Subsumption-Linear Q-Resolution for QBF Theorem Proving

Allen Van Gelder

Computer Science Dept., SOE–3, Univ. of California,
Santa Cruz, CA 95064 USA avg@cs.ucsc.edu

Abstract. Subsumption-Linear Q-Resolution (SLQR) is introduced for proving theorems from Quantified Boolean Formulas. It is an adaptation of SL-Resolution, which applies to propositional and first-order logic. In turn SL-Resolution is closely related to model elimination and tableau methods. A major difference from QDPLL (DPLL adapted for QBF) is that QDPLL guesses variable assignments, while SLQR guesses clauses. In prenex QBF (PCNF, all quantifier operations are outermost) a propositional formula D is called a nontrivial consequence of a QBF Ψ if Ψ is true (has at least one model) and D is true in every model of Ψ. Due to quantifiers, one cannot simply negate D and look for a refutation, as in propositional and first-order logic. Most previous work has addressed only the case that D is the empty clause, which can never be a nontrivial consequence.

This paper shows that SLQR with the operations of resolution on both existential and universal variables as well as universal reduction is inferentially complete for closed PCNF that are free of asymmetric tautologies; i.e., if D is logically implied by Ψ, there is a SLQR derivation of D from Ψ. A weaker form called SLQR–ures omits resolution on universal variables. It is shown that SLQR–ures is not inferentially complete, but is refutationally complete for closed PCNF.

1 Introduction

Theorem proving, i.e., showing that a given formula $\mathcal{F}$ logically implies another formula $\mathcal{G}$, is a fundamental task in any logic. We assume the reader is familiar with standard terminology of logic, as found in several texts [4, 8]. Recent work on high-performance solvers for propositional formulas and quantified boolean formulas (QBFs) has focused on determining a given formula's *satisfiability*, or *truth value*. For propositional formulas this emphasis is partly justified by the fact that $\mathcal{F}$ logically implies $\mathcal{G}$ if and only if $(\mathcal{F} \wedge \neg\mathcal{G})$ is *unsatisfiable*. The QBF analogy of this simple relationship does not hold. That is, $\overrightarrow{Q} \cdot (\mathcal{F} \wedge \neg\mathcal{G})$ may be false but $\overrightarrow{Q} \cdot \mathcal{F}$ does not logically imply $\overrightarrow{Q} \cdot \mathcal{G}$, where QBF logical implication is defined in Def. 1.1.

Definition 1.1 Let $\Phi = \overrightarrow{Q}.\mathcal{F}$ be a closed QBF; that is, $\overrightarrow{Q}$ is the quantifier prenex, $\mathcal{F}$ is a propositional formula, and every variable in $\mathcal{F}$ appears in the prenex. We say that a propositional formula D on the same set of variables as

$\mathcal{F}$ is a ***logical consequence*** of $\mathcal{F}$, written $\mathcal{F} \models D$, if D is true in every model of $\mathcal{F}$. We say that a propositional formula D on the same set of variables as $\mathcal{F}$ is a ***nontrivial consequence*** of Φ if Φ is true (has at least one model tree) and $\mathcal{F} \models D$. $\Box$

Due to quantifiers, one cannot simply negate D and look for a refutation, as in propositional and first-order logic. Most previous work has addressed only the case that D is the empty clause [3], which can never be a *nontrivial* consequence. We say a proof system is *inferentially complete* if any Ψ that is logically implied by Φ can be proven from Φ in the proof system; in other words all logical consequences of Φ are provable in the proof system.

All propositional formulas have a logically equivalent formula in *conjunctive normal form* (CNF), i.e., as a set of conjunctively joined *clauses* that are themselves disjunctively joined sets of literals. Propositional resolution is essentially inferentially complete for propositional CNF; technically, *clausal subsumption* is also needed in case a clause derived from $\mathcal{F}$ by resolution properly subsumes a clause that is a logical consequence of $\mathcal{F}$.

Alls QBFs have a logically equivalent formula in *prenex conjunctive normal form* (PCNF), i.e., all quantifiers are outermost operations and the remaining propositional formula, commonly called the *matrix*, is expressed in CNF. A QBF is said to be *closed* if every variable is quantified.

A further technical condition is important for inferential completeness: A QBF is said to be *AT-free* if it contains no *asymmetric tautologies*, as defined and studied by Heule *et al.* [6]. QBFs translated from applications are normally AT-free, but certain preprocessing operations might introduce asymmetric tautologies.

Although the precise definition is quite technical, a simple example of asymmetric tautology is a set of clauses in which a certain variable, say x is accompanied by some other variable, say y with the same polarity as x in each clause containing x. The variable y may occur in some other clauses, as well. All resolutions with x as the clashing variable are tautologous. Please see the cited paper for further details.

Although it is known that QU-resolution is inferentially complete for closed AT-free PCNF [20], we are not aware of any *implemented* QBF proof system with this property.

For propositional CNF formulas a *model* is a partial assignment that satisfies every clause. For closed PCNF formulas with k universal variables a *model* is a *set* of 2^k prefix-ordered total assignments that comprises a *strategy for the E-player*, such that each assignment in the set has a different assignment to the universal variables and satisfies every clause in the matrix [16, 21, 20]. The term *model tree* is often used to emphasize the structural constraints. (See Def. 2.4 for structural details). For both logics Φ logically implies Ψ if and only if *every* model of Φ is also a model of Ψ. The additional complexity of model trees compared to a single assignment explains why many theorem-proving ideas do not transfer easily from CNF to PCNF.

