Electronic Communications of the EASST
Volume 70 (2014)

Proceedings of the
14th International Workshop on
Automated Verification of Critical Systems (AVoCS 2014)

QBF with Soft Variables

Sven Reimer, Matthias Sauer, Paolo Marin, Bernd Becker

15 pages

Guest Editors: Marieke Huisman, Jaco van de Pol
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

QBF with Soft Variables

Sven Reimer, Matthias Sauer, Paolo Marin, Bernd Becker

Institute of Computer Science, Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee 051, D-79110 Freiburg, Germany
{reimer|sauerm|marin|becker}@informatik.uni-freiburg.de

Abstract: QBF formulae are usually considered in prenex form, i.e. the quantifier block is completely separated from the propositional part of the QBF. Among others, the semantics of the QBF is defined by the sequence of the variables within the prefix, where existentially quantified variables depend on all universally quantified variables stated to the left.

In this paper we extend that classical definition and consider a new quantification type which we call *soft variable*. The idea is to allow a flexible position and quantifier type for these variables. Hence the type of quantifier of the soft variable can also be altered. Based on this concept, we present an optimization problem seeking an optimal prefix as defined by user-given preferences. We state an algorithm based on MaxQBF, and present several applications – mainly from verification area – which can be naturally translated into the optimization problem for QBF with soft variables. We further implemented a prototype solver for this formalism, and compare our approach to previous work, that differently from ours does not guarantee optimality and completeness.

Keywords: QBF, MaxQBF, prefix, dependency, optimization problem

1 Introduction

For design automation tasks in safety-critical or other domains where precise answers are necessary, applications employing quantified Boolean formulae (QBF) logic have been demonstrated to be an effective solution: In contrast to the traditional SAT-based 01X-encoding [JBM+00], QBF delivers accurate answers by accurately considering unknown and unspecified signals [SB01], which is also named Z_i-encoding [HB07]. In particular universally quantified variables in QBF are used to accurately model the behavior of unknown circuit lines. The encoding of such problems is often not trivial and the effort required to solve them strongly depends on the prefix order, i. e. the user-given dependencies between the existential and universal variables. Additionally, there is an increasing interest for optimization problems with QBF [BLV08, IJM13] in the game theory domain [CP04]. However, in the classical definition of a QBF, the prefix is known and fixed. Research interests considering the possibility of changing the given prefix structure is focused on simplifying the solving to a given QBF without changing its meaning, e.g. by optimizing decision strategies in search-based solving [GNT07]. More generalized approaches apply so-called dependency schemes [Sam08] focusing on tractable algorithms, i. e. heuristics with polynomial complexity, optimizing the prefix order wrt. the dependencies between the variables.

As we will show in this paper, there is a relation between the prefix order and the optimization problem for maximizing unknown values within a circuit [RS04, NSB07, PR00]. E. g., one seeks for a maximum number of don't care signals at the input of a circuit in order to generalize a found solution [RS04]. Since in QBF these don't care or unknown values are modeled by universally quantified variables, one needs the possibility to change the quantifier for a particular variable from existential to universal and vice versa in order to optimize the number of unknowns. In pure QBF, quantifiers have to be given a-priori and cannot be changed.

This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS).

In this paper we present a new formalism based on QBF supporting a dynamic prefix using so-called *soft variables*. In contrast to previous perceptions, our approach allows to change the prefix position of such soft variables dynamically. In particular, the quantifier from existential to universal can be altered and vice versa. Technically the mechanism provides a set of possible QBF prefixes. The challenge is to find a satisfiability preserving prefix such that the soft variables are quantified following an ordering as close as possible to user-given preferences.

We define an optimization problem for QBF with soft variables and an algorithm returning the optimal prefix. The algorithm is based on the optimization problem called MaxQBF [CFLS93]. We highlight the importance of our formalism by showing applications from different verification areas, which are naturally covered by our formalism. Previous work on these applications has two main disadvantages:

- Most approaches use a SAT-based encoding, which is less precise than a QBF represenation (with soft variables).
- Applications using QBF formulations usually need more encoding effort than approximate SAT encodings. As we will show, the optimization problem for solving QBF with soft variables is PSPACE-complete (as QBF), but allows a more compact and in many cases "easier" representation of the problem statement than pure QBF.

We developed a prototype solver for solving both: 1) MaxQBF based on [IJM13] and 2) QBF with soft variables based on MaxQBF. First experimental results demonstrate the applicability of our implementation and the advantage of the solution compared to heuristic approaches.

The paper is structured as follows: In §2 we introduce basic information, terminology, and notation used throughout the paper. In §3 the concept of QBF with soft variables is introduced as well as an algorithm to solve the problem. We introduce applications for the formalism and first experimental results for some of those in §4. Lastly, §5 concludes the paper and discusses future work on this topic.

2 Preliminaries

In this section we introduce the notation and some background on solving techniques, and some further details necessary for a good comprehension of this paper. We assume that propositional logic and the SAT problem is familiar to the reader. The interested reader is referred to [BHMW09] for further insight in optimization problems in propositional logic and QBF.

2.1 QBF

The logic of quantified Boolean formulae (QBF) is an extension of SAT by bounding the variables to quantifiers $\mathcal{Q} \in \{\exists, \forall\}$. In the following, we will consider prenex conjunctive normal form formulae (PCNF) $\psi = \mathcal{Q}_1 X_1 \ldots \mathcal{Q}_n X_n . \varphi$, where φ is a quantifier free matrix in CNF. We denote with $\mathcal{V}$ the set of variables occurring in φ. We call $P = \mathcal{Q}_1 X_1 \ldots \mathcal{Q}_n X_n$ the prefix of ψ. W.l.o.g. $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$ for all $i \in \{1, \ldots, n-1\}$, i. e. the existential and universal quantifiers have an alternating order, and $X_1, \ldots, X_n$ are disjoint sets of variables with $X_1 \bigcup \ldots \bigcup X_n := X$.[1]

We define a quantifier level function $\delta : X \to \mathbb{N}$ for a variable x as $\delta(x) = i$ for $x \in X_i$. In the following, w.l.o.g. we set $\mathcal{Q}_1 = \exists$, in particular, variables on odd quantification levels are always existentially quantified and on even levels universally quantified. We say that an existential variable $x \in X_i$ depends on all universal variables $y \in X_j$ with $j < i$ and $\mathcal{Q}_j = \forall$, i. e. the assignment of x depends on the assignment of y.

A variable $x \in W = \mathcal{V} \setminus X$ is not bound by any quantifier and is called free variable. If ψ contains free variables, then ψ is an open QBF, otherwise it is a closed formula. Given a set of variables $\mathcal{V}$

[1] Note that $X_i = \emptyset$ or $X \neq \mathcal{V}$ may hold

and an open QBF ψ, we say that some variable valuation $f \in (W \to \mathbb{B})$ is a model of ψ iff $\psi(f) = \top$, and $\psi(W)$ is satisfiability equivalent to the closed QBF where the free variables W are existentially quantified at level 1.

