be a subset of variables of X that are locally redundant in F. Let X'' be a non-empty subset of X'. Then one cannot claim that the variables of X'' are locally redundant in F. (However it is true that they are globally redundant in F.)

For the rest of the Appendix we will use only the notion of globally redundant variables (introduced by Definition 7).

Definition 17: Let X be a set of Boolean variables. Let C be a clause where $Vars(C) \subseteq X$. Let $Vars(\boldsymbol{q})$ be a partial assignment to variables of X. Denote by $C_{\boldsymbol{q}}$ the clause that is

- equal to 1 (a tautologous clause) if C is satisfied by $\boldsymbol{q}$;

- obtained from C by removing the literals falsified by $\boldsymbol{q}$, if C is not satisfied by $\boldsymbol{q}$.

Definition 18: Let $F(X,Y)$ be a CNF formula and $\boldsymbol{q}$ be a partial assignment to variables of X. Let X' and X'' be subsets of X. We will say that the variables of X'' are **locally irredundant** in $Dis(F_{\boldsymbol{q}},X')$ if every X''^{*}-boundary point of $Dis(F_{\boldsymbol{q}},X')$ where $X''^{*} \subseteq X''$ that is $(X \setminus Vars(\boldsymbol{q}))$-removable in $Dis(F_{\boldsymbol{q}},X')$ is X-unremovable in F. We will say that the variables of X'' **are redundant in $Dis(F_{\boldsymbol{q}},X')$ modulo local irredundancy**.

Remark 7: The fact that variables of X'' are locally irredundant in $Dis(F_{\boldsymbol{q}},X')$ means that the latter has an X''^{*}-boundary point $\boldsymbol{p}$ where $X''^{*} \subseteq X''$ that cannot be turned into a satisfying assignment in the subspace specified by $\boldsymbol{q}$ (because the values of variables of $Vars(\boldsymbol{q})$ cannot be changed). However, $\boldsymbol{p}$ can be transformed into a satisfying assignment if variables of $Vars(\boldsymbol{q})$ are allowed to be changed. This means that $\boldsymbol{p}$ can be eliminated only by an X-clause (implied by F) but cannot be eliminated by a clause depending only on variables of Y. Points like $\boldsymbol{p}$ can be ignored.

Lemma 5: Let $F(X,Y)$ be a CNF formula. Let $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ be partial assignments to variables of X that are resolvable on variable x. Denote by $\boldsymbol{q}$ the partial assignment $Res(\boldsymbol{q_1},\boldsymbol{q_2},x)$ (see Definition 11). Let X_1 (respectively X_2) be the subsets of variables of X already proved redundant in $F_{\boldsymbol{q_1}}$ (respectively $F_{\boldsymbol{q_2}}$). Let the set of variables X^* where $X^* = X_1 \cap X_2$ be non-empty. Then the variables of X^* are redundant in $F_{\boldsymbol{q}}$ modulo local irredundancy.

Proof: Assume that the variables of X^* are not redundant in $F_{\boldsymbol{q}}$ and then show that this irredundancy is local. According to Definition 7, irredundancy of X^* means that there is an X'^{*}-boundary point $\boldsymbol{p}$ where $X'^{*} \subseteq X^*$ that is $(X \setminus Vars(\boldsymbol{q}))$-removable in $F_{\boldsymbol{q}}$. Since $\boldsymbol{p}$ is an extension of $\boldsymbol{q}$, it is also an extension of $\boldsymbol{q_1}$ or $\boldsymbol{q_2}$. Assume for the sake of clarity that $\boldsymbol{p}$ is an extension of $\boldsymbol{q_1}$.

The set of clauses falsified by $\boldsymbol{p}$ in $F_{\boldsymbol{q}}$ and $F_{\boldsymbol{q_1}}$ is specified by the set of clauses of F falsified by $\boldsymbol{p}$. If a clause C of F is satisfied by $\boldsymbol{p}$, then clause $C_{\boldsymbol{q}}$ (see Definition 17) is either

- not in $F_{\boldsymbol{q}}$ (because is C satisfied by $\boldsymbol{q}$) or
- in $F_{\boldsymbol{q}}$ and is satisfied by $\boldsymbol{p}$.

The same applies to the relation between clause $C_{\boldsymbol{q_1}}$ and CNF formula $F_{\boldsymbol{q_1}}$. Let C be a clause falsified by $\boldsymbol{p}$. Then C cannot be satisfied by $\boldsymbol{q}$ and so the clause $C_{\boldsymbol{q}}$ is in $F_{\boldsymbol{q}}$ The same applies to $C_{\boldsymbol{q_1}}$ and $F_{\boldsymbol{q_1}}$.

