Fast DQBF Refutation

17th International Conference on Theory and Applications of Satisfiability Testing (SAT)

Bernd Finkbeiner
Leander Tentrup

Reactive Systems Group
Saarland University

July 16th, 2014

DQBF

Dependency Quantified Boolean Formula

$$\forall x_1, x_2, x_3 \ldots \quad \exists_{H_1} y_1. \quad \exists_{H_2} y_2. \quad \exists_{H_3} y_3 \ldots \quad \varphi$$

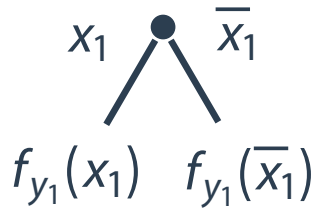
DQBF

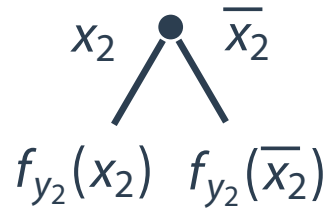
Dependency Quantified Boolean Formula

$$\forall x_1, x_2, x_3 \ldots \quad \exists_{H_1} y_1. \quad \exists_{H_2} y_2. \quad \exists_{H_3} y_3 \ldots \quad \varphi$$

Example

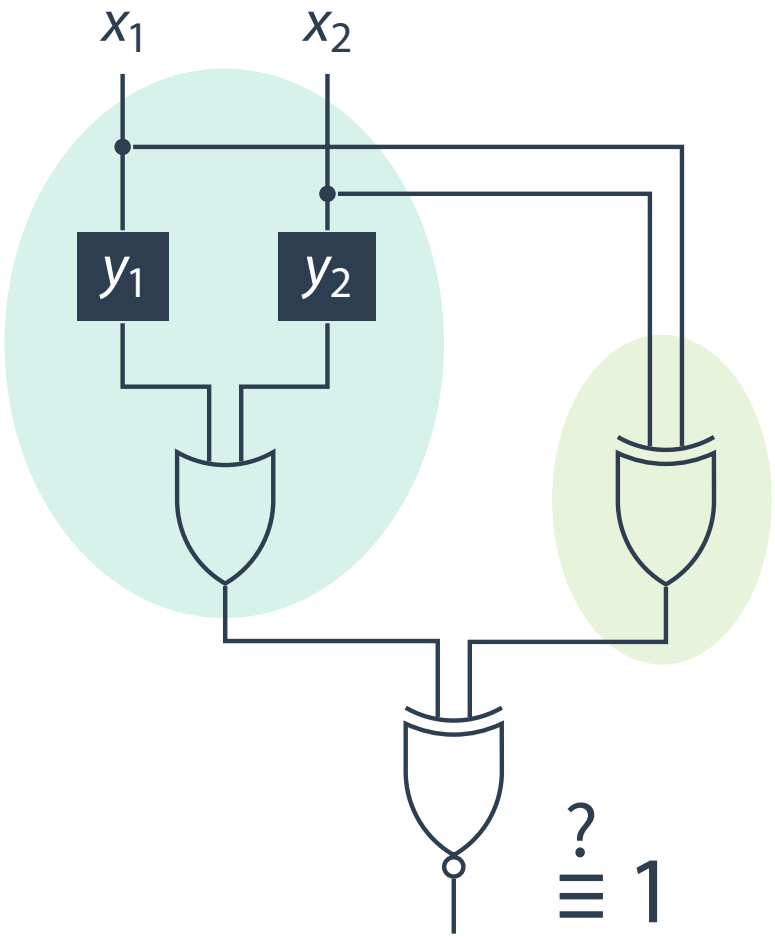
$\forall x_1, x_2. \exists_{\{x_1\}} y_1. \exists_{\{x_2\}} y_2. \varphi$





Partial Equivalence Checking (PEC)[1]

