2-DQBF Solving and Certification via Property-Directed Reachability Analysis

Long-Hin Fung*, Che Cheng*, Yu-Wei Fan*, Tony Tan†, Jie-Hong Roland Jiang*
*National Taiwan University, Taipei, Taiwan
{r12922017, r11943097, r11943096, jhjiang}@ntu.edu.tw
†University of Liverpool, Liverpool, England
tonytan@liverpool.ac.uk

***Abstract*—Recently a refined complexity analysis of the satisfiability of Dependency Quantified Boolean Formula (DQBF) was established. In particular, it is shown that the satisfiability of 3-DQBF (i.e., DQBF with 3 existential variables) is NEXP-complete and it becomes PSPACE-complete for 2-DQBF. While all state of the art DQBF solvers focus on general DQBF, it is natural to ask if there is an efficient approach for solving 2-DQBF – similar to how modern SAT solvers differentiate between 2-SAT and 3-SAT instances.**

In this paper we show how to exploit modern Property Directed Reachbility (PDR) solvers to solve 2-DQBF instances. We present a novel linear time reduction from 2-DQBF instances to PDR instances and show how to convert the inductive-invariant certificates provided by PDR solvers to the Skolem-function certificates for 2-DQBF instances. The experimental results show that the approach is indeed more efficient than other state-of-the-art DQBF solvers, at least in solving 2-DQBF instances.

***Index Terms*—Dependency quantified Boolean formula (DQBF), property directed reachability (PDR), model extraction, Skolem functions**

I. Introduction

The dependency quantified Boolean formula (DQBF) [1, 2] extends the quantified Boolean formula (QBF) [3] with Henkin quantifiers [4], which allow the dependency set of an existential variable to be explicitly specified. This extension makes DQBF a natural formalism for important applications in system synthesis and verification, such as black-box synthesis [5, 6], controller synthesis [7], engineering change order [8], distributed synthesis for LTL fragments [9], etc, all of which are beyond the expressiveness of QBF. These amotivated the development of various DQBF solvers, e.g., `HQS` [10], `Pedant` [11, 12], `DQBDD` [13]. However, the extension also lifts the complexity of the satisfiability for DQBF to NEXPTIME-complete [1], in contrast to QBF which is "only" PSPACE-complete.

Recent theoretical advancement uncovers important properties of special sub-classes of DQBF. For example, the satisfiability of DQBF$^{\text{de}}$ – DQBF with disjoint or equal dependency sets and CNF matrix – is shown to be in PSPACE [14]. Recently in [15] it is shown that the complexity of DQBF depends on the number of existential variables in the same way as the complexity of SAT depends on the width of the clauses. For example, the complexity of 2-DQBF, i.e., DQBF with 2 existential variables, is PSPACE-complete and for 3-DQBF, it becomes NEXP-complete. This is analogous to SAT whose complexity is NL-complete and NP-complete for 2-SAT and 3-SAT, respectively. The main difference is the exponential blow-up for the DQBF counterpart.

Analogous to modern SAT solvers that often differentiate between 2-SAT and 3-SAT instances and solve them using different methods, it is natural to ask whether we can do the same for DQBF. In this paper, we investigate this question. We are going to exploit the fact that 2-DQBF is essentially a succinct version of 2-CNF formula [15], which implies that the (un)satisfiability of 2-DQBF can be established by checking whether there is a cycle in the (implicit) implication graph that contains two vertices whose labeling assignments are contradicting. We transform such a cycle detection problem into a reachability problem in a way similar to the liveness-to-safety conversion [16]. This conversion allows the state-of-the-art model checking algorithms, such as IC3 [17], Property-Directed Reachability (PDR) [18], and Abstractly Verifying Reachability (AVR) [19], to be exploited for 2-DQBF solving.*

We also show how to convert the certificates provided by the PDR algorithm to the certificates for 2-DQBF. For a false 2-DQBF instance, the trace of the corresponding reachability can be converted directly to a certificate that witnesses the falsity. For a true 2-DQBF instance, however, the conversion is not as straightforward. In this case, the PDR algorithm gives us an inductive state set separating the initial states from the final states derived from the PDR computation and this set is only an over-approximation of the reachable states. We will show how to progressively refine the transition system until it outputs an inductive set that reflects the appropriate Skolem functions for the original 2-DQBF instance.

Note that at first glance, 2-DQBF and PDR may seem to have little in common. PDR is essentially about graph reachability problem, while 2-DQBF is about circuits with two black-boxes. Thus, in retrospect, it is surprising that 2-DQBF can be solved using PDR, owing to the connection between 2-DQBF and 2-SAT established in [15].

Experimental results show that our approach outperforms all state-of-the-art DQBF solvers which demonstrates the effectiveness of our approach, at least in solving 2-DQBF instances. We also compare it with QBF solvers where we

*In this paper we refer to the IC3-based model-checking algorithms as PDR algorithms.

https://doi.org/10.34727/2024/isbn.978-3-85448-065-5_25

reduce 2-DQBF instances to QBF instances. In all instances, the QBF solvers time out. This is not surprising since the only known reduction to QBF is the one in [3], which yields a quadratic blow-up in the number of variables, hence, instances quickly become too large and beyond the capability of even the best QBF solvers.

This paper is organized as follows. In Section II we briefly review the basic notations on DQBF and PDR. We show the reduction from 2-DQBF instances to PDR instances as well as the Skolem functions extraction in Section III. Our experimental results are presented in Section IV. We conclude with Section V. Our code and benchmarks are publicly available on https://github.com/LH104729/2-DQBF-Solving-and-Certification-via-Property-Directed-Reachability-Analysis.

II. Preliminaries

Let $\Sigma = \{0, 1\}$, where 0 and 1 represent the Boolean values *false* and *true*, respectively. As usual, $\neg 0 = 1$ and $\neg 1 = 0$. Let Σ^i and Σ^* denote the sets of Boolean strings of length i and arbitrary length, respectively. We use the symbols a, b, c to denote Boolean constants, i.e., elements in Σ, and the bar version $\bar{a}, \bar{b}, \bar{c}$ to denote Boolean constant vectors, i.e., strings in Σ^* with $|\bar{a}|$ denoting the length of $\bar{a}$. Tuples of values from Σ are written as strings, e.g., 100 denotes $(1, 0, 0)$. Boolean variables are denoted by x, y, z, u, v and the bar version $\bar{x}, \bar{y}, \bar{z}, \bar{u}, \bar{v}$ denote vectors of Boolean variables with $|\bar{x}|$ denoting the length of $\bar{x}$. We insist that in a vector $\bar{x}$ there is no variable occurring more than once. We write $\bar{x}'$ to denote the vector obtained by priming all the variables in $\bar{x}$. For convenience, we view the vectors $\bar{x}, \bar{y}, \bar{z}$ as sets of variables and use set-theoretic operations on them, e.g., $\bar{z} \subseteq \bar{x}$ means every variable in $\bar{z}$ also occurs in $\bar{x}$.

As usual, $\varphi(\bar{x})$ denotes a (Boolean) formula[†] with variables $\bar{x}$. When the variables are not relevant or clear from the context, we simply write φ. For $\varphi(\bar{x})$ and $\psi(\bar{z})$ where $\bar{z} \subseteq \bar{x}$, we write $\varphi(\bar{x}) \Rightarrow \psi(\bar{z})$ to denote that every satisfying assignment of φ is also a satisfying assignment of ψ.

Let $\varphi(\bar{x})$ be a formula. Let $\bar{z} = (z_1, \ldots, z_m) \subseteq \bar{x}$ and $\bar{z}' = (z'_1, \ldots, z'_m)$. We write $\varphi[\bar{z}/\bar{z}']$ to denote the formula obtained by simultaneously substituting each z_i with z'_i for each $1 \leqslant i \leqslant m$. For a string $\bar{a} = (a_1, \ldots, a_m) \in \Sigma^m$, $\varphi[\bar{z}/\bar{a}]$ denotes the formula obtained by assigning each a_i to z_i. When $\bar{z} = \bar{x}$, we just write $\varphi[\bar{a}]$ instead of $\varphi[\bar{x}/\bar{a}]$.

For $\bar{z} \subseteq \bar{x}$ and $\bar{a} \in \Sigma^{|\bar{x}|}$, we write $\bar{a}\big|_{\bar{x}\downarrow\bar{z}}$ to denote the projection of $\bar{a}$ to the components in $\bar{z}$ according to the order of the variables in $\bar{x}$. For example, if $\bar{x} = (x_1, \ldots, x_5)$ and $\bar{z} = (x_1, x_2, x_5)$, then $00101\big|_{\bar{x}\downarrow\bar{z}}$ is 001, i.e., the projection of 00101 to its 1^{st}, 2^{nd} and 5^{th} bits.

A. Dependency Quantified Boolean Formula (DQBF)

A *dependency quantified Boolean formula* (DQBF) in prenex normal form is a formula of the form:

$$\Phi \;\coloneqq\; \forall \bar{x}\; \exists y_1(\bar{z}_1) \cdots\; \exists y_k(\bar{z}_k)\;\; \phi \tag{1}$$

[†]The results in this paper also hold when a formula is written in circuit form, thus, the term "formula" can be taken to also mean "circuit".

where each $\bar{z}_i \subseteq \bar{x}$ and ϕ, called the *matrix*, is a quantifier-free Boolean formula using variables in $\bar{x} \cup \{y_1, \ldots, y_k\}$. We called $\bar{x}$ the *universal variables*, $y_1, \ldots, y_k$ the *existential variables*, and each $\bar{z}_i$ the *dependency set* of y_i. We call Φ a k-DQBF, where k is the number of existential variables in Φ.

A DQBF Φ in the form of Eq. (1) is *satisfiable* if there is a tuple $(f_1, \ldots, f_k)$ of functions, where $f_i : \Sigma^{|\bar{z}_i|} \to \Sigma$ for every $1 \leqslant i \leqslant k$, and by replacing each y_i with $f_i(\bar{z}_i)$, the formula ϕ becomes a tautology. The tuple $(f_1, \ldots, f_k)$ is called the *(satisfying) Skolem functions* for Φ and we say that Φ is satisfiable by the Skolem functions $(f_1, \ldots, f_k)$, i.e., the Skolem functions form a model of Φ.

