Formal Proof that PRIMES is in P

with a taste of HOL4

Hing-Lun Chan

College of Engineering and Computer Science
Australian National University
joseph.chan@anu.edu.au

23 Oct 2015, COMP2600/6260

Outline

Is 91 prime?

Is 91 prime?

Trial division (known since antiquity)

- Check: Is 91 divisible by a smaller prime?

Is 91 prime?

Trial division (known since antiquity)

- Check: Is 91 divisible by a smaller prime?
  - not divisible by $2, 3, 5$,
  - but divisible by 7, hence 91 is COMPOSITE.
- *n* is prime iff it has no proper divisor.

Is 91 prime?

Trial division (known since antiquity)

- Check: Is 91 divisible by a smaller prime?
  - not divisible by $2, 3, 5$,
  - but divisible by 7, hence 91 is COMPOSITE.
- n is prime iff it has no proper divisor.

Fermat's method (around 1640)

- Check: Can 91 be expressed as $x^2 - y^2$ with $x - y \neq 1$?

Is 91 prime?

Trial division (known since antiquity)

- Check: Is 91 divisible by a smaller prime?
  - not divisible by $2, 3, 5$,
  - but divisible by 7, hence 91 is COMPOSITE.
- n is prime iff it has no proper divisor.

Fermat's method (around 1640)

- Check: Can 91 be expressed as $x^2 - y^2$ with $x - y \neq 1$?
  - $91 = 100 - 9 = 10^2 - 3^2$, hence 91 must be COMPOSITE.

Is 91 prime?

Trial division (known since antiquity)

- Check: Is 91 divisible by a smaller prime?
  - not divisible by $2, 3, 5$,
  - but divisible by 7, hence 91 is COMPOSITE.
- n is prime iff it has no proper divisor.

Fermat's method (around 1640)

- Check: Can 91 be expressed as $x^2 - y^2$ with $x - y \neq 1$?
  - $91 = 100 - 9 = 10^2 - 3^2$, hence 91 must be COMPOSITE.
- By $x^2 - y^2 = (x - y)(x + y)$,
  $91 = (10 - 3)(10 + 3) = 7 \times 13$.
- n is prime iff it is not a difference of two non-consecutive squares.

Is 91 prime?

AKS method (August 2002)

- Check that the number n is power-free, *i.e.*, not square, cube, *etc*.
- Find a suitable parameter k, and compute ℓ based on k and n.
- GCD checks: if $\gcd(n, j) = 1$ for $j = 1 \ldots k$.
- Polynomial checks: if $(x + c)^n \equiv x^n + c \pmod{n, x^k - 1}$ for $c = 1 \ldots \ell$.

Is 91 prime?

AKS method (August 2002)

- Check that the number n is power-free, *i.e.*, not square, cube, *etc.*
- Find a suitable parameter k, and compute ℓ based on k and n.
- GCD checks: if $\gcd(n, j) = 1$ for $j = 1 \ldots k$.
- Polynomial checks: if $(x + c)^n \equiv x^n + c \pmod{n, x^k - 1}$ for $c = 1 \ldots \ell$.
  - For power-free 91, parameter $k = 37$, $\ell = 36$.
  - GCD checks: found $\gcd(91, 7) \neq 1$, so `COMPOSITE` 91.

Is 91 prime?

AKS method (August 2002)

- Check that the number n is power-free, *i.e.*, not square, cube, *etc*.
- Find a suitable parameter k, and compute ℓ based on k and n.
- GCD checks: if $\gcd(n, j) = 1$ for $j = 1 \ldots k$.
- Polynomial checks: if $(x + c)^n \equiv x^n + c \pmod{n, x^k - 1}$ for $c = 1 \ldots \ell$.
  - For power-free 91, parameter $k = 37$, $\ell = 36$.
  - GCD checks: found $\gcd(91, 7) \neq 1$, so `COMPOSITE` 91.
  - Even if this is missed, Polynomial checks give:

$$(x + 1)^{91} \not\equiv x^{91} + 1 \pmod{91, x^{37} - 1}$$

  - Again `COMPOSITE` 91.

