Primes, feasible computations and reasoning

Ondřej Ježil

Department of algebra, Charles University, Prague

Based on joint work with Raheleh Jalali
Institute of Computer Science, Czech Academy of Sciences, Prague

September 9, 2024

Outline

1. Primality testing
2. Feasibility of computations: Complexity theory
3. Feasibility of proofs: Bounded arithmetic
4. Formalization of the correctness of the AKS algorithm in bounded arithmetic.

Outline

1. Primality testing
2. Feasibility of computations: Complexity theory
3. Feasibility of proofs: Bounded arithmetic
4. Formalization of the correctness of the AKS algorithm in bounded arithmetic.

Questions and remarks during the talk are very welcome!

Definition

A number $p \in \mathbb{N}$ *is called a prime if* $p > 1$ *and* p *has no nontrivial divisors.*

Definition

A number $p \in \mathbb{N}$ is called a prime if $p > 1$ and p has no nontrivial divisors.

Exercise

Can you use this definition to determine if the following number is a prime?

2106800732833597706307195705474469424926136873112326458104745687770320164079926789400548792757695 1–
601821767003813882303695154485989728507094460976554996886298647627850807732402816244768564719732 23–
766401466562169055974085501809337335924570625143372572946144701541013306558460953858000220988661 08–
719034192901256958183461580924275314837795769862690721642146705295171082618791918454138913343631 10–
070273630426433132184997541746133187406885847969653006790696804617596751665002857237805566365516 81–
038389826862723791173790479016397786477588977368875258729097122126735064035044936730312725075620 25–
1360365167806284927865418893 1

Primality tests

- (200 BC) Sieve of Eratosthenes – essentially trial division

Primality tests

- (200 BC) Sieve of Eratosthenes – essentially trial division
- (1976–1980) Miller–Rabin Probabilistic polynomial-time algorithm.

Primality tests

- (200 BC) Sieve of Eratosthenes – essentially trial division
- (1976–1980) Miller–Rabin Probabilistic polynomial-time algorithm.
  - Used in practice!

Primality tests

- (200 BC) Sieve of Eratosthenes – essentially trial division
- (1976–1980) Miller–Rabin Probabilistic polynomial-time algorithm.
    - Used in practice!
    - Returns *no* for the number from the exercise.

Primality tests

- (200 BC) Sieve of Eratosthenes – essentially trial division
- (1976–1980) Miller–Rabin Probabilistic polynomial-time algorithm.
  - Used in practice!
  - Returns *no* for the number from the exercise.
- (2002) [Agrawal, Kayal, Saxena] The AKS Algorithm; The first deterministic polynomial-time algorithm.

Primality tests

- (200 BC) Sieve of Eratosthenes – essentially trial division
- (1976–1980) Miller–Rabin Probabilistic polynomial-time algorithm.
  - Used in practice!
  - Returns *no* for the number from the exercise.
- (2002) [Agrawal, Kayal, Saxena] The AKS Algorithm; The first deterministic polynomial-time algorithm.
  - Of theoretical value, not used in practice.

Primality tests

- (200 BC) Sieve of Eratosthenes – essentially trial division
- (1976–1980) Miller–Rabin Probabilistic polynomial-time algorithm.
  - Used in practice!
  - Returns *no* for the number from the exercise.
- (2002) [Agrawal, Kayal, Saxena] The AKS Algorithm; The first deterministic polynomial-time algorithm.
  - Of theoretical value, not used in practice.
- Polynomial-time algorithm $\approx$ fast $\approx$ feasible

Primality is a feasible property.

ZFC $\vdash$ Primality is a feasible property.

π : ZFC $\vdash$ Primality is a feasible property.

π : ZFC $\vdash$ Primality is a feasible property.

Problem

But is π itself feasible?

Complexity theory: What is feasibility?

- Complexity theory collects problems into complexity classes depending on what kind of 'algorithm' can solve them.

Complexity theory: What is feasibility?

- Complexity theory collects problems into complexity classes depending on what kind of ‘algorithm’ can solve them.

Theorem

P *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time algorithm.*

Complexity theory: What is feasibility?

- Complexity theory collects problems into complexity classes depending on what kind of 'algorithm' can solve them.

