Mechanisation of AKS Algorithm

Part 1 — the Main Theorem

Hing-Lun Chan[1] Michael Norrish[2]

joseph.chan@anu.edu.au
Australian National University

Michael.Norrish@nicta.com.au
Canberra Research Lab., NICTA;
also, Australian National University

ITP 2015, Nanjing, China

Basic Primality Test

Theorem (Basic Primality Test.)

$$\vdash 1 < n \wedge \gcd(c, n) = 1 \Rightarrow (\text{prime } n \iff (X + \boldsymbol{c})^n \equiv (X^n + \boldsymbol{c}) \pmod{n}))$$

Proof.

($\Rightarrow$) Apply Freshman's Theorem:

$\text{prime } n \wedge \text{poly p} \wedge \text{poly q} \Rightarrow (\text{p} + \text{q})^n \equiv \text{p}^n + \text{q}^n \pmod{n}$

by identifying p as $\boldsymbol{X}$ and q as $\boldsymbol{c}$,

then apply Fermat's Little Theorem: $\text{prime } n \Rightarrow \boldsymbol{c}^n \equiv \boldsymbol{c} \pmod{n}$

□

Basic Primality Test

Theorem (Basic Primality Test.)

$$\vdash 1 < n \wedge \gcd(c, n) = 1 \Rightarrow$$
$$(\text{prime } n \iff (\mathrm{X} + \boldsymbol{c})^n \equiv (\mathrm{X}^n + \boldsymbol{c}) \pmod{n})$$

Proof.

($\Rightarrow$) Apply Freshman's Theorem:

$\text{prime } n \wedge \text{poly p} \wedge \text{poly q} \Rightarrow (\mathrm{p} + \mathrm{q})^n \equiv \mathrm{p}^n + \mathrm{q}^n \pmod{n}$

by identifying p as $\boldsymbol{X}$ and q as $\boldsymbol{c}$,

then apply Fermat's Little Theorem: $\text{prime } n \Rightarrow \boldsymbol{c}^n \equiv \boldsymbol{c} \pmod{n}$

($\Leftarrow$) Can be shown by properties of binomial coefficients. □

Making a Primality Test

Theorem (Basic Primality Test.)

$\vdash 1 < n \land \gcd(c, n) = 1 \Rightarrow$
$(\text{prime } n \iff (X + \boldsymbol{c})^n \equiv (X^n + \boldsymbol{c}) \pmod{n})$

Given a number $n > 1$, is n a prime?

- Since $\gcd(1, n) = 1$, pick $c = 1$, then this theorem applies.
- Perform one Freshman-Fermat identity check in $\mathbb{Z}_n$, *i.e.*, prime $n \iff (\boldsymbol{X} + \boldsymbol{1})^n \equiv \boldsymbol{X}^n + \boldsymbol{1} \pmod{n}$.

This is a deterministic primality test, however:

Making a Primality Test

Theorem (Basic Primality Test.)

$\vdash 1 < n \wedge \gcd(c, n) = 1 \Rightarrow$
$(\text{prime } n \iff (X + \mathbf{c})^n \equiv (X^n + \mathbf{c}) \pmod{n})$

Given a number $n > 1$, is n a prime?

- Since $\gcd(1, n) = 1$, pick $c = 1$, then this theorem applies.
- Perform one Freshman-Fermat identity check in $\mathbb{Z}_n$, *i.e.*, prime $n \iff (\boldsymbol{X} + \mathbf{1})^n \equiv \boldsymbol{X}^n + \mathbf{1} \pmod{n}$.

This is a deterministic primality test, however:

- upon binomial expansion, perform about n checks in mod n.
- worse than simple trial divisions with $\sqrt{n}$ checks!

Making a Primality Test

Theorem (Basic Primality Test.)

$\vdash 1 < n \land \gcd(c, n) = 1 \Rightarrow$
$(\text{prime } n \iff (X + \boldsymbol{c})^n \equiv (X^n + \boldsymbol{c}) \pmod n))$

Given a number $n > 1$, is n a prime?

- Since $\gcd(1, n) = 1$, pick $c = 1$, then this theorem applies.
- Perform one Freshman-Fermat identity check in $\mathbb{Z}_n$, *i.e.*, prime $n \iff (\boldsymbol{X} + \mathbf{1})^n \equiv \boldsymbol{X}^n + \mathbf{1} \pmod n$.

This is a deterministic primality test, however:

- upon binomial expansion, perform about n checks in mod n.
- worse than simple trial divisions with $\sqrt{n}$ checks!

Enter Manindra Agrawal, Neeraj Kayal, and Nitin Saxena (AKS).

AKS twists

The AKS team modifies the Freshman-Fermat identity check:

- Perform the polynomial identity checks in $\pmod{n,\ \boldsymbol{X}^k - \mathbf{1}}$ for some suitably chosen k.
- Check a range of coprime values c, for $0 < c \leq \ell$, up to some maximum limit ℓ.

AKS twists

The AKS team modifies the Freshman-Fermat identity check:

- Perform the polynomial identity checks in $\pmod{n,\ \boldsymbol{X}^k - \mathbf{1}}$ for some suitably chosen k. Remainder has only up to k terms.
- Check a range of coprime values c, for $0 < c \leq \ell$, up to some maximum limit ℓ. Provide more ways to weed out composites.

AKS twists

The AKS team modifies the Freshman-Fermat identity check:

- Perform the polynomial identity checks in (mod n, $\boldsymbol{X}^k - \mathbf{1}$) for some suitably chosen k. Remainder has only up to k terms.
- Check a range of coprime values c, for $0 < c \leq \ell$, up to some maximum limit ℓ. Provide more ways to weed out composites.

The AKS choice of parameters k and ℓ:

- $\text{order}_k(n) \geq (2(\log n + 1))^2$
- $\ell = 2\sqrt{k}(\log n + 1)$

AKS twists

The AKS team modifies the Freshman-Fermat identity check:

- Perform the polynomial identity checks in $(\text{mod } n,\ \boldsymbol{X}^k - \mathbf{1})$ for some suitably chosen k. Remainder has only up to k terms.
- Check a range of coprime values c, for $0 < c \leq \ell$, up to some maximum limit ℓ. Provide more ways to weed out composites.

