Satisfiability Modulo Finite Fields

or: Alex's Quals, Part II

Alex Ozdemir, Gereon Kremer, Cesare Tinelli, Clark Barrett

How Private Cryptocurrencies are Built

Idea

Programming

bugs!

Equations ($\mathbb{Z}_p$)

$x_1^2 + x_2 = \omega_1$

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

$\vdots$

ZKP

(Zero-Knowledge Proof)

Private Cryptocurrency

insolvent!

Idea

Programming

bugs!

Assembly

```
%a = sub %0, 1
%b = add %a, %0
```

CPU

Application

crash!

The Verification Problem

Idea

Programming

Equations ($\mathbb{Z}_p$)

$x_1^2 + x_2 = \omega_1$

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

⋮

ZKP

(Zero-Knowledge Proof)

Private Cryptocurrency

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 (previously)

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

- A theory of finite fields ($\mathbb{Z}_p$)
- Benchmarks
  - verifying ZKPs of Boolean function evaluation
- A decision procedure
  - based on computer algebra
  - constructs UNSAT cores
  - avoids field polynomials
- An implementation in cvc5
  - Empirically fast

this talk

Automated Verification

(SMT) solvers + $\mathbb{Z}_p$

Part I: Prime-Order Finite Fields

- Written $\mathbb{Z}_p$
- A set of integers: $\{0, \dots, p-1\}$
- Operations:
  - $+ \pmod p$
  - $\times \pmod p$
  - $=$

Theory of prime-order finite fields

- For each prime p
  - A sort FF_p
  - Function symbols $+, \times, =$
  - Constants $0, 1, \dots, p-1$
- Notes:
  - fixed p
  - distinct "sub-theories"
  - no conversions

$X = X+1$

UNSAT

$X = X+1$

$\vee$

$Y^2 = Y$

SAT : $X=0, Y=0$

Part II: Verifying encodings of Boolean function evaluation

Encoding a Boolean Function as Equations

- Consider a Boolean $f(z_i) \to z$
  - Ex: $z = f(z_0, z_1) = z_0 \wedge z_1$
- We want a system ϕ':
  - In variables:
    - z_i': represents z_i
    - z': represents z
    - w_i': additional variables
  - Such that
    - $[\phi' \wedge \bigwedge_i z_i' = \text{ite}(z_i, 1, 0)] \to z' = \text{ite}(f(z_i), 1, 0)$

equations hold

inputs are valid

output is correct

For our example:

$$z' = z_0' z_1'$$

Two Properties

Soundness

Any solution with valid inputs has the correct output.

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

Determinism

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'

How We Generate Verification Problems

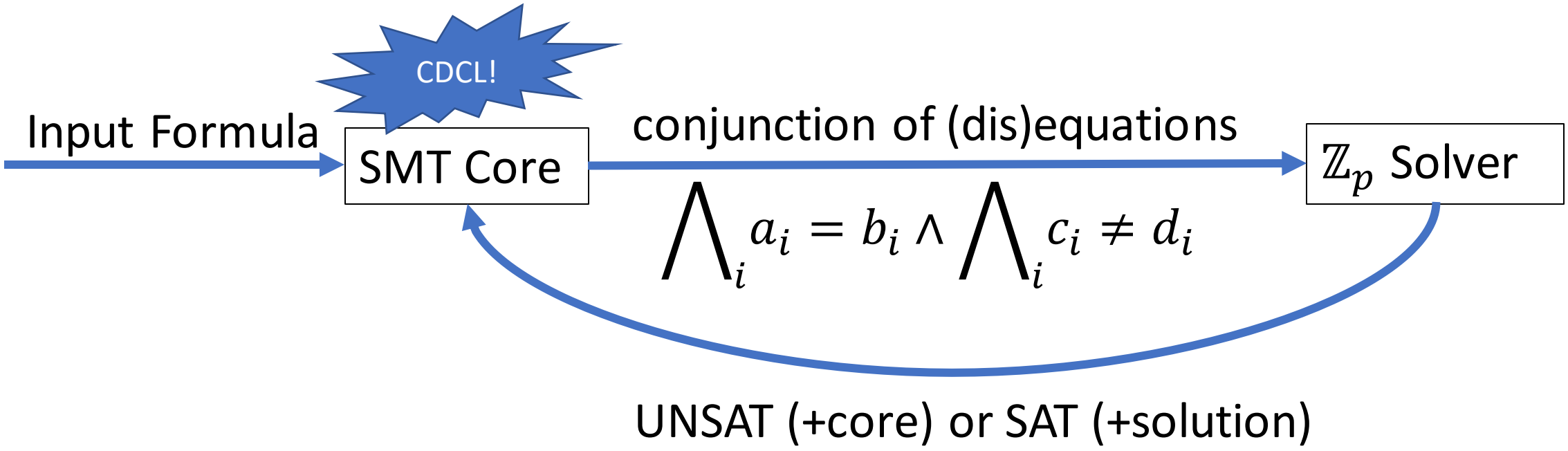
1. Sample a Boolean function $f(z_i) \rightarrow z$
2. Compile it to a system of equations: $\phi \leftarrow \mathrm{ZKPCompiler}(f)$
3. (Optional) remove some equations
4. Write the soundness and determinism conditions

