Making IP = PSPACE Practical: Efficient Interactive Protocols for BDD Algorithms

Philipp Czerner[1]

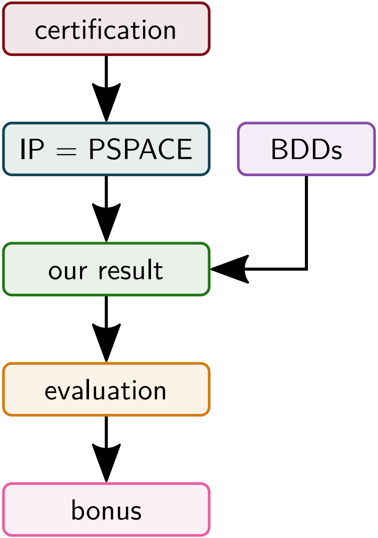
collaboration with

Eszter Couillard[1], Javier Esparza[1], Rupak Majumdar[2]

[1]Department of Informatics, TU Munich
[2]Max Planck Institute for Software Systems

July 30, 2023

Outline



Is this formula satisfiable?

$$
\begin{aligned}
& (x \vee y \vee \neg z) \\
\wedge\ & (\neg x \vee \neg z \vee w) \\
\wedge\ & (\neg z \vee \neg w) \\
\wedge\ & (\neg y \vee z \vee \neg w) \\
\wedge\ & (\neg x \vee z) \\
\wedge\ & (x \vee y \vee \neg w) \\
\wedge\ & (x \vee y \vee z \vee w) \\
\wedge\ & (z \vee w) \\
\wedge\ & (x \vee \neg y \vee \neg z \vee w)
\end{aligned}
$$

Is this formula satisfiable?

$$
\begin{aligned}
& (x \vee y \vee \neg z) \\
\wedge\ & (\neg x \vee \neg z \vee w) \\
\wedge\ & (\neg z \vee \neg w) \\
\wedge\ & (\neg y \vee z \vee \neg w) \\
\wedge\ & (\neg x \vee z) \\
\wedge\ & (x \vee y \vee \neg w) \\
\wedge\ & (x \vee y \vee z \vee w) \\
\wedge\ & (z \vee w) \\
\wedge\ & (x \vee \neg y \vee \neg z \vee w)
\end{aligned}
$$

No...

Is this formula satisfiable?

$$
\begin{aligned}
&(x \vee y \vee \neg z) \\
\wedge\ &(\neg x \vee \neg z \vee w) \\
\wedge\ &(\neg z \vee \neg w) \\
\wedge\ &(\neg y \vee z \vee \neg w) \\
\wedge\ &(\neg x \vee z) \\
\wedge\ &(x \vee y \vee \neg w) \\
\wedge\ &(x \vee y \vee z \vee w) \\
\wedge\ &(z \vee w) \\
\wedge\ &(x \vee \neg y \vee \neg z \vee w)
\end{aligned}
$$

No... at least my SAT-solver says so!

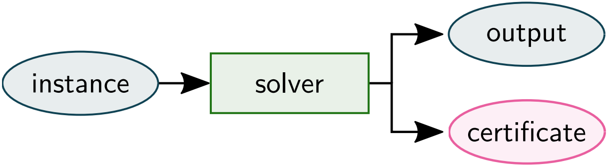
Certification

Certification

- Automated reasoning tools are complicated $\rightarrow$ correctness?

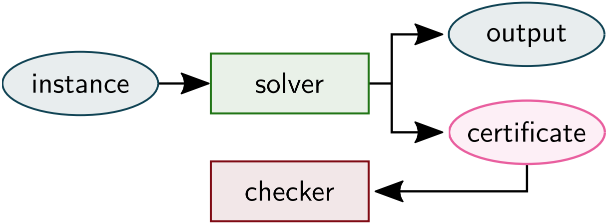
Certification

- Automated reasoning tools are complicated $\rightarrow$ correctness?
- Use certification – each answer comes with a machine-checkable certificate



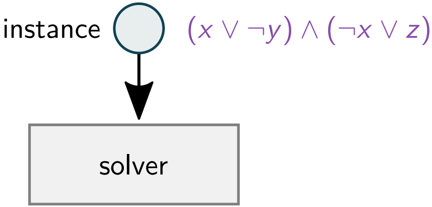
Certification

- Automated reasoning tools are complicated $\rightarrow$ correctness?
- Use certification – each answer comes with a machine-checkable certificate