It is known that the complexity of the satisfiability problem for DQBF is parametric in k, similar to k-SAT: When $k = 2$, it is PSPACE-complete and when $k = 3$, it becomes NEXP-complete and that there is a parsimonious polynomial-time reduction from general DQBF to 3-DQBF [15].

Even before [15] it is already known that k-DQBF is indeed k-CNF formula in an exponentially more succinct representation [20, 21, 22]. We will briefly review this equivalence achieved by a simple rewriting technique from [15], which will be useful later on. Let Φ be k-DQBF as in Eq. (1).

For each $1 \leqslant i \leqslant k$ and for each $\bar{c} \in \Sigma^{|\bar{z}_i|}$, let $X_{i,\bar{c}}$ be a variable and for $d \in \Sigma$, we define the literal $L^d_{i,\bar{c}}$ as:

$$L^d_{i,\bar{c}} \;\coloneqq\; \begin{cases} \neg X_{i,\bar{c}} & \text{if } d = 0 \\ X_{i,\bar{c}} & \text{if } d = 1 \end{cases}$$

Note that $L^d_{i,\bar{c}} = 1$ if and only if $X_{i,\bar{c}} = d$.

For each $(\bar{a}, \bar{b}) \in \Sigma^n \times \Sigma^k$, where $\bar{a} = (a_1, \ldots, a_n)$ and $\bar{b} = (b_1, \ldots, b_k)$, define the clause $C_{\bar{a},\bar{b}}$ as:

$$C_{\bar{a},\bar{b}} \;\coloneqq\; L^{\neg b_1}_{1,\bar{c}_1} \vee \cdots \vee L^{\neg b_k}_{k,\bar{c}_k}$$

where $\bar{c}_i = \bar{a}\big|_{\bar{x}\downarrow\bar{z}_i}$, for each $1 \leqslant i \leqslant k$. The expansion of Φ to a k-CNF formula, denoted by $\exp(\Phi)$, is defined as:

$$\exp(\Phi) \quad\coloneqq\quad \bigwedge_{(\bar{a},\bar{b})\ \text{s.t.}\ \phi[(\bar{x},\bar{y})/(\bar{a},\bar{b})]=0} C_{\bar{a},\bar{b}}$$

It is known that Φ is satisfiable if and only if its expansion $\exp(\Phi)$ is satisfiable (in the sense of Boolean formula) [15]. Moreover, a satisfying Skolem functions $(f_1, \ldots, f_k)$ for Φ correspond to a satisfying assignment of $\exp(\Phi)$, where $X_{i,\bar{c}} = f_i(\bar{c})$ for every $1 \leqslant i \leqslant k$ and $\bar{c} \in \Sigma^{|\bar{z}_i|}$.

As mentioned in the introduction, one of the main applications of DQBF is black-box synthesis, also known as Partial Equivalence Checking (PEC). It is defined as given a Boolean circuit with some black-boxes, check whether there is an implementation of the black-boxes such that the function of the whole circuit is a tautology. It can be naturally expressed as the satisfiability of DQBF where the number of black-boxes corresponds to the number of existential variables in the DQBF.

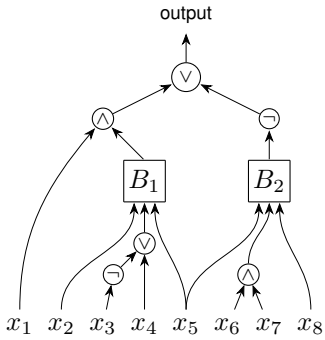


Fig. 1: A PEC example with two black-boxes B_1 and B_2 to be synthesized.

Consider, for example, the circuit with two black-boxes in Figure 1. The PEC solution is equivalent to the Skolem functions of the following 2-DQBF:

$$\forall x_1 \cdots \forall x_{10}\ \exists y_1(x_2, x_9, x_5)\ \exists y_2(x_5, x_{10}, x_8)$$
$$\Big(x_9 = (\neg x_3 \vee x_4) \wedge x_{10} = (x_6 \wedge x_7)\Big) \rightarrow \Big((x_1 \wedge y_1) \vee \neg y_2\Big)$$

The additional variables x_9, x_{10} serve as the Tseitin variable representing the values of $\neg x_3 \vee x_4$ and $x_6 \wedge x_7$, respectively, and y_1 and y_2 are the output variables of the two black-boxes with the dependency sets (x_2, x_9, x_5) and (x_5, x_{10}, x_8).

B. Property-Directed Reachability (PDR)

A *finite-state transition system* is a system $\mathcal{S} = (\bar{x}, I, T)$, where $\bar{x}$ is a finite set of Boolean variables, the initial condition $I(\bar{x})$ is a Boolean formula describing the set of initial states and the transition relation $T(\bar{x}, \bar{x}')$ is a Boolean formula describing the relation between one state and the next. A *state* in the system $\mathcal{S}$ is a Boolean assignment to the variables in $\bar{x}$. Abusing the notation, we denote the states in $\mathcal{S}$ by strings in $\Sigma^{|\bar{x}|}$. We will also view a formula $\varphi(\bar{x})$ as a set of states, i.e., the set of strings $\bar{a}$ where $\varphi[\bar{a}] = 1$.

A *trace* in $\mathcal{S}$ is a sequence of states $\bar{a}_0, \bar{a}_1, \bar{a}_2, \ldots$ such that $I[\bar{a}_0] = 1$ and for every $i \geqslant 0$, $T[\bar{a}_i, \bar{a}_{i+1}] = 1$. A state $\bar{a}$ is *reachable* (in $\mathcal{S}$), if there is a trace $\bar{a}_0, \bar{a}_1, \bar{a}_2, \ldots$ in which $\bar{a}$ appears. A formula $P(\bar{x})$ is *$\mathcal{S}$-invariant* if every state reachable in $\mathcal{S}$ satisfies P. It is *$\mathcal{S}$-inductive*, if $I(\bar{x}) \Rightarrow P(\bar{x})$ and $P(\bar{x}) \wedge T(\bar{x}, \bar{x}') \Rightarrow P(\bar{x}')$. Obviously, P is $\mathcal{S}$-invariant, whenever P is $\mathcal{S}$-inductive. The converse however is not true: It is possible that P is $\mathcal{S}$-invariant, but not $\mathcal{S}$-inductive, since the state that makes P not $\mathcal{S}$-inductive may actually be unreachable in $\mathcal{S}$. The formula P is often called a *safety property*.

Given a transition system $\mathcal{S}$ and property P, the PDR algorithm decides if P is $\mathcal{S}$-invariant. As output, it produces a sequence of Boolean formulas $F_0, F_1, \ldots, F_m$, all using variables in $\bar{x}$, such that:

- F_j is a formula that over-approximates the states that are reachable in at most j steps, for every $0 \leqslant j \leqslant m$.
- If P is $\mathcal{S}$-invariant, $F_m \Rightarrow P$ and F_m is $\mathcal{S}$-inductive.
- If P is not $\mathcal{S}$-invariant, it outputs a counterexample trace $\bar{a}_0, \bar{a}_1, \ldots, \bar{a}_m$ constructed from $F_1, \ldots, F_m$ that violates the safety property P, i.e., $P[\bar{a}_m] = 0$.

For more details on the algorithm and its implementation, we refer interested readers to [23, 17, 18, 19, 24].

III. 2-DQBF Solving and Certification

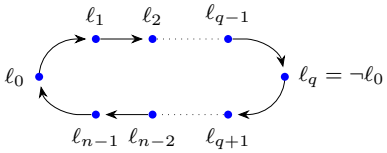
In this section we will show how to solve the satisfiability of 2-DQBF with PDR algorithm. We present a linear time reduction from 2-DQBFs to PDR instances in Section III-A. We then show how to extract the Skolem functions from the PDR certificates in Section III-B.

A. Linear time reduction from 2-DQBF to PDR

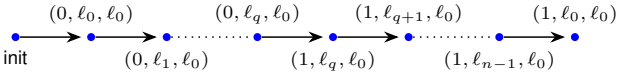
We first recall the approach for solving 2-SAT instances which inspires our use of PDR in solving 2-DQBF. Given a 2-CNF formula φ, we may assume w.l.o.g. that it is in implicative normal form, i.e., a conjunction of implications between literals: $\bigwedge_{1 \leqslant i \leqslant n}(\ell_{i,1} \rightarrow \ell_{i,2})$. It can be viewed as a directed graph $G = (V, E)$, called the implication graph of φ, where V is the set of all the literals in φ and $(\ell_{i,1}, \ell_{i,2})$ and $(\neg\ell_{i,2}, \neg\ell_{i,1})$ are edges in E for every $1 \leqslant i \leqslant n$. The formula φ is not satisfiable if and only if there is a contradicting cycle in G, i.e., a cycle that contains two contradicting literals. In other words, checking the (un)satisfiability of φ is equivalent to checking the existence of a contradicting cycle in G.

The main idea behind our encoding of 2-DQBF with a PDR instance is similar. The only difference is that in the 2-DQBF setting the edges in the implication graph are not explicitly given. Instead, they are succinctly represented by the matrix of the given 2-DQBF and it can be reduced to a PDR instance in which a counterexample trace corresponds to a contradicting cycle (in the implication graph of the expansion of 2-DQBF).

To illustrate, the following contradicting cycle:



where $\ell_q = \neg\ell_0$, can be encoded as a counterexample trace:



where init is the dummy state that serves as the initial state and the transition relation between two states can be constructed from the matrix of the 2-DQBF. The change of the first bit from 0 to 1 occurs when it reaches $(0, \ell_q, \ell_0)$ which indicates the existence of a path from ℓ_0 to $\neg\ell_0$. The safety property states that the system will not reach $(1, \ell, \ell)$ for any literal ℓ.

