On Efficient Algorithms For Partial Quantifier Elimination

Eugene Goldberg

eu.goldberg@gmail.com

Abstract. Earlier, we introduced Partial Quantifier Elimination (PQE). It is a *generalization* of regular quantifier elimination where one can take a *part* of the formula out of the scope of quantifiers. We apply PQE to CNF formulas of propositional logic with existential quantifiers. The appeal of PQE is that many problems like equivalence checking and model checking can be solved in terms of PQE and the latter can be very efficient. The main flaw of current PQE solvers is that they do not *reuse* learned information. The problem here is that these PQE solvers are based on the notion of clause redundancy and the latter is a *structural* rather than *semantic* property. In this paper, we provide two important theoretical results that enable reusing the information learned by a PQE solver. Such reusing can dramatically boost the efficiency of PQE like conflict clause learning boosts SAT solving.

1 Introduction

Partial Quantifier Elimination (PQE) is a generalization of regular quantifier elimination (QE). In PQE one can take a *part* of the formula out of the scope of quantifiers hence the name *partial*. The appeal of PQE is twofold. First, many old problems like equivalence checking, model checking and SAT and new problems like property generation [1] can be solved in terms of PQE. Second, PQE can be dramatically more efficient than QE.

PQE is defined as follows [2]. Let $F(X,Y)$ be a *propositional* formula in conjunctive normal form[1] (CNF) where X,Y are sets of variables. Let G be a subset of clauses of F. Given a formula $\exists X[F]$, the PQE problem is to find a quantifier-free formula $H(Y)$ such that $\exists X[F] \equiv H \wedge \exists X[F \setminus G]$. In contrast to *full* QE, only the clauses of G are taken out of the scope of quantifiers. In this paper, we consider PQE for formulas with only *existential* quantifiers. We will refer to H as a *solution* to PQE. Note that QE is just a special case of PQE where $G = F$ and the entire formula is unquantified. A key role in PQE solving is played by *redundancy based reasoning*: to take a set of clauses G out of $\exists X[F(X,Y)]$, one essentially needs to find a formula $H(Y)$ that makes G *redundant* in $H \wedge \exists X[F]$. Redundancy based reasoning helps to explain the

[1] Given a CNF formula F represented as the conjunction of clauses $C_0 \wedge \dots \wedge C_k$, we will also consider F as the *set* of clauses $\{C_0,\dots,C_k\}$.

efficiency of PQE over QE. If G is small, finding H that makes G redundant in $H \wedge \exists X[F]$ is much easier than making redundant the *entire* formula F.

The current PQE algorithms implement redundancy based reasoning via the machinery of D-sequents [3,4]. A D-sequent is a record stating the redundancy of a clause C in the current formula $\exists X[F]$ in subspace $\vec{q}$. Here $\vec{q}$ is an assignment to variables of F and "D" stands for "Dependency". The redundancy of C above also holds in any formula $\exists X[F^*]$ where F^* is obtained from F by adding clauses implied by F.

A PQE solver branches on variables of F until it reaches the subspace where every clause of G can be trivially proved or made redundant. At this point, so called atomic D-sequents are generated. D-sequents derived in different branches can be resolved similarly to clauses. The objective of a PQE solver is to derive, for each clause $C \in G$, the "global" D-sequent stating the redundancy of C in the final formula $\exists X[F]$ in the *entire* space. This D-sequent is obtained by resolving D-sequents generated in subspaces. The derivation of global D-sequents, in general, requires adding to $\exists X[F]$ new clauses. The unquantified clauses (i.e., those depending only Y) that have been added to the initial formula $\exists X[F]$ form a solution $H(Y)$ to the PQE problem above.

A major flaw of current PQE solvers is that they *do not reuse* D-sequents (in contrast to SAT-solvers that derive their power from reusing conflict clauses). The reason for that is as follows. A conflict clause expresses a local *inconsistency*, the latter being a *semantic* property. This simplifies reusing conflict clauses in different contexts. On the other hand, as we argue in Section 5, a D-sequent expresses a local *unobservability* that is a *structural* rather than semantic property. This means that reusing D-sequents depends on the context and so is not trivial. Importantly, current PQE algorithms repeatedly derive the same D-sequents (see Appendix C). So, the lack of reusing D-sequents cripples the performance of PQE solving. The main contribution of this paper is addressing the issue of D-sequent reusing. Namely, we present important theoretical results enabling safe reusing of D-sequents. These results apply to so-called *extended* D-sequents introduced in [5].

This paper is structured as follows. Section 2 gives some basic definitions. In Section 3 we explain PQE solving by the example of the PQE algorithm called *DS-PQE*. The notion of extended D-sequents is introduced in Section 4. Section 5 gives a high-level view of D-sequent reusing. In Sections 6-7 we lay a theoretical foundation for D-sequent reusing. Finally, in Sections 8 and 9, we present some background and make conclusions.

2 Basic Definitions

In this section, when we say "formula" without mentioning quantifiers, we mean "a quantifier-free formula".

Definition 1. *We assume that formulas have only Boolean variables. A* ***literal*** *of a variable v is either v or its negation. A* ***clause*** *is a disjunction of literals.*

*A formula F is in conjunctive normal form (**CNF**) if $F = C_0 \wedge \dots \wedge C_k$ where $C_0, \dots, C_k$ are clauses. We will also view F as the **set of clauses** $\{C_0, \dots, C_k\}$. We assume that **every formula is in CNF** unless otherwise stated.*

Definition 2. *Let F be a formula. Then $\boldsymbol{Vars(F)}$ denotes the set of variables of F and $\boldsymbol{Vars(\exists X[F])}$ denotes $Vars(F) \setminus X$.*

Definition 3. *Let V be a set of variables. An **assignment** $\vec{q}$ to V is a mapping $V' \rightarrow \{0,1\}$ where $V' \subseteq V$. We will denote the set of variables assigned in $\vec{q}$ as $\boldsymbol{Vars(\vec{q})}$. We will refer to $\vec{q}$ as a **full assignment** to V if $Vars(\vec{q}) = V$. We will denote as $\boldsymbol{\vec{q} \subseteq \vec{r}}$ the fact that a) $Vars(\vec{q}) \subseteq Vars(\vec{r})$ and b) every variable of $Vars(\vec{q})$ has the same value in $\vec{q}$ and $\vec{r}$.*

Definition 4. *A literal and a clause are said to be **satisfied** (respectively **falsified**) by an assignment $\vec{q}$ if they evaluate to 1 (respectively 0) under $\vec{q}$.*

Definition 5. *Let C be a clause. Let H be a formula that may have quantifiers, and $\vec{q}$ be an assignment to $Vars(H)$. If C is satisfied by $\vec{q}$, then $\boldsymbol{C_{\vec{q}} \equiv 1}$. Otherwise, $\boldsymbol{C_{\vec{q}}}$ is the clause obtained from C by removing all literals falsified by $\vec{q}$. Denote by $\boldsymbol{H_{\vec{q}}}$ the formula obtained from H by removing the clauses satisfied by $\vec{q}$ and replacing every clause C unsatisfied by $\vec{q}$ with $C_{\vec{q}}$.*

Definition 6. *Let G, H be formulas that may have existential quantifiers. We say that G, H are **equivalent**, written $\boldsymbol{G \equiv H}$, if $G_{\vec{q}} = H_{\vec{q}}$ for all full assignments $\vec{q}$ to $Vars(G) \cup Vars(H)$.*

Definition 7. *Let $F(X,Y)$ be a formula and $G \subseteq F$ and $G \neq \emptyset$. The clauses of G are said to be **redundant in** $\boldsymbol{\exists X[F]}$ if $\exists X[F] \equiv \exists X[F \setminus G]$. If $F \setminus G$ implies G, the clauses of G are redundant in $\exists X[F]$ but the opposite is not true.*

Definition 8. *Given a formula $\exists X[F(X,Y)]$, a clause C of F is called a **quantified clause** if $Vars(C) \cap X \neq \emptyset$. Otherwise, C is called unquantified.*

Definition 9. *Given a formula $\exists X[F(X,Y))]$ and G where $G \subseteq F$, the **Partial Quantifier Elimination (PQE)** problem is to find $H(Y)$ such that $\boldsymbol{\exists X[F] \equiv H \wedge \exists X[F \setminus G]}$. (So, PQE takes G out of the scope of quantifiers.) The formula H is called a **solution** to PQE. The case of PQE where $G = F$ is called Quantifier Elimination (**QE**).*

Example 1. Consider formula $F = C_0 \wedge \dots \wedge C_4$ where $C_0 = \overline{x}_2 \vee x_3$, $C_1 = y_0 \vee x_2$, $C_2 = y_0 \vee \overline{x}_3$, $C_3 = y_1 \vee x_3$, $C_4 = y_1 \vee \overline{x}_3$. Let $Y = \{y_0, y_1\}$ and $X = \{x_2, x_3\}$. Consider the PQE problem of taking out G consisting of a single clause C_0. That is one needs to find $H(Y)$ such that $\exists X[F] \equiv H \wedge \exists X[F \setminus \{C_0\}]$. One can show that $\exists X[F] \equiv y_0 \wedge \exists X[F \setminus \{C_0\}]$ (see Subsection 3.2) i.e., $H = y_0$ is a solution to this PQE problem.