Is 91 prime?

AKS method (August 2002)

- Check that the number n is power-free, *i.e.*, not square, cube, *etc.*
- Find a suitable parameter k, and compute ℓ based on k and n.
- GCD checks: if $\gcd(n, j) = 1$ for $j = 1 \ldots k$.
- Polynomial checks: if $(x + c)^n \equiv x^n + c \pmod{n, x^k - 1}$ for $c = 1 \ldots \ell$.
  - For power-free 91, parameter $k = 37$, $\ell = 36$.
  - GCD checks: found $\gcd(91, 7) \neq 1$, so `COMPOSITE` 91.
  - Even if this is missed, Polynomial checks give:

$$(x + 1)^{91} \not\equiv x^{91} + 1 \pmod{91, x^{37} - 1}$$

  - Again `COMPOSITE` 91.
- n is prime iff it passes all AKS checks.

Is 97 prime?

Is 97 prime?

Trial division (known since antiquity)

- not divisible by $2, 3, 5, 7$,
- since $\sqrt{97} \approx 9.85$, so 97 is $\texttt{PRIME}$.

Is 97 prime?

Trial division (known since antiquity)

- not divisible by $2, 3, 5, 7$,
- since $\sqrt{97} \approx 9.85$, so 97 is $\texttt{PRIME}$.

Fermat's method (around 1640)

- note $10^2 = 100$ is nearest to 97, try $97 = 10^2 - y^2$, fail.
- fail $97 = 11^2 - y^2 = \cdots = 48^2 - y^2$ where $48 \approx \frac{97}{2}$, so $\texttt{PRIME}$ 97.

Is 97 prime?

Trial division (known since antiquity)

- not divisible by $2, 3, 5, 7$,
- since $\sqrt{97} \approx 9.85$, so 97 is $\texttt{PRIME}$.

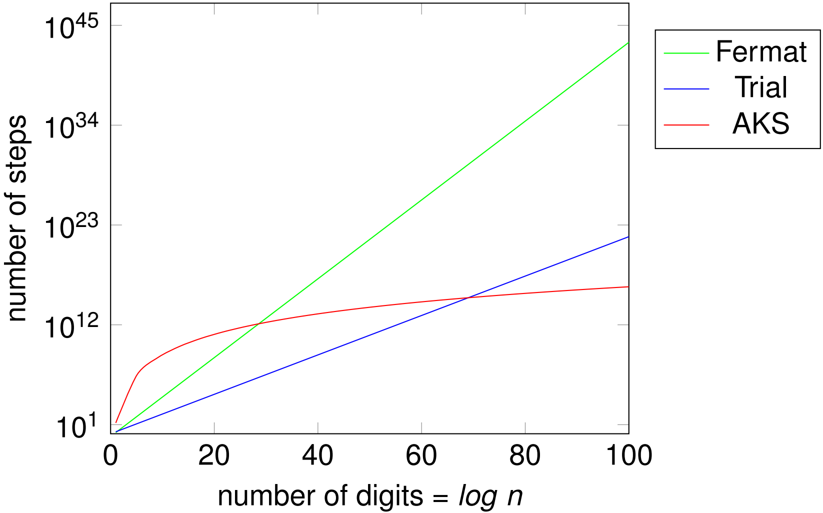
Fermat's method (around 1640)

- note $10^2 = 100$ is nearest to 97, try $97 = 10^2 - y^2$, fail.
- fail $97 = 11^2 - y^2 = \cdots = 48^2 - y^2$ where $48 \approx \frac{97}{2}$, so $\texttt{PRIME}$ 97.