The AKS choice of parameters k and ℓ:

- $\text{order}_k(n) \geq (2(\log n + 1))^2$,*i.e.*, search for k given n.
- $\ell = 2\sqrt{k}(\log n + 1)$,*i.e.*, compute ℓ from k and n.

AKS twists

The AKS team modifies the Freshman-Fermat identity check:

- Perform the polynomial identity checks in $(\text{mod } n,\ \boldsymbol{X}^k - \mathbf{1})$ for some suitably chosen k. Remainder has only up to k terms.
- Check a range of coprime values c, for $0 < c \leq \ell$, up to some maximum limit ℓ. Provide more ways to weed out composites.

The AKS choice of parameters k and ℓ:

- $\text{order}_k(n) \geq (2(\log n + 1))^2$, *i.e.*, search for k given n.
- $\ell = 2\sqrt{k}(\log n + 1)$, *i.e.*, compute ℓ from k and n.

The AKS result:

- Modifications $\Rightarrow n = p^e$ where prime $p \mid n$ for some exponent e.
- Include a power check: if n is power free, then n must be prime.

The AKS Main Theorem

The AKS primality test is based on the following theorem:

Theorem (The AKS Main Theorem.)

$$
\begin{array}{l}
\vdash \text{prime } n \iff \\
\quad 1 < n \land \text{power\_free } n \land \\
\quad \exists k. \\
\qquad \text{prime } k \land (2(\log n + 1))^2 \le \text{order}_k(n) \land \\
\qquad (\forall j.\ 0 < j \land j \le k \land j < n \Rightarrow \gcd(n,j) = 1) \land \\
\qquad (k < n \Rightarrow \\
\qquad\quad \forall c. \\
\qquad\qquad 0 < c \land c \le 2\sqrt{k}(\log n + 1) \Rightarrow \\
\qquad\qquad\quad (\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n,\ \boldsymbol{X}^k - 1})
\end{array}
$$

The AKS Main Theorem

The AKS primality test is based on the following theorem:

Theorem (The AKS Main Theorem.)

$\vdash$ prime n $\iff$
  $1 < n \land$ power_free n $\land$
  $\exists k.$
    prime k $\land$ $(2(\log n + 1))^2 \le \mathrm{order}_k(n)$ $\land$
    $(\forall j.\ 0 < j \land j \le k \land j < n \Rightarrow \gcd(n,j) = 1)$ $\land$
    $(k < n \Rightarrow$
      $\forall c.$
        $0 < c \land c \le 2\sqrt{k}(\log n + 1) \Rightarrow$
          $(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n,\ \boldsymbol{X}^k - 1})$

Note the checks: power free, GCD tests and Polynomial modulo tests.

The AKS Algorithm

The AKS Main theorem corresponds to the following algorithm:

Input: integer $n > 1$.

1. If ($n = b^m$ for some base b with $m > 1$), return `COMPOSITE`.
2. Search for a prime k satisfying $\text{order}_k(n) \geq (2(\log n + 1))^2$.
3. For each ($j = 1$ to k) if ($j = n$) break, else if ($\gcd(j, n) \neq 1$), return `COMPOSITE`.
4. If ($k \geq n$), return `PRIME`.
5. For each ($c = 1$ to ℓ) where $\ell = 2\sqrt{k}(\log n + 1)$, if $(\boldsymbol{X} + \boldsymbol{c})^n \not\equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n, \boldsymbol{X}^k - 1}$, return `COMPOSITE`.
6. return `PRIME`.

Mechanisation of AKS Algorithm

AKS Algorithm

- Found in August 2002 by the team: Agrawal, Kayal and Saxena.
- The first deterministic polynomial-time primality-testing algorithm.
- Correctness of algorithm relies on a new theorem in finite fields.

Mechanisation of AKS Algorithm

AKS Algorithm

- Found in August 2002 by the team: Agrawal, Kayal and Saxena.
- The first deterministic polynomial-time primality-testing algorithm.
- Correctness of algorithm relies on a new theorem in finite fields.

Its Mechanisation

- Show the correctness of AKS Main Theorem in a theorem-prover.
- Establish upper bounds on the parameters of AKS algorithm.
- Demonstrate that AKS algorithm is indeed polynomial-time.

Mechanisation of AKS Algorithm

AKS Algorithm

- Found in August 2002 by the team: Agrawal, Kayal and Saxena.
- The first deterministic polynomial-time primality-testing algorithm.
- Correctness of algorithm relies on a new theorem in finite fields.

Its Mechanisation

- Show the correctness of AKS Main Theorem in a theorem-prover.
- Establish upper bounds on the parameters of AKS algorithm.
- Demonstrate that AKS algorithm is indeed polynomial-time.

These correspond to Part 1, Part 2, and Part 3 of our work.

Easy and Hard

The if-part ($\Rightarrow$) is easy:

- prime $n \Rightarrow \gcd(c, n) = 1$ for $1 \leq c \leq \ell$, when $\ell < n$.
- Thus the if-part of Basic Primality Test applies, *i.e.*,
  $(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n}$ for $1 \leq c \leq \ell$.
- or, $(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n,\ \boldsymbol{X}^k - 1}$ for $1 \leq c \leq \ell$.

Easy and Hard

The if-part ($\Rightarrow$) is easy:

- prime $n \Rightarrow \gcd(c, n) = 1$ for $1 \leq c \leq \ell$, when $\ell < n$.
- Thus the if-part of Basic Primality Test applies, *i.e.*,
  $(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n}$ for $1 \leq c \leq \ell$.
- or, $(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n, \boldsymbol{X}^k - 1}$ for $1 \leq c \leq \ell$.

The only-if part ($\Leftarrow$) is hard:

- There is not much we can assert for a general $n > 1$.
- Except that $n > 1$ must have a prime factor p that divides n.
- AKS modifications $\Rightarrow n = p^e$ where prime $p \mid n$ for some e.
- This is shown by a wonderful idea discovered by the AKS team.
- Once this is shown, power free check ensures $n = p$, a prime.