We now formally present the construction. Given a 2-DQBF $\Phi := \forall\bar{x}\exists y_0(\bar{z}_0)\exists y_1(\bar{z}_1)\ \ \phi(\bar{x}, y_0, y_1)$, where $|\bar{x}| = n$, we construct the transition system $\mathcal{S} = (\bar{r}, I, T)$ where each component is as follows.

The number of variables in $\bar{r}$ is $|\bar{r}| = 6+n+\max(|\bar{z}_0|, |\bar{z}_1|)$. For convenience, we denote $\bar{r}$ as the concatenation of the following vectors:

r_0	r_1	k	b	$\bar{x}$	k_T	b_T	$\bar{z}_T$

where $|r_0| = |r_1| = |k| = |b| = |k_T| = |b_T| = 1$, $|\bar{x}| = n$ and $|\bar{z}_T| = \max(|\bar{z}_0|, |\bar{z}_1|)$. Note that the variables in $\bar{x}$ are reused as variables in the transition system $\mathcal{S}$.

The intended meaning of $\bar{r}$ is to encode a tuple (a, ℓ_i, ℓ_0). The bit r_0 indicates whether it is the initial state. The bit r_1 is the "flag" bit indicating whether we have encountered $(1, \neg\ell_0, \ell_0)$. The vector $(k, b, \bar{x})$ corresponds to the literal $\ell_i := L^b_{k,\bar{z}_k}$, and $(k_T, b_T, \bar{z}_T)$ corresponds to the literal ℓ_0.

The initial condition $I(\bar{r})$ is $\bar{r} = 10\cdots0$ that encodes the dummy init state. The transition relation $T(\bar{r}, \bar{r}')$ is defined as $\varphi_1 \vee \varphi_2 \vee \varphi_3$ where φ_1 encodes the transition from the initial state, φ_2 encodes the transition from the state (r_1, ℓ_i, ℓ_0) to $(r'_1, \ell_{i+1}, \ell_0)$, and φ_3 encodes the transition when the flag bit changes from 0 to 1. Formally, they are defined as follows.

$$
\begin{aligned}
\varphi_1 \;\coloneqq\; & (\bar{r} = 10\cdots0) \wedge (r'_0 = 0) \wedge (r'_1 = 0) \\
& \wedge \begin{cases} (k'_T = 0) \wedge ((b', \bar{z}'_0) = (b'_T, \bar{z}'_T)) & \text{if } k' = 0 \\ (k'_T = 1) \wedge ((b', \bar{z}'_1) = (b'_T, \bar{z}'_T)) & \text{if } k' = 1 \end{cases} \\
\varphi_2 \;\coloneqq\; & (r_0 = r'_0 = 0) \wedge (r_1 = r'_1) \wedge (k' = \neg k) \\
& \wedge \bar{z}_0 \cap \bar{z}_1 = \bar{z}'_0 \cap \bar{z}'_1 \\
& \wedge (k_T, b_T, \bar{z}_T) = (k'_T, b'_T, \bar{z}'_T) \\
& \wedge \begin{cases} \neg\phi[(\bar{x} \setminus \bar{z}_0)/(\bar{x}' \setminus \bar{z}'_0), y_0/b, y_1/\neg b'] & \text{if } k = 0 \\ \neg\phi[(\bar{x} \setminus \bar{z}_1)/(\bar{x}' \setminus \bar{z}'_1), y_0/\neg b', y_1/b] & \text{if } k = 1 \end{cases} \\
\varphi_3 \;\coloneqq\; & (r_0 = r'_0 = 0) \wedge (r_1 = 0) \wedge (r'_1 = 1) \\
& \wedge (k, b, \bar{z}_k) = (k_T, \neg b_T, \bar{z}_T) \\
& \wedge (k, b, \bar{z}_k) = (k', b', \bar{z}'_k) \\
& \wedge (k_T, b_T, \bar{z}_T) = (k'_T, b'_T, \bar{z}'_T)
\end{aligned}
$$

Remark 1. The formula φ_2 captures all the edges in the implication graph of $\exp(\Phi)$ in the sense that:

- For every $\bar{s}_0, \bar{s}_1 \in \Sigma^{|\bar{r}|}$, if $\varphi_2[\bar{s}_0, \bar{s}_1] = 1$, then $L^{d_0}_{a_0,\bar{c}_0} \vee L^{d_1}_{a_1,\bar{c}_1}$ is a clause in $\exp(\Phi)$ where $d_i = \bar{s}_i\big|_{\bar{r}\downarrow b}$, $a_i = \bar{s}_i\big|_{\bar{r}\downarrow k}$ and $\bar{c}_i = \bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_i}$ for $0 \leqslant i \leqslant 1$.
- Conversely, for every clause $L^{d_0}_{a_0,\bar{c}_0} \vee L^{d_1}_{a_1,\bar{c}_1}$ in $\exp(\Phi)$, there is $\bar{s}_0, \bar{s}_1 \in \Sigma^{|\bar{r}|}$, such that $\varphi_2[\bar{s}_0, \bar{s}_1] = 1$, where $d_i = \bar{s}_i\big|_{\bar{r}\downarrow b}$, $a_i = \bar{s}_i\big|_{\bar{r}\downarrow k}$ and $\bar{c}_i = \bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_i}$ for $0 \leqslant i \leqslant 1$.

Finally, the safety invariant property $P(\bar{r})$ is defined as:

$$P(\bar{r}) \;\coloneqq\; (k, b, \bar{z}_k) = (k_T, b_T, \bar{z}_T) \rightarrow \neg r_1$$

That $\mathcal{S}$ and P capture precisely the satisfiability of Φ is stated formally in Theorem 1.

Theorem 1. *Φ is not satisfiable if and only if the safety property P is not $\mathcal{S}$-invariant.*

Proof. We will show that there is a contradicting cycle in the implication graph of $\exp(\Phi)$ if and only if there is trace in $\mathcal{S}$ that violates the safety property $P(\bar{r})$.

(if) Suppose there is a counterexample trace $\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_m$. We will use the following notations. For every $1 \leqslant i \leqslant m$:

$$d_i = \bar{s}_i\big|_{\bar{r}\downarrow b},\ a_i = \bar{s}_i\big|_{\bar{r}\downarrow k} \text{ and } \bar{c}_i = \bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_i}}.$$

Also let:

$$d_T = \bar{s}_i\big|_{\bar{r}\downarrow b_T},\ a_T = \bar{s}_i\big|_{\bar{r}\downarrow k_T} \text{ and } \bar{c}_T = \bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_T}}.$$

Note that $\bar{s}_i\big|_{\bar{r}\downarrow b_T}$ and $\bar{s}_i\big|_{\bar{r}\downarrow k_T}$ stay the same for every $1 \leqslant i \leqslant m$, thus, d_T, a_T and $\bar{c}_T$ are well defined. Let L_i denote the literal $L^{d_i}_{a_i,\bar{c}_i}$, for each $1 \leqslant i \leqslant m$ and L_T the literal $L^{d_T}_{a_T,\bar{c}_T}$.

By the definition of $\mathcal{S}$ and P, we have:

$$\bar{s}_1\big|_{\bar{r}\downarrow r_1} = 0,\ \bar{s}_m\big|_{\bar{r}\downarrow r_1} = 1 \text{ and } L_1 = L_m = L_T.$$

Since $\bar{s}_1\big|_{\bar{r}\downarrow r_1}$ and $\bar{s}_m\big|_{\bar{r}\downarrow r_1}$ differ, there is an index $1 \leqslant q \leqslant m$ such that:

$$\bar{s}_q\big|_{\bar{r}\downarrow r_1} = 0 \text{ and } \bar{s}_{q+1}\big|_{\bar{r}\downarrow r_1} = 1.$$

That is, q is the index when the flag bit changes from 0 to 1. This change only happens when $\varphi_3[\bar{s}_q, \bar{s}_{q+1}] = 1$, which means L_q and L_T are contradicting literals.

Since φ_1 holds only on the dummy init state, for every $1 \leqslant i \leqslant m$ where $i \neq q$:

$$\varphi_2[(\bar{r}, \bar{r}')/(\bar{s}_i, \bar{s}_{i+1})] = 1.$$

By Remark 1, each $L_i \vee L_{i+1}$ is a clause in $\exp(\Phi)$, for every $1 \leqslant i \leqslant m-1$ where $i \neq q$. By routine inspection, these clauses form a cycle in the implication graph of $\exp(\Phi)$ that contains the literals L_T and L_q which are contradicting literals.

(only if) Suppose there is a contradicting cycle in the implication graph of $\exp(\Phi)$. Let the cycle be:

$$\ell_1 \rightarrow \cdots \rightarrow \ell_q \rightarrow \ell_{q+1} \rightarrow \cdots \rightarrow \ell_m = \ell_1.$$

where ℓ_q is $\neg\ell_1$.

For each $1 \leqslant i \leqslant m$, let $d_i, a_i, \bar{c}_i$ be such that $L^{d_i}_{a_i,\bar{c}_i} = \ell_i$. Let $d_T, a_T, \bar{c}_T$ be such that $L^{d_T}_{a_T,\bar{c}_T} = \ell_1$.

Consider the following trace $\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_{m+1}$, where each $\bar{s}_i$ is as follows.

- $\bar{s}_0 = 10\cdots0$.
- For $1 \leqslant i \leqslant q$, $\bar{s}_i\big|_{\bar{r}\downarrow r_0} = 0$, $\bar{s}_i\big|_{\bar{r}\downarrow r_1} = 0$, $\bar{s}_i\big|_{\bar{r}\downarrow k} = a_i$, $\bar{s}_i\big|_{\bar{r}\downarrow b} = d_i$, and $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_i}} = \bar{c}_i$.
- For $i = q+1$, $\bar{s}_i\big|_{\bar{r}\downarrow r_0} = 0$, $\bar{s}_i\big|_{\bar{r}\downarrow r_1} = 1$, $\bar{s}_i\big|_{\bar{r}\downarrow k} = a_q$, $\bar{s}_i\big|_{\bar{r}\downarrow b} = \neg d_q$, and $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_q}} = \bar{c}_q$
- For $q+2 \leqslant i \leqslant m$, $\bar{s}_i\big|_{\bar{r}\downarrow r_0} = 0$, $\bar{s}_i\big|_{\bar{r}\downarrow r_1} = 1$, $\bar{s}_i\big|_{\bar{r}\downarrow k} = a_{i-1}$, $\bar{s}_i\big|_{\bar{r}\downarrow b} = d_{i-1}$, and $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_{i-1}}} = \bar{c}_{i-1}$.
- For $1 \leqslant i \leqslant m+1$, $\bar{s}_i\big|_{\bar{r}\downarrow k_T} = a_T$, $\bar{s}_i\big|_{\bar{r}\downarrow b_T} = d_T$ and $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_T}} = \bar{c}_T$.