The co-factor of a propositional formula is defined as $\varphi|_x = \varphi[x = \top]$ and $\varphi|_{\overline{x}} = \varphi[x = \bot]$ respectively, where $\varphi[x = c]$ denotes a propositional formula with every occurrence of x replaced by the value $c \in \{\top, \bot\}$. Similarly, $\psi|_x$ is defined as the QBF where the matrix φ is replaced by $\varphi|_x$ and all superfluous quantifications are excluded from the prefix (and analogously for $\psi|_{\overline{x}}$). We define the semantics of QBF recursively as follows:

$$\mathcal{Q}x\ \psi = \begin{cases} \psi|_x \wedge \psi|_{\overline{x}} & \text{if } \mathcal{Q} = \forall \\ \psi|_x \vee \psi|_{\overline{x}} & \text{if } \mathcal{Q} = \exists \end{cases}$$

If φ does contain an empty clause, ψ is unsatisfied, otherwise satisfied in case no variable is left unassigned.

2.2 MaxQBF

MaxQBF (also referred to as MaxQSAT in the literature) is an extension of QBF for optimization problems [CFLS93]. The semantics is similar to the analogous optimization problem for SAT, called MaxSAT. A MaxSAT procedure tries to satisfy as many clauses of a propositional formula as possible. We denote these clauses as soft clauses. The optimization problem for QBF ensures that these soft clauses have to be satisfied for *every* branch of the universal variables. There are some extensions of MaxSAT which can be adapted quite naturally to MaxQBF:

- Weighted MaxSAT/QBF: Each soft clause c is associated with a non-negative integer weight $\omega(c)$. If a clause is satisfied, the clause gains $\omega(c)$ as score, otherwise the score of the clause is 0. The objective is to maximize the sum of the scores.
- Partial MaxSAT/QBF: There are two types of clauses: hard clauses and soft clauses. Hard clauses must be satisfied, while soft clauses may be satisfied. The objective is to maximize the number of satisfied soft clauses.
- Weighted Partial MaxSAT/QBF: Combination of the two concepts stated above.

To the best of our knowledge there exist only two references proposing exact algorithms to solve optimization problems over quantified formulae: [BLV08] considers quantified constraint optimization problems (QCOP) as extension of constraint optimization problems (COP). In [IJM13] the authors propose two algorithms for solving the related QMaxSAT problem: Given a QBF instead of optimizing the number of satisfied soft clauses, the goal is to maximize a linear pseudo-Boolean cost function. The MaxQBF problem can be easily expressed by this formalism and vice versa.

3 QBF with Soft Variables

In this section we describe the concept of soft variables in QBF, as well as an algorithmic approach for solving this formalism using MaxQBF, and an algorithm for solving MaxQBF problems. Finally, we state some details of our implementations.

3.1 Soft Variables

The syntax of soft variables in the context of a QBF is defined as follows.

Definition 1 Consider a QBF $\psi = P.\varphi$, a Boolean variable $s \in \mathcal{V}$ and a set of natural numbers L. We call s a soft variable iff the following properties hold:

1. s does not occur in P, and
2. L is a set of quantification levels on which s is designated to be quantified, also called the *quantification set of* s.

We denote with $\mathcal{H}_L s$ a soft variable s, where L contains all possible prefix positions. Furthermore we write $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)$, indicating that $s_1, \ldots, s_n$ are soft variables of ψ with possible prefix positions L^j, for each $j \in \{1, \ldots, n\}$. We write S_ψ indicating the set of all soft variables in ψ.

Moreover, we allow soft variable groups S, written $\mathcal{H}_L S$. We write $\psi(\mathcal{H}_{L^1} S_1, \ldots, \mathcal{H}_{L^n} S_n)$ for a QBF ψ with different groups of soft variables S^j. We also allow combinations of soft variables and groups of soft variables within a QBF. In the following we distinguish the two cases by using capital letters for groups and small letters for variables.

Definition 2 Let $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)$ be a QBF with soft variables. We call $\Lambda : \mathcal{S}_\psi \to \mathbb{N}$ the level function of ψ which maps each variable s_j to a quantification level $l \in L^j$. Given such a level function Λ we denote $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)_\Lambda$ as the QBF where every soft variable is mapped to one possible quantification level within P of ψ according to Λ. For each soft variable of one soft variable group $s_j \in S$ the level function has to be same value, i. e. $s_j = l$ for a fixed level l and for all $s_j \in S$.

Intuitively $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)$ is a set of QBF with different prefixes for the soft variables, to be more precise every QBF resulting from every possible level function Λ of ψ.

Example 1 *Consider the QBF with soft variables* ψ_1 *with a matrix* $\varphi_1 = (s_1 \vee y) \wedge (s_1 \vee z) \wedge (\overline{y} \vee z) \wedge (\overline{s_1} \vee \overline{y})$*, where* s_1 *is a soft variable in the scope of all existential levels:*

$$\psi_1(\mathcal{H}_{\{1,3\}} s_1) = \forall y \exists z.(s_1 \vee y) \wedge (s_1 \vee z) \wedge (\overline{y} \vee z) \wedge (\overline{s_1} \vee \overline{y})$$

By definition we are allowed to set s_1 *either to the first or third (i. e. an existential) level, resulting in the two possible level functions with* $\Lambda_1(s_1) = 1$ *and* $\Lambda_2(s_1) = 3$*, and therefore in two possible prefixes* $P_1 = \exists s_1 \forall y \exists z$ *and* $P_2 = \forall y \exists z \exists s_1$*. The QBF* $\psi_1(\mathcal{H}_{\{1,3\}} s_1)_{\Lambda_1} = P_1.\varphi_1$ *is unsatisfiable, whereas* $\psi_1(\mathcal{H}_{\{1,3\}} s_1)_{\Lambda_2} = P_2.\varphi_1$ *results in a satisfied matrix for all branches of the universal variable* y*, i. e. is satisfied.*

Now consider the formula ψ_2 *with a matrix* $\varphi_2 = (y \vee z) \wedge (s_2 \vee z) \wedge (\overline{s_2} \vee \overline{y} \vee \overline{z})$ *where* s_2 *is a soft variable with level* 2 *and* 3 *as possible prefix positions:*

$$\psi_2(\mathcal{H}_{\{2,3\}} s_2) = \forall y \exists z.(y \vee z) \wedge (s_2 \vee z) \wedge (\overline{s_2} \vee \overline{y} \vee \overline{z})$$

By applying the soft variable concept we obtain two possible level functions with $\Lambda_3(s_2) = 2$ *and* $\Lambda_4(s_2) = 3$ *and the respective prefixes* $P_3 = \forall y \forall s_2 \exists z$ *and* $P_4 = \forall y \exists z \exists s_2$*. Both* $\psi_2(\mathcal{H}_{\{2,3\}} s_2)_{\Lambda_3} = P_3.\varphi_2$ *and* $\psi_2(\mathcal{H}_{\{2,3\}} s_2)_{\Lambda_4} = P_4.\varphi_2$ *are satisfiable.*

Based on this syntax we briefly define the semantics of a QBF with soft variables.

