Dependency Stochastic Boolean Satisfiability: A Logical Formalism for NEXPTIME Decision Problems with Uncertainty

Nian-Ze Lee,[1] **Jie-Hong R. Jiang**[1, 2]

[1] Graduate Institute of Electronics Engineering, National Taiwan University
[2] Department of Electrical Engineering, National Taiwan University
No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
{d04943019, jhjiang}@ntu.edu.tw

Abstract

Stochastic Boolean Satisfiability (SSAT) is a logical formalism to model decision problems with uncertainty, such as *Partially Observable Markov Decision Process* (POMDP) for verification of probabilistic systems. SSAT, however, is limited by its descriptive power within the PSPACE complexity class. More complex problems, such as the NEXPTIME-complete *Decentralized POMDP* (Dec-POMDP), cannot be succinctly encoded with SSAT. To provide a logical formalism of such problems, we extend the *Dependency Quantified Boolean Formula* (DQBF), a representative problem in the NEXPTIME-complete class, to its stochastic variant, named *Dependency SSAT* (DSSAT), and show that DSSAT is also NEXPTIME-complete. We demonstrate the potential applications of DSSAT to circuit synthesis of probabilistic and approximate design. Furthermore, to study the descriptive power of DSSAT, we establish a polynomial-time reduction from Dec-POMDP to DSSAT. With the theoretical foundations paved in this work, we hope to encourage the development of DSSAT solvers for potential broad applications.

1 Introduction

Satisfiability (SAT) solvers (Biere, Heule, and van Maaren 2009) have been successfully applied to numerous research fields including artificial intelligence (Nilsson 2014; Russell and Norvig 2016), electronic design automation (Marques-Silva and Sakallah 2000; Wang, Chang, and Cheng 2009), software verification (Bérard et al. 2013; Jhala and Majumdar 2009), etc. The tremendous benefits have encouraged the development of more advanced decision procedures for satisfiability with respect to more complex logics beyond pure propositional. For example, solvers of the satisfiability modulo theories (SMT) (De Moura and Bjørner 2011; Barrett and Tinelli 2018) accommodate first order logic fragments; quantified Boolean formula (QBF) (Narizzano, Pulina, and Tacchella 2006; Büning and Bubeck 2009) allows both existential and universal quantifiers; stochastic Boolean satisfiabilty (SSAT) (Littman, Majercik, and Pitassi 2001; Majercik 2009) models uncertainty with random quantification; and dependency QBF (DQBF) (Balabanov, Chiang, and Jiang 2014; Scholl and Wimmer 2018) equips Henkin quantifiers to describe multi-player games with partial information. Due to their simplicity and generality, various satisfiability formulations are under active investigation.

Among the quantified decision procedures, QBF and SSAT are closely related. While SSAT extends QBF to allow random quantifiers to model uncertainty, they are both PSPACE-complete (Stockmeyer and Meyer 1973). A number of SSAT solvers have been developed and applied in probabilistic planning, formal verification of probabilistic design, partially observable Markov decision process (POMDP), and analysis of software security. For example, solver `MAXPLAN` (Majercik and Littman 1998) encodes a conformant planning problem as an exist-random quantified SSAT formula; solver `ZANDER` (Majercik and Littman 2003) deals with partially observable probabilistic planning by formulating the problem as a general SSAT formula; solver `DC-SSAT` (Majercik and Boots 2005) relies on a divide-and-conquer approach to speedup the solving of a general SSAT formula. Solvers `ressat` and `erssat` (Lee, Wang, and Jiang 2017, 2018) are developed for random-exist and exist-random quantified SSAT formulas respectively, and applied to the formal verification of probabilistic design (Lee and Jiang 2018). POMDP has also been studied under the formalism of SSAT (Majercik and Littman 2003; Salmon and Poupart 2019). Recently, bi-directional polynomial-time reductions between SSAT and POMDP are established (Salmon and Poupart 2019). The quantitative information flow analysis for software security is also investigated as an exist-random quantified SSAT formula (Fremont, Rabe, and Seshia 2017).

In view of the close relation between QBF and SSAT, we raise the question what would be the formalism that extends DQBF to the stochastic domain. We formalize the *dependency SSAT* (DSSAT) as the answer to the question. We prove that DSSAT has the same NEXPTIME-complete complexity as DQBF (Peterson, Reif, and Azhar 2001), and therefore it can succinctly encode decision problems with uncertainty in the NEXPTIME complexity class.

To highlight the benefits of DSSAT over DQBF, we note that DSSAT intrinsically represents an optimization problem (the answer is the maximum satisfying probability) while DQBF is a decision problem (the answer is either true or false). The optimization nature of DSSAT potentially allows broader applications of the formalism. Moreover, DSSAT is often preferable to DQBF in expressing problems involving

uncertainty and probabilities. As case studies, we investigate its applicability in probabilistic system design/verification and artificial intelligence.

In system design of the post Moore's law era, the practice of *very large scale integration* (VLSI) circuit design experiences a paradigm shift in design principles to overcome the obstacle of physical scaling of computation capacity. Probabilistic design (Chakrapani et al. 2008) and approximate design (Venkatesan et al. 2011) are two such examples of emerging design methodologies. The former does not require logic gates to be error-free, but rather allowing them to function with probabilistic errors. The latter does not require the implementation circuit to behave exactly the same as the specification, but rather allowing their deviation to some extent. These relaxations to design requirements provide freedom for circuit simplification and optimization. We show that DSSAT can be a useful tool for the analysis of probabilistic design and approximate design.

The theory and applications of Markov decision process and its variants are among the most important topics in the study of artificial intelligence. For example, the decision problem involving multiple agents with uncertainty and partial information is often considered as a decentralized POMDP (Dec-POMDP) (Oliehoek, Amato et al. 2016). The independent actions and observations of the individual agents make POMDP for single-agent systems not applicable and require the more complex Dec-POMDP. Essentially the complexity is lifted from the PSPACE-complete policy evaluation of finite-horizon POMDP to the NEXPTIME-complete Dec-POMDP. We show that Dec-POMDP is polynomial time reducible to DSSAT.

To sum up, the main results of this work include:

- formulating the DSSAT problem (Section 3),
- proving its NEXPTIME-completeness (Section 4), and
- showing its applications in:
  - analyzing probabilistic/approximate design (Section 5)
  - modeling Dec-POMDP (Section 6).

Our results may encourage the development of DSSAT solvers to enable potential broad applications.