Theorem

P *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time algorithm.*

- $\textsc{OddNumbers}, \textsc{PerfectSquares}, \textsc{SolvableLinearSystems}, \textsc{SolvableDiophantineLinearSystems} \in \mathbf{P}$

Complexity theory: What is feasibility?

- Complexity theory collects problems into complexity classes depending on what kind of 'algorithm' can solve them.

Theorem

P *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time algorithm.*

- $\textsc{OddNumbers}, \textsc{PerfectSquares}, \textsc{SolvableLinearSystems}, \textsc{SolvableDiophantineLinearSystems} \in \mathbf{P}$
- AKS: Primes $\in$ **P**

Complexity theory: What is feasibility?

- Complexity theory collects problems into complexity classes depending on what kind of 'algorithm' can solve them.

Theorem

P *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time algorithm.*

- OddNumbers,PerfectSquares,SolvableLinearSystems,SolvableDiophantineLinearSystems $\in$ **P**
- AKS: Primes $\in$ **P**
- Cobham's Thesis: **P** is exactly the set of all feasible problems/properties.

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

- Nondeterministic algorithm: Can guess information leading to a 'yes' answer.

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

- Nondeterministic algorithm: Can guess information leading to a 'yes' answer.
- $\mathrm{SAT}, \mathrm{CLIQUE} \in \mathbf{NP}$

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

- Nondeterministic algorithm: Can guess information leading to a 'yes' answer.
- SAT, CLIQUE $\in$ **NP**
- Expected to differ from **P** and generally considered infeasible.

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

- Nondeterministic algorithm: Can guess information leading to a 'yes' answer.
- $\mathsf{SAT}, \mathsf{CLIQUE} \in \mathbf{NP}$
- Expected to differ from **P** and generally considered infeasible.

Theorem

coNP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time co-nondeterministic algorithm.*

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

- Nondeterministic algorithm: Can guess information leading to a 'yes' answer.
- SAT, CLIQUE $\in$ **NP**
- Expected to differ from **P** and generally considered infeasible.

Theorem

coNP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time co-nondeterministic algorithm.*

- Co-nondeterministic algorithm: Can guess information leading to a 'no' answer.

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

- Nondeterministic algorithm: Can guess information leading to a ‘yes’ answer.
- $\mathsf{SAT}, \mathsf{CLIQUE} \in$ **NP**
- Expected to differ from **P** and generally considered infeasible.

Theorem

coNP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time co-nondeterministic algorithm.*

- Co-nondeterministic algorithm: Can guess information leading to a ‘no’ answer.
- Primes $\in$ **coNP**

Complexity theory II

Theorem

NP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time nondeterministic algorithm.*

- Nondeterministic algorithm: Can guess information leading to a 'yes' answer.
- SAT, CLIQUE $\in$ **NP**
- Expected to differ from **P** and generally considered infeasible.

Theorem

coNP *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some polynomial time co-nondeterministic algorithm.*

- Co-nondeterministic algorithm: Can guess information leading to a 'no' answer.
- Primes $\in$ **coNP**
- Expected to differ from both **P** and **NP** and generally considered infeasible.

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

- Algorithm from the polynomial hierarchy: Can alternate between guessing information leading to ‘yes’ and ‘no’ finitely many times.

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

- Algorithm from the polynomial hierarchy: Can alternate between guessing information leading to 'yes' and 'no' finitely many times.
- The problem: [Can a given circuit be simplified?] $\in$ **PH**

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

- Algorithm from the polynomial hierarchy: Can alternate between guessing information leading to 'yes' and 'no' finitely many times.
- The problem: [Can a given circuit be simplified?] $\in$ **PH**
- **P**, **NP**, **coNP** $\subseteq$ **PH**, expected to be strict, considered infeasible

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

- Algorithm from the polynomial hierarchy: Can alternate between guessing information leading to 'yes' and 'no' finitely many times.
- The problem: [Can a given circuit be simplified?] $\in$ **PH**
- $\mathbf{P}, \mathbf{NP}, \mathbf{coNP} \subseteq \mathbf{PH}$, expected to be strict, considered infeasible
- Set $\leftrightarrow$ Property $\leftrightarrow$ Problem to decide if an input satisfies the property

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

- Algorithm from the polynomial hierarchy: Can alternate between guessing information leading to ‘yes’ and ‘no’ finitely many times.
- The problem: [Can a given circuit be simplified?] $\in$ **PH**
- $\mathbf{P}, \mathbf{NP}, \mathbf{coNP} \subseteq \mathbf{PH}$, expected to be strict, considered infeasible
- Set $\leftrightarrow$ Property $\leftrightarrow$ Problem to decide if an input satisfies the property
- Central question: Does $\mathbf{P} = \mathbf{NP}$?

