SMT + Finite Fields

A foundation for field-related verification

Alex Ozdemir, Gereon Kremer, Cesare Tinelli, Clark Barrett [CAV'23]

Verifying Equations for a ZKP:

Idea

bugs!

Compilation

Equations ($\mathbb{Z}_p$)

$x_1^2 + x_2 = \omega_1$

$\omega_1(\omega_1 - 1) = 0$

$\vdots$

ZKP

(Zero-Knowledge Proof)

Private

Cryptoc

insolvent!

Can we verify this?

SMT Solver: a general-purpose theorem prover/constraint solver

Specification

Boolean logic ($\mathbb{Z}_2$)

Bit-vectors ($\mathbb{Z}_{2^b}$)

Arrays …

SMT

Implementation

equations in $\mathbb{Z}_p$

Not SMT

~~equations in $\mathbb{Z}$ (mod p)~~

~~equations in $\mathbb{Z}_{2^b}$ (mod p)~~

Our Contribution: $\mathbb{Z}_p$ reasoning in SMT

- A theory of finite fields ($\mathbb{Z}_p$)
- Verification problems
  - translation validation for ZKP compilers (Boolean functions)
- A decision procedure
  - uses computer algebra
  - avoids field polynomials
- An implementation in cvc5

Automated Verification

(SMT) solvers

$+ \mathbb{Z}_p$

Bonus: An application to partially verified finite-field blasting (in CirC)

Prime-Order Finite Fields

- Written $\mathbb{Z}_p$
- A set of integers: $\{0, \dots, p-1\}$
- 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

Encoding a Boolean Function as Equations

- Consider a Boolean formula $f(z_i) \to z$
- Consider a system ϕ':
  - In variables:
    - z_i': represents z_i
    - z': represents z
    - additional variables
- ϕ' is **sound** w.r.t f when this is valid:
  - $\underbrace{[\phi'}_{\text{equations hold}} \wedge \underbrace{\bigwedge_i z_i' = \text{ite}(z_i, 1{,}0)]}_{\text{inputs are valid}} \to \underbrace{z' = \text{ite}(f(z_i), 1{,}0)}_{\text{output is correct}}$

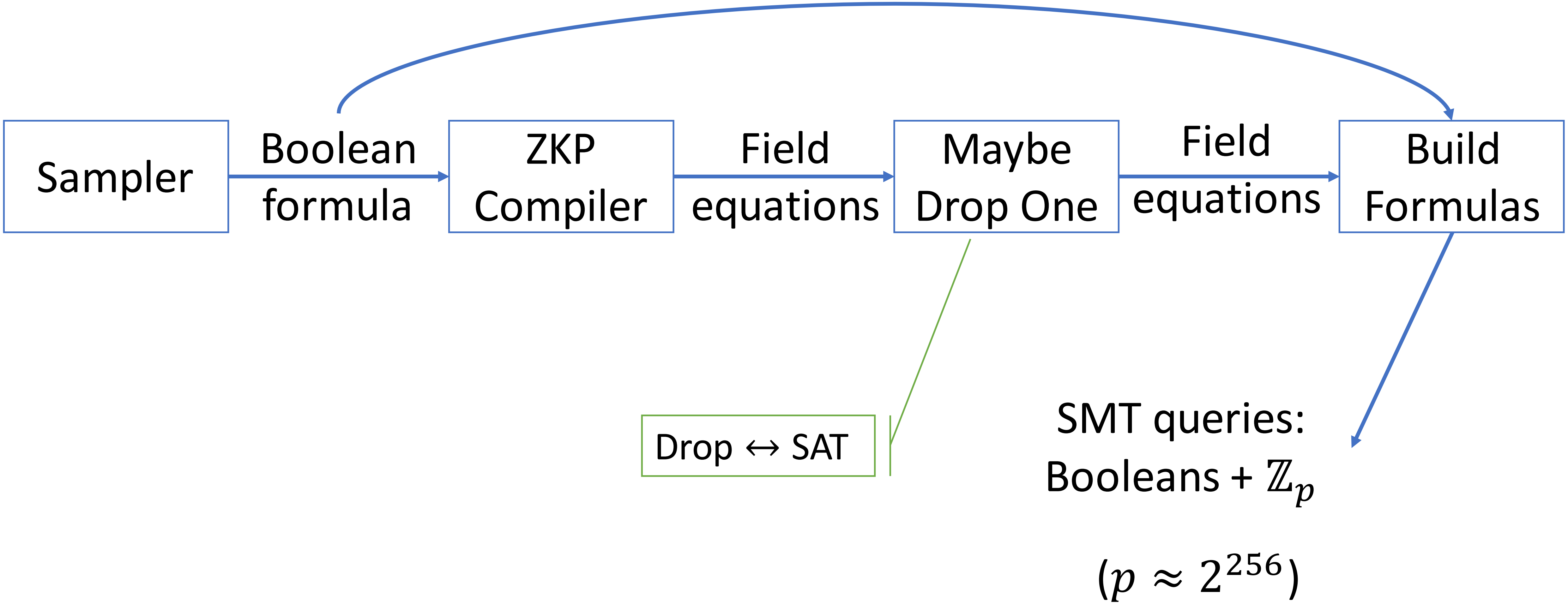
equations hold $\wedge$ inputs are valid $\longrightarrow$ output is correct

