Equivalence Checking for Partial Implementations Revisited

Karina Gitina Sven Reimer Matthias Sauer Ralf Wimmer
Christoph Scholl Bernd Becker
Albert-Ludwigs-University Freiburg, Germany
{gitina,reimer,sauerm,wimmer,scholl,becker}@tf.uni-freiburg.de

Abstract

In this paper we consider the problem of checking whether a partial implementation can (still) be extended to a complete design which is equivalent to a given full specification. In particular, we investigate the relationship between the equivalence checking problem for partial implementations (PEC) and the validity problem for quantified Boolean formulae (QBF) with so-called Henkin quantifiers. Our analysis leads us to a sound and complete algorithmic solution to the PEC problem as well as to an exact complexity theoretical classification of the problem.

1. Introduction

Finding errors in system designs as early as possible is of utmost importance to reduce development costs and time-to-market. Therefore the verification of incomplete or partial system designs has received a lot of research efforts during the last decade [11, 12, 9, 6, 10, 7, 8, 15].
Assume we are given a system design such that some parts are so-called "*black boxes*", i. e., modules the internal structure of which is not known. The reasons for having such black boxes are 1) parts of the system are still to be implemented, 2) parts which are supposed not to influence the validity of some properties have been removed to simplify the verification task (e. g., multiplier or memory modules), and 3) parts are removed to enable the localization of errors: If parts of the design have been removed and for all possible implementations of the removed parts the error does not disappear, the remaining parts must be erroneous.
Regarding black boxes, research focuses on checking whether a partial design can be extended such that it is equivalent to a given specification, i. e., whether the specification can be *realized*. We call this the partial equivalence checking problem (PEC). If it turns out that there is no feasible extension, the available parts are erroneous and must be fixed. This helps to detect errors in an early stage of a design.
As in [11], we consider here the case that the partial design is a combinational circuit containing black boxes and the specification is a combinational circuit as well. Generalizations to sequential circuits (based on bounded model checking) may be performed as in [6].

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) and the Research Training Group "Embedded Microsystems" (GRK 1103).

A series of approximative and exact methods to solve PEC is presented in [11]. If approximative algorithms report that there is no black box implementation for the partial design, such that the specification can be realized, the desired functionality is indeed not realizable. However, if the algorithms reports realizability, this can be due to the approximative character of the method, and the desired functionality may nevertheless be unrealizable. The algorithms in [11] are based on solving SAT or QBF formulations of PEC. The SAT formulations in general can be solved efficiently, but they provide rather coarse approximations. Their accuracy is improved in several steps, leading to QBF formulations that can solve PEC for one black box exactly. The authors of [11] additionally give a characterization of PEC realizability, which is exact also for multiple black boxes. However, this characterization does not imply a feasible algorithmic method for solving the problem.
We show that for solving PEC with multiple black boxes exactly, an extension of QBF called dependency quantified Boolean formulae (DQBF) can be used. A DQBF is a propositional formula with Henkin-quantifiers [5]. With these quantifiers one can explicitly specify the universally quantified variables on which the existential ones depend—in contrast to QBF without Henkin quantifiers, where an existentially quantified variable depends on all other quantifiers appearing on the left of this variable in the prefix. The first algorithmic approach that considers DQBF is stated in [3]. The algorithm is based on the QBF-extension QDLL for the search-based DLL [2] algorithm for SAT.
We additionally answer the question about the complexity of PEC. This has been an open problem so far. We do this by showing that PEC is equivalent to DQBF. A direct consequence of this is that PEC lies in the same complexity class as DQBF, namely both are NEXPTIME-complete.
The remainder of the paper is structured as follows. In Section 2 we give the foundations of DQBF and equivalence checking of partial designs. In Section 3 we prove the equivalence of both and show how to translate a partial design into a DQBF. We give some pointers for current and future applications in Section 4. Section 5 concludes the paper and provides hints on future work.

2. Foundations

We introduce dependency quantified Boolean formulae and partial equivalence checking.