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

- Algorithm from the polynomial hierarchy: Can alternate between guessing information leading to 'yes' and 'no' finitely many times.
- The problem: [Can a given circuit be simplified?] $\in$ **PH**
- $\mathbf{P}, \mathbf{NP}, \mathbf{coNP} \subseteq \mathbf{PH}$, expected to be strict, considered infeasible
- Set $\leftrightarrow$ Property $\leftrightarrow$ Problem to decide if an input satisfies the property
- Central question: Does $\mathbf{P} = \mathbf{NP}$?

Complexity theory III

Theorem

PH *is the set of all subsets of* $\mathbb{N}$ *which are recognized by some algorithm from the polynomial hierarchy.*

- Algorithm from the polynomial hierarchy: Can alternate between guessing information leading to 'yes' and 'no' finitely many times.
- The problem: [Can a given circuit be simplified?] $\in$ **PH**
- $\mathbf{P}, \mathbf{NP}, \mathbf{coNP} \subseteq \mathbf{PH}$, expected to be strict, considered infeasible
- Set $\leftrightarrow$ Property $\leftrightarrow$ Problem to decide if an input satisfies the property
- Central question: Does $\mathbf{P} = \mathbf{NP}$? That is, does 'magically' guessing positive information add power to short computations?

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity
- Nonexample:

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity
- Nonexample:
  - ZFC $\vdash$ "There is a function $f : \mathbb{N} \to \mathbb{N}$ which is not computed by *any algorithm*."

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity
- Nonexample:
  - ZFC $\vdash$ "There is a function $f : \mathbb{N} \to \mathbb{N}$ which is not computed by *any algorithm*."
  - While true, this function is not only infeasible, but truly uncomputable.

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity
- Nonexample:
  - ZFC $\vdash$ "There is a function $f : \mathbb{N} \to \mathbb{N}$ which is not computed by *any algorithm*."
  - While true, this function is not only infeasible, but truly uncomputable.
  - However, similar theorems can be used to prove a statement, which itself talks about feasible concepts.

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity
- Nonexample:
  - ZFC $\vdash$ "There is a function $f : \mathbb{N} \to \mathbb{N}$ which is not computed by *any algorithm*."
  - While true, this function is not only infeasible, but truly uncomputable.
  - However, similar theorems can be used to prove a statement, which itself talks about feasible concepts.
  - ZFC $\vdash$ Con(PA), that is ZFC $\vdash$ "finite mathematics is consistent".

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity
- Nonexample:
  - ZFC $\vdash$ "There is a function $f : \mathbb{N} \to \mathbb{N}$ which is not computed by *any algorithm*."
  - While true, this function is not only infeasible, but truly uncomputable.
  - However, similar theorems can be used to prove a statement, which itself talks about feasible concepts.
  - ZFC $\vdash$ Con(PA), that is ZFC $\vdash$ "finite mathematics is consistent".
  - But by Gödel's theorem, PA $\nvdash$ Con(PA).

Bounded arithmetic: Feasibility of proofs

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitively correspond to reasoning with a given complexity class.
- Some literature:
  - Krajíček (1995): Bounded Arithmetic, Propositional Logic and Complexity Theory
  - Cook and Nguyen (2010): Logical foundation of proof complexity
- Nonexample:
  - ZFC $\vdash$ "There is a function $f : \mathbb{N} \to \mathbb{N}$ which is not computed by *any algorithm*."
  - While true, this function is not only infeasible, but truly uncomputable.
  - However, similar theorems can be used to prove a statement, which itself talks about feasible concepts.
  - ZFC $\vdash$ Con(PA), that is ZFC $\vdash$ "finite mathematics is consistent".
  - But by Gödel's theorem, PA $\nvdash$ Con(PA).
  - So we can gain provability of feasible statements from infeasible ones.