2 Preliminaries

In this section, we provide background knowledge about SSAT, DQBF, probabilistic design, and Dec-POMDP.

In the sequel, Boolean values TRUE and FALSE are represented by symbols $\top$ and $\bot$, respectively; they are also treated as 1 and 0, respectively, in arithmetic computation. Boolean connectives $\neg, \vee, \wedge, \Rightarrow, \equiv$ are interpreted in their conventional semantics. Given a set V of variables, an *assignment* α is a mapping from each variable $x \in V$ to $\mathbb{B} = \{\top, \bot\}$, and we denote the set of all assignments over V by $\mathcal{A}(V)$. An assignment α *satisfies* a Boolean formula ϕ over a set V of variables if ϕ yields $\top$ after substituting all occurrences of every variable $x \in V$ with its assigned value $\alpha(x)$ and simplifying ϕ under the semantics of Boolean connectives. A Boolean formula ϕ over a set V of variables is a *tautology* if every assignment $\alpha \in \mathcal{A}(V)$ satisfies ϕ.

2.1 Stochastic Boolean Satisfiability

SSAT was first proposed by Papadimitriou and described as *games against nature* (Papadimitriou 1985). An SSAT formula Φ over a set $V = \{x_1, \ldots, x_n\}$ of variables is of the form: $Q_1 x_1, \ldots, Q_n x_n.\phi$, where each $Q_i \in \{\exists, \mathrm{R}^p\}$ and Boolean formula ϕ over V is quantifier-free. Symbol $\exists$ denotes an existential quantifier, and R^p denotes a randomized quantifier, which requires the probability that the quantified variable equals $\top$ to be $p \in [0, 1]$. Given an SSAT formula Φ, the quantification structure $Q_1 x_1, \ldots, Q_n x_n$ is called the *prefix*, and the quantifier-free Boolean formula ϕ is called the *matrix*.

Let x be the outermost variable in the prefix of an SSAT formula Φ. The satisfying probability of Φ, denoted by $\Pr[\Phi]$, is defined recursively by the following four rules:

a) $\Pr[\top] = 1$,

b) $\Pr[\bot] = 0$,

c) $\Pr[\Phi] = \max\{\Pr[\Phi|_{\neg x}], \Pr[\Phi|_x]\}$, if x is existentially quantified,

d) $\Pr[\Phi] = (1-p)\Pr[\Phi|_{\neg x}] + p\Pr[\Phi|_x]$, if x is randomly quantified by R^p,

where $\Phi|_{\neg x}$ and $\Phi|_x$ denote the SSAT formulas obtained by eliminating the outermost quantifier of x via substituting the value of x in the matrix with $\bot$ and $\top$, respectively.

The *decision version* of SSAT is stated as follows. Given an SSAT formula Φ and a threshold $\theta \in [0, 1]$, decide whether $\Pr[\Phi] \geq \theta$. On the other hand, the *optimization version* asks to compute $\Pr[\Phi]$. The decision version of SSAT was shown to be PSPACE-complete (Papadimitriou 1985).

2.2 Dependency Quantified Boolean Formula

DQBF was formulated as *multiple-person alternation* (Peterson and Reif 1979). In contrast to the *linearly ordered* prefix used in QBF, i.e., an existentially quantified variable will depend on all of its preceding universally quantified variables, the quantification structure in DQBF is extended with Henkin quantifiers, where the dependency of an existentially quantified variable on the universally quantified variables can be explicitly specified.

A DQBF Φ over a set $V = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ of variables is of the form:

$$\forall x_1, \ldots, \forall x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi, \tag{1}$$

where each $D_{y_j} \subseteq \{x_1, \ldots, x_n\}$ denotes the set of variables that variable y_j can depend on, and Boolean formula ϕ over V is quantifier-free. We denote the set $\{x_1, \ldots, x_n\}$ (resp. $\{y_1, \ldots, y_m\}$) of universally (resp. existentially) quantified variables of Φ by $V_\Phi^\forall$ (resp. $V_\Phi^\exists$).

Given a DQBF Φ, it is satisfied if for each variable y_j, there exists a function $f_j : \mathcal{A}(D_{y_j}) \to \mathbb{B}$, such that after substituting variables in $V_\Phi^\exists$ with their corresponding functions respectively, matrix ϕ yields a tautology over $V_\Phi^\forall$. The set of functions $\mathcal{F} = \{f_1, \ldots, f_m\}$ is called a set of *Skolem* functions for Φ. In other words, Φ is satisfied by $\mathcal{F}$ if

$$\min_{\beta \in \mathcal{A}(V_\Phi^\forall)} \mathbb{1}_{\phi|_\mathcal{F}}(\beta) = 1, \tag{2}$$

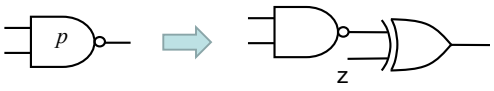

Figure 1: Conversion of the distillation operation.

where $\mathbb{1}_{\phi|_{\mathcal{F}}}(\cdot)$ is an indicator function to indicate whether an assignment over $V_\Phi^\forall$ belongs to the set of satisfying assignments of matrix ϕ, when variables in $V_\Phi^\exists$ are substituted by their Skolem functions in $\mathcal{F}$. That is, $\phi|_{\mathcal{F}} = \{\beta \mid \phi(\beta(x_1), \ldots, \beta(x_n), f_1|_\beta, \ldots, f_m|_\beta) \equiv \top\}$, where $f_j|_\beta$ is the logical value derived by substituting every $x_i \in D_{y_j}$ with $\beta(x_i)$ in function f_j. The satisfiability problem of DQBF was shown to be NEXPTIME-complete (Peterson, Reif, and Azhar 2001).

2.3 Probabilistic Design

In this paper, a *design* refers to a combinational Boolean logic circuit, which is a directed acyclic graph $G = (V, E)$, where V is a set of vertices, and $E \subseteq V \times V$ is a set of edges. Each vertex in V can be a primary input, primary output, or an intermediate gate. An intermediate gate is associated with a Boolean function. An edge $(u, v) \in E$ signifies the connection from u to v, denoting the associated Boolean function of v may depend on u. A circuit is called a *partial design* if some of the intermediate gates are black boxes, that is, their associated Boolean functions are not specified.

