A Formalization of Pratt’s Primality Certificates

By Simon Wimmer and Lars Noschinski

August 16, 2018

Abstract

In 1975, Pratt introduced a proof system for certifying primes [1]. He showed that a number $p$ is prime iff a primality certificate for $p$ exists. By showing a logarithmic upper bound on the length of the certificates in size of the prime number, he concluded that the decision problem for prime numbers is in NP. This work formalizes soundness and completeness of Pratt’s proof system as well as an upper bound for the size of the certificate.

Contents

1 Pratt’s Primality Certificates 2
  1.1 Soundness . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  1.2 Completeness . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  1.3 Proof Method Setup . . . . . . . . . . . . . . . . . . . . . . 6

theory Pratt-Certificate

imports
  Complex-Main
  Lehmer-Lehmer
  HOL-Library.Code-Target-Numeral

begin

definition mod-exp :: nat ⇒ nat ⇒ nat ⇒ nat
where [code-abbrev]: mod-exp b e m = (b ^ e) mod m

lemma mod-exp-code [code]:
mod-exp b e m =
  (if e = 0 then if m = 1 then 0 else 1
   else if even e then mod-exp ((b * b) mod m) (e div 2) m
   else (b * mod-exp ((b * b) mod m) (e div 2) m) mod m)
(proof)

lemma eval-mod-exp [simp]:
mod-exp b 0 (Suc 0) = 0
\[ m \neq 1 \implies \text{mod-exp } b \ 0 \ m = 1 \]
\[ \text{mod-exp } b \ 1 \ m = b \ \mod m \]
\[ \text{mod-exp } b \ (\text{Suc } 0) \ m = b \ \mod m \]
\[ \text{mod-exp } b \ (\text{numeral } \ (\text{num.Bit0 } n)) \ m = \text{mod-exp } (b^2 \mod m) \ (\text{numeral } n) \ m \]
\[ \text{mod-exp } b \ (\text{numeral } \ (\text{num.Bit1 } n)) \ m = b \ast \text{mod-exp } (b^2 \mod m) \ (\text{numeral } n) \ m \mod m \]

\section{Pratt’s Primality Certificates}

This work formalizes Pratt’s proof system as described in his article “Every Prime has a Succinct Certificate”\cite{1}. The proof system makes use of two types of predicates:

- \( \text{Prime}(p) \): \( p \) is a prime number
- \( (p, a, x) \): \( \forall q \in \text{prime-factors}(x). \ [a^{((p - 1) \ \div q)} \neq 1] \ \mod p \)

We represent these predicates with the following datatype:

\textbf{datatype} \ \textit{pratt} = \textit{Prime} \ \textit{nat} | \textit{Triple} \ \textit{nat} \ \textit{nat} \ \textit{nat}

Pratt describes an inference system consisting of the axiom \((p, a, 1)\) and the following inference rules:

- **R1**: If we know that \((p, a, x)\) and \([a^{((p - 1) \ \div q)} \neq 1] \ \mod p\) hold for some prime number \( q \) we can conclude \((p, a, qx)\) from that.
- **R2**: If we know that \((p, a, p - 1)\) and \([a^{(p - 1)} = 1] \ \mod p\) hold, we can infer \( \text{Prime}(p) \).

Both rules follow from Lehmer’s theorem as we will show later on.

A list of predicates (i.e., values of type \textit{pratt}) is a \textit{certificate}, if it is built according to the inference system described above. I.e., a list \( x \neq xs \) is a certificate if \( xs \) is a certificate and \( x \) is either an axiom or all preconditions of \( x \) occur in \( xs \).

We call a certificate \( xs \) a \textit{certificate for} \( p \), if \( \text{Prime} \ p \) occurs in \( xs \).

The function \textit{valid-cert} checks whether a list is a certificate.

\textbf{fun} \ \textit{valid-cert} :: \textit{pratt list} \ \Rightarrow \ \textit{bool} \ \textit{where}

\[ \text{valid-cert } [] = \text{True} \]
\[ | \ \text{R2} : \text{valid-cert } (\text{Prime } p \# xs) \longleftrightarrow 1 < p \land \text{valid-cert } xs \]
\[ \land \ (\exists a . \ [a^{((p - 1) = 1]} \ \mod p] \land \text{Triple } p \ a \ (p - 1) \in \text{set } xs) \]
\[ | \ \text{R1} : \text{valid-cert } (\text{Triple } p \ a \ x \# xs) \longleftrightarrow p > 1 \land 0 < x \land \text{valid-cert } xs \land (x=1 \lor \]
\[ (\exists q y. x = q \ast y \land \text{Prime } q \in \text{set } xs \land \text{Triple } p \ a \ y \in \text{set } xs \]
\[ \land [a^{((p - 1) \ \div q)} \neq 1] \ \mod p)) \]