- ZoKrates
- CirC
- CirC-ZoK

- Removals influence condition validity:
  - No removals: VALID (or the compiler has a bug!)
  - Removals: likely INVALID
- Benchmarks are SMT-LIB2
  - QF_FF: mix of Boolean and $\mathbb{Z}_p$ ($p \approx 2^{256}$)

Part III: Decision Procedure

SMT + Theory Solvers



$\mathbb{Z}_p$ Solver Pseudocode

1. Given: equalities & disequalities
2. Convert each to form: $t_i = 0$
   - t_i: a polynomial in the input variables
3. Exist polys p_i such that $1 = \sum_i p_i t_i$?
   - Then $1 = 0$! Return UNSAT (with core)
4. Backtracking search for a solution
   - If ∃ univariate $t_i(X_j)$: factor & branch
   - If $\dim(t_i) = 0$: compute a univariate, factor, & branch
   - Else: exhaustive search

next

Rules:

Note: incomplete!

- $\exists p_i . 1 = \sum_i p_i t_i \Rightarrow \text{UNSAT}$
- $\neg(\exists p_i . 1 = \sum_i p_i t_i) \not\Rightarrow \text{SAT}$
- Example: $X^2 + 1 = 0$ is UNSAT in $\mathbb{Z}_3$
- $X = 0$
- $X \neq 0 \Rightarrow XW - 1 = 0$ except $X = i$ ü

- Could recover completeness by adding $X^p - X$ for all variables X
- math fact: roots: all of $\mathbb{Z}_p$
- high degree → terrible performance

- Could perform *terribly.*
- Experiments show otherwise.
- Insight: + dimension ⇝ many solutions

Part IV: First check & UNSAT cores

Equations and polynomials: dual view

Equation view:

- Equations: $t_i = 0$
- Q: do they imply $1 = 0$?
- Inference rules:
  - $t_i = 0$ implies $pt_i = 0$ $(\forall p)$
  - $t_i = 0, t_j = 0$ implies $t_i + t_j = 0$
- So:
  - If $1 = \sum_i p_i t_i$ (for some p_i)
  - Then the equations imply $1 = 0$

