Satisfiability Modulo Finite Fields with applications to Zero-Knowledge Proof Compilers

Alex Ozdemir, Gereon Kremer, Riad S. Wahby, Cesare Tinelli, Fraser Brown, Clark Barrett

“Bounded Verification for Finite-Field Blasting” (CAV’23)

“Satisfiability Modulo Finite Fields” (CAV’23)

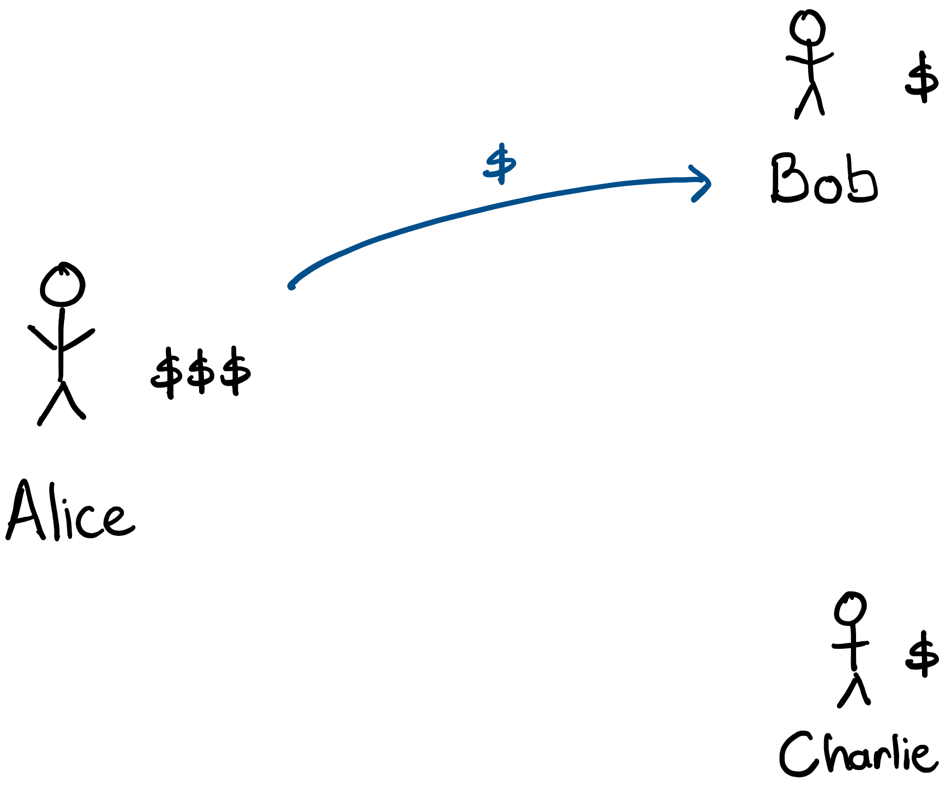
Stanford | Center for Automated Reasoning

Stanford University

Carnegie Mellon University

IOWA

Private cryptocurrencies



integrity:

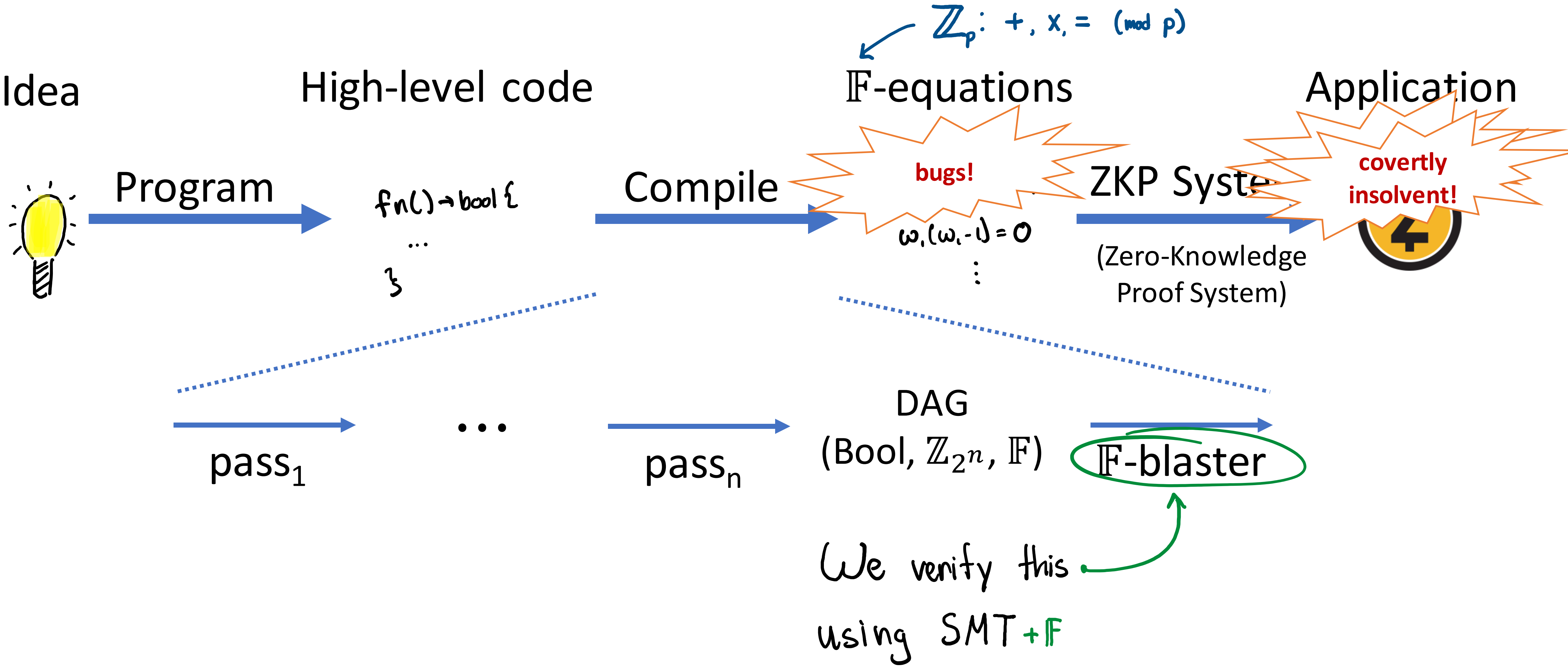
- authorization
- conservation of $

privacy:

- transactions are hidden

Possible with: Zero-Knowledge Proofs (ZKPs)

How a private cryptocurrency is built



Today’s Talk

I. Correctness for a ZKP compiler

II. A framework for verifiable $\mathbb{F}$-blasting

- With automatic (bounded) verification

III. Satisfiability modulo finite fields

IV. Case study: bugs in CirC’s $\mathbb{F}$-blaster

Part I: Correct ZKP compilers

What is a ZKP System?

class ($\mathbb{Z}_p$ equations)

predicate public instance

$\phi(x, w) \in \mathcal{C}$

secret witness

$\exists w, \phi(x, w)$?