Definition 10. *Let clauses C',C'' have opposite literals of exactly one variable $w \in Vars(C') \cap Vars(C'')$. Then C',C'' are called **resolvable** on w. Let C be the clause consisting of the literals of C' and C'' minus those of w. Then C is said to be obtained by **resolution** of C' and C'' on w.*

Definition 11. *Let C be a clause of a formula F and $w \in Vars(C)$. The clause C is said to be* ***blocked*** *[6] in F at the variable w if no clause of F is resolvable with C on w.*

Proposition 1. *Let a clause C be blocked in a formula $F(X,Y)$ with respect to a variable $x \in X$. Then C is redundant in $\exists X[F]$, i.e., $\exists X[F \setminus \{C\}] \equiv \exists X[F]$.*

The proofs of propositions are given in Appendix A.

3 PQE solving

In this section, we briefly describe the PQE algorithm called *DS-PQE* [2]. Our objective here is just to give an idea of how the PQE problem can be solved. So, in Subsection 3.1, we present a high-level description of this algorithm. (The pseudo-code of *DS-PQE* is given in Appendix B.) Subsections 3.2 and 3.3 provide examples of PQE solving.

3.1 High-level view

Like all existing PQE algorithms, *DS-PQE* uses *redundancy based reasoning* justified by the proposition below.

Proposition 2. *Formula $H(Y)$ is a solution to the PQE problem of taking G out of $\exists X[F(X,Y)]$ (i.e., $\exists X[F] \equiv H \wedge \exists X[F \setminus G]$) iff*

1. *$F \Rightarrow H$ and*
2. *$H \wedge \exists X[F] \equiv H \wedge \exists X[F \setminus G]$*

Thus, to take G out of $\exists X[F(X,Y)]$, it suffices to find a formula $H(Y)$ implied by F that makes G *redundant* in $H \wedge \exists X[F]$. We will refer to the clauses of G as **target** ones. Below, we provide some basic facts about *DS-PQE*. Since taking out an unquantified clause is trivial, we assume that G contains only *quantified* clauses. *DS-PQE* finds a solution to the PQE problem above by branching on variables of F. The idea here is to reach a subspace $\vec{q}$ where every clause of G can be easily proved or made redundant in $\exists X[F]$. Like a SAT-solver, *DS-PQE* runs Boolean Constraint Propagation (BCP). If a conflict occurs in subspace $\vec{q}$, *DS-PQE* generates a conflict clause K and adds it to F to *make* clauses of G redundant in subspace $\vec{q}$. However, most frequently, proving redundancy of G in a subspace does not require a conflict or adding a new clause. Importantly, *DS-PQE* branches on unquantified variables, i.e., those of Y, *before* quantified ones. So, when *DS-PQE* produces a conflict clause K from clauses of F, the quantified variables are resolved out *before* unquantified.

If a target clause C becomes unit[2] in subspace $\vec{q}$, *DS-PQE temporarily* extends the set of target clauses G. Namely, *DS-PQE* adds to G every clause

[2] An unsatisfied clause is called *unit* if it has only one unassigned literal. Due to special decision making of *DS-PQE* (variables of Y are assigned before those of X), if the target clause C becomes unit, its unassigned variable is always in X.

that is resolvable with C on its only unassigned variable (denote this variable as x). This is done to facilitate proving redundancy of C. If the added clauses are proved redundant in subspace $\vec{q}$, the clause C is blocked at x and so is redundant in subspace $\vec{q}$. The fact that *DS-PQE* extends G means that it may need to prove redundancy of clauses other than those of G. The difference is that every clause of the original set G must be proved redundant *globally* whereas clauses added to G need to be proved redundant only *locally* (in some subspaces).

To express the redundancy of a clause C in a subspace $\vec{q}$, *DS-PQE* uses a record $(\exists X[F], \vec{q}\,) \rightarrow C$ called a **D-sequent**. It states the redundancy of C in the current formula $\exists X[F]$ in subspace $\vec{q}$. This D-sequent also holds in any formula $\exists X[F^*]$ where F^* is obtained from F by adding clauses implied by F. So, we will **omit** the first parameter and will write the D-sequent above as $\vec{q} \rightarrow C$. We will assume that it represents the redundancy of C in subspace $\vec{q}$ for the *current formula* whatever it is. For the sake of simplicity, in this section we use "regular" D-sequents presented in [4]. In the next section, we will recall extended D-sequents introduced in [5] that will be used for the rest of the paper.

A D-sequent derived for a target clause when its redundancy is easy to prove is called **atomic**. D-sequents derived in different branches can be resolved similarly to clauses[3]. For every target clause C of the original set G, *DS-PQE* uses such resolution to eventually derive the D-sequent $\emptyset \rightarrow C$. The latter states that C is globally redundant in the final formula $\exists X[F]$. At this point *DS-PQE* terminates. The solution $H(Y)$ to the PQE problem found by *DS-PQE* consists of the unquantified clauses added to the initial formula $\exists X[F]$ to make G redundant.

3.2 An example of PQE solving

Here we show how *DS-PQE* solves Example 1 introduced in Section 2. Recall that one takes $G = \{C_0\}$ out of $\exists X[F(X,Y)]$ where $F = C_0 \wedge \dots \wedge C_4$ and $C_0 = \overline{x}_2 \vee x_3$, $C_1 = y_0 \vee x_2$, $C_2 = y_0 \vee \overline{x}_3$, $C_3 = y_1 \vee x_3$, $C_4 = y_1 \vee \overline{x}_3$ and $Y = \{y_0, y_1\}$ and $X = \{x_2, x_3\}$. That is, one needs to find $H(Y)$ such that $\exists X[F] \equiv H \wedge \exists X[F \setminus \{C_0\}]$.

Assume that *DS-PQE* picks the variable y_0 for branching and first explores the branch $\vec{q}\,' = (y_0 = 0)$. In subspace $\vec{q}\,'$, clauses C_1, C_2 become unit. After assigning $x_2 = 1$ to satisfy C_1, the clause C_0 turns into unit too and a conflict occurs (to satisfy C_0 and C_2, one has to assign the opposite values to x_3). After a standard conflict analysis [7], the conflict clause $K = y_0$ is obtained by resolving C_1 and C_2 with C_0. To *make* C_0 redundant in subspace $\vec{q}\,'$, *DS-PQE* adds K to F. The redundancy of C_0 is expressed by the D-sequent $\vec{q}\,' \rightarrow C_0$. This D-sequent is an example of an *atomic* D-sequent. It asserts that C_0 is redundant in the current formula $\exists X[F]$ in subspace $\vec{q}\,'$. More information about D-sequents is given in Sections 5-7.

Having finished the first branch, *DS-PQE* considers the second branch: $\vec{q}\,'' = (y_0 = 1)$. Since C_1 is satisfied by $\vec{q}\,''$, no clause of F is resolvable with C_0 on variable x_2 in subspace $\vec{q}\,''$. Hence, C_0 is blocked at variable x_2 and thus redundant

[3] In the previous papers (e.g., [4]) we called the operation of resolving D-sequents *join*.

in $\exists X[F]$ in subspace $\vec{q}''$. So, *DS-PQE* generates the D-sequent $\vec{q}'' \rightarrow C_0$. This D-sequent is another example of an *atomic* D-sequent. It states that C_0 is *already* redundant in $\exists X[F]$ in subspace $\vec{q}''$ (without adding a new clause). Then *DS-PQE* resolves the D-sequents $(y_0 = 0) \rightarrow C_0$ and $(y_0 = 1) \rightarrow C_0$ above on y_0. This resolution produces the D-sequent $\emptyset \rightarrow C_0$ stating the redundancy of C_0 in $\exists X[F]$ in the *entire* space (i.e., globally). Recall that $F_{fin} = K \wedge F_{init}$ where F_{fin} and F_{init} denote the final and initial formula F respectively. That is K is the only unquantified clause added to F_{init}. So, *DS-PQE* returns $K = y_0$ as a solution $H(Y)$. The clause K is indeed a solution since it is implied by F_{init} and C_0 is redundant in $K \wedge \exists X[F_{init}]$. So both conditions of Proposition 2 are met and thus $\exists X[F_{init}] \equiv y_0 \wedge \exists X[F_{init} \setminus \{C_0\}]$.

3.3 An example of adding temporary targets

Let $F = C_0 \wedge C_1 \wedge C_2 \wedge \dots$ where $C_0 = y_0 \vee x_1$, $C_1 = \overline{x}_1 \vee x_2 \vee x_3$, $C_2 = \overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3$. Let C_1 and C_2 be the only clauses of F with the literal $\overline{x}_1$. Consider the problem of taking formula G out of $\exists X[F(X,Y)]$ where $G = \{C_0\}$ (we assume that $y_0 \in Y$ and $x_1, x_2, x_3 \in X$). Suppose *DS-PQE* explores the branch $\vec{q} = (y_0 = 0)$. In the subspace $\vec{q}$, the target clause C_0 turns into the unit clause x_1. In this case, *DS-PQE* adds to the set of targets G the clauses C_1 and C_2 i.e., the clauses that are resolvable with C_0 on x_1.