AKS method (August 2002)

- for power-free 97, parameter $k = 41, \ell = 37$.
- GCD checks: pass all $\gcd(97, j) = 1$, for $1 \leq j \leq 41$.
- Polynomial checks:
  - $(x + 1)^{97} \equiv x^{97} + 1 \pmod{97, x^{41} - 1}$ pass,
  - $(x + 2)^{97} \equiv x^{97} + 2 \pmod{97, x^{41} - 1}$ pass, $\cdots$, up to
  - $(x + 37)^{97} \equiv x^{97} + 37 \pmod{97, x^{41} - 1}$, all pass.
- hence 97 is $\texttt{PRIME}$.

Primality Tests Comparison



What is PRIMES in P?

PRIMES = the problem of Primality Testing

- Given an integer $n > 1$, determine if n is prime.

P = the class of Polynomial-time Algorithm

- When step count is bounded by a polynomial of input size ($\log n$).
- Such polynomial-time algorithms are deemed practical (or useful).

What is PRIMES in P?

PRIMES = the problem of Primality Testing

- Given an integer $n > 1$, determine if n is prime.

P = the class of Polynomial-time Algorithm

- When step count is bounded by a polynomial of input size ($\log n$).
- Such polynomial-time algorithms are deemed practical (or useful).

PRIMES $\in P$? = a long-standing Question

- PRIMES $\in P$ — if you can find one such class P algorithm.
- PRIMES $\notin P$ — if you can prove no such class P algorithm exists.

What is PRIMES in P?

PRIMES = the problem of Primality Testing

- Given an integer $n > 1$, determine if n is prime.

P = the class of Polynomial-time Algorithm

- When step count is bounded by a polynomial of input size ($\log n$).
- Such polynomial-time algorithms are deemed practical (or useful).

PRIMES $\in P$? = a long-standing Question

- PRIMES $\in P$ — if you can find one such class P algorithm.
  i.e., if you can find an algorithm and prove it is in class P.
- PRIMES $\notin P$ — if you can prove no such class P algorithm exists.

What is PRIMES in P?

PRIMES = the problem of Primality Testing

- Given an integer $n > 1$, determine if n is prime.

P = the class of Polynomial-time Algorithm

- When step count is bounded by a polynomial of input size ($\log n$).
- Such polynomial-time algorithms are deemed practical (or useful).

PRIMES $\in P$? = a long-standing Question

- PRIMES $\in P$ — if you can find one such class P algorithm.
  i.e., if you can find an algorithm and prove it is in class P.
- PRIMES $\notin P$ — if you can prove no such class P algorithm exists.

Failure to show PRIMES $\in P$ promotes the belief: PRIMES $\notin P$ ...

AKS: PRIMES is in P

PRIMES

- Given an integer $n > 1$, determine if n is prime.

Methods

- Trial division (since antiquity), takes up to $\sqrt{n}$ steps, *i.e.*, $O(n)$.
- Fermat's method (around 1640), takes up to $\frac{n}{2}$ steps, *i.e.*, $O(n)$.

AKS: PRIMES is in P

PRIMES

- Given an integer $n > 1$, determine if n is prime.

Methods

- Trial division (since antiquity), takes up to $\sqrt{n}$ steps, *i.e.*, $O(n)$.
- Fermat's method (around 1640), takes up to $\frac{n}{2}$ steps, *i.e.*, $O(n)$.
- AKS method (August 2002), can be shown to be $O((\log n)^{7\frac{1}{2}})$.

AKS: PRIMES is in P

PRIMES

- Given an integer $n > 1$, determine if n is prime.

Methods

- Trial division (since antiquity), takes up to $\sqrt{n}$ steps, *i.e.*, $O(n)$.
- Fermat's method (around 1640), takes up to $\frac{n}{2}$ steps, *i.e.*, $O(n)$.
- AKS method (August 2002), can be shown to be $O((\log n)^{7\frac{1}{2}})$.

Analysis

- size of $n =$ number of digits to represent n, measured by $\log n$.
- $O(n) = O(2^{\log n})$ is an exponential function of $(\log n)$.
- $O((\log n)^{7\frac{1}{2}})$ is a polynomial function of $(\log n)$.

AKS: PRIMES is in P

PRIMES

- Given an integer $n > 1$, determine if n is prime.

