Davis and Putnam Meet Henkin: Solving DQBF with Resolution

Joshua Blinkhorn[1], Tomáš Peitl[1(✉)], and Friedrich Slivovsky[2]

[1] Friedrich-Schiller-Universität Jena, Jena, Germany
{joshua.blinkhorn,tomas.peitl}@uni-jena.de
[2] TU Wien, Vienna, Austria
fslivovsky@ac.tuwien.ac.at

Abstract. Davis-Putnam resolution is one of the fundamental theoretical decision procedures for both propositional logic and quantified Boolean formulas.

Dependency quantified Boolean formulas (*DQBF*) are a generalisation of QBF in which dependencies of variables are listed explicitly rather than being implicit in the order of quantifiers. Since DQBFs can succinctly encode synthesis problems that ask for Boolean functions matching a given specification, efficient DQBF solvers have a wide range of potential applications. We present a new decision procedure for DQBF in the style of Davis-Putnam resolution. Based on the merge resolution proof system, it directly constructs partial strategy functions for derived clauses. The procedure requires DQBF in a normal form called H-Form. We prove that the problem of evaluating DQBF in H-Form is NEXP-complete. In fact, we show that any DQBF can be converted into H-Form in linear time.

1 Introduction

Continuing advances in the performance of propositional satisfiability (SAT) solvers are enabling a growing number of applications in the area of electronic design automation [28], such as model checking [6], synthesis [24], and symbolic execution [3]. In artificial intelligence, SAT solvers are a driving force behind recent progress in constrained sampling and counting [19], and they act as combinatorial search engines in competitive planning tools [10]. In most of these cases, SAT solvers deal with problems from complexity classes beyond NP and propositional encodings that grow super-polynomially in the size of the original instances. Clever techniques such as incremental solving can partly alleviate this issue, but ultimately the underlying asymptotics lead to formulas that are too large to be solved by even the most efficient SAT solvers.

This research was supported by the Vienna Science and Technology Fund (WWTF) under grant number ICT19-060, and by the Austrian Science Fund (FWF) under grant number J-4361N.

C.-M. Li and F. Manyà (Eds.): SAT 2021, LNCS 12831, pp. 30–46, 2021.
https://doi.org/10.1007/978-3-030-80223-3_4

This has prompted the development of decision procedures for more succinct generalizations of propositional logic such as Quantified Boolean Formulas (QBFs). Deciding satisfiability of QBFs is PSPACE-complete [25] and thus believed to be much harder than SAT, but in practice the trade-off between encoding size and tractability can be in favour of QBF [13]. Dependency QBF (DQBF) in turn generalise QBF [1,2]. Whereas the nesting of quantifiers implicitly determines the arguments of Skolem (or Herbrand) functions of a QBF, *Henkin quantifiers* explicitly specify the arguments of Skolem (or Herbrand) functions in a DQBF. As a result, DQBFs can succinctly encode problems concerning the existence of Boolean functions subject to a set of constraints. For instance, equivalence checking of partial circuit designs (PEC) can be naturally encoded as DQBF [16].

Existing decision procedures for DQBF either use quantifier expansion to obtain an equivalent propositional formula or QBF, or else adapt search-based algorithms from QBF by introducing additional constraints to make sure the search tree is consistent with the dependency sets of the input DQBF. Semantically, reasoning at the level of functions is more natural, but recent attempts at lifting conflict-driven clause learning (CDCL) to the level of Skolem functions are currently limited to 2QBF [21].

Our main contribution is a new decision algorithm for DQBF that operates directly at the level of functions. Based on the *merge resolution* (M-Res) proof system [4], it maintains a set of clauses annotated with partial Herbrand functions. Like the original Davis-Putnam procedure [12], it successively eliminates (existentially quantified) variables by creating all possible resolvents at each step. Crucially, resolvents are created only for pairs of clauses with partial Herbrand functions that are consistent and can be combined into a larger partial Herbrand function. Once all variables have been eliminated, either the set of clauses is empty, in which case the input DQBF is true, or it contains the empty clause, in which case the formula is false and the Herbrand functions in the annotation form a countermodel. In contrast to variable elimination by Q-resolution [5,18], where innermost existentially quantified variables must be eliminated first, our algorithm may eliminate variables in any order. While this is not surprising in DQBF, where there is no syntactic ordering of variables, it means that our algorithm can be used to eliminate variables of a QBF in arbitrary order, too—possibly at the cost of increased computational complexity.

There is a surprising obstacle in the way of generalizing variable elimination by resolution to DQBF—it is insufficient to resolve only clauses that contain the current pivot variable being eliminated. In fact, we may need to resolve even pairs of clauses neither of which contains the pivot variable. The requisite combination of weakening and resolution has previously been studied under the name *w-resolution* [8,9]. In turn, w-resolution paves the way for a seemingly absurd case: a clause can now be resolved with itself—*self-resolution*. While self-resolution is not essential, we show that it is a very natural explanation for why we keep certain clauses between individual elimination steps. That understanding casts the algorithm in a different light; as a series of transformations, which result in a normal form where strategies are recorded explicitly.

Strictly speaking, our algorithm (as well as merge resolution) operates on *H-form* DQBF, where Henkin quantifiers specify the arguments of *universal* variables and the matrix is in conjunctive normal form [2]. NEXP-hardness of evaluating DQBF in this form does not immediately follow from known results [1, 23], and determining the complexity of this problem was recently stated as an open question [4]. As a further contribution, we show that it is in fact NEXP-complete, and that DQBF in H-form and the more frequently studied *S-form* (where Henkin quantifiers are used for *existential* variables) are interconvertible at a linear overhead while preserving strategies. Thus our variable elimination algorithm can be used to evaluate and construct (counter)models of arbitrary DQBFs.

The paper is structured as follows: after preliminaries in Sect. 2, we give our decision procedure in Sect. 3, and discuss NEXP-completeness of H-form DQBF in Sect. 4, concluding with a summary in Sect. 5.

2 Preliminaries

H-Form DQBF Intuition. The notion of H-form DQBF is arguably counter-intuitive, and so instead of a formal definition, we start informally. Consider an *S-form DQBF*, i.e. a formula of the form $\forall u_1 \cdots \forall u_m \exists x_1(S_{x_1}) \cdots \exists x_n(S_{x_n}) \cdot \phi$, where each existential variable x_i has a dependency set $S_{x_i} \subseteq \{u_1, \ldots, u_m\}$, and ϕ is a DNF. The goal with such a formula is to find a set of functions—called a *model*—for the existential variables respecting the dependencies so that after substitution into ϕ, the formula becomes a tautology in the universal variables. An example of such a formula is

$$\Psi = \forall u_1\ \forall u_2\ \exists x_1(u_1)\ \exists x_2(u_2) \quad (\overline{u_1} \wedge x_2) \vee (u_1 \wedge \overline{x_2}) \vee (\overline{u_2} \wedge x_1) \vee (u_2 \wedge \overline{x_1})$$

along with the model $x_1 = u_1, x_2 = \overline{u_2}$—whose substitution into Ψ indeed produces a tautology. An H-form DQBF with a CNF matrix is then simply a negation of an S-form DQBF with a DNF matrix, where strategies are sought for universal variables and the goal is to make the substituted formula unsatisfiable, rather than valid.

H-form DQBF Syntax. A *variable* is an element z of the countable set $\mathbb{V}$. A *literal* is a variable z or its *negation* $\overline{z}$. The negation of a literal a is denoted $\overline{a}$, where $\overline{\overline{z}} := z$ for any variable z. A *clause* is a disjunction of literals. A *conjunctive normal form formula* (CNF) is a conjunction of clauses. The set of variables appearing in a formula ψ is denoted $\text{vars}(\psi)$. For ease, we often write clauses as sets of literals, and CNFs as sets of clauses.

