Mechanisation of AKS Algorithm

Hing-Lun Chan

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

Abstract. The AKS algorithm (by Agrawal, Kayal and Saxena) is a significant theoretical result proving "PRIMES in P", and a brilliant application of ideas from finite fields. Mechanisation of this result is a milestone in theorem-proving.

1 Introduction

When the paper [1] of the AKS algorithm first appeared in 2002, it caused a sensation because it resolved the problem of "PRIMES in P" in the affirmative: that testing whether a number n is prime can done in polynomial-time (*i.e.* the time taken is bounded by the bit-size of n). After a decade, the algorithm is still not widely used in practice, due to the high degree of the polynomial involved in the time estimate.

The aim of my PhD work is to gave a mechnical proof of the AKS algorithm in HOL. As will be explained later, there had been several attempts to do this, but so far none is successful. I have studied the original papers and other expositions of the algorithm, and note that the theory is within my area of interest: finite fields. I hope I can overcome all the hurdles and deliver a complete mechanisation of AKS algorithm.

2 Scope of Work

Since the AKS algorithm is a primality test, it has the form:

- Input an integer $n > 1$.
- Do some computations involving n, in a finite number of steps.
- Output a result: n is `PRIME` or n is `COMPOSITE`.

The mechanisation of AKS algorithm consists of two parts:

- Algorithm Correctness
  When the algorithm returns `PRIME`, input n is indeed a prime, and vice versa.
- Complexity Analysis
  The total number of steps is bounded by a polynomial function of the bit-size of input, *i.e* $\log n$.

3 Theory

The idea behind the AKS algorithm is an innovation taken from the theory of polynomials over finite fields. The starting point is this deterministic primality test:

Theorem 1. Fermat's Little Theorem for polynomials.
For any number $n > 1$ and any value a with $gcd(a, n) = 1$,

$$(x + a)^n \equiv x^n + a \mod n \Leftrightarrow n \quad \textit{is prime}.$$

Therefore to check whether n is prime, just put $a = 1$, and compute. However, this is not practical: it is even worse than the naive primality test of trying all possible factors from 1 to $\sqrt{n}$, since the expansion of the left side involves n terms, the degree of the polynomial.

To reduce the task to polynomial-time, the AKS team proposes to compute the polynomial remainder of both sides by division of some $x^r - 1$. If the degree r of the polynomial modulo is much less than n, this is a signicant reduction in the number of terms. To compensate for reduction in the number of terms, the AKS team suggests verifying more polynomial identities:

$$(x + a)^n \equiv x^n + a \mod (x^r - 1, n) \quad \text{for} \quad 1 \leq a \leq s$$

for values of a from 1 up to some limit s. Their hope is that, by suitably chosen values of r and s, if a number n can pass all the polynomial checks, then n must be prime.

It can be shown that, if this approach works, the overall number of steps is $O(rs \times \log^2 n)$. Hence if the modulo polynomial degree $r \sim O(\log^h n)$ for some h, and the limit $s \sim O(\log^k n)$ for some k, there will be a deterministic primality test in polynomial-time.

In their original paper, the AKS team proved that the idea works (Section 4), and found the existence of value $r \sim O(log^6 n)$, and $s \sim O(\sqrt{r} \log n) \sim O(\log^4 n)$, hence the overall complexity of algorithm is $O(\log^{12} n)$. A later revised version would give $r \sim O(log^3 n)$, reducing the overall complexity to $O(\log^{7\frac{1}{2}} n)$

4 Methodology

The AKS team essentially proved the following:

Theorem 2. The AKS theorem.
For any number $n > 1$, there exists values r and s bounded by polynomial functions of $\log n$, such that

$$(x + a)^n \equiv x^n + a \mod (x^r - 1, n) \quad \textit{for} \quad 1 \leq a \leq s$$

iff n is power of a prime p, i.e. *$n = p$, or p^2, or p^3,* etc.

By eliminating power forms of n in preliminary tests, this is the key for the validity of the AKS algorithm.

The AKS team also explained how to choose the values for r and s, thereby giving a complexity analysis of the algorithm, showing that it is polynomial-time.

My proposed method to tackle the problem of AKS mechanisation is:

- Correctness:
  - The AKS theorem is true.
  - The AKS algorithm is correct by the AKS theorem.

- Complexity:
  - The value r is $O(\log^h n)$ for some constant h.
  - The limit s is $O(\log^k n)$ for some constant k.
  - The total number of steps is bounded by $O(rs \times \log^2 n)$.

5 Literature

Others have been working on mechnical proofs of AKS algorithm, *e.g*, in ACL2 [2], or Coq [3].

- Towards the Verification of The AKS Primality Test in ACL2 (2004)
  - by Cynthia Campos, Dr. François Modave and Dr. Steve Roach.
  - Only verified: if input is prime, AKS returns PRIME.

- The Correctness of the AKS Primality Test in Coq (2008):
  - by Flávio L. C. de Moura and Ricardo Tadeu.
  - Only proved: if input is prime, AKS returns PRIME.

- The AKS Algorithm in ACL2
  - ACL2 project proposal from Ruben Gamboa.
  - With collaboration from Mike Gordon using HOL.

6 Contribution

Since the AKS algorithm establishes "PRIMES in P", it is a significant theoretical result. A mechanical proof of this claim will be a high point in theorem-proving. Moreover, given the disappointing previous attempts, I hope to find ways to overcome such difficulties. Even if I cannot succeed, I may be able to reveal exactly where the difficulties are, and expect some new development in theorem-proving to overcome such obstacles.

7 Work Required

- Foundation Work:
  - Build Monoid theory in HOL4.
  - Build Group theory from Monoid theory.
  - Build Ring theory using Group and Monoid.
  - Build Field theory using Ring and Group.
  - Build Polynomial theory using Field and Ring.

- Apply to AKS:
  - Code in HOL4: $AKS\ n$ that returns true or false upon input n.
  - Prove in HOL4: $AKS\ n$ returns true iff n is prime.
  - Prove in HOL4: number of steps of $AKS\ n$ is bound by $O(\log^k n)$ for some k.

8 Possible Roadblocks

- The AKS proof involves quite advanced topics in polynomials over finite fields. I may have underestimated the amount of work requried to perform its mechanization.
- The complexity analysis of algorithm is all about function growth rates. That may involves thinking about continuous variables, which may prove to be difficult for theorem provers. At any rate, it seems that very few complexity analysis have been done by theorem provers.
- The estimation of the parameter r in the AKS algorithm involves a knowledge about the distribution of primes. The standard result is the prime number theorem, although other equivalent formulations are known. Any proof about prime distribution seems to require analysis of a continuous variable, which is not easy to do in a theorem prover.

References

1. Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.
2. Cynthia Campos, Dr. Francois Modave, and Dr. Steve Roach. Towards the Verification of The AKS Primality Test in ACL2. Technical report, Department of Computer Science, University of Texas at El Paso, November 2004.
3. Flávio L. C. de Moura1 and Ricardo Tadeu. The Correctness of the AKS Primality Test in Coq. Technical report, Departamento de Ciéncia da Computaćáo, Universidade de Brasília, July 2008.