Definition 3 A QBF with soft variables $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)$ is satisfied iff there exists a level function Λ such that $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)_\Lambda$ is satisfied. If for all possible level functions the resulting QBF is unsatisfied, we say $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)$ is unsatisfiable.

We want to consider an optimization problem for QBF with soft variables. To do so, we first define a score function σ as follows:

Definition 4 Let $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)$ be a QBF with soft variables $s_1, \ldots, s_n$. The score function $\sigma : (S_\psi \times \mathbb{N}) \to \mathbb{N}$ is defined for each variable s_j as $\sigma(s_j, l) = \chi_{s_j,l}$, where $\chi_{s_j,l}$ is a user-given score

and l_j the corresponding level with $\Lambda(s_j) = l_j$. The overall score χ_Λ is the sum of all scores for one level function Λ: $\chi_\Lambda = \sum_{j=1}^{n} \chi_{s_j,l_j}$.

The score function allows to define the optimization problem $\Omega(\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n))$ as follows:

Definition 5 Given a QBF with soft variables $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)$ and a score function σ of ψ, the optimization problem $\Omega(\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n))$ is to find a level function Λ maximizing the score χ_Λ such that $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)_\Lambda$ is satisfied, i. e. $\Omega(\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)) = \max_\Lambda \chi_\Lambda$. If $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n)_\Lambda$ is unsatisfied, the score is $\chi_\Lambda = 0$.

Example 2 Consider the QBF ψ_1 of Example 1. We chose the following scores: $\sigma(s_1, 1) = 2$ and $\sigma(s_1, 3) = 1$. The QBF $\psi_1(\mathcal{H}_{\{1,3\}} s_1)_{\Lambda_1}$ is unsatisfied, and therefore the score evaluates to $\chi_{\Lambda_1} = \chi_{s_1,1} = 0$. Since $\psi_1(\mathcal{H}_{\{1,3\}} s_1)_{\Lambda_2}$ is satisfied, we yield the score $\chi_{\Lambda_2} = \chi_{s_1,3} = 1$, which is also the maximum score over all possible level functions Λ of ψ_1 and thus the result of the maximization problem $\Omega(\psi(\mathcal{H}_{\{1,3\}} s_1))$ is 1 with Λ_2.

Considering ψ_2 from Example 1, we chose the scores $\sigma(s, 2) = 2$ and $\sigma(s, 3) = 1$. $\psi_2(\mathcal{H}_{\{2,3\}} s_2)_{\Lambda_3}$ as well as $\psi_2(\mathcal{H}_{\{2,3\}} s_2)_{\Lambda_4}$ are satisfied and gain the scores $\chi_{\Lambda_3} = \chi_{s_2,2} = 2$ and $\chi_{\Lambda_4} = \chi_{s_2,3} = 1$. Hence, we obtain $\Omega(\psi_2(\mathcal{H}_{\{2,3\}} s_2)) = 2$ with level function Λ_3 as solution to the optimization problem.

Please note, in this example we chose the weights for both formulae in a meaningful way: The QBF where the soft variable is quantified on a level which is more likely to be unsatisfiable gets the higher score.

Proposition 1 *Solving the optimization problem $\Omega(\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n))$ is PSPACE-complete.*

Proof. (Sketch) To decide the optimization problem we need to solve a QBF problem for each possible level function. QBF is in PSPACE [SM73] and since we only need to store the currently optimal level function and its score, which needs polynomial space, the optimization problem for QBF with soft variables is also in PSPACE.

For PSPACE-hardness we reduce from QBF which is PSPACE-hard [SM73]. Let ψ be a QBF $\psi = \mathcal{Q}_1 X_1 \ldots \mathcal{Q}_n X_n . \varphi$. Let m be a new existential level with $m > n$. We define a QBF with soft variables: $\psi'(\mathcal{H}_{\{1,m\}} X_1, \ldots, \mathcal{H}_{\{n,m\}} X_n) = \varphi$, i. e. ψ' is obtained from ψ by eliminating P and declaring each variable in X as part of a soft variable group X_i. The possible prefix positions are chosen such that the whole group X_i is either quantified at level i as in ψ, or at the new level m. We define the score function σ such that for each soft variable group X_i both $\sigma(X_i, m) = 0$ and $\sigma(X_i, i) = 1$ holds, i. e. the preference is to set the group to the level it belonged to in ψ. The maximization problem for QBF with soft variables tries to set all groups of ψ' to the corresponding position in ψ. From the result of the QBF with soft variables problem $\Omega(\psi') = \max_\Lambda \chi_\Lambda$ we directly obtain the satisfiability value of ψ: If at least one soft variable group is not set at level $i < m$ in λ of ψ' or $\max_\Lambda \chi_\Lambda = 0$ holds, the QBF ψ is unsatisfiable. Otherwise, if all groups X_i are set to level i, the QBF ψ is satisfiable. □

3.2 Algorithm

In this section we describe an algorithm to solve the optimization problem for QBF with soft variables which is based on Weighted Partial MaxQBF. To do so, we present an approach for solving MaxQBF (and its extensions) based on iterative MaxSAT algorithms.

3.2.1 Solving QBF with Soft Variables

Let $\psi(\mathcal{H}_{L^1} s_1, \ldots, \mathcal{H}_{L^n} s_n) = P.\varphi$ be a QBF with soft variables $s_1, \ldots, s_n$, P a prefix over the variables X, and φ a quantifier free matrix over the variables $\mathcal{V}$. In the following we show how to transform

this problem into a Weighted Partial MaxQBF problem. Therefore, we need to transform our formalism into a QBF without any soft variables.

We denote $e_{\max}$ of ψ as the innermost existential level with no further (existential or universal) quantification level right of $e_{\max}$. Hence, $e_{\max}$ is the right-most existential level wrt. the current prefix P and all levels $l \in L^j$ for all $j \in \{1,\ldots,n\}$.

The following extensions/modifications have to be applied for each soft variable s_j, $j \in \{1,\ldots,n\}$: We quantify s_j on level $e_{\max}$ existentially such that the appearances of s_j in the matrix is well-defined. For each level $l \in L^j$ we introduce a new quantification of a helper variable s_j^l on level l and a new free variable f_j^l of ψ. These helper variables allows us to alter the quantification levels by setting f_j^l appropriately. Therefore, we connect these variables to φ by the constraint $f_j^l \Rightarrow (s_j^l \equiv s_j)$ for each $l \in L^j$, i. e. we set s_j to s_j^l (and hence s_j quantified on level $e_{\max}$ semantically "behaves" like the variable s_j^l quantified on level l) if the free variable f_j^l is set to $\top$. We have to ensure that exactly one f_j^l for all $l \in L^j$ is set to $\top$ and all other f_j^k, $k \neq l$ are set to $\bot$, which is also known as an exactly-one constraint. We add the encoding of such a constraint for each new free variable of every soft variable to φ. We declare every clause of φ as well as every additional constraint we introduced so far as hard clause of the Weighted Partial MaxQBF instance. Finally, we add an additional unit clause for each free variable f_j^l, declared as soft clause with weight $\sigma(s_j, l)$.