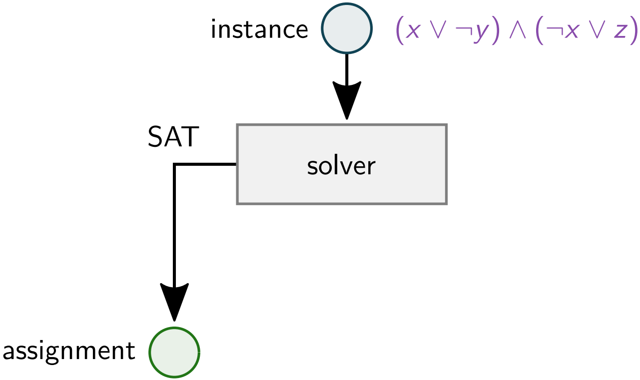


- It suffices to ensure correctness of the certificate checker

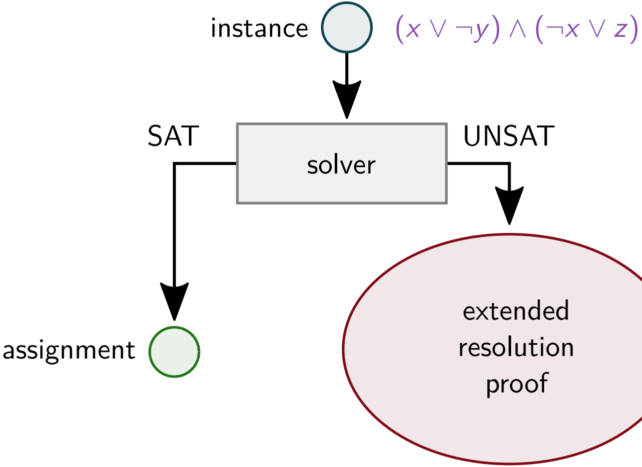
SAT – boolean satisfiability



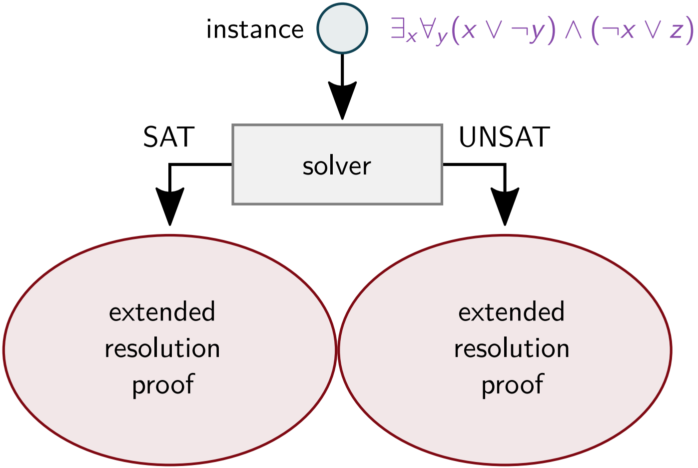
SAT – boolean satisfiability



SAT – boolean satisfiability



QBF – quantified boolean satisfiability



This talk applies to QBF as well.

Extended Resolution Proofs

- Used for (UN)SAT, QBF

Extended Resolution Proofs

- Used for (UN)SAT, QBF
- Essentially a list of clauses, each of which is implied by the previous clauses

Extended Resolution Proofs

- Used for (UN)SAT, QBF
- Essentially a list of clauses, each of which is implied by the previous clauses
- Properties:
  - "efficiently" checkable

Extended Resolution Proofs

- Used for (UN)SAT, QBF
- Essentially a list of clauses, each of which is implied by the previous clauses
- Properties:
  - “efficiently” checkable
  - long (exponential in size of the input)

Extended Resolution Proofs

- Used for (UN)SAT, QBF
- Essentially a list of clauses, each of which is implied by the previous clauses
- Properties:
  - "efficiently" checkable
  - long (exponential in size of the input)
- Certificates can be many terabytes (!) in size
  - e.g. 200 TiB in [Heule,Kullmann,Marek 2016] to solve the boolean Pythagorean Triples problem

The problem

- Huge resolution proofs are difficult to handle

The problem

- Huge resolution proofs are difficult to handle
- In some cases, it can take even longer to verify the proof than to solve the instance (!)

Polynomial-time certification?!

No.

No. However...

Interactive Protocols – Summary

We sacrifice:

Interactive Protocols – Summary

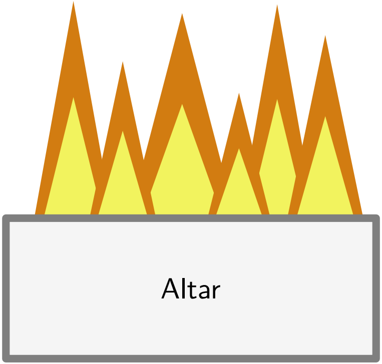
We sacrifice:

- certainty
- non-interactivity

Interactive Protocols – Summary

We sacrifice:

- certainty
- non-interactivity

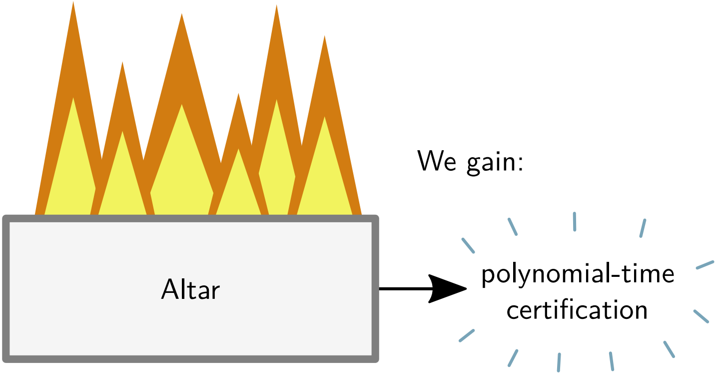


Interactive Protocols – Summary

We sacrifice:

- certainty
- non-interactivity

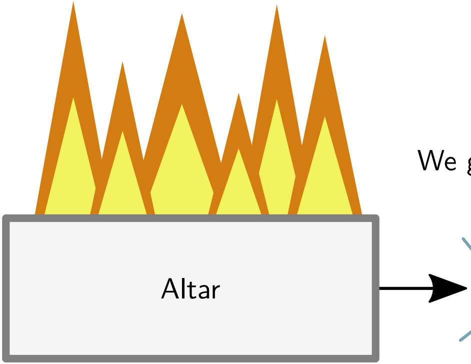
We gain:



Interactive Protocols – Summary

We sacrifice:

- certainty
- non-interactivity



We gain:

polynomial-time certification

A breakthrough

- IP = PSPACE [Lund,Fortnow,Karloff,Nisan 1990], [Shamir 1992]

A breakthrough

- IP $=$ PSPACE [Lund,Fortnow,Karloff,Nisan 1990], [Shamir 1992]
  - famous breakthrough in complexity theory

A breakthrough

- IP = PSPACE [Lund,Fortnow,Karloff,Nisan 1990], [Shamir 1992]
  - famous breakthrough in complexity theory
- demonstrates that efficient certification is possible via interactive protocols, for *any* PSPACE problem

A breakthrough

- IP = PSPACE [Lund,Fortnow,Karloff,Nisan 1990], [Shamir 1992]
  - famous breakthrough in complexity theory
- demonstrates that efficient certification is possible via interactive protocols, for *any* PSPACE problem
  - i.e. SAT, QBF, model counting, ...

Interactive Protocols

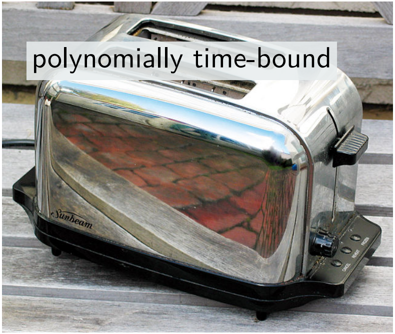
Verifier

Prover

Interactive Protocols

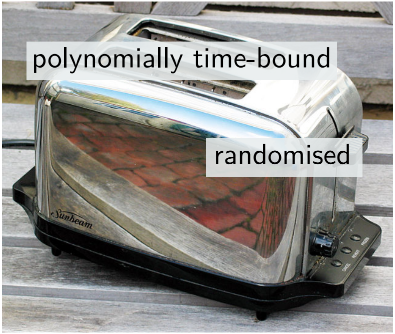
Verifier

Prover



Interactive Protocols

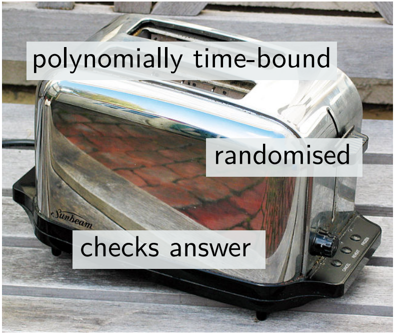
Verifier



Prover

Interactive Protocols

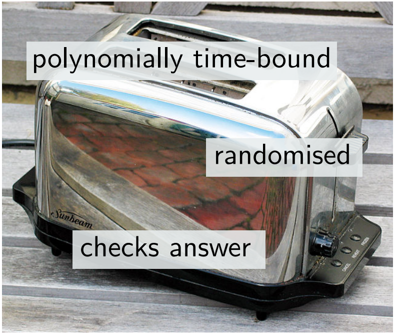
Verifier



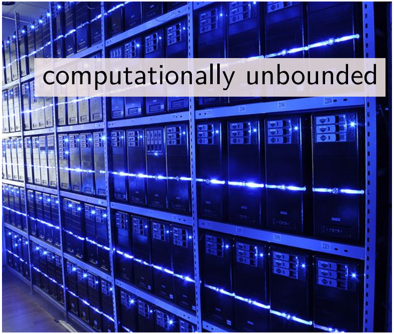
Prover

Interactive Protocols

Verifier

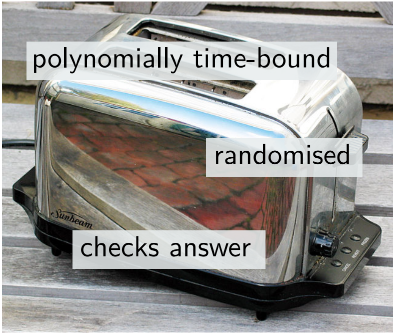


Prover

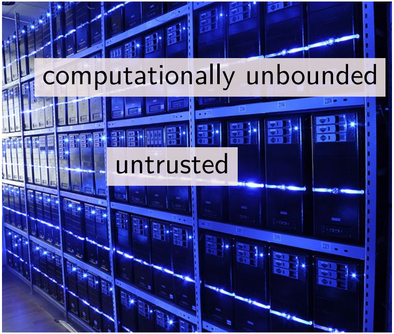


Interactive Protocols

Verifier

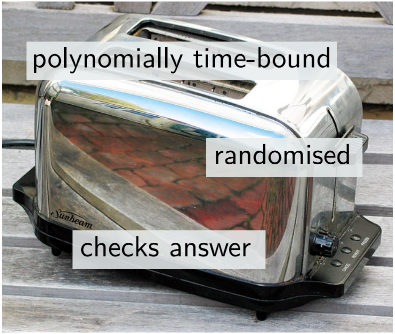


Prover

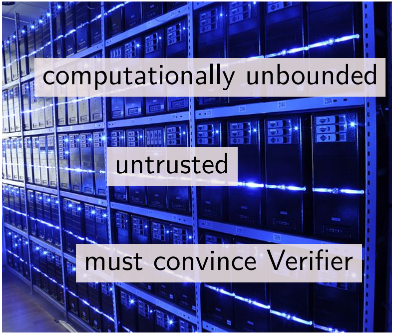


Interactive Protocols

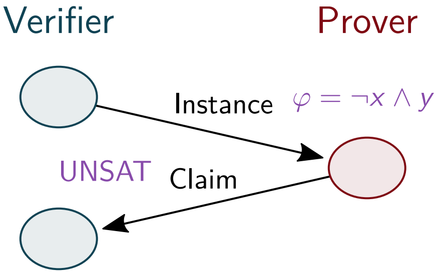
Verifier



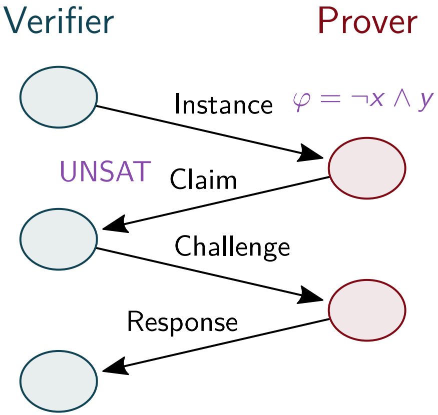
Prover



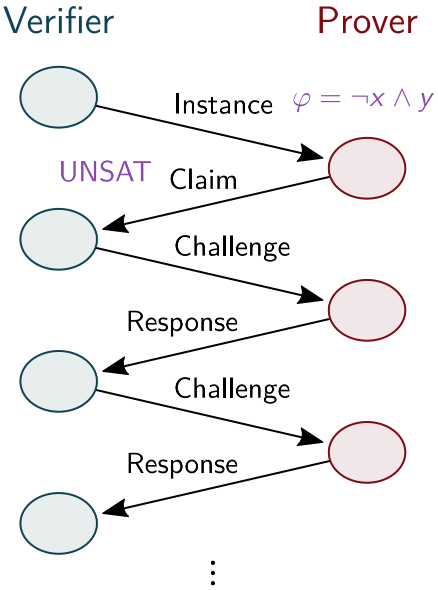
Interactive Protocols



Interactive Protocols



Interactive Protocols



Interactive Protocols

- One-sided error

Interactive Protocols

- One-sided error
  - If the claim is correct, Prover can always convince Verifier, if it follows the protocol

Interactive Protocols

- One-sided error
  - If the claim is correct, Prover can always convince Verifier, if it follows the protocol
  - If the claim is incorrect, with high probability Prover cannot convince Verifier, regardless of Prover's behaviour

Interactive Protocols

- One-sided error
  - If the claim is correct, Prover can always convince Verifier, if it follows the protocol
  - If the claim is incorrect, with high probability Prover cannot convince Verifier, regardless of Prover's behaviour
- "with high probability" means $1 - 2^{-n}$, where n is the size of the input

Interactive Protocols

- One-sided error
  - If the claim is correct, Prover can always convince Verifier, if it follows the protocol
  - If the claim is incorrect, with high probability Prover cannot convince Verifier, regardless of Prover's behaviour
- "with high probability" means $1 - 2^{-n}$, where n is the size of the input $\rightarrow$ negligible in practice

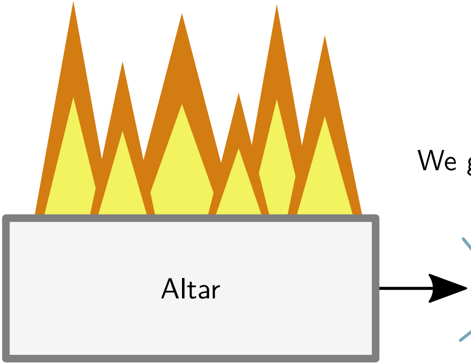
Interactive Protocols

- One-sided error
  - If the claim is correct, Prover can always convince Verifier, if it follows the protocol
  - If the claim is incorrect, with high probability Prover cannot convince Verifier, regardless of Prover's behaviour
- "with high probability" means $1 - 2^{-n}$, where n is the size of the input $\rightarrow$ negligible in practice
- IP is the class of problems that admit such a protocol

Interactive Protocols – Summary

We sacrifice:

- certainty
- non-interactivity



We gain:

polynomial-time certification

Why do we want this?

- Make certification faster

Why do we want this?

- Make certification faster
- Leverage computational asymmetry between parties
  - e.g. a cloud provider offers a QBF-service

Why do we want this?

- Make certification faster
- Leverage computational asymmetry between parties
  - e.g. a cloud provider offers a QBF-service
- Split performance-critical and trusted parts of software

From theory to practice

- Why has IP = PSPACE not already been used for certification?

From theory to practice

- Why has IP = PSPACE not already been used for certification?
- The result constructs an interactive protocol for QBF

From theory to practice

- Why has $\text{IP} = \text{PSPACE}$ not already been used for certification?
- The result constructs an interactive protocol for QBF
- While Verifier is efficient ...

From theory to practice

- Why has IP $=$ PSPACE not already been used for certification?
- The result constructs an interactive protocol for QBF
- While Verifier is efficient ...
- ... Prover is naive; best-case exponential-time (!)

From theory to practice

- Why has IP $=$ PSPACE not already been used for certification?
- The result constructs an interactive protocol for QBF
- While Verifier is efficient ...
- ... Prover is naive; best-case exponential-time (!)
  - iterating over all assignments
  - completely impractical!

From theory to practice

- Why has IP $=$ PSPACE not already been used for certification?
- The result constructs an interactive protocol for QBF
- While Verifier is efficient ...
- ... Prover is naive; best-case exponential-time (!)
  - iterating over all assignments
  - completely impractical!
- In practice, SAT and QBF are solved using advanced data structures and heuristics

From theory to practice

- Why has IP $=$ PSPACE not already been used for certification?
- The result constructs an interactive protocol for QBF
- While Verifier is efficient ...
- ... Prover is naive; best-case exponential-time (!)
  - iterating over all assignments
  - completely impractical!
- In practice, SAT and QBF are solved using advanced data structures and heuristics
  - e.g. DPLL, CDCL for SAT; QCDCL, BDDs for QBF

From theory to practice

- Why has $\mathrm{IP} = \mathrm{PSPACE}$ not already been used for certification?
- The result constructs an interactive protocol for QBF
- While Verifier is efficient ...
- ... Prover is naive; best-case exponential-time (!)
  - iterating over all assignments
  - completely impractical!
- In practice, SAT and QBF are solved using advanced data structures and heuristics
  - e.g. DPLL, CDCL for SAT; QCDCL, BDDs for QBF

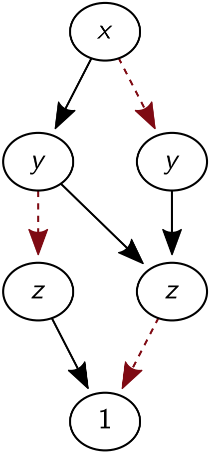
Problem: how do we generate interactive certificates with practical approaches?

BDDs

BDDs

- Reduced Ordered Binary Decision Diagrams (BDDs)

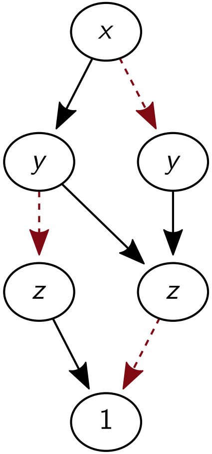
$$x \wedge (y \oplus z)$$
$$\vee \neg x \wedge y \wedge \neg z$$



BDDs

- Reduced Ordered Binary Decision Diagrams (BDDs)
- Unique encoding of boolean functions with efficient boolean operations

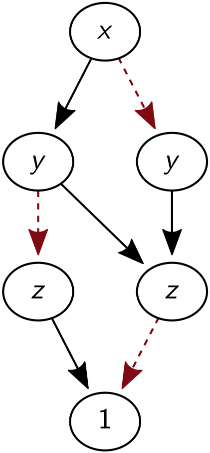
$$x \wedge (y \oplus z) \vee \neg x \wedge y \wedge \neg z$$



BDDs

- Reduced Ordered Binary Decision Diagrams (BDDs)
- Unique encoding of boolean functions with efficient boolean operations
- Are used effectively for QBF, CTL model checking (and many other problems)
  - not as good for SAT, though

$$x \wedge (y \oplus z) \\ \vee \neg x \wedge y \wedge \neg z$$



Our result

Our result

- We give an interactive protocol:

Our result

- We give an interactive protocol:

Theorem. Let φ denote a QBF instance with n variables.

Our result

- We give an interactive protocol:

Theorem. Let φ denote a QBF instance with n variables.

1. Verifier executes in time $\mathcal{O}(n^2|\varphi|)$, with negligible failure probability, and

Our result

- We give an interactive protocol:

Theorem. Let φ denote a QBF instance with n variables.