By routine inspection, $\bar{s}_0, \ldots, \bar{s}_{m+1}$ is a counterexample trace that violates the safety property P. □

B. Proof extraction and model extraction

In this section, we will show how to extract the certificate for the original 2-DQBF from the certificate provided by the PDR algorithm. Let Φ be the given 2-DQBF as in the previous section and let $\mathcal{S} = (\bar{r}, I, T)$ and P be the constructed transition system and the safety property.

If Φ is unsatisfiable, the PDR algorithm would give us a counterexample trace $\bar{s}_0, \ldots, \bar{s}_m$ from which we can construct the contradicting cycle in the implication graph of $\exp(\Phi)$ as described in the (if) direction in the proof of Theorem 1. Such a contradicting cycle is the certificate for the unsatisfiability.

Now, consider the case when Φ is satisfiable. By Theorem 1, P is $\mathcal{S}$-invariant and the PDR algorithm outputs a Boolean formula $S(\bar{r})$ that is $\mathcal{S}$-invariant and also is an over-approximation of the reachable states from the initial states. We will show how to extract the Skolem functions f_0 and f_1 for y_0 and y_1, respectively, from the formula $S(\bar{r})$.

We first describe the main idea. Let G_Φ be the implication graph of $\exp(\Phi)$. Let $\mathrm{Tr}(G_\Phi)$ be the transitive closure of G_Φ. To avoid clutter with parentheses, we write $L \to L'$ to denote an edge from L to L'. The graph $\mathrm{Tr}(G_\Phi)$ will serve as the guide in constructing the Skolem functions for Φ. The intuition is that if the edge $X_{i,\bar{c}} \to \neg X_{i,\bar{c}}$ is present in $\mathrm{Tr}(G_\Phi)$, then we have to assign $X_{i,\bar{c}}$ to 0, which corresponds to the function f_i where $f_i(\bar{c}) = 0$. Similarly, if the edge $\neg X_{i,\bar{c}} \to X_{i,\bar{c}}$ is present in $\mathrm{Tr}(G_\Phi)$, then we have to assign $X_{i,\bar{c}}$ to 1, which corresponds to the function f_i where $f_i(\bar{c}) = 1$. If both edges are not present in $\mathrm{Tr}(G_\Phi)$, we can freely assign $X_{i,\bar{c}}$ to either 0 or 1. This intuition motivates us to introduce the following definition.

Definition 1. Let $0 \leqslant i \leqslant 1$ and $\bar{c} \in \Sigma^{|\bar{z}_i|}$. The variable $X_{i,\bar{c}}$ is *free* (w.r.t. Φ), if both edges $X_{i,\bar{c}} \to \neg X_{i,\bar{c}}$ and $\neg X_{i,\bar{c}} \to X_{i,\bar{c}}$ are not in $\mathrm{Tr}(G_\Phi)$.

We make a few observations stated formally below that give us the criterion the satisfying Skolem functions should obey.

(O1) If a variable $X_{i,\bar{c}}$ is free, there are Skolem functions (f_0, f_1) where $f_i(\bar{c}) = 0$ and (g_0, g_1) where $g_i(\bar{c}) = 1$. In other words, if $X_{i,\bar{c}}$ is free, we can assign the value of $f_i(\bar{c})$ to either 0 or 1.

(O2) If $X_{i,\bar{c}} \to \neg X_{i,\bar{c}}$ is an edge in $\mathrm{Tr}(G_\Phi)$, then the value of $f_i(\bar{c})$ must be 0 for every Skolem function f_0, f_1 for Φ.

(O3) If $\neg X_{i,\bar{c}} \to X_{i,\bar{c}}$ is an edge in $\mathrm{Tr}(G_\Phi)$, then the value of $f_i(\bar{c})$ must be 1.

(O4) It is not possible that both $X_{i,\bar{c}} \to \neg X_{i,\bar{c}}$ and $\neg X_{i,\bar{c}} \to X_{i,\bar{c}}$ are edges in $\mathrm{Tr}(G_\Phi)$, since both edges forms a contradicting cycle, which will contradict the assumption that Φ is satisfiable.

The main technical difficulty in constructing the Skolem functions is that we do not have the graph $\mathrm{Tr}(G_\Phi)$ explicitly, but only the formula $S(\bar{r})$. To connect $S(\bar{r})$ with $\mathrm{Tr}(G_\Phi)$, we view the formula $S(\bar{r})$ as a graph G_S, where the set of vertices is the same as the set of vertices in G_Φ and the set of edges is as follows. For $b_1, k_1, b_2, k_2 \in \Sigma$, for $\bar{c}_1 \in \Sigma^{|\bar{z}_{k_1}|}$ and $\bar{c}_2 \in \Sigma^{|\bar{z}_{k_2}|}$, $(L^{b_1}_{k_1,\bar{c}_1}, L^{b_2}_{k_2,\bar{c}_2})$ is an edge in G_S if and only if

$$S[0, 0, k_2, b_2, \mathrm{Ext}_{\bar{x}}(\bar{c}_2, \bar{z}_{k_2}), k_1, b_1, \bar{c}_1] = 1, \tag{2}$$

where $\mathrm{Ext}_{\bar{x}}(\bar{c}_2, \bar{z}_{k_2})$ is the assignment of $\bar{x}$ where all variables in $\bar{z}_{k_2}$ are assigned according to $\bar{c}$ and all variables in $\bar{x} \setminus \bar{z}_{k_2}$ are assigned with 0.

The intuition of Eq. (2) is as follows. Recall that the states in $\mathcal{S}$ represent the tuple (b, ℓ_i, ℓ_0) for some literals ℓ_i, ℓ_0. The set of reachable states in $\mathcal{S}$ are such tuples where there is a path from ℓ_0 to ℓ_i in the graph G_Φ, or equivalently, $\ell_0 \to \ell_i$ is an edge in $\mathrm{Tr}(G_\Phi)$. Now the graph G_S can be viewed as an over-approximation of $\mathrm{Tr}(G_\Phi)$, i.e., it contains all the edges in $\mathrm{Tr}(G_\Phi)$ and possibly some other edges that are not in $\mathrm{Tr}(G_\Phi)$.

Lemma 1 below states some useful facts on G_S.

Lemma 1.
- *The set of edges in G_S is an over-approximation of the set of edges in Tr(G_Φ).*
- *If $L_1 \to L_2$ is an edge in G_S and $L_2 \to L_3$ is an edge in Tr(G_Φ), then $L_1 \to L_3$ is an edge in G_S.*
- *If $L \to \neg L$ is an edge in Tr(G_Φ), then $\neg L \to L$ is not an edge in G_S.*

Proof. For the first bullet item, let $L \to L'$ be an edge in $\mathrm{Tr}(G_\Phi)$. Since $\mathrm{Tr}(G_\Phi)$ is the transitive closure of G_Φ, there is a path from L to L' in G_Φ, say:

$$L = L_1 \to L_2 \to \cdots \to L_m = L'$$

Let $d_i, a_i, \bar{c}_i$ be such that $L^{d_i}_{a_i,\bar{c}_i} = L_i$. Then in the transition system $\mathcal{S} = (\bar{r}, I, T)$, consider the states:

$$\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_m,$$

where $\bar{s}_0 = 10\cdots0$ and for each $1 \leqslant i \leqslant m$:

- $\bar{s}_i\big|_{\bar{r}\downarrow r_0} = 0$,
- $\bar{s}_i\big|_{\bar{r}\downarrow r_1} = 0$,
- $\bar{s}_i\big|_{\bar{r}\downarrow k} = a_i$,
- $\bar{s}_i\big|_{\bar{r}\downarrow b} = d_i$,
- $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_i}} = \bar{c}_i$,
- $\bar{s}_i\big|_{\bar{r}\downarrow k_T} = a_1$,
- $\bar{s}_i\big|_{\bar{r}\downarrow b_T} = d_1$,
- $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_T}} = \bar{c}_1$.

It is routine to verify that $\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_m$ is a trace in $\mathcal{S}$. In particular, the state $\bar{s}_m$ is reachable, i.e.:

$$S[0, 0, a_m, d_m, \mathrm{Ext}_{\bar{x}}(\bar{c}_m), a_1, d_1, \bar{c}_1] = 1.$$

Hence by the definition of G_S, $L_0 \to L_n$ is an edge in G_S, and thus, $\mathrm{Tr}(G_\Phi)$ is a subgraph of G_S.

For the second bullet item, let $L_i = L^{d_i}_{a_i,\bar{c}_i}$ for $i = 1, 2, 3$ with $L_1 \to L_2$ being an edge in G_S and $L_2 \to L_3$ being an edge in $\mathrm{Tr}(G_\Phi)$. Consider the states $\bar{s}_0, \bar{s}_1, \bar{s}_2, \bar{s}_3$, where $\bar{s}_0 = 10\cdots0$ and for each $1 \leqslant i \leqslant 3$:

- $\bar{s}_i\big|_{\bar{r}\downarrow r_0} = 0$,
- $\bar{s}_i\big|_{\bar{r}\downarrow r_1} = 0$,
- $\bar{s}_i\big|_{\bar{r}\downarrow k} = a_i$,
- $\bar{s}_i\big|_{\bar{r}\downarrow b} = d_i$,

- $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_i}} = \bar{c}_i$,
- $\bar{s}_i\big|_{\bar{r}\downarrow k_T} = a_1$,
- $\bar{s}_i\big|_{\bar{r}\downarrow b_T} = d_1$, and
- $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_T}} = \bar{c}_1$.