An *H-form dependency quantified Boolean formula* (DQBF) is a sentence of the form $\Psi := \exists x_1 \cdots \exists x_n \forall u_1(H_{u_1}) \cdots \forall u_m(H_{u_m}) \cdot \psi$, where the part that holds quantification information is called the *prefix*, and the *matrix* ψ is a CNF. In the quantifier prefix, each universal variable u_i is associated with a *dependency set* H_{u_i}, which is a subset of the existential variables $\{x_1, \ldots, x_n\}$. With $\text{vars}_\exists(\Psi)$

and $\mathrm{vars}_\forall(\Psi)$ we denote the existential and universal variable sets of Ψ, and with $\mathrm{vars}(\Psi)$ their union. We deal only with DQBFs for which $\mathrm{vars}(\psi) \subseteq \mathrm{vars}(\Psi)$.

H-form DQBF Semantics. An *assignment* α to a set Z of Boolean variables is a function from Z into the set $\{0, 1, *\}$. An assignment whose range is $\{0, 1\}$ is called *total*. The set of all assignments to Z is denoted $\langle\!\langle Z \rangle\!\rangle$, and the set of all total assignments is denoted $\langle Z \rangle$. The *domain restriction* of α to a subset Z' of its domain is written $\alpha{\restriction_{Z'}}$. We say that α *extends* α', denoted by $\alpha' \subseteq \alpha$, when $\alpha(z) = \alpha'(z)$ for each $z \in \mathrm{dom}(\alpha')$ with $\alpha'(z) \in \{0, 1\}$.

The *restriction* of a formula ψ by an assignment α, denoted $\psi[\alpha]$, is the result of substituting each variable z in the preimage $\alpha^{-1}(\{0, 1\})$ by $\alpha(z)$, followed by applying the standard simplifications for Boolean constants, i.e. $\overline{0} \mapsto 1$, $\overline{1} \mapsto 0$, $\phi \vee 0 \mapsto \phi$, $\phi \vee 1 \mapsto 1$, $\phi \wedge 1 \mapsto \phi$, and $\phi \wedge 0 \mapsto 0$. We say that α *satisfies* ψ when $\psi[\alpha] = 1$, and *falsifies* ψ when $\psi[\alpha] = 0$.

For a DQBF $\Psi := \exists x_1 \cdots \exists x_n \forall u_1(H_{u_1}) \cdots \forall u_m(H_{u_m}) \cdot \psi$, any set of functions $h := \{h_u : u \in \mathrm{vars}_\forall(\Psi)\}$ of the form $h_u : \langle H_u \rangle \to \langle\!\langle \{u\} \rangle\!\rangle$ is called a *strategy* for Ψ. For convenience, we use the alias $h(\alpha) := \{h_u(\alpha{\restriction_{H_u}}) : u \in \mathrm{vars}_\forall(\Psi)\}$. A strategy for Ψ is called *winning* when each combined assignment $\alpha \cup h(\alpha)$ falsifies ψ. The terms 'winning strategy' and '*countermodel*' are used interchangably. A DQBF is called *false* when it has a *countermodel*, otherwise it is called *true*.

3 Davis-Putnam Resolution for H-Form DQBF

In this section we describe a decision procedure for H-form DQBF in the style of Davis-Putnam resolution. We start by explaining the high-level idea by comparison to propositional DP-resolution.

In a nutshell, DP-resolution for propositional logic eliminates variables by exhaustive resolution—pick variables one at a time in arbitrary order, for every variable produce all resolvents, and then drop all clauses containing the eliminated variable.[1] If at the end the clause set is empty, the formula is satisfiable. If, on the other hand, we are left with the empty clause (we have eliminated all variables, so any clause must be empty), the formula is unsatisfiable, and we have constructed a resolution refutation.

For DQBF we adapt this process in three ways: First, we will only eliminate existential variables. We can still do so in arbitrary order.

Second, we treat universal variables in the spirit of the DQBF proof system M-Res [4]—by splitting clauses into the existential part and a partial-strategy part, initially constructed from universal literals. Strategies may prevent resolution steps if they mismatch; or they may be updated for variables that depend on the pivot—similarly to how it is done in M-Res—with a consistency check in place of the originally used and more strict isomorphism test. Consequently, at the end we obtain either the empty set, in which case the formula is true, or a

[1] The algorithm described by Davis and Putnam [12] also considers unit clauses and pure literals, but since these are neither necessary for completeness, nor complete on their own, we think of DP-resolution as consisting of variable elimination.

set containing clause-strategy pairs with empty existential parts, in which case the formula is false, and the partial strategies form a countermodel.

Third, when eliminating an existential variable x, we will need to *weaken* clauses that do not contain any literal on x with both x and $\overline{x}$ (separately), and such weakened clauses will enter the elimination step for x. M-Res is incomplete for DQBF without weakening, and the same issue forces us to include weakening in our algorithm as well. No such thing is necessary in the propositional case, intuitively because the only way how a variable can directly interact with a clause is if it occurs in the clause. In DQBF however, existential variables can affect dependent universal variables and thereby interact in complex ways with clauses where they do not occur at all. An elegant way of capturing this is by incorporating weakening directly into the resolution rule—resulting in a system known as *w-resolution* [8].

We begin the algorithm exposition by defining some relations and operations in Subsect. 3.1. The algorithm itself is described in Subsect. 3.2, and its correctness and completeness are shown in Subsect. 3.3. We discuss suitable data structures for the storage and manipulation of strategies in Subsect. 3.4.

3.1 Strategy Operations

We introduce a consistency relation and two operations for the manipulation of individual strategy functions.

Definition 1. *Let X be a set of variables and $\varepsilon, \delta \in \langle\!\langle X \rangle\!\rangle$. We say that ε and δ are* consistent*, denoted by $\varepsilon \simeq \delta$, if for every $x \in X$ for which $\varepsilon(x) \neq *$ and $\delta(x) \neq *$ we have $\varepsilon(x) = \delta(x)$.*

By abuse of notation, we treat (partial) assignments as both functions and sets of literals, i.e. an assignment ε corresponds to the set of literals it satisfies, namely $\{x : \varepsilon(x) = 1\} \cup \{\overline{x} : \varepsilon(x) = 0\}$. Through this correspondence we define the *union* of two assignments, and we say that δ *extends (is an extension of)* ε if $\varepsilon \subseteq \delta$.

Lemma 1. *Let X be a set of variables and $\varepsilon, \delta \in \langle\!\langle X \rangle\!\rangle$. The following conditions are equivalent:* (1) ε and δ are consistent; (2) there is an assignment $\gamma \in \langle\!\langle X \rangle\!\rangle$ which extends both ε and δ; (3) $\varepsilon \cup \delta$ is an assignment.

Furthermore, any assignment that extends both ε and δ also extends $\varepsilon \cup \delta$.

Let Ψ be a DQBF, let $u \in \mathrm{vars}_\forall(\Psi)$ be a universal variable, and let h_u and h'_u be individual strategy functions for the variable u; that is, functions from $\langle H_u \rangle$ into $\langle\!\langle \{u\} \rangle\!\rangle$.