Example

$Z = Z_0 \wedge Z_1 \wedge Z_2$

$Z' = Z_0' Z_1' Z_2'$

Generating Verification Problems

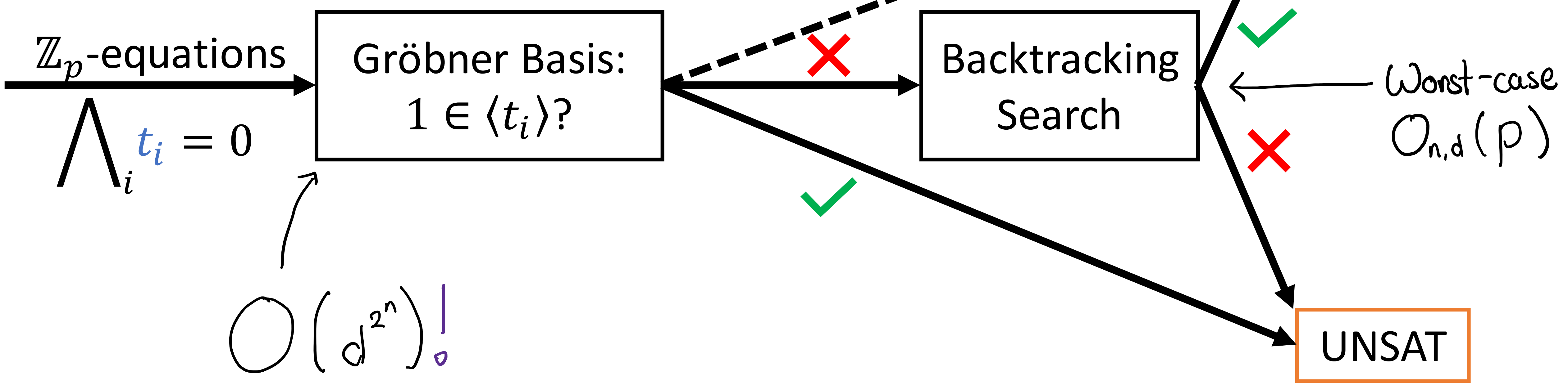


Decision Procedure:

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

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

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

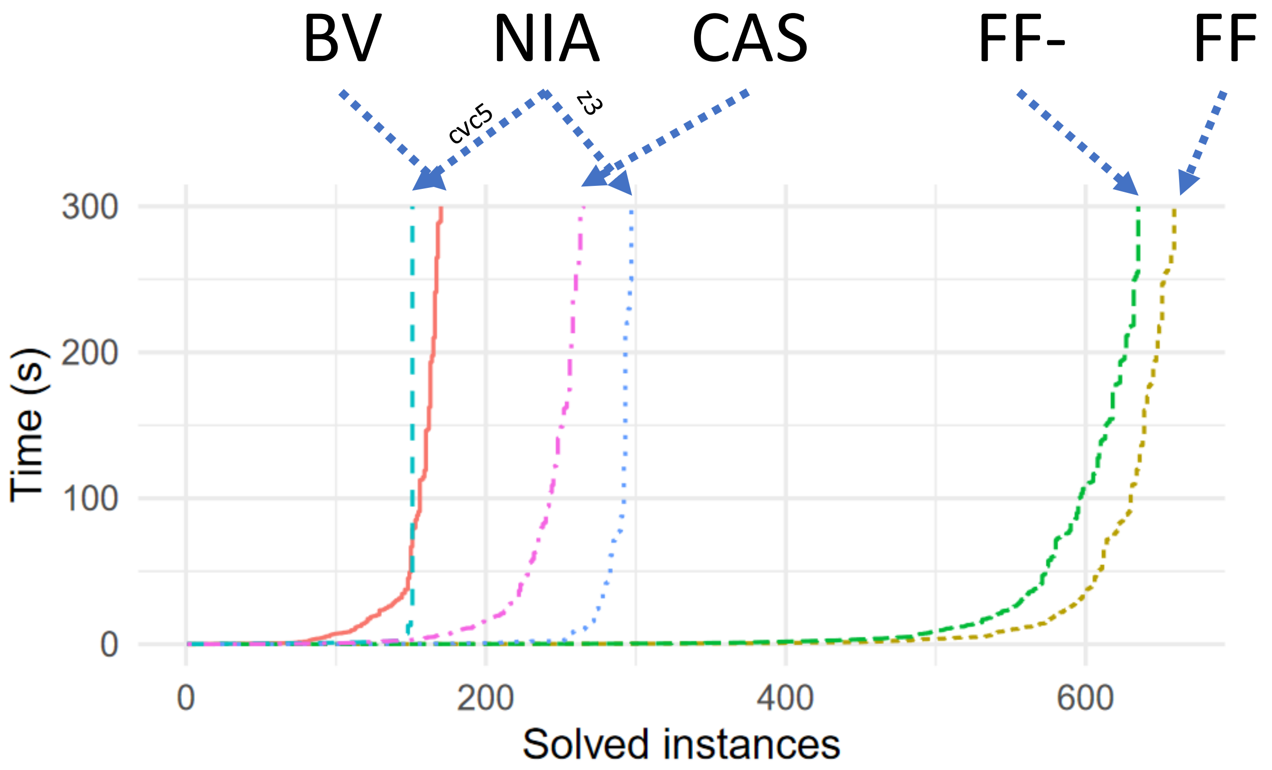


Main Experiment

Full-size field: $p \approx 2^{255}$

What we benchmarked:

- Our solver (FF)
- Our solver, no cores (FF-)
- Embed $\mathbb{Z}_p$ in bit-vectors (BV)
- Embed $\mathbb{Z}_p$ in integers (NIA)
- Embed Booleans in $\mathbb{Z}_p$ (CAS)



solvers: bitwuzla-1.0pre, z3-4.11.2, cvc5 nightly 17 Dec '22, CoCoALib 0.998

More in the paper

- Translation validation for determinism
- Full decision procedure
  - backtracking search
  - for *all* finite fields
  - UNSAT cores
    - Calculus for proving ideal membership
- Detailed benchmarks

Future Work:

- Asymptotic improvements?
  - $O_{d,n}(p)$ to $O_{d,n}(\log p)$?
- Incremental computation *within* field solver.

Bounded Verification in CirC

“Bounded Verification for Finite Field Blasting”

Alex Ozdemir, Riad S. Wahby, Fraser Brown, Clark Barrett [CAV’23]

The Verification Problem

Idea

Programming/
Compilation

Equations ($\mathbb{Z}_p$)

$$x_1^2 + x_2 = w_1$$
$$w_1(w_1 - 1) = 0$$
$$\vdots$$

ZKP

(Zero-Knowledge Proof)

Private Cryptocurrency

Can we verify this?