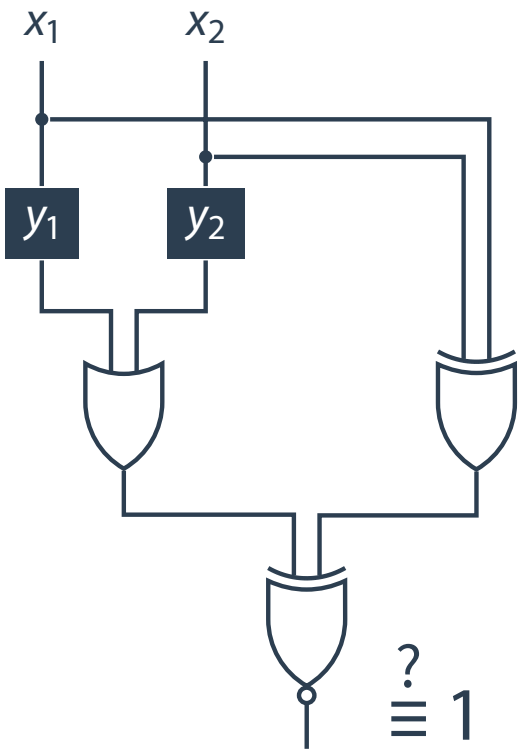
Partial Circuit



Specification

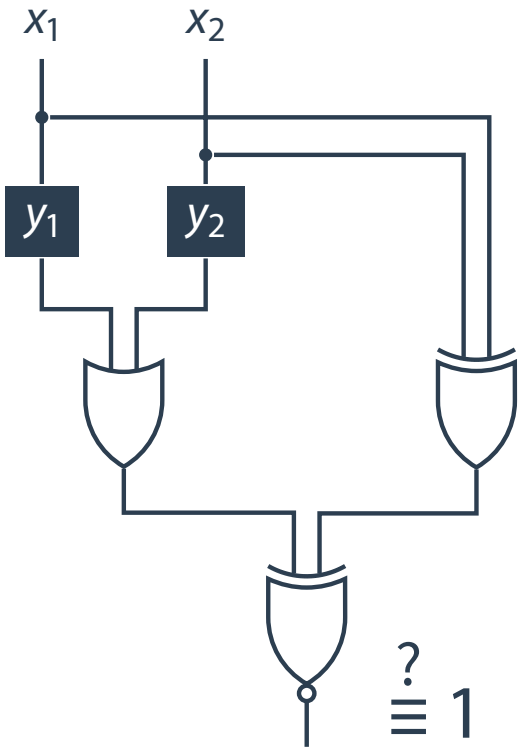
[1] Christoph Scholl and Bernd Becker. "Checking Equivalence for Partial Implementations". In: DAC '01. New York, NY, USA: ACM, 2001, pp. 238–243

Encoding of PEC in DQBF



$$\Leftrightarrow \quad \forall x_1, x_2.\, \exists_{\{x_1\}} y_1.\, \exists_{\{x_2\}} y_2.\, (y_1 \vee y_2) \leftrightarrow (x_1 \otimes x_2)$$

Encoding of PEC in DQBF



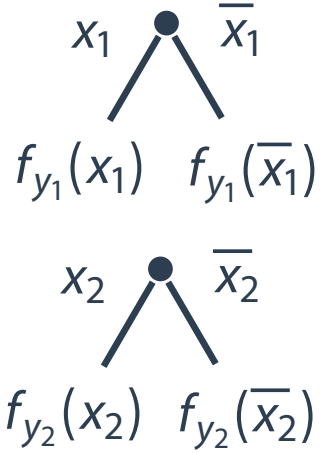
$$\Leftrightarrow \quad \forall x_1, x_2.\, \exists_{\{x_1\}} y_1.\, \exists_{\{x_2\}} y_2.\, (y_1 \vee y_2) \leftrightarrow (x_1 \otimes x_2)$$

This paper: **Focus on refutation ⇒ Detect errors in PEC instances**

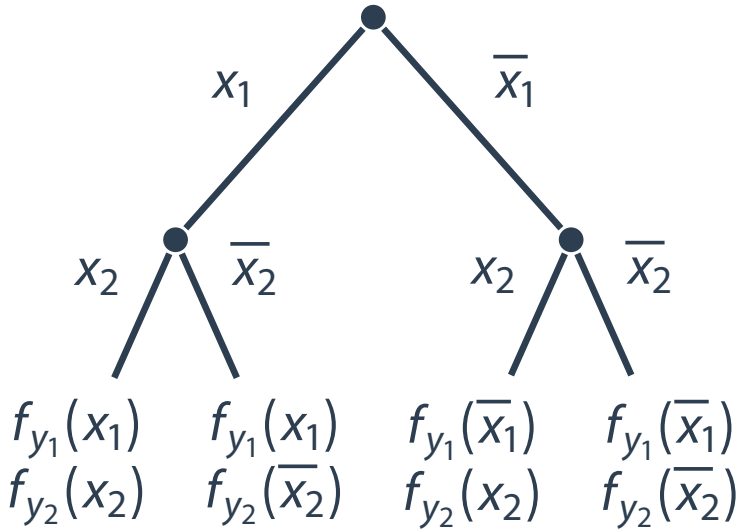
Models of DQBF

A *candidate model* is a composition of Skolem functions

Example ($\forall x_1, x_2.\, \exists_{\{x_1\}} y_1.\, \exists_{\{x_2\}} y_2.\, \varphi$)



$\Rightarrow$



Definition (Model M)

A model is a candidate model where all paths satisfy φ.