Since $\boldsymbol{p}$ falsifies the same clauses of F in $F_{\boldsymbol{q_1}}$ and $F_{\boldsymbol{q}}$, it is an X'^{*}-boundary point of $F_{\boldsymbol{q_1}}$. Let P be the set of $2^{|X \setminus Vars(\boldsymbol{q_1})|}$ points obtained from $\boldsymbol{p}$ by changing assignments to variables of $X \setminus Vars(\boldsymbol{q_1})$. Since the variables of X^* are redundant in $F_{\boldsymbol{q_1}}$, then P has to contain a point satisfying $F_{\boldsymbol{q_1}}$. This means that point $\boldsymbol{p}$ of $F_{\boldsymbol{q}}$ can be turned into an assignment satisfying F if the variables that are in $Vars(\boldsymbol{q}) \setminus Vars(\boldsymbol{q_1})$ are allowed to change their values. So the irredundancy of X^* in $F_{\boldsymbol{q}}$ can be only local.

Remark 8: In Definition 10 of D-sequent $(F,X',\boldsymbol{q}) \rightarrow X''$, we did not mention local irredundancy. However, in the rest of the Appendix we assume that the variables of X' in $F_{\boldsymbol{q}}$ and those of X'' in $Dis(F_{\boldsymbol{q}},X')$ may have local irredundancy. For the sake of simplicity, we do not mention this fact with the exception of Lemmas 7 and 8. In particular, in Lemma 8, we show that D-sequents derived by *DDS_impl* can only have local irredundancy and so the latter can be safely ignored.

Remark 9: Checking if a set of variables X', where $X' \subseteq (X \setminus Vars(\boldsymbol{q}))$ is irredundant in $F_{\boldsymbol{q}}$ *only locally* is hard. For that reason *DDS_impl* does not perform such a check. However, one has to introduce the notion of local irredundancy because the latter may appear when resolving D-sequents (see Lemma 5). Fortunately, given a D-sequent $(F,X',\boldsymbol{q}) \rightarrow X''$, one does not need to check if irredundancy of variables X' in $F_{\boldsymbol{q}}$ or X'' in $Dis(F_{\boldsymbol{q}},X')$ (if any) is local. According to Lemma 8, this irredundancy is always local. Eventually a D-sequent $(F,\emptyset,\emptyset) \rightarrow X$ is derived that does not have any local irredundancy (because the partial assignment $\boldsymbol{q}$ of this D-sequent is empty).

Lemma 6: Let $F(X,Y)$ be a CNF formula and $\boldsymbol{q}$ be a partial assignment to variables of X. Let X^* where $X^* \subseteq X$ be a set of variables redundant in $F_{\boldsymbol{q}}$. Let sets X' and X'' form a partition of X^* i.e. $X^* = X' \cup X''$ and $X' \cap X'' = \emptyset$. Then D-sequent $(F,X',\boldsymbol{q}) \rightarrow X''$ holds.

Proof: Assume the contrary i.e. that the D-sequent $(F,X',\boldsymbol{q}) \rightarrow X''$ does not hold. According to Definition 10, this means that either

A) variables of X' are not redundant in $F_{\boldsymbol{q}}$ or
B) variables of X'' are not redundant in $Dis(F_{\boldsymbol{q}},X')$.

CASE A: This means that there exists an X'^{+}-boundary point $\boldsymbol{p}$ (where $X'^{+} \subseteq X'$ and $\boldsymbol{q} \leq \boldsymbol{p}$) that is removable in $F_{\boldsymbol{q}}$. This implies that the variables of X'^{+} are not a set of redundant variables. On the other hand, since $X'^{+} \subseteq X'$ and the variables of X' are redundant, the variables of X'^{+} are redundant too. Contradiction.

CASE B: This means that there exists an X''^{+}-boundary point $\boldsymbol{p}$ (where $X''^{+} \subseteq X''$ and $\boldsymbol{q} \leq \boldsymbol{p}$) that is removable in $Dis(F_{\boldsymbol{q}},X')$. Note that point $\boldsymbol{p}$ is an X^{*+}-boundary point of $F_{\boldsymbol{q}}$ where $X^{*+} \subseteq X^*$ (because $F_{\boldsymbol{q}}$ consists of the clauses of $Dis(F_{\boldsymbol{q}},X')$ plus some X'-clauses). Since the variables of X^* are redundant in $F_{\boldsymbol{q}}$ the point $\boldsymbol{p}$ cannot be removable. Then there is a point $\boldsymbol{p^*}$ obtained by flipping the variables of $X \setminus Vars(\boldsymbol{q})$ that satisfies $F_{\boldsymbol{q}}$. Point $\boldsymbol{p^*}$ also satisfies $Dis(F_{\boldsymbol{q}},X')$. Hence, the point $\boldsymbol{p}$ cannot be removable in $Dis(F_{\boldsymbol{q}},X')$. Contradiction.