2.1. Dependency Quantified Boolean Formulas

A propositional formula φ over a set of variables V in *conjunctive normal form (CNF)* is a conjunction of clauses. A *clause* c is a disjunction of literals, usually written as a set of literals $c = \{l_1, \ldots, l_n\}$. A *literal* l is either a variable $X \in V$ or its negation $\neg X$.
For a vector $A = (A_1, \ldots, A_k)$ of Boolean variables we also write $\forall A$ as an abbreviation for $\forall A_1 \ldots \forall A_k$ and accordingly for $\exists A$.

Definition 1 *A* dependency quantified Boolean formula (DQBF) *has the following structure:*

$$\forall X_1 \forall X_2 \ldots \forall X_n \exists Y_1(S_1) \exists Y_2(S_2) \ldots \exists Y_m(S_m).\varphi(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$$

with $S_1,\ldots,S_m \subseteq \{X_1,\ldots,X_n\}$ and $\varphi(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$ a quantifier-free propositional formula in CNF (also called matrix*), containing only the variables $X_1,\ldots,X_n,Y_1,\ldots,Y_m$.*

A quantified Boolean formula (QBF) in the traditional sense is a DQBF such that $S_1 \subseteq S_2 \cdots \subseteq S_m$. Usually such a QBF is written with a quantifier prefix where $\exists Y_1$ follows directly after $\forall S_1$, $\exists Y_2$ directly after $\forall(S_2 \setminus S_1)$, etc.
The truth value of a DQBF is typically defined in terms of Skolem functions for the existential variables.

Definition 2 *Let $\mathscr{F}_k = \{f : \mathbb{B}^k \to \mathbb{B}\}$ denote the set of all Boolean functions with k arguments. A DQBF*

$$\forall X_1 \forall X_2 \ldots \forall X_n \exists Y_1(S_1) \exists Y_2(S_2) \ldots \exists Y_m(S_m).\varphi(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$$

with dependency sets $S_i \subseteq \{X_1,\ldots,X_n\}$ for $i = 1,\ldots,m$ is satisfiable *iff there are functions $s_i \in \mathscr{F}_{|S_i|}$ for $i = 1,\ldots,m$ such that*

$$\varphi(X_1,\ldots,X_n,s_1(S_1),s_2(S_2),\ldots,s_m(S_m))$$

is a tautology (i. e., satisfied for all assignments of $X_1,\ldots,X_n$).

Deciding whether a DQBF is satisfied is known to be NEXPTIME-complete [1], while deciding QBF is "only" PSPACE-complete [13].

Example 1 *Consider the following DQBF:*

$$\forall X_1\ \forall X_2\ \forall I\ \exists Y_1(X_1,I)\ \exists Y_2(X_2,I).\ \Big(\big(I \oplus \overline{X_1 \wedge X_2}\big) \vee \big(\overline{Y_1 \wedge Y_2} \equiv (X_1 \oplus X_2)\big)\Big)$$

It is satisfied for Skolem functions $s_1(X_1,I) = \overline{X_1 \wedge I}$ for Y_1 and $s_2(X_2,I) = \overline{X_2 \wedge I}$ for Y_2. Replacing Y_1 and Y_2 with their Skolem functions yields:

$$\forall X_1 \forall X_2 \forall I.\ \Big(\big(I \oplus \overline{X_1 \wedge X_2}\big) \vee \big(\overline{\overline{X_1 \wedge I} \wedge \overline{X_2 \wedge I}} \equiv (X_1 \oplus X_2)\big)\Big)$$

We have to show that the matrix is satisfied for all assignments of X_1, X_2, and I. Simple transformations yield:

