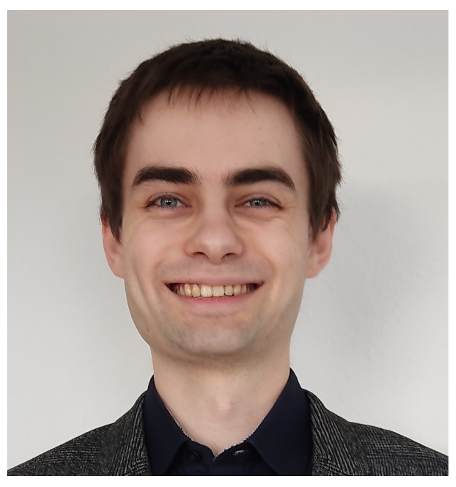
Making IP = PSPACE Practical with BDD Algorithms



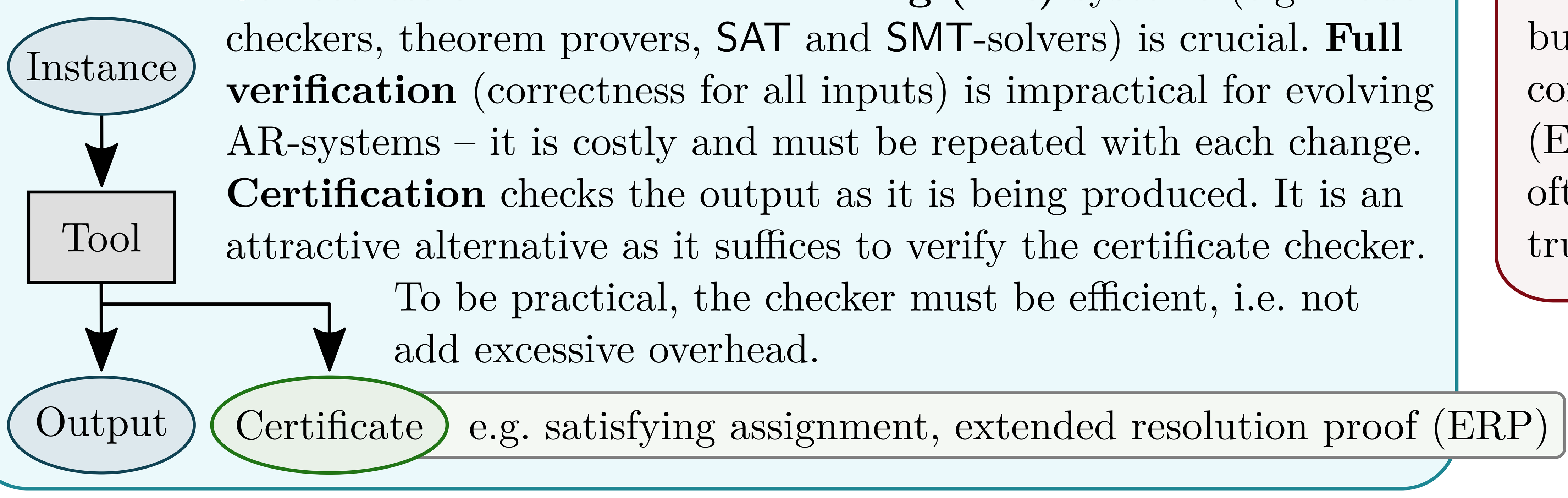
Philipp Czerner

czerner@in.tum.de

Supervisors: Javier Esparza & Helmut Seidl

Collaborators: Eszter Couillard & Javier Esparza (ConVeY) & Rupak Majumdar

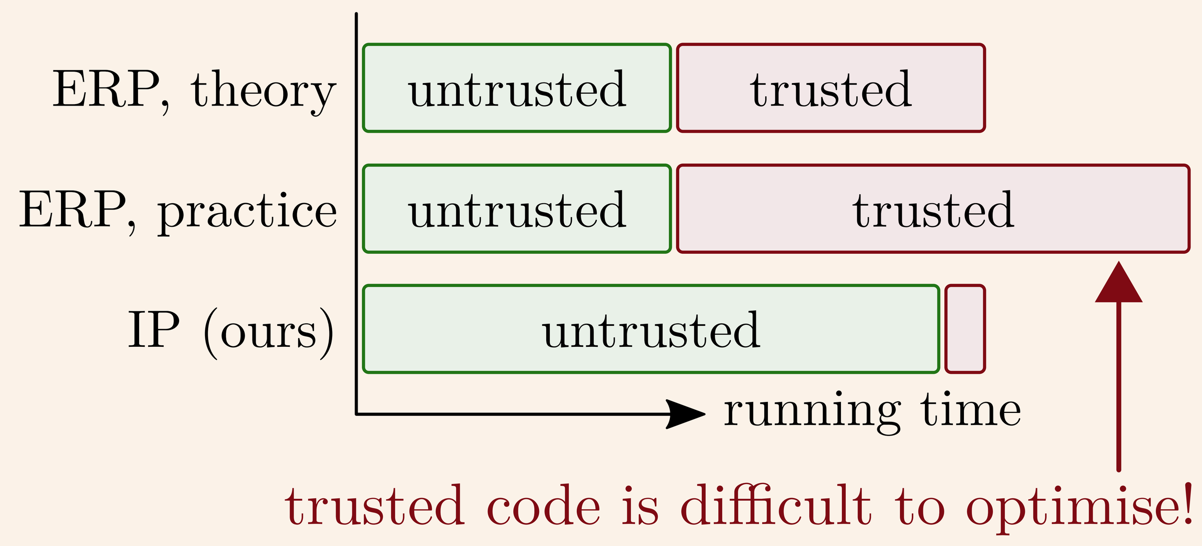
Evolving AR-Systems



Correctness of **automated reasoning (AR)** systems (e.g. model checkers, theorem provers, SAT and SMT-solvers) is crucial. **Full verification** (correctness for all inputs) is impractical for evolving AR-systems – it is costly and must be repeated with each change. **Certification** checks the output as it is being produced. It is an attractive alternative as it suffices to verify the certificate checker. To be practical, the checker must be efficient, i.e. not add excessive overhead.

Efficient Certification

Efficient non-interactive certificates exist for SAT, but not for UNSAT or PSPACE problems, which are common in AR. Instead, extended resolution proofs (ERPs) are used. In practice, certificate validation is often too expensive, as it must be performed by trusted code that cannot be optimised well.