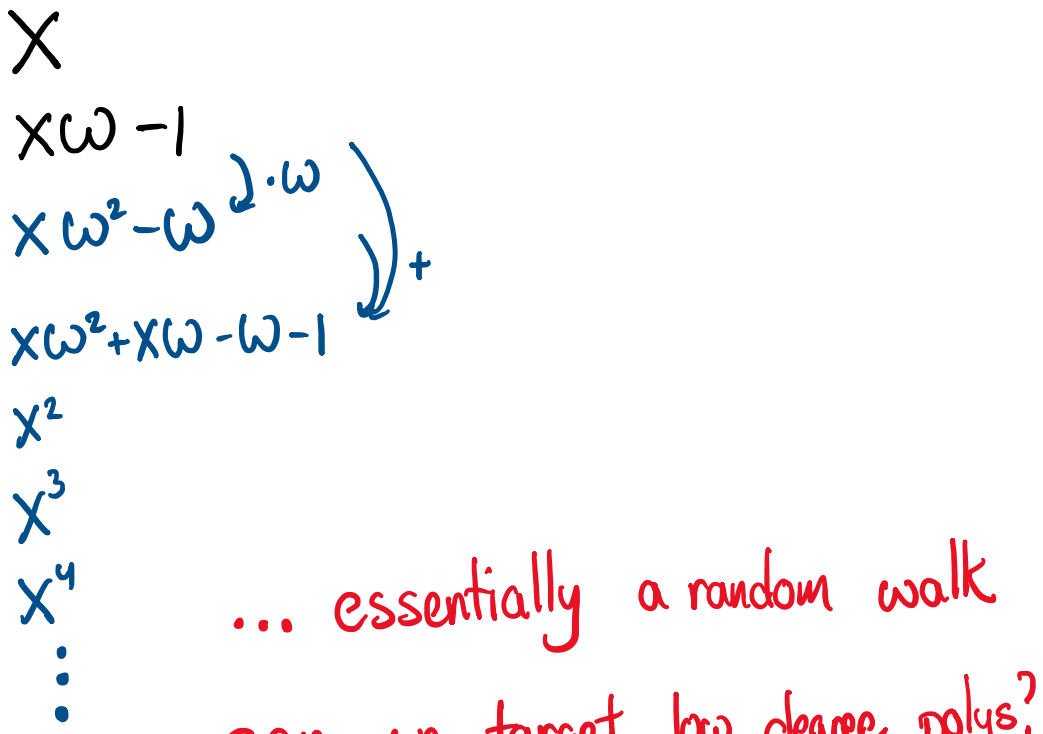
Polynomial view:

- Polynomials: t_i
- Define: $\langle t_i \rangle = \{\sum_i p_i t_i : p_i\}$
  - The set is called an *ideal*
  - closed under scaling and addition
- Q: $1 \in \langle t_i \rangle$?
- Example:
  - $t_0 = X, \quad t_1 = XW - 1$
  - $p_0 = W, \quad p_1 = -1$

$XW + -XW + 1 = 1 \neq 0$

Strawman algorithm

Idea: add polys until we find 1:



Monomial Ordering: A ‘direction’

- A *monomial*: $X_1^{e_1} \cdots X_k^{e_k}$
- The ‘graded reverse lexicographical order’ (grevlex)
  - Lexicographic on
    $(e_1 + \cdots + e_k, e_1, \ldots, e_k)$

$X_1 < X_1^2$

$X_1 < X_2^2$

$X_2^2 < X_1 X_2$

- Note: the poly 1 is grevlex-min
- $LM(f)$: largest monomial of f
- $LT(f)$, $LC(f)$: its term & coefficient

Reduction: making progress

fn reduce$(p, \{d_i\}) \to r$:

  $r \leftarrow p$

  while $\exists i. \exists t \in r. LM(d_i) \mid t$:

    $r \leftarrow r - \frac{t}{LM(d_i)} d_i$

  return $r / LC(r)$

replaces t with grevlex< monomials

Examples:

reduce$(WX - 1, \{X\})$:

r = ωX-1 - ω(X)

r = -1

return 1

reduce$(WX - 1, \{XY, YZ - 1\})$:

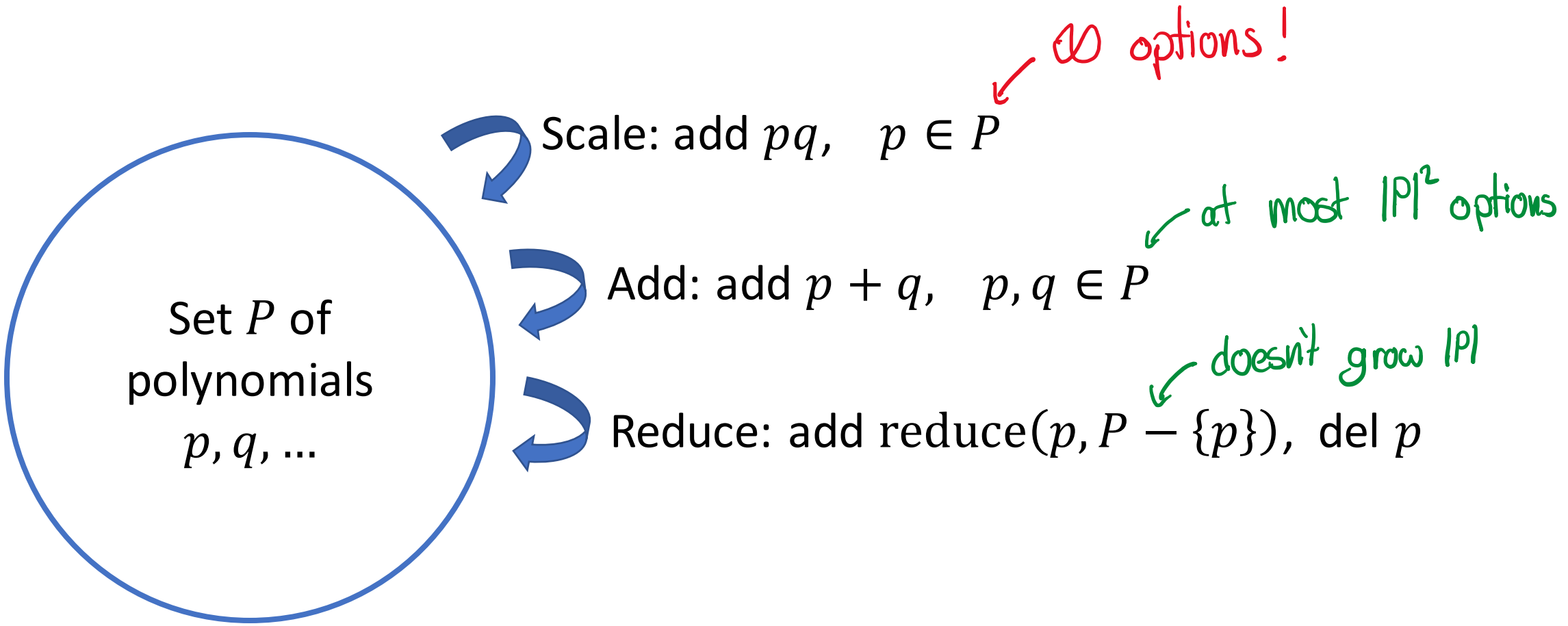
r = ωX-1

return ωX-1

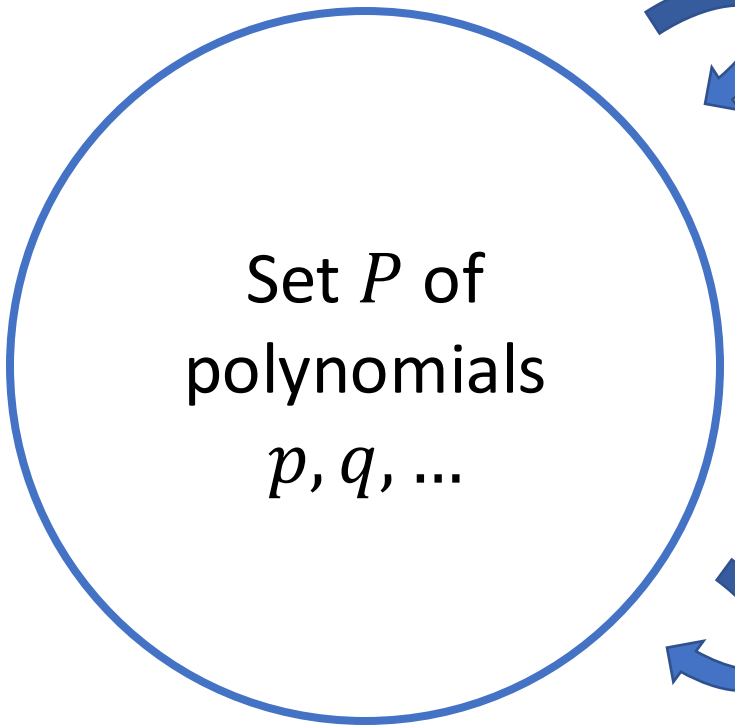
- Reduction denoted as: $p \to_{\{d_i\}} r$
- If $p \to_D r$, then $\langle p, D \rangle = \langle r, D \rangle$

translation: reduction preserves implied equations

Candidate algorithm



Buchberger's algorithm



[Buchberger '65, corollary]:
With Reduce and SPoly, then

$$1 \in \langle P \rangle$$

iff $1 \in P$ (at fixed point)