The idea here is to facilitate proving redundancy of C_0 in subspace $\vec{q}$. Namely, if C_1 and C_2 are proved redundant in subspace $\vec{q}$, the target C_0 becomes *blocked* and hence redundant in subspace $\vec{q}$. Clauses C_1, C_2 are added to G only in subspace $\vec{q}$ i.e., *temporarily*. As soon as *DS-PQE* backtracks from this subspace, C_1, C_2 are removed from G.

4 Extended D-sequents

Our results on D-sequent reusing presented in the next three sections are formulated in terms of extended D-sequents. The latter were introduced in [5]. In Subsection 4.1, we explain the motivation for extending the notion of a D-sequent. Subsection 4.2 presents extended *atomic* D-sequents. In Subsection 4.3 we introduce the *resolution* of extended D-sequents. (In Subsection 3.2 we already gave examples of atomic D-sequents and resolution of in terms of regular D-sequents.) For the sake of clarity, we will describe the machinery of extended D-sequents by the example of *DS-PQE*. However, the contents of this and the next three sections applies to any branching algorithm based on extended D-sequents.

4.1 Motivation for extending the notion of a D-sequent

Example 2 below shows that an indiscriminate reusing of "regular" D-sequents introduced in [4] can lead to *circular reasoning*. Definition 12 extends the notion of a D-sequent [5] to avoid circular reasoning.

Example 2. Let formula $\exists X[F(X,Y)]$ have clauses $C_0 = x_0 \vee x_2 \vee x_3$ and $C_1 = x_1 \vee x_2 \vee x_3$. Suppose in some branch $(\dots, x_0 = 0, \dots)$ *DS-PQE* derived the D-sequent $(x_0 = 0) \rightarrow C_1$ due to the fact that C_0 implies C_1 when $x_0 = 0$. (So C_1 is redundant in subspace $x_0 = 0$.) Suppose in another branch $(\dots, x_1 = 0, \dots)$ *DS-PQE* derived the D-sequent $(x_1 = 0) \rightarrow C_0$ due to the fact that C_1 implies C_0 when $x_1 = 0$. Suppose *DS-PQE* explores the branch $\vec{q} = (x_0 = 0, x_1 = 0)$. Although these D-sequents hold in subspace $\vec{q}$ *individually*, applying them *together* means claiming that *both* C_0 and C_1 are redundant in subspace $\vec{q}$. This is the wrong conclusion derived by circular reasoning ■

Definition 12. *Given a formula $\exists X[F]$, an **extended** D-sequent is a record $(\vec{q}, U) \rightarrow C$ stating that a clause C is redundant in formula $\exists X[F]$ in subspace $\vec{q}$. The set U specifies the set of quantified clauses used to prove redundancy of C in subspace $\vec{q}$. The assignment $\vec{q}$ is called the **conditional** of this D-sequent. The set U is called its **construction set**.*

The idea of [5] is to reuse an extended D-sequent $(\vec{q}, U) \rightarrow C$ *only* if all the clauses of U are still present in $\exists X[F]$ in subspace $\vec{q}$. Consider Example 2 above. Note that C_0 was used to prove C_1 redundant and vice versa. So, the extension of the D-sequents mentioned there looks like $(\vec{q}, U') \rightarrow C_1$ where $U' = \{C_0\}$ and $(\vec{q}, U'') \rightarrow C_0$ where $U'' = \{C_1\}$. (More details about atomic extended D-sequents are given in the next subsection.) If, for instance, one applies $(\vec{q}, U') \rightarrow C_1$ to remove C_1 from $\exists X[F]$ in subspace $\vec{q}$, the D-sequent $(\vec{q}, U'') \rightarrow C_0$ *cannot* be reused because $C_1 \in U''$ has been removed. Thus, one avoids circular reasoning.

4.2 Extended atomic D-sequents

Definition 13. *Suppose DS-PQE takes a set of clauses G out of $\exists X[F]$. Suppose DS-PQE entered a subspace $\vec{q}$ and C is a clause of G. DS-PQE derives a D-sequent called **atomic** if one of the conditions below is met*

- *$\vec{q}$ satisfies C*
- *$C_{\vec{q}}$ is implied by $C'_{\vec{q}}$ where C' is another clause of F*
- *$C_{\vec{q}}$ is blocked in $F_{\vec{q}}$ at an unassigned variable $x \in X$*

Now we describe how extended atomic D-sequents are built. Assume *DS-PQE* enters a subspace $\vec{q}$ where C is **satisfied**. Then *DS-PQE* derives a D-sequent $(\vec{r}, U) \rightarrow C$ where $U = \emptyset$ and $\vec{r}$ is a smallest subset of $\vec{q}$ still satisfying C. Now, assume that C is **implied** by another clause C' of F in subspace $\vec{q}$. Then *DS-PQE* derives a D-sequent $(\vec{r}, U) \rightarrow C$ where $\vec{r}$ is the smallest subset of $\vec{q}$ such that C is still implied by C' in subspace $\vec{r}$. Here $U = \{C'\}$ because C is redundant due to the presence of C'.

Finally, assume that C is **blocked** in subspace $\vec{q}$ at a variable $x \in X$. Then every clause of F resolvable with C on x is either satisfied by $\vec{q}$ or proved redundant in subspace $\vec{q}$. Let $C_0, \dots, C_k$ be the clauses of F resolvable with C on x that are proved redundant in subspace $\vec{q}$. Let $(\vec{q}_0, U_0) \rightarrow C_0, \dots, (\vec{q}_k, U_k) \rightarrow C_k$, be the D-sequents stating the redundancy of those clauses where $\vec{q_i} \subseteq \vec{q}$, $i = 0, \dots, k$. Then *DS-PQE* derives a D-sequent $(\vec{r}, U) \rightarrow C$ where $\vec{r}$ is a smallest subset of $\vec{q}$ such that C is still blocked at x in subspace $\vec{r}$. The construction set U equals $U_0 \cup \dots \cup U_k$.

Example 3. Let $C = x_0 \vee x_1$ be a target clause and *DS-PQE* enters subspace $\vec{q} = (x_0 = 1, x_1 = 0, \dots)$ where C is satisfied. Then *DS-PQE* derives the D-sequent $(\vec{r}, U) \rightarrow C$ where $\vec{r} = (x_0 = 1)$ is the smallest subset of $\vec{q}$ that still satisfies C. The construction set U is empty.

4.3 Resolution of extended D-sequents

Definition 14. *Let $(\vec{q}\,', U') \rightarrow C$ and $(\vec{q}\,'', U'') \rightarrow C$ be extended D-sequents. Let the conditionals $\vec{q}\,'$ and $\vec{q}\,''$ have the following property: precisely one variable v of $Vars(\vec{q}\,') \cap Vars(\vec{q}\,'')$ has different values in $\vec{q}\,'$ and $\vec{q}\,''$. Let $\vec{q}$ be equal to $\vec{q}\,' \cup \vec{q}\,'$ minus the assignments to v and $U = U' \cup U''$. The D-sequent $(\vec{q}, U) \rightarrow C$ is said to be obtained by the* ***resolution*** *of the D-sequents above on v.*

Example 4. Consider the D-sequents $(\vec{q}\,', U') \rightarrow C$ and $(\vec{q}\,'', U'') \rightarrow C$ where $\vec{q}\,' = (x_0 = 0, x_1 = 1, x_2 = 1)$ and $\vec{q}\,'' = (x_0 = 0, x_1 = 0, x_3 = 0)$. The conditionals $\vec{q}\,'$ and $\vec{q}\,''$ have exactly one variable that is assigned differently in them (namely x_1). So, these D-sequents can be resolved on x_1. The result of resolution is the D-sequent $(\vec{q}, U) \rightarrow C$ where $\vec{q} = (x_0 = 0, x_2 = 1, x_3 = 0)$ and $U = U' \cup U''$.

5 A High-Level View Of D-sequent Reusing

The main flaw of current PQE solvers is that they do not reuse learned information. In Appendix C we show experimentally that due to the lack of D-sequent reusing *DS-PQE* generates the same D-sequents over and over again. This suggests that the performance of PQE solvers like *DS-PQE* can be dramatically improved by reusing learned D-sequents. The same applies to any algorithms based on the machinery of D-sequents. In this section, we give a high-level view of D-sequent reusing and in the following two sections we describe two important theoretical results facilitating such reusing. As before, for the sake of clarity, we describe D-sequent reusing by the example of *DS-PQE*.

5.1 Difference between reusing conflict clauses and D-sequents

Consider the SAT problem specified by $\exists X[F(X)]$. A typical SAT solver enumerates only subspaces where the formula is unsatisfiable (i.e., locally inconsistent). As soon as a satisfying assignment is found, this SAT solver terminates. Inconsistency is a *semantic* property. That is, if a formula R is unsatisfiable and $R' \equiv R$, then formula R' is unsatisfiable too. For this reason, one can easily easily reuse a conflict clause (specifying local inconsistency) in different contexts.