Bounded arithmetic II

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitivelly correspond to reasoning with a given complexity class.

Bounded arithmetic II

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitivelly correspond to reasoning with a given complexity class.
- Cook's $\mathbf{PV}_1$ is a theory having function symbols for every polynomial-time algorithm, and induction for every polynomial-time property. That is, if $p \in \mathbf{P}$, then there is an axiom

$$[(p(0) \wedge (\forall x)(p(x) \rightarrow p(x+1))) \rightarrow (\forall x)(p(x))] \in \mathbf{PV}_1.$$

Bounded arithmetic II

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitivelly correspond to reasoning with a given complexity class.
- Cook's $\mathbf{PV}_1$ is a theory having function symbols for every polynomial-time algorithm, and induction for every polynomial-time property. That is, if $p \in \mathbf{P}$, then there is an axiom

$$[(p(0) \wedge (\forall x)(p(x) \rightarrow p(x+1))) \rightarrow (\forall x)(p(x))] \in \mathbf{PV}_1.$$

Bounded arithmetic II

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitivelly correspond to reasoning with a given complexity class.
- Cook's $\mathbf{PV}_1$ is a theory having function symbols for every polynomial-time algorithm, and induction for every polynomial-time property. That is, if $p \in \mathbf{P}$, then there is an axiom

$$[(p(0) \wedge (\forall x)(p(x) \rightarrow p(x+1))) \rightarrow (\forall x)(p(x))] \in \mathbf{PV}_1.$$

Theorem

If for $p \in \mathbf{P}$:

$$\mathbf{PV}_1 \vdash (\forall x)(\exists y)p(x,y),$$

then there is a function computable in polynomial-time function f computing for each x a y such that $p(x, f(x))$.

Bounded arithmetic II

- The field of bounded arithmetic describes axiomatizations of mathematics which intuitivelly correspond to reasoning with a given complexity class.
- Cook's $\mathbf{PV}_1$ is a theory having function symbols for every polynomial-time algorithm, and induction for every polynomial-time property. That is, if $p \in \mathbf{P}$, then there is an axiom

$$[(p(0) \wedge (\forall x)(p(x) \rightarrow p(x+1))) \rightarrow (\forall x)(p(x))] \in \mathbf{PV}_1.$$

Theorem

If for $p \in \mathbf{P}$:

$$\mathbf{PV}_1 \vdash (\forall x)(\exists y)p(x, y),$$

then there is a function computable in polynomial-time function f computing for each x a y such that $p(x, f(x))$.

- There are many other bounded arithmetic theories for different complexity classes.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity
  - It's not hard to prove that $\mathbf{NP} \neq \mathbf{coNP} \implies \mathbf{P} \neq \mathbf{NP}$.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity
  - It's not hard to prove that $\mathbf{NP} \neq \mathbf{coNP} \implies \mathbf{P} \neq \mathbf{NP}$.
  - "If guessing negative and positive information have different powers then guessing positive information adds power."

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity
  - It's not hard to prove that $\mathbf{NP} \neq \mathbf{coNP} \implies \mathbf{P} \neq \mathbf{NP}$.
  - "If guessing negative and positive information have different powers then guessing positive information adds power."
  - Showing (strong) unprovability in a bounded arithmetic gives lower bounds for *propositional* proof systems.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity
  - It's not hard to prove that $\mathbf{NP} \neq \mathbf{coNP} \implies \mathbf{P} \neq \mathbf{NP}$.
  - "If guessing negative and positive information have different powers then guessing positive information adds power."
  - Showing (strong) unprovability in a bounded arithmetic gives lower bounds for *propositional* proof systems.
  - Lower bounds for all p.p.s. $\implies \mathbf{NP} \neq \mathbf{coNP}$.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity
  - It's not hard to prove that $\mathbf{NP} \neq \mathbf{coNP} \implies \mathbf{P} \neq \mathbf{NP}$.
  - "If guessing negative and positive information have different powers then guessing positive information adds power."
  - Showing (strong) unprovability in a bounded arithmetic gives lower bounds for *propositional* proof systems.
  - Lower bounds for all p.p.s. $\implies \mathbf{NP} \neq \mathbf{coNP}$.
- Complexity: If $\mathbf{PV}_1 \vdash$ "the AKS algorithm is correct", then factoring integers is easy.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity
  - It's not hard to prove that $\mathbf{NP} \neq \mathbf{coNP} \implies \mathbf{P} \neq \mathbf{NP}$.
  - "If guessing negative and positive information have different powers then guessing positive information adds power."
  - Showing (strong) unprovability in a bounded arithmetic gives lower bounds for *propositional* proof systems.
  - Lower bounds for all p.p.s. $\implies \mathbf{NP} \neq \mathbf{coNP}$.
- Complexity: If $\mathbf{PV}_1 \vdash$ "the AKS algorithm is correct", then factoring integers is easy.