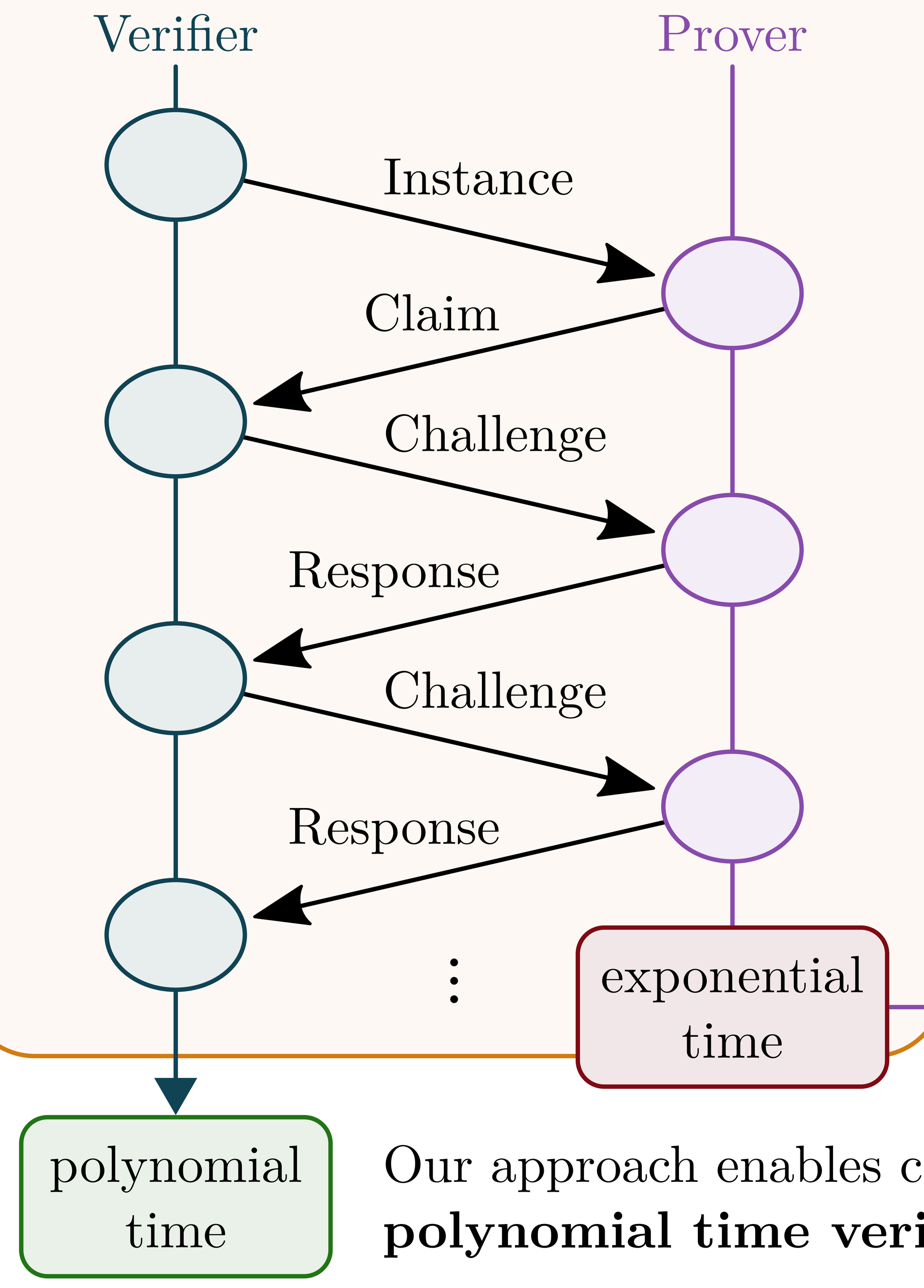


Goal: Efficient Certification for PSPACE

The famous IP = PSPACE breakthrough in complexity theory [1,2] proves existence of efficient certification through interactive protocols (IPs) for any PSPACE problem. This has not been used in automated reasoning – until now. We combine it with binary decision diagrams, which are successfully used in practice, to get the first practical certification method for PSPACE with polynomial-time verification.

Interactive Protocols

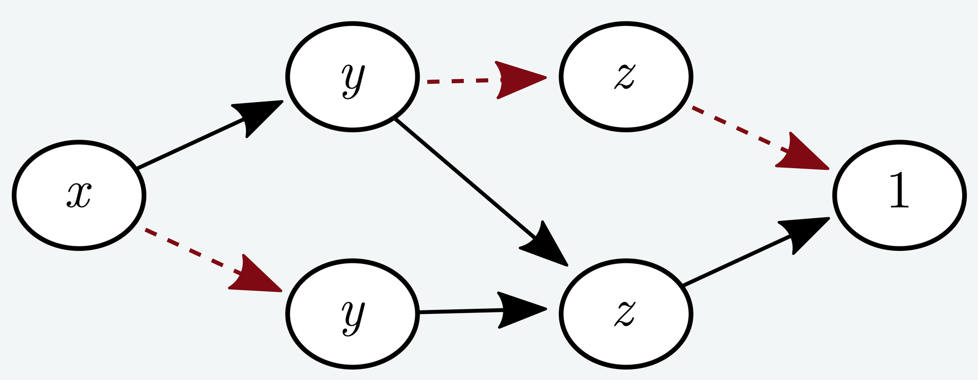
Polynomial Verifier checks claims of unbounded, but untrusted, Prover



Our approach enables certification with **polynomial time verification cost**

BDDs Binary Decision Diagrams

Uniquely represent arbitrary boolean functions; efficient boolean operations.