$\underline{\text{Prove}_{\mathcal{C}}(\phi, x, w)}$ $\underline{\text{Verify}_{\mathcal{C}}(\phi, x)}$

$\{\perp, \top\}$

for payments

Properties:

- *complete + sound**: valid $w \Leftrightarrow V$ accepts ← integrity
- *zero-knowledge**: hides w ← privacy

* usually under computational assumptions

What is a correct ZKP compiler from $\mathcal{C} \to \mathcal{C}'$?

high-level

ZKPS-compatible

Compile($\phi(x, w) \in \mathcal{C}$) outputs:

- $\phi'(x', w') \in \mathcal{C}'$
- $E_x(x) \to x'$
- $E_w(x, w) \to w'$

Our definition: correctness as **equisatisfiability**:

<u>demonstrable completeness</u>

$$\forall x, \forall w,$$

$$\phi(x, w) \Rightarrow \phi'(E_x(x), E_w(x, w))$$

<u>demonstrable soundness</u>

$$\exists \textit{ eff. } I(x, w') \to w, \ \forall x, \ \forall w',$$

$$\phi'(E_x(x), w') \Rightarrow \phi(x, I(x, w'))$$

Properties of our definition

Theorem 1 (ZKPS Generalization):

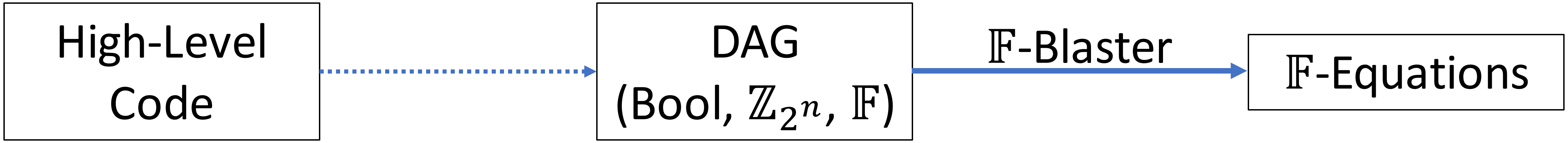
Correct ZKP compiler from $\mathcal{C} \to \mathcal{C}'$ + Secure ZKPS for $\mathcal{C}'$ = Secure ZKPS for $\mathcal{C}$

Theorem 2 (Compiler Composition):

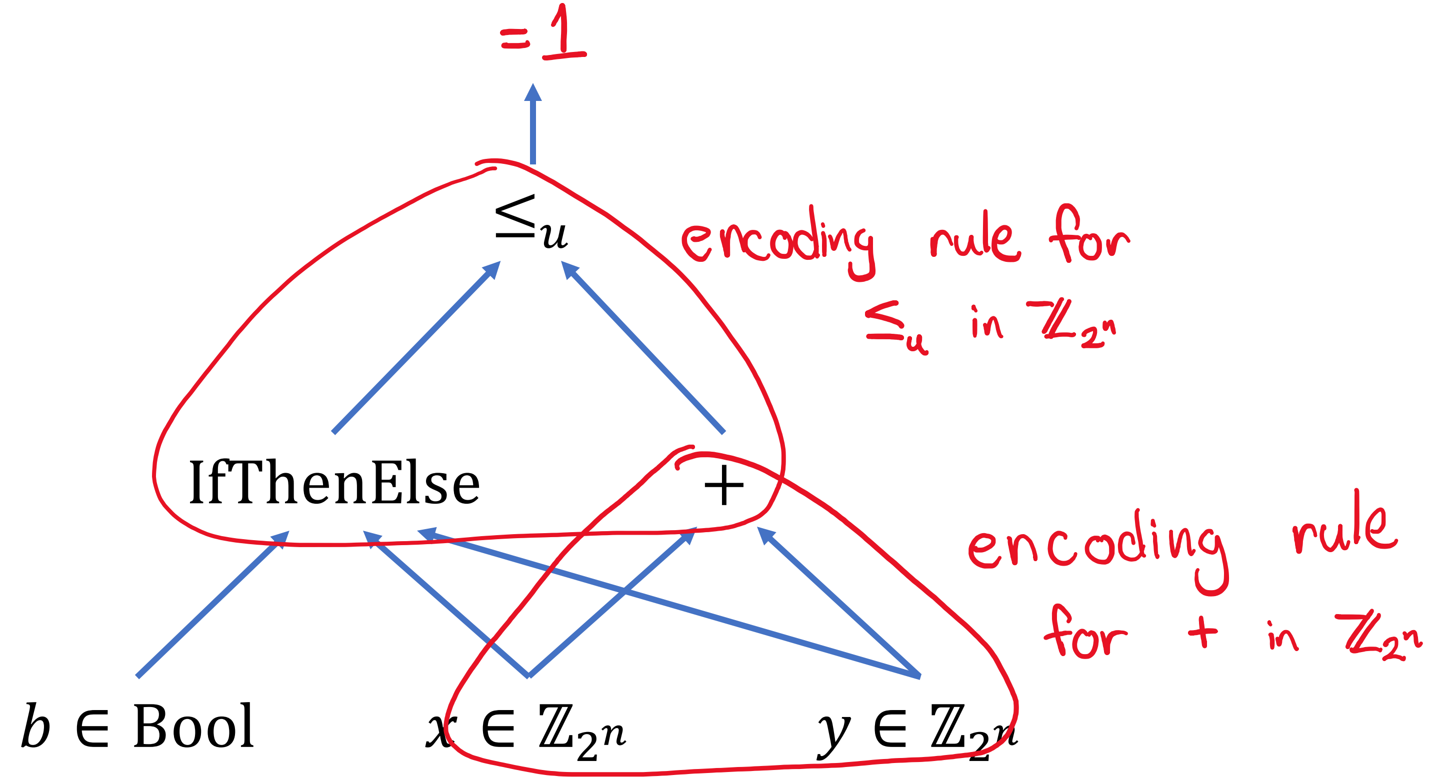
Correct ZKP compiler from $\mathcal{C} \to \mathcal{C}'$ + Correct ZKP compiler from $\mathcal{C}' \to \mathcal{C}''$ = Correct ZKP compiler from $\mathcal{C} \to \mathcal{C}''$

Part II: Verifiable $\mathbb{F}$-blasting

The architecture of a Finite-Field blaster




```
(b ? x : y)
 <= (x + y)
```



encodings: $b' \in \{0,1\} \subset \mathbb{F}$ $x' \in \{0,\dots,2^n-1\} \subset \mathbb{F}$ $y'_1,\dots,y'_n \in \{0,1\} \subset \mathbb{F}$

Example rule: n-ary Boolean AND (complex)

- Encode a Boolean function:
  - $y = x_1 \wedge \cdots \wedge x_n$
- Assume:
  - $p \gg n$
  - Field variables x_i':
    - $x_i' = \text{IfThenElse}(x_i, 1,0)$
- Task:
  - emit $\mathbb{F}$-equations, new variables
  - Ensure $y' = \text{IfThenElse}(y, 1,0)$

what does this mean?