By similar arguments as the first bullet item, we can verify that the state $\bar{s}_1$ is reachable from $\bar{s}_0$, $\bar{s}_2$ is reachable from $\bar{s}_1$, and $\bar{s}_3$ is reachable from $\bar{s}_2$. Hence,

$$S[0, 0, a_3, d_3, \text{Ext}_{\bar{x}}(\bar{c}_3), a_1, d_1, \bar{c}_1] = 1,$$

and $L_1 \to L_3$ is an edge in G_S.

For the third bullet item, let $L \to \neg L$ is an edge in $\text{Tr}(G_\Phi)$ and let $L = L^d_{a,\bar{c}}$. Suppose to the contrary that $\neg L \to L$ is an edge in G_S. Consider the states $\bar{s}_0, \bar{s}_1, \bar{s}_2, \bar{s}_3, \bar{s}_4$, where:

- $\bar{s}_0 = 10\cdots 0$
- $\bar{s}_1\big|_{\bar{r}\downarrow r_0} = 0$, $\bar{s}_1\big|_{\bar{r}\downarrow r_1} = 0$, $\bar{s}_1\big|_{\bar{r}\downarrow k} = a$, $\bar{s}_1\big|_{\bar{r}\downarrow b} = \neg d$, $\bar{s}_1\big|_{\bar{r}\downarrow \bar{z}_{a_i}} = \bar{c}$,
- $\bar{s}_2\big|_{\bar{r}\downarrow r_0} = 0$, $\bar{s}_2\big|_{\bar{r}\downarrow r_1} = 0$, $\bar{s}_2\big|_{\bar{r}\downarrow k} = a$, $\bar{s}_2\big|_{\bar{r}\downarrow b} = d$, $\bar{s}_2\big|_{\bar{r}\downarrow \bar{z}_{a_i}} = \bar{c}$,
- $\bar{s}_3\big|_{\bar{r}\downarrow r_0} = 0$, $\bar{s}_3\big|_{\bar{r}\downarrow r_1} = 1$, $\bar{s}_3\big|_{\bar{r}\downarrow k} = a$, $\bar{s}_3\big|_{\bar{r}\downarrow b} = d$, $\bar{s}_3\big|_{\bar{r}\downarrow \bar{z}_{a_i}} = \bar{c}$,
- $\bar{s}_4\big|_{\bar{r}\downarrow r_0} = 0$, $\bar{s}_4\big|_{\bar{r}\downarrow r_1} = 1$, $\bar{s}_4\big|_{\bar{r}\downarrow k} = a$, $\bar{s}_4\big|_{\bar{r}\downarrow b} = d$, $\bar{s}_4\big|_{\bar{r}\downarrow \bar{z}_{a_i}} = \bar{c}$,
- For $1 \leqslant i \leqslant 4$, $\bar{s}_i\big|_{\bar{r}\downarrow k_T} = a$, $\bar{s}_i\big|_{\bar{r}\downarrow b_T} = \neg d$, and $\bar{s}_i\big|_{\bar{r}\downarrow \bar{z}_{a_T}} = \bar{c}$.

It is routine to check that $\bar{s}_i$ is reachable from $\bar{s}_{i-1}$ for every $1 \leqslant i \leqslant 4$. In particular, the state $\bar{s}_4$ violates the property P, which is a contradiction to the existence of S. Hence $\neg L \to L$ is not an edge in G_S. □

Now consider the following candidate Skolem functions for $0 \leqslant i \leqslant 1$:

$$f_i(\bar{z}_i) \;:=\; S[0, 0, i, 1, \text{Ext}_{\bar{x}}(\bar{z}_i), i, 0, \bar{z}_i] = 1 \wedge S[0, 0, i, 0, \text{Ext}_{\bar{x}}(\bar{z}_i), i, 1, \bar{z}_i] = 0$$

where $\text{Ext}_{\bar{x}}(\bar{z}_i)$ denotes the substitution of all variables in $\bar{x} \setminus \bar{z}_i$ with 0. Intuitively it means that for every $\bar{c} \in \Sigma^{|\bar{z}_i|}$, we assign $f_i(\bar{c}) = 1$ if and only if $L^0_{i,\bar{c}} \to L^1_{i,\bar{c}}$ is an edge in G_S and $L^1_{i,\bar{c}} \to L^0_{i,\bar{c}}$ is not an edge in G_S.

Note that after the first call of the PDR algorithm, the candidate Skolem functions are not necessary the correct ones, because the formula $S(\bar{r})$ is only an over-approximation of the reachable states, as shown in the following example.

Example 1. Let $\psi := \forall x\ \exists y_0(x) \exists y_1(x)\ y_0 \neq y_1$, which is obviously satisfiable. The first call of the PDR algorithm gives us the formula $S(\bar{r})$ where the graph G_S is depicted in Figure 2. The candidate Skolem functions f_0, f_1 defined by G_S are the constant function 0, which obviously are not the correct Skolem functions for ψ.

To verify that the candidate functions f_0, f_1 are indeed the Skolem functions for Φ, we check the satisfiability of the formula:

$$\neg\phi(\bar{x}, y_0, y_1) \ \wedge\ y_0 = f_0(\bar{z}_0) \ \wedge\ y_1 = f_1(\bar{z}_1) \qquad (3)$$

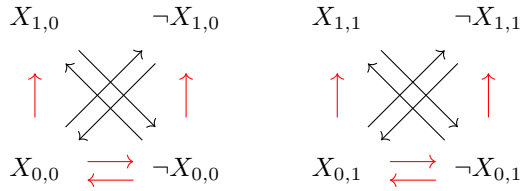


Fig. 2: The transitive closure of the implication graph of the expansion $\exp(\psi)$ has only the black edges, while the graph G_S also contains the red edges.

If it is unsatisfiable, then f_0, f_1 are indeed Skolem functions for y_0, y_1. Otherwise, we need to refine either f_0 or f_1. Let $(\bar{a}, b_0, b_1)$ be a satisfying assignment of the formula in Eq. (3). Let $\bar{c}_0 = \bar{a}\big|_{\bar{x}\downarrow \bar{z}_0}$ and $\bar{c}_1 = \bar{a}\big|_{\bar{x}\downarrow \bar{z}_1}$. It implies the clause $C_{\bar{a},b_0,b_1}$ in $\exp(\Phi)$ is violated, i.e., both literals $L^{\neg b_0}_{0,\bar{c}_0}$ and $L^{\neg b_1}_{1,\bar{c}_1}$ have value 0. Note also that since $\phi(\bar{a}, b_0, b_1) = 0$, by definition, the clause $C_{\bar{a},b_0,b_1}$ is in $\exp(\Phi)$, hence, both $L^{b_0}_{0,\bar{c}_0} \to L^{\neg b_1}_{1,\bar{c}_1}$ and $L^{b_1}_{1,\bar{c}_1} \to L^{\neg b_0}_{0,\bar{c}_0}$ are edges in G_Φ, hence, in $\text{Tr}(G_\Phi)$. In this case we can show that both $X_{0,\bar{c}_0}$ and $X_{1,\bar{c}_1}$ are free, as stated formally in the following lemma.

Lemma 2. *Suppose Φ is satisfiable and suppose $(\bar{a}, b_0, b_1)$ is a satisfying assignment of the formula in Eq. (3). Let $\bar{c}_0 = \bar{a}\big|_{\bar{x}\downarrow \bar{z}_0}$ and $\bar{c}_1 = \bar{a}\big|_{\bar{x}\downarrow \bar{z}_1}$. Then, both $X_{0,\bar{c}_0}$ and $X_{1,\bar{c}_1}$ are free.*

Proof. Since $(\bar{a}, b_0, b_1)$ is a satisfying assignment of the formula in Eq. (3), it is also a satisfying assignment for $\neg\phi$. Thus, the clause $C_{\bar{a},b_0,b_1}$ is in $\exp(\Phi)$, which means that:

$$L^{b_1}_{1,\bar{c}_1} \to L^{\neg b_0}_{0,\bar{c}_0} \text{ and } L^{b_0}_{0,\bar{c}_0} \to L^{\neg b_1}_{1,\bar{c}_1}$$

are both edges in G_Φ, hence, in $\text{Tr}(G_\Phi)$.

Assume to the contrary that at least one of $X_{0,\bar{c}_0}$ and $X_{1,\bar{c}_1}$ is not free. We first assume that $X_{0,\bar{c}_0}$ is not free, i.e., one of $L^{\neg b_0}_{0,\bar{c}_0} \to L^{b_0}_{0,\bar{c}_0}$ and $L^{b_0}_{0,\bar{c}_0} \to L^{\neg b_0}_{0,\bar{c}_0}$ is an edge in $\text{Tr}(G_\Phi)$. The case when $X_{1,\bar{c}_1}$ is not free can be treated in a similar manner.

If $L^{\neg b_0}_{0,\bar{c}_0} \to L^{b_0}_{0,\bar{c}_0}$ is an edge $\text{Tr}(G_\Phi)$, then there is a sequence of edges in $\text{Tr}(G_\Phi)$:

$$L^{b_1}_{1,\bar{c}_1} \to L^{\neg b_0}_{0,\bar{c}_0} \to L^{b_0}_{0,\bar{c}_0} \to L^{\neg b_1}_{1,\bar{c}_1}.$$

Since $\text{Tr}(G_\Phi)$ is the transitive closure of G_ϕ, the edge $L^{b_1}_{1,\bar{c}_1} \to L^{\neg b_1}_{1,\bar{c}_1}$ is also in $\text{Tr}(G_\Phi)$ and hence in G_S. By Lemma 1, $L^{\neg b_1}_{1,\bar{c}_1} \to L^{b_1}_{1,\bar{c}_1}$ is not an edge in G_S. By the construction of f_1, we have $f_1(\bar{c}_1) = \neg b_1$, which contradicts the assumption that $(\bar{a}, b_0, b_1)$ is a satisfying assignment of $\neg\phi \wedge y_0 = f_0 \wedge y_1 = f_1$.

