Lifting (D)QBF Preprocessing and Solving Techniques to (D)SSAT

Che Cheng[1], **Jie-Hong R. Jiang**[1,2]

[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
{b07901012, jhjiang}@ntu.edu.tw

Abstract

Dependency stochastic Boolean satisfiability (DSSAT) generalizes stochastic Boolean satisfiability (SSAT) in existential variables being Henkinized allowing their dependencies on randomized variables to be explicitly specified. It allows NEXPTIME problems of reasoning under uncertainty and partial information to be compactly encoded. To date, no decision procedure has been implemented for solving DSSAT formulas. This work provides the first such tool by converting DSSAT into SSAT with dependency elimination, similar to converting dependency quantified Boolean formula (DQBF) to quantified Boolean formula (QBF). Moreover, we extend (D)QBF preprocessing techniques and implement the first standalone (D)SSAT preprocessor. Experimental results show that solving DSSAT via dependency elimination is highly applicable and that existing SSAT solvers may benefit from preprocessing.

1 Introduction

Boolean satisfiability (SAT) solving has become an indispensable computation engine in various applications, such as artificial intelligence (Nilsson 1982) and formal verification (Vizel, Weissenbacher, and Malik 2015). Its success has triggered solver development for tackling problems with complexities beyond NP. For instance, MAX-SAT (Li and Manya 2021) enables the encoding of optimization problems that is suitable for encoding various constrained optimization problems (Safarpour et al. 2007); quantified Boolean formula (QBF) (Kleine Büning and Bubeck 2009) allows universal and existential quantification on the Boolean variables; dependency quantified Boolean formula (DQBF) (Scholl and Wimmer 2018) generalizes QBF by allowing incomparable dependencies using Henkin quantifiers.

Another widely explored field takes the idea of counting into consideration. In particular, (un)weighted propositional model counting (Gomes, Sabharwal, and Selman 2009) is concerned with computing the number of satisfying assignments to a Boolean formula. This formalism allows encoding of probabilistic inference (Sang, Bearne, and Kautz 2005). Furthermore, stochastic Boolean satisfiability (SSAT) (Papadimitriou 1985; Littman, Majercik, and Pitassi 2001) generalizes QBF by replacing universal quantifiers with randomized ones, allowing decision and optimization problems under uncertainty to be modeled (Hnich et al. 2011; Lee and Jiang 2018).

Recently, Lee and Jiang (2021) formulated dependency SSAT (DSSAT) to capture decision-making under both uncertainty and partial information. With its NEXPTIME-hard complexity and the counting nature inherited from SSAT, DSSAT can compactly encode complex problems such as probabilistic partial equivalence checking (Lee and Jiang 2021). Due to its recent formulation, no solver has been developed for DSSAT. On the other hand, SSAT solvers have been actively developed, and it has been shown that techniques developed for QBF solving, such as clause selection (Janota and Marques-Silva 2015), may be extended to SSAT solving (Chen, Huang, and Jiang 2021).

The success in SAT, QBF, and DQBF solving is partly due to efficient preprocessing techniques (Eén and Biere 2005; Manthey 2012; Giunchiglia, Marin, and Narizzano 2010; Biere, Lonsing, and Seidl 2011; Wimmer, Scholl, and Becker 2019). Preprocessing is a fast procedure that aims to simplify a given formula in terms of its variable count, clause count, and dependency structure, while keeping the overhead minimized. Although there are prior efforts on preprocessing of propositional model counting (Lagniez and Marquis 2014; Lagniez, Lonca, and Marquis 2016), there is no known preprocessor for SSAT to the best of our knowledge. Given the importance of preprocessing in SAT, QBF, and DQBF solving, questions such as which of the techniques can be applied to (D)SSAT, and whether existing SSAT solvers may benefit from preprocessing arise naturally. Moreover, when a problem is computationally more complex, the time spent on preprocessing becomes less significant compared to the main solving procedure, and we may thus benefit from more expensive preprocessing techniques. As solving DSSAT is highly complex, developing preprocessing techniques for DSSAT is highly motivated.

The main results of this work are as follows. First, we propose a dependency-elimination algorithm that converts DSSAT formulas to SSAT formulas, ready to be solved by SSAT solvers. Combining the algorithm with an existing SSAT solver forms a solution to DSSAT solving. Second, we examine (D)QBF preprocessing techniques for their potential extensions to (D)SSAT and develop the first standalone preprocessor for SSAT and DSSAT formulas. The feasibility and effectiveness of the DSSAT solving flow and the preprocessor are justified experimentally.

The rest of this paper is organized as follows. First, Section 2 defines notations and introduces essential backgrounds. In Section 3, we propose DSSAT solving with dependency elimination. Section 4 studies the applicability of (D)QBF preprocessing techniques for (D)SSAT. Experimental results are evaluated in Section 5. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section, we define the adopted notations, and briefly review the foundations of DQBF, SSAT, and DSSAT.

In the sequel, Boolean values TRUE and FALSE are represented by $\top$ and $\bot$, respectively. Boolean connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\equiv$ are interpreted in their conventional semantics. A *literal* l is either a Boolean variable v or its negation $\neg v$, and we write $\operatorname{var}(l) = v$ to denote the corresponding variable. A clause C is a disjunction of literals. A Boolean formula ϕ is in the conjunctive normal form (CNF) if it is a conjunction of clauses. Whenever appropriate, we also view a clause C as a set of literals and a CNF formula ϕ as a set of clauses. The *substitution* of variable v with formula ψ in formula ϕ is denoted by $\phi[\psi/v]$, and the notation is generalized to literals, where $\phi[\psi/l]$ denotes $\phi[\psi/\operatorname{var}(l)]$ if $l = \operatorname{var}(l)$, and $\phi[\neg\psi/\operatorname{var}(l)]$ if $l = \neg\operatorname{var}(l)$. In addition, we write $\phi[\psi/v_1, \ldots, v_k]$ to denote the substitution of $v_1, \ldots, v_k$ with ψ, i.e., $\phi[\psi/v_1][\psi/v_2]\cdots[\psi/v_k]$.

Let V be a set of Boolean variables. An *assignment* α on V is a mapping from V to $\mathbb{B} = \{\top, \bot\}$, and we denote the set of all possible assignments on V by $\mathcal{A}(V)$. The *domain restriction* of $\alpha \in \mathcal{A}(V_1)$ to a set $V_2 \subseteq V_1$ is denoted by $\alpha|_{D_2}$.

We write $\Phi = \mathcal{Q}.\phi$ to denote a quantified formula, where $\mathcal{Q}$ is the quantifier *prefix*, and ϕ is the *matrix*. Unless specified otherwise, the matrix ϕ is assumed to be in CNF. Given a quantified formula $\Phi = \mathcal{Q}.\phi$, a variable v and a formula ψ, we also write $\Phi[\psi/v]$ to denote $\mathcal{Q}.\phi[\psi/v]$.