$$\begin{aligned}
& \Big(\big(I \oplus \overline{X_1 \wedge X_2}\big) \vee \big(\overline{\overline{X_1 \wedge I} \wedge \overline{X_2 \wedge I}} \equiv (X_1 \oplus X_2)\big)\Big) \textit{ is a tautology} \\
\textit{iff}\quad & \Big(\big(I \equiv \overline{X_1 \wedge X_2}\big) \Rightarrow \big((X_1 \wedge I \vee X_2 \wedge I) \equiv (X_1 \oplus X_2)\big)\Big) \textit{ is a tautology} \\
\textit{iff}\quad & \big((X_1 \wedge \overline{X_1 \wedge X_2}\ \vee\ X_2 \wedge \overline{X_1 \wedge X_2}) \equiv (X_1 \oplus X_2)\big) \textit{ is a tautology} \\
\textit{iff}\quad & \big((X_1 \wedge (\overline{X_1} \vee \overline{X_2})\ \vee\ X_2 \wedge (\overline{X_1} \vee \overline{X_2})) \equiv (X_1 \oplus X_2)\big) \textit{ is a tautology} \\
\textit{iff}\quad & \big((X_1 \wedge \overline{X_2}\ \vee\ X_2 \wedge \overline{X_1}) \equiv (X_1 \oplus X_2)\big) \textit{ is a tautology} \\
\textit{iff}\quad & \big((X_1 \oplus X_2) \equiv (X_1 \oplus X_2)\big) \textit{ is a tautology.}
\end{aligned}$$

The last formula is obviously a tautology. Thus, the original DQBF is satisfied with the given Skolem function.

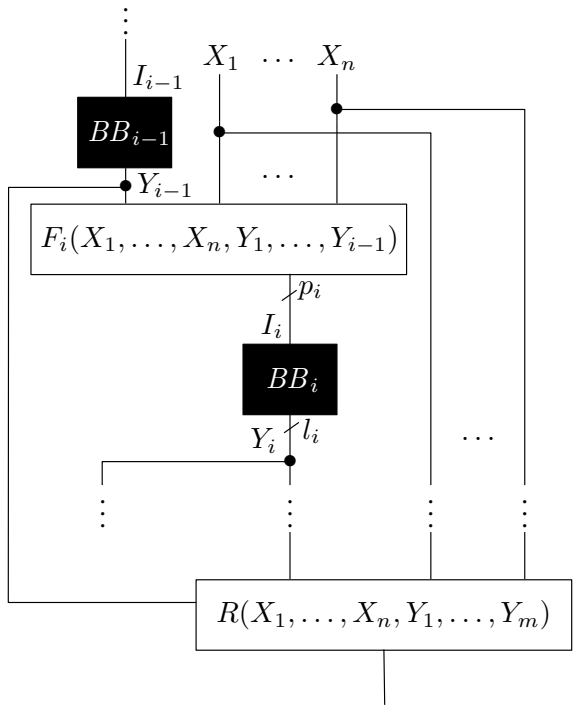

Figure 1: The notations for PEC

2.2. Partial Equivalence Checking

First, we introduce notations for partial combinational circuits P:

- $X_1,\ldots,X_n$ are the primary inputs of the circuit.
- $BB_1,\ldots,BB_m$ are the black boxes of the circuit in topological order[1].
- $I_i = (I_{i,1},\ldots,I_{i,p_i})$ are the inputs of BB_i ($i = 1,\ldots,m$).
- $Y_i = (Y_{i,1},\ldots,Y_{i,l_i})$ are the outputs of BB_i ($i = 1,\ldots,m$).
- $I_{i,j} = F_{i,j}(X_1,\ldots,X_n,Y_1,\ldots,Y_{i-1})$ is the Boolean function defining the input j of BB_i. We write $I_i = F_i(X_1,\ldots,X_n,Y_1,\ldots,Y_{i-1})$ for the vector of functions defining $I_{i,1},\ldots,I_{i,p_i}$.
- $R(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$ is the output function of the circuit.

These notations are illustrated in Figure 1.

[1] To guarantee that the circuit is combinational, we assume that $BB_1,\ldots,BB_m$ are in topological order, i. e., BB_i does not depend on BB_j for $i < j$.

