On Proofs for SAT and QBF

Benjamin Kiesl

Vienna University of Technology, Institute of Information Systems
Topic of the Talk

- This talk is about our work during the last four months.
This talk is about our work during the last four months.

Our work centered around proofs for SAT and QBF. It resulted in two papers and one abstract:
This talk is about our work during the last four months.

Our work centered around proofs for SAT and QBF. It resulted in two papers and one abstract:

Marijn J.H. Heule, Benjamin Kiesl, and Armin Biere: Short Proofs Without New Variables
(Accepted at CADE)
This talk is about our work during the last four months. Our work centered around proofs for SAT and QBF. It resulted in two papers and one abstract:

Marijn J.H. Heule, Benjamin Kiesl, and Armin Biere: *Short Proofs Without New Variables*  
(Accepted at CADE)

Benjamin Kiesl, Marijn J.H. Heule, and Martina Seidl: *A Little Blocked Literal Goes a Long Way*  
(Submitted to SAT)
This talk is about our work during the last four months.

Our work centered around proofs for SAT and QBF. It resulted in two papers and one abstract:

Marijn J.H. Heule, Benjamin Kiesl, and Armin Biere: Short Proofs Without New Variables (Accepted at CADE)

Benjamin Kiesl, Marijn J.H. Heule, and Martina Seidl: A Little Blocked Literal Goes a Long Way (Submitted to SAT)

Marijn J.H. Heule and Benjamin Kiesl: The Potential of Interference-Based Proof Systems (Extended Abstract, Submitted to the ARCADE workshop)
Outline

- Overview on SAT and corresponding proofs.
  - What are proofs and why do we care about them?
Outline

- Overview on SAT and corresponding proofs.
  - What are proofs and why do we care about them?
- Short summary of our first paper:
  - In the paper, we introduce new proof systems for SAT solving.
Overview on SAT and corresponding proofs.
  • What are proofs and why do we care about them?

Short summary of our first paper:
  • In the paper, we introduce new proof systems for SAT solving.

Short summary of our second paper:
  • We show how two important proof systems for QBF are related.
The Satisfiability Problem of Propositional Logic (SAT)

- Given a propositional formula in conjunctive normal form, is it satisfiable?
The Satisfiability Problem of Propositional Logic (SAT)

- Given a propositional formula in conjunctive normal form, is it satisfiable?
- A literal is a variable \( x \) or the negation \( \bar{x} \) of a variable \( x \).
The Satisfiability Problem of Propositional Logic (SAT)

- Given a propositional formula in conjunctive normal form, is it satisfiable?
- A literal is a variable \( x \) or the negation \( \bar{x} \) of a variable \( x \).
- A clause is a disjunction \( l_1 \lor \cdots \lor l_n \) of literals.
The Satisfiability Problem of Propositional Logic (SAT)

- Given a propositional formula in conjunctive normal form, is it satisfiable?
- A literal is a variable $x$ or the negation $\overline{x}$ of a variable $x$.
- A clause is a disjunction $l_1 \lor \cdots \lor l_n$ of literals.
- A formula (in CNF) is a conjunction $C_1 \land \cdots \land C_n$ of clauses.
The Satisfiability Problem of Propositional Logic (SAT)

- Given a propositional formula in conjunctive normal form, is it satisfiable?
- A literal is a variable $x$ or the negation $\bar{x}$ of a variable $x$.
- A clause is a disjunction $l_1 \lor \cdots \lor l_n$ of literals.
- A formula (in CNF) is a conjunction $C_1 \land \cdots \land C_n$ of clauses.

Example:

$$(a \lor \bar{b}) \land (c) \land (\bar{a} \lor \bar{c})$$
A (truth) assignment is a mapping from variables to the truth values 0 (false) and 1 (true).
A (truth) assignment is a mapping from variables to the truth values 0 (false) and 1 (true).

An assignment \( \tau \) satisfies . . .

- . . . a variable \( x \) if \( \tau(x) = 1 \).
- . . . a literal \( l \) if \( l = x \) and \( \tau(x) = 1 \), or \( l = \overline{x} \) and \( \tau(x) = 0 \).
- . . . a clause if it satisfies at least one literal in the clause.
- . . . a formula if it satisfies all its clauses.
A (truth) assignment is a mapping from variables to the truth values 0 (false) and 1 (true).

An assignment $\tau$ satisfies...

- ...a variable $x$ if $\tau(x) = 1$.
- ...a literal $l$ if $l = x$ and $\tau(x) = 1$, or $l = \overline{x}$ and $\tau(x) = 0$. 
A (truth) assignment is a mapping from variables to the truth values 0 (false) and 1 (true).

An assignment $\tau$ satisfies . . .

- . . . a variable $x$ if $\tau(x) = 1$.
- . . . a literal $l$ if $l = x$ and $\tau(x) = 1$, or $l = \bar{x}$ and $\tau(x) = 0$.
- . . . a clause if it satisfies at least one literal in the clause.
A (truth) assignment is a mapping from variables to the truth values 0 (false) and 1 (true).

An assignment \( \tau \) satisfies...

- ...a variable \( x \) if \( \tau(x) = 1 \).
- ...a literal \( l \) if \( l = x \) and \( \tau(x) = 1 \), or \( l = \overline{x} \) and \( \tau(x) = 0 \).
- ...a clause if it satisfies at least one literal in the clause.
- ...a formula if it satisfies all its clauses.
A (truth) assignment is a mapping from variables to the truth values 0 (false) and 1 (true).

An assignment \( \tau \) satisfies...

- ... a variable \( x \) if \( \tau(x) = 1 \).
- ... a literal \( l \) if \( l = x \) and \( \tau(x) = 1 \), or \( l = \bar{x} \) and \( \tau(x) = 0 \).
- ... a clause if it satisfies at least one literal in the clause.
- ... a formula if it satisfies all its clauses.

**SAT:**

- Given a formula \( F \), does there exist an assignment that satisfies \( F \)?
The Satisfiability Problem of Propositional Logic (SAT)

\[(x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z})\]
The Satisfiability Problem of Propositional Logic (SAT)

Input Formula

\[(x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z})\]
The Satisfiability Problem of Propositional Logic (SAT)

Formulas can be seen as sets of clauses

\[ \{ x \lor y, \quad \overline{x} \lor \overline{y}, \quad z \lor \overline{z} \} \]
The Satisfiability Problem of Propositional Logic (SAT)

Clauses can be seen as sets of literals

\[
\{\{x, y\}, \{\bar{x}, \bar{y}\}, \{z, \bar{z}\}\}
\]
The Satisfiability Problem of Propositional Logic (SAT)

\[(x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z})\]
The Satisfiability Problem of Propositional Logic (SAT)

\((x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z})\)
The Satisfiability Problem of Propositional Logic (SAT)

\[(x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z})\]
The Satisfiability Problem of Propositional Logic (SAT)

\[(x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z})\]
The Satisfiability Problem of Propositional Logic (SAT)

\[(x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z})\]
The Satisfiability Problem of Propositional Logic (SAT)

\[ \overline{x} \land y \land (x \lor \overline{y}) \land (z \lor \overline{z}) \]

Input Formula

Formulas can be seen as sets of clauses
Clauses can be seen as sets of literals
The Satisfiability Problem of Propositional Logic (SAT)

\[(\overline{x} \land y \land (x \lor \overline{y}) \land (z \lor \overline{z}))\]
The Satisfiability Problem of Propositional Logic (SAT)

\[ \overline{x} \land y \land (x \lor \overline{y}) \land (z \lor \overline{z}) \]

Input Formula

Formulas can be seen as sets of clauses
Clauses can be seen as sets of literals
The Satisfiability Problem of Propositional Logic (SAT)

\[ (\overline{x} \lor y) \land \overline{(x \lor \overline{y})} \land (z \lor \overline{z}) \]

Unsatisfiable
Certifying Satisfiability and Unsatisfiability

- Certifying **satisfiability** of a formula is easy:

  - Just consider a satisfying assignment: $x \overline{y} z \left( x \lor y \right) \land \left( \overline{x} \lor \overline{y} \right) \land \left( z \lor \overline{z} \right)$
  - We can easily check that the assignment is satisfying: Just check for every clause if it has a satisfied literal!

Certifying **unsatisfiability** is not so easy:

- If a formula has $n$ variables, there are $2^n$ possible assignments.
- Checking whether every assignment falsifies the formula is costly.
- More compact certificates of unsatisfiability are desirable.
Certifying Satisfiability and Unsatisfiability

- Certifying **satisfiability** of a formula is easy:
  - Just consider a **satisfying assignment**:
    \[(x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z})\]
Certifying Satisfiability and Unsatisfiability

Certifying satisfiability of a formula is easy:

- Just consider a satisfying assignment: \( x\bar{y}z \)

\[
(x \lor \bar{y}) \land (x \lor \bar{y}) \land (z \lor \bar{z})
\]
Certifying Satisfiability and Unsatisfiability

Certifying \textit{satisfiability} of a formula is easy:

- Just consider a \textit{satisfying assignment}: $x\bar{y}z$

$$ (x \lor \bar{y}) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z}) $$

- We can easily check that the assignment is satisfying:
  Just check for every clause if it has a satisfied literal!
Certifying Satisfiability and Unsatisfiability

- **Certifying satisfiability** of a formula is easy:
  - Just consider a satisfying assignment: \( x\overline{y}z \)
  
  \[
  (x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z})
  \]

  - We can easily check that the assignment is satisfying: Just check for every clause if it has a satisfied literal!

- **Certifying unsatisfiability** is not so easy:
Certifying Satisfiability and Unsatisfiability

Certifying **satisfiability** of a formula is easy:

- Just consider a **satisfying assignment**: \( x\bar{y}z \)
  
  \[ (x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z}) \]

- We can easily check that the assignment is satisfying:
  Just check for every clause if it has a satisfied literal!

Certifying **unsatisfiability** is not so easy:

- If a formula has \( n \) variables, there are \( 2^n \) possible assignments.
Certifying Satisfiability and Unsatisfiability

- **Certifying satisfiability** of a formula is easy:
  - Just consider a satisfying assignment: $x\bar{y}z$

    $$(x \lor y) \land (x \lor \bar{y}) \land (z \lor \bar{z})$$

  - We can easily check that the assignment is satisfying:
    Just check for every clause if it has a satisfied literal!

- **Certifying unsatisfiability** is not so easy:
  - If a formula has $n$ variables, there are $2^n$ possible assignments.
  - Checking whether every assignment falsifies the formula is costly.
Certifying Satisfiability and Unsatisfiability

Certifying **satisfiability** of a formula is easy:

- Just consider a **satisfying assignment**: \( x\overline{y}z \)

\[ (x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z}) \]

- We can easily check that the assignment is satisfying:
  Just check for every clause if it has a satisfied literal!

Certifying **unsatisfiability** is not so easy:

- If a formula has \( n \) variables, there are \( 2^n \) possible assignments.
- Checking whether **every** assignment falsifies the formula is costly.
- More compact certificates of unsatisfiability are desirable.
Certifying Satisfiability and Unsatisfiability

- **Certifying satisfiability** of a formula is easy:
  - Just consider a satisfying assignment: \( x\bar{y}z \)
  - \((x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z})\)
  - We can easily check that the assignment is satisfying:
    Just check for every clause if it has a satisfied literal!

- **Certifying unsatisfiability** is not so easy:
  - If a formula has \( n \) variables, there are \( 2^n \) possible assignments.
  - Checking whether every assignment falsifies the formula is costly.
  - More compact certificates of unsatisfiability are desirable.
    - Proofs
What Is a Proof in SAT?

- In general, a **proof** is a **string** that **certifies** the unsatisfiability of a formula.
What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable.
What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable (but can be of exponential size with respect to a formula)
What Is a Proof in SAT?

- In general, a **proof** is a **string** that **certifies the unsatisfiability** of a formula.
  - Proofs are **efficiently** (usually polynomial-time) checkable (but can be of exponential size with respect to a formula)
- **Example:** Resolution proofs
What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable (but can be of exponential size with respect to a formula)
- Example: Resolution proofs
  - A resolution proof is a sequence $C_1, \ldots, C_n$ of clauses.
What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable (but can be of exponential size with respect to a formula)
- Example: Resolution proofs
  - A resolution proof is a sequence $C_1, \ldots, C_n$ of clauses.
  - Every clause is either contained in the formula or derived from two earlier clauses via the resolution rule:

$$
\begin{array}{c}
C \lor I \\
\bar{I} \lor D
\end{array} \Rightarrow 
\begin{array}{c}
C \lor D
\end{array}
$$
What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable (but can be of exponential size with respect to a formula)

Example: Resolution proofs

- A resolution proof is a sequence $C_1, \ldots, C_n$ of clauses.
- Every clause is either contained in the formula or derived from two earlier clauses via the resolution rule:

$$
\frac{C \lor I \quad \bar{I} \lor D}{C \lor D}
$$

- $C_n$ is the empty clause (containing no literals).
What Is a Proof in SAT?

- In general, a proof is a string that certifies the unsatisfiability of a formula.
  - Proofs are efficiently (usually polynomial-time) checkable (but can be of exponential size with respect to a formula)
- Example: Resolution proofs
  - A resolution proof is a sequence $C_1, \ldots, C_n$ of clauses.
  - Every clause is either contained in the formula or derived from two earlier clauses via the resolution rule:
    \[
    \frac{C \lor \bar{l} \lor D}{C \lor \bar{l} \lor D} \quad \frac{C \lor l}{C \lor D}
    \]
  - $C_n$ is the empty clause (containing no literals).
  - There exists a resolution proof for every unsatisfiable formula.
Resolution Proofs

Example: \( F = (\bar{x} \lor \bar{y} \lor z) \land (\bar{z}) \land (x \lor \bar{y}) \land (\bar{u} \lor y) \land (u) \)
Resolution Proofs

- Example: \( F = (\bar{x} \lor \bar{y} \lor z) \land (\bar{z}) \land (x \lor \bar{y}) \land (\bar{u} \lor y) \land (u) \)

- Resolution proof:
  \( (\bar{x} \lor \bar{y} \lor z), (\bar{z}), (\bar{x} \lor \bar{y}), (x \lor \bar{y}), (\bar{y}), (\bar{u} \lor y), (\bar{u}), (u), \emptyset \)
Resolution Proofs

- Example: \( F = (\bar{x} \lor \bar{y} \lor z) \land (\bar{z}) \land (x \lor \bar{y}) \land (\bar{u} \lor y) \land (u) \)

- Resolution proof:

\[
(\bar{x} \lor \bar{y} \lor z), (\bar{z}), (\bar{x} \lor \bar{y}), (x \lor \bar{y}), (\bar{y}), (\bar{u} \lor y), (\bar{u}), (u), \emptyset
\]

\[
\begin{array}{c}
\bar{x} \lor \bar{y} \lor z & \bar{z} \\
\hline
\bar{x} \lor \bar{y} & \bar{x} \lor \bar{y} \\
\bar{u} \lor y & \bar{u} \lor y \\
\bar{u} & \bar{u} \\
\hline
\emptyset
\end{array}
\]

Drawbacks of resolution:
- For many seemingly simple formulas, there are only resolution proofs of exponential size.
- State-of-the-art solving techniques are not succinctly expressible.
Resolution Proofs

- Example: \( F = (\overline{x} \lor \overline{y} \lor z) \land (\overline{z}) \land (x \lor \overline{y}) \land (\overline{u} \lor y) \land (u) \)

- Resolution proof:
  \( (\overline{x} \lor \overline{y} \lor z), (\overline{z}), (\overline{x} \lor \overline{y}), (x \lor \overline{y}), (\overline{y}), (\overline{u} \lor y), (\overline{u}), (u), \emptyset \)

- Drawbacks of resolution:
Resolution Proofs

- **Example:** \( F = (\bar{x} \lor \bar{y} \lor z) \land (\bar{z}) \land (x \lor \bar{y}) \land (\bar{u} \lor y) \land (u) \)

- **Resolution proof:**

\[
(\bar{x} \lor \bar{y} \lor z), (\bar{z}), (\bar{x} \lor \bar{y}), (x \lor \bar{y}), (\bar{y}), (\bar{u} \lor y), (\bar{u}), (u), \emptyset
\]

\[
\begin{array}{c}
\begin{array}{c}
\bar{x} \lor \bar{y} \lor z \\
\bar{z}
\end{array} \\
\hline
\bar{x} \lor \bar{y}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\bar{u} \lor y \\
\bar{u}
\end{array} \\
\hline
\emptyset
\end{array}
\]

- **Drawbacks** of resolution:
  - For many seemingly simple formulas, there are only resolution proofs of exponential size.
Resolution Proofs

- **Example:** \( F = (\bar{x} \lor \bar{y} \lor z) \land (\bar{z}) \land (x \lor \bar{y}) \land (\bar{u} \lor y) \land (u) \)

- **Resolution proof:**
  \[
  (\bar{x} \lor \bar{y} \lor z), (\bar{z}), (\bar{x} \lor \bar{y}), (x \lor \bar{y}), (\bar{y}), (\bar{u} \lor y), (\bar{u}), (u), \emptyset
  \]

- **Drawbacks of resolution:**
  - For many seemingly simple formulas, there are only resolution proofs of exponential size.
  - State-of-the-art solving techniques are not succinctly expressible.
Properties of a Desirable Proof System for SAT

1. Succinctness: Proofs of unsatisfiability should be short strings that certify the unsatisfiability of formulas.

2. Efficient Checkability: It should be easy to verify that a proof is correct, i.e., that it certifies the unsatisfiability of a formula.

3. Practicability: SAT solvers should be able to produce proofs. State-of-the-art techniques should be expressible in the system.

4. (Soundness and completeness.)
Properties of a Desirable Proof System for SAT

1. **Succinctness**: Proofs of unsatisfiability should be short strings that certify the unsatisfiability of formulas.
Properties of a Desirable Proof System for SAT

1. **Succinctness:** Proofs of unsatisfiability should be short strings that certify the unsatisfiability of formulas.

2. **Efficient Checkability:** It should be easy to verify that a proof is correct, i.e., that it certifies the unsatisfiability of a formula.
Properties of a Desirable Proof System for SAT

1. **Succinctness**: Proofs of unsatisfiability should be short strings that certify the unsatisfiability of formulas.

2. **Efficient Checkability**: It should be easy to verify that a proof is correct, i.e., that it certifies the unsatisfiability of a formula.

3. **Practicability**: SAT solvers should be able to produce proofs.
Properties of a Desirable Proof System for SAT

1. **Succinctness:** Proofs of unsatisfiability should be short strings that certify the unsatisfiability of formulas.

2. **Efficient Checkability:** It should be easy to verify that a proof is correct, i.e., that it certifies the unsatisfiability of a formula.

3. **Practicability:** SAT solvers should be able to produce proofs.
   - State-of-the-art techniques should be expressible in the system.
Properties of a Desirable Proof System for SAT

1. **Succinctness:** Proofs of unsatisfiability should be short strings that certify the unsatisfiability of formulas.

2. **Efficient Checkability:** It should be easy to verify that a proof is correct, i.e., that it certifies the unsatisfiability of a formula.

3. **Practicability:** SAT solvers should be able to produce proofs.
   - State-of-the-art techniques should be expressible in the system.

4. (Soundness and completeness.)
Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is implied by the premises.

\[
\frac{C \lor I}{C \lor D} \quad (\text{res}) \quad \frac{A \quad A \rightarrow B}{B} \quad (\text{mp})
\]

- Different approach: Allow not only implied conclusions.
- Require only that the addition of facts preserves satisfiability.
- Reason also about the absence of facts.

This leads to interference-based proof systems.
Traditional Proofs vs. Interference-Based Proofs

In traditional proof systems, everything that is inferred, is implied by the premises.

\[
\begin{align*}
\frac{C \lor I}{C \lor \bar{I} \lor \bar{D}} & \quad \text{(res)} \quad \frac{A}{A \rightarrow B} & \quad \text{(mp)}
\end{align*}
\]

When inferring something, we reason about the presence of facts.

- If certain premises are present, infer the conclusion.
Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is implied by the premises.

\[
\frac{C \lor l}{C \lor D} \quad (\text{res}) \quad \frac{A}{B} \quad (\text{mp})
\]

- When inferring something, we reason about the presence of facts.
  - If certain premises are present, infer the conclusion.

- Different approach: Allow not only implied conclusions.
Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is implied by the premises.

\[ \frac{C \lor I}{C \lor D} \quad \frac{\bar{I} \lor D}{C \lor D} \quad \frac{A}{B} \quad \frac{A \rightarrow B}{B} \]

(res) (mp)

- When inferring something, we reason about the presence of facts.
  - If certain premises are present, infer the conclusion.
- Different approach: Allow not only implied conclusions.
  - Require only that the addition of facts preserves satisfiability.
Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is implied by the premises.

\[
\frac{C \lor I}{C \lor D} \quad \frac{\bar{I} \lor D}{(\text{res})} \quad \frac{A}{B} \quad \frac{A \rightarrow B}{(\text{mp})}
\]

- When inferring something, we reason about the presence of facts.
  - If certain premises are present, infer the conclusion.

- Different approach: Allow not only implied conclusions.
  - Require only that the addition of facts preserves satisfiability.
  - Reason also about the absence of facts.
Traditional Proofs vs. Interference-Based Proofs

- In traditional proof systems, everything that is inferred, is implied by the premises.

\[
\frac{C \lor I}{C \lor D} \quad (\text{res}) \quad \frac{A}{B} \quad (\text{mp})
\]

- When inferring something, we reason about the presence of facts.
  - If certain premises are present, infer the conclusion.

- Different approach: Allow not only implied conclusions.
  - Require only that the addition of facts preserves satisfiability.
  - Reason also about the absence of facts.

- This leads to interference-based proof systems.
Interference-Based Proof Systems

- Interference-based proof systems generalize traditional proof systems.
Interference-Based Proof Systems

- **Interference-based proof systems** generalize traditional proof systems.
- An **interference-based proof** is a sequence of clauses.
Interference-Based Proof Systems

- Interference-based proof systems generalize traditional proof systems.
- An interference-based proof is a sequence of clauses.
  - **Idea**: The clauses are added to the initial formula step-by-step.
Interference-Based Proof Systems

- Interference-based proof systems generalize traditional proof systems.
- An interference-based proof is a sequence of clauses.
  - Idea: The clauses are added to the initial formula step-by-step.
  - Added clauses need not be implied, but their addition must preserve satisfiability:
Interference-Based Proof Systems

- **Interference-based proof systems** generalize traditional proof systems.

- An **interference-based proof** is a sequence of clauses.
  - **Idea**: The clauses are added to the initial formula step-by-step.
  - Added clauses need not be implied, but their addition must preserve **satisfiability**:
    - If the formula is satisfiable, then the formula obtained by adding the clause is also satisfiable.
Interference-Based Proof Systems

- **Interference-based proof systems** generalize traditional proof systems.

- An **interference-based proof** is a sequence of clauses.
  - **Idea**: The clauses are added to the initial formula step-by-step.
  - Added clauses need not be implied, but their addition must preserve satisfiability:
    - If the formula is satisfiable, then the formula obtained by adding the clause is also satisfiable.
    - If the (unsatisfiable) empty clause, $\emptyset$, can be added, then the original formula must be unsatisfiable.
Interference-Based Proof Systems

- **Interference-based proof systems** generalize traditional proof systems.

- An **interference-based proof** is a sequence of clauses.
  - **Idea**: The clauses are added to the initial formula step-by-step.
  - Added clauses need not be implied, but their addition must preserve satisfiability:
    - If the formula is satisfiable, then the formula obtained by adding the clause is also satisfiable.
    - If the (unsatisfiable) empty clause, $\emptyset$, can be added, then the original formula must be unsatisfiable.
      - The empty clause is unsatisfiable because it has no literal that could be true.
Interference-Based Proofs

It should be efficiently checkable whether clause additions preserve satisfiability. Clauses whose addition preserves satisfiability are called redundant.

Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.

- Example: Addition of resolution asymmetric tautologies (RATs).
Interference-Based Proofs

It should be efficiently checkable whether clause additions preserve satisfiability. Clauses whose addition preserves satisfiability are called redundant.

Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.

• Example: Addition of resolution asymmetric tautologies (RATs).
Interference-Based Proofs

It should be efficiently checkable whether clause additions preserve satisfiability. Clauses whose addition preserves satisfiability are called redundant.

Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.

• Example: Addition of resolution asymmetric tautologies (RATs).
Interference-Based Proofs

It should be efficiently checkable whether clause additions preserve satisfiability. Clauses whose addition preserves satisfiability are called redundant.

Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.

• Example: Addition of resolution asymmetric tautologies (RATs).
Interference-Based Proofs

It should be efficiently checkable whether clause additions preserve satisfiability. Clauses whose addition preserves satisfiability are called redundant. Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.

- Example: Addition of resolution asymmetric tautologies (RATs).
It should be efficiently checkable whether clause additions preserve satisfiability.
It should be efficiently checkable whether clause additions preserve satisfiability.

Clauses whose addition preserves satisfiability are called redundant.
It should be efficiently checkable whether clause additions preserve satisfiability.

- Clauses whose addition preserves satisfiability are called redundant.

⇒ Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.
Interference-Based Proofs

It should be efficiently checkable whether clause additions preserve satisfiability.

Clauses whose addition preserves satisfiability are called redundant.

Idea: Allow only the addition of clauses that fulfill an efficiently checkable redundancy criterion.

- Example: Addition of resolution asymmetric tautologies (RATs).
DRAT: An Interference-Based Proof System

- Popular example of an interference-based proof system: DRAT
Popular example of an interference-based proof system: DRAT

DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).
Popular example of an interference-based proof system: DRAT

- DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).
  - It can be efficiently checked if a clause is a RAT.
Popular example of an interference-based proof system: DRAT

DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).

- It can be efficiently checked if a clause is a RAT.
- RATs are not necessarily implied by the formula.
DRAT: An Interference-Based Proof System

- Popular example of an interference-based proof system: DRAT
- DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).
  - It can be efficiently checked if a clause is a RAT.
  - RATs are not necessarily implied by the formula.
  - But RATs are redundant: their addition preserves satisfiability.
DRAT: An Interference-Based Proof System

- Popular example of an interference-based proof system: DRAT
- DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).
  - It can be efficiently checked if a clause is a RAT.
  - RATs are not necessarily implied by the formula.
  - But RATs are redundant: their addition preserves satisfiability.
  - A RAT check involves reasoning about the absence of facts.
DRAT: An Interference-Based Proof System

- Popular example of an interference-based proof system: DRAT
- DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).
  - It can be efficiently checked if a clause is a RAT.
  - RATs are not necessarily implied by the formula.
  - But RATs are redundant: their addition preserves satisfiability.
  - A RAT check involves reasoning about the absence of facts.
    - A clause is a RAT w.r.t. a formula if the formula contains no clause such that . . .
Popular example of an interference-based proof system: DRAT

DRAT allows the addition of so-called resolution asymmetric tautologies (RATs) to a formula (whatever that means).

- It can be efficiently checked if a clause is a RAT.
- RATs are not necessarily implied by the formula.
- But RATs are redundant: their addition preserves satisfiability.
- A RAT check involves reasoning about the absence of facts.
  - A clause is a RAT w.r.t. a formula if the formula contains no clause such that . . .

Are there more general types of redundant clauses than RATs?
Redundant Clauses

- Strong proof systems allow the addition of many redundant clauses.
Redundant Clauses

- Strong proof systems allow the addition of many redundant clauses.
Redundant Clauses

- Strong proof systems allow the addition of many redundant clauses.
Redundant Clauses

- Strong proof systems allow the addition of many redundant clauses.

- Are there stronger redundancy notions that are efficiently checkable?
Strong proof systems allow the addition of many redundant clauses.

Are there stronger redundancy notions that are efficiently checkable?
Redundant Clauses

- Strong proof systems allow the addition of many redundant clauses.

- Are there stronger redundancy notions that are efficiently checkable?

⇒ Short Proofs Without New Variables
Short Proofs Without New Variables: Main Contributions

- We introduced **new clause-redundancy notions**:
  - Propagation-redundant (PR) clauses
  - Set-propagation-redundant (SPR) clauses
  - Literal-propagation-redundant (LPR) clauses
We introduced new clause-redundancy notions:

- Propagation-redundant (PR) clauses
- Set-propagation-redundant (SPR) clauses
- Literal-propagation-redundant (LPR) clauses

LPR clauses coincide with RAT.
We introduced new clause-redundancy notions:

- Propagation-redundant (PR) clauses
- Set-propagation-redundant (SPR) clauses
- Literal-propagation-redundant (LPR) clauses

LPR clauses coincide with RAT.

SPR clauses strictly generalize RATs.
We introduced new clause-redundancy notions:
- Propagation-redundant (PR) clauses
- Set-propagation-redundant (SPR) clauses
- Literal-propagation-redundant (LPR) clauses

LPR clauses coincide with RAT.
SPR clauses strictly generalize RATs.
PR clauses strictly generalize SPR clauses.
We introduced new clause-redundancy notions:
- Propagation-redundant (PR) clauses
- Set-propagation-redundant (SPR) clauses
- Literal-propagation-redundant (LPR) clauses

LPR clauses coincide with RAT.

SPR clauses strictly generalize RATs.

PR clauses strictly generalize SPR clauses.

The redundancy notions provide the basis for new proof systems.
New Landscape of Redundancy Notions

- SAT-EQ
- PR
- SPR
- LPR
- RAT
- RS
- SET
- BC
- EQ
- RUP
- S

- New

- Satisfiability equivalence

- Logical equivalence
The new proof systems can give short proofs of formulas that are considered hard.
The new proof systems can give short proofs of formulas that are considered hard.

We have short SPR and PR proofs for the well-known pigeon hole formulas.
Stronger Proof Systems: What Are They Good For?

- The new proof systems can give short proofs of formulas that are considered hard.
- We have short SPR and PR proofs for the well-known pigeon hole formulas.
  - Pigeon hole formulas have only exponential-size resolution proofs.
Stronger Proof Systems: What Are They Good For?

- The new proof systems can give short proofs of formulas that are considered hard.
- We have short SPR and PR proofs for the well-known pigeon hole formulas.
  - Pigeon hole formulas have only exponential-size resolution proofs.
  - If the addition of new variables via definitions is allowed, there are polynomial-size proofs.
The new proof systems can give short proofs of formulas that are considered hard.

We have short SPR and PR proofs for the well-known pigeon hole formulas.

- Pigeon hole formulas have only exponential-size resolution proofs.
- If the addition of new variables via definitions is allowed, there are polynomial-size proofs.
  - So-called extended resolution proofs.
The new proof systems can give short proofs of formulas that are considered hard.

We have short SPR and PR proofs for the well-known pigeon hole formulas.

- Pigeon hole formulas have only exponential-size resolution proofs.
- If the addition of new variables via definitions is allowed, there are polynomial-size proofs.
  - So-called extended resolution proofs.

Our proofs do not require new variables.

- Search space of possible clauses is finite.
The new proof systems can give short proofs of formulas that are considered hard.

We have short SPR and PR proofs for the well-known pigeon hole formulas.

- Pigeon hole formulas have only exponential-size resolution proofs.
- If the addition of new variables via definitions is allowed, there are polynomial-size proofs.
  - So-called extended resolution proofs.

Our proofs do not require new variables.

- Search space of possible clauses is finite.
- Makes search for such clauses easier.
Short Proofs Without New Variables: Conclusion

- We introduced new redundancy notions for SAT.
Short Proofs Without New Variables: Conclusion

- We introduced new redundancy notions for SAT.
- The redundancy notions strictly generalize RAT.

Open problem: Automatically generate such short proofs.
Short Proofs Without New Variables: Conclusion

- We introduced new redundancy notions for SAT.
- The redundancy notions strictly generalize RAT.
- Proof systems based on these redundancy notions are strong.
Short Proofs Without New Variables: Conclusion

- We introduced new redundancy notions for SAT.
- The redundancy notions strictly generalize RAT.
- Proof systems based on these redundancy notions are strong.
  - They allow for short proofs without new variables.
We introduced new redundancy notions for SAT.
The redundancy notions strictly generalize RAT.
Proof systems based on these redundancy notions are strong.
  • They allow for short proofs without new variables.
Short Proofs Without New Variables: Conclusion

- We introduced new redundancy notions for SAT.
- The redundancy notions strictly generalize RAT.
- Proof systems based on these redundancy notions are strong.
  - They allow for short proofs without new variables.
- Proofs for the pigeon hole formulas are hand-crafted.
Short Proofs Without New Variables: Conclusion

- We introduced new redundancy notions for SAT.
- The redundancy notions strictly generalize RAT.
- Proof systems based on these redundancy notions are strong.
  - They allow for short proofs without new variables.
- Proofs for the pigeon hole formulas are hand-crafted.
  - Open problem: Automatically generate such short proofs.
What the Reviewers Say

Reviewer 1:
“I find the topic interesting and I believe the authors did a great job when writing the paper (…) I believe this paper contains solid work that should be presented at CADE.”
What the Reviewers Say

Reviewer 1:
“I find the topic interesting and I believe the authors did a great job when writing the paper (…) I believe this paper contains solid work that should be presented at CADE.”

Reviewer 2:
“The paper is very well written, is easy to follow and a pleasure to read. The authors address an important problem.”
What the Reviewers Say

Reviewer 1:
“I find the topic interesting and I believe the authors did a great job when writing the paper (…) I believe this paper contains solid work that should be presented at CADE.”

Reviewer 2:
“The paper is very well written, is easy to follow and a pleasure to read. The authors address an important problem.”

Reviewer 3:
“The presented proof system is novel and powerful, the results in the paper are interesting, and the paper fits the scope of CADE.”
Jayadev Misra: 
“Good work is more important than good reviews.”
A Little Blocked Literal Goes a Long Way: Overview

- Deals with proofs for quantified Boolean formulas (QBFs).
Deals with proofs for quantified Boolean formulas (QBFs).

Short overview on QBF and corresponding proof systems.
A Little Blocked Literal Goes a Long Way: Overview

- Deals with proofs for quantified Boolean formulas (QBFs).
- Short overview on QBF and corresponding proof systems.
- We show that QRAT (the QBF generalization of DRAT) can polynomially simulate long-distance resolution.
Deals with proofs for quantified Boolean formulas (QBFs).

Short overview on QBF and corresponding proof systems.

We show that QRAT (the QBF generalization of DRAT) can polynomially simulate long-distance resolution.

We have an implementation and evaluation of the simulation.
Satisfiability of Quantified Boolean Formulas (QSAT)

“For every truth value of $x$, does there exist a truth value of $y$, such that . . .”

\[ \forall x \exists y \forall z \ (x \lor y) \land (\neg x \lor \neg y) \land (z \lor \neg z) \]
Satisfiability of Quantified Boolean Formulas (QSAT)

\[ \forall x \exists y \forall z \ (x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z}) \]
∀x∃y∀z (x ∨ y) ∧ (x ∨ y) ∧ (z ∨ z) ∧ (z ∨ z)
∀x∃y∀z (x ∨ y) ∧ (x ∨ y) ∧ (z ∨ z)
Satisfiability of Quantified Boolean Formulas (QSAT)

\[ \forall x \exists y \forall z \ (x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z}) \]
Satisfiability of Quantified Boolean Formulas (QSAT)

∀x∃y∀z (x ∨ y) ∧ (x ∨ ¬y) ∧ (z ∨ ¬z)
Satisfiability of Quantified Boolean Formulas (QSAT)

∀x∃y∀z (x ∨ y) ∧ (x ∨ y) ∧ (z ∨ z)

Diagram showing a tree structure with variables and their negations, illustrating the satisfiability process.
∀x∃y∀z (x ∨ y) ∧ (¬x ∨ ¬y) ∧ (z ∨ ¬z)
∀x∃y∀z (x ∨ y) ∧ (¬x ∨ ¬y) ∧ (z ∨ ¬z)
Satisfiability of Quantified Boolean Formulas (QSAT)

$$\forall x \exists y \forall z \ (x \lor y) \land (\overline{x} \lor \overline{y}) \land (z \lor \overline{z})$$
Satisfiability of Quantified Boolean Formulas (QSAT)

∀x∃y∀z (x ∨ y) ∧ (¬x ∨ ¬y) ∧ (z ∨ ¬z)
Satisfiability of Quantified Boolean Formulas (QSAT)

\[ \forall x \exists y \forall z (x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z}) \]
∀x∃y∀z (x ∨ y) ∧ (x̅ ∨ y̅) ∧ (z ∨ z̅)
Satisfiability of Quantified Boolean Formulas (QSAT)

\[ \forall x \exists y \forall z \ (x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z}) \]
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
  - Allows for short proofs both in theory and in practical solving.
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
  - Allows for short proofs both in theory and in practical solving.
- Other approach: QRAT (interference-based!)
  - QRAT is the QBF generalization of RAT.
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
  - Allows for short proofs both in theory and in practical solving.
- Other approach: QRAT (interference-based!)
  - QRAT is the QBF generalization of RAT.
  - Perfect for certifying the correctness of the preprocessing.
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
  - Allows for short proofs both in theory and in practical solving.
- Other approach: QRAT (interference-based!)
  - QRAT is the QBF generalization of RAT.
  - Perfect for certifying the correctness of the preprocessing.
- It was unclear how LQ-Res and QRAT are related.
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
  - Allows for short proofs both in theory and in practical solving.
- Other approach: QRAT (interference-based!)
  - QRAT is the QBF generalization of RAT.
  - Perfect for certifying the correctness of the preprocessing.
- It was unclear how LQ-Res and QRAT are related.
  - If there is a short LQ-Res proof of a QBF, is there also a short QRAT proof?

Short = polynomial with respect to the size of the formula.

Our answer: Yes!
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
  - Allows for short proofs both in theory and in practical solving.
- Other approach: QRAT (interference-based!)
  - QRAT is the QBF generalization of RAT.
  - Perfect for certifying the correctness of the preprocessing.
- It was unclear how LQ-Res and QRAT are related.
  - If there is a short LQ-Res proof of a QBF, is there also a short QRAT proof?
  - Short = polynomial with respect to the size of the formula.
Proof Systems for QBF: LQ-Res vs. QRAT

- Proof systems for QBF are similar to proof systems for SAT.
- There are several resolution systems.
- Most popular system: long-distance resolution (LQ-Res)
  - Allows for short proofs both in theory and in practical solving.
- Other approach: QRAT (interference-based!)
  - QRAT is the QBF generalization of RAT.
  - Perfect for certifying the correctness of the preprocessing.
- It was unclear how LQ-Res and QRAT are related.
  - If there is a short LQ-Res proof of a QBF, is there also a short QRAT proof?
  - Short = polynomial with respect to the size of the formula.
  - Our answer: Yes!
Simulating LQ-Res With QRAT

- How to show that there is a short QRAT proof for every short LQ-Res proof?
Simulating LQ-Res With QRAT

How to show that there is a short QRAT proof for every short LQ-Res proof?

Answer: With a simulation procedure.
Simulating LQ-Res With QRAT

How to show that there is a short QRAT proof for every short LQ-Res proof?

Answer: With a simulation procedure.

- Takes as input an LQ-Res proof and transforms it into a short QRAT proof.
Simulating LQ-Res With QRAT

- **How to show** that there is a short QRAT proof for every short LQ-Res proof?

> **Answer:** With a simulation procedure.

- Takes as input an LQ-Res proof and transforms it into a short QRAT proof.
Simulation Procedure: Results

- Our simulation procedure produces a QRAT proof with at most a quadratic blow-up in size.
Simulation Procedure: Results

- Our simulation procedure produces a QRAT proof with at most a quadratic blow-up in size.
- We implemented the procedure, the tool is called 1d2qrat.
Simulation Procedure: Results

- Our simulation procedure produces a QRAT proof with at most a quadratic blow-up in size.
- We implemented the procedure, the tool is called 1d2qrat.
  - Takes a long-distance proof in the so-called QPR format.
  - Outputs a QRAT proof.
Simulation Procedure: Results

- Our simulation procedure produces a QRAT proof with at most a quadratic blow-up in size.
- We implemented the procedure, the tool is called *ld2qrat*.
  - Takes a long-distance proof in the so-called QPR format.
  - Outputs a QRAT proof.
  - Several optimizations to reduce proof size.
Simulation Procedure: Results

- Our simulation procedure produces a QRAT proof with at most a quadratic blow-up in size.
- We implemented the procedure, the tool is called 1d2qrat.
  - Takes a long-distance proof in the so-called QPR format.
  - Outputs a QRAT proof.
  - Several optimizations to reduce proof size.
  - Resulting proofs are reasonably short.
Simulation Procedure: Results

- Our simulation procedure produces a QRAT proof with at most a quadratic blow-up in size.
- We implemented the procedure, the tool is called 1d2qrat.
  - Takes a long-distance proof in the so-called QPR format.
  - Outputs a QRAT proof.
  - Several optimizations to reduce proof size.
  - Resulting proofs are reasonably short.
- With the tool it is now possible to merge a QRAT proof of a preprocessor with a long-distance proof of a search-based solver.
Simulation Procedure: Results

- Our simulation also gave insight for constructing short QRAT proofs by hand.
Simulation Procedure: Results

- Our simulation also gave insight for constructing short QRAT proofs by hand.
  - Formulas well-known for having short LQ-Res proofs but being hard for other proof systems: Kleine Büning formulas
Simulation Procedure: Results

- Our simulation also gave insight for constructing short QRAT proofs by hand.
  - Formulas well-known for having short LQ-Res proofs but being hard for other proof systems: Kleine Büning formulas
  - We have hand-crafted QRAT proofs of these formulas that are shorter than the LQ-Res proofs.
New Proof-Complexity Landscape for QBF

- Open question: Can QRAT also simulate LQU\(^+\)-Res, a system that is stronger than LQ-Res?
A Little Blocked Literal . . . : Conclusion

- We shed light on the relationship between LQ-Res and QRAT
A Little Blocked Literal . . . : Conclusion

- We shed light on the relationship between LQ-Res and QRAT
  - LQ-Res is a popular system for QBF solving.
We shed light on the relationship between LQ-Res and QRAT

- LQ-Res is a popular system for QBF solving.
- QRAT is the best system for QBF preprocessing.
We shed light on the relationship between LQ-Res and QRAT

- LQ-Res is a popular system for QBF solving.
- QRAT is the best system for QBF preprocessing.

QRAT turns out to be stronger than LQ-Res.
A Little Blocked Literal . . . : Conclusion

- We shed light on the relationship between LQ-Res and QRAT
  - LQ-Res is a popular system for QBF solving.
  - QRAT is the best system for QBF preprocessing.
- QRAT turns out to be stronger than LQ-Res.
- Our new tool allows to transform LQ-Res proofs into QRAT proofs.
But I did not spend my whole time writing papers. (Fortunately.)
Found A Fantastic Collaborator/Supervisor
Found A Fantastic Collaborator/Supervisor/Friend
Had Also a Lot of Fun With His Husband
Lived Together With Magnificent Roommates
Had a Great Time With Lindy and Devon
And Last But Not Least: Met a Cool Group!