2.1 Dependency Quantified Boolean Formulas

Dependency quantified Boolean formula (DQBF) is proposed to capture nonlinearity in dependencies that cannot be captured by quantified Boolean formula (QBF). Instead of a linear order on all variables, DQBF uses Henkin quantifiers (Henkin and Karp 1965) that explicitly specify the set of variables on which an existential variable may depend.

Definition 1 (DQBF syntax)**.** Let $V = U \cup E$ be a set of Boolean variables, where $U = \{u_1, \ldots, u_n\}$ and $E = \{e_1, \ldots, e_m\}$ are the sets of *universal* and *existential* variables, respectively. A DQBF Φ over V is of the form

$$\forall u_1, \ldots, \forall u_n, \exists e_1(D_{e_1}), \ldots, \exists e_m(D_{e_m}).\phi\,,$$

where each $D_{e_j} \subseteq U$ is the *dependency set* of variable e_j, and ϕ is a quantifier-free Boolean formula over V.

For each existential variable e_j, a *Skolem function* $f_j\colon \mathcal{A}(D_{e_j}) \to \mathbb{B}$ specifies the assignment of e_j under each assignment to its dependency D_{e_j}. The vector of functions $\mathcal{F} = (f_1, \ldots, f_m)$ is called a *strategy* for Φ. We shall write $\phi|_{\mathcal{F}}$ to denote the formula obtained by substituting each e_j in ϕ with the corresponding function f_j. A DQBF is satisfiable if and only if there exists a strategy $\mathcal{F} = (f_1, \ldots, f_m)$ such that $\phi|_{\mathcal{F}}$ is a tautology over variables U. Deciding the satisfiability of a DQBF is NEXPTIME-complete (Peterson, Reif, and Azhar 2001).

2.2 Stochastic Boolean Satisfiability

Stochastic Boolean satisfiability (SSAT) is initially proposed in (Papadimitriou 1985) as a *game against nature*.

Definition 2 (SSAT syntax)**.** An SSAT formula Φ over variables V is of the form

$$Q_1 x_1, Q_2 x_2, \ldots, Q_n x_n.\phi\,,$$

where each $Q_i \in \{\exists, \text{Ⅎ}^{p_i}\}$, and ϕ is a quantifier-free Boolean formula over V. The symbol Ⅎ^{p} denotes a *randomized quantifier*, where $\text{Ⅎ}^{p}\, r$ states that r evaluates to $\top$ with probability p. We also write $\Pr[r = \top]$ to denote the probability p in case the probability is not specified in the context.

The satisfying probability of Φ is defined recursively by

- $\Pr[\top] = 1, \Pr[\bot] = 0$,
- $\Pr[\exists x.\Phi'] = \max(\Pr[\Phi'[\bot/x]], \Pr[\Phi'[\top/x]])$, and
- $\Pr[\text{Ⅎ}^{p}\, x.\Phi'] = (1-p)\cdot\Pr[\Phi'[\bot/x]] + p\cdot\Pr[\Phi'[\top/x]]$.

Given an SSAT formula Φ and some $\theta \in [0, 1]$, deciding whether $\Pr[\Phi] \geq \theta$ is PSPACE-complete (Papadimitriou 1985).

2.3 Dependency Stochastic Boolean Satisfiability

Dependency stochastic Boolean satisfiability (DSSAT) is proposed in (Lee and Jiang 2021) as a generalization of SSAT and DQBF.

Definition 3 (DSSAT syntax)**.** Let $V = R \cup E$ be a set of Boolean variables, where $R = \{r_1, \ldots, r_n\}$ and $E = \{e_1, \ldots, e_m\}$ are the sets of *randomized* and *existential* variables, respectively. A DSSAT formula Φ over V is of the form

$$\text{Ⅎ}^{p_1} r_1, \ldots, \text{Ⅎ}^{p_n} r_n, \exists e_1(D_{e_1}), \ldots, \exists e_m(D_{e_m}).\phi\,,$$

where each $D_{e_j} \subseteq R$ is the *dependency set* of variable e_j, and ϕ is a quantifier-free Boolean formula over V.

Given a strategy $\mathcal{F} = (f_1, \ldots, f_m)$ for Φ, the satisfying probability of Φ with respect to $\mathcal{F}$ is defined by

$$\Pr[\Phi|_{\mathcal{F}}] = \sum_{\alpha \in \mathcal{A}(R)} \mathbb{1}_{\phi|_{\mathcal{F}}}(\alpha) w(\alpha)\,,$$

where $\mathbb{1}_{\phi|_{\mathcal{F}}}(\alpha)$ is the indicator function indicating whether α satisfies $\phi|_{\mathcal{F}}$, and $w(\alpha) = \prod_{i=1}^{n} p_i^{\alpha(r_i)}(1-p_i)^{1-\alpha(r_i)}$ is the weighting function for the assignment.

Given a DSSAT formula Φ and some $\theta \in [0, 1]$, deciding whether $\Pr[\Phi] \geq \theta$ is NEXPTIME-complete (Lee and Jiang 2021).

3 Dependency Elimination

The idea of dependency elimination is proposed in (Wimmer et al. 2017) as a generalization to expansion-based techniques that transform a DQBF into QBF and then solve it using a QBF solver. Instead of expanding universal variables completely to linearize the quantifier prefix, the technique eliminates the dependency of a single variable e on $u \in D_e$, thereby enabling a more general choice of dependencies to eliminate.

Theorem 1 (Dependency Elimination for DQBF (Theorem 1 of Wimmer et al. 2017))**.** *Let Φ be a DQBF and assume, without loss of generality, that $u_1 \in D_{e_1}$. Then, Φ is equisatisfiable to*

$$\Phi' = \forall u_1, \ldots, \forall u_n, \exists e_1^0(D_{e_1} \setminus \{u_1\}), \exists e_1^1(D_{e_1} \setminus \{u_1\}),$$
$$\exists e_2(D_{e_2}), \ldots, \exists e_m(D_{e_m}).\phi\big[(\neg u_1 \wedge e_1^0) \vee (u_1 \wedge e_1^1)/e_1\big] .$$

In addition, the problem of finding the optimal set of dependencies to eliminate is formulated as an integer linear program with dynamically added constraints.

As opposed to DQBF, where an expanded universal variable can be removed from the formula using universal reduction, the counting nature of DSSAT makes the elimination-based methods invalid on randomized variables. On the other hand, dependency elimination generalizes naturally to the setting of DSSAT, as it merely deals with the dependency structure. We modify Theorem 1 as follows.