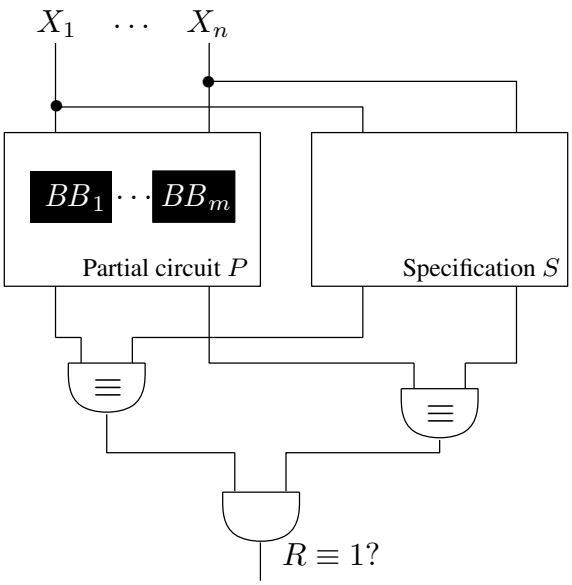

Figure 2: Combination of the partial design with the specification

Definition 3 Partial Equivalence Checking (PEC) *considers the following decision problem: Given a partial combinational circuit P and a specification S, which is a combinational circuit (without black boxes). Are there implementations of the black boxes such that the partial circuit becomes equivalent to the specification? If these implementations exist, we say the PEC is satisfied, otherwise unsatisfied.*

We combine P and S into a single partial circuit C such that there is an implementation of P which is equivalent to S iff there is an implementation of the combined circuit such that the single output of C is 1 for all assignments of its primary inputs. To do so we require the output function R for C to be 1. This is illustrated in Figure 2.

Example 2 *Consider the partial circuit in Figure 3. The dashed part on the left-hand side is the partial design to be checked, the XOR-gate on the right-hand side is the specification. They have been combined into a single circuit by connecting their outputs with an equivalence gate. The property which is to be checked is: Can the two black boxes be implemented such that the partial circuit becomes equivalent to the XOR-gate?*

3. PEC and DQBF

In this section we show that deciding PEC is equivalent to deciding DQBF. To do so, we specify a linear transformation from DQBF to PEC and vice versa, and show for both cases that the DQBF is satisfied if and only if the corresponding PEC is satisfied.

Lemma 1 *Any DQBF can be translated into an equivalent PEC with linear effort.*

Proof: Consider a DQBF

$$\psi = \forall X_1 \forall X_2 \ldots \forall X_n \exists Y_1(S_1) \exists Y_2(S_2) \ldots \exists Y_m(S_m).\varphi(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$$

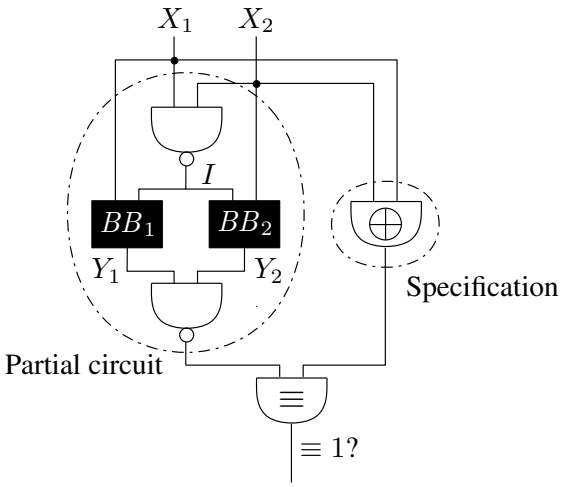

Figure 3: Example for PEC

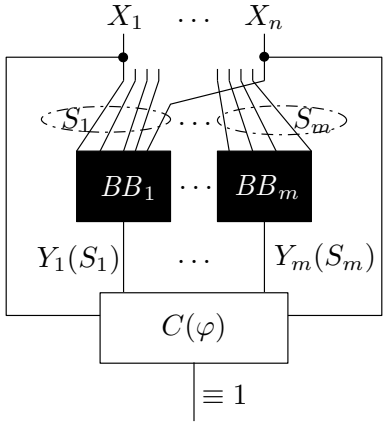

Figure 4: Translation from DQBF to PEC

with $S_1, \ldots, S_m \subseteq \{X_1, \ldots, X_n\}$ and its matrix $\varphi(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ in CNF.