Now consider the PQE problem of taking a set of clauses G out of $\exists X[F(X, Y)]$. A PQE algorithm solves this problem by deriving D-sequents stating redundancy of clauses of G in subspaces. One can relate such redundancy with local *unobservability*[4]. We mean that if a clause C is redundant in $\exists X[F]$ in subspace $\vec{q}$, the presence of C does not affect the result of solving the PQE problem at hand in this subspace. So, C is "unobservable" there. (Appendix D gives an example of employing D-sequents to express the unobservability of a subcircuit in a subspace.) Note that redundancy and hence unobservability is a *structural* rather than semantic property. Namely, if a clause B is redundant in formula R and $R' \equiv R$, this *does not* necessarily mean that B is redundant in R' too. This fact makes reusing D-sequents *non-trivial* because it depends on the context.

5.2 Clarifying what D-sequent reusing means

A few definitions below clarify what safe reusing of a D-sequent means.

Definition 15. *We will say that a D-sequent* $(\vec{q},U) \rightarrow C$ ***holds*** *for a formula* $\exists X[F]$ *if* $C_{\vec{q}}$ *is redundant in* $\exists X[F_{\vec{q}}]$ *(i.e.,* C *is redundant in* $\exists X[F]$ *in subspace* $\vec{q}$*).*

Definition 16. *We will call assignments* $\vec{q}\,'$ *and* $\vec{q}\,''$ ***consistent*** *if every variable of* $Vars(\vec{q}\,') \cap Vars(\vec{q}\,'')$ *has identical values in* $\vec{q}\,'$ *and* $\vec{q}\,''$*.*

Definition 17. *Let* $(\vec{q}_0,U_0) \rightarrow C_0,\dots,(\vec{q}_i,U_i) \rightarrow C_i$ *be a set of D-sequents with consistent conditionals that hold for* $\exists X[F]$ *individually. Let* $\vec{q}\,'$ *be an arbitrary assignment where* $\vec{q}\,' \supseteq \vec{q}_0,\dots,\vec{q}\,' \supseteq \vec{q}_i$*. The reuse of the D-sequents above means removing the clauses* $\{C_0,\dots,C_i\}$ *together from* $\exists X[F]$ *in subspace* $\vec{q}\,'$ *as redundant. We call such a reuse* ***safe*** *if these clauses are indeed jointly redundant in subspace* $\vec{q}\,'$*.*

6 Reusing Single D-sequent And Scale-Down Property

According to Definition 17, to be safely reused, a D-sequent $(\vec{q},U) \rightarrow C$ must have at least the property that, for every $\vec{q}\,' \supseteq \vec{q}$, the D-sequent $(\vec{q}\,',U) \rightarrow C$ holds too. (From now on when we say a D-sequent we mean an **extended** D-sequent.) That is, if C is redundant in subspace $\vec{q}$ it should be redundant in any subspace *contained* in subspace $\vec{q}$. If so, we will say that the D-sequent above has the **scale-down** property. In general, redundancy of a clause does not scale down (see Example 5). Previously, to get around this problem, we used a convoluted and counter-intuitive definition of clause redundancy in a subspace [8]. In this section, we describe a much nicer and simpler solution based on the notion of *constructive* D-sequents. The latter are exactly the D-sequents produced by the existing algorithms. We show that constructive D-sequents either have the scale-down property (Proposition 3) or violating this property does not affect their reusability (Remark 1).

[4] In general, when solving a problem involving a quantified formula one has to enumerate subspaces where this formula is unsatisfiable and those where it is *satisfiable*. So, one cannot get away with just recording local inconsistencies as it is done in SAT solving.

Example 5. Let $\exists X[F(X)]$ be a formula where $F = C_0 \wedge C_1 \wedge C_2$ and $C_0 = x_0 \vee x_1$, $C_1 = x_0 \vee \overline{x}_1$, $C_2 = \overline{x}_0 \vee x_1$ and $X = \{x_0, x_1\}$. Since F is satisfiable, C_0 is redundant in $\exists X[F]$ in the entire space (because $\exists X[F] = \exists X[F \setminus \{C_0\}] = 1$). Let us show that this redundancy *does not scale down*. Consider the subspace $\vec{q}\,' = (x_0 = 0)$. In this subspace, $(C_0)_{\vec{q}\,'} = x_1$, $(C_1)_{\vec{q}\,'} = \overline{x}_1$, $(C_2)_{\vec{q}\,'} \equiv 1$. So, the formula F is unsatisfiable in subspace $\vec{q}\,'$. Note that removing C_0 makes F satisfiable in subspace $\vec{q}\,'$. So, C_0 is *not* redundant in subspace $\vec{q}\,'$ being redundant in the entire space.

Definition 18. *A D-sequent is called* ***constructive*** *if*

- *it is an atomic D-sequent or*
- *it is obtained by resolving two constructive D-sequents*

Proposition 3. *Let $(\vec{q}, U) \rightarrow C$ be a constructive D-sequent stating the redundancy of a clause C in $\exists X[F]$ in subspace $\vec{q}$. Assume that its derivation does not involve D-sequents stating redundancy of clauses other than C. Then this D-sequent has the scale-down property.*

Remark 1. Let D denote the D-sequent $(\vec{q}, U) \rightarrow C$ from Proposition 3. In this proposition, we assume that when deriving D one did not use redundancy of clauses different from C. This assumption essentially means that no clause of U has been removed as redundant in subspace $\vec{q}$ by the time D is derived. If this assumption is broken, one cannot guarantee that D has the scale-down property. However, as we show in the next section one can still safely reuse D even if some clauses of U are removed as redundant in subspace $\vec{q}$. The only exception (i.e., the case when D *cannot* be reused) is given in Proposition 4.

Example 6. Let us show how one can express the redundancy of clause C_0 from Example 5 by a constructive D-sequent. One can build such a D-sequent by branching on x_0. In the branch $x_0 = 0$, one has to add to F the conflict clause $K = x_0$ to *make* C_0 redundant in subspace $x_0 = 0$. Then the atomic D-sequent $(\vec{q}\,', U') \rightarrow C_0$ holds for $\exists X[F \wedge K]$ where $\vec{q}\,' = (x_0 = 0)$ and $U' = \{K\}$. In the branch $x_1 = 1$, the atomic D-sequent $(\vec{q}\,'', U'') \rightarrow C_0$ holds since C_0 is satisfied by $\vec{q}\,'' = (x_0 = 1)$. Here $U'' = \emptyset$. By resolving these D-sequents on x_0 one derives a constructive D-sequent $(\vec{q}, U) \rightarrow C_0$ for the formula $\exists X[F \wedge K]$ where $\vec{q} = \emptyset$ and $U = \{K\}$. This D-sequent expresses the global redundancy of C in $\exists X[F \wedge K]$ (instead of $\exists X[F]$) and has the scale-down property.

7 Reusing Multiple D-sequents

In [5] some severe limitations were imposed on reusing an extended D-sequent. This was done to avoid circular reasoning when multiple D-sequents are reused *jointly*. In this section, we show that one can *lift those limitations* without risking to produce the wrong result. Arguably, this will make reusing D-sequents dramatically more effective.

Let C be a clause of a formula $\exists X[F]$. Let D denote an D-sequent $(\vec{q},U) \rightarrow C$. In [5], we showed that D can be safely reused only if all clauses of U are present in the current formula i.e., none of the clauses of U is removed from $\exists X[F]$ in subspace $\vec{q}$ as satisfied or redundant. This is very restrictive. Indeed, a clause of U can be in two states: present/not present in the formula. So, the total number of possible states of U is $2^{|U|}$. The restriction above means that D can be reused only in **one** of $2^{|U|}$ cases. The proposition below **lifts this restriction**.

Proposition 4. *Given a formula $\exists X[F]$ and a derived D-sequent $(\vec{q},U) \rightarrow C$, the latter can be safely reused in subspace $\vec{q}^* \supseteq \vec{q}$ with one exception. Namely, this D-sequent cannot be reused if another D-sequent $(\vec{q}',U') \rightarrow C'$ was applied earlier where $C' \in U$ and $C \in U'$ (and $\vec{q}^* \supseteq \vec{q}'$).*

Proposition 4 dramatically boosts the applicability of a D-sequent $(\vec{q},U) \rightarrow C$. It allows to reuse this D-sequent *regardless* of whether clauses of U are present in subspace $\vec{q}^*$. The only *exception* to such reusing is that a removed clause $C' \in U$ is proved redundant using C itself. (Then reusing $(\vec{q},U) \rightarrow C$ leads to circular reasoning.)