DQBF Solving

Exact methods

- Fröhlich et al.: A DPLL Algorithm for Solving DQBF
- Gitina et al.: non-public expansion-based solver
- Fröhlich et al.: iDQ, instantiation-based DQBF solver

QBF Abstraction

Gitina et al.: Equivalence checking of partial designs using DQBF

Expansion-Based Solving

Idea: **Expand the whole BDT**

$$\forall x_1, x_2. \exists_{\{x_1\}} y_1. \exists_{\{x_2\}} y_2. \varphi$$

Eliminate quantifier x_1:

$$\forall x_2. \exists_{\{x_2\}} y_2. \exists_{\varnothing} y_1, y_1'. \varphi|_{x_1=0} \wedge \varphi'|_{x_1=1}$$

Exponential blow-up

QBF Abstraction

Idea: **Linearize the Quantification**

$$\forall x_1, x_2.\ \exists_{\{x_1\}} y_1.\ \exists_{\{x_2\}} y_2.\ \varphi$$

3 possible linearizations:

$$\forall x_1, x_2.\ \exists y_1, y_2. \qquad \text{weakest}$$

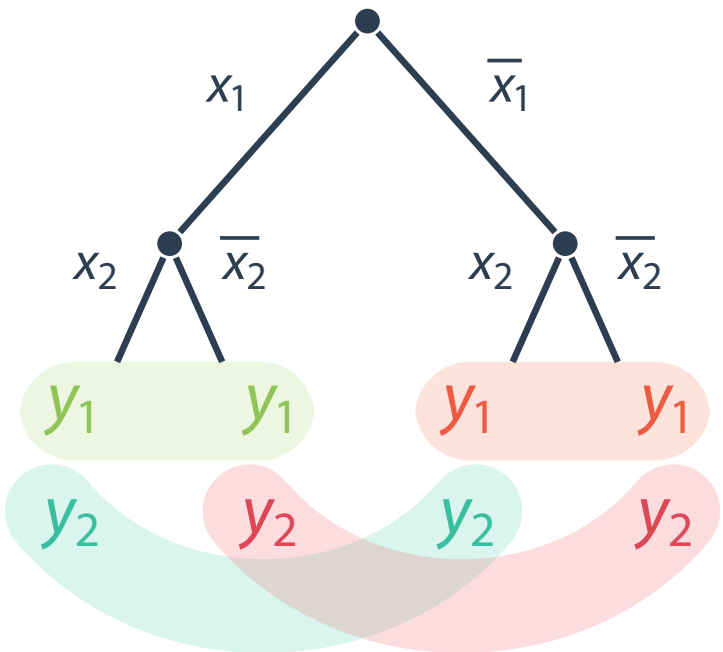
$$\left.\begin{array}{l} \forall x_1.\ \exists y_1.\ \forall x_2.\ \exists y_2. \\ \forall x_2.\ \exists y_2.\ \forall x_1.\ \exists y_1. \end{array}\right\} \qquad \text{strongest}$$

Too coarse: All QBF abstractions of
$\forall x_1, x_2.\ \exists_{\{x_1\}} y_1.\ \exists_{\{x_2\}} y_2.\ (y_1 \vee y_2) \leftrightarrow (x_1 \otimes x_2)$ are satisfiable

Consistency

- Another characterization of Skolem functions
- Property **between** paths

Different assignments of y on two paths P_1 and P_2 are *consistent* if the assignments of H_y on P_1 and P_2 are different