The matrix φ can be easily transformed into a circuit $C(\varphi)$ with inputs $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ with at most a linear increase in size. The input Y_i of $C(\varphi)$ is the output of a new black box BB_i in the PEC. The inputs of BB_i are exactly the signals in S_i. Requiring the output of $C(\varphi)$ to be constantly 1 is equivalent to comparing the incomplete circuit P with the 1-function as the specification S. The translation is illustrated in Figure 4.
If the DQBF ψ is satisfied, then there exist Skolem functions s_i, depending on the variables in S_i, for all Y_i such that

$$\forall X_1 \forall X_2 \ldots \forall X_n : \varphi(X_1, \ldots, X_n, s_1(S_1), s_2(S_2), \ldots, s_m(S_m))$$

is satisfied, i. e. φ is a tautology. These Skolem functions can be used as implementations of the black boxes.

On the other hand the PEC is satisfied, if there exist implementations for all black boxes BB_i such that the requirements hold. The Boolean functions corresponding to these implementations can be used as Skolem functions for the DQBF.
Following directly from the construction of the PEC, Skolem functions s_i for all variables Y_i exist, such that ϕ is satisfied, if and only if implementations for all BB_i exist, such that the PEC is satisfied. Therefore any DQBF can be translated with linear effort into a PEC and the DQBF is satisfied iff the PEC is satisfied. □

Lemma 2 *Any PEC can be translated into an equivalent DQBF with linear effort.*

Proof: Consider a PEC with black boxes $BB_1, \ldots, BB_m$. We assume that the partial circuit and the specification have already been combined into a single circuit with the requirement that the output has be to constantly 1. We follow the notations introduced in Section 2.2, i. e., a black box BB_i has outputs $Y_{i,1}, \ldots, Y_{i,l_i}$ and inputs $I_{i,1}, \ldots, I_{i,p_i}$ etc.
We assume w. l. o. g. that $Y_i \cap I_j = \emptyset$ for all i, j. That means no output of a black box is directly connected to an input of another black box. Since we will need to use universal quantification for black box inputs, but existential quantification for black box outputs, having $Y_i \cap I_j \neq \emptyset$ would lead to a contradiction. If our assumption is violated, i. e., $Y_i \cap I_j \neq \emptyset$, we can insert a buffer between BB_i and BB_j to separate the outputs of BB_i and the inputs of BB_j. Since a buffer "computes" the identity function $\text{buffer}(x) = x$, this does not change the functionality of the circuit and causes at most a linear blow-up of the circuit.
We first construct the quantifier prefix of the DQBF. The primary inputs $X_1, \ldots, X_n$ and the black box inputs $I_1, \ldots, I_m$ are universally quantified, all other variables are existentially quantified. The dependency set of black box outputs Y_i contains exactly the inputs I_i of BB_i. Hence the quantifier prefix is

$$\forall X_1 \ldots \forall X_n \forall I_1 \ldots \forall I_m \exists Y_1(I_1) \ldots \exists Y_m(I_m) \ .$$

If the black boxes are not directly connected to the primary inputs but to internal signals we have to take into account that not all possible combinations of values may arrive at the inputs of the black boxes. Since we use a universal quantification for the black box inputs we have to ensure that our formula is satisfied if the value of the black box inputs I_i deviates from the values obtained as a function $F_i(X_1, \ldots, X_n, Y_1, \ldots, Y_{i-1})$.

$$\begin{aligned} \varphi(X_1, \ldots, X_n, I_1, \ldots, I_m, Y_1, \ldots, Y_m) = {} & \big(I_1 \not\equiv F_1(X_1, \ldots, X_n)\big) \vee \cdots \\ & \vee \big(I_m \not\equiv F_m(X_1, \ldots, X_m, Y_1, \ldots, Y_{m-1})\big) \\ & \vee R(X_1, \ldots X_n, Y_1, \ldots, Y_m) \ . \end{aligned}$$

This formula is not necessarily given in CNF. By applying Tseitin transformation [14], which is essentially introducing auxiliary variables for the internal signals of the circuit, one can obtain a CNF φ' that is satisfiability equivalent to φ and whose size is linear in the size of φ. Let A be the vector of these auxiliary variables, which are existentially quantified in the quantifier prefix. Their dependency set encompasses all universally quantified variables.