A Weighted Partial MaxQBF solver maximizes these weights, i. e. the score function σ is directly mapped to a the maximum number of satisfied unit soft clauses triggering a soft variable to be set on the corresponding position in the prefix. For the free variables f_j^l we obtain a model from which we can extract the level function Λ: if f_j^l is set to $\top$, $\Lambda(s_j) = l$ holds. By construction, χ_Λ is the maximum score of $\psi(\mathrel{\mathrlap{\forall}\exists}_{L^1} s_1, \ldots, \mathrel{\mathrlap{\forall}\exists}_{L^n} s_n)$.

Remark 1 *Handling groups S of soft variables is analogous to the method described above. Instead of introducing a free variable f_j^l for each variable and level, we only have to introduce one f_S^l for each level and group, i. e. every variable of a group S is either quantified simultaneously on l or not. The additionally introduced constraint is $f_S^l \Rightarrow (s_j^l \equiv s_j)$ for each $s_j \in S$.*

Example 3 *Consider ψ_1 from our running example. First, we introduce the soft variable $\mathrel{\mathrlap{\forall}\exists}_{\{1,3\}} s_1$ into P by introducing an existential quantification $\exists s_1$ on level $e_{max} = 3$. By definition, $\mathrel{\mathrlap{\forall}\exists}_{\{1,3\}} s_1$ can be quantified on both existential levels 1 and 3, therefore we introduce two existential helper variables s_1^1 and s_1^3 as well as two new free variables f_1^1 and f_1^3. Moreover, we introduce a CNF representation for the constraints $f_1^1 \Rightarrow (s_1^1 \equiv s_1)$ and $f_1^3 \Rightarrow (s_1^3 \equiv s_1)$, the constraints for the exactly-one-constraint, and the unit clauses for the free variables into φ_1, resulting in:*

$$\exists s_1^1 \forall y \exists s_1^3 \exists z \exists s_1. \quad \overbrace{(s_1 \vee y) \wedge (s_1 \vee z) \wedge (\overline{y} \vee z) \wedge (\overline{s_1} \vee \overline{y})}^{\text{original } \varphi_1} \wedge \underbrace{(f_1^1 \vee s_1^1 \vee \overline{s_1}) \wedge (\overline{f_1^1} \vee \overline{s_1^1} \vee s_1)}_{f_1^1 \Rightarrow (s_1^1 \equiv s_1)} \wedge \underbrace{(\overline{f_1^3} \vee s_1^3 \vee \overline{s_1}) \wedge (\overline{f_1^3} \vee \overline{s_1^3} \vee s_1)}_{f_1^3 \Rightarrow (s_1^3 \equiv s_1)} \wedge \underbrace{(\overline{f_1^1} \vee \overline{f_1^3}) \wedge (f_1^1 \vee f_1^3)}_{\text{exactly-one}} \wedge \underbrace{(f_1^1) \wedge (f_1^3)}_{\text{soft clauses}}$$

This instance is passed to a Weighted Partial MaxQBF solver, where (f_1^1) and (f_1^3) are declared as soft clauses with weight 2 and 1 respectively. All other clauses are declared as hard clauses.

Analogously we obtain for ψ_2 the following Weighted Partial MaxQBF instance:

$$\begin{aligned}\forall s_2^2 \forall y \exists s_2^3 \exists z \exists s_2. \quad & (y \vee z) \wedge (s_2 \vee z) \wedge (\overline{s_2} \vee \overline{y} \vee \overline{z}) \wedge \\ & (\overline{f_2^2} \vee s_2^2 \vee \overline{s_2}) \wedge (\overline{f_2^2} \vee s_2^2 \vee s_2) \wedge (\overline{f_2^3} \vee s_2^3 \vee \overline{s_2}) \wedge (\overline{f_2^3} \vee \overline{s_2^3} \vee s_2) \wedge \\ & (\overline{f_2^2} \vee \overline{f_2^3}) \wedge (f_2^2 \vee f_2^3) \wedge (f_2^2) \wedge (f_2^3)\end{aligned}$$

with the soft clauses (f_2^2) *and* (f_2^3) *and weights* 2 *and* 1*, respectively.*

3.2.2 Solving MaxQBF

In this section we present an approach for solving MaxQBF and its extensions based on iterative methods from MaxSAT [ZSM03]. The method is also mentioned in [IJM13], but we present some further details. First, we describe the iterative approach in MaxSAT and then we introduce how to adapt this technique for MaxQBF.

Iterative MaxSAT solvers as introduced in [ZSM03] add a new relaxation literal r to each soft clause. If r is set to $\bot$, the corresponding soft clause has to be satisfied, otherwise $(r = \top)$ the clause is *relaxed*, i. e., it is satisfied by the relaxation literal and hence, the (original) soft clause does not have to be satisfied by the set of variables belonging to the original CNF. Let φ be the original MaxSAT instance with m soft clauses including a unique relaxation literal $(r_1, \ldots, r_m)$ for each soft clause. In an iterative approach the relaxation literals are connected to the inputs of a cardinality network [Sin05] and the instance $\varphi \wedge \beta(r_1, \ldots, r_m) \wedge (\overline{o_i})$ is handed over to a SAT solver, where $\beta(r_1, \ldots, r_m)$ is the CNF encoding for such a cardinality network, with $r_1, \ldots, r_m$ as inputs and $o_1, \ldots, o_m$ as outputs. The additional unit clause $(\overline{o_i})$ demands at least i arbitrary inputs of the network to be set to $\bot$, therefore i soft clauses have to be satisfied. If this SAT instance is satisfiable, there are at least i simultaneously satisfied soft clauses, otherwise there exists no solution with i satisfied soft clauses. The solution is narrowed by the incremental usage of the underlying SAT solver until a value k is identified with $\varphi \wedge \beta(r_1, \ldots, r_m) \wedge (\overline{o_k})$ being satisfiable and $\varphi \wedge \beta(r_1, \ldots, r_m) \wedge (\overline{o_{k+1}})$ being unsatisfiable. This value k is the maximum number of simultaneously satisfied soft clauses.

This procedure can be easily extended to the Partial MaxSAT and Weighted MaxSAT concepts: In Partial MaxSAT, hard clauses are not connected to the network, and in Weighted MaxSAT a soft clause c is connected $\omega(c)$ times into the network, where $\omega(c)$ is the weight of the clause c. The algorithm for Weighted Partial MaxSAT is obtained by a straight-forward combination of both methods.

We adapt this concept for MaxQBF by using an incremental QBF solver. The cardinality network and the relaxation variables are encoded as in MaxSAT. The variables for the encoding are added as free variables of the QBF, since a soft clause has to be satisfied for every branch of the universal variables and we are able to obtain a model for these variables as a result from solving the open QBF. Likewise to MaxSAT we call the QBF solver incrementally in order to find a value k such that the QBF with k satisfied soft clauses is satisfied, but unsatisfied with $k+1$ clauses. The extension for Partial MaxQBF, Weighted MaxQBF and Weighted Partial MaxQBF is done in the same manner as for the MaxSAT extension.