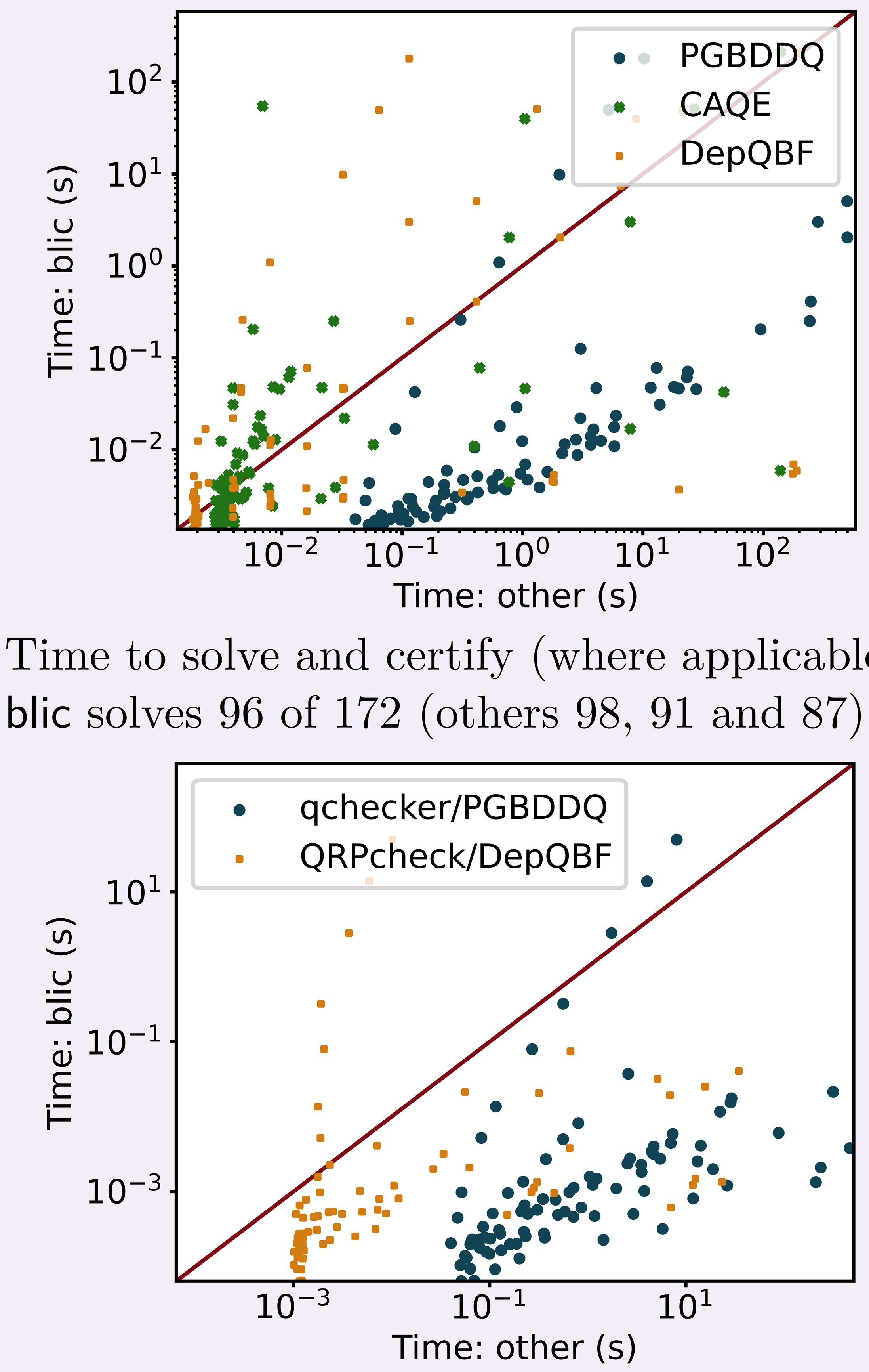
$x \wedge (y \oplus z) \vee \neg x \wedge y \wedge \neg z$

While exponential in the worst-case, in practice BDDs are often effective. They are used for CTL model checking, circuit equivalence, and many more.

We show: any BDD-based algorithm yields a Prover implementation with **constant-factor overhead!** (compared with the BDD algorithm)

Evaluation

We implement our approach as blic [3], a new certifying QBF solver, and compare against state-of-the-art certifying (PGBDDQ, DepQBF) and non-certifying (CAQE) solvers [4,5,6], on the crafted instances track of QBF Eval 2022.



Time to solve and certify (where applicable). blic solves 96 of 172 (others 98, 91 and 87)

Time to verify certificate. Excepting the pathological LONSING family, blic is always faster, with median factor of 385 (!).

Directions for Future Work

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

[1] Lund, Fortnow, Karloff, Nisan, 1990 [2] Shamir, 1992 [3] https://gitlab.lrz.de/i7/blic [4] https://github.com/rebryant/pgbdd
[5] https://lonsing.github.io/depqbf/ [6] https://www.react.uni-saarland.de/tools/caqe