What do we gain by studying feasible proofs?

- Bounded reverse mathematics:
  - Understanding the strength of reasoning with a concrete complexity class, and relationship of different axioms.
- Meta-complexity
  - $\mathbf{PV}_1$ can formulate a large part of contemporary complexity theory.
  - Is it possible to show that $\mathbf{PV}_1 \nvdash \mathbf{P} = \mathbf{NP}$?
  - This would imply the existence of a structure $\mathbb{M} \models \mathbf{PV}_1$ which behaves very much like $\mathbb{N}$ but $\mathbb{M} \models [\mathbf{P} \neq \mathbf{NP}]$.
  - Possibly easier than $\mathbf{P} \neq \mathbf{NP}$ but still wide open!
- Proof complexity
  - It's not hard to prove that $\mathbf{NP} \neq \mathbf{coNP} \implies \mathbf{P} \neq \mathbf{NP}$.
  - “If guessing negative and positive information have different powers then guessing positive information adds power.”
  - Showing (strong) unprovability in a bounded arithmetic gives lower bounds for *propositional* proof systems.
  - Lower bounds for all p.p.s. $\implies \mathbf{NP} \neq \mathbf{coNP}$.
- Complexity: If $\mathbf{PV}_1 \vdash$ “the AKS algorithm is correct”, then factoring integers is easy. Then cryptography is broken.

AKS and the generalized Fermat's Little Theorem

Theorem

If $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ *and* $\gcd(a, n) = 1$, *then*

$$n \text{ is a prime} \iff (X + a)^n \equiv X^n + a \pmod{n}.$$

AKS and the generalized Fermat's Little Theorem

Theorem

If $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ *and* $\gcd(a, n) = 1$, *then*

$$n \text{ is a prime } \iff (X + a)^n \equiv X^n + a \pmod{n}.$$

- Takes too long!

AKS and the generalized Fermat's Little Theorem

Theorem

If $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ *and* $\gcd(a, n) = 1$, *then*

$$n \text{ is a prime } \iff (X + a)^n \equiv X^n + a \,(mod\ n).$$

- Takes too long!
- AKS show that:

AKS and the generalized Fermat's Little Theorem

Theorem

If $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ *and* $\gcd(a, n) = 1$, *then*

$$n \text{ is a prime} \iff (X + a)^n \equiv X^n + a \pmod{n}.$$

- Takes too long!
- AKS show that:
  - If we find r such that $\mathrm{ord}_r(n) > \log^2(n)$ and for enough a:

$$(X + a)^n \equiv X^n + a \pmod{n, X^r - 1},$$

AKS and the generalized Fermat's Little Theorem

Theorem

If $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ *and* $\gcd(a, n) = 1$, *then*

$$n \text{ is a prime} \iff (X + a)^n \equiv X^n + a \pmod{n}.$$

- Takes too long!
- AKS show that:
  - If we find r such that $\mathrm{ord}_r(n) > \log^2(n)$ and for enough a:

$$(X + a)^n \equiv X^n + a \pmod{n, X^r - 1},$$

AKS and the generalized Fermat's Little Theorem

Theorem

If $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ *and* $\gcd(a, n) = 1$, *then*

$$n \text{ is a prime} \iff (X + a)^n \equiv X^n + a \pmod{n}.$$

- Takes too long!
- AKS show that:
  - If we find r such that $\operatorname{ord}_r(n) > \log^2(n)$ and for enough a:

$$(X + a)^n \equiv X^n + a \,(\text{mod } n, X^r - 1),$$

then n is a power of a prime.

AKS and the generalized Fermat's Little Theorem

Theorem

If $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ *and* $\gcd(a, n) = 1$, *then*

$$n \text{ is a prime} \iff (X + a)^n \equiv X^n + a \,(mod\ n).$$

- Takes too long!
- AKS show that:
  - If we find r such that $\mathrm{ord}_r(n) > \log^2(n)$ and for enough a:

    $$(X + a)^n \equiv X^n + a \pmod{n, X^r - 1},$$

    then n is a power of a prime.
- The proof mostly involves elementary results about finite fields

The AKS algorithm

```
Input:  integer n > 1.
1.  If (n = a^b for a ∈ N and b > 1), output COMPOSITE.
2.  Find the smallest r such that o_r(n) > log^2 n.
3.  If 1 < (a,n) < n for some a ≤ r, output COMPOSITE.
4.  If n ≤ r, output PRIME.[1]
5.  For a = 1 to ⌊sqrt(φ(r)) log n⌋ do
        if ((X + a)^n ≠ X^n + a (mod X^r − 1, n)), output COMPOSITE;
6.  Output PRIME;
```

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.
- S^1_2 is a stronger theory of bounded arithmetic, which includes 'short **NP** induction'.

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.
- S^1_2 is a stronger theory of bounded arithmetic, which includes 'short **NP** induction'.

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.
- S_2^1 is a stronger theory of bounded arithmetic, which includes 'short **NP** induction'. Still would result in a polynomial-time factoring algorithm.

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.
- S^1_2 is a stronger theory of bounded arithmetic, which includes 'short **NP** induction'. Still would result in a polynomial-time factoring algorithm.
- The axiom $iWPHP(\mathbf{PV})$ is a well-studied axiom stating non-existence of polynomial-time injective function $f : \{1, \ldots, 2n\} \rightarrow \{1, \ldots, n\}$.

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.
- S^1_2 is a stronger theory of bounded arithmetic, which includes 'short **NP** induction'. Still would result in a polynomial-time factoring algorithm.
- The axiom $iWPHP(\mathbf{PV})$ is a well-studied axiom stating non-existence of polynomial-time injective function $f : \{1, \ldots, 2n\} \to \{1, \ldots, n\}$.

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.
- S^1_2 is a stronger theory of bounded arithmetic, which includes 'short **NP** induction'. Still would result in a polynomial-time factoring algorithm.
- The axiom $iWPHP(\mathbf{PV})$ is a well-studied axiom stating non-existence of polynomial-time injective function $f : \{1, \ldots, 2n\} \to \{1, \ldots, n\}$. Unprovable in both $\mathbf{PV}_1$ and S^1_2 unless cryptographic hashes can be broken.

Our work

- Motivating question: What is the weakest possible theory of bounded arithmetic, which can prove the correctness of the AKS algorithm?
- $\mathbf{PV}_1$-formalization of the correctness ought to be hard or impossible – it gives polynomial-time factoring algorithm.
- S^1_2 is a stronger theory of bounded arithmetic, which includes 'short **NP** induction'. Still would result in a polynomial-time factoring algorithm.
- The axiom $iWPHP(\mathbf{PV})$ is a well-studied axiom stating non-existence of polynomial-time injective function $f : \{1, \ldots, 2n\} \to \{1, \ldots, n\}$. Unprovable in both $\mathbf{PV}_1$ and S^1_2 unless cryptographic hashes can be broken.
- $S^1_2 + iWPHP(\mathbf{PV})$ is likely still not enough to naturally formalize the original proof of correctness — we introduce two new algebraic axioms, such that with their addition the original proof can be formalized.

Our work II – New algebraic axioms, GFLT

- The simpler axiom to state is at the heart of the AKS algorithm:

Definition (Generalized Fermat's little Theorem)

Let p be a prime and f a polynomial coded by a sequence of coefficients of length equal to its degree. Then for every $a \leq p$ we have:

$$(X + a)^p \equiv X^p + a \,(mod\ p, f).$$

Where the exponentiation is computed by iterated squaring modulo f.