Lemma 7: Let $F(X,Y)$ be a CNF formula and $\boldsymbol{q}$ be a partial assignment to variables of X. Let D-sequent $(F,X',\boldsymbol{q}) \rightarrow X''$ hold modulo local irredundancy. That is the variables of X' and X'' are redundant in $F_{\boldsymbol{q}}$ and $Dis(F_{\boldsymbol{q}},X')$ respectively modulo local irredundnacy. Then the variables of $X' \cup X''$ are redundant in $F_{\boldsymbol{q}}$ modulo local iredundancy.

Proof: Denote by X^* the set $X' \cup X''$. Let $\boldsymbol{p}$ be a removable X^{+}-boundary point of $F_{\boldsymbol{q}}$ where $X^{+} \subseteq X^*$. Let us consider the two possible cases:

- $X^+ \subseteq X'$ (and so $X^+ \cap X'' = \emptyset$). Since $\boldsymbol{p}$ is removable, the variables of X' are irredundant in $F_{\boldsymbol{q}}$. Since this irredundancy can only be local one can turn $\boldsymbol{p}$ into an assignment satisfying F. This means that the irredundancy of variables X^* in F due to point $\boldsymbol{p}$ is local.
- $X^+ \not\subseteq X'$ (and so $X^+ \cap X'' \neq \emptyset$). Then $\boldsymbol{p}$ is an X''^+-boundary point of $Dis(F_{\boldsymbol{q}}, X')$ where $X''^+ = X^+ \cap X''$. Indeed, for every variable x of X^+ there has to be a clause C of $F_{\boldsymbol{q}}$ falsified by $\boldsymbol{p}$ such that $Vars(C) \cap X^+ = \{x\}$. Otherwise, x can be removed from X^+, which contradicts the assumption that $\boldsymbol{p}$ is an X^+-boundary point. This means that for every variable x of X''^+ there is a clause C falsified by $\boldsymbol{p}$ such that $Vars(C) \cap X''^+ = \{x\}$.

  Let P denote the set of all $2^{|X \setminus (Vars(\boldsymbol{q}) \cup X')|}$ points obtained from $\boldsymbol{p}$ by flipping values of variables of $X \setminus (Vars(\boldsymbol{q}) \cup X')$. Let us consider the following two possibilities.
  - Every point of P falsifies $Dis(F_{\boldsymbol{q}}, X')$. This means that the point $\boldsymbol{p}$ is a removable X''^+- boundary point of $Dis(F_{\boldsymbol{q}}, X')$. Hence the variables of X'' are irredundant in $Dis(F_{\boldsymbol{q}}, X')$. Since this irredundancy is local, point $\boldsymbol{p}$ can be turned into an assignment satisfying F by changing values of variables of X. Hence the irredundancy of X^* in F due to point $\boldsymbol{p}$ is local.
  - A point $\boldsymbol{d}$ of P satisfies $Dis(F_{\boldsymbol{q}}, X')$. Let us consider the following two cases.
    - $\boldsymbol{d}$ satisfies $F_{\boldsymbol{q}}$. This contradicts the fact that $\boldsymbol{p}$ is a removable X^+-boundary point of $F_{\boldsymbol{q}}$. (By flipping variables of $X \setminus Vars(\boldsymbol{q})$ one can obtain a point satisfying $F_{\boldsymbol{q}}$.)
    - $\boldsymbol{d}$ falsifies some clauses of $F_{\boldsymbol{q}}$. Since $F_{\boldsymbol{q}}$ and $Dis(F_{\boldsymbol{q}}, X')$ are different only in X'-clauses, $\boldsymbol{d}$ is an X'^*-boundary point of $F_{\boldsymbol{q}}$ where $X'^* \subseteq X'$. Since $\boldsymbol{p}$ is a removable X^+-boundary point of $F_{\boldsymbol{q}}$, $\boldsymbol{d}$ is a removable X'^*-boundary point of $F_{\boldsymbol{q}}$. So the variables of X' are irredundant in $F_{\boldsymbol{q}}$. Since this irredundancy is local, the point $\boldsymbol{d}$ can be turned into an assignment satisfying F by changing the values of X. Then, the same is true for point $\boldsymbol{p}$. So the irredundancy of X^* in F due to point $\boldsymbol{p}$ is local.