Naively, binary $\wedge$:

- $t_0' = 1$
- $t_{i+1}' = x_i' t_i'$
- $y' = t_n'$

simple, but inefficient

Fewer non-linear $\times$s:

- Idea: $y' = \text{AreEqual}(n, \sum_i x_i')$
- Implemented as:
  1) $(n - \sum_i x_i')z' = 1 - y'$
  2) $(n - \sum_i x_i')y' = 0$

!?

complete?
sound?

A framework for a verifiable $\mathbb{F}$-blaster

Encoding Scheme

Original Term	Encoding
Boolean x	$x' \in \{0,1\} \in \mathbb{F}$
bit-vector $x \in \mathbb{Z}_{2^b}$	$x' \in \{0, \dots, 2^b - 1\} \in \mathbb{F}$
bit-vector $x \in \mathbb{Z}_{2^b}$	$x'_1, \dots, x'_b \in \{0,1\} \in \mathbb{F}$
field elem. $x \in \mathbb{F}$	$x' \in \mathbb{F}$

+

Encoding Rules

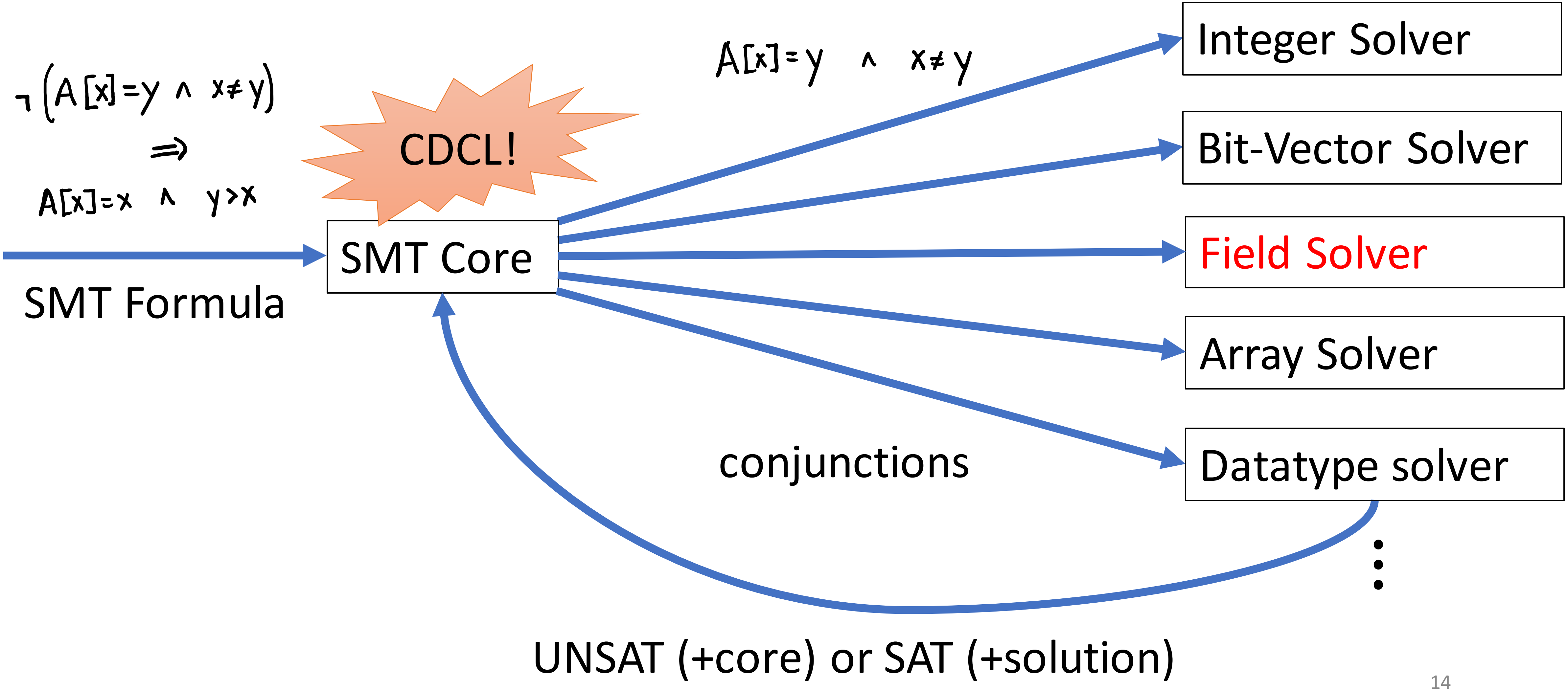
bv.add	bool.and	*conversions*
bv.sub	bool.or	*new vars*
bv.lshl	...	*priv. vars*
bv.shr	ff.add	*assertions*
bv.ashl	ff.div	...
...	...	

$\boxed{\text{Theorem 3: VCs} \Rightarrow \text{correct } \mathbb{F}\text{-blaster}}$

- DSL for
  - encoding schemes
  - encoding rules
- Operational semantics
  - $\mathbb{F}$-blaster
- VCs for each rule:
  - "assumes validly encoded inputs"
  - "must validly encode outputs"
  - Idea: verify with SMT+$\mathbb{F}$

Part III: Satisfiability Modulo Finite Fields

Satisfiability Modulo Theories



Prime-Order Finite Fields

- Written $\mathbb{Z}_p$, $\mathbb{F}_p$, or $\mathbb{F}$
- A set of integers: $\{0, \ldots, p-1\}$
- Operations:
  - $+$ (mod p)
  - $\times$ (mod p)
  - $=$
- Our theory: fixed p

$X = X+1$
$\vee$
$Y^2 = Y$

SAT : $X=0, Y=0$

$X = X+1$

UNSAT

$\mathbb{F}$ theory solver, preprocessing:

Step 1: homogenize (convert disequalities to equalities)

$$\bigwedge_i a_i = b_i \land \bigwedge_i c_i \neq d_i$$

$$\bigwedge_i a_i - b_i = 0$$

$$\bigwedge_i c_i \neq d_i \longrightarrow \bigwedge_i (c_i - d_i)w - 1 = 0$$

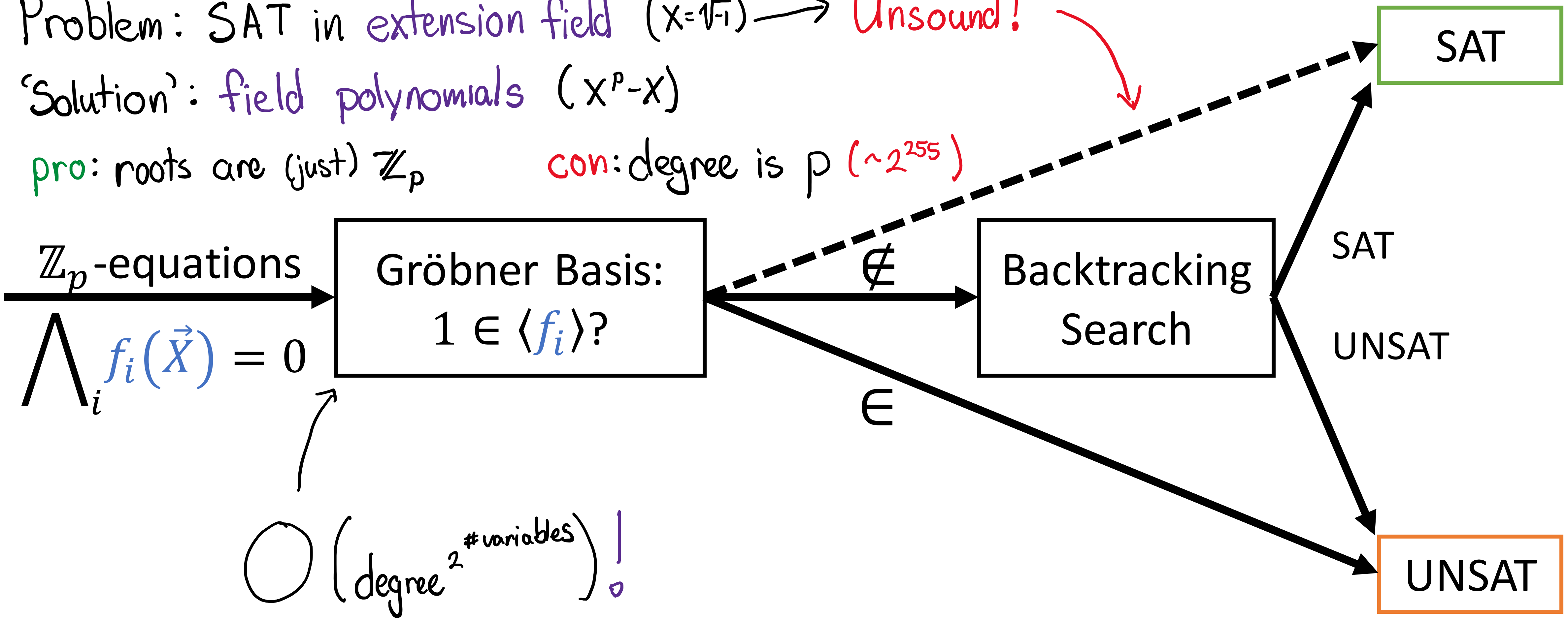
fresh

$\mathbb{F}$ theory solver, main sketch:

Problem: SAT in extension field $(x=\sqrt{-1}) \rightarrow$ Unsound!

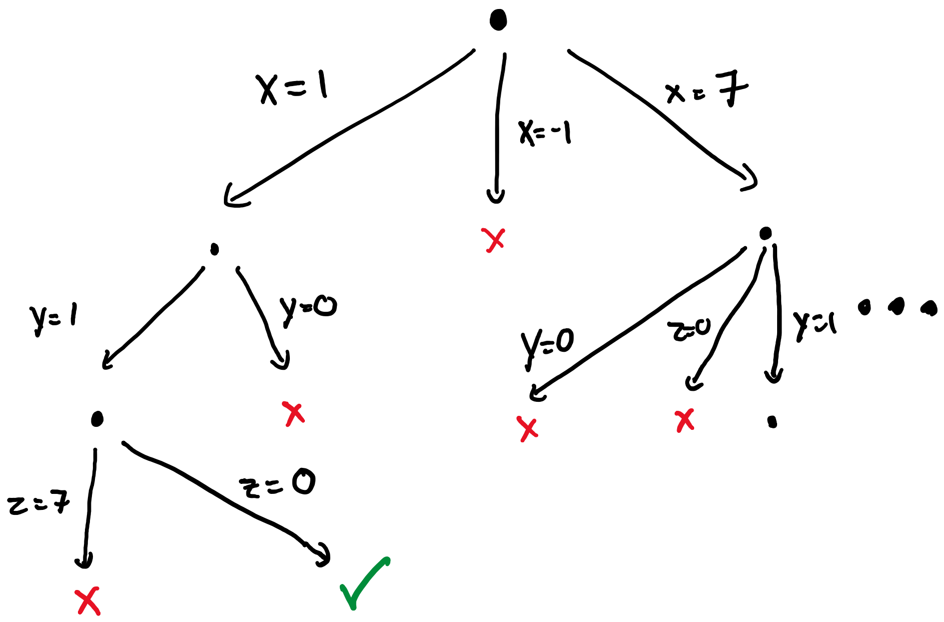
'Solution': field polynomials $(x^p - x)$

pro: roots are (just) $\mathbb{Z}_p$ con: degree is p $(\sim 2^{255})$



$O\left(\text{degree}^{2^{\#\text{variables}}}\right)$!