If $L^{b_0}_{0,\bar{c}_0} \to L^{\neg b_0}_{0,\bar{c}_0}$ is an edge in $\text{Tr}(G_\Phi)$, by Lemma 1, $L^{b_0}_{0,\bar{c}_0} \to L^{\neg b_0}_{0,\bar{c}_0}$ is an edge in G_S and $L^{\neg b_0}_{0,\bar{c}_0} \to L^{b_0}_{0,\bar{c}_0}$ is not an edge in G_S. By the construction of f_0, we have $f_0(\bar{c}_0) = \neg b_0$, which contradicts the assumption that $(\bar{a}, b_0, b_1)$ is a satisfying assignment of $\neg\phi \wedge (y_0 = f_0) \wedge (y_1 = f_1)$. □

Remark 2. It is worth noting that Lemma 2 does not contradict with the fact that (f_0, f_1) are not the correct Skolem functions for Φ. It is possible that there are correct Skolem functions

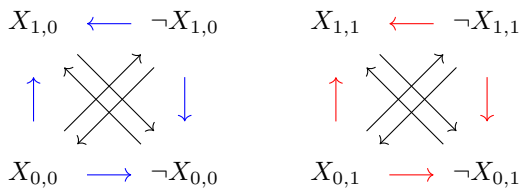


Fig. 3: The closure of the original implication graph are the black edges in black. Edges added to the closure after adding $X_{0,0} \to \neg X_{0,0}$ are in blue. The red edges are the over-aprroximated edges.

g_0, g_1, where g_0 agrees with f_0 on $\bar{c}_0$, but differ from f_1 on $\bar{c}_1$, or that g_0 differs from f_0 on $\bar{c}_0$, but agrees with f_1 on $\bar{c}_1$.

To "fix" the candidate functions f_0, f_1, we add edges into the graph G_S. Consider the 2-DQBF ψ in Example 1. Taking the constant functions $f_0, f_1 = 0$, the formula in Eq. (3) has a satisfying assignment: $x = 0, y_0 = 0, y_1 = 0$. Lemma 2 implies that both $X_{0,0}$ and $X_{1,0}$ are free. We can "refine" the function f_i by adding the edge $X_{0,0} \to \neg X_{0,0}$, which is equivalent to fixing the value $f_0(x)$ to 0. Note that to add the edge $X_{0,0} \to \neg X_{0,0}$ into the graph G_S, we only need to add it to the transition of the system $\mathcal{S}$. After calling the PDR algorithm again, the graph G_S is now as in Figure 3 and the candidate Skolem functions become $f_0 = 0, f_1 = 1$, which are indeed correct Skolem functions of ψ.

We formalise this idea in Algorithm 1. The while-loop corresponds to the refinement of the candidate Skolem functions. In Line 10 we force an assignment the assignment $f_{0,\bar{c}_0} = b_0$ by adding the edge $L^{\neg b_0}_{0,\bar{c}_0} \to L^{b_0}_{0,\bar{c}_0}$ to the transition relation T. The correctness of Algorithm 1 is stated formally in Theorem 2.

Theorem 2. *Algorithm 1 is correct.*

Proof. The correctness of the reduction to the PDR instance $\mathcal{S}$ and P follows from Theorem 1. What is left is to show that the output Skolem functions f_0, f_1 are indeed the satisfying Skolem functions for Φ (when Φ is satisfiable).

It suffices to show that the refinement step is correct. Suppose Φ is satisfiable. Let $(\bar{a}, b_0, b_1)$ be a satisfying assignment of $\neg\varphi \wedge y_0 = f_0 \wedge y_1 = f_1$. Lemma 2 implies that both $f_0(\bar{c}_0)$ and $f_1(\bar{c}_1)$ are free, where $\bar{c}_0 = \bar{a}\big|_{\bar{x}\downarrow\bar{z}_0}$ and $\bar{c}_1 = \bar{a}\big|_{\bar{x}\downarrow\bar{z}_1}$. Thus, there are satisfying Skolem functions for Φ regardless of what we choose to assign in the refinement step.

In each refinement step, we force one assignment, and the number of free variables decreases by at least one per refinement. Hence the algorithm will terminate and the output (f_0, f_1) are correct Skolem functions. □

Note that in the refinement step in Algorithm 1, we choose to force $f_0(\bar{c}_0) = b_0$, which will implicitly force $f_1(\bar{c}_1) = \neg b_1$, due to the clause $C_{\bar{a},b_0,b_1}$ in $\exp(\Phi)$. Alternatively, we may choose to force $f_0(\bar{c}_0) = b_0$, but this choice will not implicitly force the value of $f(\bar{c}_1)$. In our experiments we implement the forcing of $f_0(\bar{c}_0) = b_0$, which we believe is more efficient than the other due to the "implicit" forcing.

Algorithm 1 2DQR

Input: 2-DQBF $\Phi := \forall\bar{x}\ \exists y_0(\bar{z}_0)\exists y_1(\bar{z}_1)\ \varphi(\bar{x}, y_0, y_1)$
1: Run the reduction as in Section III-A on Φ.
2: Let $\mathcal{S} = (\bar{r}, I, T)$ and P be the output.
3: Run the PDR algorithm on $\mathcal{S}$ with the safety property P
4: **if** P is $\mathcal{S}$-invariant **then** ▷ The input Φ is satisfiable
5: Let $S(\bar{r})$ be the inductive safe set formula.
6: Construct the formulas $f_0(\bar{z}_0)$ and $f_1(\bar{z}_1)$ based on S.
7: **while** $\neg\varphi \wedge y_0 = f_0 \wedge y_1 = f_1$ is satisfiable **do** ▷ Refinement
8: Let $(\bar{a}, b_0, b_1)$ be the satisfying assignment.
9: Let $\bar{c}_0 = \bar{a}\big|_{\bar{x}\downarrow\bar{z}_0}$ and $\bar{c}_1 = \bar{a}\big|_{\bar{x}\downarrow\bar{z}_1}$.
10: $T \leftarrow T \vee \textsc{Assign}(0, \bar{c}_0, b_0)$ ▷ Forcing an assignment
11: Call the PDR algorithm on $\mathcal{S} = (\bar{r}, I, T)$ and P.
12: Let $S(\bar{r})$ be the inductive safe set formula.
13: Construct the formulas $f_0(\bar{z}_0)$ and $f_1(\bar{z}_1)$ based on S.
14: **return** f_0, f_1.
15: **else** ▷ The input Φ is not satisfiable
16: **return** UNSAT.

17: **procedure** $\textsc{Assign}(i, \bar{c}, b_0)$
18: **return** $(r_0 = r'_0 = 1) \wedge (r_1 = r'_1) \wedge (k = k' = i) \wedge (\bar{z}_i = \bar{z}'_i = \bar{c}) \wedge (\neg b = b' = b_0) \wedge (k_T, b_T, \bar{z}_T) = (k'_T, b'_T, \bar{z}'_T)$

IV. Experimental Evaluation

Benchmarks: We generate two families of benchmarks: PEC and succinct graph 2-colorability, which are then converted to 2-DQBF.

(PEC) We generate PEC instances with two black-boxes from the ISCAS89 benchmarks [25], where we randomly choose two "sub-circuits" from each circuit and replace them with two black-boxes. We ensure that the dependency sets of the sub-circuits ranges from "being disjoint" to "being almost equal". The instances have 33-674 (universal) variables and the states in the constructed transition systems have 61-1071 bits. In total, there are 624 instances. Each instance is post-processed with the command `fraig` in `ABC` [26] and the resulting test case is in circuit form.

The above method would generate satisfiable instances. We obtain unsatisfiable instances by swapping the dependencies set. Though such swapping does not always guarantee unsatisfiability, but it almost always produces unsatisfiable instances.

(Succinct graph 2-colorability) This family of benchmarks is based on the succinct graph models introduced in [27] where, instead of being given the list of edges in the graph, we are given a Boolean circuit that represents the edges in the graph. Let $C(\bar{u}, \bar{v})$ be a Boolean circuit where $|\bar{u}| = |\bar{v}| = n$. It represents a graph G_C where $\{0,1\}^n$ is the set of vertices and two vertices $\bar{a}$ and $\bar{b}$ are adjacent if and only if $C(\bar{a}, \bar{b}) = 1$. The problem of *succinct graph 2-colorability* is defined as: Given a circuit C, decide if the graph G_C is 2-colorable.

We generate 2-colorability instances by first generating two random permutation circuits $D, D' : \{0,1\}^n \to \{0,1\}^n$. We consider the graph where (x, x') is an edge if the first bit of $D(x)$ is the same as the first bit of $D'(x')$. If we let $D' = D$, the graph defined by C is bipartite, i.e. 2-colorable. Otherwise, the graph is unlikely to be bipartite.

Each random permutation is constructed in m rounds. In each round, we randomly pick $k \in [n]$, generate a clause $c \subseteq \{x_1, \cdots, x_n\}$ with $\Pr[x_i \in c] = \frac{1}{2}$ for $i \neq k$ and $x_k \notin c$, and let x_k to be $x_k \oplus c$. We generate one instance for each $n \in \{2, \cdots, 127\}$ and set $m = 2n$. Again, each instance is post-processed with the command `fraig` in `ABC` [26]. The resulting instances are in circuit form.

We then use the following reduction to obtain 2-DQBF. On input circuit $C(\bar{u}, \bar{v})$ where $|\bar{u}| = |\bar{v}| = n$, let Φ be the following 2-DQBF:

$$\forall \bar{x}_1 \forall \bar{x}_2 \; \exists y_1(\bar{x}_1) \exists y_2(\bar{x}_2) \quad \big(\bar{x}_1 = \bar{x}_2 \;\rightarrow\; y_1 = y_2\big) \\ \wedge \; \big(C(\bar{x}_1, \bar{x}_2) \;\rightarrow\; y_1 \neq y_2\big)$$

where $|\bar{x}_1| = |\bar{x}_2| = n$. Intuitively, we regard the values $0, 1$ as the colors and view a coloring on the vertices as a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$. The formula Φ states that y_1 and y_2 must represent the same function and that two adjacent vertices have different colors. Thus, Ψ is satisfiable if and only if the graph G_C is 2-colorable.