Example 7. Let $\exists X[F]$ be a formula where $F = C_0 \wedge C_1 \wedge C_2 \wedge \dots$ and $C_0 = x_0 \vee x_1 \vee x_2$, $C_1 = x_1 \vee x_2 \vee x_3$, $C_2 = \overline{x}_0 \vee \overline{x_2}$, and $x_i \in X$. Since C_0 implies C_1 in subspace $\vec{q} = (x_0 = 0)$, the D-sequent $(\vec{q},U) \rightarrow C_1$ holds where $U = \{C_0\}$. It states that C_0 was used to prove redundancy of C_1 in subspace $\vec{q}$. Assume that C_2 is the only clause of F with the literal $\overline{x}_0$. Then F has no clauses resolvable with C_0 on x_0 and hence C_0 is blocked at x_0. So, the D-sequent $(\vec{q}',U') \rightarrow C_0$ holds where $\vec{q}' = \emptyset$, $U' = \emptyset$ and C_0 can be removed from F as redundant. According to Proposition 4, the D-sequent $(\vec{q},U) \rightarrow C_1$ still can be *safely reused* even though the clause C_0 of U is removed from the formula.

8 Some Background

Information on QE in propositional logic can be found in [9,10,11,12,13,14]. QE by redundancy based reasoning is presented in [3,4]. One of the merits of such reasoning is that it allows to introduce *partial* QE. A description of PQE algorithms and their sources can be found in [2,1,15,16,17].

Removal of redundancies is used in pre-processing procedures [18,19,20]. Such procedures typically look for shallow redundancies that can be easily identified (e.g. they search for clauses that are trivially blocked). The objective here is to *minimize/simplify* the formula at hand. A PQE solver can also be viewed as a tool for identifying redundancies. In particular, in some applications (see e.g., [1]) it is simply used to check if a subset of clauses G is redundant in $\exists X[F]$. This version of the problem is called a *decision* PQE problem. However, the goal here is to solve the problem rather than optimize $\exists X[F]$. So, the PQE solver keeps proving redundancy of G even if this requires exploring a deep search tree. The general case of the PQE problem where one takes G out of the scope of quantifies is even further away from optimization of $\exists X[F]$. Here one needs to *make* G redundant by adding a formula $H(Y)$ whose size can be much larger than that of G.

In [15] we introduced the notion of a certificate clause that can be viewed as an advanced form of a D-sequent. The idea here is to prove a clause C redundant in subspace $\vec{q}$ by deriving a clause K that *implies* C in subspace $\vec{q}$. The clause K is called a certificate. The advantage of a certificate clause over a D-sequent is that the former can be added to $\exists X[F]$ (because $\exists X[F] \equiv \exists X[F \wedge K]$).

The introduction of certificate clauses addresses the problem of reusing the information learned by a PQE solver. The presence of K in the formula makes the redundancy of the clause C above in subspace $\vec{q}$ obvious. However the machinery of certificates has the following flaw. The certificate K above can be represented as $C' \vee C''$ where C' is falsified by $\vec{q}$ and C'' consists of literals of C. If C'' contains all the literals of C, then adding K to $\exists X[F]$ does not make sense (because K is implied by C). If one uses K as a proof of redundancy in subspace $\vec{q}$ *without* adding it to $\exists X[F]$, one runs into the same problem as with regular D-sequents. Namely, indiscriminate reusing of the certificate K can lead to circular reasoning.

9 Conclusions

We consider the problem of taking a subset of clauses G out of the scope of quantifiers in a propositional formula $\exists X[F]$. We refer to this problem as partial quantifier elimination (PQE). We solve PQE using the machinery of D-sequents where a D-sequent is a record stating that a clause C is redundant in $\exists X[F]$ in subspace $\vec{q}$. We use this machinery because in contrast to a SAT algorithm enumerating only the subspaces where the formula is unsatisfiable, a PQE solver has to also enumerate those where the formula is *satisfiable*. The main flaw of the current PQE solvers is that they do not reuse D-sequents. Reusing D-sequents is not trivial because redundancy of a clause is a structural rather than semantic property. We show that by using an extended version of D-sequents one can safely reuse them. The next natural step is show the viability of our approach *experimentally*.

References

1. E. Goldberg, "Partial quantifier elimination and propertry generation," in *Enea, C., Lal, A. (eds) Computer Aided Verification, CAV-23,Lecture Notes in Computer Science, Part II*, vol. 13965, 2023, pp. 110–131.
2. E. Goldberg and P. Manolios, "Partial quantifier elimination," in *Proc. of HVC-14*. Springer-Verlag, 2014, pp. 148–164.
3. ——, "Quantifier elimination by dependency sequents," in *FMCAD-12*, 2012, pp. 34–44.
4. ——, "Quantifier elimination via clause redundancy," in *FMCAD-13*, 2013, pp. 85–92.
5. E. Goldberg, "Quantifier elimination with structural learning," Tech. Rep. arXiv: 1810.00160 [cs.LO], 2018.

6. O. Kullmann, “New methods for 3-sat decision and worst-case analysis,” *Theor. Comput. Sci.*, vol. 223, no. 1-2, pp. 1–72, 1999.
7. J. Marques-Silva and K. Sakallah, “Grasp – a new search algorithm for satisfiability,” in *ICCAD-96*, 1996, pp. 220–227.
8. E. Goldberg and P. Manolios, “Quantifier elimination by dependency sequents,” *Formal Methods in System Design*, vol. 45, no. 2, pp. 111–143, 2014.
9. K. McMillan, “Applying sat methods in unbounded symbolic model checking,” in *Proc. of CAV-02.* Springer-Verlag, 2002, pp. 250–264.
10. J. Jiang, “Quantifier elimination via functional composition,” in *Proceedings of the 21st International Conference on Computer Aided Verification*, ser. CAV-09, 2009, pp. 383–397.
11. J. Brauer, A. King, and J. Kriener, “Existential quantification as incremental sat,” ser. CAV-11, 2011, pp. 191–207.
12. W. Klieber, M. Janota, J.Marques-Silva, and E. Clarke, “Solving qbf with free variables,” in *CP*, 2013, pp. 415–431.
13. N. Bjorner and M. Janota, “Playing with quantified satisfaction,” in *LPAR*, 2015.
14. M. Rabe, “Incremental determinization for quantifier elimination and functional synthesis,” in *CAV*, 2019.
15. E. Goldberg, “Partial quantifier elimination by certificate clauses,” Tech. Rep. arXiv:2003.09667 [cs.LO], 2020.
16. The source of *ds-pqe*, http://eigold.tripod.com/software/ds-pqe.tar.gz.
17. The source of *EG-PQE*$^+$, http://eigold.tripod.com/software/eg-pqe-pl.1.0.tar.gz.
18. N. Eén and A. Biere, “Effective preprocessing in sat through variable and clause elimination,” in *SAT*, 2005, pp. 61–75.
19. M. Järvisalo, M. Heule, and A. Biere, “Inprocessing rules,” ser. IJCAR-12, 2012, pp. 355–370.
20. R. Joseph, M. Heule, and R. Bryant, “Preprocessing of propagation redundant clauses,” in *Automated Reasoning.* Springer International Publishing, 2022, pp. 106–124.
21. E. Goldberg, “Some pqe problems that were generated off hwmcc-13 benchmarks,” https://eigold.tripod.com/benchmarks/hwmcc13_exmps.tar.gz.
22. HardWare Model Checking Competition 2013 (HWMCC-13), http://fmv.jku.at/hwmcc13/.

Appendix

A Proofs Of Propositions

A.1 Propositions of Section 2

Proposition 1. *Let a clause C be blocked in a formula $F(X,Y)$ with respect to a variable $x \in X$. Then C is redundant in $\exists X[F]$ i.e., $\exists X[F \setminus \{C\}] \equiv \exists X[F]$.*

Proof. It was shown in [6] that adding a clause $B(X)$ blocked in $G(X)$ to the formula $\exists X[G]$ does not change the value of this formula. This entails that removing a clause $B(X)$ blocked in $G(X)$ does not change the value of $\exists X[G]$ either. So, B is redundant in $\exists X[G]$.

Now, let us return to the formula $\exists X[F(X,Y)]$. Let $\vec{y}$ be a full assignment to Y. Then the clause C of the proposition at hand is either satisfied by $\vec{y}$ or $C_{\vec{y}}$

is blocked in $F_{\vec{y}}$ with respect to x. (The latter follows from the definition of a blocked clause.) In either case, $C_{\vec{y}}$ is redundant in $\exists X[F_{\vec{y}}]$. Since this redundancy holds in every subspace $\vec{y}$, the clause C is redundant in $\exists X[F]$.

A.2 Propositions of Section 3

Proposition 2. *Formula $H(Y)$ is a solution to the PQE problem of taking G out of $\exists X[F(X,Y)]$ (i.e., $\exists X[F] \equiv H \wedge \exists X[F \setminus G]$) iff*

1. *$F \Rightarrow H$ and*
2. *$H \wedge \exists X[F] \equiv H \wedge \exists X[F \setminus G]$*

Proof. **The if part.** Assume that conditions 1, 2 hold. Let us show that $\exists X[F] \equiv H \wedge \exists X[F \setminus G]$. Assume the contrary i.e., there is a full assignment $\vec{y}$ to Y such that $\exists X[F] \neq H \wedge \exists X[F \setminus G]$ in subspace $\vec{y}$.

There are two cases to consider here. First, assume that F is satisfiable and $H \wedge (F \setminus G)$ is unsatisfiable in subspace $\vec{y}$. Then there is an assignment $(\vec{x},\vec{y})$ satisfying F (and hence satisfying $F \setminus G$). This means that $(\vec{x},\vec{y})$ falsifies H and hence F does not imply H. So, we have a contradiction. Second, assume that F is unsatisfiable and $H \wedge (F \setminus G)$ is satisfiable in subspace $\vec{y}$. Then $H \wedge F$ is unsatisfiable too. So, condition 2 does not hold and we have a contradiction.
The only if part. Assume that $\exists X[F] \equiv H \wedge \exists X[F \setminus G]$. Let us show that conditions 1 and 2 hold. Assume that condition 1 fails i.e., $F \not\Rightarrow H$. Then there is an assignment $(\vec{x},\vec{y})$ satisfying F and falsifying H. This means that $\exists X[F] \neq H \wedge \exists X[F \setminus G]$ in subspace $\vec{y}$ and we have a contradiction. To prove that condition 2 holds, one can simply conjoin both sides of the equality $\exists X[F] \equiv H \wedge \exists X[F \setminus G]$ with H.

A.3 Propositions of Section 6

Lemmas 1-5 are used in the proof of Proposition 3.

Lemma 1. *Let $(\vec{q},U) \rightarrow C$ be a D-sequent that holds for a formula $\exists X[F(X,Y)]$ where C is a quantified clause of F. Then this D-sequent also holds for any formula $\exists X[F']$ obtained from $\exists X[F]$ by adding clauses implied by F.*

Proof. The fact that $(\vec{q},U) \rightarrow C$ holds means that $C_{\vec{q}}$ is redundant in $\exists X[F_{\vec{q}}]$. Denote $C_{\vec{q}}$ as C^* and $F_{\vec{q}}$ as F^*. Redundancy of C^* in $\exists X[F^*]$ means that there is no full assignment $\vec{y}$ to Y such that $F^*_{\vec{y}}$ is unsatisfiable and $(F^* \setminus \{C^*\})_{\vec{y}}$ is satisfiable. In other words, the fact that $F^*_{\vec{y}}$ is unsatisfiable implies that $(F^* \setminus \{C^*\})_{\vec{y}}$ is unsatisfiable too. The same is true if F^* (i.e., $F_{\vec{q}}$) is replaced with $F'_{\vec{q}}$.

Lemma 1 allows to "align" D-sequents derived at different times so that they can be resolved. Suppose a D-sequent $(\vec{q},U) \rightarrow C$ was derived for $\exists X[F]$. Suppose another D-sequent $(\vec{q}\,',U') \rightarrow C$ was derived later for a formula $\exists X[F']$ where F' was obtained from F by adding clauses implied by F. Then according to Lemma 1, the D-sequent $(\vec{q},U) \rightarrow C$ holds for $\exists X[F']$ too. So, by resolving these two D-sequents one obtains a D-sequent that holds for $\exists X[F']$.

Lemma 2. *Let C be a quantified clause of $\exists X[F]$. Let C be satisfied in subspace $\vec{q}$ and thus redundant there. Let $(\vec{q},U) \rightarrow C$ be the D-sequent stating this redundancy of C where $U = \emptyset$. Then this D-sequent has the scale-down property.*

Proof. If $\vec{q}$ satisfies C, then every $\vec{q}\,' \supseteq \vec{q}$ satisfies C too. So, $(\vec{q}\,',U) \rightarrow C$ holds.

Lemma 3. *Let C be a quantified clause of $\exists X[F]$. Let C be implied by another clause $B \in F$ in subspace $\vec{q}$ so, the D-sequent $(\vec{q},U) \rightarrow C$ holds where $U = \{B\}$. This D-sequent has the scale-down property.*

Proof. If B implies C in subspace $\vec{q}$, it also implies C in every subspace $\vec{q}\,' \supseteq \vec{q}$. So, $(\vec{q}\,',U) \rightarrow C$ holds.

Lemma 4. *Let C be a quantified clause of $\exists X[F]$. Let every clause of F resolvable with C on $x \in X$ be satisfied by an assignment $\vec{q}$. So, C is blocked at x in subspace $\vec{q}$ and hence redundant in this subspace. Then the D-sequent $(\vec{q},U) \rightarrow C$, $U = \emptyset$, stating this redundancy has the scale-down property.*

Proof. Let $\vec{q}\,'$ be a superset of $\vec{q}$. Then we have two possibilities. First, $\vec{q}\,'$ satisfies C. Then the latter is redundant in subspace $\vec{q}\,'$. Second, $\vec{q}\,'$ does not satisfy C. Then the fact that $\vec{q}$ satisfies all clauses of F resolvable with C on x implies that the same is true for $\vec{q}\,'$. Hence C is blocked in subspace $\vec{q}\,'$ at x. In either case, C is redundant in subspace $\vec{q}\,'$ and so the D-sequent $(\vec{q}\,',U) \rightarrow C$ holds.

Note that Lemma 4 holds only for a *special* case of being blocked. In general, when C is blocked in subspace $\vec{q}$, some clauses resolvable with C on x are *proved* redundant in subspace $\vec{q}$ rather than satisfied by $\vec{q}$. Then the construction set U can be non-empty.

Lemma 5. *Let C be a clause of a formula F. Let $(\vec{r},U^*) \rightarrow C$ and $(\vec{s},U^{**}) \rightarrow C$ be D-sequents that hold for $\exists X[F]$ and that can be resolved on a variable v. Let these D-sequents have the scale-down property. Then the D-sequent $(\vec{q},U) \rightarrow C$, $U = U^* \cup U^{**}$ obtained by resolving these D-sequents on v has the scale-down property too.*

Proof. Recall that by the definition of resolution of D-sequents, $\vec{q}$ equals $\vec{r} \cup \vec{s}$ minus assignments to v. Assume, for the sake of clarity, that $v = 0$ in $\vec{r}$ and $v = 1$ in $\vec{s}$. Let $\vec{q}\,' \supseteq \vec{q}$ hold. One needs to consider the following two possibilities. First, assume that $\vec{q}\,'$ assigns a value to v. If $v = 0$, then $\vec{q}\,' \supseteq \vec{r}$ holds and C is redundant in subspace $\vec{q}\,'$ (so, $(\vec{q}\,',U) \rightarrow C$ holds). If $v = 1$, then $\vec{q}\,' \supseteq \vec{s}$ holds and C is redundant in subspace $\vec{q}\,'$ as well. Second, assume that $\vec{q}\,'$ *does not* assign any value to v. Let us split the subspace $\vec{q}\,'$ into subspaces $\vec{q}\,'_0$ and $\vec{q}\,'_1$ where $\vec{q}\,'_0 = \vec{q}\,' \cup \{(v = 0)\}$ and $\vec{q}\,'_1 = \vec{q}\,' \cup \{(v = 1)\}$. Since $\vec{q}\,'_0 \supseteq \vec{r}$, the clause C is redundant in subspace $\vec{q}\,'_0$. Since $\vec{q}\,'_1 \supseteq \vec{s}$, C is redundant in subspace $\vec{q}\,'_1$ too. So, C is redundant in subspace $\vec{q}\,'$.

Proposition 3. *Let $(\vec{q}, U) \rightarrow C$ be a constructive D-sequent stating the redundancy of a clause C in $\exists X[F]$ in subspace $\vec{q}$. Assume that its derivation does not involve D-sequents stating redundancy of clauses other than C. Then this D-sequent has the scale-down property.*

Proof. Assume, for the sake of clarity, that D-sequents are generated by *DS-PQE*. Let $\mathbb{D}$ denote the set of D-sequents involved in generating the D-sequent at hand. By the assumption of our proposition, every D-sequent of $\mathbb{D}$ specifies redundancy of the clause C. We prove this proposition by induction on the order in which the D-sequents of $\mathbb{D}$ are derived by *DS-PQE*. Denote by D_n the D-sequent with the order number n. First, let us prove the base case i.e., this proposition holds for the first D-sequent D_1. Let D_1 be equal to $(\vec{q}_1, U_1) \rightarrow C$. The clause C is either satisfied, implied by another clause or blocked in subspace $\vec{q}_1$. Then according to Lemmas 2,3 and 4, D_1 has the scale-down property.

Now, we prove that the D-sequent D_n has the scale-down property assuming that this property holds for the D-sequents $D_1, \dots, D_{n-1}$. Let D_n be equal to $(\vec{q}_n, U_n) \rightarrow C$ and $\vec{q}\,' \supseteq \vec{q}_n$. Assume that C is redundant in subspace $\vec{q}_n$ because it is satisfied there or implied by an existing clause of F. Then, according to Lemmas 2,3, the D-sequent D_n has the scale-down property. Now, assume that C is blocked in subspace $\vec{q}_n$ at a variable $x \in X$. Since D-sequents of clauses other than C are not used in derivation of the target D-sequent $(\vec{q}, U) \rightarrow C$, the construction set U_n is empty. (If $U_n \neq \emptyset$, at least one clause resolvable with C on x is not present in the formula because it was proved redundant in subspace $\vec{q}$ rather than satisfied by $\vec{q}$.) Then, one can apply Lemma 4 to claim that D_n scales down. Finally, assume that D_n is obtained by resolving D-sequents D_i and D_j where $i, j < n$. Then according to Lemma 5, the D-sequent D_n has the scale-down property too.

A.4 Propositions of Section 7

Again, we assume, for the sake of clarity, that D-sequents are generated by *DS-PQE*. Before proving Proposition 4, we need to give a few definitions. Let D denote a D-sequent $(\vec{q}, U) \rightarrow C$. We will say that D is **active** if *DS-PQE* entered a subspace $\vec{q}\,'$ where $\vec{q}\,' \supseteq \vec{q}$ and temporarily removed the clause C as redundant. We will say that D is *reused* if it is an old D-sequent that was already active at least once and then became inactive when *DS-PQE* moved to another subspace. We will say that D is *used* if we do not care whether D is a new D-sequent applied for the first time or it is an old D-sequent being reused.

Proposition 4. *Given a formula $\exists X[F]$ and a derived D-sequent $(\vec{q}, U) \rightarrow C$, the latter can be safely reused in subspace $\vec{q}^{\,*} \supseteq \vec{q}$ with one exception. Namely, this D-sequent cannot be reused if another D-sequent $(\vec{q}\,', U') \rightarrow C'$ was applied earlier where $C' \in U$ and $C \in U'$ (and $\vec{q}^{\,*} \supseteq \vec{q}\,'$).*

Proof. We will prove this proposition by induction on the number of active D-sequents. Let D_n denote the n-th active D-sequent used by *DS-PQE*. Let us prove the base case that the D-sequent D_1 can be safely used. Let D_1 be equal to $(\vec{q}_1, U_1) \rightarrow C_1$. Since D_1 is the only active D-sequent so far no other D-sequent was used in derivation of D_1. Then from Proposition 3 it follows that D_1 holds in subspace $\vec{q}^*$.

Now, we prove the induction step. Namely, we show that a D-sequent D_n can be safely used under the assumption that the D-sequents $D_1, \dots, D_{n-1}$ have already been correctly applied. Let D_n be equal to $(\vec{q}_n, U_n) \rightarrow C_n$. If every clause of the set U_n is present $\exists X[F]$ or satisfied by the current assignment $\vec{q}^*$, then from Proposition 3 it follows that D_n can be safely used. Now assume that some clauses of U_n have been temporarily removed from $\exists X[F]$ as redundant. Let C_i be one of such clauses where $i < n$. Let D_i denote the D-sequent that was used by *DS-PQE* to remove C_i from $\exists X[F]$ in subspace $\vec{q}^*$. Let D_i be equal to $(\vec{q}_i, U_i) \rightarrow C_i$.

Let us temporarily add C_i (and the other clauses of U_n removed as redundant) back to $\exists X[F]$ to obtain a formula $\exists X[F^*]$. Adding these clauses to $\exists X[F]$ is a safe operation because they are *implied* by the original formula F. (*DS-PQE* adds only conflict clauses and they are implied by the original formula F.) Since every clause of U_n is either present in $\exists X[F^*]$ or satisfied, one can safely apply the D-sequent D_n to remove C_n from $\exists X[F^*]$. Our final step is to remove C_i (and the other clauses we temporarily added to the current formula) again. This can be done for the following reasons. First, by the induction hypothesis, the D-sequent D_i can be safely used for $\exists X[F]$ in subspace $\vec{q}^*$. Second, the proposition at hand implies that the set U_i of D_i does not contain C_n. So D_i holds for $\exists X[F \setminus \{C_n\}]$ too (because C_n was not used in proving C_i redundant). Third, using Lemma 1 one can claim that D_i holds for $\exists X[F^* \setminus \{C_n\}]$ and so can be safely used. After removing C_i and the other clauses that were temporarily added, one recovers the current formula $\exists X[F]$ but without the clause C_n. So, the D-sequent D_n has been safely applied to $\exists X[F]$ in subspace $\vec{q}^*$.

B Pseudocode of *DS-PQE*

The pseudocode of *DS-PQE* is shown in Figure 1. (A more detailed description of *DS-PQE* can be found in [2].) We assume here that *DS-PQE* derives "regular" D-sequents introduced in [4] and does not reuse them. Since *DS-PQE* recursively calls itself, it accepts four parameters: a formula $\exists X[F]$, the subset G of F to take out, an assignment $\vec{q}$ and a set of D-sequents *Ds*. This call of *DS-PQE* solves the PQE problem of taking G out of $\exists X[F]$ in subspace $\vec{q}$. The set *Ds* contains D-sequents of the clauses of F already proved redundant in subspace $\vec{q}$ (in earlier calls of *DS-PQE*). In the first call of *DS-PQE* both $\vec{q}$ and *Ds* are empty. *DS-PQE* returns the current formula F and the set *Ds* of D-sequents stating the redundancy of each clause of G in subspace $\vec{q}$, i.e., $|Ds| = |G|$. The solution $H(Y)$ of the PQE problem produced by *DS-PQE* consists of the unquantified clauses of the *final* formula $\exists X[F]$ added to the *initial* formula $\exists X[F]$. The initial (respectively final) formula $\exists X[F]$ is passed over in the first call of *DS-PQE* (respectively returned by the first call).

// ξ denotes the PQE problem
// i.e., $(\exists X[F], G)$
//
$\mathit{ds\text{-}pqe}(\xi, \vec{q}, \mathit{Ds})\{$
1 $\vec{q}^{\,*} := \mathit{bcp}(F, G, \vec{q})$ }
2 if $(\neg \mathit{done}(\xi, \vec{q}^{\,*}, \mathit{Ds}))$
3 goto *Branch*
4 if $(\mathit{confl}(F, \vec{q}^{\,*}))$ {
5 $K := \mathit{cnfl\_cls}(F, \vec{q}, \vec{q}^{\,*})$
6 $F := F \cup \{K\}$
7 $\mathit{Ds} := \mathit{dseqs}_1(\xi, \vec{q}, K, \mathit{Ds})\}$
8 else
9 $\mathit{Ds} := \mathit{dseqs}_2(\xi, \vec{q}, \vec{q}^{\,*}, \mathit{Ds})$
10 return(ξ, Ds)
11 *Branch* : //----------
12 $G := \mathit{add\_trgs}(F, G, \vec{q}^{\,*})$
13 $v := \mathit{pick\_var}(\vec{q}^{\,*}, X, Y)$
14 $\vec{q}^{\,\prime} := \vec{q}^{\,*} \cup \{v = 0\}$
15 $(F, \mathit{Ds}') := \mathit{ds\text{-}pqe}(\xi, \vec{q}^{\,\prime}, \mathit{Ds})$
16 if $(\mathit{done}(\xi, \vec{q}^{\,*}, \mathit{Ds}'))$
17 $\{\mathit{Ds} := \mathit{Ds}'$; goto *Finish*$\}$
18 $\vec{q}^{\,\prime\prime} := \vec{q}^{\,*} \cup \{v = 1\}$
19 $(F, \mathit{Ds}'') := \mathit{ds\text{-}pqe}(\xi, \vec{q}^{\,\prime\prime}, \mathit{Ds})$
20 $\mathit{Ds} := \mathit{res\_dseqs}(\mathit{Ds}', \mathit{Ds}'', v)$
21 *Finish* : //----------
22 $G := \mathit{rmv\_trgs}(G, \vec{q}, \vec{q}^{\,*})$
23 $\mathit{Ds} := \mathit{shorten}(F, \mathit{Ds}, \vec{q}, \vec{q}^{\,*})$
24 return$(\xi, \mathit{Ds})\}$

Fig. 1: *DS-PQE*

DS-PQE starts by checking if F has unit clauses in subspace $\vec{q}$ (line 1, of Fig. 1). If so, it runs Boolean Constraint Propagation (BCP) extending the assignment $\vec{q}$ to $\vec{q}^{\,*}$. Assume that there is at least one clause of G, for which no *atomic* D-sequent can be derived in subspace $\vec{q}^{\,*}$ (see Definition 13). Then *DS-PQE* goes to the *branching* part of the algorithm (lines 2-3). Otherwise, *DS-PQE* derives D-sequents for the clauses of G that were not proved redundant yet in subspace $\vec{q}$ and terminates the current call of *DS-PQE* (lines 4-10). Namely, if a conflict occurs in subspace $\vec{q}^{\,*}$, *DS-PQE* derives a conflict clause K that is falsified by $\vec{q}$ and adds it to F (lines 5-7). Then it constructs an atomic D-sequent $\vec{r} \rightarrow C$ for each target clause C that is not proved redundant yet in subspace $\vec{q}$. Here $\vec{r}$ is the smallest subset of $\vec{q}$ that falsifies K. This D-sequent states that C is redundant in any subspace where K is falsified. If no conflict occurs, each remaining clause C of G is proved redundant in subspace $\vec{q}$ by showing that it is satisfied, implied by an existing clause or blocked. Then an atomic D-sequent is derived for such a clause C (line 9). Finally, *DS-PQE* returns the current formula $\exists X[F]$ and the set of D-sequents *Ds* for the clauses of G (line 10).

The branching part of *DS-PQE* is shown in lines 12-20. First, *DS-PQE* extends the set of target clauses G (line 12). Namely, for each target clause C that becomes unit during BCP, *DS-PQE* adds to G the clauses of F that are resolvable with C on its unassigned variable. These new clauses are *temporarily* added to G. They are removed from G in the finish part (line 22). Then, *DS-PQE* picks a variable v to branch on (line 13). *DS-PQE* assigns unquantified variables (i.e., those of Y) before quantified (i.e., those of X). So, v is in X only if all variables of Y are already assigned. Then *DS-PQE* recursively calls itself to explore the branch $\vec{q}^{\,\prime} := \vec{q}^{\,*} \cup \{v = 0\}$ (lines 14-15). If the conditionals of the D-sequents from the set Ds' returned by *DS-PQE* do not depend on variable v, the branch $\vec{q}^{\,*} \cup \{v = 1\}$ can be skipped. So, *DS-PQE* jumps to the finishing part (lines 16-17). Otherwise, *DS-PQE* explores the branch $\vec{q}^{\,*} \cup \{v = 1\}$ (lines 18-19). Finally, *DS-PQE* resolves D-sequents of both branches whose conditionals depend on variable v (line 20). Namely, if the D-sequent of a clause $C \in G$ found in the first branch contains $v = 0$ in its conditional, it is resolved with the D-sequent of C found in the second branch on variable v (the conditional of this D-sequent contains the assignment $v = 1$).

The finish part of *DS-PQE* is shown in lines 22-24. In line 22, *DS-PQE* removes from G the target clauses added earlier in line 12. Then *DS-PQE* shortens the conditionals of the D-sequents of the target clauses by getting rid of assignments added to $\vec{q}$ by BCP (line 23). This procedure is similar to the conflict clause generation by a SAT-solver (where the latter eliminates from the conflict clause the implied assignments made at the conflict level during BCP). Finally, *DS-PQE* returns the current formula $\exists X[F]$ and the D-sequents generated for the clauses of G.

C The Benefit Of Reusing D-sequents

Table 1: Repeated generation of the same D-sequents

name	1		2		3		4	
	all	core	all	core	all	core	all	core
ex1	10,919	32	8,468	69	8,002	55	7,390	95
ex2	4,038	373	2,901	393	2,339	285	2,204	267
ex3	2,282	657	522	3	448	2	348	1
ex4	562	50	211	19	73	8	72	8
ex5	3,320	102	2,066	14	741	15	741	7
ex6	808	256	404	131	402	236	232	4
ex7	9,546	5	6,687	10	6,497	13	4,604	1

In this appendix, we describe an experiment showing that *DS-PQE* tends to repeatedly generate the same D-sequents. This experiment substantiates the claim of Section 5 that reusing D-sequents should boost the performance of PQE solving. In this experiment we used PQE problems that can be downloaded from [21]. Each PQE problem is to take a single clause out of a formula $\exists X[F]$ specifying the set of states of a sequential circuit reachable in n transitions. The circuits used in those PQE problems were taken from the HWMCC-13 set [22].

Table 2: Real names of benchmarks

short name	name used in [21]
ex1	6s106.10.unrem.probs.345
ex2	6s110.9.unrem.probs.3221
ex3	6s254.10.unrem.probs.955
ex4	6s255.10.unrem.probs.1535
ex5	6s255.10.unrem.probs.2167
ex6	bob12m05m.3.unrem.probs.5062
ex7	bob9234specmulti.10.unrem.probs.353

In this experiment, we used PQE problems that were too hard for *DS-PQE* to solve. So, for each problem, we just ran *DS-PQE* for 10 seconds and collected data on D-sequents it generated. In this data, we took into account only non-atomic D-sequents i.e., those whose generation involved at least one resolution on a *decision variable*. The reason for that is that the generation of non-atomic D-sequents is computationally harder than that of atomic D-sequents. So, the repeated production of the same non-atomic D-sequents is more costly. Table 1 provides results for a sample of 7 PQE problems. The first column of this table shows the short names assigned to the PQE problems. The names of the problems under which they are listed in [21] are given in Table 2.

Let $G = \{C\}$ be the original set of clauses to take out of $\exists X[F]$. As we mentioned in Section 3, *DS-PQE* temporarily extends G. So, *DS-PQE* generated D-sequents not only for the original clause of G but for many other clauses. In Table 1, for each PQE problem, we present results for the *four* clauses that were present in G for which *DS-PQE* generated the largest number of non-atomic D-sequents. So, the columns 2-9 of Table 1 specify four pairs. The first column of a pair shows the total number of non-atomic D-sequents generated for a particular target clause of G. The second column gives the number of the "core" D-sequents that were generated many times. Consider, for example, the first pair of columns of the instance *ex1*. The first and second columns of this pair give the numbers 10,919 and 32 respectively. It means that *DS-PQE* generated 10,919 non-atomic D-sequents but the latter were just repetitions of the same 32 D-sequents. The results of Table 1 show that *DS-PQE* generated a lot of identical D-sequents. So reusing learned D-sequents should be quite helpful.

D Unobservability And D-sequents

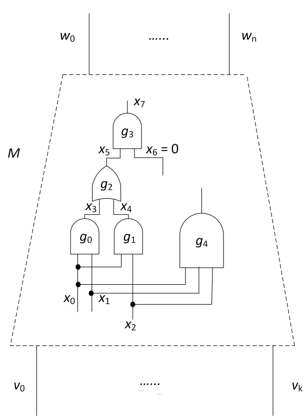

Fig. 2: Unobservable gates

In this appendix, we illustrate the relation between unobservability and D-sequents by an example. Consider the fragment of circuit $M(X,V,W)$ shown in Figure 2. Here X,V,W are sets of internal, input and output variables respectively. Let $F(X,V,W) = F_{g_0} \wedge F_{g_1} \wedge F_{g_2} \wedge F_{g_3} \wedge \dots$ be a formula specifying M where F_{g_i} describes the functionality of gate g_i. Namely, $\boldsymbol{F_{g_0}} = C_0 \wedge C_1 \wedge C_2$ where $C_0 = \overline{x_0} \vee \overline{x_1} \vee x_3$, $C_1 = x_0 \vee \overline{x_3}$, $C_2 = x_1 \vee \overline{x_3}$, $\boldsymbol{F_{g_1}} = C_3 \wedge C_4 \wedge C_5$ where $C_3 = \overline{x_0} \vee \overline{x_2} \vee x_4$, $C_4 = x_0 \vee \overline{x_4}$, $C_5 = x_2 \vee \overline{x_4}$, $\boldsymbol{F_{g_2}} = C_6 \wedge C_7 \wedge C_8$ where $C_6 = x_3 \vee x_4 \vee \overline{x_5}$, $C_7 = \overline{x_3} \vee x_5$, $C_8 = \overline{x_4} \vee x_5$, $\boldsymbol{F_{g_3}} = C_9 \wedge C_{10} \wedge C_{11}$ where $C_9 = \overline{x_5} \vee \overline{x_6} \vee x_7$, $C_{10} = x_5 \vee \overline{x_7}$, $C_{11} = x_6 \vee \overline{x_7}$.

Consider the problem of taking a set of clauses G out of $\exists X[F]$ where $x_i \in X, i = 0,\dots,7$. Assume, for the sake of simplicity, that $C_0,\dots,C_{11}$ are in G. Assume that this PQE problem is solved by *DS-PQE* that derives regular D-sequents introduced in [4]. We also assume that *DS-PQE* is currently in subspace $\vec{r} = (x_6 = 0)$. Note that the gates g_0, g_1, g_2 are "unobservable" in this subspace. That is their values do not affect the output of M no matter what input $\vec{v}$ is applied (as long as $\vec{v}$ produces the assignment $x_6 = 0$). Here $\vec{v}$ is a full assignment to $V = \{v_0,\dots,v_k\}$. Let us show that after entering the subspace $\vec{r}$, *DS-PQE* derives atomic D-sequents $\vec{r} \rightarrow C_i$,$i = 0,\dots,8$. These D-sequents express the unobservability of g_0, g_1, g_2 in subspace $\vec{r}$.

First consider the clauses of F_{g_3}. Since C_9 is satisfied by $\vec{r}$ and C_{10} is implied by the clause C_{11} in subspace $\vec{r}$, C_9 and C_{10} are removed $\exists X[F]$ as redundant in subspace $\vec{r}$. Now consider the clauses C_6, C_7, C_8 of F_{g_2}. Since the clauses C_9 and C_{10} are removed, the clauses of F_{g_2} are blocked at the variable x_5 in subspace $\vec{r}$. So, the D-sequents $\vec{r} \rightarrow C_i$, $i = 6,7,8$ are derived. Since the clauses of F_{g_2} are removed from the formula in subspace $\vec{r}$, the clauses of F_{g_0} and F_{g_1} are blocked at x_3 and x_4 respectively. So, the D-sequents $\vec{r} \rightarrow C_i$, $i = 0, \ldots, 5$ are derived.