Methods

- Trial division (since antiquity), takes up to $\sqrt{n}$ steps, *i.e.*, $O(n)$.
- Fermat's method (around 1640), takes up to $\frac{n}{2}$ steps, *i.e.*, $O(n)$.
- AKS method (August 2002), can be shown to be $O((\log n)^{7\frac{1}{2}})$.

Analysis

- size of $n =$ number of digits to represent n, measured by $\log n$.
- $O(n) = O(2^{\log n})$ is an exponential function of $(\log n)$.
- $O((\log n)^{7\frac{1}{2}})$ is a polynomial function of $(\log n)$.

Title of AKS's 2002 paper: PRIMES is in P.
First known deterministic *polynomial-time* primality-testing algorithm.

The AKS Algorithm

6 steps in pseudo-code:

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 ($\gcd(n, j) \neq 1$), 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`.

The AKS Main Theorem

The AKS algorihtm works because it is based on:

Theorem (The AKS Main Theorem.)

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

The AKS Main Theorem

The AKS algorihtm works because it is based on:

Theorem (The AKS Main Theorem.)

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

Proof.

ITP2015 `http://www.inf.kcl.ac.uk/staff/urbanc/itp-2015/`, a joint paper with my suprevisor, Michael Norrish. □

Formal Proof

A Theorem

- pre-conditions $\Rightarrow$ conclusion

A Mathematical Proof

- presents a series of logical arguments.
- "understood" by peers.
- using high-level concepts.

Formal Proof

A Theorem

- pre-conditions $\Rightarrow$ conclusion

A Mathematical Proof

- presents a series of logical arguments.
- “understood” by peers.
- using high-level concepts.

A Formal Proof

- presents a series of logical deductions.
- “understood” by theorem-prover.
- work out all the details.

Formal Proof

A Theorem

- pre-conditions $\Rightarrow$ conclusion

A Mathematical Proof

- presents a series of logical arguments.
- "understood" by peers.
- using high-level concepts.

A Formal Proof

- presents a series of logical deductions.
- "understood" by theorem-prover.
- work out all the details.

A Special Issue on Formal Proof
Notices of the American Mathematical Society, December 2008.
`http://www.ams.org/notices/200811/`

Formal Proof Examples

Four Colour Theorem	proposed by Francis Guthrie (1852) computer-aided proof: Appel & Haken (1976) formalized by Gonthier's team (2000-2005)

Formal Proof Examples

Four Colour Theorem	proposed by Francis Guthrie (1852) computer-aided proof: Appel & Haken (1976) formalized by Gonthier's team (2000-2005)
Odd Order Theorem	conceived by William Burnside (1911) proof (255 pages): Feit & Thompson (1963) formalized by Gonthier's team (2006-2012)

Formal Proof Examples

Four Colour Theorem	proposed by Francis Guthrie (1852) computer-aided proof: Appel & Haken (1976) formalized by Gonthier's team (2000-2005)
Odd Order Theorem	conceived by William Burnside (1911) proof (255 pages): Feit & Thompson (1963) formalized by Gonthier's team (2006-2012)
3D Sphere Packing	stated by Johannes Kepler (1611) computer-aided proof: Thomas Hales (1998) formalized in Flyspeck project (2003-2014)

Formal Proof Examples

Four Colour Theorem	proposed by Francis Guthrie (1852) computer-aided proof: Appel & Haken (1976) formalized by Gonthier's team (2000-2005)
Odd Order Theorem	conceived by William Burnside (1911) proof (255 pages): Feit & Thompson (1963) formalized by Gonthier's team (2006-2012)
3D Sphere Packing	stated by Johannes Kepler (1611) computer-aided proof: Thomas Hales (1998) formalized in Flyspeck project (2003-2014)
AKS Primality Testing	found by Agrawal, Kayal and Saxena (2002) if-part verified: de Moura and Tadeu (2008) formalized AKS Main Theorem only (2015)