This paper introduces Subsumption-Linear Q-Resolution (SLQR) for ***proving theorems*** from Quantified Boolean Formulas in PCNF. SLQR is an adaptation of SL-Resolution, which applies to propositional and first-order logic.

A major difference between SLQR and QDPLL (DPLL adapted for QBF) is that QDPLL guesses and backtracks on *variable assignments*, while SLQR guesses and backtracks on *clauses*. The inferential power of SLQR is compared with other Q-Resolution and tableau strategies.

A primary motivation for the SLQR discipline is to reduce the search space compared to ad-hoc heuristics for choosing the next resolution operation. Several optimizations reduce the choices while preserving completeness:

1. One operand of every resolution operation is the immediately preceding derived clause (linearity).
2. When a clashing literal needs to be chosen in the first operand, there are restrictions on which literals need to be considered, and once a literal that meets those restrictions has been chosen, alternative choices of clashing literal need not be considered.
3. When backtrackable choices are made for the second clause operand, logical analysis is used to rule out many unnecessary choices.

SL-Resolution is closely related to model elimination [12, 13, 15, 18, 2] and tableau methods [11]. Discussions and thorough bibliographies may be found in several texts [14, 8]. Terminology varies among these sources.

Letz adapted a tableau-oriented point of view for QBF solving [10]. However, his solver `Semprop` branches on variables, similarly to QDPLL solvers such as `depQBF`, `QuBE`, and others.

After introducing and analyzing needed technical machinery for *prefix-ordered QU-resolution* in Section 2 this paper introduces *Subsumption-Linear Q-Resolution* (SLQR) in Section 4, including the special operation *ancestor resolution*, and proves that SLQR is inferentially complete.

2 Preliminaries

In their most general form, *quantified boolean formulas* (QBFs) generalize propositional formulas by adding universal and existential quantification of boolean variables (often abbreviated to "variables"). A *quantified variable* is denoted by $\forall\, u$ (variable u is universal) or $\exists\, e$ (variable e is existential). A *literal* is a variable or a negated variable. See [8] for a thorough introduction and a review of any unfamiliar terminology.

Definition 2.1 The ***truth value*** of a *closed* QBF is either 0 (*false*) or 1 (*true*), as defined by induction on its principal operator.

1. $(\exists\, e\, \Psi(e)) = 1$ iff $(\Psi(0) = 1$ or $\Psi(1) = 1)$.
2. $(\forall\, u\, \Psi(u)) = 0$ iff $(\Psi(0) = 0$ or $\Psi(1) = 0)$.
3. Other operators have the same semantics as in propositional logic.

This definition emphasizes the connection of QBF to two-person games, in which player E (Existential) tries to set existential variables to make the QBF evaluate to 1, and player A (Universal) tries to set universal variables to make the QBF evaluate to 0 (see [9] for more details). $\square$

Definition 2.2 For this paper QBFs are in ***prenex conjunctive normal form* (PCNF)**, and are *closed*; i.e., $\Psi = \overrightarrow{Q}\,.\,\mathcal{F}$ consists of a quantifier prefix $\overrightarrow{Q}$ and a set of quantifier-free clauses $\mathcal{F}$ (often called the *matrix*) such that every variable in $\mathcal{F}$ occurs in $\overrightarrow{Q}$. The number of clauses in $\mathcal{F}$ is denoted by $|\mathcal{F}|$. In the context of a matrix, clauses are understood to be combined conjunctively.

A ***clause*** is a disjunctively connected set of literals. A clause is called *tautologous* if it contains some literal and its complement; otherwise it is called *non-tautologous.* Clauses are frequently written enclosed in square brackets (e.g., $[p, q, \overline{r}]$) and [] denotes the empty clause.

We follow certain notational conventions for boolean variables and literals (signed variables) to make reading easier: Lowercase letters near the ***beginning*** of the alphabet (e.g., b, c, d, e) denote existential literals, while lowercase letters near the ***end*** of the alphabet (e.g., u, v, w, x) denote universal literals, while ***middle*** letters (e.g., p, q, r) are of unspecified quantifier type. Quantifier types are implied frequently throughout the paper without restating this convention.

In contexts where a literal is expected, p might denote a positive or negative literal, while $\overline{p}$ denotes the negation of p. To emphasize that p stands for a ***variable***, rather than a *literal*, the notation $|p|$ is used. Clauses may be written as $[p, q, \overline{r}]$); [] denotes the empty clause.

For set-combining operations on clauses, besides $\cup$ for union and $\cap$ for intersection, we use $+$ for ***disjoint union***, $-$ for ***set difference***, and write p instead of $[p]$ when it is an operand for one of these operations. Thus $C + p$ adds p to a clause that does not already contain p, while $C - p$ removes p from a clause that might or might not contain p.

The symbols α, β, and γ denote (possibly empty) sequences of literals or sets of literals, depending on context; $vars(\alpha)$ denotes the set of variables underlying the literals of α. (Because a clause is a set, a notation like $[p, \alpha]$ implicitly specifies that p is not in α.) The symbol $\bot$ is sometimes used as a literal denoting *false* and is treated as being outer to all other literals. $\square$

Definition 2.3 The ***quantifier prefix*** (often shortened to ***prenex***) is a sequence of quantified variables. A variable closer to the beginning (end) of the sequence is said to be ***outer*** (***inner***) to another variable. The prenex is partitioned into ***quantifier blocks*** (abbreviated to ***qblocks***). Each quantifier block is a maximal consecutive subsequence of the prenex with variables with the same quantifier type, and has a unique ***quantifier depth***, denoted as *qdepth*, with the outermost qblock having $qdepth = 1$. The notation $p \prec q$ means that p is in a qblock outer to the qblock of q. The notation $p \preceq q$ means that p is the same qblock as q or $p \prec q$. There is no special notation for p and q being in the same qblock. The notation is extended to sets of variables or literals in the obvious ways; e.g., $P \prec q$ means that each $p \in P$ satisfies $p \prec q$. In situations where variables within a quantifier block are considered to have a fixed order, $p \prec\!\prec q$ means: p precedes q in the same quantifier block or $p \prec q$.