Setup: We implement our method, which we call `2DQR`, using `AVR` [19] as the PDR solver and `Z3` [28] as the SAT solver and a parser for the SMT-LIB 2 format. We compare its performance (with or without Skolem function generation for satisfiable test cases) with `DQBDD` [13], `HQS` [10] and `Pedant` [11, 12]. We run `DQBDD` without generating the Skolem functions, while `HQS` and `Pedant` are run both with and without Skolem function generation. The latest version of `HQS` does not support Skolem function generation. When the Skolem functions are needed, we use an older version of `HQS`. Otherwise, we use the latest version. Since `HQS` and `Pedant` do not take circuit form as input, Tseitin transformation is applied to the instances before being fed to `HQS` and `Pedant`. Each solver had 600 seconds to solve each instance. We run the experiments on Ubuntu 22.04.3 LTS with 48 GB of 2400 MHz DDR4 memory and i5-13400 CPU.

Results: In the first batch of experiments, we compare `2DQR` with other DQBF solvers on the PEC instances. The cactus plots in Figure 4a show the results, where the horizontal axis corresponds to the running time (s) and the vertical axis to the number of solved instances. `2DQR` means without Skolem function generation and `2DQR_skolem` means with Skolem function generation. The time is measured starting from when the input DQBF is read until it terminates/time out, i.e., it includes *the reduction time to PDR instance*, *the pre-processing step `fraig` in `ABC`* and *the output generation*.

For satisfiable instances, `2DQR` outperforms the other solvers by large margins. We remark that the Skolem function generation introduces little run-time overhead since in most cases, they need very few extra calls to the PDR solver. For unsatisfiable instances, `2DQR` outperforms `HQBDD` and `HQS`, while `Pedant` outperformed `2DQR` by a small margin.

Next, we provide a pairwise comparison between our method and other solvers. The scatter plots in the log scale are shown in Figure 4b, where each point represents an instance. The horizontal axis corresponds to the time spent by `2DQR` and the vertical axis represents that by the compared solver.

In most plots, there are a lot of points lying on the bottom right plane, a lot of which have minor differences and are solved within 10 seconds by both methods. Also, there are a lot of points lying on the top boundary of the graph, indicating that there are a lot of cases that are solved by `2DQR` but not the others. As for the graph of `Pedant` v.s. `2DQR` on the unsatisfiable cases, the dots are quite close to the center line, and there are only four cases in which `Pedant` solves but `2DQR` does not.

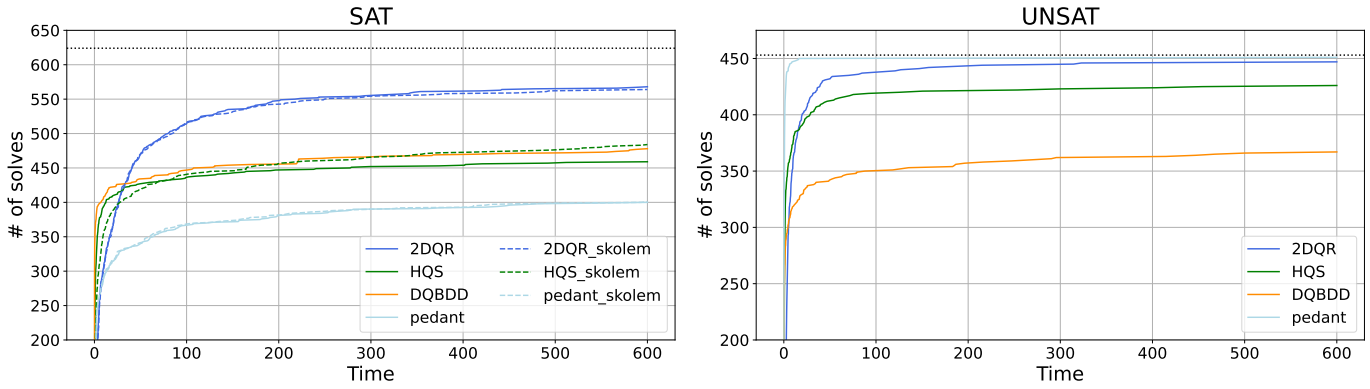
Next, we analyze the circuit size generated by the three methods, `2DQR` generates an AIG in SMT2 format, while `HQS` and `Pedant` generate an AIG in AIGER format. For a fair comparison, we first remove the Tseitin variables from the Skolem function given by `HQS` and `Pedant` using `ABC`, which is done by first removing the definition from the file and running `read skolem.aig; write skolem.aig` on `ABC`. As for `2DQR`, we transform the SMT2 format to verilog format and run `read skolem.v; strash; write skolem.aig`. We also use `fraig` in `ABC` to do some optimization.

For the post-processing steps above, `2DQR` takes 26 seconds, `2DQR` with `fraig` 33 seconds, `HQS` 10 seconds, `HQS` with `fraig` 28 seconds, and `Pedant` 74 seconds. However, `Pedant` with `fraig` takes more than a day to post-process, so we exclude it here. In Figure 4c, we plot the number of AND gates in the Skolem function as a scatter plot to give a pairwise comparison between solvers. Only instances solved by both solvers appear in this plot. It shows that `2DQR` and `HQS` performed similarly, and both outperformed `Pedant` with quite a large margin. When `fraig` is used, that are slight reductions on the number of AND gates for every solver. Figure 4d shows that the performance of `2DQR` and `HQS` are similar, but `2DQR` and `2DQR` with `fraig` are slightly better than `HQS` and `HQS` with `fraig`. Note also that `Pedant`'s distribution is not close to the others.

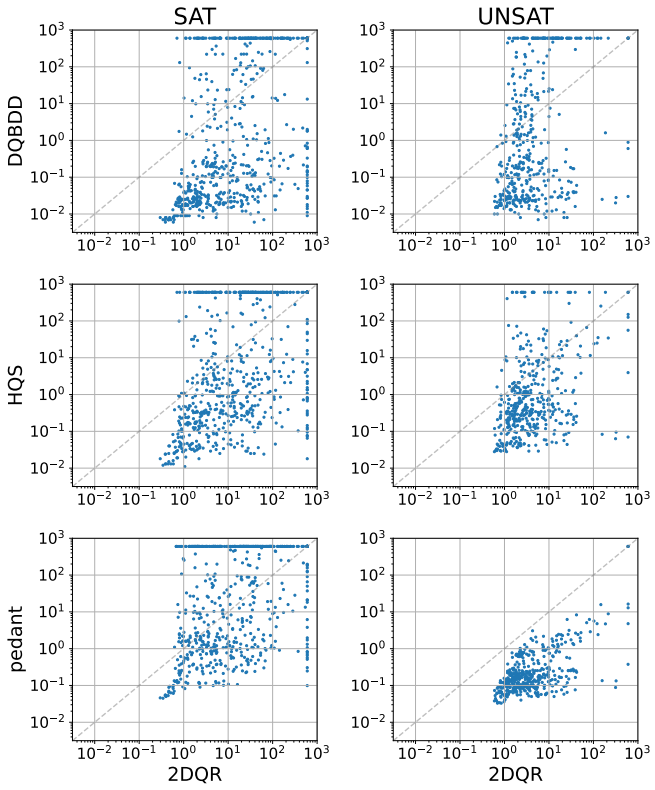
In the construction of Skolem functions, on most PEC instances, there is no extra call to the PDR algorithm. Out of 624 instances, two require 1 extra call, one requires 2 extra calls, one requires 17 extra calls, one requires 22 extra calls and one 29 extra calls.

In the second batch of experiments, we consider the 2-colorability instances. Figure 5 shows the results, where the horizontal axis corresponds to the number of bits of the graph, and the vertical axis corresponds to the time needed to solve the instance. The vertical axis is in log scale for better resolution. For the satisfiable instances, `2DQR` outperforms the other solvers by quite a large margin. Here, each instance needs at least one more extra PDR call for the generation of the Skolem function, which is expected as we need to choose a color to assign to a partition before we can get a coloring. For the unsatisfiable instances, `2DQR`, `DQBDD`, and `HQS` could not solve any instance of size at least 12 bits, but `Pedant` solves almost all of them.

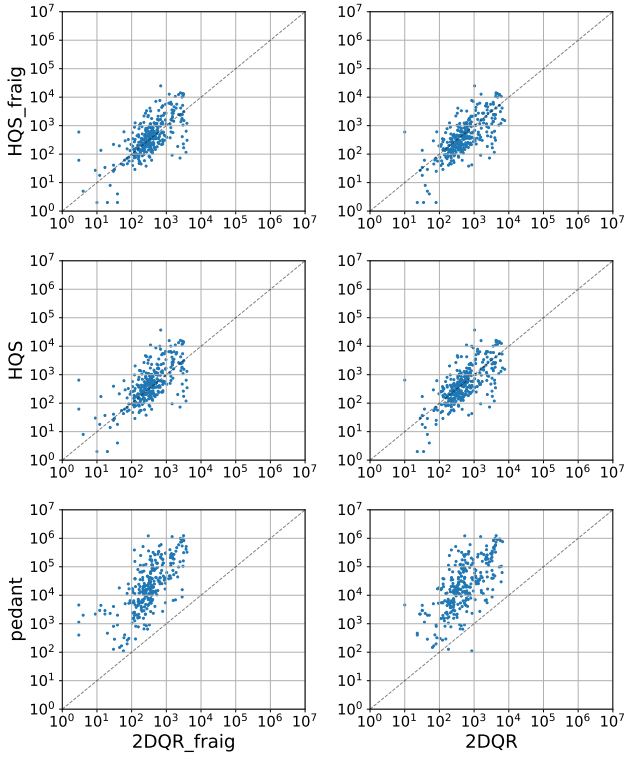
Additional remark: PEC (with 2 black boxes) and succinct 2-colorability are both PSPACE-complete [15, 29]. Thus, it is natural to ask if we can reduce them to QBF instances and use



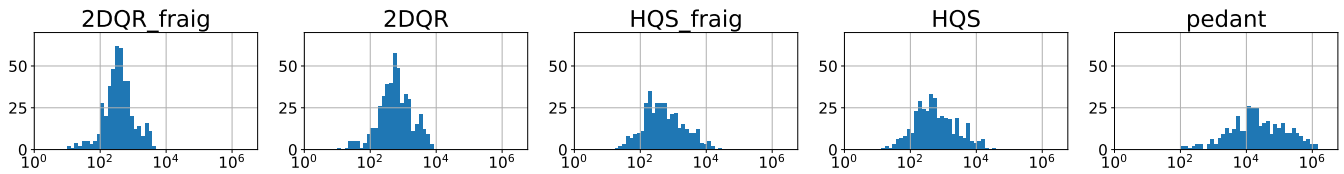
(a) Number of solves vs time in PEC instances. The black horizontal dotted line indicates the number of instances.