Proposition 7: Let $F(X, Y)$ be a CNF formula. Let D-sequents S_1 and S_2 be equal to $(F, X_1, \boldsymbol{q_1}) \rightarrow X'$ and $(F, X_2, \boldsymbol{q_2}) \rightarrow X'$ respectively. Let $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ be resolvable on variable x. Denote by $\boldsymbol{q}$ the partial assignment $Res(\boldsymbol{q_1},\boldsymbol{q_2},x)$ and by X^* the set $X_1 \cap X_2$. Then, if S_1 and S_2 hold, the D-sequent S equal to $(F, X^*, \boldsymbol{q}) \rightarrow X'$ holds too.

Proof: Lemma 7 implies that the variables of $X_1 \cup X'$ and $X_2 \cup X'$ are redundant in $F_{\boldsymbol{q_1}}$ and $F_{\boldsymbol{q_2}}$ respectively. From Lemma 5, one concludes that the variables of the set $X'' = (X_1 \cup X') \cap (X_2 \cup X')$ are redundant in $F_{\boldsymbol{q}}$. From Definition 10 it follows that $X_1 \cap X' = X_2 \cap X' = \emptyset$. So $X'' = X^* \cup X'$ where $X^* \cap X' = \emptyset$. Then, from Lemma 6, it follows that the D-sequent $(F, X^*, \boldsymbol{q}) \rightarrow X'$ holds.

SUBSECTION: Derivation of a D-sequent

Proposition 8: Let $F(X, Y)$ be a CNF formula and $\boldsymbol{q}$ be a partial assignment to variables of X. Let X_{red} be the variables proved redundant in $F_{\boldsymbol{q}}$. Let x be the only variable of X that is not in $Vars(\boldsymbol{q}) \cup X_{red}$. Let D-sequent $(F, X_{red}, \boldsymbol{q}) \rightarrow \{x\}$ hold. Then D-sequent $(F, X'_{red}, \boldsymbol{g}) \rightarrow \{x\}$ holds where $\boldsymbol{g}$ and X'_{red} are defined as follows. Partial assignment $\boldsymbol{g}$ to variables of X satisfies the two conditions below (implying that $\boldsymbol{g} \leq \boldsymbol{q}$):

1) Let C be a $\{x\}$-clause of F that is not in $Dis(F_{\boldsymbol{q}}, X_{red})$. Then either
   - $\boldsymbol{g}$ contains an assignment satisfying C or
   - D-sequent $(F, X^*_{red}, \boldsymbol{g^*}) \rightarrow \{x^*\}$ holds where $\boldsymbol{g^*} \leq \boldsymbol{g}$, $X^*_{red} \subset X_{red}$, $x^* \in (X_{red} \cap Vars(C))$.
2) Let $\boldsymbol{p_1}$ be a point such that $\boldsymbol{q} \leq \boldsymbol{p_1}$. Let $\boldsymbol{p_1}$ falsify a clause of F with literal x. Let $\boldsymbol{p_2}$ be obtained from $\boldsymbol{p_1}$ by flipping the value of x and falsify a clause of F with literal $\overline{x}$. Then there is a non-$\{x\}$-clause C of F falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ such that $(Vars(C) \cap X) \subseteq Vars(\boldsymbol{g})$.

The set X'_{red} consists of all the variables already proved redundant in $F_{\boldsymbol{g}}$. That is every redundant variable x^* of X_{red} with D-sequent $(F, X^*_{red}, \boldsymbol{g^*}) \rightarrow \{x^*\}$ such that $\boldsymbol{g^*} \leq \boldsymbol{g}$, $X^*_{red} \subset X_{red}$ is in X'_{red}.

Proof: Assume the contrary i.e. D-sequent $(F, X'_{red}, \boldsymbol{g}) \rightarrow \{x\}$ does not hold, and so variable x is not redundant in $Dis(F_{\boldsymbol{g}}, X'_{red})$. Hence there is a point $\boldsymbol{p}$, $\boldsymbol{g} \leq \boldsymbol{p}$ that is a removable $\{x\}$-boundary point of $Dis(F_{\boldsymbol{g}}, X'_{red})$.

Let C be an $\{x\}$-clause of F. Note that $Dis(F_{\boldsymbol{g}}, X'_{red})$ cannot contain the clause $C_{\boldsymbol{g}}$ if the clause $C_{\boldsymbol{q}}$ is not in $Dis(F_{\boldsymbol{q}}, X_{red})$. If $C_{\boldsymbol{q}}$ is not in $Dis(F_{\boldsymbol{q}}, X_{red})$, then $\boldsymbol{g}$ either satisfies C or C contains a variable of X_{red} that is also in X'_{red} (and hence $C_{\boldsymbol{g}}$ contains a redundant variable and so is not in $Dis(F_{\boldsymbol{g}}, X'_{red})$).