A few special operations on sets of literals are defined. A prenex $\overrightarrow{Q}$ is assumed to be known by the context. For a set S of literals:

$$\mathsf{exist}(S) = \{\text{the existential literals in } S\} \tag{1}$$
$$\mathsf{univ}(S) = \{\text{the universal literals in } S\} \tag{2}$$
$$(S \prec q) = \{\text{the literals in } S \text{ outer to } q\} \tag{3}$$
$$(q \prec S) = \{\text{the literals in } S \text{ inner to } q\} \tag{4}$$

Depending on context, the set of literals might be a clause, a prenex, a partial assignment, or other logical expression. □

Definition 2.4 Let a closed PCNF $\Psi = \overrightarrow{Q}.\mathcal{F}$ be given. Let $\mathbf{V}$ denote the variables of Ψ. A ***QBF strategy*** for Ψ is a set of boolean functions $\{p_j(\beta_j)\}$, where p_j ranges over the variables of one quantifier type and β_j consists of all variables q of the *opposite quantifier type* such that $q \prec p_j$. The function $p_j(\beta_j)$ is called a *Skolem function* if p_j is existential and is called an *Herbrand function* if p_j is universal. For Skolem functions $\beta_j = \mathsf{univ}(\mathbf{V}) \prec p_j$; for Herbrand functions $\beta_j = \mathsf{exist}(\mathbf{V}) \prec p_j$.

A ***winning strategy*** for player E is a QBF strategy in which p_j ranges over the *existential* variables such that $\mathcal{F}$ always evaluates to 1 if player E always chooses $p_j = p_j(\beta_j)$ when p_j is the outermost unassigned variable in the two-person game mentioned in Def. 2.1. A ***winning strategy*** for player A is a QBF strategy in which p_j ranges over the *universal* variables such that $\mathcal{F}$ always evaluates to 0 if player A always chooses $p_j = p_j(\beta_j)$ when p_j is the outermost unassigned variable in the same game. Exactly one of the players has a winning strategy. Winning strategies can be generalized to closed QBFs that are not in prenex conjunctive normal form, whose variables may have only a *partial* order [9].

A clause D is said to be ***logically implied*** by Ψ if $\overrightarrow{Q}.(\mathcal{F} \cup \{D\})$ has the same set of winning strategies for player E as does Ψ. The term ***logical consequence*** is also used. In this case, D is said to be a ***strategy-sound inference*** from Ψ, following [21]. As a less stringent requirement, a clause D is said to be a ***safe inference*** from Ψ if $\overrightarrow{Q}.(\mathcal{F} \cup \{D\})$ has the same truth value as Ψ (i.e., adding D does not change the set of winning strategies for player E from nonempty to empty).

Dually, deletion of a clause D from Ψ is said to be a ***strategy-sound operation*** if $\overrightarrow{Q}.(\mathcal{F} - \{D\})$ has the same set of winning strategies for player E as does Ψ. A clause deletion is said to be a ***safe operation*** if $\overrightarrow{Q}.(\mathcal{F} - \{D\})$ has the same truth value as Ψ (i.e., deleting D does not change the set of winning strategies for player E from empty to nonempty). □

Definition 2.5 The proof system known as *Q-resolution* consists of two operations, *resolution* and *universal reduction.* Resolution is defined as usual, except that the clashing literal is always existential; resolvents must be non-tautologous for Q-resolution. Universal reduction is special to QBF.

$$\mathsf{res}_e(C_1, C_2) = \alpha \cup \beta \qquad \text{where } C_1 = [\overline{e}, \alpha]\,,\ C_2 = [e,\ \beta] \tag{5}$$
$$\mathsf{unrd}_u(C_3) = \gamma \qquad \text{where } C_3 = [\gamma,\ u]\,. \tag{6}$$

$\mathsf{unrd}_u(C_3)$ is defined only if u is ***tailing*** for γ, which means that the *qdepth* of u is greater than that of any existential literal in γ, i.e., $(u \prec \mathsf{exist}(\gamma)) = \emptyset$.

A clause is *fully reduced* if no universal reductions on it are possible. The fully reduced form of C is denoted as $\mathsf{unrd}_*(C)$. For this paper all clauses in given PCNFs are assumed to be fully reduced and non-tautological, unless stated otherwise. □

Q-resolution is of central importance for PCNFs because it is a strategy-sound and refutationally complete proof system [7, 8], as restated in Theorem 2.6 below. Recall that a clause-based proof system is *refutationally complete* if the empty clause can be derived from every formula whose truth value is 0.

Theorem 2.6 [7] Let the closed PCNF $\Psi = \overrightarrow{Q}\,.\,\mathcal{F}$ be given. Ψ evaluates to *false* if and only if [] can be derived from Ψ by Q-resolution.