A *probabilistic design* is an extension of conventional Boolean logic circuits to model the scenario where intermediate gates exhibit probabilistic behavior. In a probabilistic design, each intermediate gate has an *error rate*, i.e., the probability for the gate to produce an erroneous output. An intermediate gate is *erroneous* if its error rate is nonzero. Using the *distillation operation* (Lee and Jiang 2018), an erroneous gate can be modeled by its corresponding error-free gate XORed with an auxiliary input, which valuates to $\top$ with a probability equal to the error rate. As illustrated in Figure 1, a NAND gate with error rate p is converted to an error-free NAND gate XORed with a fresh auxiliary input z with $\Pr[z = \top] = p$ so that it triggers the error with probability p. After applying the distillation operation to every erroneous gate, all the intermediate gates in the distilled design become error-free, which makes the techniques for conventional Boolean circuit reasoning applicable.

2.4 Decentralized POMDP

Dec-POMDP is a formalism for multi-agent systems under uncertainty and with partial information. Its computational complexity was shown to be NEXPTIME-complete (Bernstein et al. 2002). In the following, we briefly review the definition, optimality criteria, and value function of Dec-POMDP from the literature (Oliehoek, Amato et al. 2016).

A Dec-POMDP is specified by a tuple $\mathcal{M} = (I, S, \{A_i\}, T, \rho, \{O_i\}, \Omega, \Delta_0, h)$, where $I = \{1, \ldots, n\}$ is a finite set of n agents, S is a finite set of states, A_i is a finite set of actions of Agent i, $T : S \times (A_1 \times \cdots \times A_n) \times S \to [0, 1]$ is a transition distribution function with $T(s, \vec{a}, s') = \Pr[s'|s, \vec{a}]$, the probability to transit to state s' from state s after taking actions $\vec{a}$, $\rho : S \times (A_1 \times \cdots \times A_n) \to \mathbb{R}$ is a reward function with $\rho(s, \vec{a})$ giving the reward for being in state s and taking actions $\vec{a}$, O_i is a finite set of observations for Agent i, $\Omega : S \times (A_1 \times \cdots \times A_n) \times (O_1 \times \cdots \times O_n) \to [0, 1]$ is an observation distribution function with $\Omega(s', \vec{a}, \vec{o}) = \Pr[\vec{o}|s', \vec{a}]$, the probability to receive observation $\vec{o}$ after taking actions $\vec{a}$ and transiting to state s', $\Delta_0 : S \to [0, 1]$ is an initial state distribution function with $\Delta_0(s) = \Pr[s^0 \equiv s]$, the probability for the initial state s^0 being state s, and h is a planning horizon, which we assume finite in this work.

Given a Dec-POMDP $\mathcal{M}$, we aim at maximizing the expected total reward $\mathbb{E}[\sum_{t=0}^{h-1} \rho(s^t, \vec{a}^t)]$ through searching an optimal *joint policy* for the team of agents. Specifically, a *policy* π_i of Agent i is a mapping from the agent's *observation history*, i.e., a sequence of observations $\underline{o_i^t} = o_i^0, \ldots, o_i^t$ received by Agent i, to an action $a_i^{t+1} \in A_i$. A joint policy for the team of agents $\vec{\pi} = (\pi_1, \ldots, \pi_n)$ maps the agents' joint observation history $\underline{\vec{o}^t} = (\underline{o_1^t}, \ldots, \underline{o_n^t})$ to actions $\vec{a}^{t+1} = (\pi_1(\underline{o_1^t}), \ldots, \pi_n(\underline{o_n^t}))$. We shall focus on deterministic policies only, as it was shown that every Dec-POMDP with a finite planning horizon has a deterministic optimal joint policy (Oliehoek, Spaan, and Vlassis 2008).

To assess the quality of a joint policy $\vec{\pi}$, its *value* is defined to be $\mathbb{E}[\sum_{t=0}^{h-1} \rho(s^t, \vec{a}^t)|\Delta_0, \vec{\pi}]$. The *value function* $V(\vec{\pi})$ can be computed in a recursive manner, where for $t = h - 1$, $V^\pi(s^{h-1}, \underline{\vec{o}^{h-2}}) = \rho(s^{h-1}, \vec{\pi}(\underline{\vec{o}^{h-2}}))$, and for $t < h - 1$,

$$V^\pi(s^t, \underline{\vec{o}^{t-1}}) = \rho(s^t, \vec{\pi}(\underline{\vec{o}^{t-1}})) + \sum_{s^{t+1} \in S} \sum_{\vec{o}^t \in \vec{O}} \Pr[s^{t+1}, \vec{o}^t | s^t, \vec{\pi}(\underline{\vec{o}^t})] V^\pi(s^{t+1}, \underline{\vec{o}^t}). \quad (3)$$

The probability $\Pr[s^{t+1}, \vec{o}^t | s^t, \vec{\pi}(\underline{\vec{o}^t})]$ in the above equation is the product of $T(s^t, \vec{\pi}(\underline{\vec{o}^t}), s^{t+1})$ and $\Omega(s^{t+1}, \vec{\pi}(\underline{\vec{o}^t}), \vec{o}^t)$. Eq. (3) is also called the *Bellman Equation* of Dec-POMDP.

Denoting the empty observation history at the first stage (i.e., $t = 0$) with the symbol $\underline{\vec{o}^{-1}}$, the value $V(\vec{\pi})$ of a joint policy equals $\sum_{s^0 \in S} \Delta_0(s^0) V^\pi(s^0, \underline{\vec{o}^{-1}})$.

3 DSSAT Formulation

In this section, we extend DQBF to its stochastic variant, named *Dependency Stochastic Boolean Satisfiability* (DSSAT).

A DSSAT formula Φ over $V = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ is of the form:

$$\text{Ⅎ}^{p_1} x_1, \ldots, \text{Ⅎ}^{p_n} x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi, \quad (4)$$

where each $D_{y_j} \subseteq \{x_1, \ldots, x_n\}$ denotes the set of variables that variable y_j can depend on, and Boolean formula ϕ over V is quantifier-free. We denote the set $\{x_1, \ldots, x_n\}$ (resp. $\{y_1, \ldots, y_m\}$) of randomly (resp. existentially) quantified variables of Φ by $V_\Phi^{\text{Ⅎ}}$ (resp. $V_\Phi^\exists$).