So, for $\boldsymbol{p}$ to be an $\{x\}$-boundary point of $F_{\boldsymbol{g}}$, there has to be $\{x\}$-clauses A and B of F such that

- they are not satisfied by $\boldsymbol{g}$ and do not contain variables of X'_{red} (so the clauses $A_{\boldsymbol{g}}$ and $B_{\boldsymbol{g}}$ are in $Dis(F_{\boldsymbol{g}}, X'_{red})$)
- A is falsified by $\boldsymbol{p}$ and B is falsified by the point obtained from $\boldsymbol{p}$ by flipping the value of x.

Let point $\boldsymbol{p_1}$ be obtained from $\boldsymbol{p}$ by flipping assignments to the variables of $Vars(\boldsymbol{q}) \setminus Vars(\boldsymbol{g})$ that disagree with $\boldsymbol{q}$. By construction $\boldsymbol{g} \leq \boldsymbol{p_1}$ and $\boldsymbol{q} \leq \boldsymbol{p_1}$. Let $\boldsymbol{p_2}$ be the point obtained from $\boldsymbol{p_1}$ by flipping the value of x. Since x is not assigned in $\boldsymbol{q}$ (and hence is not assigned in $\boldsymbol{g}$), $\boldsymbol{g} \leq \boldsymbol{p_2}$ and $\boldsymbol{q} \leq \boldsymbol{p_2}$. Then $A_{\boldsymbol{q}}$ and $B_{\boldsymbol{q}}$ are also in $F_{\boldsymbol{q}}$. As we mentioned above A and B cannot contain variables of $X_{\boldsymbol{red}}$ (otherwise they could not be in $F_{\boldsymbol{g}}$). So A and B are also in $Dis(F_{\boldsymbol{q}}, X_{red})$.

Note that clause A is falsified by $\boldsymbol{p_1}$. Assume the contrary, i.e. that A is satisfied by $\boldsymbol{p_1}$. Then the fact that $\boldsymbol{p}$ and $\boldsymbol{p_1}$ are different only in assignments to $\boldsymbol{q}$ and that $\boldsymbol{p}$ falsifies A implies that q satisfies A. But then by construction, $\boldsymbol{g}$ has to satisfy A and we have contradiction. Since B is also an $\{x\}$-clause as A, one can use the same reasoning to show that $\boldsymbol{p_2}$ falsifies B.

Since $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ falsify $\{x\}$-clauses A and B and $\boldsymbol{p_1}, \boldsymbol{p_2} \leq \boldsymbol{q}$ one can apply Condition 2 of the proposition at hand. That is there must be a clause C falsified by $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ such that $\boldsymbol{g}$ contains all the assignments of $\boldsymbol{q}$ that falsify literals of C. This means that C is not satisfied by $\boldsymbol{g}$. Besides, since due to Condition 2 every variable of $Vars(C) \cap X$ is in $Vars(\boldsymbol{g})$, every variable of $C_{\boldsymbol{g}}$ is in Y. Hence a variable of $C_{\boldsymbol{g}}$ cannot be redundant. This means that $C_{\boldsymbol{g}}$ is in $Dis(F_{\boldsymbol{g}}, X'_{red})$. Since $\boldsymbol{p}$ and $\boldsymbol{p_1}$ have identical assignments to the variables of Y, then $\boldsymbol{p}$ falsifies $C_{\boldsymbol{g}}$ too. So $\boldsymbol{p}$ cannot be an $\{x\}$-boundary point of $Dis(F_{\boldsymbol{g}}, X'_{red})$. Contradiction.

Proofs of Section VI

Lemma 8: Let $(F, X', \boldsymbol{g}) \rightarrow X''$ be a D-sequent derived by *DDS_impl* and $\boldsymbol{q}$ be the partial assignment when this D-sequent is derived. Let variables of X' be irredundant in $F_{\boldsymbol{g}}$ or variables of X'' be irredundant in $Dis(F_{\boldsymbol{g}}, X')$. Then this irredundancy is only local. (See Definition 18 and Remarks 8 and 9.)

Proof: We carry out the proof by induction in the number of D-sequents. The base step is that the statement holds for an empty set of D-sequents, which is vacuously true. The inductive step is to show that the fact that the statement holds for D-sequents $S_1, \ldots, S_n$ implies that it is true for S_{n+1}. Let us consider all possible cases.