1. Verifier executes in time $\mathcal{O}(n^2|\varphi|)$, with negligible failure probability, and
2. Prover takes $\mathcal{O}(T)$ time to solve φ and answer Verifier's challenges,

Our result

- We give an interactive protocol:

Theorem. Let φ denote a QBF instance with n variables.

1. Verifier executes in time $\mathcal{O}(n^2|\varphi|)$, with negligible failure probability, and
2. Prover takes $\mathcal{O}(T)$ time to solve φ and answer Verifier's challenges,

where T is the time the BDD algorithm takes to solve φ.

Our result

- We give an interactive protocol:

Theorem. Let φ denote a QBF instance with n variables.

1. Verifier executes in time $\mathcal{O}(n^2|\varphi|) \approx 0$, with negligible failure probability $\approx 10^{-10}$, and
2. Prover takes $\mathcal{O}(T) \approx 3T$ time to solve φ and answer Verifier's challenges,

where T is the time the BDD algorithm takes to solve φ.

(constants in practice)

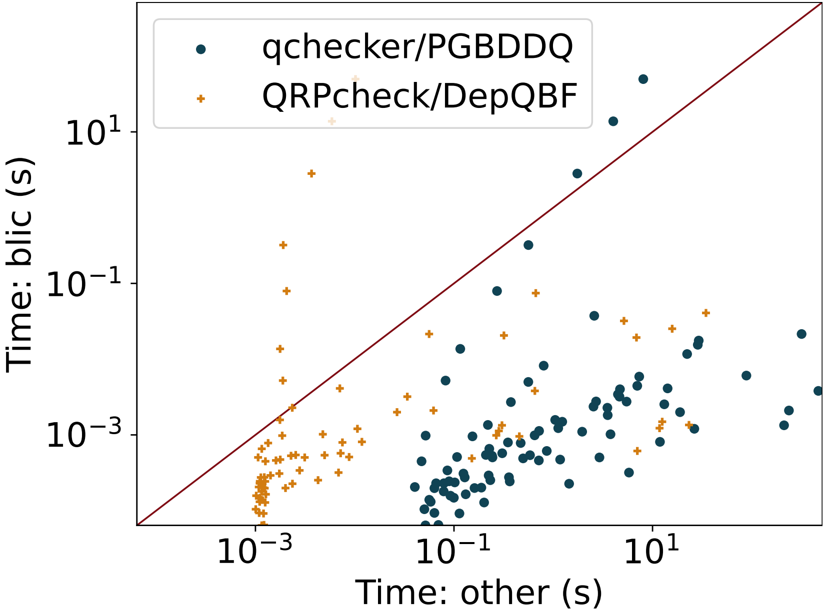
Evaluation

Evaluation

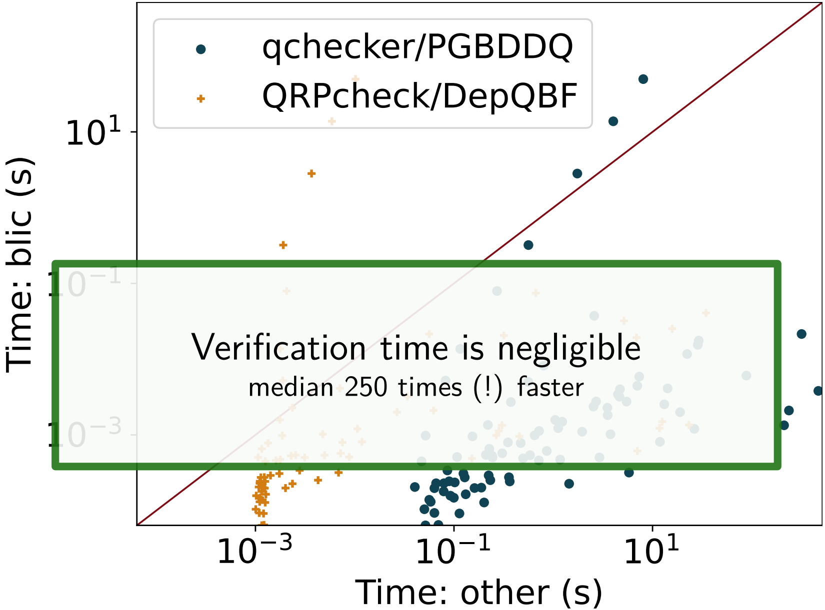
- We implement our approach as blic, a certifying QBF solver
- We compare against state-of-the-art QBF solvers CAQE, DepQBF and PGBDDQ