Our work III – New algebraic axioms, DLB

- The other axiom prohibits polynomials coded by a list of monomials to have more roots than its degree.

Our work III – New algebraic axioms, DLB

- The other axiom prohibits polynomials coded by a list of monomials to have more roots than its degree.
- As sets of this size cannot be enumerated in S^1_2, then we claim a new function symbol ι provides an injective function from the roots

Our work III – New algebraic axioms, DLB

- The other axiom prohibits polynomials coded by a list of monomials to have more roots than its degree.
- As sets of this size cannot be enumerated in S^1_2, then we claim a new function symbol ι provides an injective function from the roots

Our work III – New algebraic axioms, DLB

- The other axiom prohibits polynomials coded by a list of monomials to have more roots than its degree.
- As sets of this size cannot be enumerated in S^1_2, then we claim a new function symbol ι provides an injective function from the roots

Definition (Degree lower bound)

Let F be a finite field coded by a tuple of boolean circuits computing its operations and $f \in F[X]$ a polynomial coded by a list of monomials. Then the function $\iota(F, f, -)$ is an injective map:

$$\iota(F, f, -) : \{F\text{- roots of } f\} \to \{1, \ldots, \deg f\}.$$

Our work III – New algebraic axioms, DLB

- The other axiom prohibits polynomials coded by a list of monomials to have more roots than its degree.
- As sets of this size cannot be enumerated in S^1_2, then we claim a new function symbol ι provides an injective function from the roots

Definition (Degree lower bound)

Let F be a finite field coded by a tuple of boolean circuits computing its operations and $f \in F[X]$ a polynomial coded by a list of monomials. Then the function $\iota(F, f, -)$ is an injective map:

$$\iota(F, f, -) : \{F\text{- roots of } f\} \to \{1, \ldots, \deg f\}.$$

- The function symbol ι is then allowed to appear in the induction of S^1_2 and in the *iWPHP* instances.

Our work IV – The Main Theorem

Definition

We define the sentence AKSCorrect as

$$(\forall x)(AKSPrime(x) \leftrightarrow Prime(x)).$$

Our work IV – The Main Theorem

Definition

We define the sentence AKSCorrect as

$$(\forall x)(AKSPrime(x) \leftrightarrow Prime(x)).$$

Theorem

$$S^1_2 + iWPHP + DLB + GFLT \vdash AKSCorrect$$

Our work IV – The Main Theorem

Definition

We define the sentence AKSCorrect as

$$(\forall x)(AKSPrime(x) \leftrightarrow Prime(x)).$$

Theorem

$$S^1_2 + iWPHP + DLB + GFLT \vdash AKSCorrect$$

Lemma

$$VTC^0_2 \vdash \iota\text{-free consequences of } S^1_2 + iWPHP + DLB + GFLT$$

Our work IV – The Main Theorem

Definition

We define the sentence AKSCorrect as

$$(\forall x)(AKSPrime(x) \leftrightarrow Prime(x)).$$

Theorem

$$S^1_2 + iWPHP + DLB + GFLT \vdash AKSCorrect$$

Lemma

$$VTC^0_2 \vdash \iota\text{-free consequences of } S^1_2 + iWPHP + DLB + GFLT$$

Corollary (Main Theorem)

$$VTC^0_2 \vdash AKSCorrect$$

Overview of our results

Theory	Axioms	Theorems
VTC^0_2	**PH** induction and counting	Division of large polynomials the DLB axiom the GFLT axiom AKSCorrect
S^1_2	short **NP** induction	$2^{\lfloor m/2 \rfloor} \leq \mathrm{lcm}(1, \ldots, m)$ Cyclotomic extensions
$\mathbf{PV}_1$	**P** induction	Legendre's formula

Problems

Problem

Can this be improved? Can we discard the counting and just use the strong pigeonhole principle? That is, does

$$T_2 + PHP \vdash AKSCorrect?$$

Problems

Problem

Can this be improved? Can we discard the counting and just use the strong pigeonhole principle? That is, does

$$T_2 + PHP \vdash AKSCorrect?$$

Problem

Can we show DLB and GFLT are hard for $\mathbf{PV}_1$ *or* S^1_2 *under some hardness assumptions?*

$VTC^0_2 \vdash$ "Thank you for your attention!"