Reduce: add $\text{reduce}(p, P - \{p\})$, del p

SPoly: for $p, q \in P$:

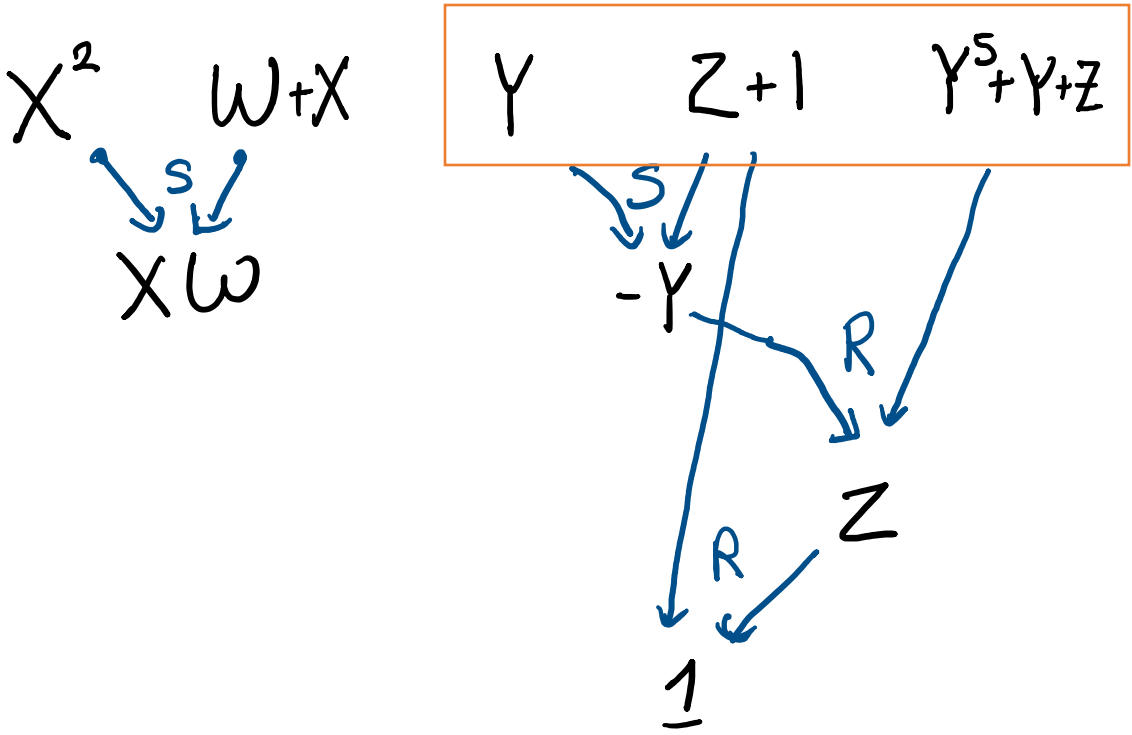
$s \leftarrow LT(p)q - LT(q)p$

add $\text{reduce}(s, P)$ if non-zero

UNSAT cores

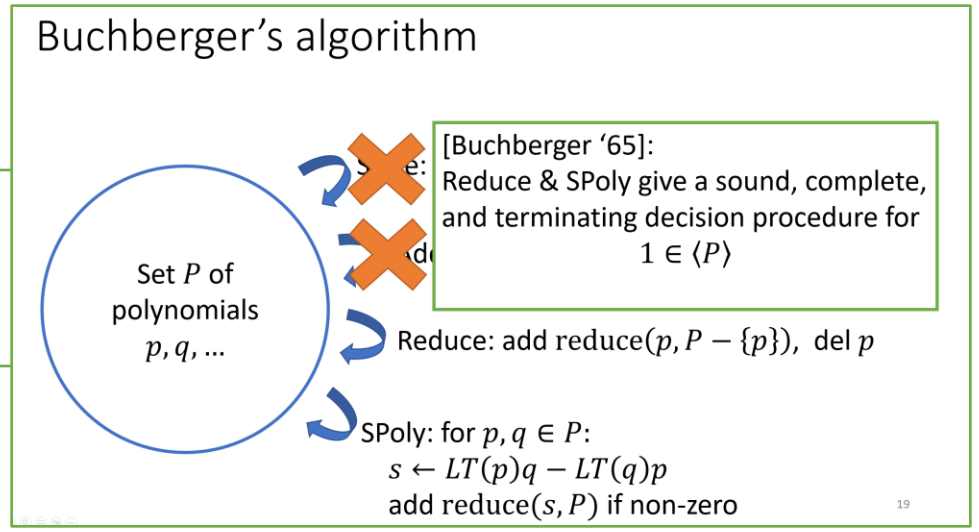
Computable from a simple blame-tree:

- if $s \leftarrow LT(p)q - LT(q)p$
  - s blames p, q
- if $p \rightarrow_{\{p_i\}} r$
  - r blames p
  - r blames each (used) p_i
- traverse blame graph for UNSAT core