Theorem 2 (Dependency Elimination for DSSAT)**.** *Let Φ be a DSSAT formula and assume, without loss of generality, that $r_1 \in D_{e_1}$. Then, Φ has the same satisfying probability as*

$$\Phi' = \text{Я}\, r_1, \ldots, \text{Я}\, r_n, \exists e_1^0(D_{e_1} \setminus \{r_1\}), \exists e_1^1(D_{e_1} \setminus \{r_1\}),$$
$$\exists e_2(D_{e_2}), \ldots, \exists e_m(D_{e_m}).\phi\big[(\neg r_1 \wedge e_1^0) \vee (r_1 \wedge e_1^1)/e_1\big] .$$

Proof. Note that in the proof of Theorem 1 in (Wimmer et al. 2017), each Skolem function for Φ has a corresponding Skolem function for Φ' that satisfies exactly the same universal assignments. Hence, applying dependency elimination on the DSSAT formula Φ results in some Φ' with the same satisfying probability, witnessed by the corresponding Skolem function. □

4 Preprocessing for (D)SSAT

Even with the high degree of freedom in manipulating dependencies, dependency elimination may blow up the formula exponentially. Its massive success in DQBF solving relies heavily on the power of DQBF preprocessing techniques in HQSpre (Wimmer, Scholl, and Becker 2019). Since there is currently no SSAT—let alone DSSAT—preprocessor to the best of the authors' knowledge, we modify HQSpre to adapt to the stochastic nature (D)SSAT formulas. In addition to making the translation from DSSAT to SSAT practical, the implementation may also serve as a standalone preprocessor for SSAT and DSSAT formulas. A summary of lifted techniques is shown in Table 1.

Technique	Applicability	Note
Unit propagation	△	Theorem 3
Pure literal	△	Theorem 4
Equivalent literals	△	Theorem 5
Universal reduction	X	Theorem 6
SAT-based filters	O	Proposition 1
Rewriting	O	Proposition 1
Resolution	O	Proposition 1
Subsumption	O	Proposition 1
Self-subsumption	O	Proposition 1
Hidden subsumption	O	Proposition 1
Vivification	O	Proposition 1

Note: The symbol "O" denotes that a technique can be applied to (D)SSAT without modification; "△" denotes that a technique can be extended for (D)SSAT with modification; "X" denotes that a technique is unsound for (D)SSAT.

Table 1: Summary of DSSAT extension of DQBF preprocessing techniques.

4.1 Backbones and Monotonic Literals

Definition 4 (Backbone)**.** A literal l is a *backbone* if $\phi[\bot/l]$ is unsatisfiable.

Intuitively, a backbone is a literal assigned true in all satisfying assignments of ϕ. The most well-known backbones are unit literals.

Definition 5 (Unit literal)**.** A literal l is a *unit literal* if $\{l\} \in \phi$.

In (D)QBF, existential backbones can be propagated directly, while universal backbones immediately imply the formula is unsatisfiable. In (D)SSAT, existential unit clauses can be propagated directly. Moreover, randomized unit clauses should be propagated as well, since the unsatisfiable cofactor contributes 0 to the satisfying probability. In addition to performing Boolean constraint propagation, the satisfying probability of the backbone, i.e., $\Pr[l = \top]$, must be recorded. We summarize the handling of backbones in (D)SSAT formulas in Theorem 3.

Theorem 3. *If l is a backbone in the DSSAT formula $\Phi = \mathcal{Q}\,.\phi$, then*

$$\Pr[\Phi] = \begin{cases} \Pr[\mathcal{Q}\,.\phi[\top/l]] & \textit{if } l \textit{ is existential} \\ \Pr[l = \top] \cdot \Pr[\mathcal{Q}\,.\phi[\top/l]] & \textit{if } l \textit{ is randomized.} \end{cases}$$

Proof. Let l be a positive backbone in Φ, and let $\mathcal{F} = (s_{e_1}, \ldots, s_{e_m})$ be a strategy for Φ.

First consider the case where l is existential, and assume, without loss of generality, that $l = e_1$. Let $\mathcal{F}' = (s'_{e_1}, \ldots, s_{e_m})$ be the strategy obtained from replacing s_{e_1} with the constant function that always assigns e_1 to $\top$. Since $\phi[\bot/l]$ is unsatisfiable, we have $\Pr[\Phi|_{\mathcal{F}'}] \geq \Pr[\Phi|_{\mathcal{F}}]$. Since $\mathcal{F}$ is arbitrary, it follows that $\Pr[\Phi] = \Pr[\Phi[\top/l]]$.

Next consider the case where l is randomized, and assume, without loss of generality, that $l = r_1$. Since $\phi[\bot/l]$ is unsatisfiable, we have $\Pr[\Phi|_{\mathcal{F}}] = 0 + \Pr[l = \top] \cdot \Pr[\Phi[\top/l]|_{\mathcal{F}}]$. Since $\mathcal{F}$ is arbitrary, it follows that $\Pr[\Phi] = \Pr[l = \top] \cdot \Pr[\Phi[\top/l]]$.

The case where l is a negative backbone follows analogously. $\square$

We demonstrate the application of unit propagation in Example 1.

Example 1. Consider the DSSAT formula $\Phi_0 = \text{ᴚ}^{0.6} r_1, \text{ᴚ}^{0.8} r_2, \text{ᴚ}^{0.5} r_3, \text{ᴚ}^{0.7} r_4, \exists e_1(r_1), \exists e_2(r_2), \exists e_3(r_3), \exists e_4(r_4).(\neg e_1 \vee r_1)(e_1)(r_2 \vee r_3 \vee \neg e_4)(\neg r_2 \vee r_3)(\neg r_1 \vee \neg r_3 \vee e_2)(\neg e_2 \vee e_3)(r_4 \vee \neg e_4)(r_1 \vee e_4)(r_2 \vee \neg e_1 \vee \neg e_3)$.

Since the clause (e_1) is unit and e_1 is existential, we perform unit propagation and obtain $\Phi_1 = \text{ᴚ}^{0.6} r_1, \text{ᴚ}^{0.8} r_2, \text{ᴚ}^{0.5} r_3, \text{ᴚ}^{0.7} r_4, \exists e_2(r_2), \exists e_3(r_3), \exists e_4(r_4).(r_1)(r_2 \vee r_3 \vee \neg e_4)(\neg r_2 \vee r_3)(\neg r_1 \vee \neg r_3 \vee e_2)(\neg e_2 \vee e_3)(r_4 \vee \neg e_4)(r_1 \vee e_4)(r_2 \vee \neg e_3)$.

After the propagation, (r_1) becomes a unit clause as well, and since r_1 is randomized, we need to record the probability $p = \Pr[r_1 = \top] = 0.6$ in addition to performing unit propagation. The resulting formula is $\Phi_2 = \text{ᴚ}^{0.8} r_2, \text{ᴚ}^{0.5} r_3, \text{ᴚ}^{0.7} r_4, \exists e_2(r_2), \exists e_3(r_3), \exists e_4(r_4).(r_2 \vee r_3 \vee \neg e_4)(\neg r_2 \vee r_3)(\neg r_3 \vee e_2)(\neg e_2 \vee e_3)(r_4 \vee \neg e_4)(r_2 \vee \neg e_3)$.

We next discuss monotonic variables.