3.2.3 Implementation Details

We implemented a Weighted Partial MaxQBF solver as well as the algorithm for solving QBF with soft variables.

For the MaxQBF solver we used the QBF solver `quantom` [RPSB12] and implemented incremental functionality based on [MMB12]. For the model of the free variables (representing the level function Λ) we adapt techniques of [BELM12]. For scalability reasons we also use a preprocessor for QBF. The preprocessor is implemented for incremental usage (see also [MMLB12]) as well as model preservation techniques known from SAT [EB05] adapted to QBF. All variables which are introduced in context of a soft variable are set as "Don't touch"[KLSB11] in the preprocessor, i. e. these variable are excluded from several preprocessing techniques, among others variable elimination and pure literal detection.

For an exactly-one-constraint with more than 5 possible levels (i. e. $|L^j| > 5$), we use the encoding for $\mathrm{LT}_{\mathrm{SEQ}}^{n,1}$ as presented in [Sin05] using $\mathcal{O}(|L^j|)$ clauses and $\mathcal{O}(|L^j|)$ additional auxiliary variables

only. Otherwise we introduce a standard one-hot encoding using $\mathcal{O}(|L^j|^2)$ additional clauses.[2]

For some instances (cf. §4.1) the encoding can be simplified to a pure Partial MaxQBF problem if the following requirements hold: 1) the soft variables are only defined over two different levels and 2) for all soft variables the weight function assigns for one level 0 and for the other level a constant value $c > 0$. If this is the case we can add just one free variable per soft variable with the constraints: $f_j^l \Rightarrow (s_j^l \equiv s_j)$ and $\overline{f_j^l} \Rightarrow (s_j^k \equiv s_j)$, where l is the preferred level with weight c and k is the level with weight 0. Moreover, just one soft clause (f_j^l) for each soft variable is added.

4 Applications

In this section we present several applications, mainly from verification and testing of circuits, covered by QBF with soft variables. In §4.1 we present optimization problems considering unknowns in a circuit, and in §4.2 we present further applications, for example optimal solutions for dependency schemes.

The measurements for all case studies were performed on a machine using one core of a 3.3 GHz Intel Xeon, and limiting the memory to 4 GB.

4.1 Maximizing Unknowns in a Circuit

There are plenty of applications which ask for maximizing or optimizing unknown values within a circuit. There are two general problem statements, given a circuit together with a property which has to hold, we want to optimize: 1) The solution by introducing unknown values for the internal circuit lines or the inputs, and 2) secondary objectives in presence of unspecified values such that the property still holds. In the first case the unknown values generalize (or uniform) the solution, whereas in the second case some parts of the circuit are abstracted by introducing universal quantifications in an appropriate QBF and we want to optimize further objectives in presence of these unknowns.

In the following part of this section we briefly review classical algorithms discussing differences to the QBF with soft variable formulation. Finally, we introduce specific problem statements as well as first case studies for some application with our prototype solver. We define a metric representing the quality loss of heuristic methods compared to our approach as: $loss(Method) = 1 - \frac{X_{Method}}{X_{QBF}}$ where X_{Method} is the number of maximized unknowns computed by an approximate method, and X_{QBF} is the value computed by QBF using soft variables.

4.1.1 Comparison to Previous Work

To model unknown values in a circuit, commonly $01X$-encoding [JBM+00] applying SAT-based methods is used. Another popular SAT-based heuristic is *lifting* [RS04] which was introduced for minimal counterexamples (cf. §4.1.6). This method does not need any $01X$-encoding, instead the optimization is done directly on the extracted model of a satisfied SAT instance.

However, $01X$-logic is a pessimistic abstraction. In many applications, more precise solutions which then need a QBF formulation for the unknown values [SB01] are preferred. And although in an optimal case the lifting technique yields solutions as exact as a QBF formulation, it does not guarantee optimal ones due to its heuristic methodology. In QBF each unknown value is associated to a universally quantified variable. In contrast to SAT-based methods, the maximization with classical QBF-based methods is a harder problem, since the semantics of QBF binds each variable to its quantifier, i. e. the quantifier is statically declared and cannot be changed to the purpose of optimizing over unknown lines.

[2] As shown in [Sin05] the encoding for $\mathrm{LT}_{\mathrm{SEQ}}^{n,1}$ is superior to the naïve encoding for $n > 5$.

QBF with soft variables overcomes this issue. For any line which is part of the unknown maximization a soft variable is introduced, and this allows to switch between existential (known) and universal (unknown) quantification (values). Shortly, the QBF with soft variables concept provides a compact representation for modelling unknowns, which is more precise than classical $01X$-based approaches and guarantees optimal solution in general. More details of the encoding are presented in the following subsections.

4.1.2 Uniform Counterexamples

Exploiting partial designs is useful e. g. in the following cases: 1) Abstraction of complex parts of the system, 2) early step of the design process where not all parts are done, and 3) for diagnosis – in case a bug is present in the design and it is still present while parts are removed, there must be errors outside the black boxes. There exist different encoding methods for these hidden parts (usually called *black boxes*) [SB01], and the approach for using them in a BMC context [NSB07] is the so-called *Black Box Bounded Model Checking (BBBMC)*.

One question in BBBMC is whether a property of the system is satisfied for each possible realization of the black boxes. A BBBMC can be written as the following QBF problem: $\psi = \exists X_1 \forall Z_1 \exists X_2 \forall Z_2 \ldots \exists X_k \forall Z_k . I_0 \wedge T_{0,1} \ldots \wedge T_{k-1,k} \wedge \neg P_k$, where X_i are the inputs of the transition relation and Z_i are the outputs of the black box at depth i. If the property cannot be fulfilled, it is interesting to obtain a counterexample for the inputs X_i. Using this encoding, only the assignments for the inputs X_1 are the same for each possible black box implementation – all other inputs may differ depending on the black box implementation. To overcome this issue one can define a uniform QBF prefix as follows: $\psi_{\text{uni}} = \exists X_1 \ldots \exists X_k \forall Z_1 \ldots \forall Z_k . I_0 \wedge T_{0,1} \ldots \wedge T_{k-1,k} \wedge \neg P_k$. Note that these two representations are not equivalent: ψ ensures that the primary inputs of each unrolling depth can "react" to the black box output, whereas in ψ_{uni} the inputs are independent from the black box implementation. Hence, ψ is more accurate than ψ_{uni}, nonetheless ψ_{uni} returns a more general counterexample. In [NSB07] the authors specify partially uniform QBF, i. e., parts of a counterexample which can be generalized without losing accuracy are identified. The proposed method uses symbolic model checking modelling the unknown values with the pessimistic $01X$-logic.