Previous benchmarks: translation validation

What about (automatic) verification?

The CirC Compiler Infrastructure

Program → IR[…] → IR[Bool,BV,$\mathbb{Z}_p$] → $\mathbb{Z}_p$-Equations

Front-End

Optimization,
Simplification

"Finite-Field Basting"

<u>Encoding Rules</u>

bv.add	bv.ule	bv.sgt
bv.sub	bv.ult	bv.or
bv.lshl	bv.ugt	bv.xor

ff.neg
ff.mul

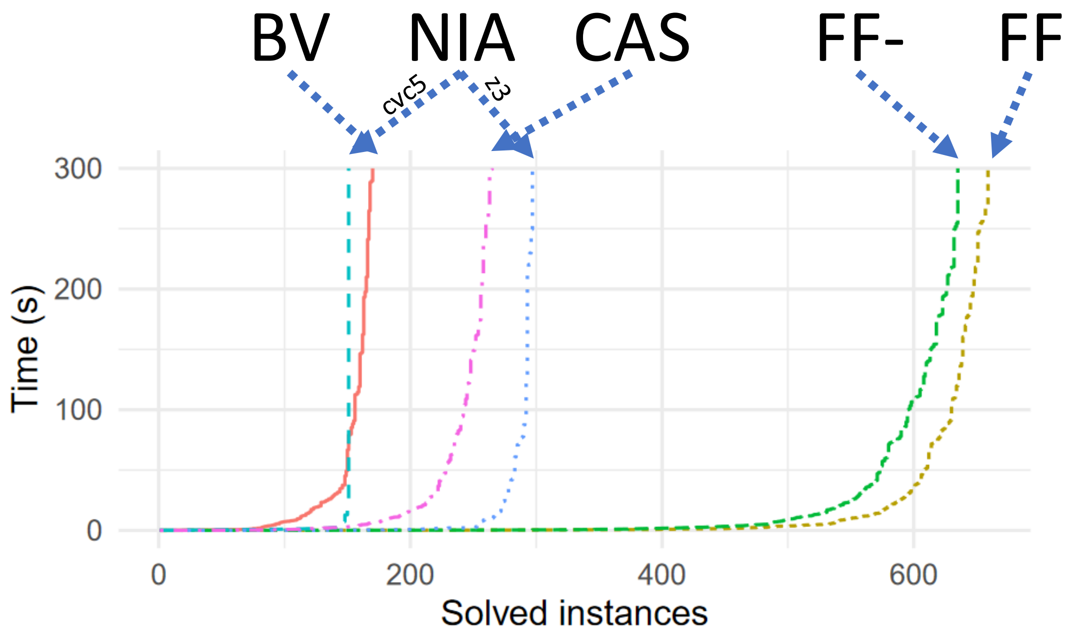
Each rule
→
SMT query

4 bugs!

IR traversal → induction proof
valid encoding → inductive invariant

Satisfiability Modulo Finite Fields

Alex Ozdemir, Gereon Kremer, Cesare Tinelli, Clark Barrett [CAV'23] ePrint: https://ia.cr/2023/091



- Solver for SMT with $\mathbb{Z}_p$
- First SMT $\mathbb{Z}_p$ benchmarks
  - ZKP security
- Application: CirC's $\mathbb{Z}_p$-blaster
- Try it! (cvc5 main, all APIs)

Bounded Verification for Finite-Field Blasting

Alex Ozdemir, Riad S. Wahby, Fraser Brown, Clark Barrett [CAV'23]

Appendices

Two Properties

<u>Soundness</u>

Any solution with valid inputs has the correct output.

$$\phi' \wedge \bigwedge_i z_i' = \mathrm{ite}(z_i, 1,0) \\ \rightarrow \\ z' = \mathrm{ite}(f(z_i), 1,0)$$