The resulting DQBF is:

$$\psi = \forall X_1 \ldots \forall X_n \forall I_1 \ldots \forall I_m \exists Y_1(I_1) \ldots \exists Y_m(I_m) \exists A(X_1, \ldots, X_n, I_1, \ldots, I_m). \\ \varphi'(X_1, \ldots, X_n, I_1, \ldots, I_m, Y_1, \ldots, Y_m, A) \ .$$

As in Lemma 1 ψ is satisfied if we can replace all $Y_i(I_i)$ with Skolem functions $s_i(I_i)$ such that φ' is a tautology. The Skolem functions s_i exists if and only if there are implementations for the black boxes BB_i of the PEC, such that the PEC is satisfied. Therefore any PEC can be translated with linear effort into a DQBF and the PEC is satisfied iff the DQBF is satisfied. □

We illustrate this transformation with an example:

Example 3 *Consider again the PEC shown in Figure 3. The corresponding DQBF has already been given in Example 1:*

$$\forall X_1 \ \forall X_2 \ \forall I \ \exists Y_1(X_1, I) \ \exists Y_2(X_2, I). \ \Big(\big(I \oplus \overline{X_1 \wedge X_2} \big) \vee \big(\overline{Y_1 \wedge Y_2} \equiv (X_1 \oplus X_2) \big) \Big)$$

The primary inputs X_1 and X_2 as well as the input I of the black boxes have to be universally quantified. The outputs Y_1 and Y_2 are existentially quantified. Signal Y_1 depends on X_1 and I, since BB_1 has exactly these signals as inputs. For the matrix we require that either the input I is inconsistently assigned ($I \oplus \overline{X_1 \wedge X_2}$) or the requirement has to be satisfied, i. e., $\overline{Y_1 \wedge Y_2} \equiv (X_1 \oplus X_2)$. A solution of the PEC is obtained by replacing both BB_1 and BB_2 by a NAND-gate.

Following Lemma 2, the formulation of a PEC as DQBF leads to a new approach for solving this problem. A first algorithmic approach to solve DQBF is stated in [3].
We have shown that for each PEC P there is a DQBF Q with size linear in the size of P such that P is satisfiable iff Q is true. Conversely, for each DQBF Q there is a PEC P such that the size of P is linear in the size of Q and Q is true iff P is satisfiable. This is captured in the following theorem:

Theorem 1 *PEC and DQBF are equivalent.*

Proof: Lemma 1 gives a polynomial-time transformation from DQBF to PEC, Lemma 2 a polynomial-time transformation from PEC to DQBF. Therefore both problems are equivalent.□

Corollary 1 *PEC is NEXPTIME-complete.*

Proof: PEC is NEXPTIME-hard, since DQBF is NEXPTIME-hard [1] and Lemma 1 holds. PEC is in NEXPTIME, since DQBF is in NEXPTIME [1] and Lemma 2 holds. Therefore PEC is NEXTIME-complete. □

The transformation of PEC to DQBF enables us to benefit from recent advances in solver technology to solve PEC. At the same time, a growing number of relevant applications for DQBF will push the development of efficient solvers, making it possible to verify increasingly complex designs.

4. Applications

We have shown the equivalence of DQBF and PEC for circuits with multiple black boxes. Such circuit designs get more and more important as today's nanoscale design processes incorporate pre-designed intellectual property(IP)-cores from different vendors. Such a design flow gets increasingly common especially when designing complex systems-on-a-chip (SoC) [4]. In such cases, only the expected functionality of the externally designed IP-core is given based on the specifications. However, due to design bugs or a malicious modification, the real circuitry may differ from the expected behavior. In addition, the detailed netlist of the IP-core may not be given at all.
Proving that there exist no implementations of the external IP-cores that lead to a violation of security or safety properties is an important question in such designs. Based on the PEC formulation shown in Section 2.2 such properties can be expressed as follows. The SoC is modelled with the external IP-cores as black boxes and checked for equivalence against a representation of the circuit with the inverted property. The system is safe, if the resulting PEC-instance is proven to be unsatisfiable. Otherwise an implementation of the IP-cores exists that leads to a violation of the property.