Evaluation

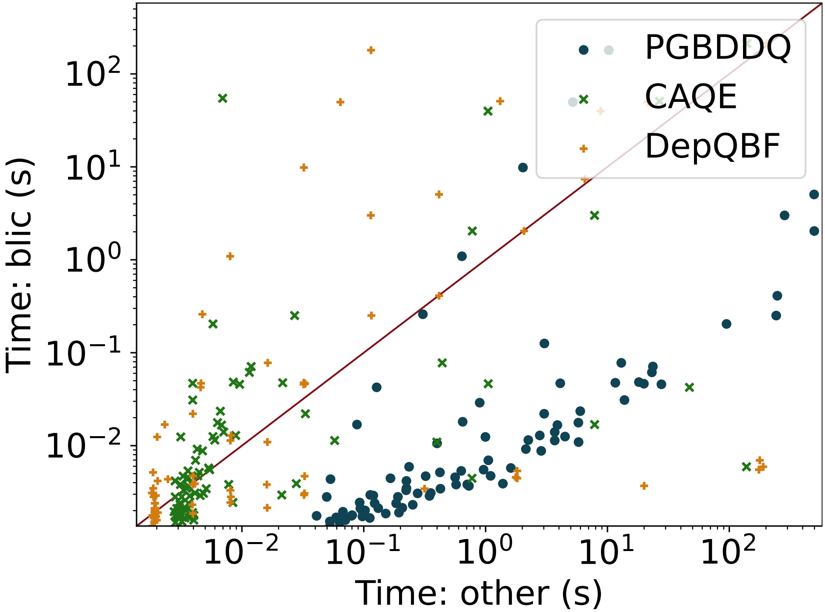
- We implement our approach as blic, a certifying QBF solver
- We compare against state-of-the-art QBF solvers CAQE, DepQBF and PGBDDQ
- DepQBF and PGBDDQ are certifying as well, using extended resolution proofs
- Benchmarks are taken from the crafted instances track of the QBF Evaluation 2022



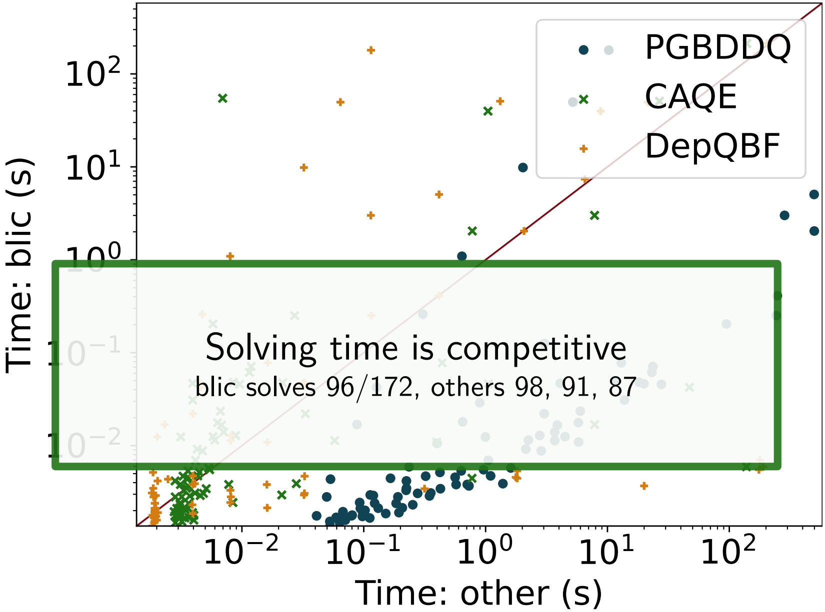
Time to verify certificate (Verifier / external specialised checkers)



Time to verify certificate (Verifier / external specialised checkers)



Time to solve instance and certify solution



Time to solve instance and certify solution

Conclusions

First practical approach with polynomial-time certificate verification!

Conclusions

First practical approach with polynomial-time certificate verification!

- Checking time of the interactive certificate are negligible (median 250 times faster!)

Conclusions

First practical approach with polynomial-time certificate verification!

- Checking time of the interactive certificate are negligible (median 250 times faster!)
- Competitive performance (blic solves 96 of 172 benchmarks, others 98, 91 and 87)

Conclusions

First practical approach with polynomial-time certificate verification!

- Checking time of the interactive certificate are negligible (median 250 times faster!)
- Competitive performance (blic solves 96 of 172 benchmarks, others 98, 91 and 87)
- Generating interactive certificates is low-overhead (factor $\sim$3)

Conclusions

First practical approach with polynomial-time certificate verification!

- Checking time of the interactive certificate are negligible (median 250 times faster!)
- Competitive performance (blic solves 96 of 172 benchmarks, others 98, 91 and 87)
- Generating interactive certificates is low-overhead (factor $\sim$3)
- Error probability is negligible (at most 10^{-10} here)

Conclusions

First practical approach with polynomial-time certificate verification!

- Checking time of the interactive certificate are negligible (median 250 times faster!)
- Competitive performance (blic solves 96 of 172 benchmarks, others 98, 91 and 87)
- Generating interactive certificates is low-overhead (factor $\sim$3)
- Error probability is negligible (at most 10^{-10} here)
- Can be applied to any BDD algorithm

Conclusions

First practical approach with polynomial-time certificate verification!

- Checking time of the interactive certificate are negligible (median 250 times faster!)
- Competitive performance (blic solves 96 of 172 benchmarks, others 98, 91 and 87)
- Generating interactive certificates is low-overhead (factor $\sim$3)
- Error probability is negligible (at most 10^{-10} here)
- Can be applied to any BDD algorithm

Thank you for your attention! Questions?