QBF with Soft Variables Formulation The uniform counterexample problem can be solved optimally in the number of inputs which are uniform by applying QBF with soft variables. Let $x^i_j \in X_i$ and $|X_i|$ be the number of variables in X_i. We state the QBF with soft variables problem as follows: $\psi(\mathcal{H}_{\{1,3\}} x^3_1, \ldots, \mathcal{H}_{\{1,3\}} x^3_{|X_3|}, \ldots, \mathcal{H}_{\{1,k\}} x^k_{|X_k|}) = \exists X_1 \forall Z_1 \exists \emptyset \forall Z_1 \ldots \exists \emptyset \forall Z_k . I_0 \wedge T_{0,1} \ldots \wedge T_{k-1,k} \wedge \neg P_k$, i. e. for each input variable at level i we introduce a soft variable trying to shift the quantifier to the first level. Once a variable is quantified at the first level and the QBF is still satisfied, this input is independent from every black box and therefore uniform. We choose the score function σ such that level 1 is more advisable than level i, the solution of the QBF with soft variables problem is the optimal uniform counterexample.

For this and all upcoming applications in §4.1, the score function assigns the same score for each variable, e. g. 1 for the preferred level and 0 for the non-preferred (cf. §3.2.3). In this case, we obtain a globally optimal result. But it would be also possible to prefer particular variables by assigning larger scores.

4.1.3 Diagnosis

As described in §4.1.2, black boxes can be used for diagnosis tasks: If a property of the system is violated regardless of the excluded parts, the failure has to occur in the non-excluded part. A challenge is to identify the black boxed part automatically, such that the remaining sub-circuit is as small as possible. In this case already the non-abstracted circuit is sufficient for the violation of

the property which would simplify the diagnosis task. To the best of our knowledge this problem statement is not tackled so far, even with pessimistic $01X$-encoding.

Using QBF with Soft Variables Using the standard Tseitin-encoding [Tse68] to produce a propositional formula, every line l_i is associated to a Boolean variable. Hence we can define a soft variable for each line[3] obtaining the following QBF with soft variables: $\psi(\mathbb{H}_{\{2,3\}} l_1, \ldots, \mathbb{H}_{\{2,3\}} l_k) = \exists\emptyset\forall\emptyset\exists x_1 \ldots \exists x_n.\varphi \wedge \neg P$, where x_i are inputs of the circuit, φ its encoding, and $\neg P$ the violated property. By preferring universal quantification in the score function, the result of the optimization problem indicates a maximal number of lines, which can be excluded such that the bug is still present. Note, that this result is not accurate anymore in this scenario as discussed in §5. In order to reduce the complexity one may not consider every line but only specific ones, or consider different groups of lines separately and examine them incrementally.

4.1.4 Circuit Initialization

The problem of circuit initialization is a well known problem in the area of testing [PR00] and is closely related to state reachability problems known from BMC [RSSB14]. The problem asks whether a sequential circuit is initializable, i. e. all flip-flops can be set to a known value assuming the initial state is (completely) unknown. If this is not the case (which usually applies), one can seek for two related optimization questions: 1) starting from a unknown state what is the maximum number of initialized flip-flops, or 2) what is the smallest number of initially controlled flip-flops needed to initialize the complete circuit. For both objectives a minimal trace length is preferred.

Classical approaches either solve this problem heuristically (e. g. [PR00]) or are complete, but only with the pessimistic $01X$-logic (cf. [RSSB14]). So far there is no approach that applies a complete method using the accurate QBF modelling for the unknown values.

Using QBF with Soft Variables Based on the method in [RSSB14], we can define a BMC problem based on QBF with soft variables. Let $g^i_j \in G_i$ be a variable representing the value of a flip-flop at depth i and $|G_i|$ the number of variables in G_i. For the objective 1) we obtain: $\psi(\mathbb{H}_{\{1,2\}} g^k_1, \ldots, \mathbb{H}_{\{1,2\}} g^k_{|G_k|}) = \exists\emptyset\forall G_0 \exists G_1 \ldots \exists G_{k-1}.I_0 \wedge T_{0,1} \ldots \wedge T_{k-1,k} \wedge P_k$. If the score functions recommend to quantify a soft variable existentially on the first level, the maximum number of initialized flip-flops in time step k is given by the number of these existentially quantified variables. The second objective 2) can be expressed as: $\psi(\mathbb{H}_{\{2,3\}} g^0_1, \ldots, \mathbb{H}_{\{2,3\}} g^0_{|G_0|}) = \exists G_k \forall\emptyset\exists G_1 \ldots \exists G_{k-1}.I_0 \wedge T_{0,1} \ldots \wedge T_{k-1,k} \wedge P_k$. Here, G_k is quantified on the first level since we have to ensure that the value for the flip-flops in the k'th time step are fixed to a specific known value independently from the potentially unknown flip-flop values of the initial time step 0. By applying a score function favoring the quantification of soft variables at the universal level 2, the result for this optimization problem is equivalent to the minimum number of controlled flip-flops in the first time step.

Note, in contrast to the method in [RSSB14], this approach does not contain a proof whether more flip-flops can be initialized in further time steps, but it provides more accurate results as shown in the following case study.

Case Study We applied the encoding to the benchmark `b06` from the ITC 99 benchmark series which is commonly used in the EDA community and previously proven to be not completely initializable using $01X$-encoding [RSSB14]. However, using QBF with soft variables, we have identified an initialization sequence of 5 time steps driving each flip-flop to a specified value.

[3] It may be meaningful to define groups of lines (e. g. buses or outputs of a specific module such as multiplier), which can be defined analogously.

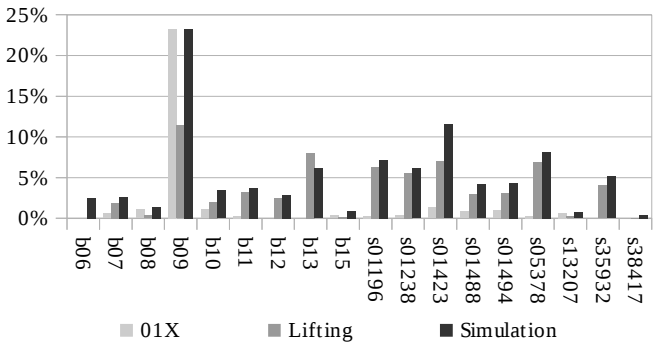

Figure 1: Loss for test pattern relaxation using approximated methods

4.1.5 Test Pattern Relaxation

In the area of testing [JG03] and in particular Automatic Test Pattern Generation (ATPG) the number of specified input bits in a test pattern is a major quality metric. Partially specified test patterns, so-called test cubes, serve as a foundation for further post-processing steps controlling secondary objectives, e. g. compaction, or test power reduction. As the quality of these methods depends on the number of unspecified values, test patterns with a greater amount of unspecified inputs are preferred.

Using QBF with Soft Variables Once a test pattern is found, we introduce soft variables for all inputs $x_1, \ldots, x_n$ in order to preferably treat them as unknowns. A resulting QBF may look like: $\psi(\mathbb{X}_{\{2,3\}}x_1, \ldots, \mathbb{X}_{\{2,3\}}x_n) = \exists\emptyset\forall\emptyset\exists Y.\varphi$, where φ is the encoding of the circuit using variables V and the property justifying the test pattern. If the score function is built favoring the universal quantification level, one obtains the maximum number of lines that can be excluded from the test pattern such that the fault is still visible.