$\mathbb{Z}_p$ Solver Pseudocode

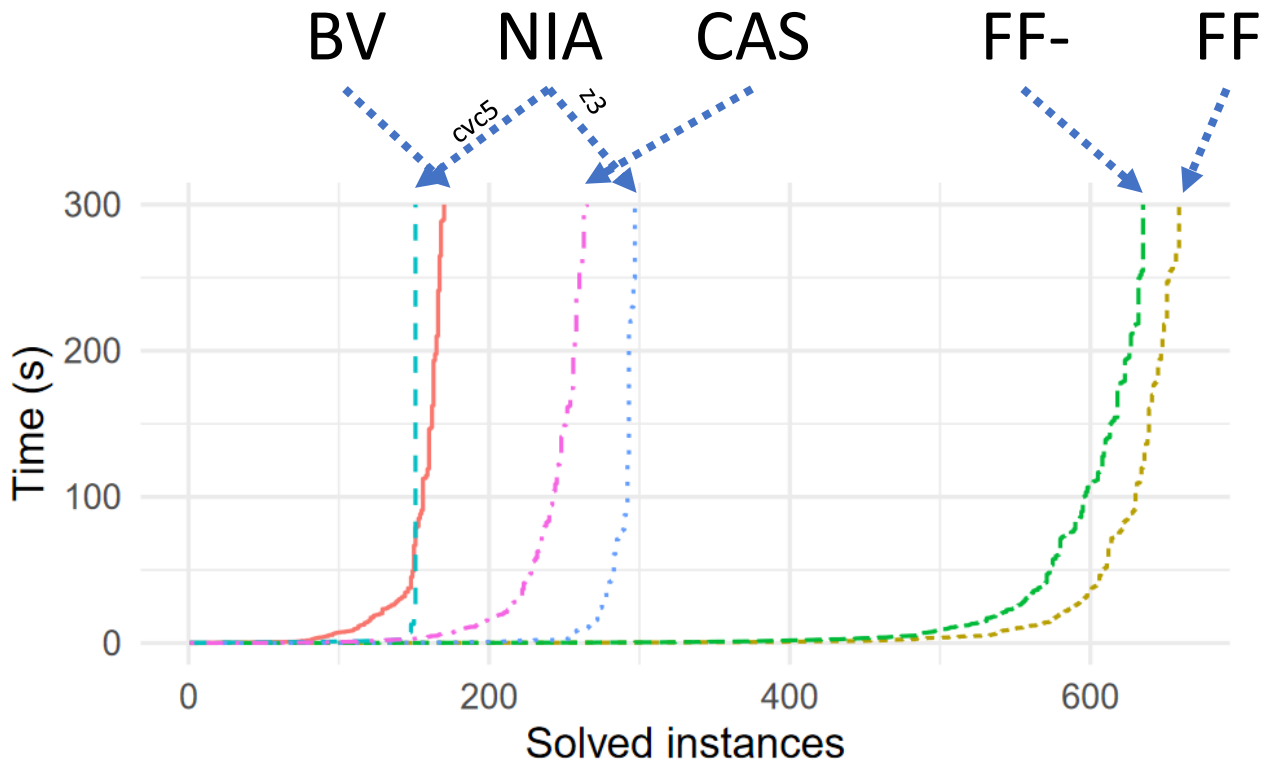
1. Given: equalities & disequalities
2. Convert each to form: $t_i = 0$
   - t_i: a polynomial in the input variables
3. Exist polys p_i such that $1 = \sum_i p_i t_i$?
   - Then $1 = 0$! Return UNSAT (with core)
4. Backtracking search for a solution
   - If ∃ univariate $t_i(X_j)$: factor & branch
   - If $\dim(t_i) = 0$: compute a univariate, factor, & branch
   - Else: exhaustive search