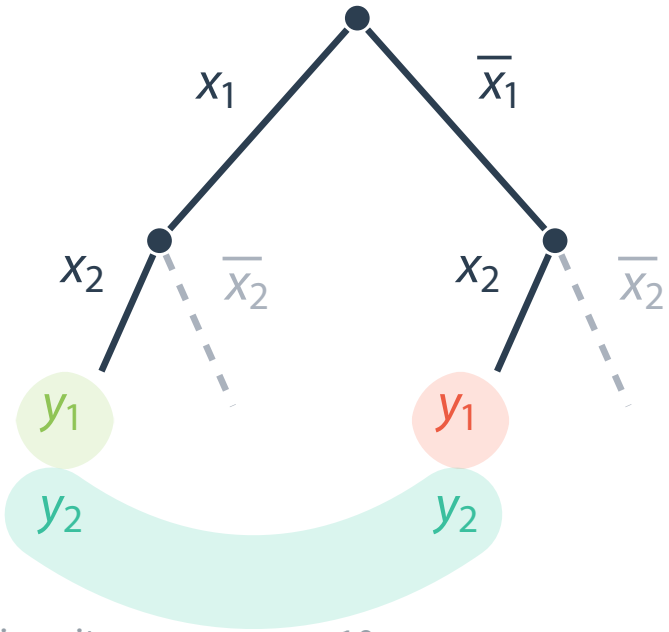


Bounded Unsatisfiability

Unsatisifability can be usually shown without expanding the whole BDT

A set of paths $\mathcal{P}$ can already rule out any satisfying assignment:

If there is no *consistent* satisfying assignment on $\mathcal{P}$, then there is no consistent satisfying assignment for the original formula



Bounded Unsatisfiability

Definition (*k*-bounded unsatisfiable)

For a $k \geq 1$, a DQBF formula Φ is *k-bounded unsatisfiable* if there exists a set of paths $\mathcal{P}$ with $|\mathcal{P}| \leq k$ such that there does not exist a consistent satisfying assignment over $\mathcal{P}$.

Bounded Unsatisfiability

Definition (*k*-bounded unsatisfiable)

For a $k \geq 1$, a DQBF formula Φ is *k-bounded unsatisfiable* if there exists a set of paths $\mathcal{P}$ with $|\mathcal{P}| \leq k$ such that there does not exist a consistent satisfying assignment over $\mathcal{P}$.

Theorem

A DQBF formula Φ is unsatisfiable iff it is k-bounded unsatisfiable for some $k \geq 1$.

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths
- such that for every *consistent* assignment of the existential variables

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths
- such that for every *consistent* assignment of the existential variables
- at least one path violates the matrix

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths
- such that for every *consistent* assignment of the existential variables
- at least one path violates the matrix

$$\Phi = \forall x_1, x_2. \exists_{\{x_1\}} y_1. \exists_{\{x_2\}} y_2. \varphi$$

bunsat($\Phi, k = 2$) =

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths
- such that for every *consistent* assignment of the existential variables
- at least one path violates the matrix

$$\Phi = \forall x_1, x_2. \exists_{\{x_1\}} y_1. \exists_{\{x_2\}} y_2. \varphi$$

$$\text{bunsat}(\Phi, k = 2) = \exists x_1^1, x_1^2, x_2^1, x_2^2.$$

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths
- such that for every *consistent* assignment of the existential variables
- at least one path violates the matrix

$$\Phi = \forall x_1, x_2. \exists_{\{x_1\}} y_1. \exists_{\{x_2\}} y_2. \varphi$$

$$\text{bunsat}(\Phi, k = 2) = \exists x_1^1, x_1^2, x_2^1, x_2^2. \forall y_1^1, y_1^2, y_2^1, y_2^2.$$

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths
- such that for every *consistent* assignment of the existential variables
- at least one path violates the matrix

$$\Phi = \forall x_1, x_2. \exists_{\{x_1\}} y_1. \exists_{\{x_2\}} y_2. \varphi$$

$$\text{bunsat}(\Phi, k = 2) = \exists x_1^1, x_1^2, x_2^1, x_2^2. \forall y_1^1, y_1^2, y_2^1, y_2^2.$$
$$\bigwedge_{i \in \{1,2\}} ((y_i^1 \leftrightarrow y_i^2) \vee (x_i^1 \not\leftrightarrow x_i^2)) \rightarrow$$