- *Consistency:* We say that h_u and h'_u are consistent (written $h_u \simeq h'_u$) when $h_u(\varepsilon) \simeq h'_u(\varepsilon)$ for each $\varepsilon \in \langle H_u \rangle$.
- *Union:* Provided $h_u \simeq h'_u$, their *union* is $(h_u \circ h'_u)(\varepsilon) := h_u(\varepsilon) \cup h'_u(\varepsilon)$.
- *If-then-else:* For each $x \in \mathrm{vars}_\exists(\Psi)$, we define the *if x then h_u else h'_u* function

$$\left(h_u \overset{x}{\bowtie} h'_u\right)(\varepsilon) := \begin{cases} h_u(\varepsilon) & \text{if } \varepsilon(x) = 1\,, \\ h'_u(\varepsilon) & \text{if } \varepsilon(x) = 0\,, \end{cases} \qquad \varepsilon \in \langle H_u \rangle\,.$$

3.2 Definition of the Construction

Given a DQBF $\exists x_1 \cdots \exists x_n \forall u_1(H_{u_1}) \cdots \forall u_m(H_{u_m}) \cdot \psi$, we define recursively a collection of sets $\mathsf{DP}(\Psi, i)$, for i in $\{0, \ldots, n\}$. Each $\mathsf{DP}(\Psi, i)$ is a set of *clause-strategy* pairs. A clause-strategy pair is of the form (C, h), where C is a clause with $\mathrm{vars}(C) \subseteq \mathrm{vars}_\exists(\Psi)$, and h is a strategy for Ψ.

We will obtain the set $\mathsf{DP}(\Psi, i)$ by applying *w-resolution*—resolution preceded by weakening—to $\mathsf{DP}(\Psi, i-1)$. The *w-resolvent* of C and D, over a *pivot* z with $z \notin C$ and $\overline{z} \notin D$, is defined as $C \cup_z D := (C \setminus \{\overline{z}\}) \cup (D \setminus \{z\})$ [8,9]. The w-resolvent is equal to the traditional resolvent if the pivot literals are present in the clauses, but it additionally extends resolution to cases when the pivot is absent from one or both premises—the condition $z \notin C$ and $\overline{z} \notin D$ ensures that weakening by the corresponding pivot literal does not create a tautology.

The recursive definition begins with $\mathsf{DP}(\Psi, 0) := \{(C_\exists, h^{C_\forall}) : C \in \psi\}$, where $C_\exists$ and $C_\forall$ are the existential and universal subclauses of C, and the strategy $h^{C_\forall}$ is the collection of constant functions

$$h_u^{C_\forall}(\varepsilon) := \begin{cases} u \mapsto 0 & \text{if } u \in C_\forall\,, \\ u \mapsto 1 & \text{if } \overline{u} \in C_\forall\,, \\ u \mapsto * & \text{otherwise}\,, \end{cases} \qquad \varepsilon \in \langle H_u \rangle\,,$$

over $u \in \mathrm{vars}_\forall(\Psi)$. Here, $\mathsf{DP}(\Psi, 0)$ is merely a representation of the matrix of Ψ as clause-strategy pairs. The universal subclauses are replaced by strategies, in which each individual literal is represented by the falsifying constant function.

For $i \geq 1$, we define the set $\mathsf{R}(\Psi, i)$ as consisting of all resolvent clause-strategy pairs $(C_0 \cup_{x_i} C_1, h^{1,0})$ for $(C_0, h^0) \neq (C_1, h^1) \in \mathsf{DP}(\Psi, i-1)$ satisfying (a) $x_i \notin C_0$ and $\overline{x_i} \notin C_1$, (b) $C_0 \cup_{x_i} C_1$ is not a tautology, and (c) $h_u^1 \simeq h_u^0$, for each u with $x_i \notin H_u$, where the strategy $h^{1,0}$ is the collection of functions

$$h_u^{1,0} := \begin{cases} h_u^1 \overset{x}{\bowtie} h_u^0 & \text{if } x_i \in H_u\,, \\ h_u^1 \circ h_u^0 & \text{if } x_i \notin H_u\,, \end{cases}$$

over $u \in \mathrm{vars}_\forall(\Psi)$. Finally we define $\mathsf{DP}(\Psi, i)$ as the set

$$\mathsf{R}(\Psi, i) \cup \{(C, h) \in \mathsf{DP}(\Psi, i-1) : x_i \notin \mathrm{vars}(C)\}\,,$$

The set $\mathsf{R}(\Psi, i)$ consists of all possible w-resolvents with pivot x_i formed from clause-strategy pairs (C_1, h^1) and (C_0, h^0) in the previous set $\mathsf{DP}(\Psi, i-1)$, where the individual strategy functions h_u^1, h_u^0 must be consisent whenever u does not depend on x_i. The strategy for the resolvent is the union of h_u^1 and h_u^0 when u is indeed independent of x_i, otherwise it is 'if x_i then h_u^1 else h_u^0.' [2]

[2] Note that we still take the if-then-else even if the functions are compatible, and in particular also if one of the functions is undefined. This is slightly counter-intuitive at first because we could just take the union in those cases, but the if-then-else results in a more compatible strategy and is in fact necessary to ensure completeness.

Note that, for any clause-strategy pair $(C, h) \in \mathsf{DP}(\Psi, i)$, each individual function h_u depends only on the variables $\{x_1, \ldots, x_i\} \cap H_u$. This is an important observation, which we use later in our proof of completeness (Theorem 2).

We will be particularly interested in the final set of clause-strategy pairs generated by this process. Hence we write $\mathsf{DP}(\Psi) := \mathsf{DP}(\Psi, n)$. An immediate consequence of the construction is that each clause-strategy pair $(C, h) \in \mathsf{DP}(\Psi)$ has the empty clause $C = \Box$. The construction is summarised in Aglorithm 1.

Algorithm 1. Davis-Putnam resolution for DQBF.

function DP(Ψ)
  $\Psi^* = \mathsf{DP}(\Psi, 0)$
  for $x \in \mathrm{vars}_\exists(\Psi)$ **do**
    $\Psi^* = \Psi^* \cup$ WEAKEN_AND_RESOLVE(x, Ψ^*)
    $\Psi^* = \Psi^* \setminus \{(C, h) \in \Psi^* : x \in \mathrm{vars}(C)\}$
  end for
  return $\Psi^* \neq \emptyset$
end function

function WEAKEN_AND_RESOLVE(x, Ψ^*)
  $\mathsf{R} = \emptyset$
  for all $(C_0, h^0) \neq (C_1, h^1) \in \Psi^* \times \Psi^*$ **do**
    if $x \notin C_0$, $\overline{x} \notin C_1$ **and** $h^0_u \simeq h^1_u$ when $x \notin H_u$ **then**
      $h^{1,0} = \{h^1_u \overset{x}{\bowtie} h^0_u : x \in H_u\} \cup \{h^1_u \circ h^0_u : x \notin H_u\}$
      $\mathsf{R} = \mathsf{R} \cup \{(C_1 \cup_x C_0, h^{1,0})\}$
    end if
  end for
  return R
end function

There is a crucial difference compared to propositional or even QBF DP-resolution. While in those cases we only resolve pairs of clauses that do contain the pivot, here we need to resolve all pairs that have a w-resolvent (provided that the strategies are compatible where necessary). An interesting special case that arises out of this is *self-resolution*: when we take the w-resolvent of a clause with itself. It is readily verified that a clause C has a self-resolvent on a variable x if, and only if, $x \notin \mathrm{vars}(C)$. Self-resolving C on any variable simply produces C again. Moreover, since both the *self-union* and the *if-then-else* of any strategy function is equivalent to itself, self-resolving an entire clause-strategy pair makes no change to it. Thus, keeping the set $\{(C, h) \in \mathsf{DP}(\Psi, i-1) : x_i \notin \mathrm{vars}(C)\}$ for $\mathsf{DP}(\Psi, i)$ is tantamount to self-resolving each of those clauses and keeping only resolvents, discarding $\mathsf{DP}(\Psi, i-1)$ fully. This allows us to see the algorithm in a slightly different light; as a series of formula transformations. However, self-resolving clauses is not the most intuitive thing to do, and so for the sake of clarity and similarity to other versions of DP-resolution we assume we always resolve different clause-strategy pairs, as written in the pseudocode of Algorithm 1. We invite the reader to appreciate how adopting self-resolution and

full discarding would eliminate case distinctions from some of the forthcoming proofs, arguably making them more elegant, if less humanly.