Introspective Relation

AKS polynomial identity checks involve double moduli:

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n,\ \boldsymbol{X}^k - 1}$$

Introspective Relation

AKS polynomial identity checks involve double moduli:

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n,\ \boldsymbol{X}^k - 1}$$

In the context of $\mathbb{Z}_n$, which is a ring for a general n:

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv \boldsymbol{X}^n + \boldsymbol{c} \pmod{\boldsymbol{X}^k - \mathbf{1}}$$

Introspective Relation

AKS polynomial identity checks involve double moduli:

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n, \boldsymbol{X}^k - 1}$$

In the context of $\mathbb{Z}_n$, which is a ring for a general n:

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv \boldsymbol{X}^n + \boldsymbol{c} \pmod{\boldsymbol{X}^k - \mathbf{1}}$$

Rewriting with polynomial substitution, for a general ring $\mathcal{R}$:

$$(\boldsymbol{X} + \boldsymbol{c})^n[\boldsymbol{X}] \equiv (\boldsymbol{X} + \boldsymbol{c})[\boldsymbol{X}^n] \pmod{\boldsymbol{X}^k - \mathbf{1}}$$

Introspective Relation

AKS polynomial identity checks involve double moduli:

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv (\boldsymbol{X}^n + \boldsymbol{c}) \pmod{n,\ \boldsymbol{X}^k - 1}$$

In the context of $\mathbb{Z}_n$, which is a ring for a general n:

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv \boldsymbol{X}^n + \boldsymbol{c} \pmod{\boldsymbol{X}^k - \mathbf{1}}$$

Rewriting with polynomial substitution, for a general ring $\mathcal{R}$:

$$(\boldsymbol{X} + \boldsymbol{c})^n[\boldsymbol{X}] \equiv (\boldsymbol{X} + \boldsymbol{c})[\boldsymbol{X}^n] \pmod{\boldsymbol{X}^k - \mathbf{1}}$$

Denote $n \bowtie \mathrm{p}$: n is *introspective* to polynomial p, when:

$$\vdash n \bowtie \mathrm{p} \iff \mathrm{poly}\ \mathrm{p} \land 0 < k \land \mathrm{p}^n \equiv \mathrm{p}[\boldsymbol{X}^n] \pmod{\boldsymbol{X}^k - \mathbf{1}}$$

AKS Idea

The hard-part of AKS Main Theorem is proved by:

- Shifting the perspective from $\mathbb{Z}_n$ to $\mathbb{Z}_p$, where prime $p \mid n$.
- Showing that introspective relationship is preserved:

$$n \bowtie_{\mathbb{Z}_n} \boldsymbol{X} + \boldsymbol{c} \quad \Rightarrow \quad n \bowtie_{\mathbb{Z}_p} \boldsymbol{X} + \boldsymbol{c}$$

- Freshman-Fermat provides for free: $p \bowtie_{\mathbb{Z}_p} \boldsymbol{X} + \boldsymbol{c}$
- Construct sets based on these pair of introspective identities.
- Note that $\boldsymbol{X}^k - \mathbf{1} \in \mathcal{F}[X]$ has a monic irreducible factor $\boldsymbol{h}$.
- Construct finite sets based on parameter k and irreducible $\boldsymbol{h}$.
- By the choices of k and ℓ, squeeze out: $n = p^e$ for some e.

AKS Introspective Game

number $\bowtie$ polynomial

$(\boldsymbol{X} + \boldsymbol{c})^n \equiv \boldsymbol{X}^n + \boldsymbol{c} \pmod{\boldsymbol{X}^k - \mathbf{1}}$

$$n \bowtie (\boldsymbol{X} + \boldsymbol{c})$$
$$p \bowtie (\boldsymbol{X} + \boldsymbol{c})$$
$$0 < c \leq \ell$$

AKS Introspective Game

number ⋈ polynomial

$$(\boldsymbol{X} + \boldsymbol{c})^n \equiv \boldsymbol{X}^n + \boldsymbol{c} \pmod{\boldsymbol{X}^k - \mathbf{1}}$$

$\mathcal{N}$: Numbers

$$\mathcal{N} = \{m \mid \gcd(m, k) = 1 \wedge \forall\, 0 < c \le \ell, m \bowtie \boldsymbol{X} + \boldsymbol{c}\}$$