We say that a proof system is *inferentially complete* if whenever D is logically implied (see Def. 2.4), then some subset of D can be derived in the proof system. Note that Q-resolution is not inferentially complete. A simple example is

$$\forall u\, \exists e\, \exists f.\, \{[u,\ e]\,,\ [\overline{u}\,,\ f]\}\,.$$

Nothing can be derived by Q-resolution, but the clause $[e,\ f]$ is logically implied, which can be seen by enumerating all the winning strategies $\{e(u),\ f(u)\}$ and observing that $[e,\ f]$ evaluates to 1 for all values of u in each strategy.

The proof system known as QU-resolution is Q-resolution with the added operation of resolution on universal variables. QU-resolution is inferentially complete for closed PCNF and is able to provide exponentially shorter refutations for certain QBF families [20]. However, the challenge for using QU-resolution in practice is knowing when universal resolution is likely to be productive.

Definition 2.7 A ***QU-derivation*** or ***Q-derivation***, often denoted as Π or Γ or Σ, is a rooted directed acyclic graph (DAG) in which each vertex is either an original clause (a DAG leaf), or a proof operation (an internal vertex). A *Q(U)-refutation* is a Q(U)-derivation of the empty clause. This paper follows the convention that DAG edges are directed *away from* the root. A Q(U)-derivation is *tree-like* if every internal vertex has only one incoming edge, except that the root has no incoming edge.

A *subderivation* of a Q(U)-derivation Π is any rooted sub-DAG of Π whose vertices consist of some root vertex V and all DAG vertices of Π reachable from V and whose edges are the induced edges for this vertex set.

In a proof DAG, each internal vertex is represented as a tuple with fields consisting of:

- a specified operation type (resolution or universal reduction or "copy"),
- a specified clashing literal or universal-reduction literal (null for "copy"),

– one or two directed edge(s) to its operand(s),
– a derived clause.

(See Fig. 1.) The same tuple may be used to represent a leaf, in which case the operation type is "leaf", the clashing literal is null, there are no outgoing edges, and the clause is an original clause. When there is no confusion, a vertex may be referred to by its clause; however, the same clause may appear in more than one vertex.

The "copy" just transfers the same clause to another vertex, and is included for technical reasons. A DAG containing copy operations (and correctly derived clauses) is called a ***generalized derivation***. The copy operations can be "spliced out" in the obvious manner to produce a derivation: If V contains a copy operation, replace all incoming edges to V by edges to the child of V. See [19] for details on propositional derivations. The QBF variant is developed in [5].

In the normal case of a resolution operation, the first, or left, edge goes to a vertex whose clause contains the *negation* of the clashing literal, and the second, or right, edge goes to a vertex whose clause contains the clashing literal. In any case, the union of the two operand clauses may not contain any complementary pair of literals other than the clashing literals.

We say that a literal q *has a proof operation* at the (internal) DAG vertex V if q or $\overline{q}$ is the literal specified in V; we say that a literal q *has a proof operation* in Π if q has a proof operation at some DAG vertex in Π.

For a proof DAG Π, $\mathsf{root}(\Pi)$ is the clause at the root, $\mathsf{leaves}(\Pi)$ is the *set* of clauses in the leaves, and

$$\mathsf{support}(\Pi) = \overrightarrow{Q}'.\,\mathsf{leaves}(\Pi), \tag{7}$$

where $\overrightarrow{Q}'$ is the subsequence of $\overrightarrow{Q}$ that contains only the variables that appear in $\mathsf{leaves}(\Pi)$. □

Definition 2.8 An ***assignment*** is a partial function from variables to truth values, and is usually represented as the set of literals that it maps to *true*. Assignments are denoted by ρ, σ, τ, etc. A *total assignment* assigns a truth value to every variable.

Application of an assignment σ to a logical expression, followed by truth-value simplifications,[1] is called a ***restriction***. Restrictions are denoted by $q\lceil_\sigma$, $C\lceil_\sigma$, $\mathcal{F}\lceil_\sigma$, etc. If σ assigns variables that are quantified in Ψ, those quantifiers are deleted in $\Psi\lceil_\sigma$, and their variables receive the assignment specified by σ. □

3 Prefix-Ordered QU-Resolution

This section examines the restriction on QU-resolution derivations to be prefix-ordered, as defined below. The main result of this section is Lemma 3.6, which

[1] I.e., simplifications where one operand is 0 or 1.

concludes that prefix-ordered QU-resolution is inferentially complete. This is a stepping stone to the main results of the paper about SLQ resolution in Section 4.

In analogy with *regular propositional resolution* as defined by Kleine Büning and Lettmann [8], who cite Tseitin's classical paper, we define regularity for QU-resolution derivations. Definition Def. 3.1 is more precise than one that is often seen, which specifies that no variable has more than one proof operation on any path in Γ. The two definitions are equivalent for refutations, but not for derivations in general.

For example, the four propositional clauses $[b, \neg e]$ $[e, \neg c]$ $[c, \neg d]$ $[b, e]$ derive $[d, e]$, but the derivation should not be called regular because a proof operation on e is needed..

Definition 3.1 A QU-resolution derivation Γ is said to be ***regular in*** p if no derived clause D that contains $|p|$ has a proof operation on $|p|$ on some path in Γ from D to a leaf. A QU-resolution derivation Γ is said to be ***regular*** if it is regular in p for all variables $|p|$ that have proof operations in Γ. □

Definition 3.2 We define QU-resolution to be ***prefix-ordered*** if the literals that have proof operations appear in the quantifier-prefix order on every path in the proof DAG, with the outermost closest to the root. □

A prefix-ordered QU-refutation is necessarily regular, but other prefix-ordered QU-derivations are not necessarily regular. The ensuing material requires some technical terminology, defined next.