Given a DSSAT formula Φ and a set of Skolem functions $\mathcal{F} = \{f_j : \mathcal{A}(D_{y_j}) \to \mathbb{B} \mid j = 1, \ldots, m\}$, the satisfying

probability $\Pr[\Phi|_{\mathcal{F}}]$ of Φ with respect to $\mathcal{F}$ is defined by the following equation:

$$\Pr[\Phi|_{\mathcal{F}}] = \sum_{\alpha \in \mathcal{A}(V_{\Phi}^{\forall})} \mathbb{1}_{\phi|_{\mathcal{F}}}(\alpha) w(\alpha), \quad (5)$$

where $\mathbb{1}_{\phi|_{\mathcal{F}}}(\cdot)$ is the indicator function defined in Section 2.2 and $w(\alpha) = \prod_{i=1}^{n} p_i^{\alpha(x_i)}(1-p_i)^{1-\alpha(x_i)}$ is the weighting function for assignments. In other words, the satisfying probability is the summation of weights of satisfying assignments over $V_{\Phi}^{\forall}$. The weight of an assignment can be understood as its occurring probability in the space of $\mathcal{A}(V_{\Phi}^{\forall})$.

The *decision version* of DSSAT is stated as follows. Given a DSSAT formula Φ and a threshold $\theta \in [0,1]$, decide whether there exists a set of Skolem functions $\mathcal{F}$ such that $\Pr[\Phi|_{\mathcal{F}}] \geq \theta$. On the other hand, the *optimization version* asks to find a set of Skolem functions to maximize the satisfying probability of Φ.

The formulation of SSAT can be extended by incorporating universal quantifiers, resulting in a unified framework named *extended* SSAT (Majercik 2009), which subsumes both QBF and SSAT. In the extended SSAT, besides the four rules discussed in Section 2.1 for calculating the satisfying probability of an SSAT formula Φ, the following rule is added: $\Pr[\Phi] = \min\{\Pr[\Phi|_{\neg x}], \Pr[\Phi|_{x}]\}$, if x is universally quantified. Similarly, an *extended* DSSAT formula Φ over a set of variables $\{x_1, \ldots, x_n, y_1, \ldots, y_m, z_1, \ldots, z_l\}$ is of the form:

$$Q_1 v_1, \ldots, Q_{n+l} v_{n+l}, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi, \quad (6)$$

where $Q_i v_i$ equals either $\forall^{p_k} x_k$ or $\forall z_k$ for some k with $v_i \neq v_j$ for $i \neq j$, and each $D_{y_j} \subseteq \{x_1, \ldots, x_n, z_1, \ldots, z_l\}$ denotes the set of randomly and universally quantified variables which variable y_j can depend on. The satisfying probability of Φ with respect to a set of Skolem functions $\mathcal{F} = \{f_j : \mathcal{A}(D_{y_j}) \to \mathbb{B} \mid j = 1, \ldots, m\}$, denoted by $\Pr[\Phi|_{\mathcal{F}}]$, can be computed by recursively applying the aforementioned five rules to the induced formula of Φ with the existential variables y_j being substituted with their respective Skolem functions f_j. Under the above computation scheme, both Eq. (2) and Eq. (5) are special cases, where the variables preceding the existential quantifiers in the prefixes are solely universally or randomly quantified, and hence the fifth or the fourth rule is applied to calculate $\Pr[\Phi|_{\mathcal{F}}]$.

Note that in the above extension the Henkin-type quantifiers are only defined for the existential variables. Although the extended formulation increases practical expressive succinctness, the computational complexity is not changed as to be shown in the next section.

4 DSSAT Complexity

In the following, we show that the decision version of DSSAT is NEXPTIME-complete.

Theorem 1. *DSSAT is NEXPTIME-complete.*

Proof. To show that DSSAT is NEXPTIME-complete, we have to show that it belongs to the NEXPTIME complexity class and that it is NEXPTIME-hard.

First, to see why DSSAT belongs to the NEXPTIME complexity class, observe that a Skolem function for an existentially quantified variable can be guessed and constructed in nondeterministic exponential time with respect to the number of randomly quantified variables. Given the guessed Skolem functions, the evaluation of the matrix, summation of weights of satisfying assignments, and comparison against the threshold θ can also be performed in exponential time. Overall, the whole procedure is done in nondeterministic exponential time with respect to the input size, and hence DSSAT belongs to the NEXPTIME complexity class.

Second, to see why DSSAT is NEXPTIME-hard, we reduce the NEXPTIME-complete problem DQBF to DSSAT as follows. Given an arbitrary DQBF:

$$\Phi_Q = \forall x_1, \ldots, \forall x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi,$$

we construct a DSSAT formula:

$$\Phi_S = \forall^{0.5} x_1, \ldots, \forall^{0.5} x_n, \exists y_1(D_{y_1}), \ldots, \exists y_m(D_{y_m}).\phi$$

by changing every universal quantifier to a randomized quantifier with probability 0.5. The reduction can be done in polynomial time with respect to the size of Φ_Q. We will show that Φ_Q is satisfiable if and only if there exists a set of Skolem functions $\mathcal{F}$ such that $\Pr[\Phi_S|_{\mathcal{F}}] \geq 1$.

The "only if" direction: As Φ_Q is satisfiable, there exists a set of Skolem functions $\mathcal{F}$ such that after substituting the existentially quantified variables with the corresponding Skolem functions, matrix ϕ becomes a tautology over variables $\{x_1, \ldots, x_n\}$. Therefore, $\Pr[\Phi_S|_{\mathcal{F}}] = 1 \geq 1$.

The "if" direction: As there exists a set of Skolem functions $\mathcal{F}$ such that $\Pr[\Phi_S|_{\mathcal{F}}] \geq 1$, after substituting the existentially quantified variables with the corresponding Skolem functions, each assignment $\alpha \in \mathcal{A}(\{x_1, \ldots, x_n\})$ must satisfy ϕ, i.e., ϕ becomes a tautology over variables $\{x_1, \ldots, x_n\}$. Otherwise, the satisfying probability $\Pr[\Phi_S|_{\mathcal{F}}]$ must be less than 1 as the weight 2^{-n} of some unsatisfying assignment is missing from the summation. Therefore, Φ_Q is satisfiable. □

When DSSAT is extended with universal quantifiers, its complexity remains in the NEXPTIME complexity class as the fifth rule of the satisfying probability calculation does not incur any complexity overhead. Therefore the following corollary is immediate.