Formal Proof Examples

Four Colour Theorem	proposed by Francis Guthrie (1852) computer-aided proof: Appel & Haken (1976) formalized by Gonthier's team (2000-2005)
Odd Order Theorem	conceived by William Burnside (1911) proof (255 pages): Feit & Thompson (1963) formalized by Gonthier's team (2006-2012)
3D Sphere Packing	stated by Johannes Kepler (1611) computer-aided proof: Thomas Hales (1998) formalized in Flyspeck project (2003-2014)
AKS Primality Testing	found by Agrawal, Kayal and Saxena (2002) if-part verified: de Moura and Tadeu (2008) formalized AKS Main Theorem only (2015)

I still have to formalize the AKS algorithm, and show it is indeed in P!

Formal Proof in HOL4

What is HOL4?

- an interactive theorem-prover, or *proof-assistant*.
- a descendent of the original HOL (Higher Order Logic) from 1988.

Formal Proof in HOL4

What is HOL4?

- an interactive theorem-prover, or *proof-assistant*.
- a descendent of the original HOL (Higher Order Logic) from 1988.
- can be installed in Unix, Mac OS X, or Windows PC/laptop.
- runs on top of Standard ML (a programming language).

Formal Proof in HOL4

What is HOL4?

- an interactive theorem-prover, or *proof-assistant*.
- a descendent of the original HOL (Higher Order Logic) from 1988.
- can be installed in Unix, Mac OS X, or Windows PC/laptop.
- runs on top of Standard ML (a programming language).
- starts up with Basic Libraries on sets, maps, numbers, lists, *etc*.
- includes an extensive collection of additional Libraries for work on specific topics.

Formal Proof in HOL4

What is HOL4?

- an interactive theorem-prover, or *proof-assistant*.
- a descendent of the original HOL (Higher Order Logic) from 1988.
- can be installed in Unix, Mac OS X, or Windows PC/laptop.
- runs on top of Standard ML (a programming language).
- starts up with Basic Libraries on sets, maps, numbers, lists, *etc*.
- includes an extensive collection of additional Libraries for work on specific topics.

HOL4 (sources on GitHub)

`http://hol-theorem-prover.org/`

`http://github.com/HOL-Theorem-Prover/HOL/`

AKS Source Repository

Source:
`http://bitbucket.org/jhlchan/hol/src/aks/theories`

- Helper Theories
  - AKSinteger — integer square-root and integer logarithm.
  - AKSpoly — polynomial evaluation by polynomial substitution.
  - AKScyclo — special properties of cyclotomic polynomials.
- AKS Theories
  - AKSintro — introspective relation essential to AKS proof.
  - AKSshift — shifting introspective relation between Rings.
  - AKSsets — sets involved in AKS proof.
  - AKSmaps — mappings involved in AKS proof.
  - AKSorder — the existence of an AKS parameter related to order.
  - AKStheorem — the main theorem in AKS proof.

AKS Source Repository

Source:
`http://bitbucket.org/jhlchan/hol/src/aks/theories`

- Helper Theories
  - AKSinteger — integer square-root and integer logarithm.
  - AKSpoly — polynomial evaluation by polynomial substitution.
  - AKScyclo — special properties of cyclotomic polynomials.
- AKS Theories
  - AKSintro — introspective relation essential to AKS proof.
  - AKSshift — shifting introspective relation between Rings.
  - AKSsets — sets involved in AKS proof.
  - AKSmaps — mappings involved in AKS proof.
  - AKSorder — the existence of an AKS parameter related to order.
  - AKStheorem — the main theorem in AKS proof.

These are built upon other libraries:
algebraic structures, polynomials, finite fields, vector space, *etc*.

HOL Demo

First, set up the goal to be proved in HOL4 Proof Manager.

```
> g `1 + 1 = 2`;

val it =
   Proof manager status: 1 proof.
1. Incomplete goalstack:
     Initial goal:

     1 + 1 = 2

: proof
```

Then, apply one or more tactics to prove the goal:

```
> e (DECIDE_TAC);

OK..
val it = Initial goal proved.
|- 1 + 1 = 2: proof
```