Definition 3.3 A clause C ***subsumes*** clause D if the literals of C comprise a subset of the literals of D or if D is tautologous. Subsumption is *proper* if the subset is proper. In this sense, any tautologous clause is treated as containing every possible literal and is properly subsumed by any non-tautologous clause.

Minimality of clauses and sets of clauses is important in the technical material. A set of clauses is ***minimal*** under stated conditions if no proper subset of its clauses satisfies all of the conditions. Minimality of the set does not require minimum cardinality.

A clause C is ***QU-minimal*** for a PCNF Ψ if it is derivable from Ψ by QU-resolution and no proper subset of $\mathsf{unrd}_*(C)$ is derivable from Ψ by QU-resolution. A clause C is ***Q-minimal*** for a PCNF Ψ if it is derivable from Ψ by Q-resolution and no proper subset of $\mathsf{unrd}_*(C)$ is derivable from Ψ by Q-resolution.

Q-minimality of C does not require minimum cardinality; that is, some other clause E such that $|E| < |C|$ may be derivable by Q-resolution, provided that E does not properly subsume $\mathsf{unrd}_*(C)$. The same holds for QU-minimality. □

Definition 3.4 A QU-derivation Π is ***QU-irreducible*** if:

1. The clause derived in $\mathsf{root}(\Pi)$, say D, is QU-minimal for $\mathsf{support}(\Pi)$,
2. $\mathsf{leaves}(\Pi)$ is minimal for the QU-derivation of D from $\mathsf{support}(\Pi)$,
3. Π contains no proof operations on variables in D,
4. all proper subderivations of Π are QU-irreducible.

Ψ	$\exists d$	$\exists e$	$\forall u$	$\exists f$
C_1		e	u	$\overline{f}$
C_2	d			f
C_3	$\overline{d}$	$\overline{e}$		f

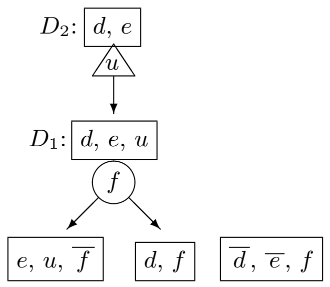

Fig. 1. PCNF Ψ in chart form (left) and proof DAG (right) for Example 3.5. Circles enclose the clashing literal for resolution; triangles denote universal reduction. C_3 is not part of the DAG rooted at D_2, but is its own trivial DAG.

Note that this definition does not require that the set of *DAG vertices* is minimal. In particular, every QU-irreducible derivation has a tree-like version.

Q-irreducible derivations are defined analogously. □

Example 3.5 To illustrate Q-minimality and QU-minimality, consider the formula Ψ, shown in Fig. 1 as a clause-literal incidence graph (chart form for short). No universal resolutions are possible so Q and QU properties are the same. Let:

$$\begin{array}{r|l} D_1 = \mathsf{res}_f(C_1,\, C_2) = [d,\, e,\, u] & \Pi_1 = \text{the subderivation whose root is } D_1 \\ D_2 = \mathsf{unrd}_u(D_1) = [d,\, e] \quad & \Pi_2 = \text{the derivation of } D_2 \\ & \Pi_3 = \text{the zero-step derivation of } C_3. \end{array}$$

Then D_1 is Q-minimal for Ψ even though D_2 is a proper subset, because the difference is only tailing universal literals. Also, $[d,\, f]$ is Q-derivable and narrower than D_1, but it is not a subset of D_1.

However, C_3 is not Q-minimal for Ψ even though it is an original clause, because $\mathsf{res}_d(C_3,\, C_2)$ is a proper subset of $\mathsf{unrd}_*(C_3)$. But the trivial subderivation Π_3 *is* Q-irreducible, because $\mathsf{leaves}(\Pi_3) = \{C_3\}$. □

To see that points 1 and 2 of Def. 3.4 are consistent, add a new "indicator" literal a_j to each clause $C_j \in \mathcal{F}$, the matrix. Replace the clause to be derived by $\left[D,\, \bigvee_j a_j\right]$. Then points 1 and 2 are both true if and only if $\left[D,\, \bigvee_j a_j\right]$ is Q-minimal for the modified clauses.

We need the following Lemma 3.6 for analyzing SLQR. QU-minimal clauses and minimal sets of clauses are important in the ensuing material. Recall the terminology in Def. 3.3 and Def. 3.4.

Lemma 3.6 Let the closed PCNF $\Psi = \overrightarrow{Q}.\mathcal{F}$ be given. By convention, every clause in $\mathcal{F}$ is non-tautological and fully reduced. Let clause D be QU-minimal for Ψ. Then D can be derived from Ψ by a QU-derivation Γ such that Γ is prefix-ordered, regular, tree-like and QU-irreducible.