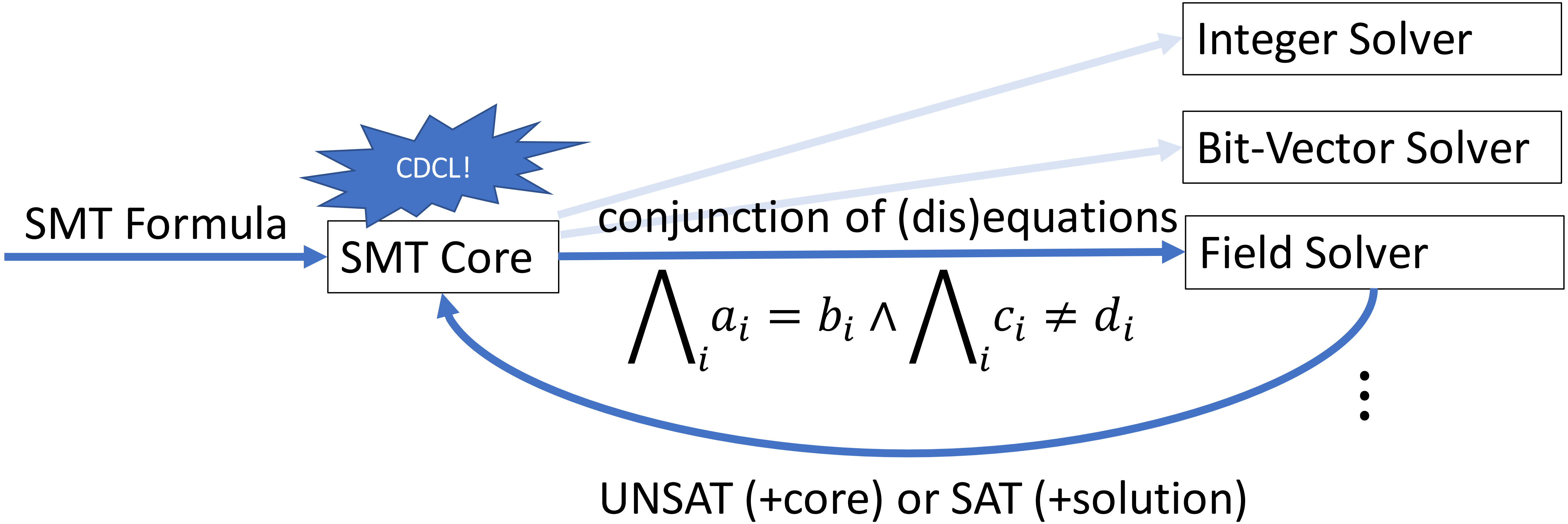
<u>Determinism</u>

Any solution, for fixed inputs, has a unique output.

$$\phi' \wedge \phi'' \wedge \bigwedge_i z_i' = z_i'' \\ \rightarrow \\ z' = z''$$

copy of ϕ'

What does a theory solver do?