Corollary 1. *The decision problem of DSSAT extended with universal quantifiers of Eq.* (6) *is NEXPTIME-complete.*

5 Application: Analyzing Probabilistic and Approximate Partial Design

After formulating DSSAT and proving its NEXPTIME-completeness, we show its application to the analysis of probabilistic design and approximate design. Specifically, we consider the probabilistic version of the *topologically constrained logic synthesis problem* (Sinha, Mishchenko, and Brayton 2002; Balabanov, Chiang, and Jiang 2014), or equivalently the *partial design problem* (Gitina et al. 2013).

In the *(deterministic) partial design problem*, we are given a specification function $G(X)$ over primary input variables

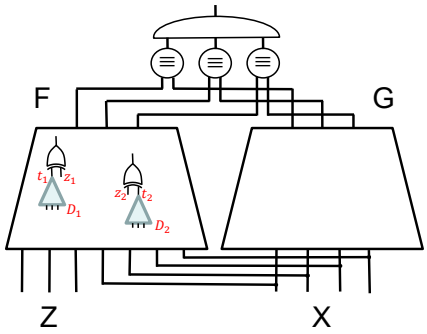

Figure 2: Circuit for the equivalence checking of probabilistic partial design.

X and a *partial design* C_F with black boxes to be synthesized. The Boolean functions induced at the primary outputs of C_F can be described by $F(X, T)$, where T corresponds to the variables of the black box outputs. Each black box output t_i is specified with its input variables (i.e., dependency set) $D_i \subseteq X \cup Y$ in C_F, where Y represents the variables for intermediate gates in C_F referred to by the black boxes. The partial design problem aims at deriving the black box functions $\{h_1(D_1), \ldots, h_{|T|}(D_{|T|})\}$ such that substituting t_i with h_i in C_F makes the resultant circuit function equal $G(X)$. The above partial design problem can be encoded as a DQBF problem; moreover, the partial equivalence checking problem is shown to be NEXPTIME-complete (Gitina et al. 2013).

Specifically, the DQBF that encodes the partial equivalence checking problem is of the form:

$$\forall X, \forall Y, \exists T(D).$$
$$(Y \equiv E(X)) \rightarrow (F(X, T) \equiv G(X)), \tag{7}$$

where D consists of $(D_1, \ldots, D_{|T|})$, E corresponds to the defining functions of Y in C_F, and the operator "$\equiv$" denotes elementwise equivalence between its two operands.

The above partial design problem can be extended to its probabilistic variant, which is illustrated by the circuit shown in Figure 2. The *probabilistic partial design problem* is the same as the deterministic partial design problem except that C_F is a distilled probabilistic design (Lee and Jiang 2018) with black boxes, whose functions at the primary outputs can be described by $F(X, Z, T)$, where Z represents the variables for the auxiliary inputs that trigger errors in C_F (including the errors of the black boxes) and T corresponds to the variables of the black box outputs. Each black box output t_i is specified with its input variables (i.e., dependency set) $D_i \subseteq X \cup Y$ in C_F. When t_i is substituted with h_i in C_F, the function of the resultant circuit is required to be sufficiently close to the specification with respect to some expected probability.

The hardness of the problem is stated in Theorem 2, whose proof can be found in the full version of this work at https://arxiv.org/abs/1911.04112.

Theorem 2. *The probabilistic partial design equivalence checking problem is NEXPTIME-complete.*

Moreover, the probabilistic partial design problem can be encoded with the following DSSAT formula

$$\text{ꓤ}X, \text{ꓤ}Z, \forall Y, \exists T(D).$$
$$(Y \equiv E(X)) \rightarrow (F(X, Z, T) \equiv G(X)), \tag{8}$$

where the primary input variables are randomly quantified with probability p_{x_i} of $x_i \in X$ to reflect their weights, and the error-triggering auxiliary input variables Z are randomly quantified according to the pre-specified error rates of the erroneous gates in C_F. Notice that the above DSSAT formula takes advantage of the extension with universal quantifiers as discussed previously.

In approximate design, a circuit implementation may deviate from its specification by a certain extent. The amount of deviation can be characterized in a way similar to the error probability calculation in probabilistic design. For approximate partial design, the equivalence checking problem can be expressed by the DSSAT formula:

$$\text{ꓤ}X, \forall Y, \exists T(D).$$
$$(Y \equiv E(X)) \rightarrow (F(X, T) \equiv G(X)), \tag{9}$$

which differs from Eq. (8) only in requiring no auxiliary inputs. The probabilities of the randomly quantified primary input variables are determined by the approximation criteria in measuring the deviation. For example, when all the input assignments are of equal weight, the probabilities of the primary inputs are all set to 0.5.

6 Application: Modeling Dec-POMDP

In this section we demonstrate the descriptive power of DSSAT to model NEXPTIME-complete problems by constructing a polynomial-time reduction from Dec-POMDP to DSSAT. Our reduction is an extension of that from POMDP to SSAT proposed in the previous work (Salmon and Poupart 2019).

In essence, given a Dec-POMDP $\mathcal{M}$, we will construct in polynomial time a DSSAT formula Φ such that there is a joint policy $\vec{\pi}$ for $\mathcal{M}$ with value $V(\vec{\pi})$ if and only if there is a set of Skolem functions $\mathcal{F}$ for Φ with satisfying probability $\Pr[\Phi|_{\mathcal{F}}]$, and $V(\vec{\pi}) = \Pr[\Phi|_{\mathcal{F}}]$.

First we introduce the variables used in construction of the DSSAT formula and their domains. To improve readability, we allow a variable x to take values from a finite set $U = \{x_1, \ldots, x_K\}$ (Salmon and Poupart 2019). Under this setting, a randomized quantifier ꓤ over variable x specifies a distribution $\Pr[x \equiv x_i]$ for each $x_i \in U$. We also define a scaled reward function:

$$r(s, \vec{a}) = \frac{\rho(s, \vec{a}) - \min_{s', \vec{a}'} \rho(s', \vec{a}')}{\sum_{s'', \vec{a}''} [\rho(s'', \vec{a}'') - \min_{s', \vec{a}'} \rho(s', \vec{a}')]}$$

such that $r(s, \vec{a})$ forms a distribution over all pairs of s and $\vec{a}$, i.e., $\forall s, \vec{a}. r(s, \vec{a}) \geq 0$ and $\sum_{s, \vec{a}} r(s, \vec{a}) = 1$. We will use the following variables:

- $x_s^t \in S$: the state at stage t,
- $x_a^{i,t} \in A_i$: the action taken by Agent i at stage t,
- $x_o^{i,t} \in O_i$: the observation received by Agent i at stage t,

$$\bigwedge_{0\le t\le h-2}[x_p^t \equiv \bot \to \bigwedge_{i\in I} x_o^{i,t} \equiv 0 \wedge x_s^{t+1} \equiv 0 \wedge x_p^{t+1} \equiv \bot] \tag{10}$$

$$x_p^{h-1} \equiv \bot \tag{11}$$

$$\bigwedge_{s\in S}\bigwedge_{\vec{a}\in\vec{A}}[x_p^0 \equiv \bot \wedge x_s^0 \equiv s \wedge \bigwedge_{i\in I} x_a^{i,0} \equiv a_i \to x_r^0 \equiv N_r(s,\vec{a})] \tag{12}$$

$$\bigwedge_{1\le t\le h-1}\bigwedge_{s\in S}\bigwedge_{\vec{a}\in\vec{A}}[x_p^{t-1} \equiv \top \wedge x_p^t \equiv \bot \wedge x_s^t \equiv s \wedge \bigwedge_{i\in I} x_a^{i,t} \equiv a_i \to x_r^t \equiv N_r(s,\vec{a})] \tag{13}$$

$$\bigwedge_{0\le t\le h-2}\bigwedge_{s\in S}\bigwedge_{\vec{a}\in\vec{A}}\bigwedge_{s'\in S}[x_p^t \equiv \top \wedge x_s^t \equiv s \wedge \bigwedge_{i\in I} x_a^{i,t} \equiv a_i \wedge x_s^{t+1} \equiv s' \to x_{T_{s,\vec{a}}}^t \equiv s'] \tag{14}$$

$$\bigwedge_{0\le t\le h-2}\bigwedge_{s'\in S}\bigwedge_{\vec{a}\in\vec{A}}\bigwedge_{\vec{o}\in\vec{O}}[x_p^t \equiv \top \wedge x_s^{t+1} \equiv s' \wedge \bigwedge_{i\in I} x_a^{i,t} \equiv a_i \wedge \bigwedge_{i\in I} x_o^{i,t} \equiv o_i \to x_{\Omega_{s',\vec{a}}}^t \equiv N_\Omega(\vec{o})] \tag{15}$$

Figure 3: The formulas used to encode a Dec-POMDP $\mathcal{M}$.

- $x_r^t \in S \times (A_1 \times \ldots \times A_n)$: the reward earned at stage t,
- $x_T^t \in S$: transition distribution at stage t,
- $x_\Omega^t \in O_1 \times \ldots \times O_n$: observation distribution at stage t,
- $x_p^t \in \mathbb{B}$: used to sum up rewards across stages.

We represent elements in the sets S, A_i, and O_i by integers, i.e., $S = \{0, 1, \ldots, |S| - 1\}$, etc., and use indices s, a_i, and o_i to iterate through them, respectively. On the other hand, a special treatment is required for variables x_r^t and x_Ω^t, as they range over Cartesian products of several sets. We will give a unique number to an element in a product set as follows. Consider $\vec{Q} = Q_1 \times \ldots \times Q_n$, where each Q_i is a finite set. An element $\vec{q} = (q_1, \ldots, q_n) \in \vec{Q}$ is numbered by $N(q_1, \ldots, q_n) = \sum_{i=1}^n q_i(\prod_{j=1}^{i-1} |Q_j|)$. In the following construction, variables x_r^t and x_Ω^t will take values from the numbers given to the elements in $S \times \vec{A}$ and $\vec{O}$ by $N_r(s, \vec{a})$ and $N_\Omega(\vec{o})$, respectively.

We begin by constructing a DSSAT formula for a Dec-POMDP with $h = 1$. Under this setting, the derivation of the optimal joint policy is simplified to finding an action for each agent such that the expectation value of the reward function is maximized, i.e.,

$$\vec{a}^* = \arg\max_{\vec{a}\in\vec{A}} \sum_{s\in S} \Delta_0(s) r(s, \vec{a})$$

The DSSAT formula below encodes the above equation:

$$\exists x_s^0, \exists x_r^0, \exists x_a^{1,0}(D_{x_a^{1,0}}), \ldots, \exists x_a^{n,0}(D_{x_a^{n,0}}).\phi,$$

where the distribution of x_s^0 follows $\Pr[x_s^0 \equiv s] = \Delta_0(s)$, the distribution of x_r^0 follows $\Pr[x_r^0 \equiv N_r(s, \vec{a})] = r(s, \vec{a})$, each $D_{x_a^{i,0}} = \emptyset$, and the matrix:

$$\phi = \bigwedge_{s\in S}\bigwedge_{\vec{a}\in\vec{A}}[x_s^0 \equiv s \wedge \bigwedge_{i\in I} x_a^{i,0} \equiv a_i \to x_r^0 \equiv N_r(s, \vec{a})].$$

As the existentially quantified variables have no dependency on randomly quantified variable, the DSSAT formula is effectively an exist-random quantified SSAT formula.

For an arbitrary Dec-POMDP with $h > 1$, we follow the two steps proposed in the previous work (Salmon and Poupart 2019), namely *policy selection* and *policy evaluation*, and adapt the policy selection step for the multi-agent setting in Dec-POMDP.

In the previous work (Salmon and Poupart 2019), an agent's policy selection is encoded by the following prefix (use Agent i as an example):

$$\exists x_a^{i,0}, \forall x_p^0, \forall x_o^{i,0}, \ldots, \forall x_p^{h-2}, \forall x_o^{i,h-2}, \exists x_a^{i,h-1}, \forall x_p^{h-1}.$$

In the above quantification, variable x_p^t is introduced to sum up rewards earned at different stages. It takes values from $\mathbb{B}$, and follows a uniform distribution, i.e., $\Pr[x_p^t \equiv \top] = \Pr[x_p^t \equiv \bot] = 0.5$. When $x_p^t \equiv \bot$, the process is stopped and the reward at stage t is earned; when $x_p^t \equiv \top$, the process is continued to stage $t + 1$. Note that variables $\{x_p^t\}$ are shared across all agents. With the help of variable x_p^t, rewards earned at different stages are summed up with an equal weight 2^{-h}. Variable $x_o^{i,t}$ also follows a uniform distribution $\Pr[x_o^{i,t} \equiv o_i] = |O_i|^{-1}$, which scales the satisfying probability by $|O_i|^{-1}$ at each stage. Therefore, we need to re-scale the satisfying probability accordingly in order to obtain the correct satisfying probability corresponding to the value of a joint policy. The scaling factor, denoted κ_h, equals $2^h(|\vec{O}||S|)^{h-1}$ (derived in the proof of Theorem 3).