Case Study In [SRP+13] an approach using an accurate QBF modeling was already presented, however without using the soft variable concept. For the common ISCAS 89 and ITC 99 benchmark circuits we generated minimal test cubes for 100 test pattern detecting small delay faults [JG03]. We used QBF with soft variables to document the quality loss of common heuristics ($01X$, lifting, and simulation) in the number of unspecified inputs compared to the optimal solution. Fig. 1 shows the quality loss for different benchmarks. As it can be seen, depending on the internal structure of the individual benchmark the heuristics can identify up to 23% less unspecified inputs for benchmark circuit `b09`.

4.1.6 Minimal Counterexamples

SAT-based Bounded Model Checking (BMC) [BCC+03] is a formal verification technique for designs modeled as finite state machines (FSM). The transition relation of the FSM is unrolled step by step and a property is verified at the current depth. The depth k is incremented until either the property is violated for the given depth, or a user-defined bound for the depth is reached. A SAT formula for the BMC problem can be written as: $\varphi = I_0 \wedge T_{0,1} \wedge \ldots \wedge T_{k-1,k} \wedge \neg P_k$, where I_0 is the initial state of the system, $T_{i,i+1}$ is the transition relation for step i and P_k is the property at step k. Note that φ is true iff the property is violated.

In case of a violated property the model of φ represents a full trace of assignments to the primary inputs of the transition relation (i.e. a counterexample) of the current unrolling depth. This trace can be used for diagnosis or abstraction refinements. In order to improve the diagnosis routines

Table 1: Results for HWMCC benchmarks

Family	Inst.	Inputs	QBF Time	QBF X	Lifting Time	Lifting X	Loss
beemi*	14 / 0	489.71	0.60	17.50	4.01	17.50	0.00
beemsch*	16 / 0	320.38	1.03	13	0.68	12	0.31
bj*	12 / 1	16.27	0.07	14.36	0.02	14.36	0.00
brpp*	5 / 0	177.40	287.46	58.40	0.14	46.20	20.76
counter*	2 / 0	82.00	0.01	21.00	0.02	20.50	2.38
dme*	20 / 0	455.35	6.95	301.20	0.37	262.45	12.12
ken*	2 / 0	268.68	3.48	161.10	0.19	141.48	7.25
pci*	5 / 0	477.25	91.64	282.50	0.29	282.50	0.00
srg*	3 / 0	81.00	0.29	25.67	0.02	11.33	52.12
texas*	9 / 1	117.63	6.29	102.88	0.49	101.63	0.80
vis*	6 / 2	78.00	49.52	45.00	0.18	44.00	2.30
total	94 / 4	288.32	25.74	102.88	0.91	92.87	5.98

the number of primary inputs, which are needed to justify the violation of the property, can be delimited.

Note that the test pattern relaxation problem as described in §4.1.5 can be seen as a special case of the minimal counterexample problem. Instead of asking whether a property is violated, we ask whether a property (detection of a fault) holds minimizing the number of specified inputs (test cube).

Using QBF with Soft Variables Consider a BMC problem φ as stated above with a counterexample at depth k. We denote $X = x_1, \ldots, x_n$ as the variables representing all inputs of the transition relations and $Y = y_1, \ldots, y_m$ all other variables in φ. In order to minimize the number of specified inputs (or maximize the unknown inputs), we state the following QBF with soft variables instance: $\psi(\mathbb{H}_{\{2,3\}} x_1, \ldots, \mathbb{H}_{\{2,3\}} x_n) = \exists\emptyset\forall\emptyset\exists Y.\varphi$, i. e. a QBF containing only existentially quantified variables Y on level 3 and the soft variables X. If we define the score function σ such that the universally quantified position gets a larger score than the existential position, the result of $\Omega(\psi)$ returns the maximum number of inputs which can be excluded from the counterexample such the property is still unsatisfied. Hence, we obtain a counterexample with a minimal number of specified inputs.

Case Study We considered 94 unsatisfiable BMC benchmarks from 11 classes of the Hardware Model Checking Competitions (2010, 2011 and 2012). We selected the Bounded Model Checker `cip` [KLSB11] which is able to extract counterexamples from buggy benchmark designs. For some groups with smaller number of inputs we computed the minimal counterexamples for these benchmarks, and compared them with the results obtained by applying a lifting algorithm.

The results are given in Table 1. In the first three columns the family name, the number of overall / non-solved QBF instances within a timeout of 30 CPU minutes, and the average number of inputs of the original counterexample are given. In order to compare results, the following columns contains only results for the instances where the QBF approach was able to provide a solution. The following two columns show the average run time in seconds and the average number of inputs we can exclude from the counterexample using QBF with soft variables. The next two columns show the same for the lifting approach, and the last column indicates the average quality loss of the lifting approach.

On average the loss of accuracy with lifting is about 6% compared to the exact QBF with soft variables. However, for some benchmarks the loss is over 70% (instance `brpptimeonenegnv`). The run times for these smaller examples are very reasonable with just 4 non-solved benchmarks, but our approach still has scaling issues, especially if the number of lines to maximize is large.

4.2 Further applications

For the task of finding optimal dependency schemes [Sam08] one can use the optimization problem for QBF with soft variables. Therefore, we declare a single existential variable of the original problem as soft variable allowing to be quantified on all existential prefix position. In the score function we prefer the position with the least dependencies on universally quantified variables as possible.

Another application area where QBF with soft variables may be a useful mechanism is planning and decision making, which are common problems evolving from the game theory domain. In a multi-agent environment, where more than one player is present, such problems ask to determine the best action. Hence, the goal is to find the action model with the highest score. In [YWJ07] a weighted MaxSAT solver is used to determine these models. But this approach is limited if we consider uncertainness of other players' action, the environment or possibly unknown actions performed earlier. Using MaxQBF one's own action score could be maximized with accuratly modeled behaviour of the enviroment. Moreover, by using QBF with soft variables it can be determined whether the order of the actions can be altered or even if an action can be left out in order to decrease the overall costs.

A related problem are two-players games [AMN05], which can be naturally formulated with QBF logic by giving the existential quantifier (resp. the universal quantifier) the role of the system player (resp. the environment player). As in the planning problem one could try to make own moves independent from the opponent ones using QBF with soft variables for the existential variables. This would lead to more generalized winning strategies. This problem is related to the uniform counterexamples (cf. §4.1.2) and can be tackled in a similar manner.

5 Conclusions

We presented first results on the novel concept of soft variables for quantified Boolean formulae. The related optimization problem allows to optimize the prefix according to a user-given preference. Furthermore, we introduced a MaxQBF solver and a sound and complete algorithm to solve QBF with soft variables using MaxQBF.

Recent improvements in classical algorithms for SAT and QBF lead to new research interests in answering beyond yes/no questions. Such optimization problems using quantified formulas are often hard to encode due to the static prefix or rather predefined dependencies of the variables. The concept of soft variables overcomes this issues and opens a wide field of such applications requiring an accurate model. We demonstrated the applicability of our formalism by introducing several applications in the area of formal verification, debugging, testing and artificial intelligence.