Encoding of Bounded Unsat. in QBF

Given a bound $k \geq 1$. Encode a QBF query that

- asserts that there exists k paths
- such that for every *consistent* assignment of the existential variables
- at least one path violates the matrix

$$\Phi = \forall x_1, x_2. \exists_{\{x_1\}} y_1. \exists_{\{x_2\}} y_2. \varphi$$

$$\text{bunsat}(\Phi, k = 2) = \exists x_1^1, x_1^2, x_2^1, x_2^2. \forall y_1^1, y_1^2, y_2^1, y_2^2.$$
$$\bigwedge_{i \in \{1,2\}} ((y_i^1 \leftrightarrow y_i^2) \vee (x_i^1 \not\leftrightarrow x_i^2)) \rightarrow$$
$$\neg\varphi^1 \vee \neg\varphi^2$$

Encoding of Bounded Unsat. in QBF

$$\Phi := \forall x_1, x_2, \ldots x_m . \exists_{H_1} y_1 . \exists_{H_2} y_2 \ldots \exists_{H_n} y_n . \varphi$$

$$\begin{aligned}\text{bunsat}(\Phi, k) := \exists & x_1^1, \ldots, x_m^1, x_1^2, \ldots, x_m^2, \ldots, x_m^k . \\ \forall & y_1^1, \ldots, y_n^1, y_1^2, \ldots, y_n^2, \ldots, y_n^k . \\ & \text{consistent}(\{y_1, \ldots, y_n\}, k) \rightarrow \bigvee_{1 \leq i \leq k} \neg \varphi^k\end{aligned}$$

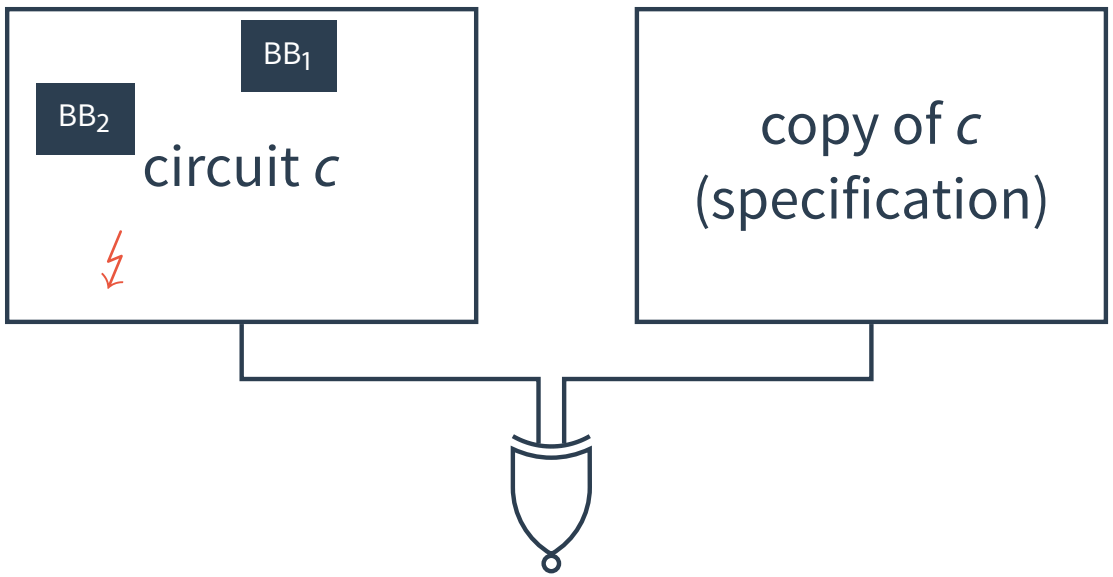
$$\text{consistent}(Y, k) := \bigwedge_{y \in Y} \bigwedge_{(i,j) \in \{1,\ldots,k\}^2} \left((y^i \leftrightarrow y^j) \vee \left(\bigvee_{x \in H_y} x^i \not\leftrightarrow x^j \right) \right)$$

Size of bunsat(Φ, k)

$|X'| = k \cdot |X|$
$|Y'| = k \cdot |Y|$
$|\varphi'| = O(|Y| \cdot k^2 \cdot \max_{y \in Y} |H_y| + k \cdot |\varphi|)$

Experimental Results
PEC Benchmarks

- There were no publicly available benchmarks for DQBF
- Build PEC examples[2]
  - Base: arithmetic circuit
  - One copy as specification
  - Introduce faults and black boxes randomly



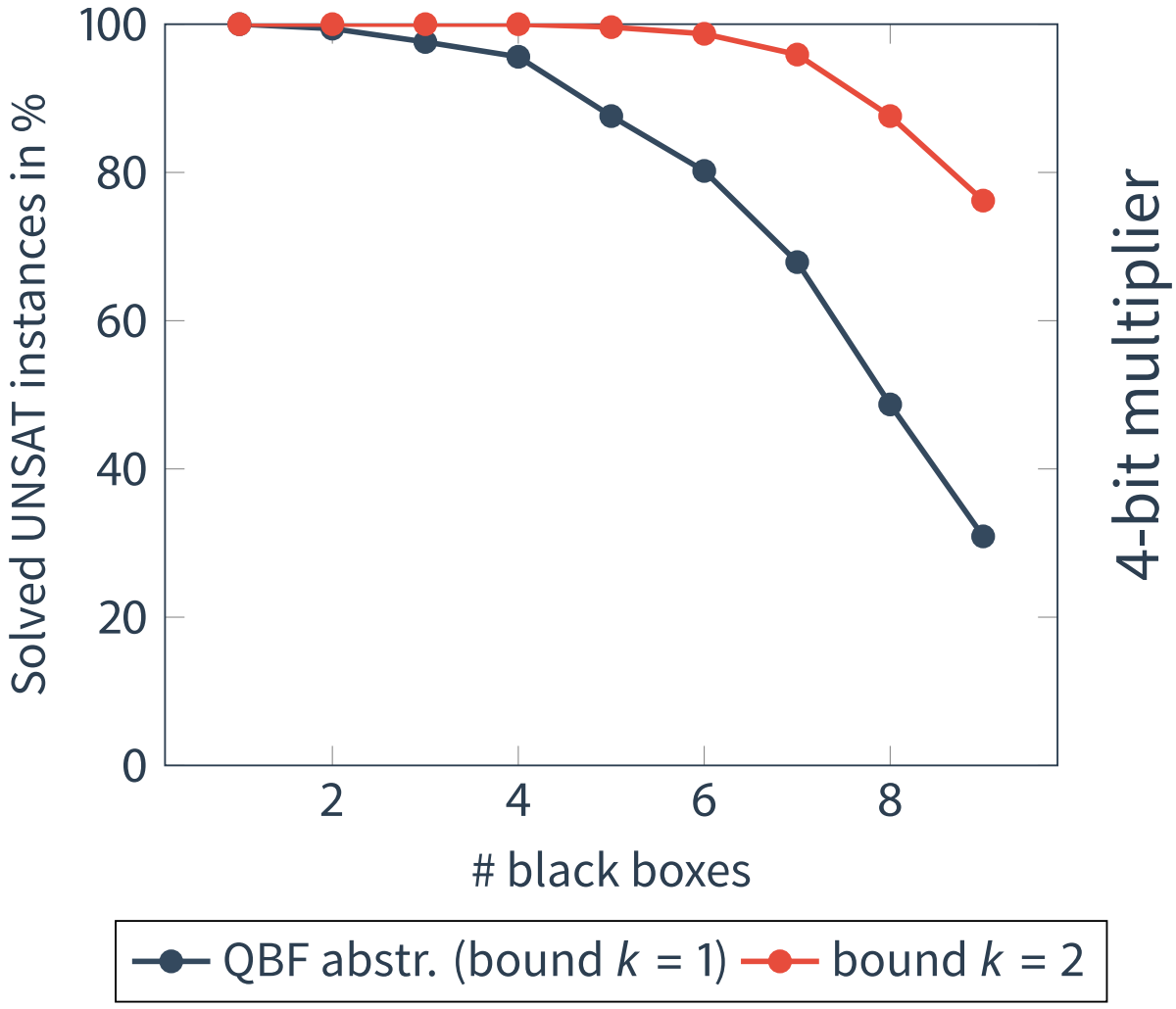
[2] Karina Gitina et al. "Equivalence checking of partial designs using dependency quantified Boolean formulae". In: *ICCD*. IEEE, 2013, pp. 396–403

Experimental Results
Solvers

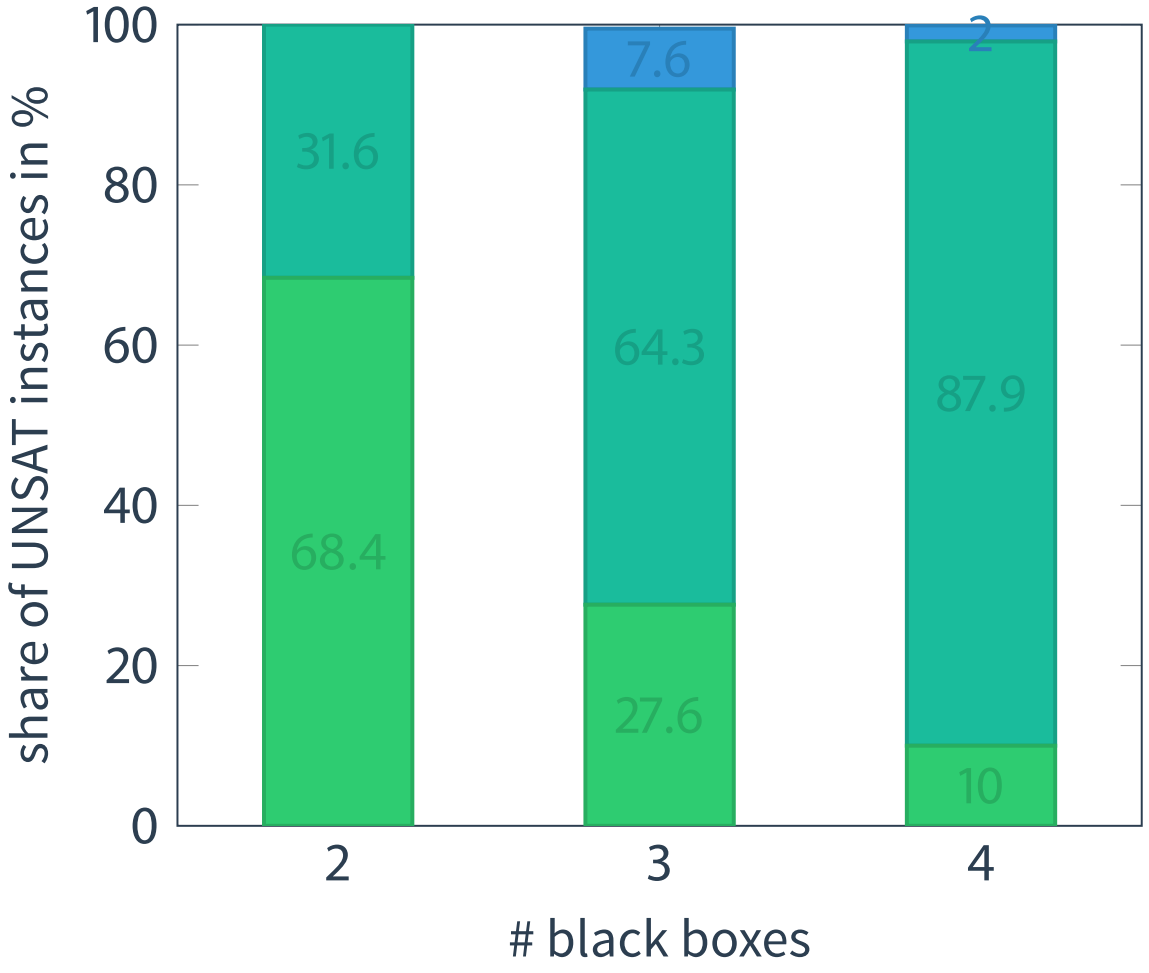
- Bounded Unsatisfiability:
  - Prototype implementing the transformation
  - Optimization: Combine with QBF abstraction
  - QBF queries: Bloqqer 031 and DepQBF 2.0
- Solvers:
  - There were no publicly available DQBF solver
  - ➤ Build an expansion-based solver on top of CUDD

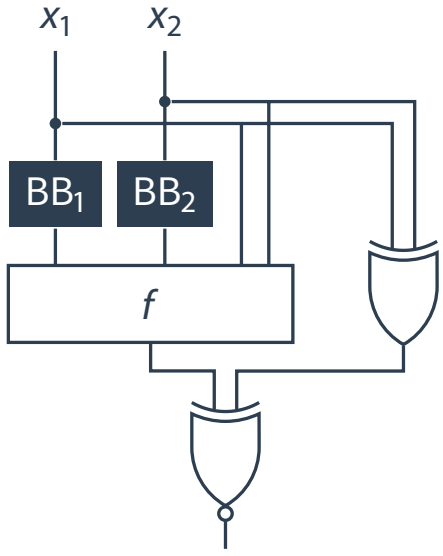
All sources are available at
`http://react.uni-saarland.de/tools/bunsat/`

Number of Solved Instances



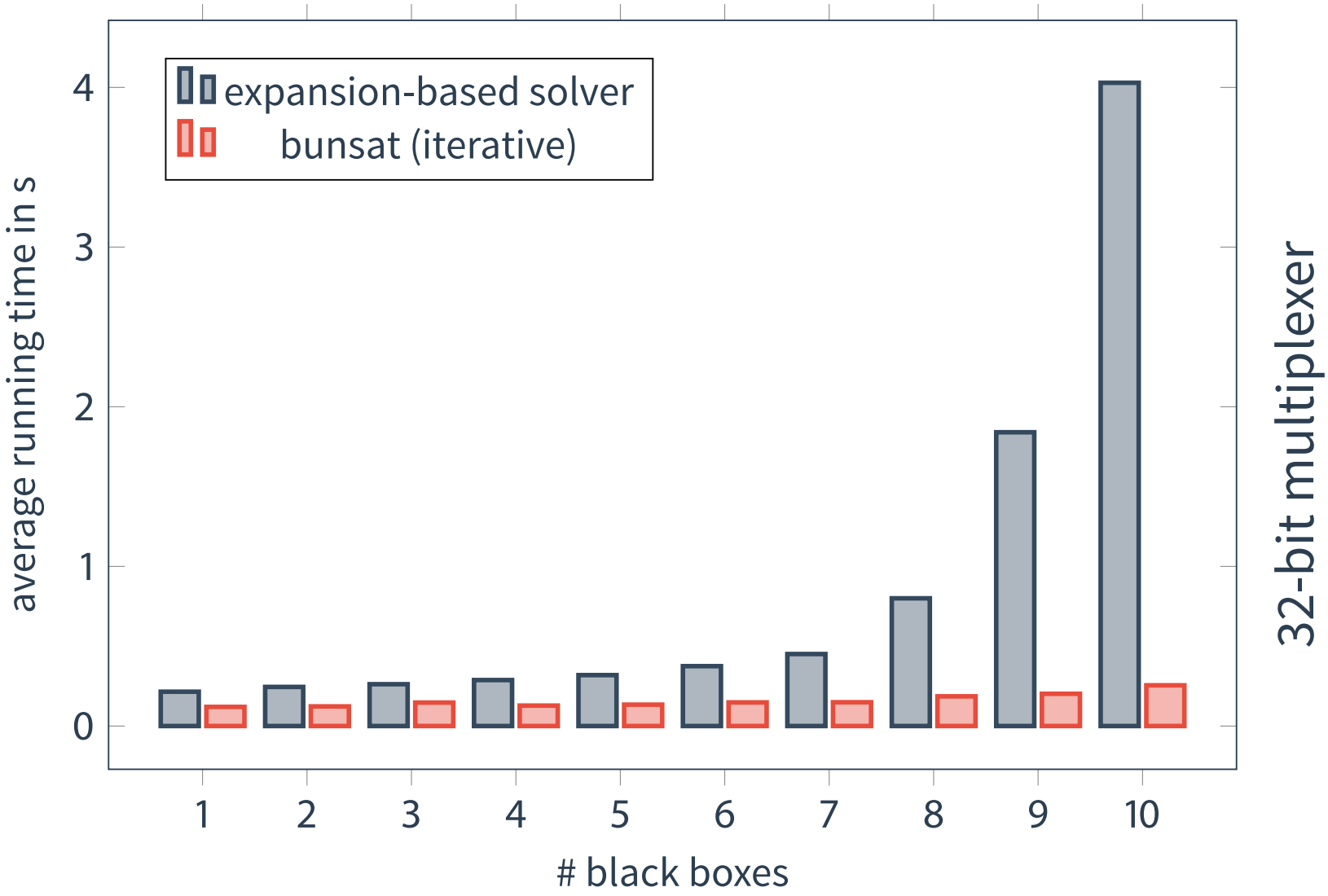
Precision





QBF abstr. (bound $k = 1$) bound $k = 2$ bound $k = 3$

Performance



Conclusion

- Fast DQBF refutation
- Improvement over
  - QBF abstraction (approximation quality)
  - expansion based methods (better scaling with respect to number of black boxes)

Conclusion

- Fast DQBF refutation
- Improvement over
  - QBF abstraction (approximation quality)
  - expansion based methods (better scaling with respect to number of black boxes)

Thank You!