Definition 6 (Monotonic variables)**.** A variable v is positively (negatively) monotonic if $\phi[\bot/v] \wedge \neg\phi[\top/v]$ (resp. $\phi[\top/v] \wedge \neg\phi[\bot/v]$) is unsatisfiable.

Intuitively, a positively (negatively) monotonic variable is such that if the assignment α with $\alpha(v) = \bot$ (resp. $\alpha(v) = \top$) satisfies ϕ, then the assignment α' that agrees with α on all variables but v also satisfies ϕ. The most well-known monotonic variables are the pure literals.

Definition 7 (Pure literal)**.** A literal l is a *pure literal* if $\neg l$ does not appear in ϕ.

Since existential variables aim at satisfying ϕ, we can assign any positively (resp. negatively) monotonic existential variable v to $\top$ (resp. $\bot$) while preserving the satisfying probability. On the other hand, as observed in propositional model counting literature (Lagniez and Marquis 2014), monotonic randomized variables cannot be used directly to simplify the formula.

Theorem 4. *If an existential variable v is positively (resp. negatively) monotonic in Φ, then $\Pr[\Phi] = \Pr[\Phi[\top/v]]$ (resp. $\Pr[\Phi[\bot/v]]$).*

Proof. Let v be positively monotonic and, without loss of generality, assume $v = e_1$. Let $\mathcal{F} = (s_{e_1}, \dots, s_{e_m})$ be a strategy for Φ, and let $\mathcal{F}' = (s'_{e_1}, \dots, s_{e_m})$ be the strategy obtained from replacing s_{e_1} with the constant function that always assigns e_1 to $\top$. Since v is positively monotonic, for any assignment $\alpha \in \mathcal{A}(R)$, α satisfies $\phi|_{\mathcal{F}'}$ whenever it satisfies $\phi|_{\mathcal{F}}$. Hence, $\Pr[\Phi|_{\mathcal{F}'}] \geq \Pr[\Phi|_{\mathcal{F}}]$. Since $\mathcal{F}$ is arbitrary, we conclude that $\Pr[\Phi] = \Pr[\Phi[\top/v]]$.

The case where v is negatively monotonic follows analogously. $\square$

We demonstrate the application of pure literal elimination in Example 2.

Example 2. Consider the DSSAT formula Φ_2 in Example 1. We note that e_4 appears negatively in all clauses, i.e. it is negatively monotonic. Since it is existentially quantified, we can assign it to $\bot$, and obtain the formula $\Phi_3 = \text{ᴚ}^{0.8} r_2, \text{ᴚ}^{0.5} r_3, \exists e_2(r_2), \exists e_3(r_3).(\neg r_2 \vee r_3)(\neg r_3 \vee e_2)(\neg e_2 \vee e_3)(r_2 \vee \neg e_3)$.[1] The recorded probability is still $p = 0.6$.

4.2 Equivalent Literals

A clause $C = (l_1 \vee l_2)$ in ϕ can be interpreted as an implication $\neg l_1 \to l_2$ and equivalently $\neg l_2 \to l_1$. By modeling literals as vertices and implications as directed edges, we can construct an implication graph. Each strongly connected component (SCC) in the implication graph forms a set of *equivalent literals*. Intuitively, the matrix may only be satisfied when all equivalent literals are assigned to the same value. A formal definition is given in Definition 8.

Definition 8 (Equivalent literals)**.** The literals l_1 and l_2 are *equivalent* with respect to formula ϕ if $\phi \equiv (\phi \wedge (l_1 \equiv l_2))$.

The cases for (D)QBF are summarized in Theorem 4 of (Wimmer, Scholl, and Becker 2019). Theorem 5 shows the corresponding results in (D)SSAT.

Theorem 5. *Let l_1 and l_2 be equivalent literals in the DSSAT formula $\Phi = \mathcal{Q}.\phi$, where $\mathrm{var}(l_1) = v_1 \neq \mathrm{var}(l_2) = v_2$.*[2]

- *If both v_1 and v_2 are randomized, then $\Pr[\Phi] = \Pr[l_1 = l_2] \cdot \Pr[\mathcal{Q}'.\phi[v/l_1, l_2]]$, where the quantifier prefix $\mathcal{Q}'$ modifies $\mathcal{Q}$ as follows:*
  - *$\text{ᴚ}^{p_{v_1}} v_1, \text{ᴚ}^{p_{v_2}} v_2$ are replaced with $\text{ᴚ}^{p_v} v$, where v is a fresh randomized variable quantified with probability*
  $$p_v = \frac{\Pr[l_1 = l_2 = \top]}{\Pr[l_1 = l_2]},$$
  - *for each existential variable e, the dependency set of e in $\mathcal{Q}'$ is defined as*
  $$D'_e = \begin{cases} D_e & \text{if } D_e \cap \{v_1, v_2\} = \varnothing \\ D_e \cup \{v\} \setminus \{v_1, v_2\} & \text{otherwise.} \end{cases}$$

Otherwise, we assume, without loss of generality, that v_1 is existential.

- *If v_2 is randomized and $v_2 \in D_{v_1}$, then $\Pr[\Phi] = \Pr[\mathcal{Q}.\phi[l_2/l_1]]$.*
- *If v_2 is existential and $D_{v_2} \subseteq D_{v_1}$, then $\Pr[\Phi] = \Pr[\mathcal{Q}.\phi[l_2/l_1]]$.*

Proof. Let l_1 and l_2 be equivalent literals in the DSSAT formula Φ, where $\mathrm{var}(l_1) = v_1 \neq \mathrm{var}(l_2) = v_2$. We assume, without loss of generality, that $l_1 = v_1$ and $l_2 = v_2$ are both positive literals to ease notation.

First consider the case where both v_1 and v_2 are randomized. We note that since $\phi \wedge (l_1 \not\equiv l_2)$ is unsatisfiable, we have

$$\begin{aligned} \Pr[\Phi] &= \Pr[l_1 = l_2 = \top] \cdot \Pr[\Phi[\top/l_1, l_2]] \\ &\quad + \Pr[l_1 = l_2 = \bot] \cdot \Pr[\Phi[\bot/l_1, l_2]] \\ &= \Pr[l_1 = l_2] \cdot (p_v \cdot \Pr[\Phi[\top/l_1, l_2]] \\ &\quad + (1 - p_v) \cdot \Pr[\Phi[\bot/l_1, l_2]]). \end{aligned}$$

[1]Note that since r_4 is no longer present in the matrix, we also drop its quantifier prefix for simplicity.

[2]If $v_1 = v_2$ and $l_1 \neq l_2$, i.e., $l_1 = \neg l_2$, then the formula is unsatisfiable.

In addition, since v_1 and v_2 agree on all satisfiable assignments, any existential variable e that may depend on one of them can essentially depend on both of them, as the assignment of e where the two disagree does not affect the satisfying probability.

Next consider the case where v_1 is existential, and, without loss of generality, assume $v_1 = e_1$. Let $\mathcal{F} = (s_{e_1}, \ldots, s_{e_m})$ be a strategy for Φ. If v_2 is randomized and $v_2 \in D_{v_1}$, let $\mathcal{F}' = (s'_{e_1}, \ldots, s_{e_m})$ be the strategy obtained from replacing s_{e_1} with the function $s'_{e_1}(\alpha) = v_2$ for $\alpha \in \mathcal{A}(D_{e_1})$. Since $\phi \wedge (l_1 \not\equiv l_2)$ is unsatisfiable, we have $\Pr[\Phi|_{\mathcal{F}'}] \geq \Pr[\Phi|_{\mathcal{F}}]$, hence $\Pr[\Phi] = \Pr[\Phi[l_2/l_1]]$. Finally, if v_2 is existential, and $D_{v_2} \subseteq D_{v_1}$, we assume, without loss of generality, that $v_2 = e_2$. Let $\mathcal{F}' = (s'_{e_1}, \ldots, s_{e_m})$ be the strategy obtained from replacing s_{e_1} with the function

$$s'_{e_1}(\alpha) = s_{e_2}(\alpha|_{D_{e_2}})$$

for $\alpha \in \mathcal{A}(D_{e_1})$. Since $\phi \wedge (l_1 \not\equiv l_2)$ is unsatisfiable, we again have $\Pr[\Phi|_{\mathcal{F}'}] \geq \Pr[\Phi|_{\mathcal{F}}]$, hence $\Pr[\Phi] = \Pr[\Phi[l_2/l_1]]$. $\square$

We demonstrate the application of equivalent literals in Example 3.

Example 3. Consider the DSSAT formula Φ_3 in Example 2. Since the implication graph induced by the two-literal clauses has a strongly connected component $\{r_2, r_3, e_2, e_3\}$, these literals are all equivalent. We start by simplifying using $r_2 \equiv r_3$. The corresponding fresh randomized variable r is quantified with probability $p_r = \frac{0.8\cdot 0.5}{0.8\cdot 0.5+0.2\cdot 0.5} = 0.8$, and the recorded probability is updated by $p' = 0.6 \cdot (0.8 \cdot 0.5 + 0.2 \cdot 0.5) = 0.3$. The resulting formula is $\Phi_4 = \text{Я}^{0.8} r, \exists e_2(r), \exists e_3(r).(\neg r \vee e_2)(\neg e_2 \vee e_3)(r \vee \neg e_3)$. Since both e_2 and e_3 may depend on r, we can then replace them with r and obtain $\Phi_5 = \top$. Hence, we conclude that $\Pr[\Phi_0] = 0.3$.

In particular, note that the cases where (1) an existential literal is equivalent to a randomized one outside of its dependency, and (2) two existential literals with incomparable dependencies[3] are equivalent are not covered by Theorem 5. For DQBF, the first case will directly imply that the formula is unsatisfiable, and the second case may be simplified by replacing v_1 and v_2 with a fresh existential variable v, whose dependency set is $D_e = D_{e_1} \cap D_{e_2}$. We remark that both cases do not hold in the setting of DSSAT. In the first case, it only shows that $\Pr[\Phi] < 1$, but there is still a non-zero probability that the existential variable "guesses" the randomized one. In the second case, we show in Example 4 that guessing based on incomparable information may be beneficial in the stochastic setting.

Example 4. Reconsider the formula Φ_3 in Example 2. Although $e_2 \equiv e_3$ holds, if we substitute them with a fresh existential variable e with dependency $D_e = D_{e_2} \cap D_{e_3} = \varnothing$, the formula will become $\Phi'_4 = \text{Я}^{0.8} r_2, \text{Я}^{0.5} r_3, \exists e(\varnothing).(\neg r_2 \vee r_3)(\neg r_3 \vee e)(r_2 \vee \neg e)$, which has satisfying probability $\Pr[\Phi'_4] = 0.4 < \Pr[\Phi_3] = 0.5$.

[3]This case only occurs in DSSAT formulas.

4.3 Other Incorporated Techniques

In addition to the techniques described above, we note that all techniques that yield a logically equivalent matrix can be lifted directly to (D)SSAT.

Proposition 1. *If a preprocessing technique always yields a logically equivalent matrix, then it is sound for (D)SSAT.*

This includes SAT-based filters (Cadoli et al. 2002), rewriting existential variables, resolution-based variable elimination (Eén and Biere 2005), subsumption elimination (Biere, Heule, and van Maaren 2009), self-subsuming resolution (Eén and Biere 2005), hidden subsumption elimination (Heule, Järvisalo, and Biere 2010), vivification (Piette, Hamadi, and Sais 2008), etc. Interested readers may refer to (Wimmer, Scholl, and Becker 2019) for implementation details.

4.4 Universal Reduction

Universal reduction is one of the most powerful preprocessing techniques for (D)QBF. Its validity is stated in Lemma 1 (Wimmer, Scholl, and Becker 2019).

Lemma 1. *Let $\mathcal{Q}.\phi \wedge C$ be a DQBF and $l \in C$ a universal literal such that for all existential literal $k \in C$ we have $\mathrm{var}(l) \notin D_{\mathrm{var}(k)}$. Then $\mathcal{Q}.\phi \wedge C$ and $\mathcal{Q}.\phi \wedge (C \setminus \{l\})$ are equisatisfiable.*

It utilizes the fact that to falsify the formula, it suffices to falsify only one clause. Hence, if the clause C is not satisfied by any literal in $C \setminus \{l\}$, then by assigning l to $\bot$, the whole formula will be falsified.

Randomized literals, on the other hand, will be assigned to both polarities with a non-zero probability. Therefore, universal reduction cannot be generalized to (D)SSAT.

Theorem 6. *Universal reduction cannot be directly generalized for (D)SSAT. That is, let $\Phi = \mathcal{Q}.\phi \wedge C$ be a DSSAT formula and $l \in C$ a randomized literal such that for all existential literal $k \in C$ we have $\mathrm{var}(l) \notin D_{\mathrm{var}(k)}$. Let $\Phi' = \mathcal{Q}.\phi \wedge (C \setminus \{l\})$. Then, it is possible that* $\Pr[\Phi] \neq \Pr[\Phi']$.

Proof. As a counterexample, consider the DSSAT formula $\Phi = \text{Я}^{0.5} r_1.(r_1)$. Applying "randomized reduction" on Φ gives $\Phi' = \bot$. Clearly, $\Pr[\Phi] = 0.5 \neq \Pr[\Phi'] = 0$. $\square$

Moreover, we argue that it is unlikely to have a sound and efficient lifting of universal reduction in (D)QBF to randomized reduction in (D)SSAT even by adding some side conditions, because an SSAT formula with only randomized variables is essentially an instance of weighted model counting, which is #P-complete. Hence, any form of "randomized reduction" would substantially increase the formula size.