As the actions of the agents can only depend on their own observation history, for the selection of a joint policy it is not obvious how to combine the quantification, i.e., the selection of a policy, of each agent into a linearly ordered prefix required by SSAT, without suffering an exponential translation cost. On the other hand, DSSAT allows to specify the dependency of an existentially quantified variable freely and is suitable to encode the selection of a joint policy. In the prefix of the DSSAT formula, variable $x_a^{i,t}$ depends on $D_{x_a^{i,t}} = \{x_o^{i,0}, \ldots, x_o^{i,t-1}, x_p^0, \ldots, x_p^{t-1}\}$.

Next, the policy evaluation step is exactly the same as that in the previous work (Salmon and Poupart 2019). The fol-

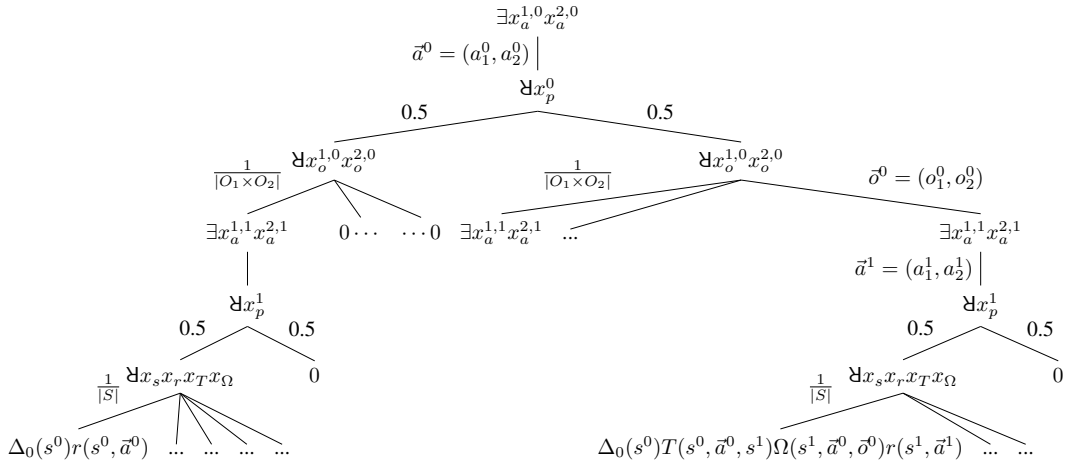

Figure 4: The decision tree of a Dec-POMDP example with two agents and $h = 2$.

lowing quantification computes the value of a joint policy:

$$\aleph x_s^t, \aleph x_r^t, t = 0, \ldots, h-1$$
$$\aleph x_T^t, \aleph x_\Omega^t, t = 0, \ldots, h-2$$

Variables x_s^t follow a uniform distribution $\Pr[x_s^t \equiv s] = |S|^{-1}$ except for variable x_s^0, which follows the initial distribution specified by $\Pr[x_s^0 \equiv s] = \Delta_0(s)$; variables x_r^t follow the distribution of the reward function $\Pr[x_r^t \equiv N_r(s, \vec{a})] = r(s, \vec{a})$; variables x_T^t follow the state transition distribution $\Pr[x_{T_{s,\vec{a}}}^t \equiv s'] = T(s, \vec{a}, s')$; variables x_Ω^t follow the observation distribution $\Pr[x_{\Omega_{s',\vec{a}}}^t \equiv N_\Omega(\vec{o})] = \Omega(s', \vec{a}, \vec{o})$. Note that these variables encode the random mechanism of a Dec-POMDP and are hidden from agents. That is, variables $x_a^{i,t}$ do not depend on the above variables.

The formulas to encode $\mathcal{M}$ are listed in Figure 3. Formula (10) encodes that when $x_p^t \equiv \bot$, i.e., the process is stopped, the observation $x_o^{i,t}$ and next state x_s^{t+1} are set to a preserved value 0, and $x_p^{t+1} \equiv \bot$. Formula (11) ensures the process is stopped at the last stage. Formula (12) ensures the reward at the first stage is earned when the process is stopped, i.e., $x_p^0 \equiv \bot$. Formula (13) requires the reward at stage $t > 0$ is earned when $x_p^{t-1} \equiv \top$ and $x_p^t \equiv \bot$. Formula (14) encodes the transition distribution from state s to state s' given actions $\vec{a}$ are taken. Formula (15) encodes the observation distribution to receive observation $\vec{o}$ under the situation that state s' is reached after actions $\vec{a}$ are taken.

Theorem 3. *The above reduction maps a Dec-POMDP $\mathcal{M}$ to a DSSAT formula Φ, such that a joint policy $\vec{\pi}$ exists for $\mathcal{M}$ if and only if a set of Skolem functions $\mathcal{F}$ exists for Φ, with $V(\vec{\pi}) = \Pr[\Phi|_{\mathcal{F}}]$.*

Due to space limit, the proof is available in the full-paper version at https://arxiv.org/abs/1911.04112. Note that the proposed reduction is a polynomial-time reduction, as the numbers of variables and clauses in the resulting DSSAT formula are polynomials of the input size of the Dec-POMDP. Below we demonstrate the reduction with an example.

Example 1. *Consider a Dec-POMDP with two agents and planning horizon $h = 2$. Given a joint policy (π_1, π_2) for Agent 1 and Agent 2, let the actions taken at $t = 0$ be $\vec{a}^0 = (a_1^0, a_2^0)$ and the actions taken at $t = 1$ under certain observations $\vec{o}^0 = (o_1^0, o_2^0)$ be $\vec{a}^1 = (a_1^1, a_2^1)$. The value of this joint policy is computed by Eq.* (3) *as*

$$V(\pi) = \sum_{s^0 \in S} \Delta_0(s^0)[r(s^0, \vec{a}^0) + \sum_{\vec{o}^0 \in \vec{O}} \sum_{s^1 \in S} T(s^0, \vec{a}^0, s^1) \Omega(s^1, \vec{a}^0, \vec{o}^0) r(s^1, \vec{a}^1)].$$

The decision tree of the converted DSSAT formula is shown in Figure 4. Note that the randomized quantifiers over variables x_p^t, x_s^t, and x_o^t will scale the satisfying probability by the corresponding factors labelled on the edges. Therefore, we have to re-scale the satisfying probability by $2^2|S||O_1 \times O_2|$, according to the scaling factor $\kappa_h = 2^h(|\vec{O}||S|)^{h-1}$. (A more detailed explanation for this example can be found in the full paper.)