3.3 Correctness and Completeness

Now we show that the Davis-Putnam construction is both *correct* and *complete*, by which we mean that $\mathsf{DP}(\Psi)$ is non-empty if (*completeness*), and only if (*correctness*), Ψ is false.

Correctness. Our proof of correctness follows the same argument as the proof of soundness in the proof system M-Res [4].[3] For any pair $(C, h) \in \mathsf{DP}(\Psi, i)$, we show that h is a *partial countermodel* for Ψ with respect to C. This means that h behaves like a countermodel on input assignments that falsify C. The notion is captured formally in the statement of the following lemma.

Lemma 2. *Given a DQBF Ψ, an existential variable x_i, a clause-strategy pair $(C, h) \in \mathsf{DP}(\Psi, i)$, and an assignment $\gamma \in \langle \mathrm{vars}_\exists(\Psi) \rangle$, the following holds:*

$$\gamma \textit{ falsifies } C \quad \Rightarrow \quad \gamma \cup h(\gamma) \textit{ falsifies } \psi\,.$$

Proof. We prove the theorem by induction on $i \in \{0, \ldots, n\}$. Let Ψ be the arbitrary DQBF $\Psi := \exists x_1 \cdots \exists x_n \forall u_1(H_{u_1}) \cdots \forall u_m(H_{u_m}) \cdot \psi$.

Base case $i = 0$. Let $(C_\exists, h^{C_\forall}) \in \mathsf{DP}(\Psi, 0)$. By definition, $h^{C_\forall}(\gamma)$ falsifies $C_\forall$ for each γ, and the lemma statement follows immediately.

Inductive Step $i \geq 1$. Let $(C, h) \in \mathsf{DP}(\Psi, i)$. Then, there are pairs (C_0, h^0) and (C_1, h^1) in $\mathsf{DP}(\Psi, i-1)$ such that $C = C_0 \cup_{x_i} C_1$ and $h = h^{1,0}$. Aiming for contradiction, suppose that there exists some $\gamma \in \langle \mathrm{vars}_\exists(\Psi) \rangle$ violating the lemma statement; that is, γ falsifies C, but $\gamma \cup h^{1,0}(\gamma)$ does not falsify ψ.

Now, let us assume for the moment that $\gamma(x_i) = 1$. For each u, let us consider the value of $h_u^{1,0}(\gamma\restriction_{H_u})$. If $x_i \in H_u$, then

$$h_u^{1,0}(\gamma\restriction_{H_u}) \;=\; \big(h_u^1 \overset{x_i}{\bowtie} h_u^0\big)(\gamma\restriction_{H_u}) \;=\; h_u^1(\gamma\restriction_{H_u})\,. \tag{1}$$

Otherwise, if $x_i \notin H_u$, then $h_u^{1,0}(\gamma\restriction_{H_u}) = \big(h_u^1 \circ h_u^0\big)(\gamma\restriction_{H_u})$, from which we get

$$h_u^1(\gamma\restriction_{H_u})(u) \neq * \quad \Rightarrow \quad h_u^{1,0}(\gamma\restriction_{H_u}) = h_u^1(\gamma\restriction_{H_u})\,, \tag{2}$$

by definition of $h_u^1 \circ h_u^0$.

From (1) and (2), we see that $h^{1,0}(\gamma)$ extends $h^1(\gamma)$. Together with the fact that $\gamma \cup h^{1,0}(\gamma)$ does not falsify ψ, we deduce that $\gamma \cup h^1(\gamma)$ does not falsify ψ. This contradicts the inductive hypothesis, which asserts the lemma statement for $(C_1, h^1) \in \mathsf{DP}(\Psi, i-1)$ and the assignment γ, which falsifies $C_1 \subseteq C \cup \{\overline{x_i}\}$.

The alternative case $\gamma(x_i) = 0$ follows the same lines, where the roles of C_1, h^1 and h_u^1 are played instead by C_0, h^0 and h_u^0. One shows that $\gamma \cup h^0(\gamma)$ does not falsify ψ, and a contradiction with the inductive hypothesis ensues.

[3] We cannot use soundness of M-Res, because our strategy compatibility notion is stronger.

The correctness of DP-resolution follows from Lemma 2.

Theorem 1. *Given a DQBF Ψ, if $\mathsf{DP}(\Psi)$ is non-empty, then Ψ is false.*

Proof. Suppose that $\mathsf{DP}(\Psi)$ is non-empty for some DQBF Ψ. Then there exists at least one pair $(\square, h) \in \mathsf{DP}(\Psi)$. Since every assignment falsifies $\square$, h is a countermodel for Ψ, by Lemma 2. Therefore Ψ is false.

Completeness. To demonstrate completeness, we must show that $\mathsf{DP}(\Psi)$ is non-empty whenever Ψ is false. A false DQBF must have at least one countermodel, h say. We show that h is 'represented' at each level of the DP construction; that is, for each $0 \leq i \leq n$ we can find a subset of $\mathsf{DP}(\Psi, i)$ whose strategies collectively describe h. Consequently the final set $\mathsf{DP}(\Psi)$ must be non-empty.

Lemma 3. *Let $\Psi := \exists x_1 \cdots \exists x_n \forall u_1(H_{u_1}) \cdots \forall u_m(H_{u_m}) \cdot \psi$, and let h be a countermodel for Ψ. For each $i \in \{0, \ldots, n\}$ and each ε in $\langle\{x_{i+1}, \ldots, x_n\}\rangle$, there exists some pair $(C, g) \in \mathsf{DP}(\Psi, i)$ such that* (a) ε falsifies C, and (b) $g(\gamma) \subseteq h(\gamma)$ for every $\varepsilon \subseteq \gamma \in \langle \mathrm{vars}_\exists(\Psi)\rangle$.

Proof. **Base case** $i = 0$. Let $\varepsilon \in \langle\{x_1, \ldots, x_n\}\rangle = \langle \mathrm{vars}_\exists(\Psi)\rangle$. Since ε is a full assignment, there is only one extension $\gamma = \varepsilon$. By definition of countermodel, $\gamma \cup h(\gamma)$ falsifies some $C \in \psi$. By definition of $\mathsf{DP}(\Psi, 0)$, there exists a clause-strategy pair $(C_\exists, g^{C_\forall}) \in \mathsf{DP}(\Psi, 0)$, where γ falsifies $C_\exists$ and $h(\gamma)$ extends $g^{C_\forall}(\gamma)$.

Inductive Step $i \geq 1$. Let $\varepsilon \in \langle\{x_{i+1}, \ldots, x_n\}\rangle$ be an assignment with extensions $\varepsilon_0 = \varepsilon \cup \{\overline{x_i}\}$ and $\varepsilon_1 = \varepsilon \cup \{x_i\}$. By the inductive hypothesis, there exists a pair $(C_0, g^0) \in \mathsf{DP}(\Psi, i-1)$ such that ε_0 falsifies C^0 and $h(\gamma_0)$ extends $g^0(\gamma_0)$ for every extension $\gamma_0 \supseteq \varepsilon_0$, and similarly $(C_1, g^1) \in \mathsf{DP}(\Psi, i-1)$ for ε_1. If $(C_0, g^0) = (C_1, g^1)$, we have $x_i \notin \mathrm{vars}(C_0)$, so $(C_0, g^0) \in \mathsf{DP}(\Psi, i)$, and it is the witness for ε.