5. Conclusion and future work

In this paper we have shown that combinational equivalence checking for partial circuits (PEC) and checking the satisfiability of dependency quantified Boolean formulae (DQBF) are equivalent. We have given both a linear transformation from PEC to DQBF and vice versa. This also answers the open question about the complexity of PEC, namely we have shown that PEC is NEXPTIME-complete.
As future work we plan to develop an efficient DQBF-solver that can be used to solve the equivalence checking problem in those cases where QBF is not accurate enough. In doing so, we will extend first ideas from [3] and we will investigate other approaches going beyond a generalization of QDLL approaches. Furthermore we will expand our approach to bounded model checking on sequential circuits.

References

[1] Salman Azhar, Gary Peterson, and John Reif. Lower bounds for multiplayer non-cooperative games of incomplete information. *Journal of Computers and Mathematics with Applications*, 41:957–992, 2001.

[2] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[3] Andreas Fröhlich, Gergely Kovásznai, and Armin Biere. A DPLL algorithm for solving DQBF. In *Proc. of the Int'l. Workshop on Pragmatics of SAT (POS)*, Trento, Italy, 2012.

[4] Rajesh K. Gupta and Yervant Zorian. Introducing core-based system design. *IEEE Design & Test of Computers*, 14(4):15–25, 1997.

[5] Leon Henkin. Some remarks on infinitely long formulas. *Infinistic Methods*, pages 167–183, 1961.

[6] Marc Herbstritt, Bernd Becker, and Christoph Scholl. Advanced SAT-techniques for bounded model checking of blackbox designs. In Magdy S. Abadir, Li-C. Wang, and Jayanta Bhadra, editors, *Proc. of the Int'l Workshop on Microprocessor Test and Verification (MTV)*, pages 37–44. IEEE, 2006.

[7] Christian Miller, Karina Gitina, Christoph Scholl, and Bernd Becker. Bounded model checking of incomplete networks of timed automata. In Magdy S. Abadir, Jay Bhadra, and Li-C. Wang, editors, *Proc. of the Int'l Workshop on Microprocessor Test and Verification (MTV)*, pages 61–66. IEEE, 2010.

[8] Georges Morbé, Florian Pigorsch, and Christoph Scholl. Fully symbolic model checking for timed automata. In *Proc. of the Int'l Conf. on Computer-Aided Verification (CAV)*, volume 6806 of *Lecture Notes in Computer Science*, pages 616–632. Springer-Verlag, 2011.

[9] Tobias Nopper and Christoph Scholl. Approximate symbolic model checking for incomplete designs. In Alan J. Hu and Andrew K. Martin, editors, *Proc. of the Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, volume 3312 of *Lecture Notes in Computer Science*, pages 290–305. Springer-Verlag, 2004.

[10] Tobias Nopper, Christoph Scholl, and Bernd Becker. Computation of minimal counterexamples by using black box techniques and symbolic methods. In Georges G. E. Gielen, editor, *Proc. of the Int'l Conf. on Computer-Aided Design (ICCAD)*, pages 273–280. IEEE, 2007.

[11] Christoph Scholl and Bernd Becker. Checking equivalence for partial implementations. In *Proc. of the Design Automation Conference (DAC)*, pages 238–243. ACM Press, 2001.

[12] Christoph Scholl and Bernd Becker. Checking equivalence for circuits containing incompletely specified boxes. In *Proc. of the Int'l Conf. on Computer Design (ICCD)*, pages 56–63. IEEE, 2002.

[13] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time (Preliminary Report). In *Proc. of the ACM Symposium on Theory of Computing (STOC)*, pages 1–9, New York, NY, USA, 1973. ACM Press.

[14] Grigorij Samuilovich Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, Part 2:115–125, 1970.

[15] Ralf Wimmer, Ernst Moritz Hahn, Holger Hermanns, and Bernd Becker. Reachability analysis for incomplete networks of Markov decision processes. In Satnam Singh, Barbara Jobstmann, Michael Kishinevsky, and Jens Brandt, editors, *Proc. of the Int'l Conf. on Formal Methods and Models for Codesign (MEMOCODE)*, pages 151–160. IEEE, 2011.