$$n \in \mathcal{N}, p \in \mathcal{N}, 1 \in \mathcal{N}.$$

$n \bowtie (\boldsymbol{X} + \boldsymbol{c})$

$p \bowtie (\boldsymbol{X} + \boldsymbol{c})$

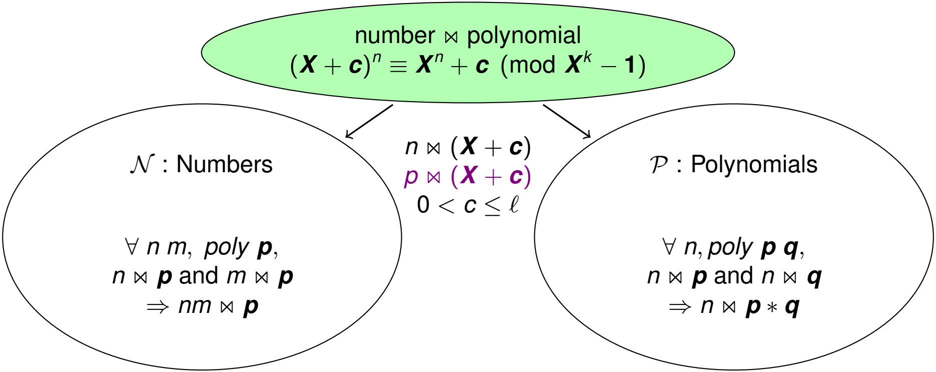
$0 < c \le \ell$

$\mathcal{P}$: Polynomials

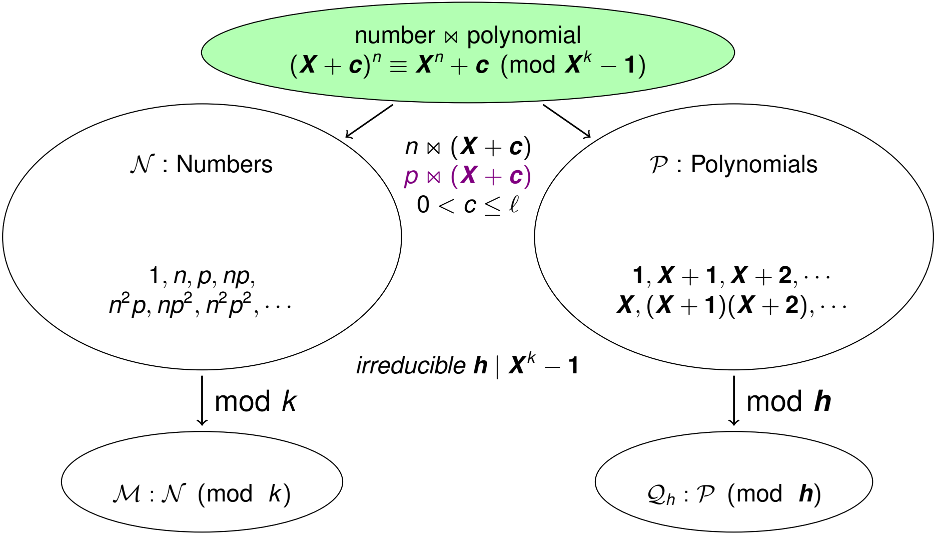
$$\mathcal{P} = \{poly\ \boldsymbol{p} \mid \forall\, m \in \mathcal{N}, m \bowtie \boldsymbol{p}\}$$

$$\forall\, 0 < c \le \ell, \boldsymbol{X} + \boldsymbol{c} \in \mathcal{P},$$
$$\mathbf{1} \in \mathcal{P}.$$