5 Experimental Results

Our implementation of the dependency elimination and preprocessing procedures, named DSSATpre,[4] is written in C++ under the framework of HQSpre (Wimmer, Scholl, and Becker 2019). Inherited from the structure of HQSpre, the dependency elimination in DSSATpre is interleaved with

[4]Available at https://github.com/NTU-ALComLab/DSSATpre.

			DSSAT (partial prep.)		DSSAT (full prep.)		DQBF	
Benchmark	Family	#Instance	#Sol	PAR2	#Sol	PAR2	#Sol	PAR2
Biere	adder_3_2	100	100	0.05	100	0.05	100	0.00
	adder_3_4	100	100	0.05	100	0.05	100	0.03
	adder_3_6	100	100	0.21	100	0.18	100	0.04
	bitcell_16_2	100	99	22.50	100	0.05	100	0.00
	bitcell_16_4	100	96	94.33	100	0.08	100	0.00
	bitcell_16_6	100	88	263.25	100	0.63	100	0.00
	lookahead_16_2	100	100	0.11	100	0.05	100	0.00
	lookahead_16_4	100	98	43.03	100	0.13	100	0.00
	lookahead_16_6	100	91	228.05	100	2.79	100	0.02
	pec_xor2	100	89	220.01	100	0.00	100	0.00
	pec_xor3	100	100	0.02	100	0.03	100	0.00
	pec_xor4	100	100	12.38	100	0.11	100	0.10
	Total	1200	1161	73.67	1200	0.35	1200	0.02
Scholl	c432	240	49	1600.98	45	1637.03	34	1722.49
	c432_nondisjoint	100	12	1775.67	16	1707.47	17	1664.52
	c499_nondisjoint	128	8	1880.47	8	1877.40	65	1002.30
	comp	240	87	1286.44	86	1293.18	148	779.52
	comp_nondisjoint	64	56	255.25	55	290.48	60	175.04
	term1_nondisjoint	44	44	0.34	44	0.21	44	8.99
	z4	240	240	0.13	237	25.14	240	0.80
	z4_nondisjoint	60	60	0.04	60	0.04	60	0.02
	Total	1116	556	1010.43	551	1020.56	668	812.74

Table 2: Experimental results of DSSAT solving using dependency elimination.

preprocessing, and is implemented as one of the optional preprocess options. Thus, unless otherwise specified, the experiments of dependency elimination were run with preprocessing fully enabled.

The experiments were conducted on a Linux machine with Intel Xeon Silver 4210 CPU processor at 2.2 GHz. All instances are subject to a time limit of 1000 seconds. The performance is judged based on the number of instances solved within the time limit and the penalized average runtime (PAR2) score, which penalizes unsolved instances with two times the time limit.

We conducted two experiments: The first experiment tests the applicability of solving DSSAT with dependency elimination. The second experiment aims to study the effectiveness of (D)SSAT preprocessing. We use DC-SSAT (Majercik and Boots 2005), ClauSSat (Chen, Huang, and Jiang 2021), and ElimSSAT (Wang et al. 2022) as the SSAT solvers. DC-SSAT adopts a CDCL-based solving scheme, ClauSSat lifts the abstraction-based technique *clause selection* from QBF, and ElimSSAT solves SSAT using *quantifier elimination*. Note that since ElimSSAT approximates non-0.5 probabilities with 4-bit precision, the performance between ElimSSAT and other solvers shall be compared with care. This does not affect the evaluation of our implementation.

5.1 Dependency Elimination

Since there are neither DSSAT benchmarks nor competing solvers in the literature, we conduct the experiment by taking DQBF benchmarks encoding partial equivalence checking (PEC) problem from (Fröhlich et al. 2014) and (Scholl and Becker 2001), and reinterpret them as the probabilistic partial design equivalence checking problem (Lee and Jiang 2018, 2021) by changing each universal variable into a randomized variable with probability 0.5. In the PEC problem, a completely specified design and a partially specified design that contains some black boxes are given. The task is to determine whether there exists some implementation for the black boxes such that both designs are functionally equivalent. In the DQBF encoding, the universal variables correspond to the primary inputs of the designs. When universal variables are reinterpreted as randomized ones associated with probability 0.5, we essentially find an implementation of the black boxes such that the two designs coincide on the largest proportion of the input assignments.

The benchmark set `Biere` consists of 12 families, each of 100 instances, and the benchmark set `Scholl` consists of 8 families, with a total number of 1116 instances. `Biere` is a benchmark set that is easy for modern DQBF solvers, while `Scholl` contains medium to hard instances.[5]

The results of DSSAT solving are shown in Table 2, where DSSAT solvings under partial and full preprocessing are compared along with the reference DQBF solving. Among the studied SSAT solvers, we report only the results of ElimSSAT in solving the SSAT formulas after dependency elimination of the translated DSSAT formulas due to the relatively good performance of ElimSSAT. As no competing DSSAT solver exists for comparison, we solve the original DQBF instances using the DQBF solver HQS (Wimmer

[5]The benchmarks are taken from https://github.com/jurajsic/DQBFbenchmarks.

		DC-SSAT				ClauSSat				ElimSSAT			
		w/o prep.		w/ prep.		w/o prep.		w/ prep.		w/o prep.		w/ prep.	
Family	#	#S	PAR2	#S	PAR2	#S	PAR2	#S	PAR2	#S	PAR2	#S	PAR2
pipesnotankage	9	0	2000.00	2	1555.56	0	2000.00	3	1334.69	2	1558.32	2	1555.57
MaxCount	25	4	1716.69	8	1360.17	8	1368.44	13	986.83	10	1219.29	9	1280.05
ev-pr-4x4	7	0	2000.00	7	44.43	1	1714.75	1	1716.24	1	1792.97	1	1715.20
Tiger	5	5	312.47	5	0.01	2	1218.31	5	1.35	5	0.58	5	0.01
stracomp	60	28	1082.50	32	971.14	60	54.39	60	37.56	44	591.04	44	583.57
depots	9	0	2000.00	0	2000.00	0	2000.00	0	2000.00	0	2000.00	0	2000.00
sand-castle	25	24	147.49	22	276.24	11	1166.99	11	1138.26	14	920.30	14	902.55
gttt_3x3	9	9	2.92	9	1.51	9	114.60	4	1167.78	0	2000.00	0	2000.00
RobotsD2	10	3	1404.59	4	1209.39	10	31.30	10	39.13	5	1099.46	5	1067.50
Tree	14	14	0.00	14	0.00	14	2.57	14	3.03	14	22.71	14	18.10
arbiter	10	0	2000.00	0	2000.00	0	2000.00	1	1800.13	0	2000.00	0	2000.00
MPEC	8	4	1126.35	4	1000.10	1	1750.01	2	1500.01	8	2.20	8	5.91
conformant	24	2	1872.74	2	1835.51	2	1849.03	2	1859.89	6	1519.18	6	1524.13
Connect2	16	16	0.24	16	14.67	10	801.28	8	1034.69	2	1871.15	13	430.51
ToiletA	77	42	946.37	62	480.57	46	888.27	52	698.95	77	36.73	74	112.08
PEC	8	0	2000.00	4	1000.32	3	1267.07	3	1250.14	7	413.23	6	561.98
k_branch_n	10	2	1602.80	6	800.25	1	1800.23	1	1800.05	2	1600.98	2	1600.13
k_ph_p	4	3	675.77	3	507.89	3	671.63	3	579.43	4	17.66	4	20.74
tlc	13	13	0.28	13	0.05	13	1.72	13	0.05	13	0.24	13	0.05
Adder	6	4	666.85	4	666.76	5	411.80	5	412.35	4	673.87	5	457.62
Counter	8	4	1015.48	8	167.90	4	1066.11	7	352.15	8	0.10	8	0.16
Total	357	177	1038.88	225	773.87	203	906.35	218	809.47	226	771.77	233	722.65