The Backtracking Search



Worst-case

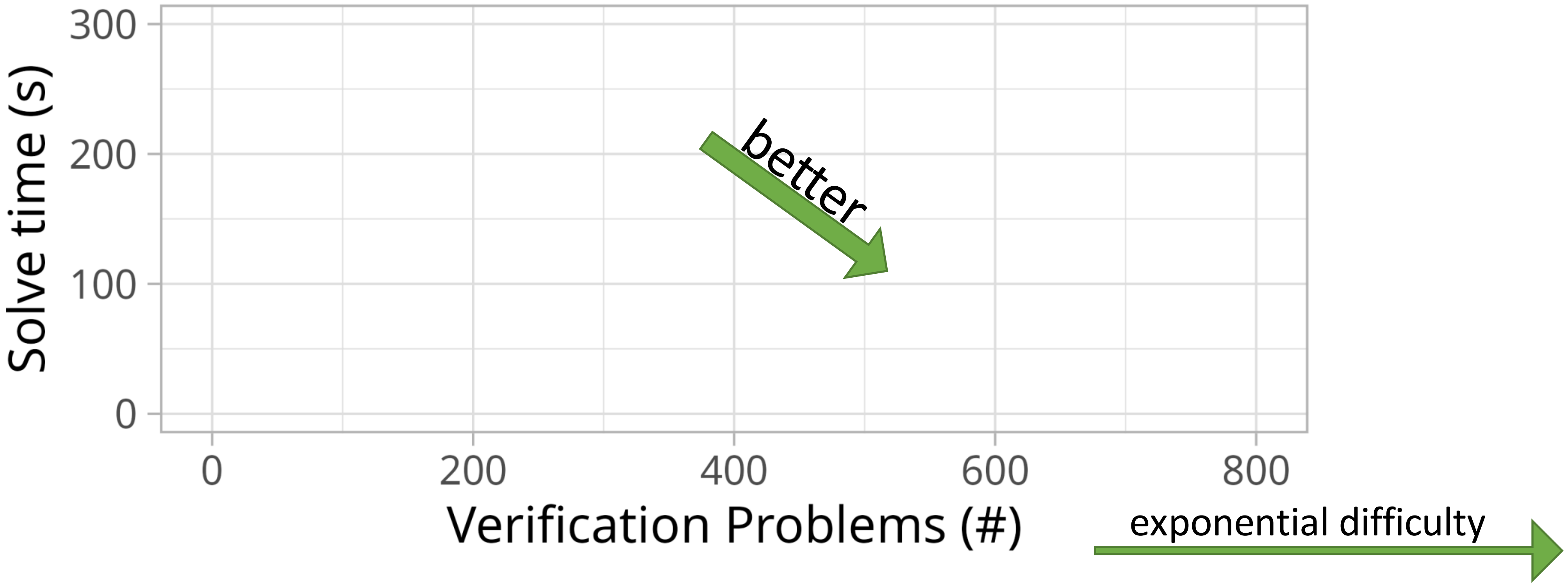
$O_{n,d}(p)$

At each node:

- Compute a GB B
- Branch in one of three ways:
  - If ∃ univariate $p(X) \in B$:
    - factor and branch on X values
  - If $\dim \mathcal{V}(\langle B \rangle) = 0$:
    - compute a *minimal* $p(X) \in \langle B \rangle$, factor, and branch on X values
  - Otherwise, exhaustion:
    - $X = 1, Y = 1, X = 2, Y = 2, \ldots$

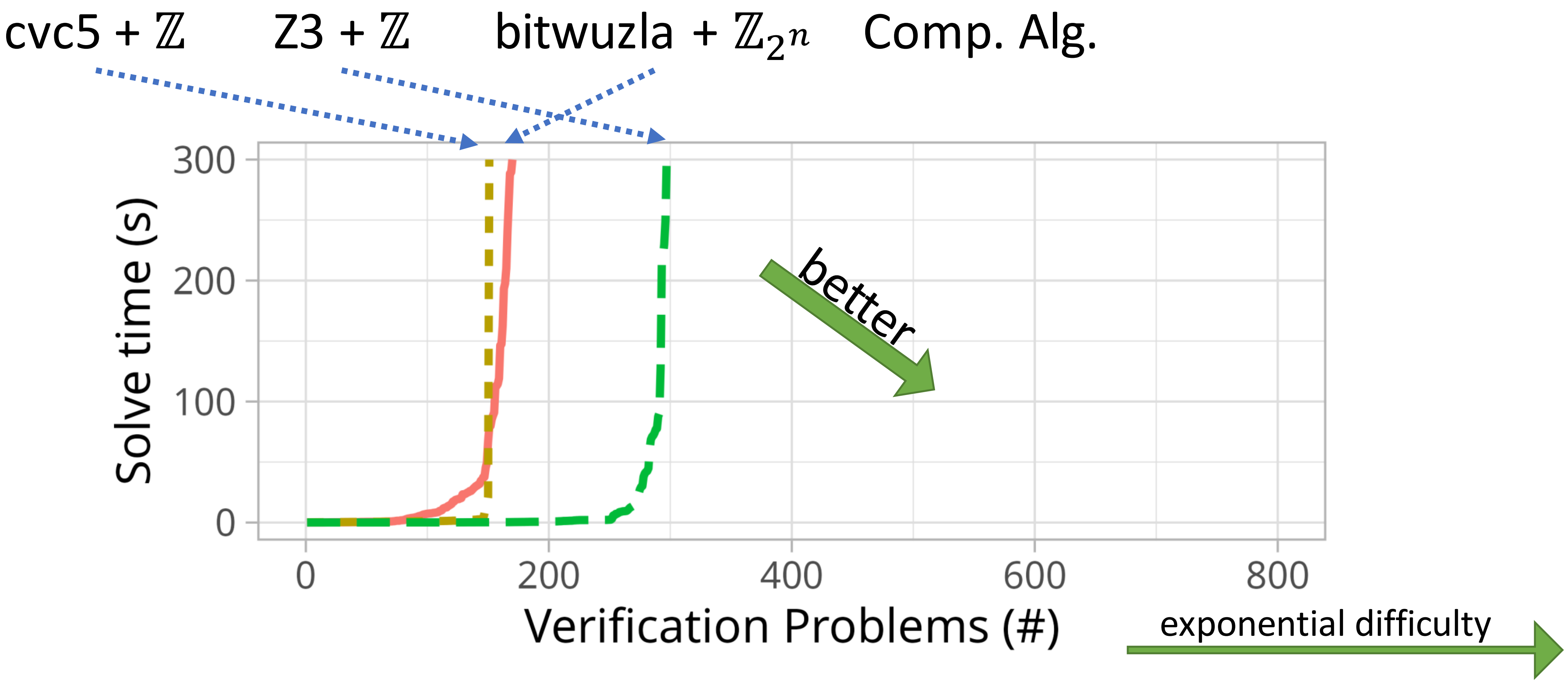
SMT + $\mathbb{Z}_p$ is the best choice for ZK verification