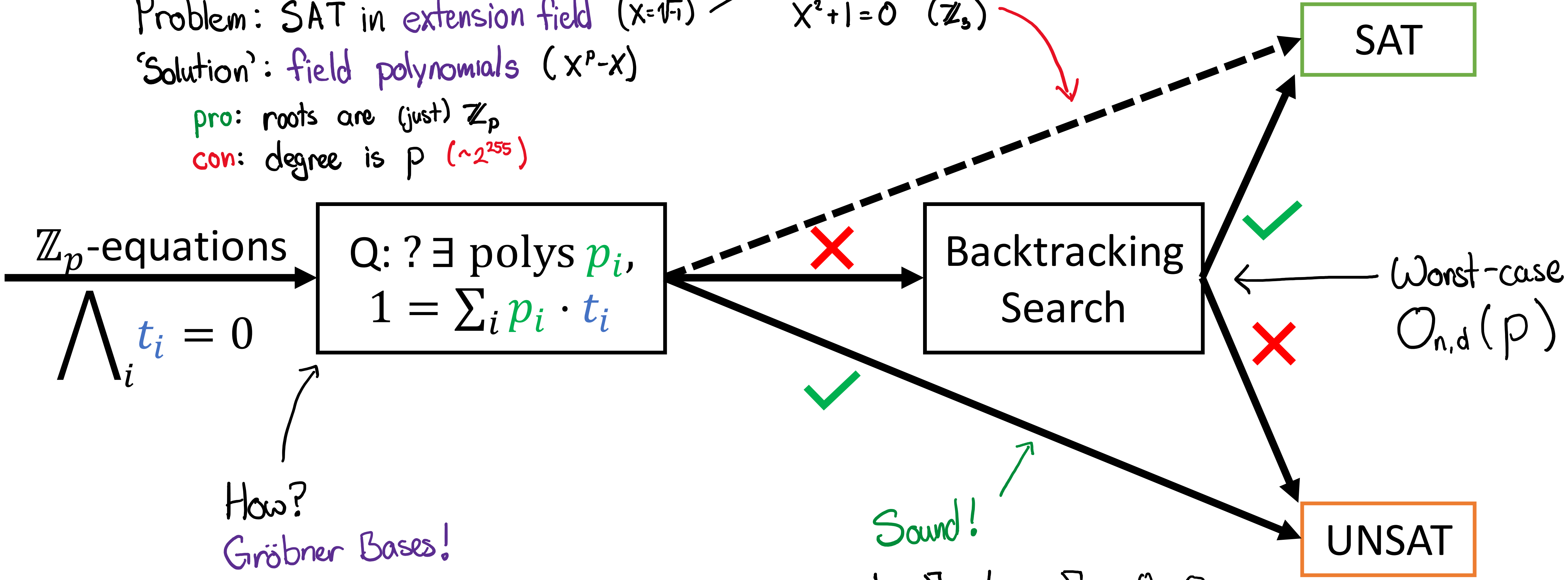
Decision Procedure:

Problem: SAT in extension field ($x = \sqrt{-1}$) → Unsound! $x^2+1=0$ ($\mathbb{Z}_3$)

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

pro: roots are (just) $\mathbb{Z}_p$

con: degree is p ($\sim 2^{255}$)



How?
Gröbner Bases!
$O(d^{2^n})$

Sound!
$1 = \sum p_i t_i = \sum_i p_i \cdot 0 = 0$

$\mathbb{Z}_p$ Solver Pseudocode

1. Given: equalities & disequalities
2. Rewrite as: $t_i = 0$
   - t_i: a polynomial in the input variables
3. Exist polys p_i such that $1 = \sum_i p_i t_i$?
   - Return UNSAT (with core)
4. Backtracking search for a solution

Incomplete, fast UNSAT

Slow, complete.
Finds solution.

Exist polys p_i such that $1 = \sum_i p_i t_i$? (⇒UNSAT)

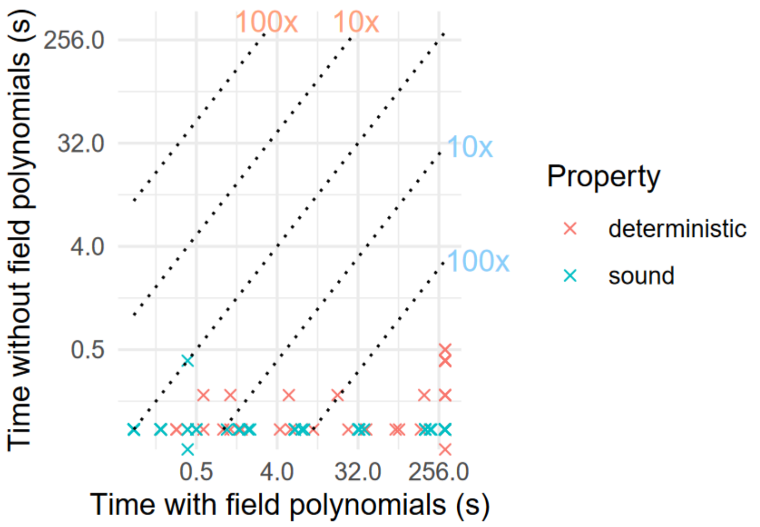
- Example:
  - $t_1 = XY - 1, t_2 = X$
  - $(-1)t_1 + (Y)t_2$
    - $= -XY + 1 + XY = 1$
- Gröbner Basis engine:
  - Determines whether the p_i exist
  - Worst-case time: $O\left(d^{2^n}\right)$
  - Empirical time: fast

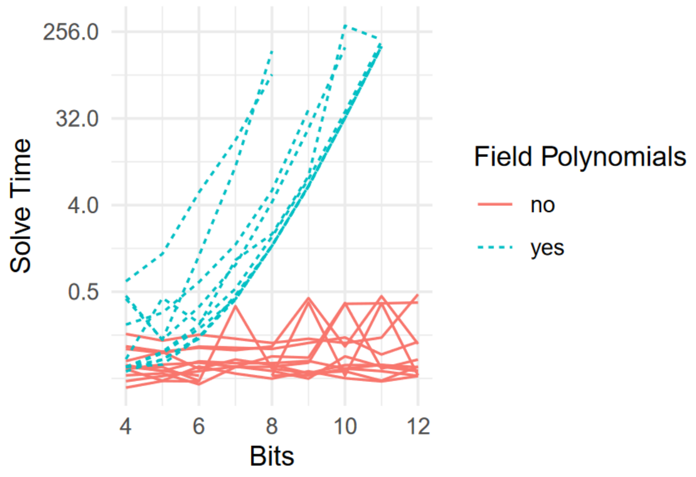
- $\exists p_i \Rightarrow \text{UNSAT}$ (why?)
  - $1 = \sum_i p_i \cdot t_i = \sum_i p_i \cdot 0 = 0$
- **Problem:**
  - $\neg\exists p_i \not\Rightarrow \text{SAT}$
    - Example: $X^2 + 1 = 0$ is UNSAT in $\mathbb{Z}_3$
    - Key issue: $X = \sqrt{-1}$ is a solution
- Classic solution:
  - Add polynomials $X^p - X$
    - $\text{roots}(X^p - X) = \{0, \dots, p-1\}$
  - **Problem:** high-degree ⇒ slow GB

Live with the incompleteness…

Part IV: Experiments

Effect of Field Polynomials





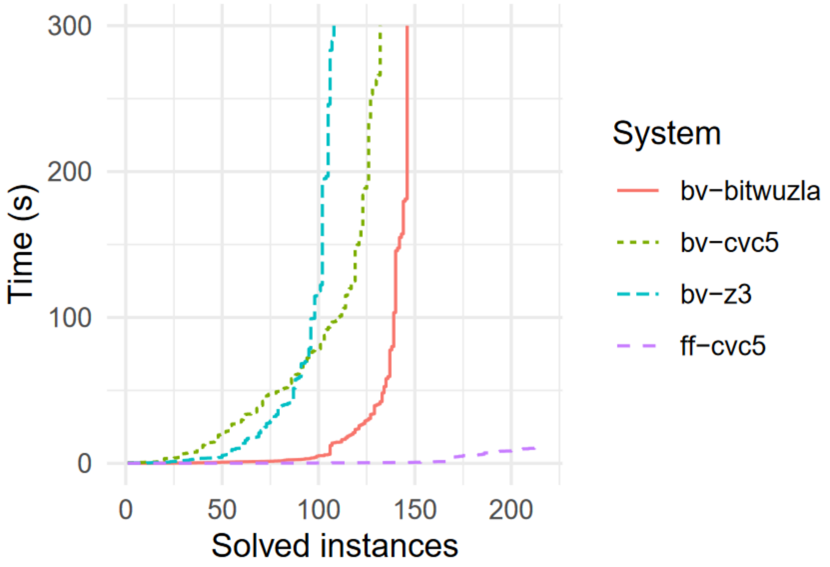
(p: 4-12 bits, 8GB, 5min)

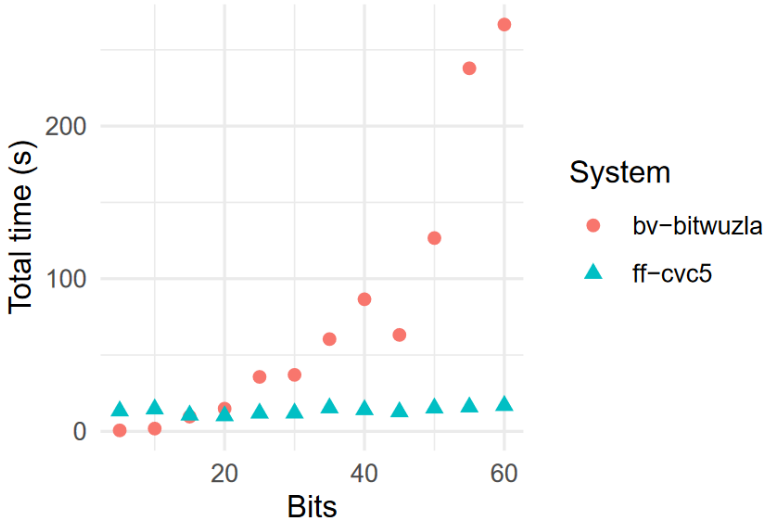
(commonly solved SAT)

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

Correctness for a ZKP compiler

<u>Syntax:</u>

equisat

$$\text{Compile}(\phi(x, w) \in \Phi) \to \phi'(x', w') \in \Phi'$$

$$\text{Ext}_x(x) \to x'$$

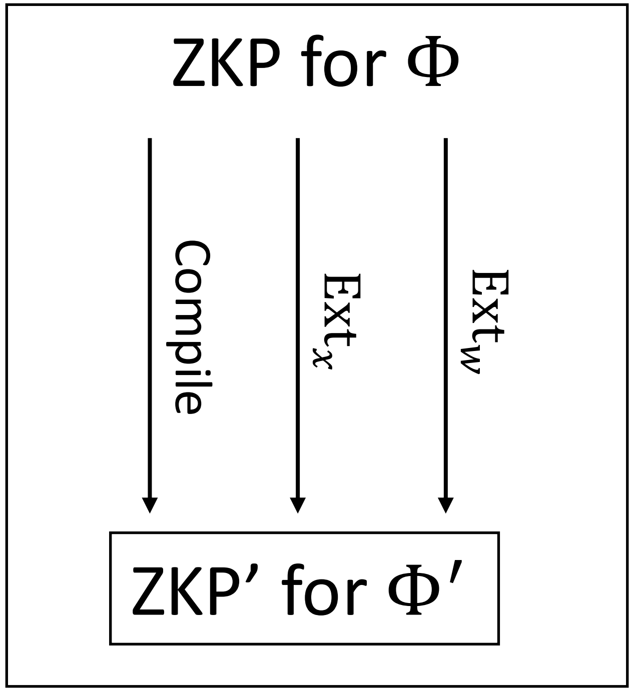
$$\text{Ext}_w(x, w) \to w'$$

<u>Correctness:</u>

- *Demonstrable* Soundness
- *Demonstrable* Completeness

<u>Key Properties:</u>

- Closed under sequential composition
- ZKP is secure



Verification Recipe

- Post-order traversal → inductive proof
- Encoding 'validity' → inductive invariants
- Each encoding rule → verification conditions
  - Soundness + completeness
  - Automatically generated
  - Automatically verified (SMT)
    - Up to tiny bounds on:
      - bit-vector length (4)
      - operator arity (4)

<u>4 Bugs</u>

- Incompleteness in `bv.*sh*`
- Incompleteness in `ff.div`
- Unsoundness in `bv.udiv`
- Unsoundness in bv comparisons
  - **Severe**