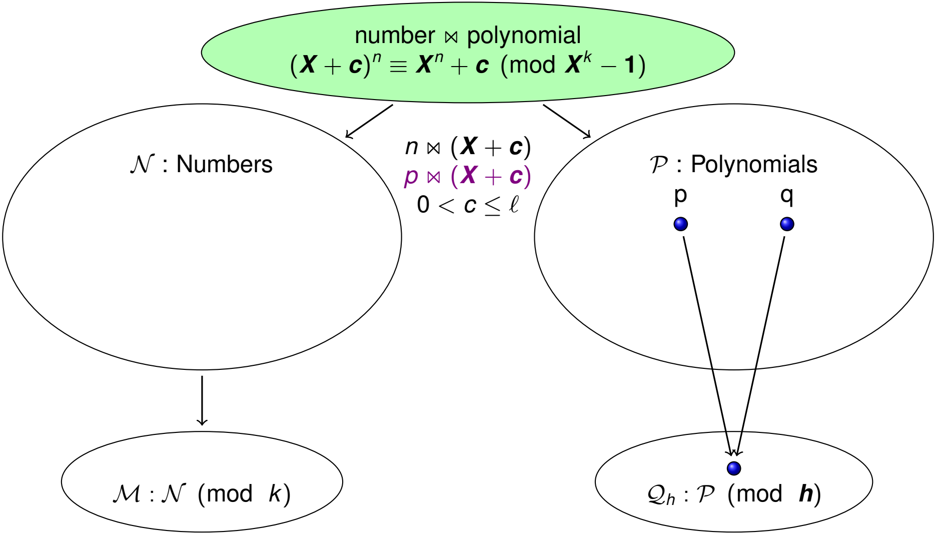
AKS Introspective Game



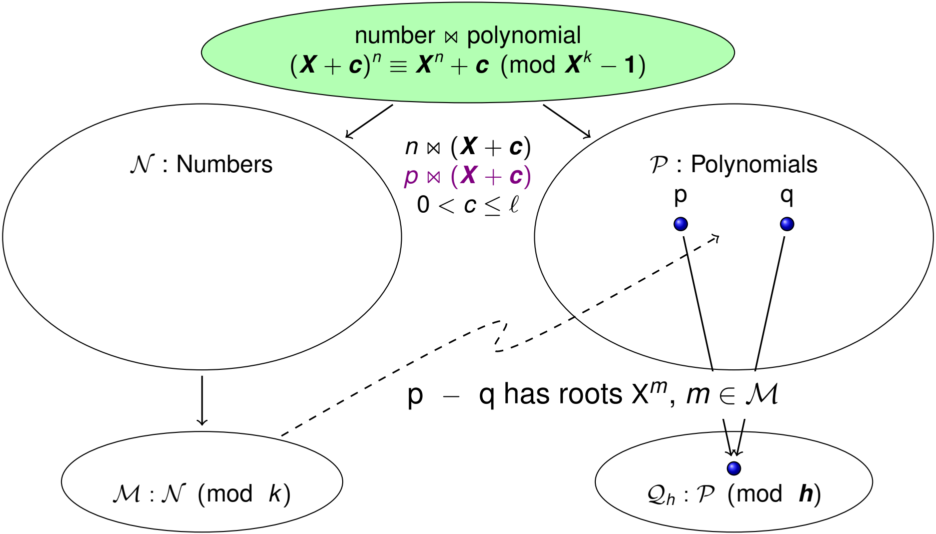
AKS Introspective Game



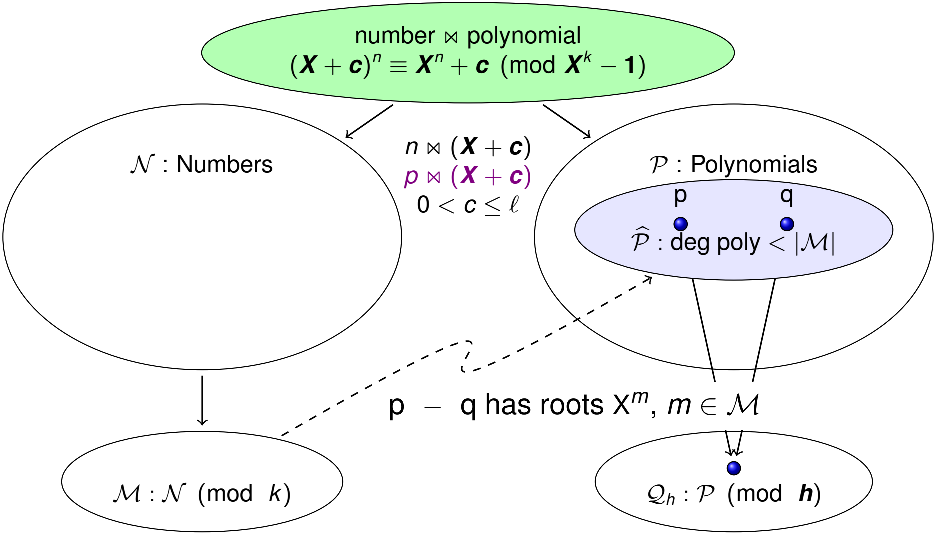
AKS Introspective Game



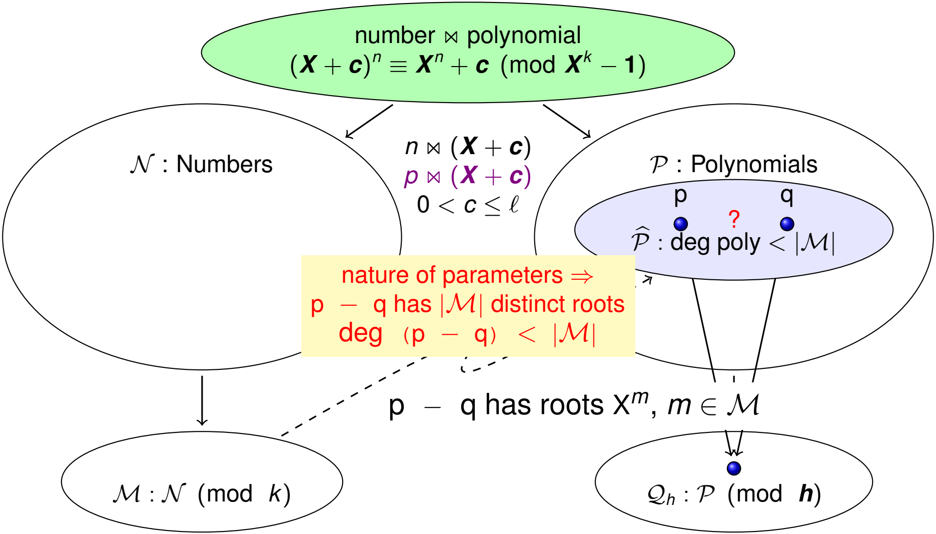
AKS Introspective Game



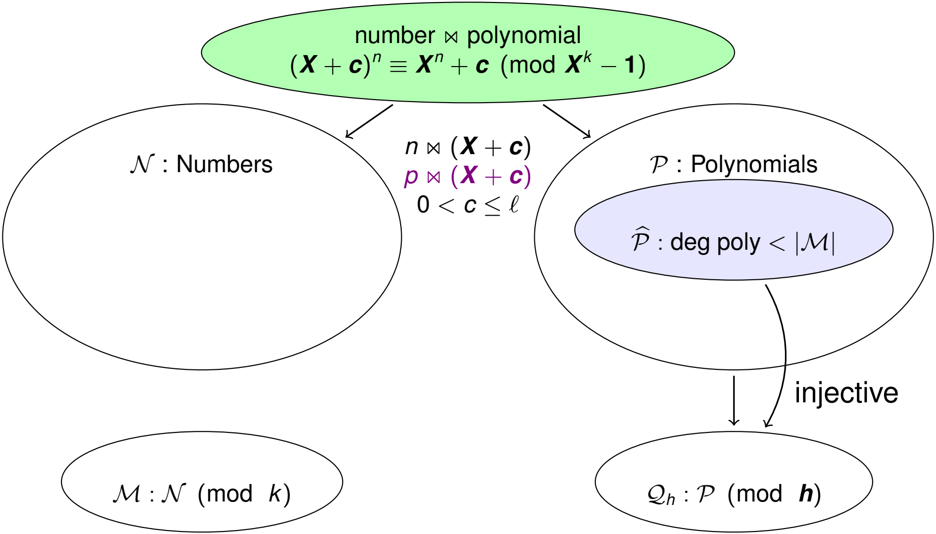
AKS Introspective Game



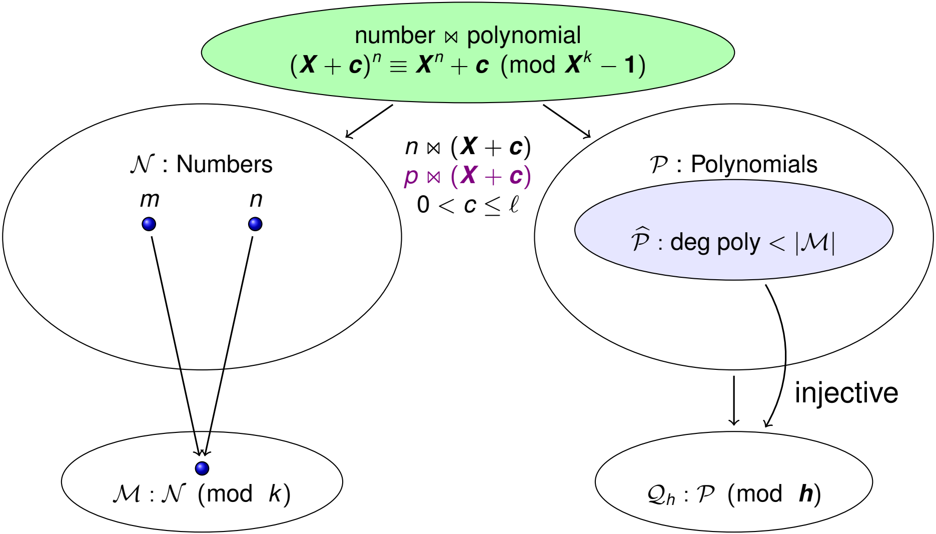
AKS Introspective Game