cvc5 + $\mathbb{Z}$ Z3 + $\mathbb{Z}$ bitwuzla + $\mathbb{Z}_{2^n}$



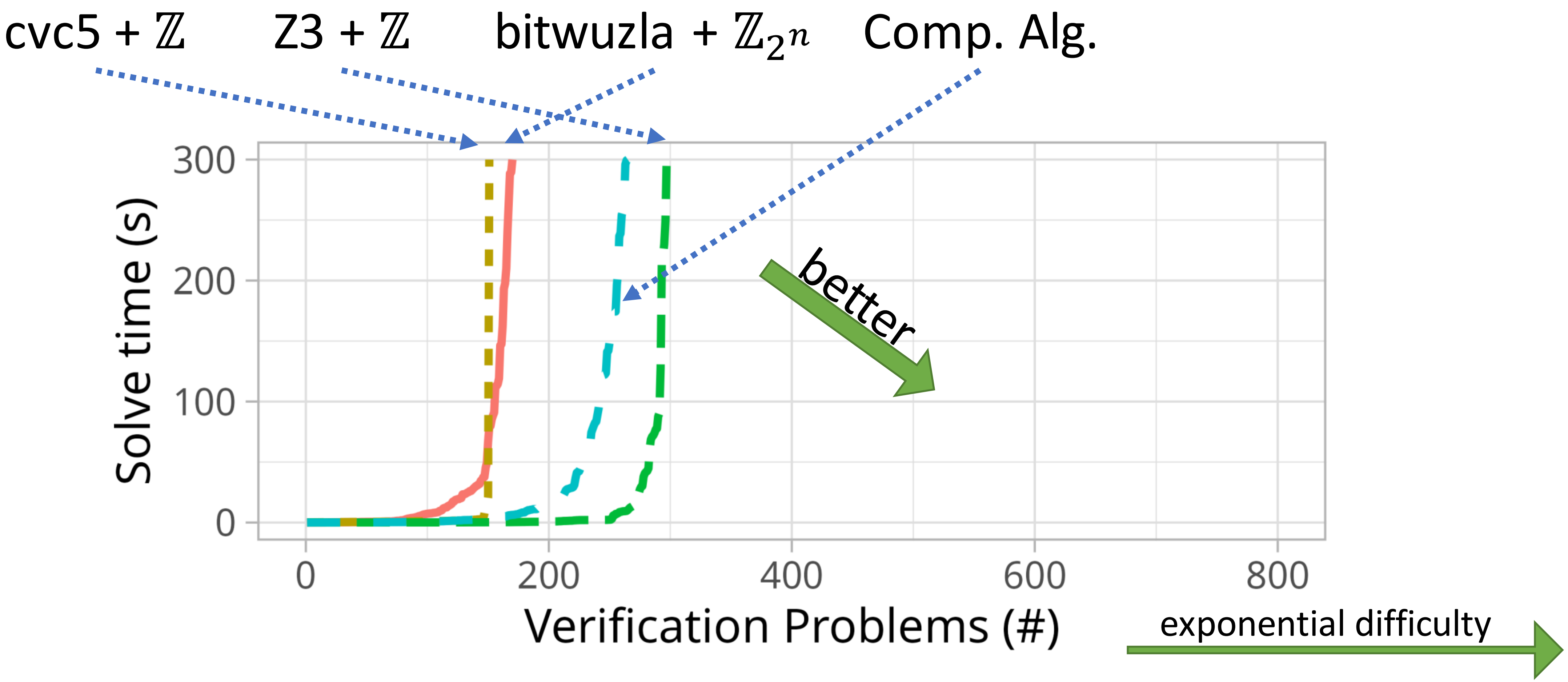
solvers: bitwuzla-1.0pre (QF_BV), z3-4.11.2 (QF_NIA), cvc5 nightly 17 Dec ’22 (QF_NIA,QF_FF), CoCoALib 0.998

SMT + $\mathbb{Z}_p$ is the best choice for ZK verification



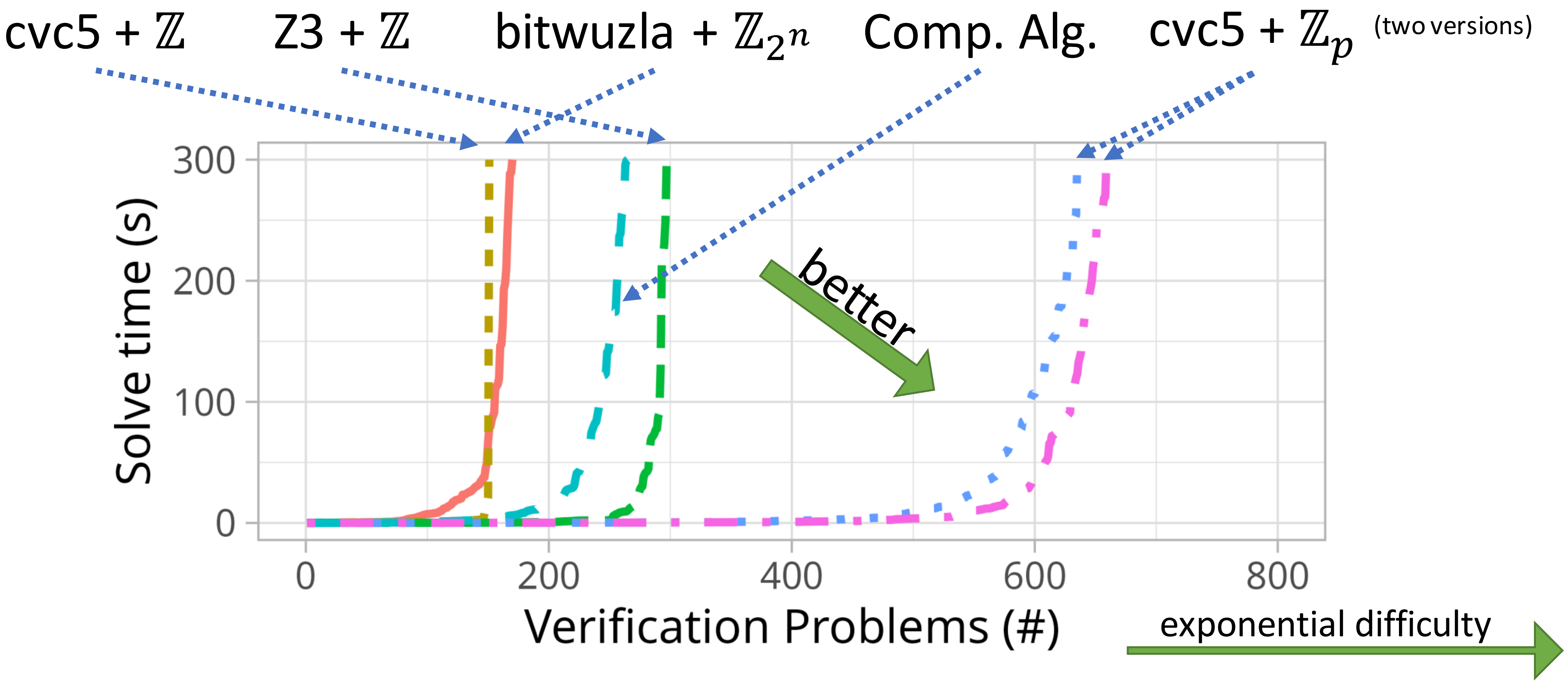
solvers: bitwuzla-1.0pre (QF_BV), z3-4.11.2 (QF_NIA), cvc5 nightly 17 Dec ’22 (QF_NIA,QF_FF), CoCoALib 0.998

SMT + $\mathbb{Z}_p$ is the best choice for ZK verification



solvers: bitwuzla-1.0pre (QF_BV), z3-4.11.2 (QF_NIA), cvc5 nightly 17 Dec ’22 (QF_NIA,QF_FF), CoCoALib 0.998