Otherwise, we claim that the pairs (C_0, g^0) and (C_1, g^1) are resolvable. Firstly, $\overline{x_i} \notin C_0$ because C_0 is falsified by ε_0 and $x_i \notin C_1$ because C_1 is falsified by ε_1; hence the existential parts have an w-resolvent, and this resolvent cannot be a tautology because it is falsified by ε. Secondly, we need to show that the strategies g^0_u and g^1_u for variables u that do not depend on x_i are consistent. Consider $u \in \mathrm{vars}_\forall(\Psi)$ with $x_i \notin H_u$, and an assignment $\gamma \in \langle \mathrm{vars}_\exists(\Psi)\rangle$. We will show that $g^0_u(\gamma{\restriction_{H_u}}) \simeq g^1_u(\gamma{\restriction_{H_u}})$. For $j \in \{0, 1\}$:

- let γ_j be γ with values of the variables $x_i, \ldots, x_n$ overwritten to match ε_j. Since $x_i \notin H_u$, we have $\gamma_0{\restriction_{H_u}} = \gamma_1{\restriction_{H_u}}$.
- Because g^j_u only depends on $x_1, \ldots, x_{i-1}$ (because we have so far only resolved on those variables), we have $g^j_u(\gamma_j{\restriction_{H_u}}) = g^j_u(\gamma{\restriction_{H_u}})$.
- Because $\varepsilon_j \subseteq \gamma_j$, by the inductive hypothesis, $g^j_u(\gamma_j{\restriction_{H_u}})) \subseteq h_u(\gamma_j{\restriction_{H_u}})$.

Because $\gamma_0{\restriction_{H_u}} = \gamma_1{\restriction_{H_u}}$, we have $h_u(\gamma_0{\restriction_{H_u}}) = h_u(\gamma_1{\restriction_{H_u}})$, and by Lemma 1 $g^0_u(\gamma_0{\restriction_{H_u}}) \simeq g^1_u(\gamma_1{\restriction_{H_u}})$. Put together, we have

$$g^0_u(\gamma{\restriction_{H_u}}) = g^0_u(\gamma_0{\restriction_{H_u}}) \simeq g^1_u(\gamma_1{\restriction_{H_u}}) = g^1_u(\gamma{\restriction_{H_u}}).$$

Thus, $g^0_u \simeq g^1_u$.

We claim that the resolvent $(C_1 \uplus_{x_i} C_0, g^{1,0})$ of (C_0, g^0) and (C_1, g^1) is the witness for ε we are looking for. Clearly, ε falsifies $C_1 \uplus_{x_i} C_0$. To verify the second condition, consider an extension $\gamma \supseteq \varepsilon$, and consider γ_0 and γ_1 with the values of x_i overwritten to 0 and 1, respectively.

Consider $u \in \mathrm{vars}_\forall(\Psi)$ with $x_i \in H_u$. Without loss of generality assume $\gamma = \gamma_0 \supseteq \varepsilon_0$. Then, by definition of $g^{1,0}$, we have $g_u^{1,0}(\gamma{\restriction_{H_u}}) = g_u^0(\gamma{\restriction_{H_u}})$, and by the inductive hypothesis $g_u^0(\gamma{\restriction_{H_u}}) \subseteq h_u(\gamma{\restriction_{H_u}})$, as required.

On the other hand, consider $u \in \mathrm{vars}_\forall(\Psi)$ with $x_i \notin H_u$, and observe that $\gamma_0{\restriction_{H_u}} = \gamma_1{\restriction_{H_u}} = \gamma{\restriction_{H_u}}$. Then, by definition of $g^{1,0}$, we have

$$g_u^{1,0}(\gamma{\restriction_{H_u}}) = g_u^0 \circ g_u^1(\gamma{\restriction_{H_u}}) = g_u^0(\gamma{\restriction_{H_u}}) \cup g_u^1(\gamma{\restriction_{H_u}}) = g_u^0(\gamma_0{\restriction_{H_u}}) \cup g_u^1(\gamma_1{\restriction_{H_u}}).$$

Because $\varepsilon_0 \subseteq \gamma_0$, we have $g_u^0(\gamma_0{\restriction_{H_u}}) \subseteq h_u(\gamma_0{\restriction_{H_u}}) = h_u(\gamma{\restriction_{H_u}})$, and similarly $g_u^1(\gamma_1{\restriction_{H_u}}) \subseteq h_u(\gamma{\restriction_{H_u}})$. Thus $g_u^0(\gamma{\restriction_{H_u}}) \cup g_u^1(\gamma{\restriction_{H_u}}) \subseteq h_u(\gamma{\restriction_{H_u}})$ by Lemma 1.

Theorem 2. *Given a DQBF Ψ, if Ψ is false, then $\mathsf{DP}(\Psi)$ is non-empty.*

Theorem 2 follows directly from Lemma 3 for $i = n$ since $\mathsf{DP}(\Psi) = \mathsf{DP}(\Psi, n)$. We will prove a slightly stronger version, which gives a finer lower bound on the size of $\mathsf{DP}(\Psi)$ based on the number of *minimal* countermodels.

Definition 2. *Let g, h be two strategies for a DQBF Ψ. We say that g* extends *h, denoted by $h \subseteq g$, if for every total assignment $\gamma \in \langle \mathrm{vars}_\exists(\Psi) \rangle$, $h(\gamma) \subseteq g(\gamma)$. A countermodel g is* minimal*, if for every countermodel h with $h \subseteq g$, $g = h$. We denote the set of minimal countermodels of Ψ by $\mu(\Psi)$.*

Since the existential part of every pair in $\mathsf{DP}(\Psi)$ is the empty clause, we can afford to abuse our notation and treat $\mathsf{DP}(\Psi)$ as a set of strategies. This allows us to state the following theorem.

Theorem 3. *For a DQBF Ψ, $\mu(\Psi) \subseteq \mathsf{DP}(\Psi)$.*

Proof. By Lemma 3, every minimal countermodel g extends some strategy h in $\mathsf{DP}(\Psi)$. By Lemma 2, h is a countermodel, and by minimality of g, $h = g$.

Theorem 2 now follows from Theorem 3 as any false DQBF must have a minimal countermodel.