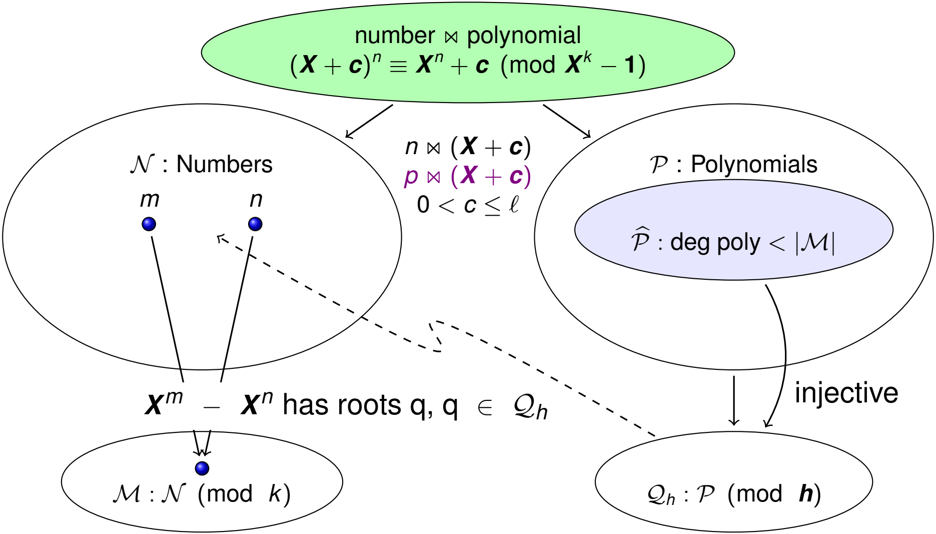
AKS Introspective Game



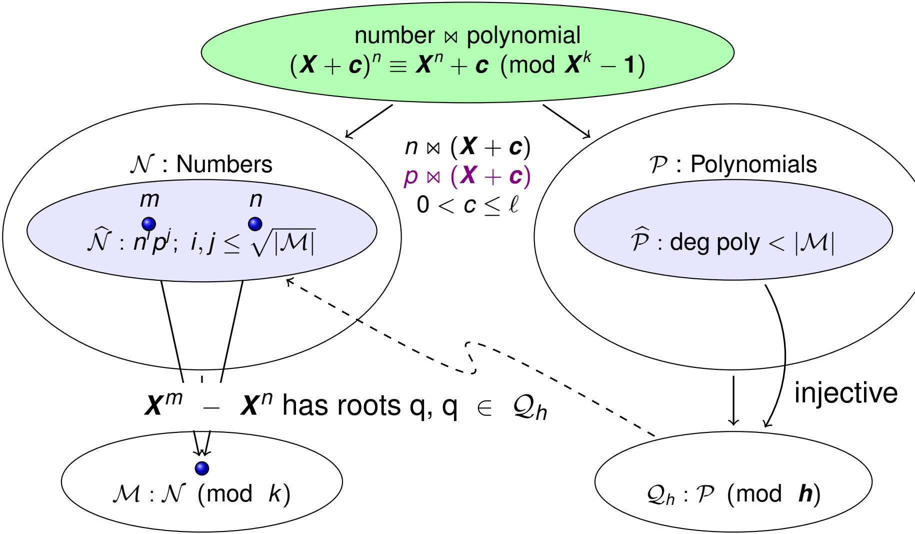
AKS Introspective Game



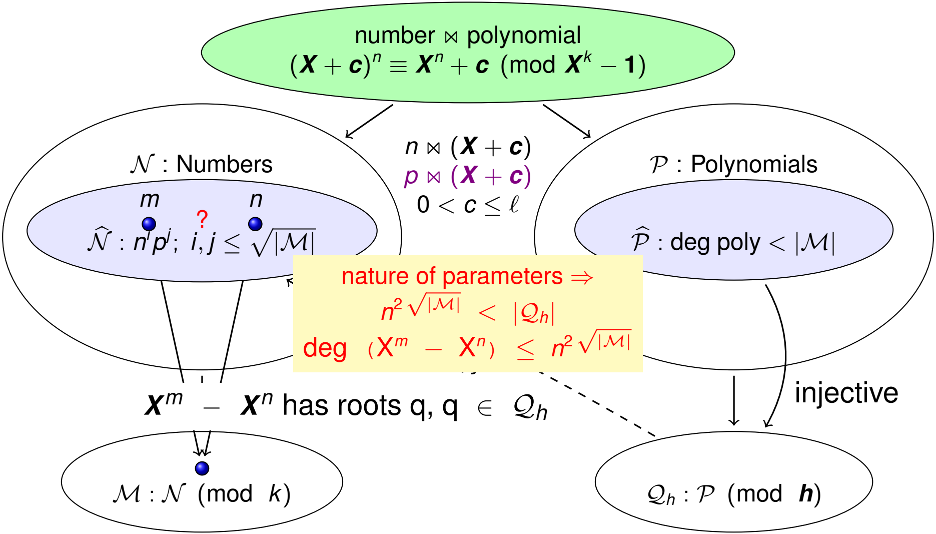
AKS Introspective Game



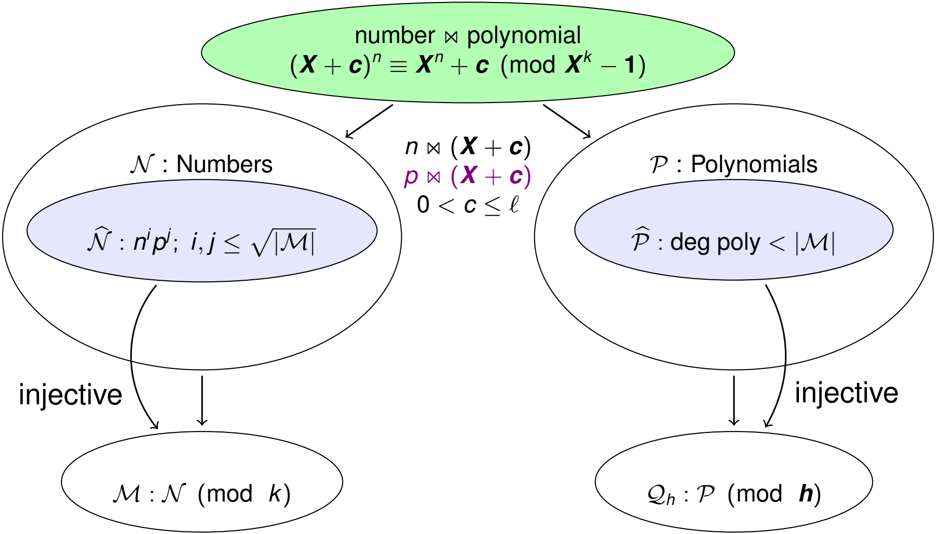
AKS Introspective Game



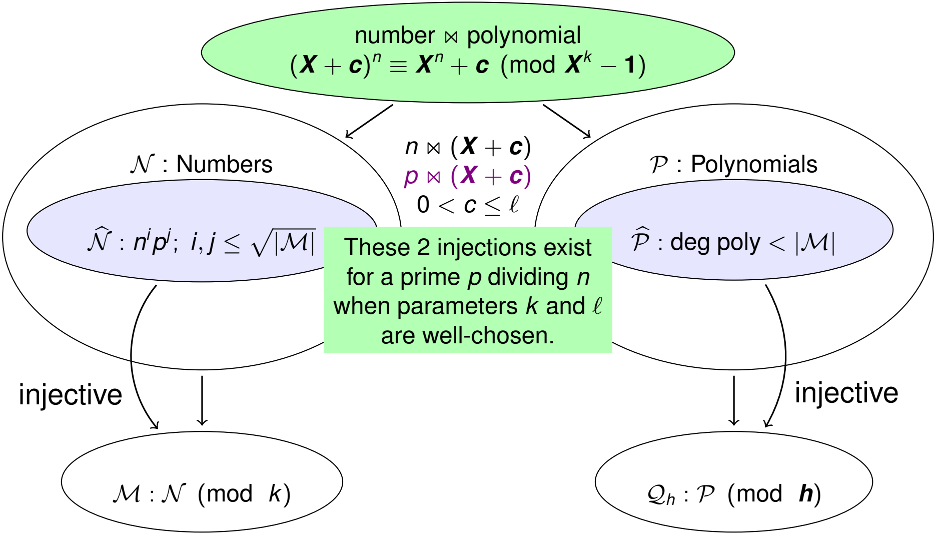
AKS Introspective Game



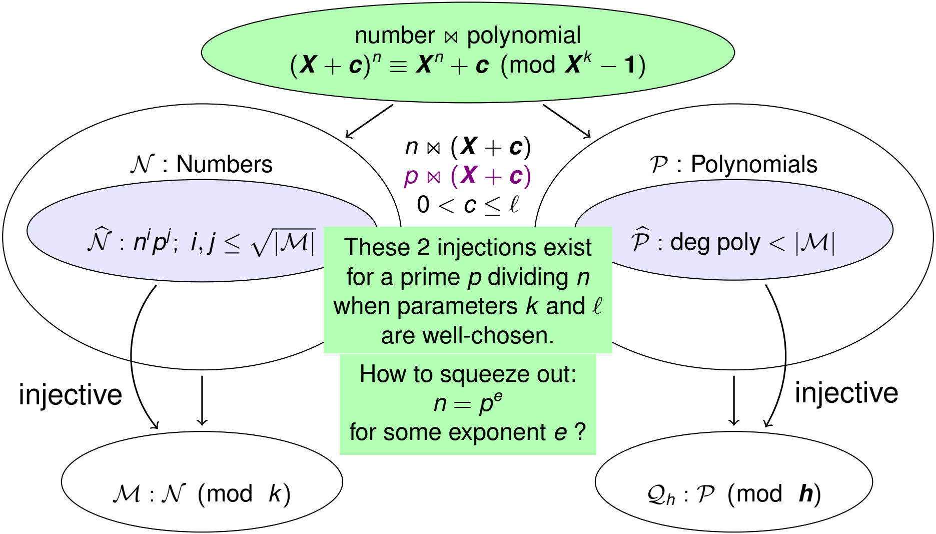
AKS Introspective Game