SMT + $\mathbb{Z}_p$ is the best choice for ZK verification

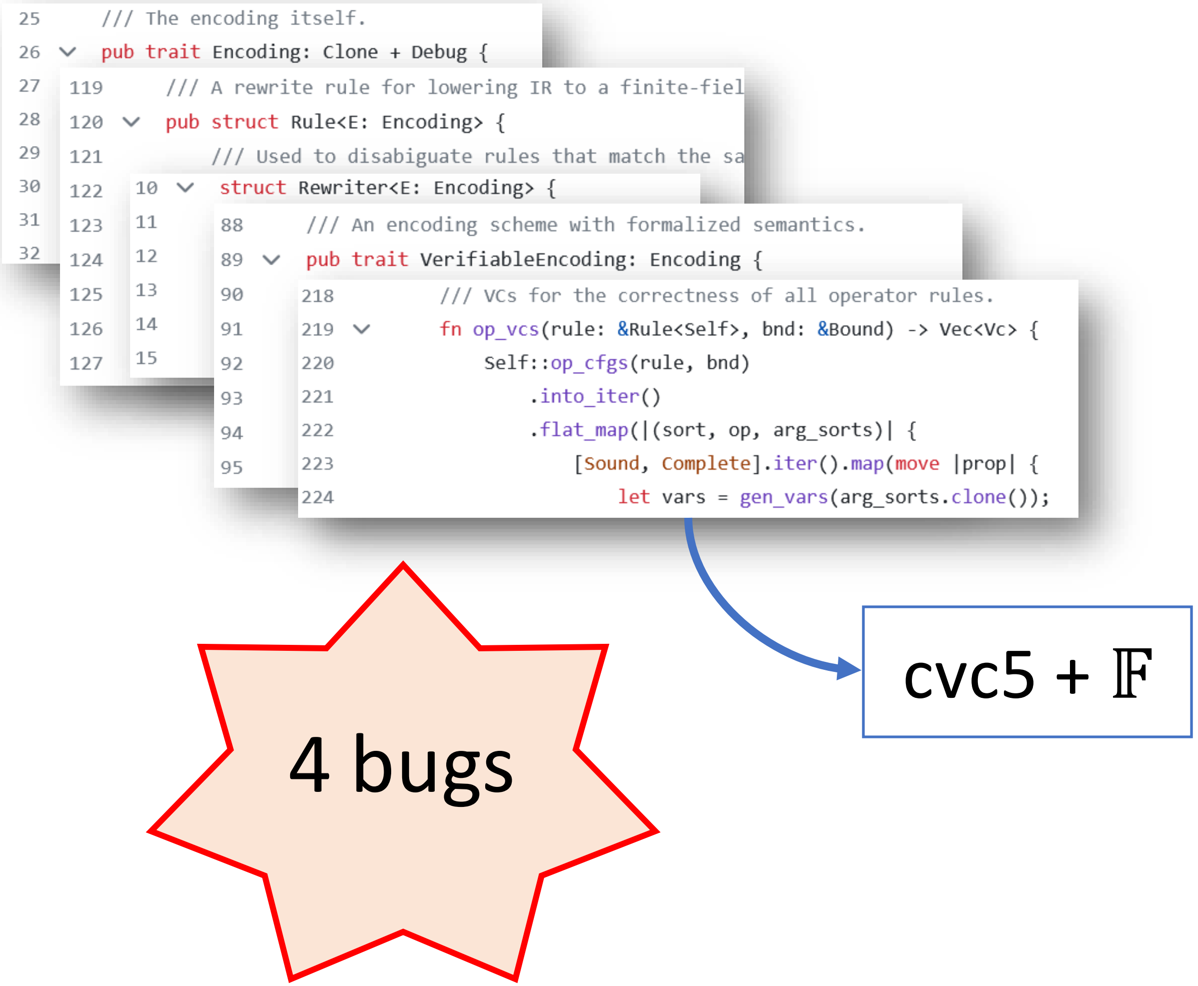


solvers: bitwuzla-1.0pre (QF_BV), z3-4.11.2 (QF_NIA), cvc5 nightly 17 Dec ’22 (QF_NIA,QF_FF), CoCoALib 0.998

Part IV: Case Study

Verified 𝔽-blasting in CirC

- CirC [IEEE S&P'22]
  - compiler infrastructure
    - to ZKPSs, MPCs, SMT, …
  - state-of-the art for ZKPS
  - 𝔽-blaster:
    - 4 encoding schemes
    - 80+ encoding rules
- Implementation:
  - 𝔽-blaster DSL
  - operational semantics
  - VC generation
  - rules from original 𝔽-blaster



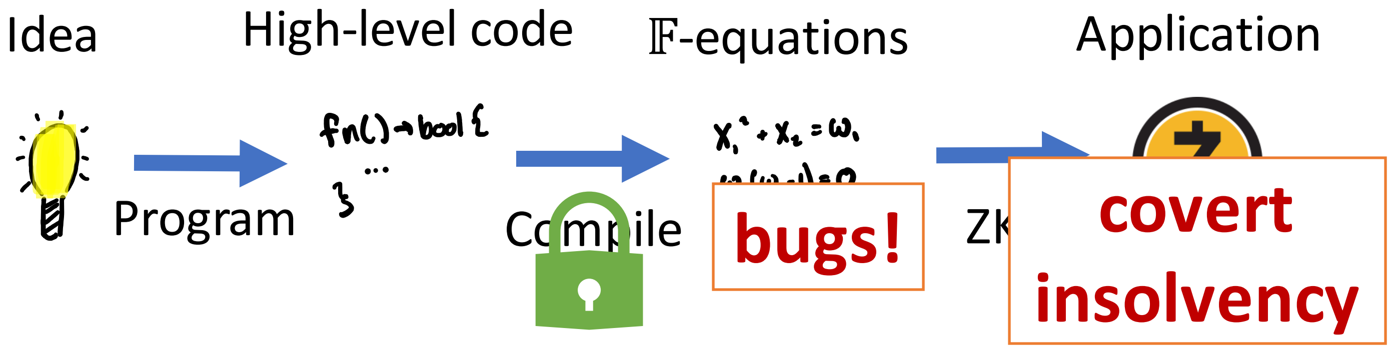
(Simplified) Soundness bug

- Rule for: $z = x \geq_s 0$
  - $x \in \mathbb{Z}_{2^b}$, signed comparison
  - $\geq_s$: signed $\geq$
- Assume: signed representation
  - $x' \in \{-2^{b-1}, \dots, 2^{b-1} - 1\} \subset \mathbb{F}$
- Goal: $z' = \text{IfThenElse}(z, 1{,}0)$
  - $z' = 1$ iff
  - $x' \in \{0, \dots, 2^{b-1} - 1\}$