Proof: Let $\mathcal{G} \subseteq \mathcal{F}$ be any subset such that D is not logically implied by any proper subset of $\mathcal{G}$. Let $\Phi = \overrightarrow{Q}.\mathcal{G}$. Then D is also QU-minimal for Φ. The proof of inferential completeness of QU-resolution in [20, Th. 5.4] constructs a QU-derivation of D from Φ with the required properties, and this is also a QU-derivation from Ψ. The cited theorem promises to derive $D^{(-)}$ but by minimality of D it must derive D exactly. ∎

4 Subsumption-Linear Q-Resolution

This section defines subsumption-linear Q-resolution (SLQR) derivations and derives the main results of the paper.. We show that SLQR has the same inferential power as full QU-resolution; i.e., SLQR is inferentially complete for AT-free PCNF formulas. As mentioned in Section 1, a QBF is said to be *AT-free* if it contains no *asymmetric tautologies* [6]. QBFs translated from applications are normally AT-free, but certain preprocessing operations might introduce asymmetric tautologies.

We also define a weaker variant SLQR–ures that does not include resolution on clashing universal literals, and show that SLQR–ures has the same inferential power as full Q-resolution when all literals in the derived clause are outermost. Hence SLQR–ures is refutationally complete. Lemma 3.6 is an important stepping stone. We also show a PCNF and a Q-derivable clause for which there is no SLQR–ures derivation.

Definition 4.1 Given a QBF $\Phi = \overrightarrow{Q}\,.\,\mathcal{F}$ and a target clause T, a ***subsumption-linear Q-resolution* (SLQR))** derivation of T consists of a *top clause* $D_0 \in \mathcal{F}$ and a sequence of $m \geq 0$ derivation steps with $D_m = T$ of the form

$$D_i = \begin{cases} \mathsf{res}_{p(i)}(D_{i-1},\, C_i) & \text{where } p(i) \text{ is any literal and } 1 \leq i \leq m \\ \mathsf{unrd}_{u(i)}(D_{i-1}) & \text{where } u(i) \text{ is universal and } 2 \leq i \leq m \end{cases} \tag{8}$$

such that each C_i is either a clause in $\mathcal{F}$ or is an earlier derived clause D_j that meets the precise criteria given below and is called an *ancestor clause.*

The D_i are called ***center clauses***. The C_i are called ***side clauses***. The literals of a side clause C_i are categorized as follows: $p(i)$ is the ***clashing*** literal; if C_i is derived, $p(i)$ is also called an ***ancestor*** literal; $q \in C_i$ is a ***target*** literal if $q \in T$; $q \in C_i$ is a ***merge*** literal if $q \in D_{i-1}$ and q is not a target literal; $q \in C_i$ is an ***extension*** literal if $q \in D_i$ and q is not in any of the preceding categories.

At the step where D_i is to be derived let D_j $(j \leq i-2)$ be an earlier derived clause and let $q \in D_j$ be the clashing literal for the derivation of D_{j+1}. Then D_j is defined to be an ***ancestor clause*** at this step in the proof if $D_j - \{q\}$ is a proper subset of each subsequently derived clause $D_{j+1}, \ldots, D_{i-1}$. If $q = \overline{p(i)}$ (the clashing literal in D_{i-1}), then the resolution of D_{i-1} and D_j is called ***ancestor resolution***, $\underline{p(i)}$ is called the ***ancestor literal***, and D_i consists of all literals in D_{i-1} except $\overline{p(i)}$. The word "subsumption" in the name "SLQR" is explained by the last relationship. If ancestor resolution is possible, other choices for side clause can be disregarded.

If D_j is an ancestor clause but $q \neq \overline{p(i)}$, q still plays a role as an ancestor literal: Some original clause must be chosen to resolve with D_{i-1}. If any *extension* *literal* of this resolution would be q, then this clause is inadmissible as a side clause at this step. A derivation that adheres to this policy (and also disallows derivation of tautologous clauses) is called ***tight*** [14]. $\square$

Considering SLQR as a proof search system, the procedure to extend D_{i-1} to D_i consists of selecting a literal in D_{i-1} for the proof operation, and if the operation is resolution, selecting a side clause. It is known from antiquity [1] that propositional SL-resolution is complete for any literal-selection policy; i.e., it is not necessary to backtrack on the selected literal and try other selections. For simplicity and attention to implementation concerns, we consider only the LIFO policy for SLQR, defined next.

Definition 4.2 Given a QBF $\Phi = \overrightarrow{Q}.\mathcal{F}$ and a target clause T, the ***LIFO selection policy***, also called the *most recently introduced policy* is defined informally as follows. In a SLQR derivation, assume that each center clause D_{i-1} is represented by a last-in, first-out stack (LIFO) of its literals that are not in T, called the ***L-stack***, as well as a separate set of literals that are in T, which we call the ***T-subset***.

The L-stack is partitioned into contiguous sections such that all literals in a given section were introduced into a center clause D_j, $j \leq i-1$, as extension literals in the earlier resolution operation that derived D_j, and these literals are in quantifier order within the section with the innermost closest to the top of the L-stack. Further, this section has been intact for all center clauses between D_j and D_{i-1}. The L-stack as a whole may not be quantifier ordered. The *LIFO selection policy* selects the literal on top of the L-stack of the current center clause, say D_{i-1}. $\square$

Whatever proof operation derives D_i, the selected literal will not be in D_i, so the L-stack of D_i may be formed by starting with that of D_{i-1}, popping the selected literal, and then possibly pushing a new section on top consisting of extension literals. A SLQR derivation develops by working on the section on top of the current L-stack until the current L-stack is empty. Readers familiar with Prolog will recognize the similarity to how the Prolog interpreter works.

4.1 Derivation Power of SLQR

This section investigates when a QU-derivable clause T also has a SLQR derivation. For propositional resolution, it is well known that the answer is essentially "always".[2] The situation for closed PCNF is not so simple.