Example 1. Let us illustrate a run of Algorithm 1 on the following DQBF Ψ:

$$\exists x_1\, \exists x_2\, \forall u_1(x_1)\, \forall u_2(x_2) \quad (\overline{u_1} \vee x_2) \wedge (u_1 \vee \overline{x_2}) \wedge (\overline{u_2} \vee x_1) \wedge (u_2 \vee \overline{x_1})$$

Algorithm 1 first constructs the set $\mathsf{DP}(\Psi, 0)$, which is

$$\big\{\ (x_2, \{u_1 = 1, u_2 = *\}),\ (\overline{x_2}, \{u_1 = 0, u_2 = *\}),\\ (x_1, \{u_1 = *, u_2 = 1\}),\ (\overline{x_1}, \{u_1 = *, u_2 = 0\})\ \big\}.$$

We begin by eliminating x_1 (we could just as well start with x_2). Resolving the two clauses that contain literals on x_1 is impossible due to strategy mismatch

on u_2, which is independent of x_1. Moving on to w-resolution, resolving (on x_1) the only two clauses that contain x_2 produces a tautology and so can be safely ignored. This leaves us with four w-resolution steps to take: clause pairs $(1,3);(1,4);(2,3);(2;4)$. Consequently, the set $\mathsf{DP}(\Psi,1)$ looks as follows:

$$\{\overbrace{(x_2,\{u_1=1\}),(\overline{x_2},\{u_1=0\})}^{\text{from } \mathsf{DP}(\Psi,0)},$$
$$(x_2,\{u_1=1 \overset{x_1}{\bowtie} *, u_2=1\}),\ (\overline{x_2},\{u_1=0 \overset{x_1}{\bowtie} *, u_2=1\}),$$
$$(x_2,\{u_1=* \overset{x_1}{\bowtie} 1, u_2=0\}),\ (\overline{x_2},\{u_1=* \overset{x_1}{\bowtie} 0, u_2=0\})\}.$$

In the next iteration we eliminate x_2. This time no weakening is necessary as all clauses contain a literal on x_2. Examining all pairs we find out that strategy mismatch on u_1 prevents resolving either of the original pairs with any of the new pairs, and that among the new pairs we can resolve only the first with the fourth and the second with the third. That finally gives us $\mathsf{DP}(\Psi,2)=\mathsf{DP}(\Psi)$:

$$(\square,\{u_1=1 \overset{x_1}{\bowtie} 0, u_2=0 \overset{x_2}{\bowtie} 1\}),\qquad (\square,\{u_1=0 \overset{x_1}{\bowtie} 1, u_2=1 \overset{x_2}{\bowtie} 0\}).$$

The strategy in the first pair can also be succinctly written as $u_1=x_1, u_2=\overline{x_2}$, and the one in the second pair is $u_1=\overline{x_1}, u_2=x_2$. It can easily be verified that both of them are indeed countermodels, in fact minimal ones. Moreover, since these strategies cannot be extended (they already assign a definitive value to all variables in all cases), and every countermodel must extend a strategy from some final pair, Ψ has no further countermodels. □

A natural question is why and how much weakening do we need to make Algorithm 1 work. The fewer clauses to resolve, the better the performance of the algorithm, and while Algorithm 1 works as presented thanks to Theorems 1 and 2, it would be ideal if we could limit ourselves to resolving only clauses that contain the pivot, like in the propositional case. Example 1 shows that does not work—without weakening, resolving on both x_1 and x_2 would be impossible due to strategy mismatch, and hence the algorithm would finish with the empty set, wrongly concluding that Ψ is true. Example 2 goes a step further—it shows that already restricting the algorithm to resolving only pairs where at least one premise contains the pivot kills completeness.

Example 2. Consider the following DQBF Ψ:

$$\exists x_1\ \exists x_2\ \forall u_1(x_1)\ \forall u_2(x_2)\quad (\overline{x_2}\vee u_1\vee\overline{u_2})\wedge(\overline{x_2}\vee\overline{u_1}\vee\overline{u_2})\wedge$$
$$(\overline{x_1}\vee x_2\vee\overline{u_1}\vee u_2)\wedge(x_1\vee x_2\vee u_1\vee u_2).$$

It is readily verified that Ψ is false, with the unique countermodel $u_1=x_1$ and $u_2=x_2$.

Imagine now that Algorithm 1 was modified to resolve only those pairs of clauses where the pivot is present in at least one clause. We will show that this variant would report the formula to be true. We start with $\mathsf{DP}(\Psi,0)$ as usual:

$$\{\ (\overline{x_2}, \{u_1 = 0, u_2 = 1\}),\ (\overline{x_2}, \{u_1 = 1, u_2 = 1\}),$$
$$(\overline{x_1}, x_2, \{u_1 = 1, u_2 = 0\}),\ (x_1, x_2, \{u_1 = 0, u_2 = 0\})\ \}.$$

Assume we first resolve on x_1. We can resolve the third and the fourth clause, the pivot is present in both premises. Tautologies on x_2 prevent all other resolution steps except with the first two clauses. But x_1 does not occur in either of those clauses, so that resolution is forbidden. Thus, $\mathsf{DP}(\Psi, 1)$ is

$$\{\ (\overline{x_2}, \{u_1 = 0, u_2 = 1\}),\ (\overline{x_2}, \{u_1 = 1, u_2 = 0\}),\ (x_2, \{u_1 = 1 \overset{x_1}{\bowtie} 0, u_2 = 0\})\ \}.$$

The u_1-strategies are now pairwise incompatible, and hence resolution on x_2 is impossible. Since all clauses contain a literal on x_2, they are all deleted, and the algorithm finishes with the empty set $\mathsf{DP}(\Psi)$, wrongly concluding that Ψ is true.

Had we resolved the first two clauses on x_1 as required, $\mathsf{DP}(\Psi, 1)$ would have instead been

$$\{\ (\overline{x_2}, \{u_1 = 0, u_2 = 1\}), (\overline{x_2}, \{u_1 = 1, u_2 = 0\}), (x_2, \{u_1 = 1 \overset{x_1}{\bowtie} 0, u_2 = 0\}),$$
$$(\overline{x_2}, \{u_1 = 0 \overset{x_1}{\bowtie} 1, u_2 = 1\}), (\overline{x_2}, \{u_1 = 1 \overset{x_1}{\bowtie} 0, u_2 = 1\})\ \},$$

and a further resolution step is possible, after which we arrive at the correct $\mathsf{DP}(\Psi) = \{(\square, \{u_1 = 1 \overset{x_1}{\bowtie} 0, u_2 = 1 \overset{x_2}{\bowtie} 0\})$, containing the unique countermodel. Notice how we have to weaken each clause that does not contain x_1 in both possible ways, and take both resolvents—only one of them ends up being useful in the next iteration, but we cannot know which one it will be upfront. □

3.4 Representing Strategies

In this subsection we discuss some details for a potential implementation of Algorithm 1. The most complicated component of the algorithm is the storage and reasoning with strategy functions, which can in general become exponentially large. Naturally, it is preferable to store strategies in such a way that consistency checking, union, and if-then-else are as fast as possible. We will show that *ordered binary decision diagrams (OBDDs)* with a fixed ordering, a well-studied target language in knowledge compilation, are a suitable data structure for all these tasks.

Definition 3 ([11,22]). *Let $\mathbb{V}$ be a countable set of propositional variables and $\leq$ a total order on $\mathbb{V}$. An* $\mathrm{OBDD}_{\leq}$ *on $\mathbb{V}$ is a finite rooted labeled directed acyclic graph $\mathcal{O}$ whose each sink is labeled with either 0 or 1, whose non-sinks have out-degree 2, are labeled with variables from $\mathbb{V}$, and their outgoing edges are labeled with the two literals of the vertex label, and such that the vertex labels along any path are pairwise different and respect the order $\leq$.*

The order $\leq$ we use for the OBDD is the same as the order in which we eliminate variables in Algorithm 1, which can be arbitrary but fixed. However,

since our strategy functions are 3-valued, we cannot simply write them down as an OBDD (which is 2-valued). Instead, we will rewrite each strategy g_u into a pair of Boolean strategy functions $(g_u^\top, g_u^\bot)$ defined as

$$g_u^\top(\gamma) = \begin{cases} 1 & \text{if } g_u(\gamma) = 1 \\ 0 & \text{otherwise} \end{cases}, \qquad g_u^\bot(\gamma) = \begin{cases} 1 & \text{if } g_u(\gamma) = 0 \\ 0 & \text{otherwise} \end{cases},$$

and we will represent $g_u^\top$ and $g_u^\bot$ as OBDDs. We refer to the pair $(g_u^\top, g_u^\bot)$ as the *Boolean basis* of g_u. Clearly, any strategy uniquely defines its Boolean basis, and for any Boolean basis it holds that $g_u^\top \wedge g_u^\bot$ is unsatisfiable. Conversely, from a Boolean basis, we can easily reconstruct the original function.