- S_{n+1} is a D-sequent $(F, X', \boldsymbol{g}) \rightarrow \{x\}$ for a monotone variable x of $Dis(F_{\boldsymbol{g}}, X')$ where $x \in (X \setminus (Vars(\boldsymbol{q}) \cup X')$. Since formula $Dis(F_{\boldsymbol{g}}, X')$ cannot have removable $\{x\}$-boundary points (see Proposition 2), variable x cannot be irredundant in $Dis(F_{\boldsymbol{g}}, X')$. The variables of X' may be irredundant in $F_{\boldsymbol{g}}$. However, this irredundancy can be only local. Indeed, using Lemma 7 and the induction hypothesis one can show that variables proved redundant for $F_{\boldsymbol{g}}$ according to the relevant D-sequents of the set $\{S_1, \ldots, S_n\}$ are indeed redundant in $F_{\boldsymbol{g}}$ modulo local irredundancy.
- S_{n+1} is a D-sequent $(F, \emptyset, \boldsymbol{g}) \rightarrow X'$ derived due to appearance of an empty clause C in $F_{\boldsymbol{g}}$. Here $\boldsymbol{g}$ is the minimum subset of assignments of $\boldsymbol{q}$ falsifying C. In this case, $F_{\boldsymbol{g}}$ has no boundary points and hence the set X' of unassigned variables of $F_{\boldsymbol{g}}$ cannot be irredundant.
- S_{n+1} is a D-sequent $(F, X', \boldsymbol{g}) \rightarrow \{x\}$ derived after making the only unassigned variable x of $Dis(F_{\boldsymbol{q}}, X_{red})$ redundant by adding resolvents on variable x. (As usual, X_{red} denotes the set of redundant variables already proved redundant in $F_{\boldsymbol{q}}$.) In this case, every removable $\{x\}$-boundary point of $Dis(F_{\boldsymbol{q}}, X_{red})$ is eliminated and so the latter cannot be irredundant in x. Due to Proposition 8, the same applies to $Dis(F_{\boldsymbol{g}}, X')$. To show that irredundancy of variables of X' in $F_{\boldsymbol{g}}$ can be only local one can use the same reasoning as in the case when x is a monotone variable.
- S_{n+1} is obtained by resolving D-sequents S_i and S_j where $1 \leq i, j \leq n$ and $i \neq j$. Let S_i,S_j and S_{n+1} be equal to $(F, X_i, \boldsymbol{q_i}) \rightarrow X''$, $(F, X_j, \boldsymbol{q_j}) \rightarrow X''$ and $(F, X', \boldsymbol{g}) \rightarrow X''$ respectively where $X' = X_i \cap X_j$ and $\boldsymbol{g}$ is obtained by resolving $\boldsymbol{q_i}$ and $\boldsymbol{q_j}$ (see Definition 11).

  Let us first show that irredundancy of X'' in $Dis(F_{\boldsymbol{g}}, X')$ can only be local. Let $\boldsymbol{p}$ be a removable X''^*-boundary point of $Dis(F_{\boldsymbol{g}}, X')$ where $X''^* \subseteq X''$.
  Then either $\boldsymbol{q_i} \leq \boldsymbol{p}$ or $\boldsymbol{q_j} \leq \boldsymbol{p}$. Assume for the sake of clarity that $\boldsymbol{q_i} \leq \boldsymbol{p}$. Consider the following two cases.

  - $\boldsymbol{p}$ is not removable in $Dis(F_{\boldsymbol{q_i}}, X_i)$. Then the irredundancy of X'' in $Dis(F_{\boldsymbol{g}}, X')$ due to point $\boldsymbol{p}$ is local. (A point satisfying $Dis(F_{\boldsymbol{q_i}}, X_i)$ can be obtained from $\boldsymbol{p}$ by changing values of some variables from $X \setminus (X_i \cup Vars(\boldsymbol{q_i}))$. The same point satisfies $Dis(F_{\boldsymbol{g}}, X')$ because $\boldsymbol{g} \leq \boldsymbol{q_i}$ and $X' \subseteq X_i$.)
  - $\boldsymbol{p}$ is also removable in $Dis(F_{\boldsymbol{q_i}}, X_i)$. This means that the variables of X' are irredundant in $Dis(F_{\boldsymbol{q_i}}, X_i)$. By the induction hypothesis, this irredundancy is local. Then one can turn $\boldsymbol{p}$ into a satisfying assignment of F by changing assignments to variables of X. Hence the irredundancy of X'' in $Dis(F_{\boldsymbol{g}}, X')$ due to point $\boldsymbol{p}$ is also local.

  Now, let us show that irredundancy of X' in $F_{\boldsymbol{g}}$ can only be local. Let $\boldsymbol{p}$ be a removable X'^*-boundary point of $F_{\boldsymbol{g}}$ where $X'^* \subseteq X'$. Again, assume for the sake of clarity that $\boldsymbol{q_i} \leq \boldsymbol{p}$. Consider the following two cases.

  - $\boldsymbol{p}$ is not removable in $F_{\boldsymbol{q_i}}$. Then the irredundancy of X' in $F_{\boldsymbol{q}}$ due to point $\boldsymbol{p}$ is local. (A point satisfying $F_{\boldsymbol{q_i}}$ can be obtained by from $\boldsymbol{p}$ by changing values of some variables from $X \setminus Vars(\boldsymbol{q_i})$. The same point satisfies $F_{\boldsymbol{g}}$ because $\boldsymbol{g} \leq \boldsymbol{q_i}$.)
  - $\boldsymbol{p}$ is also removable in $F_{\boldsymbol{q_i}}$. This means that the variables of X' (and hence the variables of X_i) are irredundant in $F_{\boldsymbol{q_i}}$. By the induction hypothesis, this irredundancy is local. Then one can turn $\boldsymbol{p}$ into a satisfying assignment of F by changing assignments to variables of X. Hence the irredundancy of X' in $F_{\boldsymbol{q}}$ due to point $\boldsymbol{p}$ is also local.

Remark 10: Note that correctness of the final D-sequent $(F, \emptyset, \emptyset) \rightarrow X$ modulo local irredundancy implies that the variables of X are redundant in F. In this case, there is no difference between just redundancy and redundancy modulo local irredundancy because $\boldsymbol{q}$ is empty. (So the value of any variable of X can be changed when checking if a boundary point is removable.)

Lemma 9: Let $F(X, Y)$ be a CNF formula and $X = \{x_1, \ldots, x_k\}$. Let $S_1, \ldots, S_k$ be D-sequents where S_i is the D-sequent $\emptyset \rightarrow \{x_i\}$. Assume that S_1 holds for the formula F, S_2 holds for the formula $Dis(F, \{x_1\})$, ...,S_k holds for the formula $Dis(F, \{x_1, \ldots, x_{k-1}\})$. (To simplify the notation we assume that D-sequents S_i have been derived in the order they are numbered). Then the variables of X are redundant in $F(X, Y)$.

Proof: Since S_1 holds, due to Proposition 3, the formula $\exists X.F$ is equivalent to $\exists(X \setminus \{x_1\}).Dis(F,\{x_1\})$. Since S_2 holds for $Dis(F,\{x_1\})$ one can apply Proposition 3 again to show that $\exists(X \setminus \{x_1\}).Dis(F,\{x_1\})$ is equivalent to $\exists(X \setminus \{x_1,x_2\}).Dis(F,\{x_1,x_2\})$ and hence the latter is equivalent to $\exists X.F$. By applying Proposition 3 $k{-}2$ more times one shows that $\exists X.F$ is equivalent to $Dis(F,X)$. According to Corollary 1, this means that the variables of X are redundant in $F(X,Y)$.

Proposition 9: *DDS_impl* is sound and complete.

Proof: First, we show that *DDS_impl* is complete. *DDS_impl* builds a binary search tree and visits every node of this tree at most three times (when starting the left branch, when backtracking to start the right branch, when backtracking from the right branch). So *DDS_impl* is complete.

Now we prove that *DDS_impl* is sound. The idea of the proof is to show that all D-sequents derived by *DDS_impl* are correct. By definition, *DDS_impl* eventually derives correct D-sequents $\emptyset \rightarrow \{x\}$ for every variable of X. From Lemma 9 it follows that this is equivalent to derivation of the correct D-sequent $\emptyset \rightarrow X$.

We prove the correctness of D-sequents derived by *DDS_impl* by induction. The base statement is that the D-sequents of an empty set are correct (which is vacuously true). The induction step is that to show that if first n D-sequents are correct, then next D-sequent S is correct too. Let us consider the following alternatives.

- S is a D-sequent built for a monotone variable of $Dis(F_{\boldsymbol{q}},X_{red})$. The correctness of S follows from Proposition 8 and the induction hypothesis (that the D-sequents derived before are correct).
- S is the D-sequent specified by a locally empty clause. In this case, S is trivially true.
- S is a D-sequent derived by *DDS_impl* in the BPE state for variable x after eliminating $\{x\}$-removable $\{x\}$-boundary points of $Dis(F_{\boldsymbol{q}},X_{red})$. The correctness of S follows form Proposition 8 and the induction hypothesis.
- S is obtained by resolving two existing D-sequents. The correctness of S follows from Proposition 7 and the induction hypothesis.

PROOFS OF SECTION VII

Definition 19: Let *Proof* be a resolution proof that a CNF formula H is unsatisfiable. Let G_{proof} be the resolution graph specified by *Proof*. (The sources of G_{proof} correspond to clauses of H. Every non-source node of G_{proof} corresponds to a resolvent of *Proof*. The sink of G_{proof} is an empty clause. Every non-source node of G_{proof} has two incoming edges connecting this note to the nodes corresponding to the parent clauses.) We will call *Proof* **irredundant**, if for every node of G_{proof} there is a path leading from this node to the sink.

Lemma 10: Let $F(X,Y)$ be equal to $F_1(X_1,Y_1) \wedge \ldots \wedge F_k(X_k,Y_k)$ where $(X_i \cup Y_i) \cap (X_j \cup Y_j) = \emptyset$, $i \neq j$. Let F be satisfiable. Let F have no $\{x\}$-removable $\{x\}$-boundary points where $x \in X_i$ and *Proof* be a resolution proof of that fact built by *DDS_impl*. Then *Proof* does not contain clauses of F_j,$j \neq i$ (that is no clause of F_j is used as a parent clause in a resolution of *Proof*).

Proof: *DDS_impl* concludes that all $\{x\}$-removable $\{x\}$-boundary points have been eliminated if the CNF formula H described in Subsection VI-C is unsatisfiable. H consists of clauses of the current formula $Dis(F_{\boldsymbol{q}},X_{red})$ and the clauses of CNF formula H_{dir}. *DDS_impl* builds an irredundant resolution proof that H is unsatisfiable. (Making *Proof* irredundant is performed by function *optimize* of Figure 4.)

Since formula F is the conjunction of independent subformulas, clauses of F_i and F_j, $j \neq i$ cannot be resolved with each other. The same applies to *resolvents* of clauses of F_i and F_j and to resolvents of clauses of $F_i \wedge H_{dir}$ and F_j. (By construction [12], H_{dir} may have only variables of $\{x\}$-clauses of F and some new variables i.e. ones that are not present in F. Since $x \in X_i$, this means that the variables of H_{dir} can only overlap with those of F_i.) Therefore, an irredundant proof of unsatisfiability of H has to contain only clauses of either formula F_j, $j \neq i$ or formula $F_i \wedge H_{dir}$. Formula F is satisfiable, hence every subformula F_j, $j = 1,\ldots,k$ is satisfiable too. So, a proof cannot consists solely of clauses of F_j,$j \neq i$. This means that *Proof* employs only clauses of $F_i \wedge H_{dir}$ (and their resolvents).

Proposition 10: *DDS_impl* is compositional regardless of how branching variables are chosen.

Proof: The main idea of the proof is to show that every D-sequent generated by *DDS_impl* has the form $\boldsymbol{g} \rightarrow X'$ where $Vars(\boldsymbol{g}) \subseteq X_i$ and $X' \subseteq X$. We will call such a D-sequent **limited to $\boldsymbol{F_i}$**. Let us carry on the proof by induction. Assume that the D-sequents generated so far are limited to F_i and show that this holds for the next D-sequent S. Since one cannot resolve clauses of F_i and F_j, $i \neq j$, if S is specified by a clause that is locally empty, S is limited to F_i.

Let S be a D-sequent generated for a monotone variable $x \in X_i$. According to Remark 4, only Condition 1 contributes to forming $\boldsymbol{g}$. In this case, $Vars(\boldsymbol{g})$ consists of

1) variables of $\{x\}$-clauses of F and
2) variables of $Vars(\boldsymbol{g^*})$ of D-sequents $\boldsymbol{g^*} \rightarrow \{x^*\}$ showing redundancy of variables x^* of $\{x\}$-clauses of F.

Every $\{x\}$-clause of F is either a clause of the original formula F_i or its resolvent. So the variables that are in $\boldsymbol{g}$ due to the first condition above are in X_i. By the induction hypothesis, the variables of $Vars(\boldsymbol{g^*})$ are also in X_i.

Let S be obtained after eliminating $\{x\}$-removable $\{x\}$-boundary points where $x \in X_i$ (see Subsection VI-C). Denote by $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$ the two parts of $\boldsymbol{g}$ specified by Condition 1 and 2 of Proposition 8. (Assignment $\boldsymbol{g}$ is the union of assignments $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$.) The variables of $Vars(\boldsymbol{g_1})$ are in X_1 for the same reasons as in the case of monotone variables.

To generate $\boldsymbol{g_2}$, *DDS_impl* uses proof *Proof* that formula H built from clauses of F and H_{dir} (see Subsection VI-C) is unsatisfiable. As we showed in Lemma 10, *Proof* employs only clauses of $F_i \wedge H_{dir}$ and their resolvents. Only clauses of formula F are taken into account when forming $\boldsymbol{g_2}$ in

Proposition 8 (i.e. clauses of H_{dir} do not affect $\boldsymbol{g_2}$). Since the only clauses of F used in *Proof* are those of F_i, then $Vars(\boldsymbol{g_2}) \subseteq X_i$.

Finally, if S is obtained by resolving two D-sequents limited to F_i, it is also limited to F_i (see Definition 12).