Table 3: Experimental results of SSAT preprocessing.

et al. 2017) as a reference. As the effect of preprocessing will be discussed in Section 5.2, we focus on comparing DSSAT solving under full preprocessing and DQBF solving. As can be seen from the table, despite the increased hardness brought by the stochastic nature of DSSAT, dependency elimination performs well on the benchmarks. With ElimSSAT, it solves all the instances from Biere and around half the instances from Scholl. In particular, HQS solves 164 SAT instances and 504 UNSAT ones from Scholl, whereas DSSATpre+ElimSSAT solves 226 SAT ones and 325 UNSAT ones. The fact that DSSATpre+ElimSSAT performs worse in UNSAT instances is expected since refuting a false DQBF is often computationally more manageable than the corresponding counting task, while knowing that a DQBF is true directly implies that the satisfying probability of the corresponding DSSAT formula is 1. Surprisingly, for the families c432 and term1_nondisjoint, DSSATpre+ElimSSAT outperforms HQS even though it is solving the harder DSSAT problem. Since HQS adopts the same dependency elimination procedure, the result may suggest that the quantifier elimination procedure in ElimSSAT works exceptionally well on these two families.

5.2 Preprocessing

In this section, we examine the effect of preprocessing on (D)SSAT solving. Since preprocessing is embedded into the dependency elimination procedure and no other DSSAT solver exists, we shall focus mainly on SSAT preprocessing.

Nonetheless, before discussing the results of SSAT preprocessing, we note that preprocessing is crucial to the success of DSSAT solving with dependency elimination. The results of DSSAT solving under partial preprocessing in Table 2 are obtained using DSSATpre with only unit propagation, pure literal elimination, and equivalent literals elimination being enabled.[6] Under this setting, 39 instances from the family Biere can no longer be solved within the time limit, while some Scholl instances unsolved under full preprocessing become solvable and some solved under full preprocessing become unsolvable. These differences suggest that preprocessing plays a nontrivial role in DSSAT solving, and the effect shall be further studied when more DSSAT solving techniques are available in the future.

To evaluate the effectiveness of preprocessing on SSAT instances, we took the benchmarks of (Chen, Huang, and Jiang 2021) for evaluation.[7] We compare the performance of SSAT solvers DC-SSAT, ClauSSat and ElimSSAT on the original and preprocessed instances to examine the power of SSAT preprocessing. The results are summarized in Table 3. We note that 30 out of the 357 instances were solved during preprocessing. From the table, we see that DC-SSAT benefits from preprocessing consistently across all families, with a reduction of $25.5\,\%$ in the PAR2 score. On the other hand, ClauSSat and ElimSSAT seem to benefit slightly from preprocessing in most families, but drastically slowed down on certain families, such as gttt_3x3 for ClauSSat

[6]These three techniques are embedded into the dependency elimination procedure and cannot be turned off.

[7]The benchmarks are taken from https://github.com/NTU-ALComLab/ClauSSat.

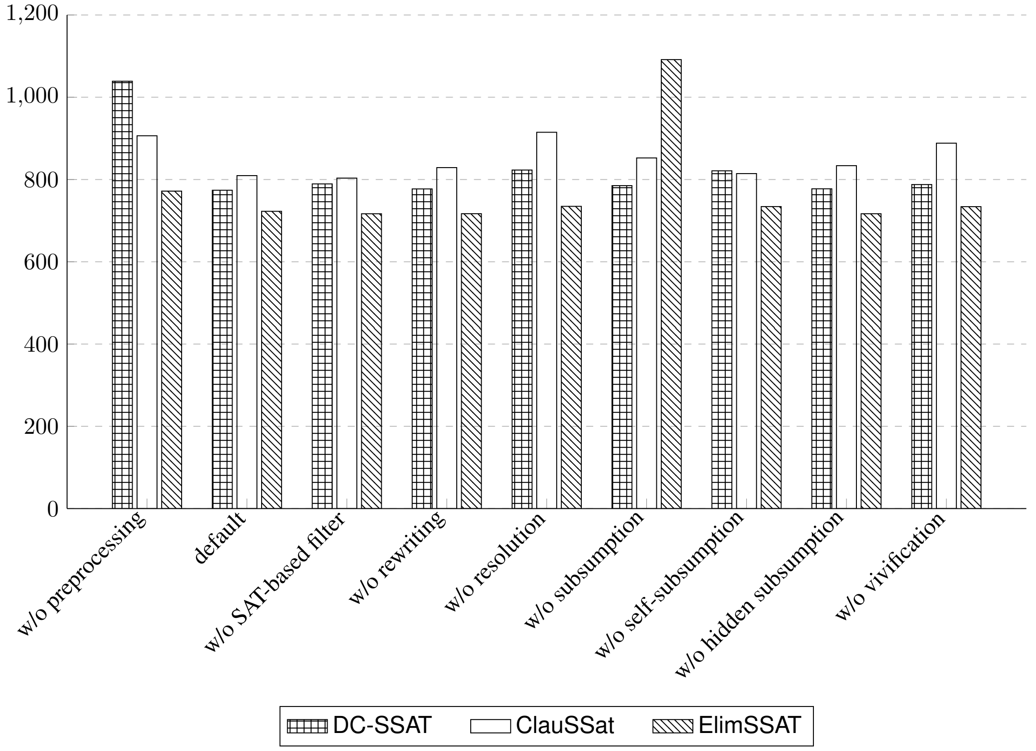

Figure 1: Ablation study of SSAT preprocessing.

and `ToiletA` for ElimSSAT. The reason behind such a phenomenon requires further investigation.

We performed an ablation study on the techniques listed in Table 1 to further examine the effect of different techniques on the three solvers. The results are shown in Figure 1, where the group "w/o preprocessing" shows the results without preprocessing, "default" shows the results with all preprocessing techniques enabled, and each of the other groups shows the results of default but with one technique being disabled. It can be seen that DC-SSAT performs roughly the same among all configurations with preprocessing. On the other hand, ClauSSat performs worse when resolution is turned off, and ElimSSAT performs terribly when subsumption is turned off. The results suggest that ClauSSat might benefit from incorporating resolution-like techniques into its procedure, and, similarly, ElimSSAT may benefit from subsumption-based techniques. Additionally, we learn that different solvers may benefit from different techniques, and the optimal configuration may vary from solver to solver. Finally, while the performance of each solver may improve slightly when a certain technique is turned off, the default setting seems to fit the three solvers well.