Bonus Slides

Future work

- Have Prover answer challenges on-the-fly, avoiding memory overhead

Future work

- Have Prover answer challenges on-the-fly, avoiding memory overhead
- Make interactive certificates convincing to third parties, e.g. by using cryptographic hashes

Future work

- Have Prover answer challenges on-the-fly, avoiding memory overhead
- Make interactive certificates convincing to third parties, e.g. by using cryptographic hashes
- Adapt other practical approaches (e.g. CDCL) to generate interactive certificates

Future work

- Have Prover answer challenges on-the-fly, avoiding memory overhead
- Make interactive certificates convincing to third parties, e.g. by using cryptographic hashes
- Adapt other practical approaches (e.g. CDCL) to generate interactive certificates
- Integrate BDD optimisations, e.g. garbage collection, sifting

How does this work?

"The key was with you all along..."

- We can re-use both the existing interactive protocol SUMCHECK and BDD-algorithms from the literature
- Only minor adjustments are needed

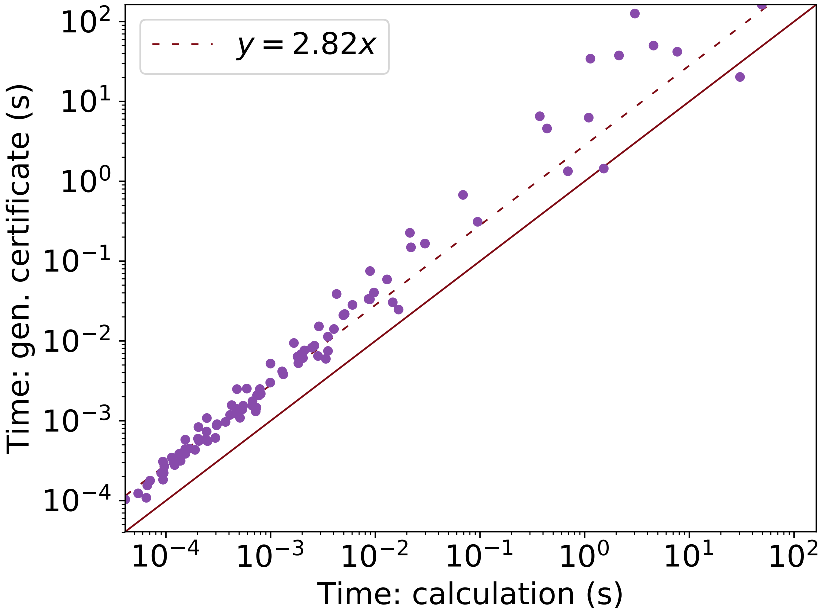
How does this work?

"The key was with you all along..."

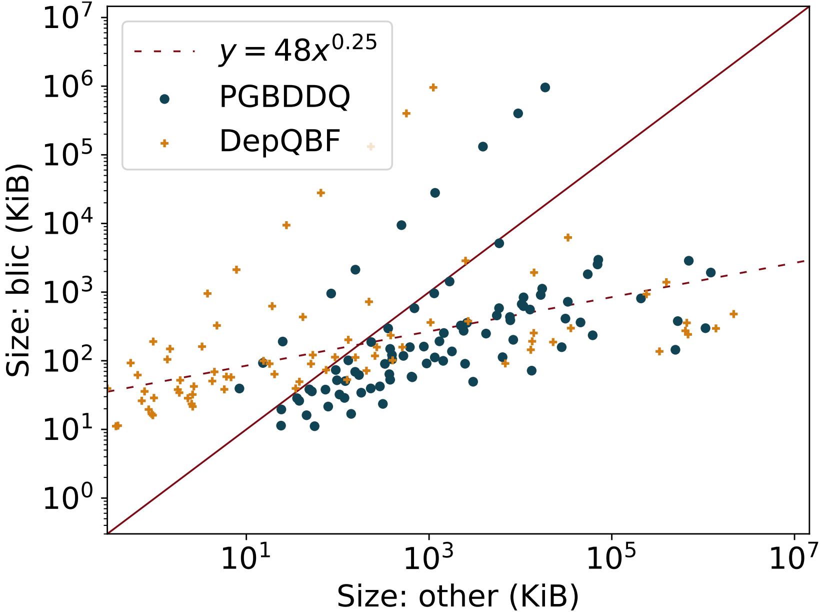
- We can re-use both the existing interactive protocol SUMCHECK and BDD-algorithms from the literature
- Only minor adjustments are needed

"... and the real treasure was the friends you made along the way."

- BDDs uniquely represent binary multilinear polynomials used in SUMCHECK
- Intermediate results from the BDD-computations encode the answers to Verifier's challenges



y = 2.82x
Time: gen. certificate (s)
Time: calculation (s)
10^2
10^1
10^0
10^-1
10^-2
10^-3
10^-4



$y = 48x^{0.25}$
PGBDDQ
DepQBF
Size: blic (KiB)
Size: other (KiB)
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7