7 Conclusions and Future Work

In this paper, we extended DQBF to its stochastic variant DSSAT and proved its NEXPTIME-completeness. Compared to the PSPACE-complete SSAT, DSSAT is more powerful to succinctly model NEXPTIME-complete decision problems with uncertainty. The new formalism can be useful in applications such as artificial intelligence and system design. Specifically, we demonstrated the DSSAT formulation of the analysis to probabilistic/approximate partial design, and gave a polynomial-time reduction from the NEXPTIME-complete Dec-POMDP to DSSAT. We envisage the potential broad applications of DSSAT and plan solver development for future work. We note that recent developments of *clausal abstraction* for QBF (Janota and Marques-Silva 2015; Rabe and Tentrup 2015) and DQBF (Tentrup and Rabe 2019) might provide a promising framework for DSSAT solving. Clausal abstraction has been lifted to SSAT (Chen, Huang, and Jiang 2021), and we are investigating its feasibility for DSSAT.

Acknowledgments

The authors are grateful to Christoph Scholl, Ralf Wimmer, and Bernd Becker for valuable discussions motivating this work. This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant No. 108-2221-E-002-144-MY3, 108-2218-E-002-073, and 109-2224-E-002-008. JHJ was supported in part by the Alexander von Humboldt Foundation during this work.

References

Balabanov, V.; Chiang, H.-J. K.; and Jiang, J.-H. R. 2014. Henkin quantifiers and Boolean formulae: A certification perspective of DQBF. *Theoretical Computer Science* 523: 86–100.

Barrett, C.; and Tinelli, C. 2018. Satisfiability modulo theories. *Handbook of Model Checking* 305–343.

Bérard, B.; Bidoit, M.; Finkel, A.; Laroussinie, F.; Petit, A.; Petrucci, L.; and Schnoebelen, P. 2013. *Systems and Software Verification: Model-checking Techniques and Tools*. Springer Science & Business Media.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27(4): 819–840.

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

Büning, H. K.; and Bubeck, U. 2009. Theory of quantified Boolean formulas. *Handbook of Satisfiability* 185: 735–760.

Chakrapani, L. N. B.; George, J.; Marr, B.; Akgul, B. E. S.; and Palem, K. V. 2008. Probabilistic design: A survey of probabilistic CMOS technology and future directions for terascale IC design. In *VLSI-SoC: Research Trends in VLSI and Systems on Chip*, 101–118.

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 35th AAAI Conference on Artificial Intelligence (AAAI)*.

De Moura, L.; and Bjørner, N. 2011. Satisfiability modulo theories: Introduction and applications. *Communications of the ACM* 54(9): 69–77.

Fremont, D. J.; Rabe, M. N.; and Seshia, S. A. 2017. Maximum model counting. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 3885–3892.

Gitina, K.; Reimer, S.; Sauer, M.; Wimmer, R.; Scholl, C.; and Becker, B. 2013. Equivalence checking of partial designs using dependency quantified Boolean formulae. In *Proceedings of the IEEE 31st International Conference on Computer Design (ICCD)*, 396–403.

Janota, M.; and Marques-Silva, J. 2015. Solving QBF by clause selection. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 325–331.

Jhala, R.; and Majumdar, R. 2009. Software model checking. *ACM Computing Surveys* 41(4): 21:1–21:54.

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.; Wang, Y.-S.; and Jiang, J.-H. R. 2017. Solving stochastic Boolean satisfiability under random-exist quantification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 688–694.

Lee, N.-Z.; Wang, Y.-S.; and Jiang, J.-H. R. 2018. Solving exist-random quantified stochastic Boolean satisfiability via clause selection. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 1339–1345.

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

Majercik, S. M. 2009. Stochastic Boolean satisfiability. *Handbook of Satisfiability* 185: 887–925.

Majercik, S. M.; and Boots, B. 2005. DC-SSAT: A divide-and-conquer approach to solving stochastic satisfiability problems efficiently. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence (AAAI)*, 416–422.

Majercik, S. M.; and Littman, M. L. 1998. MAXPLAN: A new approach to probabilistic planning. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning (AIPS)*, 86–93.

Majercik, S. M.; and Littman, M. L. 2003. Contingent planning under uncertainty via stochastic satisfiability. *Artificial Intelligence* 147(1-2): 119–162.

Marques-Silva, J. P.; and Sakallah, K. A. 2000. Boolean satisfiability in electronic design automation. In *Proceedings of the 37th Annual Design Automation Conference (DAC)*, 675–680.

Narizzano, M.; Pulina, L.; and Tacchella, A. 2006. The QBFEVAL web portal. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA)*, 494–497.

Nilsson, N. J. 2014. *Principles of Artificial Intelligence*. Morgan Kaufmann.

Oliehoek, F. A.; Amato, C.; et al. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32: 289–353.

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(7-8): 957–992.

Peterson, G. L.; and Reif, J. H. 1979. Multiple-person alternation. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science (FOCS)*, 348–363.

Rabe, M. N.; and Tentrup, L. 2015. CAQE: A certifying QBF solver. In *Proceedings of the 15th Conference on Formal Methods in Computer-Aided Design (FMCAD)*, 136–143.

Russell, S. J.; and Norvig, P. 2016. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited.

Salmon, R.; and Poupart, P. 2019. On the relationship between stochastic satisfiability and partially observable Markov decision processes. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*, 407:1–407:11.

Scholl, C.; and Wimmer, R. 2018. Dependency quantified Boolean formulas: An overview of solution methods and applications. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 3–16.

Sinha, S.; Mishchenko, A.; and Brayton, R. K. 2002. Topologically constrained logic synthesis. In *Proceedings of the 21st IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 679–686.

Stockmeyer, L. J.; and Meyer, A. R. 1973. Word problems requiring exponential time. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC)*, 1–9.

Tentrup, L.; and Rabe, M. N. 2019. Clausal abstraction for DQBF. In *Proceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 388–405.

Venkatesan, R.; Agarwal, A.; Roy, K.; and Raghunathan, A. 2011. MACACO: Modeling and analysis of circuits for approximate computing. In *Proceedings of the 30th IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 667–673.

Wang, L.-T.; Chang, Y.-W.; and Cheng, K.-T. T. 2009. *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann.