A Survey on Applications of Quantified Boolean Formulas

(Invited Paper)

Ankit Shukla, Armin Biere, Martina Seidl
Institute for Formal Models and Verification
Johannes Kepler University Linz, Austria
Email: {ankit.shukla, biere, martina.seidl}@jku.at

Luca Pulina
Intelligent system DEsign and Applications
University of Sassari, Italy
Email: lpulina@uniss.it

***Abstract*—The decision problem of quantified Boolean formulas (QBFs) is the archetypical problem for the complexity class PSPACE that contains many reasoning problems of practical relevance. Because of the availability of a rich solving infrastructure aspects QBFs provide an attractive framework for encoding and solving such reasoning problems ranging from symbolic reasoning in artificial intelligence to the formal verification and synthesis of computing systems. In this paper, we survey the different application areas that exploit QBF technology for solving their specific problems.**

I. Introduction

The last two decades have seen the rise of a rich infrastructure for solving quantified Boolean formulas (QBFs) [1], [2]. Many sophisticated solving paradigms were developed based on different proof systems of orthogonal strength and they were implemented in powerful solvers. Novel preprocessing techniques for simplifying formulas were invented as well as efficient approaches for result validation and winning strategy extraction, to name just a few examples. The progress is witnessed by the annual QBF competition called QBFEval (see for example [3] for the competition reports of 2016 and 2017). All these efforts are motivated by the success story of SAT solvers, i.e., tools for deciding the satisfiability problem of propositional logic (SAT) [4]. By being NP-complete, SAT is "evidently a killer app, because it is key to the solution of so many other problems" [5].

For problems whose complexity is beyond NP, more powerful formalisms are needed in order to play a similar disruptive role as SAT plays for solving problems in NP. One very promising direction is to use QBFs, which extend SAT with existential and universal quantifiers over propositional variables. The decision problem of QBFs is the prototypical PSPACE-problem [6]. Further, there is a strong relationship between the formulation of (closed) QBFs and the polynomial-time hierarchy. Indeed, given a closed QBF having a quantifier prefix composed of n quantifier blocks, its decision problem is at the n-th level of the polynomial hierarchy. More precisely, the decision problem is Σ_n^p-complete if the outermost quantifier is existential and Π_n^p-complete if the outermost quantifier is universal. Hence, any problem in PSPACE can be polynomially reduced to a QBF, allowing for an exponentially more succinct encoding than possible with SAT.

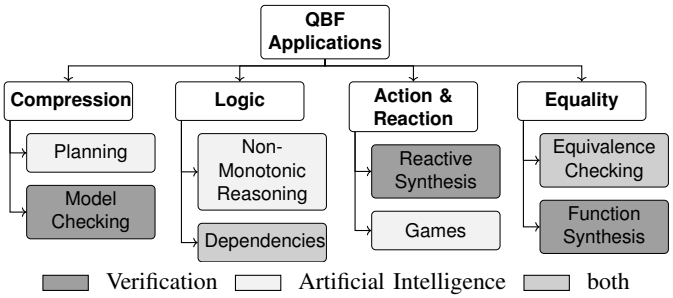


Fig. 1: Application of QBF in different domains

In this paper, we survey areas in which QBF encodings and technology for solving their problems are applied in practice. Figure 1 summarizes those areas. Although applications originate from different fields—mainly artificial intelligence and formal verification—they all use quantification for similar purposes like for compressing a SAT encoding by avoiding unrolling a transition relation, for capturing action and reaction of a two-players game, for controlling logical models, or for equivalence checking. Many of the presented QBF applications form the basis of the benchmarks that are collected at

`http://www.qbflib.org`,

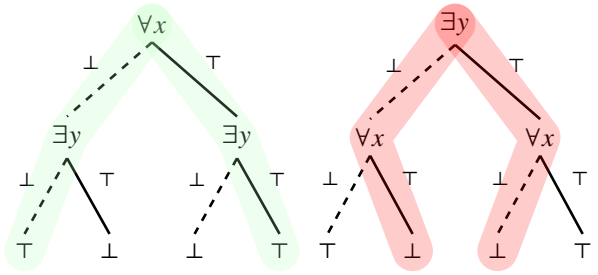
the benchmark library portal of the QBF community. We deliberately do not consider approaches that perform reductions to or from QBFs to show hardness or membership of some formalism.

II. Preliminaries

Given a set of propositional variables V, the language of propositional logic $\mathcal{L}(V)$ contains all elements of V, the truth constants $\top$ (true) and $\bot$ (false), and all well-formed formulas built from elements of $\mathcal{L}(V)$ and Boolean connectives $\neg, \vee, \wedge, \rightarrow, \leftrightarrow, \oplus, \ldots$ A formula $F \in \mathcal{L}(V)$ is satisfiable iff there is an assignment of the variables occurring in F to true and false such that F evaluates to true under this assignment under the standard semantics of the Boolean connectives. Otherwise, F is unsatisfiable. If F is satisfiable for all possible variable assignments, then F is valid. For example, $(x \leftrightarrow y)$ is a propositional formula that is satisfiable but not valid.

The language of quantified Boolean formulas $\mathcal{Q}(V)$ extends $\mathcal{L}(V)$ by quantifiers over V as follows: (1) $\mathcal{L}(V) \subseteq \mathcal{Q}(V)$, (2) if $\Phi \in \mathcal{Q}(V)$, then also $Qv\Phi \in \mathcal{Q}(V)$ where $Q \in \{\forall, \exists\}$. Then Q is the quantifier of v and Φ the scope of Qv. If all variables v occurring in a QBF Ψ are in the scope of a quantifier, then Ψ is called *closed*. We sometimes write consecutive quantifiers of the same type as a set, i.e., $Qx_1 \dots Qx_n\Phi$ is also written as $QX\Phi$ where $X = \{x_1, \dots x_n\}$. We even omit the sets of variables if we would like to capture the quantifier structure—for example, $\exists X \forall Y \exists Z$ has the structure $\exists\forall\exists$. Often QBFs have the structure $P.F$ where $P = Qx_1 \dots Qx_n$ is a quantifier prefix and F is a propositional formula, called matrix.

A QBF $\forall x\Phi$ is true iff $\Phi|_{x=\top}$ and $\Phi|_{x=\bot}$ are true. Similarly, a QBF $\exists x\Phi$ is true iff $\Phi|_{x=\top}$ or $\Phi|_{x=\bot}$ is true. By $\Phi|_{x=t}$ we denote that x is replaced by t in Φ and the resulting formula is simplified according to the standard semantics of the Boolean connectives. For example, the QBF $\forall x \exists y (x \leftrightarrow y)$ is true, while the QBF $\exists y \forall x (x \leftrightarrow y)$ is false. The semantics of QBFs imposes tree shaped models and counter-models for witnessing (un-)satisfiability. The highlighted part of the left assignment tree is a model of $\forall x \exists y (x \leftrightarrow y)$, and the highlighted part of the right assignment tree is a counter-model of $\exists y \forall x (x \leftrightarrow y)$.



Modern QBF solvers are not only able to decide if a QBF is true or false, but can also return a model or a counter-model, also called winning strategies. These are usually represented as Boolean circuits, encoding a solution of the original application problem.

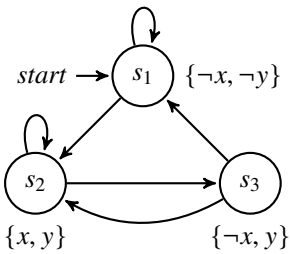
III. QBFs for Compression

Given a system model described by a state vector s and a transition relation, the *reachability* problem—i.e., checking whether a goal state can be reached from an initial state s_0 in k (or less) steps—can be expressed as the satisfiability of the propositional formula

$$I(s_0) \wedge T(s_0, s_1) \wedge \dots \wedge T(s_{k-1}, s_k) \wedge G(s_k) \tag{1}$$

where $I(s_0)$ denotes that s_0 is an initial state, $G(s_k)$ denotes that s_k is a goal state, and $T(s, s')$ the transition relation, which describes valid state transitions in the system. Example 1 presents how to symbolically encode the transition relation.

Example 1. *Given a transition system below on the left. On the right, we see the symbolic encoding of the transition relation T. To encode three states, we need two bits (x, y).*



$$T((x,y),(x',y')) = ((x' \Leftrightarrow x \vee y) \wedge (y' \Leftrightarrow y)) \vee ((x' \Leftrightarrow \neg y) \wedge (y' \Leftrightarrow x \vee \neg y))$$

$$I(x,y) = \neg x \wedge \neg y$$

$$G(x,y) = \neg x \wedge y$$

State s_1 is encoded by the pair $(\bot, \bot)$. A SAT solver could be used to find the states which can be reached in one step by solving the formula $T((\bot, \bot), (x', y'))$. One satisfying assignment is $(x', y') = (\top, \top)$, i.e., state s_2. If $I(x, y) = \neg x \wedge \neg y$ encodes the initial state, and $G(x, y) = \neg x \wedge y$ encodes the goal state, then the propositional formula $I \wedge T \wedge G$ encodes the problem to determine whether it is possible to reach the goal in one step. Obviously, this formula is unsatisfiable. However, by including T twice, $I \wedge T \wedge T \wedge G$, such a path can be found. □

The SAT encoding contains k copies of the transition relation due to the fact that T must be "unrolled" k times in order to check the reachability of the goal state in k steps. QBFs can be used to encode the problem more compactly as

$$\begin{aligned} &\exists s_0, \dots, s_k \; I(s_0) \wedge G(s_k) \wedge \\ &\forall x, x' [\vee_{i=0}^{k-1} (x = s_i) \wedge (x' = s_{i+1})] \rightarrow T(x, x') \end{aligned} \tag{2}$$

Satisfiability of the formula shows the existence of a valid path $s_0, \dots, s_k$ of length k, starting at some initial state s_0 and terminating in some goal state s_k. The last conjunct in the formula represents that for any two states x and x', if x and x' are adjacent along the sequence, then they are consistent with the transition relation. The compactness of the formula is due to containing only a single copy of the transition relation. This technique is used for encoding both automated planning and verification tasks in a similar manner.

A. Planning

Planning in artificial intelligence is the process of finding a sequence of actions that will achieve a predefined goal. The idea of encoding a planning problem in propositional logic (see [7] for a survey) was first presented in [8]. The approach translates the search for a plan that reaches a goal state in k steps into a propositional formula, whose satisfying truth assignments corresponds to valid plans. The classical planning problem with the restriction to plans that are of polynomial length w.r.t. the input size belong to the complexity class NP [9]. Generalizations of the classical planning problem like conditional planning are beyond NP. The conditional problem with the restriction of plans of polynomial length w.r.t. the input size, is Σ_2^p-complete [10], suggesting the use of QBFs as shown in Equation 2.

Conditional planning involves the presence of uncertainty in the initial state and in the nondeterminism of *effects* of actions. In planning without observability the current state of the system can not be exactly observed during plan execution.

This kind of conditional planning is known as *conformant planning*. Formally, a conformant planning problem instance is a 4-*tuple* $\langle S, I, A, G\rangle$ where S is a set of states, I and G are formulas over S, representing the sets of initial and goal states. Finally, A is a set of nondeterministic actions $\langle p, e\rangle$ with *precondition* p, a propositional formula over S and effect e over S defined recursively as: (1) $f \Rightarrow l$ is an effect where f is a formula over S and l is set of literals over S. (2) if e_1 and e_2 are effects over S then $e_1|e_2$ is also an effect over S. The operator '|' denotes the nondeterministic choice between e_1 and e_2. The effect e updates the current state and is executed if and only if its precondition is true.

Rintanen formulates the QBF encoding for conformant planning as follows [10]: *there exists* a plan (a sequence of actions), such that, *for all* possible contingencies (several initial states and nondeterministic transitions), *there exists* an execution E that reaches a goal state (prefix structure $\exists\forall\exists$). The existence of a plan is determined by

$$\exists P \forall C \exists E \left(I^0 \rightarrow \left(\bigvee_{i=0}^{k-1} \mathcal{R}(s_i, s_{i+1}, A^i, N^i) \wedge \mathrm{G}^k \right) \right) \tag{3}$$

where $P = \bigcup_{i=0}^{k-1} A^i$ is the set of actions representing a plan and $C = S^0 \cup \bigcup_{i=0}^{k-1} N^i$ denotes contingencies due to several initial states and nondeterminism, where N^i are auxiliary variables used to encode nondeterminism and S^0 the variables of the initial states. Finally, $E = \bigcup_{i=0}^{k} s_i$ denotes the sequence of states representing an execution. The plan is represented as a sequence of actions, that make transitions $\mathcal{R}(s_i, s_{i+1}, A^i, N^i)$ based on executed effects of actions. The formulas I^0 and G^k encode initial and goal states. In follow-up work [11], Rintanen presents an efficient QBF encoding of conformant planning with one alternation of quantifiers having a $\forall\exists$ structure.

Safe planning [12], an extension to generating plans, verifies safety properties, i.e. the plan never contains a bad state. The safety properties are defined in the quantified linear temporal logic. These properties are critical in robotics applications.

Recent work [13] translates planning problems into QBF which describe objects in terms of equivalence classes. The encoding performs a partial grounding with the goal to produce an exponentially smaller sized formula. It represents a single object for each ungrounded type and uses universal variables to create multiple contexts for this representation. The encoding by [14] translates the planning instance into a sequence of QBFs incrementally. The formula is then solved using an incremental QBF solver with the help of preprocessing and inprocessing techniques. The work by [15] provides more compact QBF encodings for generating plans. The encoding is based on an explanatory frame axiom and causal links. It makes improvement over the previous tree-based QBF encoding [16] of the planning problem.

B. Model Checking

Model checking [17] is an automatic verification technique widely used in industry. The aim of model checking is the verification finite-state system w.r.t. correctness properties expressed in a temporal logic like LTL. Originally, model checking relied on an explicit-state representation of the system, but due to the state-explosion problem symbolic approaches were investigated. Considerable improvement was achieved by the application of Binary Decision Diagrams, but the industrial breakthrough came with the application of SAT encodings [18] in bounded model checking (BMC).

In general, BMC verifies that a global invariant p holds in all states of the system that are reachable with k steps. For this purpose, Equation 1 is applied to determine all those states in which $\neg p$ holds by defining $G(s_k)$ respectively. If the propositional formula is true, then the model returned by the SAT solver gives the trace to the state in which p does not hold. An overview on SAT-based model checking can be found in [19].

Since the transition relation is usually huge, the increase of k becomes a limiting factor in the application of BMC. In order to avoid the explicit duplication of the transition function, QBF encodings as presented in Equation 2 have been suggested [20], [21], [22]. The diameter of the system has been shown to be an upper bound on the length k of error traces. Encodings into QBF for diameter calculations have been proposed in [23] as well as in [21]. Besides circuit verification, QBF-based model checking is applied to the verification of incomplete circuits designs [24], where universal quantification models unspecified behavior. For more details refer to [25].

IV. QBFs for Logic

Quantifiers allow both extension as well as restriction of propositional models. As example consider the encoding of the unique-SAT problem [26]. Given a propositional formula F over variables V, the following QBF is true if and only if F has exactly one model:

$$\begin{aligned} \exists v_1 \ldots v_n \, \forall u_1 \ldots u_n \Big(F \wedge \big(F[v_1/u_1, \ldots v_n/u_n] \rightarrow \\ ((v_1 \leftrightarrow u_1) \wedge \ldots \wedge (v_n \leftrightarrow u_n))\big)\Big) \end{aligned} \tag{4}$$

If this formula is true, then there is a truth assignment σ to variables $v_1, \ldots, v_n$ such that (1) F is true under this assignment (left part of the conjunction), and (2) for any assignment τ, either $\tau = \sigma$ or F is false under τ (right part of the conjunction). Similar ideas are used for finding models of non-monotonic formalisms. Further, QBFs are used for supporting the reasoning of more expressive logic as well.

A. Non-Monotonic Propositional Reasoning

In a non-monotonic setting, the addition of new knowledge can require to retract previously drawn conclusions. Various non-monotonic reasoning tasks have been translated to QBFs. For instance [27] contains QBF encodings for Abduction, Autoepistemic Logic, Disjunctive Logic Programs, Circumscription, and Default Logic. All of them belong to the second level of the polynomial hierarchy.

a) Abduction: Given a propositional theory T over variables V, *abductive reasoning* starts with a set of hypotheses $H \subseteq V$ and infers the most likely explanation $E \subseteq H$ for a statement $p \in V$. Explanation E gives a plausible, most-likely justification of p such that E is consistent with T and p can be classically inferred from $T \cup E$. Egly et al. [27] encode abductive explanation of p given hypotheses $H = \{h_1, \ldots, h_n\}$ and fresh variables $G = \{g_1, \ldots g_n\}$ by the QBF

$$\exists G\Big(\big(\exists V(T \wedge (G \rightarrow H))\big) \wedge \big(\forall V((T \wedge (G \rightarrow H)) \rightarrow p)\big)\Big)$$

where $G \rightarrow H$ is defined as $\bigwedge_{i=0}^{n}(g_i \rightarrow h_i)$. If a variable g_i is true, then $h_i \in E$, otherwise $h_i \notin E$. The left part of the conjunction of the QBF ensures that T and E are consistent, while the right part ensures that p is logically implied. Egly et al. [27] also extend this encoding to solve the relevance problem, i.e., to decide for some hypothesis h if it belongs to some explanation E. Recent work [28] revisits QBF encodings of abduction using QMaxSAT (see Sec. IV-B).

b) Autoepistemic Logic: The language of autoepistemic logic [29] extends propositional logic by a modal operator $\Box$. Intuitively, the modal atom $\Box F$ means that the propositional formula F is known. For example, the formula $\neg\Box F \rightarrow \neg F$ states that if F is not known to be true, it is assumed to be false, i.e., the formula expresses negation by failure. Given an theory T containing modal atoms M, a stable expansion exists if there is a consistent set $E \subseteq M \cup \{\neg\Box F \mid \Box F \in M\}$ such that $T \cup M \models \Box F$ for $\Box F \in E$ and $T \cup M \not\models \Box F$ for $\Box F \notin E$. For translating the search of a stable extension into a QBF decision problem [27], these conditions have to be included for all modal atoms in M. In the translation, the modal atoms $\Box F \in M$ are viewed as propositional variables M' which are existentially quantified in the outermost quantifier block. Let $v \in M'$ be the variable that stands for $\Box F \in M$. If the QBF solver finds a solution in which v is true, then $\Box F \in E$, otherwise $\neg\Box F \in E$.

c) Disjunctive Programs: A *disjunctive logic program* Π over variables V is given as a set of rules of the form "$H \rightarrow P, N$" where H is a disjunction, and P a conjunction of variables, and N a conjunction of negated variables. An interpretation $I \subseteq V$ is a *stable model* of Π if it is minimal model of Π^I, which is obtained from Π by (1) removing rules with negated variables in I and (2) deleting all negated variables. Existence of a stable model of Π is encoded as QBF with two conditions [27]. First, program Π is interpreted as propositional formula and required to have a model. Second, such a model has to be minimal w.r.t. Π^I. Variants for skeptical and brave reasoning are evaluated in [30], [31].

d) Circumscription: McCarthy [32] introduced circumscription to capture common sense reasoning that uses conjectures to draw certain conclusions. The propositional variant of circumscription is naturally formulated as a QBF [27]. Given a theory T over a set of variables V, we define

$$CIRC(T) := T \wedge \forall V'((T[V/V'] \wedge (V' \leq V)) \rightarrow (V \leq V'))$$

stating that only minimal models of T should be considered. Minimality refers to the number of variables set to true and is obtained via the universally quantified variables V': if T has a model σ, then there may not be a model σ' of $T[V/V']$ such that σ' has variables set to false which are set to true in σ. A circumscription of T is a formula ϕ over variables V for which the QBF $\forall V(CIRC(T) \rightarrow V)$ is true.

e) Default Logic: A default theory $T = (W, \Delta)$ distinguishes two kinds of information about the world: a trusty background theory W and defensible rules Δ [33]. Such rules allow the inferences of new facts under the current knowledge base, but may prohibit similar inferences under new, maybe more accurate knowledge. The decision problem of default logic asks if for a given default theory T there exists a so-called extension, a consistent set containing the background theory and all derivable facts such that no additional information can be derived. Egly et al. [27] presented two QBF encodings of this decision problem. Based on these encodings, they also implemented brave and cautious inference in QBF. While the former decides if a given formula ϕ holds in some extension, the latter decides if ϕ holds in all extensions.

f) Abstract Argumentation: Furthermore, several encodings in the context of abstract argumentation have been presented. An argumentation framework $F = (A, R)$ consists of a finite set of arguments A and attacks between arguments $R \subseteq A \times A$ [34]. A set $S \subseteq A$ is conflict free iff there are no $a, b \in S$ with $(a, b) \in R$. Depending on the concrete setting, reasoning in argumentation frameworks can result in decision problems up to the forth level of the polynomial hierarchy. A survey on different approaches to solve reasoning problems in abstract argumentation can be found in [35]. To the best of our knowledge Egly and Woltran were the first to use QBF encodings for reasoning in argumentation frameworks [36]. Arieli and Caminada presented a QBF-based formalization of abstract argumentation semantics [37]. QBF encodings of acceptance problems of abstract dialectical frameworks found at the third level of the polynomial hierarchy are presented in [38].

B. Dependencies

Other reasoning problems also make use of QBF encodings. Cooksey et al. express reasoning on relaxed memory models in second order logic for which they implemented a solver that internally relies on QBF technology [39]. Wimmer et al. presented an approach to solve DQBFs, a NEXPTIME-complete formalism, by solving equisatisfiable QBFs [40]. In contrast to QBF, dependencies between existential and universal variables can be expressed in DQBF, in the form of dependency sets. To solve a DQBF, dependencies are eliminated carefully by removing universal variables from single dependency sets at the price of generating copies of the involved existential variables. The approach is implemented in the DQBF solver HQS which won the DQBF Track of QBFEval 2018.

Another area in which QBF solvers are applied [41], is QMaxSAT, the problem of optimizing a cost function that

refers to a quantified set of constraints. In QMaxSAT there are soft clausal constraints and the solver has to find the largest subset of soft constraints such that the remaining QBF is true. For this purpose a QBF solver is required that is able to produce unsatisfiable cores.

Techniques involving QBFs have also been employed as back-end technology for tasks in complex verification frameworks. For instance, in symbolic software model checking, QBF encodings (and related approaches) have been used for detecting fixpoints [42], [43].

V. QBFs for Modeling Action and Reaction

The evaluation of QBFs can also be seen as a two-player game between the ∀-player and the ∃-player. Each player owns all the variables of the respective quantification. Given a QBF of the form $\Phi = Qx_1 \dots Qx_n\phi$ with ϕ being a propositional formula, the matrix of Φ, the ∀-player has the goal to make the matrix true while the ∃-player has the goal to make the matrix false. The moves in the game are the assignments of the variables: a variable x_i may only be assigned by its owner and only if all variables x_j with $j < i$ have been assigned. The QBF Φ is true if and only if for all moves of the ∀-player there is a move of the ∃-player such that the matrix is true, i.e., there exists a winning strategy. This two-player game semantics is exploited for reactive synthesis. The QBF solver is used to search for a safe implementation of a system controller such that for all moves by the adversarial environment, there exist moves by the controlled system such that during execution the system does not reach an unsafe state. This results in a QBF with two quantifier alternations.

A. Reactive Synthesis

The aim of synthesis is to derive the implementation of a system automatically from a formal specification. Given a finite-state transition system (S, I, U, C, T) where S is a set of states, $I \subseteq S$ are initial states, U are uncontrollable (adversarial) input variables, C are controllable input variables, and T is a transition relation of the form $S \times 2^U \times 2^C \times S$. If $(s, u, c, s') \in T$ then then it is possible to transition from state s to s' under inputs u and c. Furthermore, let $P \subseteq S$ be a set of states, described implicitly by a propositional or a linear temporal logic (LTL) formula, representing the specification. For example, a safety specification states that certain "bad things" like a crash do not happen. The challenge is now to synthesize a function which, for any state s that is reachable from an initial state and for any input u of 2^U, to set the controlled inputs c of 2^C such that $(s, u, c, s') \in T$ and $s' \in P$, i.e., s' conforms to the specification.

Again QBFs can be used to encode the existence of a winning region, that is a set of safe states such that for any input u, there exists an input c leading back to a state in the winning region. The algorithm for calculating the winning region presented by Bloem et al. [44] first includes all safe states P in an over-approximation of the winning region. Then a QBF solver is consulted to find a state from which a transition to an unsafe state (in one step) can not be avoided. That state is excluded from the previous winning region. Checking spurious states is repeated until fixpoint or an initial state has to be removed, indicating that the specification is not realizable. This approach repeatedly solves QBFs with prefix ∃∀∃. In earlier work [45], in each iteration the previously constructed QBF is extended by a QBF expressing that an additional step does not lead outside of the winning region. This not only duplicates the transition relation but also requires about $2n$ quantifier alternations.

Furthermore, debugging techniques based on QBF have been proposed [46], [22] for VLSI design. In [47] the authors show the usage of QBF encodings inside a framework aiming at debugging complex systems, in order to determine fault candidates that can fix all faulty behavior with respect to a functional specification. Sequential Automated Test Pattern Generation (ATPG) for circuit verification using QBFs has been used for the first time in [22] and further benefits of using QBFs for ATPG and circuit design were recently reported in [48].

In very recent work [49], the given LTL specification is first translated to a universal co-Büchi automaton B, that is a labeled transition system which describes a language consisting of words of infinite length. A word w is accepted iff for all runs the states that are visited infinitely often are accepting states. To synthesize a transition system T of bounded size n such that every word of T is accepted by B, the product of T and B may contain only paths that have a finite number of non-accepting states. This property is ensured by annotating the states of the product automaton such that every initial state is annotated by a number. Furthermore, the non-accepting states need to be labeled by a number that is strictly greater than the labels of its source states. The QBF encoding asks if there exist valid annotations and these annotations are encoded as bit-vectors.

B. Games

In adversarial games, two players compete to reach a destination position, at the same time preventing the opponent from reaching her destination. Determining a plan for the first player, such that she can win the game by reaching her destination irrespective of the moves by the other player is encoded in [50] as a QBF

$$\exists X(\exists V G_X \wedge \forall Y \neg G_Y) \tag{5}$$

where G_X and G_Y are goal formulas of player X and Y in conjunctive normal form. The truth assignment to X corresponds to a plan for the first player and the truth assignment to Y to a plan for the second.

The work of [51] addresses unexpected computational difficulties while modeling adversarial games as a QBF satisfaction problem. The formulation takes the legal actions of each player into the considerations while encoding the problem. This reduces the exploration of the combinatorial space by the QBF solver. The idea is to construct a QBF formula such that a contradiction is derived as soon as an illegal action is performed.

Diptarama presented an encoding of generalized Tic-Tac-Toe [52]. Two players alternately put q stones on a grid board until one achieves a given polyomino. They generated formulas with up to 15 quantifier alternations. The encoding is based on the approach of Gent and Rowley who introduced the Connect-X problem on a $w \times h$ grid. For the Connect-4 problem, they produced a QBF with 21 quantifier alternations [53].

VI. QBFs for Equality

Universal quantification also allows to ensure that two circuits or even programs have the same behavior. If one of them is not fully specified, the QBF solver is used to fill the "holes" and synthesize or optimize Boolean functions.

A. Equivalence Checking

For logic programs under the answer-set semantics, several notions of equivalence between programs have been suggested [54] including ordinary, strong, uniform, relativised equivalence as well as program comparison under projected answer sets. In [55] a tool is presented that compiles all these correspondence problems to QBFs. This tool has been used for automatically assessing the solutions to programming exercises of a university course on logic programming [56].

In [57] a QBF encoding as been used for the verification of LAN security policy specifications; they encode in QBF the problem to check if the implementation of a security policy in an enterprise LAN satisfies a given security policy model. In [58] the authors formalized in QBF the problem to check the conformance of Software Product Lines between the requirements and the design level. In [59] QBF is applied to property verification in the design of hardware power management modules.

B. Synthesis of Boolean Functions

Given a Boolean function $f(V)$ over variables V, the task of *bi-decomposition* consists of splitting $f(V)$ into $f_1(V_1, V_3)$ and $f_2(V_2, V_3)$ such that $V = V_1 \cup V_2 \cup V_3$ and $f(V) = f_1(V_1, V_3) \circ f_2(V_2, V_3)$, $\circ \in \{\vee, \wedge, \oplus\}$. As shown in [60] this task can be achieved by a QBF encoding that, on the one hand ensures that neither V_1 nor V_2 is empty, that the two sets are disjoint and in addition balanced. The resulting QBF has the prefix structure $\exists\forall$.

Wille et al. presented a QBF encoding for the synthesis of a Boolean function from reversible gates [61]. A reversible gate maps each possible input vector to a unique output vector realizing a bijective function. The presented approach realizes a given function by a cascade of reversible gates. Therefore, a QBF with prefix $\exists X \forall Y$ has to be solved. The variables of set X determine the inclusion and exclusion of the gates that are available for synthesizing the given function, while the Y variables ensure that the synthesized function and the given function are equivalent.

VII. Conclusion

In the last 20 years we have seen many applications of QBF solving in vastly diverse application domains of artificial intelligence and beyond. The historical and particularly recent advancement of the state of the art in QBF is quite remarkable, as proven by the results of the annual QBF competition QBFEVAL[1] [3]. We believe that QBF is becoming mature enough to be become a killer app for many tasks that require more powerful reasoning than SAT.

Our survey shows that QBF technology has spread among very different research communities who have to solve similar problems in different contexts. The majority of problems for which QBFs have been used are located in the lower levels of the polynomial hierarchy indicating a high demand for solvers specialized on that specific quantifier structure.

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF) under projects W1255-N23 and S11408-N23 and the LIT AI Lab funded by the State of Upper Austria.

References

[1] O. Beyersdorff, M. Janota, F. Lonsing, and M. Seidl, "Proof Systems for Practical QBF Solving." Submitted for the 2. Edition of the Handbook of Satisfiablity, to appear.

[2] E. Giunchiglia, P. Marin, and M. Narizzano, "Reasoning with quantified boolean formulas," in *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, vol. 185, pp. 761–780.

[3] L. Pulina and M. Seidl, "The 2016 and 2017 QBF Solvers Evaluations (QBFEVAL'16 and QBFEVAL'17)," *Artif. Intell.*, vol. 274, pp. 224–248, 2019.

[4] A. Biere, M. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, 2009.

[5] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*. Addison-Wesley Professional, 2015.

[6] L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time (preliminary report)," in *ToC*. ACM, 1973, pp. 1–9.

[7] J. Rintanen, "Planning and SAT," in *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, vol. 185, pp. 483–504.

[8] H. A. Kautz and B. Selman, "Planning as satisfiability," in *ECAI*, 1992, pp. 359–363.

[9] T. Bylander, "The computational complexity of propositional STRIPS planning," *Artif. Intell.*, vol. 69, no. 1-2, pp. 165–204, 1994.

[10] J. Rintanen, "Constructing conditional plans by a theorem-prover," *J. Artif. Intell. Res.*, vol. 10, pp. 323–352, 1999.

[11] ——, "Asymptotically optimal encodings of conformant planning in QBF," in *AAAI*. AAAI Press, 2007, pp. 1045–1050.

[12] M. De Luca, E. Giunchiglia, M. Narizzano, and A. Tacchella, ""Safe Planning" as a QBF evaluation problem," in *Second RoboCare Workshop*, 2005, p. 45.

[13] M. Cashmore, M. Fox, and E. Giunchiglia, "Partially grounded planning as quantified boolean formula," in *ICAPS*. AAAI, 2013.

[14] U. Egly, M. Kronegger, F. Lonsing, and A. Pfandler, "Conformant planning as a case study of incremental QBF solving," *Ann. Math. Artif. Intell.*, vol. 80, no. 1, pp. 21–45, 2017.

[15] O. Gasquet, D. Longin, F. Maris, P. Régnier, and M. Valais, "Compact tree encodings for planning as QBF," *Inteligencia Artificial*, vol. 21, no. 62, pp. 103–114, 2018.

[16] M. Cashmore, M. Fox, and E. Giunchiglia, "Planning as quantified boolean formula," in *ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. 242. IOS Press, 2012, pp. 217–222.

[17] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds., *Handbook of Model Checking*. Springer, 2018.

[18] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking without bdds," in *TACAS*, ser. LNCS, vol. 1579. Springer, 1999, pp. 193–207.

[1] http://www.qbfeval.org

[19] A. Biere and D. Kröning, "Sat-based model checking," in *Handbook of Model Checking*. Springer, 2018, pp. 277–303.

[20] N. Dershowitz, Z. Hanna, and J. Katz, "Bounded model checking with QBF," in *SAT*, ser. LNCS, vol. 3569. Springer, 2005, pp. 408–414.

[21] T. Jussila and A. Biere, "Compressing BMC encodings with QBF," *Electr. Notes Theor. Comput. Sci.*, vol. 174, no. 3, pp. 45–56, 2007.

[22] H. Mangassarian, A. G. Veneris, and M. Benedetti, "Robust QBF encodings for sequential circuits with applications to verification, debug, and test," *IEEE Trans. Computers*, vol. 59, no. 7, pp. 981–994, 2010.

[23] M. N. Mneimneh and K. A. Sakallah, "Computing vertex eccentricity in exponentially large graphs: QBF formulation and solution," in *SAT*, ser. LNCS, vol. 2919. Springer, 2003, pp. 411–425.

[24] M. Herbstritt and B. Becker, "On combining 01x-logic and QBF," in *EUROCAST*, ser. LNCS, vol. 4739. Springer, 2007, pp. 531–538.

[25] M. Benedetti and H. Mangassarian, "Qbf-based formal verification: Experience and perspectives," *JSAT*, vol. 5, no. 1-4, pp. 133–191, 2008.

[26] H. K. Büning and T. Lettmann, "Quantifizierte formeln," in *Aussagenlogik: Deduktion und Algorithmen*. Springer, 1994, pp. 361–408.

[27] U. Egly, T. Eiter, H. Tompits, and S. Woltran, "Solving advanced reasoning tasks using quantified boolean formulas," in *AAAI/IAAI*. AAAI Press / The MIT Press, 2000, pp. 417–422.

[28] A. Ignatiev, A. Morgado, and J. Marques-Silva, "Propositional abduction with implicit hitting sets," in *ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. 285. IOS Press, 2016, pp. 1327–1335.

[29] R. C. Moore, "Semantical considerations on nonmonotonic logic," *Artif. Intell.*, vol. 25, no. 1, pp. 75–94, 1985.

[30] U. Egly, T. Eiter, V. Klotz, H. Tompits, and S. Woltran, "Computing stable models with quantified boolean formulas: Some experimental results," in *Answer Set Programming*, 2001.

[31] ——, "Experimental evaluation of the disjunctive logic programming module of the system QUIP," in *WLP*, 2000, pp. 113–122.

[32] J. McCarthy, "Circumscription—a form of non-monotonic reasoning," *Artificial intelligence*, vol. 13, no. 1-2, pp. 27–39, 1980.

[33] R. Reiter, "A logic for default reasoning," *Artif. Intell.*, vol. 13, no. 1-2, pp. 81–132, 1980.

[34] D. Poole, "Explanation and prediction: an architecture for default and abductive reasoning," *Computational Intelligence*, vol. 5, no. 2, pp. 97–110, 1989.

[35] G. Charwat, W. Dvorák, S. A. Gaggl, J. P. Wallner, and S. Woltran, "Methods for solving reasoning problems in abstract argumentation - A survey," *Artif. Intell.*, vol. 220, pp. 28–63, 2015.

[36] U. Egly and S. Woltran, "Reasoning in argumentation frameworks using quantified boolean formulas," in *COMMA*, ser. Frontiers in Artificial Intelligence and Applications, vol. 144. IOS Press, 2006, pp. 133–144.

[37] O. Arieli and M. W. A. Caminada, "A qbf-based formalization of abstract argumentation semantics," *J. Applied Logic*, vol. 11, no. 2, pp. 229–252, 2013.

[38] M. Diller, J. P. Wallner, and S. Woltran, "Reasoning in abstract dialectical frameworks using quantified boolean formulas," *Argument & Computation*, vol. 6, no. 2, pp. 149–177, 2015.

[39] S. Cooksey, S. Harris, M. Batty, R. Grigore, and M. Janota, "Pridemm: A solver for relaxed memory models," *CoRR*, vol. abs/1901.00428, 2019.

[40] R. Wimmer, A. Karrenbauer, R. Becker, C. Scholl, and B. Becker, "From DQBF to QBF by dependency elimination," in *SAT*, ser. LNCS, vol. 10491. Springer, 2017, pp. 326–343.

[41] A. Ignatiev, M. Janota, and J. Marques-Silva, "Quantified maximum satisfiability," *Constraints*, vol. 21, no. 2, pp. 277–302, 2016.

[42] B. Cook, D. Kroening, and N. Sharygina, "Symbolic model checking for asynchronous boolean programs," in *SPIN*, ser. LNCS, vol. 3639. Springer, 2005, pp. 75–90.

[43] ——, "Verification of boolean programs with unbounded thread creation," *Theor. Comput. Sci.*, vol. 388, no. 1-3, pp. 227–242, 2007.

[44] R. Bloem, R. Könighofer, and M. Seidl, "SAT-Based Synthesis Methods for Safety Specs," in *VMCAI*, ser. LNCS, vol. 8318. Springer, 2014, pp. 1–20.

[45] S. Staber and R. Bloem, "Fault localization and correction with QBF," in *SAT*, ser. LNCS, vol. 4501. Springer, 2007, pp. 355–368.

[46] M. F. Ali, S. Safarpour, A. G. Veneris, M. S. Abadir, and R. Drechsler, "Post-verification debugging of hierarchical designs," in *ICCAD*. IEEE Computer Society, 2005, pp. 871–876.

[47] A. Sülflow, G. Fey, and R. Drechsler, "Using QBF to increase accuracy of sat-based debugging," in *ISCAS*. IEEE, 2010, pp. 641–644.

[48] S. Hillebrecht, M. A. Kochte, D. Erb, H. Wunderlich, and B. Becker, "Accurate QBF-based test pattern generation in presence of unknown values," in *DATE*. EDA Consortium San Jose, CA, USA / ACM DL, 2013, pp. 436–441.

[49] P. Faymonville, B. Finkbeiner, M. N. Rabe, and L. Tentrup, "Encodings of bounded synthesis," in *TACAS (1)*, ser. LNCS, vol. 10205, 2017, pp. 354–370.

[50] C. Otwell, A. Remshagen, and K. Truemper, "An effective QBF solver for planning problems," in *MSV/AMCS*. CSREA Press, 2004, pp. 311–316.

[51] C. Ansótegui, C. P. Gomes, and B. Selman, "The achilles' heel of QBF," in *AAAI*. AAAI Press / The MIT Press, 2005, pp. 275–281.

[52] Diptarama, R. Yoshinaka, and A. Shinohara, "QBF encoding of generalized tic-tac-toe," in *QBF@SAT*, ser. CEUR Workshop Proceedings, vol. 1719. CEUR-WS.org, 2016, pp. 14–26.

[53] I. P. Gent and A. Rowley, "Encoding connect-4 using quantified boolean formulae," *2nd Intl. Work. Modelling and Reform. CSP*, pp. 78–93, 2003.

[54] T. Eiter, H. Tompits, and S. Woltran, "On solution correspondences in answer-set programming," in *IJCAI*. Professional Book Center, 2005, pp. 97–102.

[55] J. Oetsch, M. Seidl, H. Tompits, and S. Woltran, "cct: A correspondence-checking tool for logic programs under the answer-set semantics," in *JELIA*, ser. LNCS, vol. 4160. Springer, 2006, pp. 502–505.

[56] ——, "cct on stage: Generalised uniform equivalence testing for verifying student assignment solutions," in *LPNMR*, ser. LNCS, vol. 5753. Springer, 2009, pp. 382–395.

[57] P. Bera, P. Dasgupta, and S. Ghosh, "A verification framework for analyzing security implementations in an enterprise lan," in *2009 IEEE International Advance Computing Conference*. IEEE, 2009, pp. 1008–1015.

[58] J. Millo, S. Ramesh, S. N. Krishna, and G. K. Narwane, "Compositional verification of software product lines," in *IFM*, ser. LNCS, vol. 7940. Springer, 2013, pp. 109–123.

[59] T. Heyman, D. Smith, Y. Mahajan, L. Leong, and H. Abu-Haimed, "Dominant Controllability Check Using QBF-Solver and Netlist Optimizer," in *SAT*, ser. LNCS, vol. 8561. Springer, 2014, pp. 227–242.

[60] H. Chen, M. Janota, and J. Marques-Silva, "Qbf-based boolean function bi-decomposition," in *DATE*. IEEE, 2012, pp. 816–819.

[61] R. Wille, H. M. Le, G. W. Dueck, and D. Große, "Quantified synthesis of reversible logic," in *DATE*. ACM, 2008, pp. 1015–1020.