The proof of the next theorem employs the framework first published by Anderson and Bledsoe [1]. Minimal clauses and minimal sets of clauses are important in the ensuing material. Recall the terminology in Def. 3.3.

Theorem 4.3 Given a closed PCNF $\Psi = \overrightarrow{Q}.\mathcal{F}$, let T be a minimal clause such that there is a QU-resolution derivation of T from Ψ, call it Π, and no proper subset of $\mathcal{F}$ permits derivation of T. Then for every clause $C_0 \in \mathcal{F}$ and for the

[2] If the derived clause is not minimal, propositional resolution may derive a subsuming clause.

LIFO selection function (see Def. 4.2) there exists a SLQR derivation of T from Ψ whose top clause is C_0. Further, for each literal $q \in T$, q has no proof operation in the SLQR derivation.

Proof: Please see https://users.soe.ucsc.edu/ avg/Papers/slqr-long.pdf. ■

4.2 Derivation Power of SLQR–ures

SLQR–ures is the same as SLQR except that resolution on clashing universal literals is not permitted. This section investigates when a Q-derivable clause T also has a SLQR–ures derivation. For propositional resolution, it is well known that the answer is essentially "always," and this is just a special case of Theorem 4.4 below.[3] The situation for closed PCNF is not so simple.

The proof of the next theorem employs the framework first published by Anderson and Bledsoe [1]. Minimal clauses and minimal sets of clauses are important in the ensuing material. Recall the terminology in Def. 3.3.

Theorem 4.4 Given a closed PCNF $\Psi = \overrightarrow{Q}\,.\,\mathcal{F}$, let T be a minimal clause such that there is a Q-resolution derivation of T from Ψ, call it Π, and no proper subset of $\mathcal{F}$ permits derivation of T by Q-resolution. Further, let the literals of T be outermost among the literals of $\mathcal{F}$. Then for every clause $C_0 \in \mathcal{F}$ there exists a SLQR–ures derivation of T from Ψ whose top clause is C_0. Further, for each literal $q \in T$, q has no proof operation in the SLQR–ures derivation. In particular, SLQR–ures is refutationally complete for closed PCNF.

Proof: The proof is similar to that of Theorem 4.3 and is omitted. The hypothesis that T is outer to all literals with proof operations ensures that whenever a universal literal is the selected literal universal reduction is available, so universal resolution is not needed. Refutational completeness follows by letting $T = []$. ■

The preceding Theorem 4.4 shows that SLQR–ures has the full inferential power of Q-resolution for a very restricted set of derived clauses.

In fact, there are important clauses that can be derived by prefix-ordered tree-like Q-resolution, but not by SLQR–ures.

Theorem 4.5 There exists a closed PCNF such that the clause $[u, h]$ is derivable by prefix-ordered tree-like Q-resolution and not by SLQR–ures, u is universal and outermost, h is existential and innermost, $[u, h]$ is minimal, and the matrix is minimal.

Proof: Please see https://users.soe.ucsc.edu/ avg/Papers/slqr-long.pdf. ■

4.3 Details for LIFO SLQR

Definition 4.6 The details of updating the stack are important, and some helpful terminology is now introduced. Proof operations are classified as follows:

1. *Reduction operation*: a universal reduction on a universal literal;

[3] If the derived clause is not minimal, propositional resolution may derive a subsuming clause.

2. *Extension operation*: a resolution that introduces at least one literal not in the U-set or in the E-stack of the current center clause;
3. *Contraction operation*: a resolution that introduces no literals into the U-set or the E-stack, but possibly adds some literals to the T-subset.

For a resolution operation, the literals in the side clause are classified as follows:

1. *Clashing literal*: does not appear in the resolvent; pop its complement from the top of the E-stack;
2. *Target literal*: any literal in T; union this with the T-subset;
3. *Universal literal*: any universal literal not in T; union this with the U-set;
4. *Merge literal*: already in the E-stack; do not push this on the E-stack;
5. *Extension literal*: none of the above; all extension literals are pushed on the E-stack in outer to inner prefix order; the innermost extension literal is on top of the new E-stack.

Extension and merge literals are existential and the terminology stems from model elimination.

To get the center-clause data structure started, define the initial center clause to be $\top$, a tautologous clause that contains all literals. We use the sound extension that $\mathsf{res}_e(\top, C) = C$ for all non-tautologous C that do *not* contain the existential variable $|e|$. If the desired top clause is C_0, the literal selection rule simply chooses some literal whose variable is not among $vars(C_0)$. Then C_0 becomes the side clause for step 0. This artificial protocol makes all original clauses in the derivation appear as side clauses and simplifies later descriptions. The literals of the C_0 are processed as described in Def. 4.6.

The foregoing description can be formalized in mathematical terms of sets and sequences. We only note that the center clauses, disregarding the T-subset and U-set, can be regarded as existential literal sequences that can be partitioned into contiguous subsequences such that each subsequence is in prefix order and contains some subset of the extension literals of a single extension operation.

Definition 4.7 A *LIFO SLQR* is an SLQR that uses the LIFO selection function and also has an admissibility requirement for side clauses used for an extension operation.

At step i (to derive D_i) suppose the selected literal is $\overline{p}$. A side clause C (which necessarily contains p) is ***inadmissible*** if for some $j < i - 1$, D_j subsumes $\mathsf{res}_p(D_{i-1}, C)$ (including the case that the resolvent is tautologous). A derivation attempt fails if the current center clause was formed by resolution with an inadmissible side clause. In this case the LIFO-selected literal is $\perp$. $\square$

Example 4.8 The motivation for inadmissible clauses is that it prevents looping [14]. Suppose the current center clause is $D_i = (\{\}, \{\beta\}, [\alpha, \overline{f}])$, where the T-subset is empty, the U-set is β, and the E-stack is $[\alpha, \overline{f}]$. Thus $\overline{f}$ is selected. Suppose there are clauses $C_1 = [f, \overline{g}]$ and $C_2 = [g, \overline{f}]$. Resolving (extending) D_i with C_1 gives $D_{i+1} = (\{\}, \{\beta\}, [\alpha, \overline{g}])$, then extending with C_2 would give $D_{i+2} = (\{\}, \{\beta\}, [\alpha, \overline{f}])$, creating a cycle. So C_2 is inadmissible to resolve with D_{i+1}. If no other side clause containing g is admissible, then the LIFO SLQR selected literal at step $i + 2$ is $\bot$, forcing the derivation attempt to fail. Thus a successfully completed LIFO SLQR never contains an inadmissible side clause. $\square$

Corollary 4.9 Given a QBF $\Psi = \overrightarrow{Q}.\mathcal{F}$, let T be a minimal clause such that there is a Q-resolution derivation of T from Ψ, call it Π, and no proper subset of $\mathcal{F}$ permits derivation of T. Further, let the literals of T be outermost among the literals of $\mathcal{F}$. Then for every clause $C_0 \in \mathcal{F}$ there exists a LIFO SLQR derivation of T from Ψ whose top clause is C_0. Further, for each literal $q \in T$, q has no proof operation in the LIFO SLQR derivation.

Proof: Please see https://users.soe.ucsc.edu/ avg/Papers/slqr-long.pdf. $\blacksquare$

5 Conclusion

Subsumption-Linear Q-Resolution (SLQR) was introduced for proving theorems from Quantified Boolean Formulas. It is an adaptation of SL-Resolution, which in turn is closely related to model elimination and tableau methods. A major difference from QDPLL (DPLL adapted for QBF) is that QDPLL guesses variable assignments, while SLQR guesses clauses. Inferential completeness of SLQR for AT-free PCNFs is shown when it is allowed to use resolution with universal clashing variables; without that operation it is refutationally complete.

Future work should study heuristics for clause selection and lemma retention.

5.1 Acknowledgment

We thank the reviewers for their careful reading and suggestions for clarifying the paper.

References

1. Anderson, R., Bledsoe, W.W.: A linear format for resolution with merging and a new technique for establishing completeness. Journal of the ACM 17(3), 525–534 (1970)
2. Astrachan, O.L., Loveland, D.W.: The use of lemmas in the model elimination procedure. Journal of Automated Reasoning 19, 117–141 (1997)
3. Beyersdorff, O., Chew, L. Janota, M.: New resolution-based QBF calculi and their proof complexity. ACM Transactions on Computation Theory 11, 1–42 (2019)
4. Burris, S.: Logic for Mathematics and Computer Science. Prentice Hall (1998)
5. Goultiaeva, A., Van Gelder, A., Bacchus, F.: A uniform approach for generating proofs and strategies for both true and false QBF formulas. In: Proc. IJCAI (2011)
6. Heule, M., Seidl, M., Biere, A.: Efficient Extraction of Skolem Functions from QRAT Proofs. In: Proc. FMCAD (2014)
7. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified boolean formulas. Information and Computation 117, 12–18 (1995)
8. Kleine Büning, H., Lettmann, T.: Propositional Logic: Deduction and Algorithms. Cambridge University Press (1999)
9. Klieber, W., Sapra, S., Gao, S., Clarke, E.: A non-prenex, non-clausal QBF solver with game-state learning. In: SAT, LNCS (2010)
10. Letz, R.: Lemma and model caching in decision procedures for quantified boolean formulas. In: Proc. TABLEAUX (LNAI 2381). pp. 160–175 (2002)
11. Letz, R., Mayr, K., Goller, C.: Controlled integration of the cut rule into connection tableau calculi. JAR 13, 297–337 (1994)
12. Loveland, D.W.: Mechanical theorem-proving by model elimination. Journal of the ACM 15(2), 236–251 (1968)
13. Loveland, D.W.: A simplified format for the model elimination theorem-proving procedure. JACM 16(3), 349–363 (1969)
14. Loveland, D.W.: Automated Theorem Proving: A Logical Basis. North-Holland, Amsterdam (1978)
15. Minker, J., Zanon, G.: An Extension to Linear Resolution with Selection Function. Information Processing Letters 14(4), 191–194 (1982)
16. Samulowitz, H., Davies, J., Bacchus, F.: Preprocessing QBF. In: Proc. CP 2006 (LNCS 4204). pp. 514–529 (2006)
17. Slivovsky, F.: Quantified CDCL with Universal Resolution. In: Proc. Sat 2022. (2022)
18. Van Gelder, A.: Autarky pruning in propositional model elimination reduces failure redundancy. Journal of Automated Reasoning 23(2), 137–193 (1999)
19. Van Gelder, A.: Input distance and lower bounds for propositional resolution proof length. In: Theory and Applications of Satisfiability Testing (SAT) (2005)
20. Van Gelder, A.: Contributions to the theory of practical quantified boolean formula solving. In: Proc. CP. pp. 647–673 (2012)
21. Van Gelder, A., Wood, S.B., Lonsing, F.: Extended failed literal detection for QBF. In: Proc. SAT. pp. 86–99 (2012)