Lemma 4. *Let g^1, g^2 be two Boolean functions such that $g^1 \wedge g^2$ is unsatisfiable. Then, there is a unique 3-valued function g such that $g^1 = g^\top$ and $g^2 = g^\bot$.*

Proof. g is defined to output 1 when g^1 outputs 1, 0 when g^2 outputs 1, and $*$ otherwise. This is well defined thanks to $g^1 \wedge g^2$ being unsatisfiable, and clearly it is the only such g.

The following proposition, which is an easy consequence of the definition, shows how to answer consistency queries with Boolean bases, as well as how to perform union and if-then-else on them.

Proposition 1. *Let g_u, h_u be strategy functions for a universal variable u of a DQBF Ψ. Then*

- $g_u \simeq h_u \iff$ *both* $g_u^\top \wedge h_u^\bot$ *and* $g_u^\bot \wedge h_u^\top$ *are unsatisfiable;*
- $(g_u \circ h_u)^\top = g_u^\top \vee h_u^\top$; $\quad (g_u \circ h_u)^\bot = g_u^\bot \vee h_u^\bot$;
- $(g_u \overset{x}{\bowtie} h_u)^\top = g_u^\top \overset{x}{\bowtie} h_u^\top$; $\quad (g_u \overset{x}{\bowtie} h_u)^\bot = g_u^\bot \overset{x}{\bowtie} h_u^\bot$;

Proposition 1 requires satisfiability checking (also known as consistency checking), taking the conjunction and the disjunction of two functions (also known as bounded conjunction and disjunction), and the if-then-else. OBDDs support consistency checking and bounded conjunction and disjunction in polynomial time [11]. Since the variables on which we perform if-then-else come in a fixed order, it is clear we can compute $g \overset{x}{\bowtie} h$ simply by creating a new x-labeled vertex pointing to g and h. The constant functions in $\mathsf{DP}(\Psi, 0)$ can be represented with 1-node OBDDs, and thus we can perform all updates and all consistency checks in polynomial time.[4] At the end, the algorithm will produce the Boolean basis of a countermodel represented as a pair of OBDDs.

4 NEXP-completeness of CNF H-Form DQBF

For this section we recall an alternative syntactic form of DQBF: A DQBF in *S-form* is an expression of the form $\forall u_1 \cdots \forall u_m \exists x_1(S_{x_1}) \cdots \exists x_n(S_{x_n}) \cdot \psi$, where

[4] In the size of the functions, which may, inevitably, become exponential.

ψ is a propositional formula. The roles of universal and existential variables are swapped; we say that an S-form DQBF is *true* if there is a *model*, i.e. a set of functions for the existential variables with the right universal dependencies whose substitution in the matrix results in a propositional tautology. It is known that evaluating S-form DQBF is NEXP-complete, even if the matrix is restricted to a CNF [1,23].

It is easy to see that evaluating H-form DQBF, like evaluating S-form DQBF, is in NEXP. Additionally, any S-form DQBF can be translated, via negation, into an H-form DQBF, which shows that evaluating H-form DQBF with a DNF matrix is NEXP-complete. If we want the resulting matrix to be a CNF, we must start from an S-form DQBF in DNF. We therefore give a linear-time reduction from S-form DQBF in CNF, which is known to be NEXP-complete, into S-form DQBF in DNF, thereby establishing NEXP-hardness of the latter, and by extension of H-form DQBF in CNF. The reduction is in fact a direct generalization of the Tseitin translation known from propositional logic and QBF [27]—we add universal Tseitin variables and make no existential variable depend on them.

We say that two DQBFs Ψ and Ψ' are *logically equivalent* if they have the same set of models.

Theorem 4. *There is a linear-time algorithm that takes an input S-form DQBF with a CNF matrix and outputs a logically equivalent S-form DQBF with a DNF matrix.*

Proof. Let $\Psi = \forall u_1 \cdots \forall u_m \exists x_1(S_{x_1}) \cdots \exists x_n(S_{x_n}) \cdot \psi$ be an S-form DQBF where the matrix $\psi = C_1 \wedge \cdots \wedge C_r$ is a CNF. We define $\mathsf{DNF}(\Psi)$ as

$$\forall t_1 \cdots \forall t_r \forall u_1 \cdots \forall u_m \exists x_1(S_{x_1}) \cdots \exists x_n(S_{x_n}) \cdot \mathsf{DNF}(\psi)\,,$$

where $\mathsf{DNF}(\psi)$ is the usual propositional Tseitin conversion into DNF applied to the matrix ψ, and whose auxiliary variables are $T := \{t_1, \ldots, t_r\}$, i.e.

$$\mathsf{DNF}(\psi) := \mathsf{DNF}(C_1) \vee \cdots \vee \mathsf{DNF}(C_r) \vee (t_1 \wedge \cdots \wedge t_r)\,,$$

where $\mathsf{DNF}(C_i) := T_i \bigvee_{a \in C_i} T_{i,a}$, $T_{i,a} := (\overline{t_i} \wedge a)$, and $T_i := (t_i \bigwedge_{a \in C_i} \overline{a})$. Note that this translation does indeed generalise QBF Tseitin translation.

Clearly, $\mathsf{DNF}(\Psi)$ can be computed in linear time. We now show that Ψ and $\mathsf{DNF}(\Psi)$ are logically equivalent. Since no existential variable depends on any T-variable, the dependency structure of both formulas is the same.

Let f model Ψ, and let $\alpha \in \langle \mathrm{vars}_\forall(\mathsf{DNF}(\Psi)) \rangle$. If $\alpha(t_1) = \cdots = \alpha(t_r) = 1$, then the top-level term $t_1 \wedge \cdots \wedge t_r$ is satisfied. Otherwise, let i be such that $\alpha(t_i) = 0$. Because f is a model for Ψ, there is a literal $a \in C_i$ for which the following holds: $\alpha{\restriction}_{\mathrm{vars}_\forall(\Psi)} \cup f(\alpha{\restriction}_{\mathrm{vars}_\forall(\Psi)})(a) = 1$. Hence, the term $T_{i,a}$ is satisfied. That means f is a model for $\mathsf{DNF}(\Psi)$ as well.

Conversely, let f be a model for $\mathsf{DNF}(\Psi)$. For an assignment $\alpha \in \langle \mathrm{vars}_\forall(\Psi) \rangle$, let $\mathcal{Z}_\alpha := \{i : C_i[\alpha \cup f(\alpha)] = 0\}$ (we can write $f(\alpha)$ because no function in f depends on any variable in T, and so α contains full information for the application of f). We show that $\mathcal{Z}_\alpha = \emptyset$ for every $\alpha \in \langle \mathrm{vars}_\forall(\Psi) \rangle$, which means

f is a model for Ψ. Let $\alpha \in \langle \text{vars}_\forall(\Psi) \rangle$. Consider the $\beta \in \langle T \rangle$ defined by $\beta(t_i) = 0 \iff i \in \mathcal{Z}_\alpha$. It is easy to see that, whether $i \in \mathcal{Z}_\alpha$ or not, $\alpha \cup \beta \cup f(\alpha \cup \beta)$ falsifies every term T_i and $T_{i,a}$, $a \in C_i$. But f is a model for $\text{DNF}(\Psi)$, so $\alpha \cup \beta \cup f(\alpha \cup \beta)$ must satisfy some term—we conclude that it satisfies the top-level term $t_1 \wedge \cdots \wedge t_r$, and hence $\mathcal{Z}_\alpha = \emptyset$.