Main Experiment

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 testbed: 8GB, 1 CPU, 5min

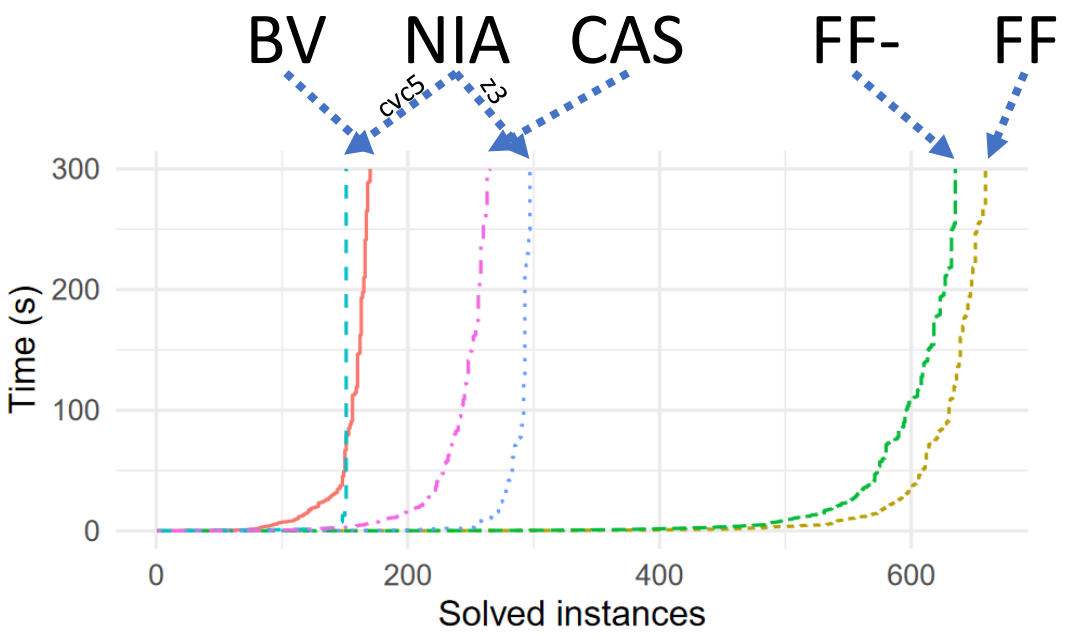
More in the paper

- Theory & decision procedure for non-prime finite fields.
- Calculus for proving ideal membership
  - Generalization of our UNSAT cores
- Avoiding field polynomials: costs & benefits
- More experiments
  - Comparison with BV for 10-60 bit fields
  - Comparison with field polynomials for 4-12 bit fields
  - …

Satisfiability Modulo Finite Fields

Alex Ozdemir, Gereon Kremer, Cesare Tinelli, Clark Barrett

ePrint: https://ia.cr/2023/091



- First SMT $\mathbb{Z}_p$ solver
  - UNSAT cores
  - No field polynomials
- First QF_FF benchmarks
  - Verifying encodings of Boolean function evaluation (for ZKPs)
- A foundation for automated verification related to $\mathbb{Z}_p$
- Try it! (cvc5 main)