(b) Scatter plot for the time needed on each instance.



(c) Scatter plot for the number of AND gates in the Skolem function on each test case. `fraig` denotes the usage of `fraig` during post processing. The axes are the number of AND gates.



(d) Histogram for the number of AND gates in the Skolem function of each method. The x-axis is the number of AND gates and the y-axis is the number of instances.

Fig. 4: Various plots for PEC instances.

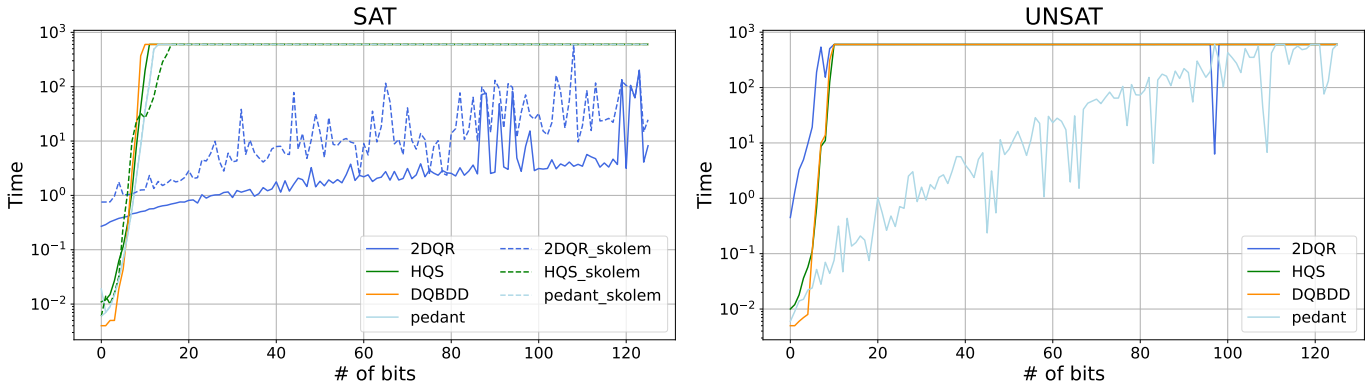


Fig. 5: Time needed to solve for n-bit 2-colorability test cases.

QBF solvers to solve them. In all instances QBF solver time out. This is not surprising since the only known reduction to QBF is the one in [3] which yield quadratic blow-up in the number of variables. For example, in the PEC instances with n universal variables, the resulting QBF would have about $24(n+2)^2$ variables. Our smallest PEC instance already uses 24 universal variables, and the resulting QBF would have at least 16000 variables, which is beyond the capability of current QBF solvers.

V. Conclusions

We introduce a novel technique to use PDR algorithms to solve 2-DQBF instances. The main insight is based on the properties that 2-DQBF is essentially a succinct 2-CNF formula. We also give a method for extracting both the positive and negative certificates for 2-DQBF based on the certificates provided by the PDR solver. We implement our reduction with `AVR` as the PDR solver and empirically show that this approach performs better than the state-of-the-art DQBF solvers on most of the PEC and 2-colorability test cases with a very large margin, except `Pedant` on some unsatisfiable 2-colorability test cases.

We believe our work is just the tip of the iceberg. First, we note that the Skolem function generation could be improved if we use an incremental approach, i.e., by modifying `AVR` to support incremental solving like [30]. Another direction is to efficiently integrate our 2-DQBF solver inside a general DQBF solver, which is very similar in spirit to how modern SAT solvers utilize 2-SAT solvers, e.g., when applying the Unit Propagation strategy, the SAT solver inadvertently is solving 2-SAT instances. We leave this as future work.

Acknowledgements

We acknowledge the generous funding from the National Science and Technology Council of Taiwan under grant NSTC 111-2923-E-002-013-MY3 and the NTU Center of Data Intelligence: Technologies, Applications, and Systems under grant NTU-113L900903.

References

[1] G. Peterson, J. Reif, and S. Azhar. “Lower bounds for multiplayer noncooperative games of incomplete information”. In: *Computers & Mathematics with Applications* 41.7 (2001), pp. 957–992.

[2] V. Balabanov, H.-J. K. Chiang, and J.-H. R. Jiang. “Henkin quantifiers and Boolean formulae: A certification perspective of DQBF”. In: *Theoretical Computer Science* 523 (2014), pp. 86–100.

[3] L. J. Stockmeyer. “The polynomial-time hierarchy”. In: *Theoretical Computer Science* 3.1 (1976), pp. 1–22.

[4] L. Henkin. “Some Remarks on Infinitely Long Formulas”. In: *Journal of Symbolic Logic* 30.1 (1961), pp. 167–183.

[5] C. Scholl and B. Becker. “Checking equivalence for partial implementations”. In: *Design Automation Conference (DAC)*. 2001, pp. 238–243.

[6] K. Gitina et al. “Equivalence checking of partial designs using dependency quantified Boolean formulae”. In: *International Conference on Computer Design (ICCD)*. 2013, pp. 396–403.

[7] R. Bloem, R. Könighofer, and M. Seidl. “SAT-Based Synthesis Methods for Safety Specs”. In: *Verification, Model Checking, and Abstract Interpretation (VMCAI)*. 2014, pp. 1–20.

[8] J.-H. R. Jiang, V. N. Kravets, and N.-Z. Lee. “Engineering Change Order for Combinational and Sequential Design Rectification”. In: *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2020, pp. 726–731.

[9] K. Chatterjee et al. “Distributed synthesis for LTL fragments”. In: *Formal Methods in Computer-Aided Design (FMCAD)*. 2013, pp. 18–25.

[10] R. Wimmer et al. “From DQBF to QBF by Dependency Elimination”. In: *International Conference on Theory and Applications of Satisfiability Testing (SAT)*. 2017, pp. 326–343.

[11] F. Reichl, F. Slivovsky, and S. Szeider. “Certified DQBF Solving by Definition Extraction”. In: *International Conference on Theory and Applications of Satisfiability Testing (SAT)*. 2021, pp. 499–517.

[12] F. Reichl and F. Slivovsky. “Pedant: A Certifying DQBF Solver”. In: *International Conference on Theory and Applications of Satisfiability Testing (SAT)*. 2022, 20:1–20:10.

[13] J. Síc and J. Strejcek. “DQBDD: An Efficient BDD-Based DQBF Solver”. In: *International Conference on Theory and Applications of Satisfiability Testing (SAT)*. 2021, pp. 535–544.

[14] C. Scholl et al. “A PSPACE Subclass of Dependency Quantified Boolean Formulas and Its Effective Solving”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 1584–1591.

[15] L. Fung and T. Tan. “On the Complexity of k-DQBF”. In: *International Conference on Theory and Applications of Satisfiability Testing (SAT)*. 2023, 10:1–10:15.

[16] A. Biere, C. Artho, and V. Schuppan. “Liveness Checking as Safety Checking”. In: *Electronic Notes in Theoretical Computer Science* 66.2 (2002), pp. 160–177.

[17] A. Bradley. “SAT-Based Model Checking without Unrolling”. In: *Verification, Model Checking, and Abstract Interpretation (VMCAI)*. 2011, pp. 70–87.

[18] N. Eén, A. Mishchenko, and R. Brayton. “Efficient Implementation of Property Directed Reachability”. In: *Formal Methods in Computer-Aided Design (FMCAD)*. 2011, pp. 125–134.

[19] A. Goel and K. Sakallah. “AVR: Abstractly Verifying Reachability”. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 2020, pp. 413–422.

[20] U. Bubeck. “Model-Based Transformations for Quantified Boolean Formulas”. PhD thesis. University of Paderborn, 2010.

[21] A. Fröhlich et al. “iDQ: Instantiation-Based DQBF Solving”. In: *Pragmatics of SAT Workshop (POS)*. 2014.

[22] V. Balabanov and J. R. Jiang. “Reducing Satisfiability and Reachability to DQBF”. In: *QBF Workshop*. 2015.

[23] A. Bradley and Z. Manna. “Checking Safety by Inductive Generalization of Counterexamples to Induction”. In: *Formal Methods in Computer-Aided Design (FMCAD)*. 2007, pp. 173–180.

[24] T. Seufert et al. “Everything You Always Wanted to Know About Generalization of Proof Obligations in PDR”. In: *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 42.4 (2023), pp. 1351–1364.

[25] F. Brglez, D. Bryan, and K. Kozminski. “Combinational profiles of sequential benchmark circuits”. In: *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*. 1989, pp. 1929–1934.

[26] R. Brayton and A. Mishchenko. “ABC: An Academic Industrial-Strength Verification Tool”. In: *International Conference on Computer Aided Verification (CAV)*. 2010, pp. 24–40.

[27] H. Galperin and A. Wigderson. “Succinct Representations of Graphs”. In: *Inf. Control.* 56.3 (1983), pp. 183–198.

[28] L. De Moura and N. Bjørner. “Z3: An efficient SMT solver”. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 2008, pp. 337–340.

[29] C. Papadimitriou and M. Yannakakis. “A Note on Succinct Representations of Graphs”. In: *Inf. Control.* 71.3 (1986), pp. 181–185.

[30] M. Blankestijn and A. Laarman. “Incremental Property Directed Reachability”. In: *Formal Methods and Software Engineering*. 2023, pp. 208–227.