- Broken approach:
  - "Attempt" an unsigned decomposition
  - Introduce $y' \in \{0, \dots, 2^{b-1} - 1\}$
    - $y' = \sum_{i=0}^{b-2} 2^i b_i'$
    - $b_i'(b_i' - 1) = 0,\ i \in \{0, \dots, b-2\}$
  - $z' = \text{AreEqual}(x', y')$
- If $x <_s 0$: y' cannot equal x'
- If $x \geq_s 0$: y' can still differ from x'!
- Potential impact: insolvency
- Fix: split x' into *signed* bits
  - flip sign bit

Satisfiability Modulo 𝔽 & Bounded Verification for 𝔽-Blasting

Alex Ozdemir, Gereon Kremer, Riad S. Wahby, Cesare Tinelli, Fraser Brown, Clark Barrett



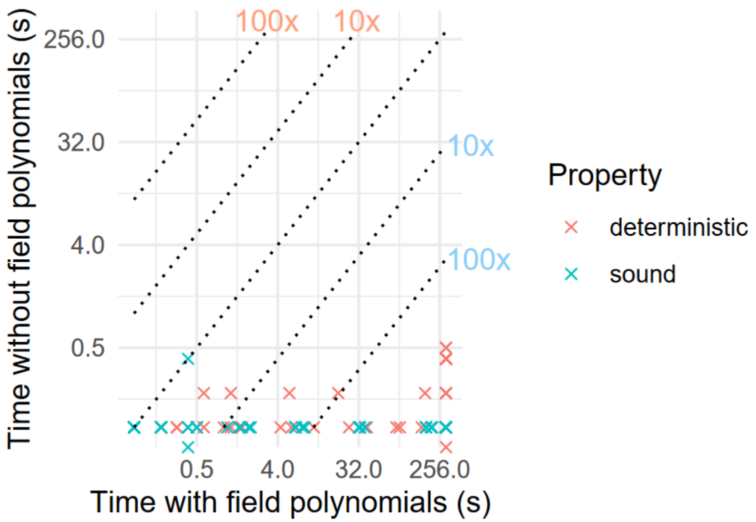
Talk Summary

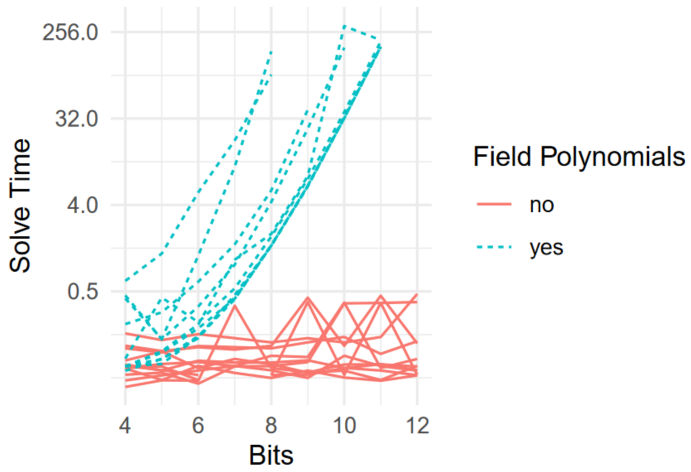
I. ZKP compiler correctness
II. Verified 𝔽-blasting
III. Satisfiability modulo 𝔽
    - Avoids field polynomials
IV. Case study: CirC (4 bugs)

Thesis: Verification for ZK is ***crucial***. With SMT+𝔽 it is ***feasible***.
What should we verify next?
Where else do fields matter?

Appendices

Effect of Field Polynomials





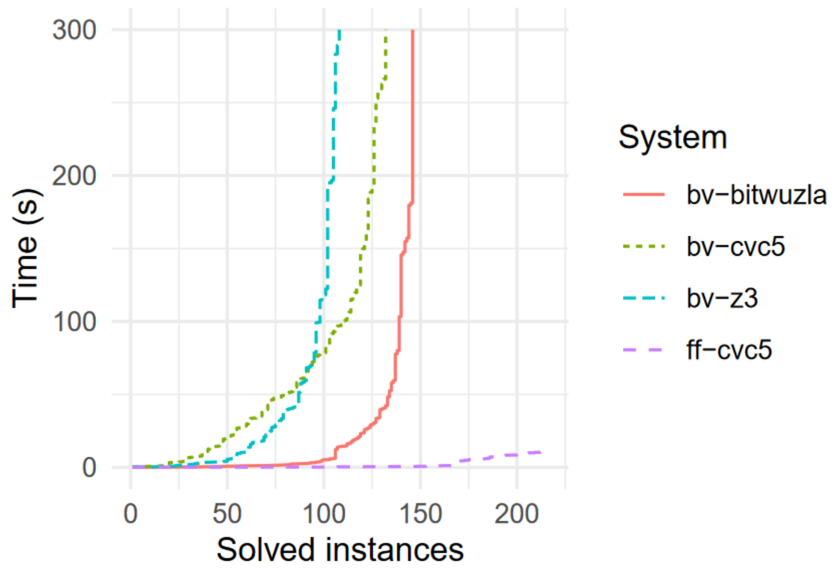
(p: 4-12 bits, 8GB, 5min)

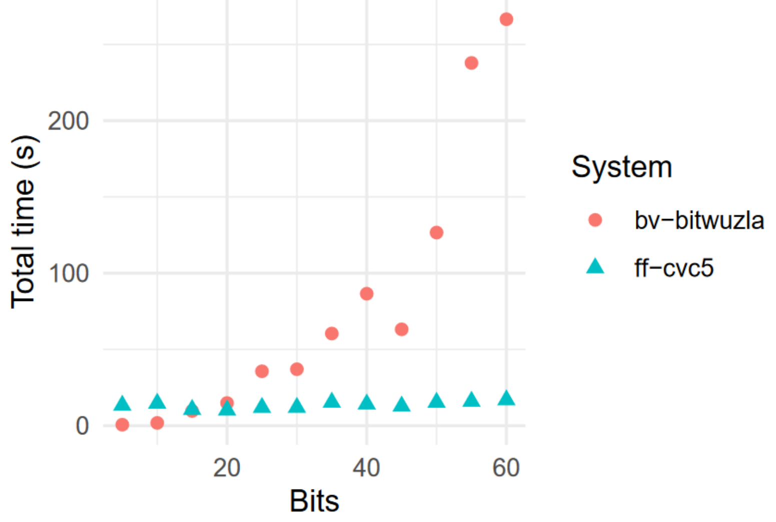
(commonly solved SAT)

Field polys: terrible for even tiny fields

Field polys scale poorly with field size

Comparison with Bit-Vectors





(p: 5-60 bits, 1CPU, 8GB, 5min)

(commonly solved)

BV is worse, even for small fields

BV scales poorly with field size