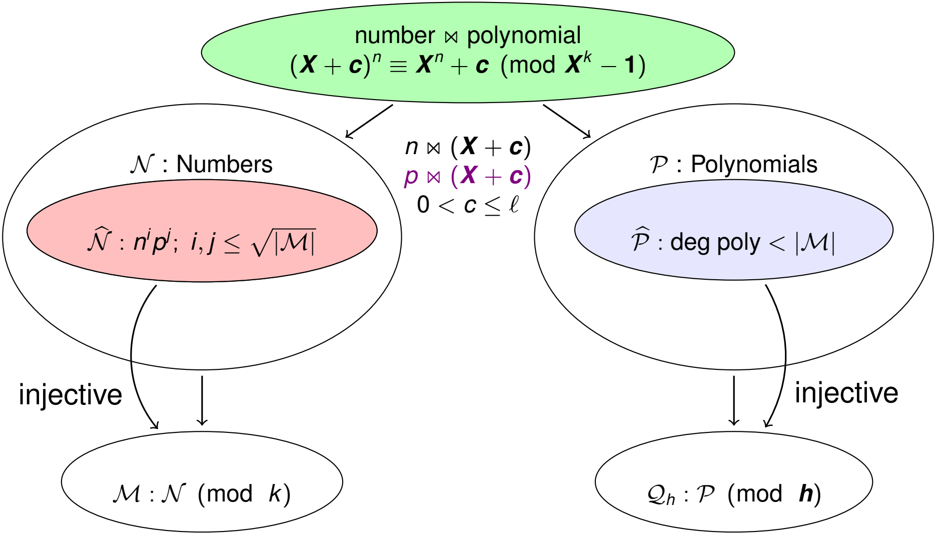
AKS Introspective Game



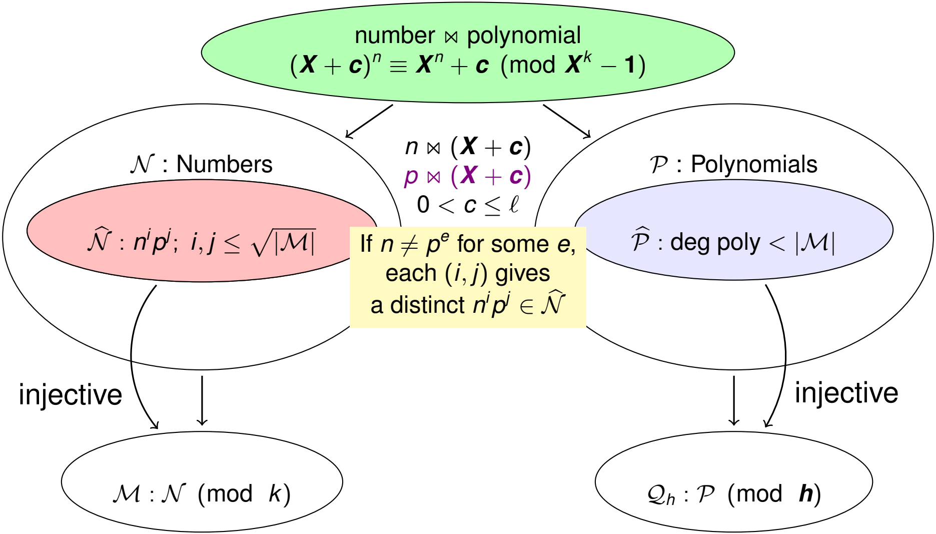
AKS Introspective Game



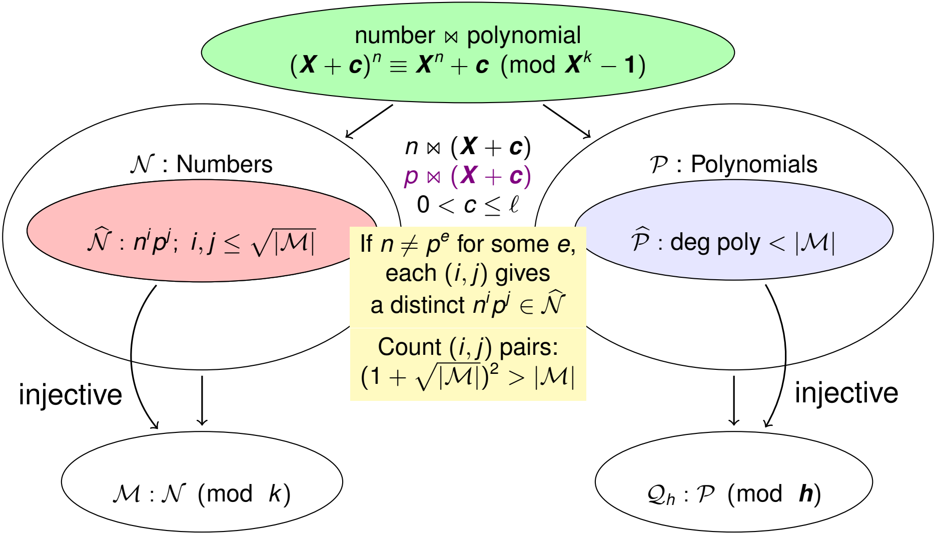
AKS Introspective Game



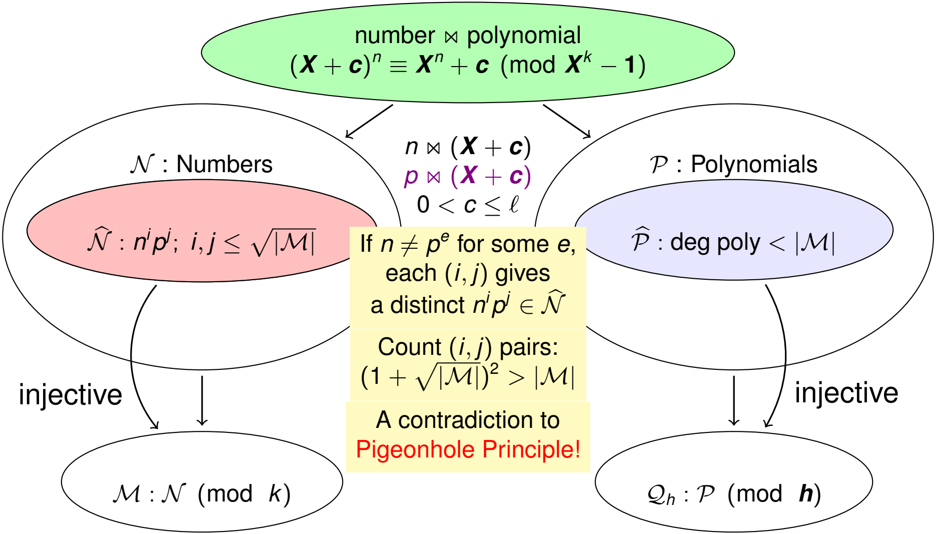
AKS Introspective Game



AKS Introspective Game



AKS Introspective Game



HOL Mechanisation

Straightforward Approach + Overloading:

```
monoid = <| carrier: 'a -> bool;
                op: 'a -> 'a -> 'a;
                id: 'a
          |>`;
```

Followed by:

```
val _ = overload_on ("*", ``g.op``);
val _ = overload_on ("#e", ``g.id``);
val _ = overload_on ("G", ``g.carrier``);
```

Some Line Counts

Underlying Algebra Library	49272
AKS specific scripts	21835

Some Line Counts

Underlying Algebra Library	49272
AKS specific scripts	21835
Core HOL	150741