Corollary 1. *Evaluating S-form DQBF in DNF and H-form DQBF in CNF is* NEXP*-complete.*

Note that the proof of Theorem 4 goes through without modification even if we omit the terms T_i. Indeed, such a version would be a generalization of the Plaisted-Greenbaum translation for propositional logic and QBF [20].

The computational complexity of H-form DQBF manifests in an interesting way. Algorithm 1 proceeds in essentially the same way as QBF (and propositional) DP-resolution, eliminating variables one by one. In the QBF case, this process runs in at most single-exponential time, since there is only a single-exponential number of different clauses. In DQBF however, that would violate the hypothesis that EXP $\neq$ NEXP, and indeed, our algorithm can in general take double-exponential time and space. This is because our objects are clause-strategy pairs, and the number of different strategies is in general double-exponential. Every variable elimination step can asymptotically square the number of objects in the database, and this repeated squaring, unchecked by a bound on the total number of available objects, results in a double-exponential blow-up. Thus, in a sense, DQBF is 'one of the hardest' problems that can still be tackled with a DP-resolution-style algorithm—repeated squaring unfolds into its worst case here and, under standard complexity assumptions, cannot work for super-double-exponential problems anymore.

5 Conclusion

We presented a new decision procedure for DQBF in the style of Davis-Putnam resolution [12]. Based on the M-Res proof system [4], it constructs partial Herbrand functions along with derived clauses. The algorithm can thus be said to reason directly at the level of strategies. This is in contrast with known decision procedures for DQBF, which rely on quantifier expansion to reduce the problem to SAT/QBF [7,17], or adapt search-based algorithms for QBF by imposing additional constraints that enforce consistency with DQBF semantics [14,15,26]. Our decision procedure requires input DQBF in H-Form, as opposed to the more commonly used S-Form [2]. We presented a linear-time algorithm that converts S-Form DQBF into H-Form DQBF, thereby showing that this requirement can be easily met. As a corollary, we establish NEXP-completeness of evaluating DQBF in H-Form.

References

1. Azhar, S., Peterson, G., Reif, J.: Lower bounds for multiplayer non-cooperative games of incomplete information. J. Comput. Math. Appl. **41**, 957–992 (2001)
2. Balabanov, V., Chiang, H.K., Jiang, J.R.: Henkin quantifiers and Boolean formulae: a certification perspective of DQBF. Theor. Comput. Sci. **523**, 86–100 (2014)
3. Baldoni, R., Coppa, E., D'Elia, D.C., Demetrescu, C., Finocchi, I.: A survey of symbolic execution techniques. ACM Comput. Surv. **51**(3), 50:1–50:39 (2018)
4. Beyersdorff, O., Blinkhorn, J., Mahajan, M.: Building strategies into QBF proofs. J. Autom. Reasoning (2020). (in Press)
5. Giunchiglia, E., Tacchella, A. (eds.): SAT 2003. LNCS, vol. 2919. Springer, Heidelberg (2004). https://doi.org/10.1007/b95238
6. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49059-0_14
7. Bubeck, U., Büning, H.K.: Dependency quantified horn formulas: models and complexity. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 198–211. Springer, Heidelberg (2006). https://doi.org/10.1007/11814948_21
8. Buss, S.R., Hoffmann, J., Johannsen, J.: resolution trees with lemmas: resolution refinements that characterize DLL algorithms with clause learning. Logical Methods Comput. Sci. **4**, (4), (2008). https://doi.org/10.2168/LMCS-4(4:13)2008, https://lmcs.episciences.org/860
9. Buss, S.R., Kolodziejczyk, L.A.: Small stone in pool. Logical Methods Comput. Sci. **10**(2), (2014). https://doi.org/10.2168/LMCS-10(2:16)2014, https://lmcs.episciences.org/852
10. Cashmore, M., Fox, M., Long, D., Magazzeni, D.: A compilation of the full PDDL+ language into SMT. In: Coles, A.J., Coles, A., Edelkamp, S., Magazzeni, D., Sanner, S. (eds.) Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016, pp. 79–87. AAAI Press (2016)
11. Darwiche, A., Marquis, P.: A knowledge compilation map. J. Artif. Intell. Res. **17**, 229–264 (2002) (electronic)
12. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3), 201–215 (1960)
13. Faymonville, P., Finkbeiner, B., Rabe, M.N., Tentrup, L.: Encodings of bounded synthesis. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10205, pp. 354–370. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54577-5_20
14. Fröhlich, A., Kovásznai, G., Biere, A.: A DPLL algorithm for solving DQBF, presented at Workshop on Pragmatics of SAT (POS) (2012). https://arise.or.at/pubpdf/Algorithm_for_Solving__DQBF_.pdf
15. Fröhlich, A., Kovásznai, G., Biere, A., Veith, H.: iDQ: instantiation-based DQBF solving. In: Berre, D.L. (ed.) Workshop on Pragmatics of SAT (POS). EPiC Series in Computing, vol. 27, pp. 103–116. EasyChair (2014)
16. Gitina, K., Reimer, S., Sauer, M., Wimmer, R., Scholl, C., Becker, B.: Equivalence checking of partial designs using dependency quantified boolean formulae. In: IEEE 31st International Conference on Computer Design, ICCD 2013, pp. 396–403. IEEE Computer Society (2013)
17. Gitina, K., Wimmer, R., Reimer, S., Sauer, M., Scholl, C., Becker, B.: Solving DQBF through quantifier elimination. In: Nebel, W., Atienza, D. (eds.) Design, Automation & Test in Europe Conference (DATE), pp. 1617–1622. ACM (2015)

18. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified Boolean formulas. Inf. Comput. **117**(1), 12–18 (1995)
19. Meel, K.S., et al.: constrained sampling and counting: universal hashing meets SAT solving. In: Darwiche, A. (ed.) Beyond NP. AAAI Workshops, vol. WS-16-05. AAAI Press (2016)
20. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symbolic Comput. **2**(3), 293–304 (1986). https://doi.org/10.1016/S0747-7171(86)80028-1
21. Rabe, M.N., Seshia, S.A.: Incremental determinization. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 375–392. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_23
22. Randal E. Bryant: Graph-based algorithms for boolean function manipulation. IEEE Trans. Comput. **C-35**(8), 677–691 (1986). https://doi.org/10.1109/TC.1986.1676819
23. Scholl, C., Jiang, J.R., Wimmer, R., Ge-Ernst, A.: A PSPACE subclass of dependency quantified Boolean formulas and its effective solving. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, pp. 1584–1591. AAAI Press (2019)
24. Solar-Lezama, A., Tancau, L., Bodík, R., Seshia, S.A., Saraswat, V.A.: Combinatorial sketching for finite programs. In: Shen, J.P., Martonosi, M. (eds.) Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2006, pp. 404–415. ACM (2006)
25. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. In: Aho, A.V., et al. (eds.) ACM Symposium on Theory of Computing (STOC), pp. 1–9. ACM (1973)
26. Tentrup, L., Rabe, M.N.: Clausal abstraction for DQBF. In: Janota, M., Lynce, I. (eds.) SAT 2019. LNCS, vol. 11628, pp. 388–405. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24258-9_27
27. Tseitin, G.S.: On the complexity of derivation in propositional calculus. Stud. Constructive Math. Math. Logic Part **2**, 115–125 (1968)
28. Vizel, Y., Weissenbacher, G., Malik, S.: Boolean satisfiability solvers and their applications in model checking. Proc. IEEE **103**(11), 2021–2035 (2015)