Overall, the experiments show that SSAT preprocessing is powerful and that state-of-the-art SSAT solvers may benefit from preprocessing.

6 Conclusions and Future Work

This paper has introduced the first DSSAT solver by lifting the dependency elimination technique for DQBF (Wimmer et al. 2017) and developed the first standalone preprocessor for SSAT and DSSAT by modifying the (D)QBF preprocessor HQSpre. Experimental results have shown the feasibility of DSSAT solving with dependency elimination and demonstrated the effectiveness of preprocessing in SSAT formula solving by DC-SSAT.

For future work, we plan to develop new methods for DSSAT solving besides dependency elimination and study the effects of DSSAT preprocessing on different DSSAT solvers.

Acknowledgements

This work was supported in part by the National Science and Technology Council of Taiwan under Grant 111-2221-E-002-182 and Grant 111-2923-E-002-013-MY3.

References

Biere, A.; Heule, M.; and van Maaren, H. 2009. *Handbook of Satisfiability*, volume 185. IOS press.

Biere, A.; Lonsing, F.; and Seidl, M. 2011. Blocked Clause Elimination for QBF. In *Proceedings of International Conference on Automated Deduction (CADE)*, 101–115.

Cadoli, M.; Schaerf, M.; Giovanardi, A.; and Giovanardi, M. 2002. An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation. *Journal of Automated Reasoning*, 28(2): 101–142.

Chen, P.-W.; Huang, Y.-C.; and Jiang, J.-H. R. 2021. A Sharp Leap from Quantified Boolean Formula to Stochastic Boolean Satisfiability Solving. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 3697–3706.

Eén, N.; and Biere, A. 2005. Effective Preprocessing in SAT Through Variable and Clause Elimination. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 61–75.

Fröhlich, A.; Kovásznai, G.; Biere, A.; and Veith, H. 2014. iDQ: Instantiation-Based DQBF Solving. In *Proceedings of the Pragmatics of SAT workshop (POS)*, volume 27 of *EPiC Series in Computing*, 103–116.

Giunchiglia, E.; Marin, P.; and Narizzano, M. 2010. sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 85–98.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2009. Model Counting. In *Handbook of Satisfiability*, 633–654. IOS Press.

Henkin, L.; and Karp, C. R. 1965. Some Remarks on Infinitely Long Formulas. *Journal of Symbolic Logic*, 30(1): 96–97.

Heule, M.; Järvisalo, M.; and Biere, A. 2010. Clause Elimination Procedures for CNF Formulas. In *Proceedings of International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, 357–371.

Hnich, B.; Rossi, R.; Tarim, S. A.; and Prestwich, S. 2011. A Survey on CP-AI-OR Hybrids for Decision Making Under Uncertainty. In *Hybrid Optimization: The Ten Years of CPAIOR*, 227–270. Springer New York.

Janota, M.; and Marques-Silva, J. 2015. Solving QBF by Clause Selection. In *Proceedings of International Conference on Artificial Intelligence (IJCAI)*, 325–331.

Kleine Büning, H.; and Bubeck, U. 2009. Theory of Quantified Boolean Formulas. In *Handbook of satisfiability*, 735–760. IOS Press.

Lagniez, J.-M.; Lonca, E.; and Marquis, P. 2016. Improving Model Counting by Leveraging Definability. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 751–757.

Lagniez, J.-M.; and Marquis, P. 2014. Preprocessing for Propositional Model Counting. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2688–2694.

Lee, N.-Z.; and Jiang, J.-H. R. 2018. Towards Formal Evaluation and Verification of Probabilistic Design. *IEEE Transactions on Computers*, 67(8): 1202–1216.

Lee, N.-Z.; and Jiang, J.-H. R. 2021. Dependency Stochastic Boolean Satisfiability: A Logical Formalism for NEXPTIME Decision Problems with Uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 3877–3885.

Li, C. M.; and Manya, F. 2021. MaxSAT, Hard and Soft Constraints. In *Handbook of satisfiability*, 903–927. IOS Press.

Littman, M. L.; Majercik, S. M.; and Pitassi, T. 2001. Stochastic Boolean Satisfiability. *Journal of Automated Reasoning*, 27(3): 251–296.

Majercik, S. M.; and Boots, B. 2005. DC-SSAT: A Divide-and-Conquer Approach to Solving Stochastic Satisfiability Problems Efficiently. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 416–422.

Manthey, N. 2012. Coprocessor 2.0 – A Flexible CNF Simplifier. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 436–441.

Nilsson, N. J. 1982. *Principles of Artificial Intelligence*. Springer Science & Business Media.

Papadimitriou, C. H. 1985. Games against Nature. *Journal of Computer and System Sciences*, 31(2): 288–301.

Peterson, G.; Reif, J.; and Azhar, S. 2001. Lower Bounds for Multiplayer Noncooperative Games of Incomplete Information. *Computers & Mathematics with Applications*, 41(7): 957–992.

Piette, C.; Hamadi, Y.; and Sais, L. 2008. Vivifying Propositional Clausal Formulae. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 525–529.

Safarpour, S.; Mangassarian, H.; Veneris, A.; Liffiton, M. H.; and Sakallah, K. A. 2007. Improved Design Debugging Using Maximum Satisfiability. In *Proceedings of Formal Methods in Computer Aided Design (FMCAD)*, 13–19.

Sang, T.; Bearne, P.; and Kautz, H. 2005. Performing Bayesian Inference by Weighted Model Counting. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 475–481.

Scholl, C.; and Becker, B. 2001. Checking Equivalence for Partial Implementations. In *Proceedings of Design Automation Conference (DAC)*, 238–243.

Scholl, C.; and Wimmer, R. 2018. Dependency Quantified Boolean Formulas: An Overview of Solution Methods and Applications. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 3–16.

Vizel, Y.; Weissenbacher, G.; and Malik, S. 2015. Boolean Satisfiability Solvers and Their Applications in Model Checking. *Proceedings of the IEEE*, 103(11): 2021–2035.

Wang, H.-R.; Tu, K.-H.; Jiang, J.-H. R.; and Scholl, C. 2022. Quantifier Elimination in Stochastic Boolean Satisfiability. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 23:1–23:17.

Wimmer, R.; Karrenbauer, A.; Becker, R.; Scholl, C.; and Becker, B. 2017. From DQBF to QBF by Dependency Elimination. In *Proceedings of International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 326–343.

Wimmer, R.; Scholl, C.; and Becker, B. 2019. The (D)QBF Preprocessor HQSpre – Underlying Theory and Its Implementation. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1): 3–52.