However, also QBF is limited if the abstracted part modeled by universally quantified variables either show sequential behavior or there are multiple black box parts with overlapping cones of the black box inputs and outputs. In these cases QBF (hence also QBF with soft variables) is not accurate anymore [GRS+13]. To overcome the latter issue one can define explicit dependencies of the variables rather than defining a linear prefix. This extension is known as *Dependency QBF* (DQBF) using so-called Henkin quantifiers, where variables are quantified with a explicitly given variable set of dependencies [Hen61]. The definition of a DQBF with soft variables is straightforward: instead of possible prefix positions a soft variable is allowed to be quantified with different dependency sets[4]. Research interest starts on focusing DQBF algorithms and applications [GRS+13], but the scalability of these algorithms is still not reasonable for tackling even more complex optimization problems (deciding pure DQBF is already NEXPTIME-complete [PRA01]). Therefore we do not consider soft variables for DQBF in this paper, but it may become relevant if appropriate algorithms for solving DQBF are present.

As future work we want to investigate the new application areas covered by our soft variable

[4] Actually, the prefix of a QBF is just a convenient notation for a linear dependency relation between the variables.

mechanism as well as dedicated solving mechanisms beyond the techniques used for MaxSAT/QBF-based algorithms to increase the scalability.

[AMN05] R. Alur, P. Madhusudan, W. Nam. Symbolic computational techniques for solving games. *STTT* 7(2):118–128, 2005.

[BCC+03] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, Y. Zhu. Bounded model checking. *Advances in Computers* 58:117–148, 2003.

[BELM12] B. Becker, R. Ehlers, M. Lewis, P. Marin. ALLQBF Solving by Computational Learning. In *ATVA*. LNCS 7561, pp. 370–384. Springer, 2012.

[BHMW09] A. Biere, M. Heule, H. van Maaren, T. Walsh (eds.). *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications 185. IOS Press, 2009.

[BLV08] M. Benedetti, A. Lallouet, J. Vautard. Quantified constraint optimization. In *Principles and Practice of Constraint Programming*. Pp. 463–477. 2008.

[CFLS93] A. Condon, J. Feigenbaum, C. Lund, P. Shor. Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions. In *Proceedings of the 25th annual ACM symposium on Theory of computing*. Pp. 305–314. 1993.

[CP04] H. Chen, M. Pál. Optimization, games, and quantified constraint satisfaction. In *Mathematical Foundations of Computer Science 2004*. Pp. 239–250. Springer, 2004.

[EB05] N. Eén, A. Biere. Effective Preprocessing in SAT through Variable and Clause Elimination. In *SAT*. 2005.

[GNT07] E. Giunchiglia, M. Narizzano, A. Tacchella. Quantifier Structure in Search-Based Procedures for QBFs. *IEEE Trans. on CAD of Integrated Circuits and Systems* 26(3):497–507, 2007.

[GRS+13] K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, B. Becker. Equivalence Checking of Partial Designs Using Dependency Quantified Boolean Formulae. In *Proc. of the 31st IEEE Int'l Conf. on Computer Design*. Pp. 396–403. 2013.

[HB07] M. Herbstritt, B. Becker. On Combining 01X-Logic and QBF. In *EUROCAST*. Lecture Notes in Computer Science 4739, pp. 531–538. Springer, 2007.

[Hen61] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods: Proceedings of the 1959 Symposium on Foundations of Mathematics*. Pp. 167–183. Pergamon Press, Sept. 1961.

[IJM13] A. Ignatiev, M. Janota, J. Marques-Silva. Quantified Maximum Satisfiability. In *SAT*. Pp. 250–266. Springer, 2013.

[JBM+00] A. Jain, V. Boppana, R. Mukherjee, J. Jain, M. Fujita, M. S. Hsiao. Testing, Verification, and Diagnosis in the Presence of Unknowns. In *VLSI Test Symp.* Pp. 263–269. 2000.

[JG03] N. K. Jha, S. K. Gupta. *Testing of Digital Systems*. Cambridge U. Press, 2003.

[KLSB11] S. Kupferschmid, M. Lewis, T. Schubert, B. Becker. Incremental Preprocessing Methods for Use in BMC. *Formal Methods in System Design*, pp. 1–20, 2011.

[MMB12] P. Marin, C. Miller, B. Becker. Incremental QBF Preprocessing for Partial Design Verification - (Poster Presentation). In *SAT*. LNCS 7317. Springer, 2012.

[MMLB12] P. Marin, C. Miller, M. Lewis, B. Becker. Verification of partial designs using incremental QBF solving. In *DATE*. Pp. 623–628. 2012.

[NSB07] T. Nopper, C. Scholl, B. Becker. Computation of minimal counterexamples by using black box techniques and symbolic methods. In *ICCAD*. Pp. 273–280. 2007.

[PR00] I. Pomeranz, S. M. Reddy. On synchronizable circuits and their synchronizing sequences. *IEEE Trans. on CAD of Integrated Circuits and Systems* 19(9):1086–1092, 2000.

[PRA01] G. Peterson, J. Reif, S. Azhar. Lower Bounds for Multiplayer Non-Cooperative Games of Incomplete Information. *Computers & Mathematics with Applications* 41(7–8):957–992, 2001.

[RPSB12] S. Reimer, F. Pigorsch, C. Scholl, B. Becker. Enhanced Integration of QBF Solving Techniques. In *MBMV*. Pp. 133–143. 2012.

[RS04] K. Ravi, F. Somenzi. Minimal Assignments for Bounded Model Checking. In *TACAS*. Lecture Notes in Computer Science 2988, pp. 31–45. Springer, 2004.

[RSSB14] S. Reimer, M. Sauer, T. Schubert, B. Becker. Using MaxBMC for Pareto-Optimal Circuit Initialization. In *DATE*. 2014.

[Sam08] M. Samer. Variable dependencies of quantified CSPs. In *Logic for Programming, Artificial Intelligence, and Reasoning*. Pp. 512–527. 2008.

[SB01] C. Scholl, B. Becker. Checking equivalence for partial implementations. In *Design Automation Conference, 2001. Proceedings*. Pp. 238 – 243. 2001.

[Sin05] C. Sinz. Towards an optimal CNF encoding of Boolean cardinality constraints. In *Principles and Practice of Constraint Programming-CP 2005*. Pp. 827–831. Springer, 2005.

[SM73] L. J. Stockmeyer, A. R. Meyer. Word problems requiring exponential time (Preliminary Report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*. STOC '73, pp. 1–9. ACM, New York, NY, USA, 1973.

[SRP+13] M. Sauer, S. Reimer, I. Polian, T. Schubert, B. Becker. Provably optimal test cube generation using quantified Boolean formula solving. In *ASP-DAC*. Pp. 533–539. IEEE, 2013.

[Tse68] G. Tseitin. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic*, 1968.

[YWJ07] Q. Yang, K. Wu, Y. Jiang. Learning action models from plan examples using weighted MAX-SAT. *Artif. Intell.* 171(2-3):107–143, 2007.

[ZSM03] H. Zhang, H. Shen, F. Manya. Exact algorithms for MAX-SAT. *Electronic Notes in Theoretical Computer Science* 86(1):190–203, 2003.