We define a function \text{size-cert} to measure the size of a certificate, assuming a binary encoding of numbers. We will use this to show that there is
a certificate for a prime number $p$ such that the size of the certificate is polynomially bounded in the size of the binary representation of $p$.

\[
\text{fun size-pratt :: pratt ⇒ real where}
\]
\[
\quad \text{size-pratt (Prime p) = } \log 2 p \\
\quad \text{size-pratt (Triple p a x) = } \log 2 p + \log 2 a + \log 2 x
\]

\[
\text{fun size-cert :: pratt list ⇒ real where}
\]
\[
\quad \text{size-cert [] = 0} \\
\quad \text{size-cert (x # xs) = 1 + size-pratt x + size-cert xs}
\]

### 1.1 Soundness

In Section 1 we introduced the predicates $\text{Prime}(p)$ and $(p, a, x)$. In this section we show that for a certificate every predicate occurring in this certificate holds. In particular, if $\text{Prime}(p)$ occurs in a certificate, $p$ is prime.

**lemma** prime-factors-one [simp]; **shows** prime-factors $\langle \text{Suc 0} \rangle = \{\}$

**lemma** prime-factors-of-prime: **fixes** $p :: \text{nat}$ **assumes** $\text{prime } p$ **shows** prime-factors $p = \{p\}$

**theorem** pratt-sound:
**assumes** 1: $\text{valid-cert } c$
**assumes** 2: $t \in \text{set } c$
**shows** $(t = \text{Prime } p \rightarrow \text{prime } p) \wedge$
\[
(t = \text{Triple } p a x \rightarrow ((\forall q \in \text{prime-factors } x. [a^\ast((p - 1) \text{ div } q) \neq 1] \mod p)) \land 0 < x))
\]

### 1.2 Completeness

In this section we show completeness of Pratt’s proof system, i.e., we show that for every prime number $p$ there exists a certificate for $p$. We also give an upper bound for the size of a minimal certificate.

The prove we give is constructive. We assume that we have certificates for all prime factors of $p - 1$ and use these to build a certificate for $p$ from that. It is important to note that certificates can be concatenated.

**lemma** valid-cert-appendI:
**assumes** valid-cert $r$
**assumes** valid-cert $s$
**shows** valid-cert $(r @ s)$

**lemma** valid-cert-concatI: $(\forall x \in \text{set } xs. \text{valid-cert } x) \implies \text{valid-cert } (\text{concat } xs)$
lemma size-pratt-le:
  fixes d :: real
  assumes \( \forall x \in \text{set } c. \text{size-pratt } x \leq d \)
  shows \( \text{size-cert } c \leq \text{length } c * (1 + d) \) \(\langle \text{proof} \rangle\)

fun build-fpc :: nat \Rightarrow nat \Rightarrow nat \Rightarrow \text{pratt list} where
  build-fpc p a r [] = [Triple p a r] |
  build-fpc p a r (y # ys) = Triple p a r \# build-fpc p a (r div y) ys

The function build-fpc helps us to construct a certificate for \( p \) from the certificates for the prime factors of \( p - 1 \). Called as \( \text{build-fpc } p \ a \ (p - 1) \ q s \) where \( q s = q_1 \ldots q_n \) is prime decomposition of \( p - 1 \) such that \( q_1 \cdots q_n = p - 1 \), it returns the following list of predicates:

\[
(p, a, p - 1), (p, a, \frac{p - 1}{q_1}), (p, a, \frac{p - 1}{q_1q_2}), \ldots, (p, a, \frac{p - 1}{q_1 \cdots q_n}) = (p, a, 1)
\]

I.e., if there is an appropriate \( a \) and and a certificate \( rs \) for all prime factors of \( p \), then we can construct a certificate for \( p \) as

\( \text{Prime } p \# \text{build-fpc } p \ a \ (p - 1) \ q s @ rs \)

The following lemma shows that build-fpc extends a certificate that satisfies the preconditions described before to a correct certificate.

lemma correct-fpc:
  assumes valid-cert \( xs \ p > 1 \)
  assumes prod-list \( q s = r \ r \neq 0 \)
  assumes \( \forall q \in \text{set } q s. \text{Prime } q \in \text{set } x s \)
  assumes \( \forall q \in \text{set } q s. [a \backslash ((p - 1) \ \text{div } q) \neq 1] \ (\text{mod } p) \)
  shows valid-cert (build-fpc \( p \ a \ q s \ @ x s \)) \(\langle \text{proof} \rangle\)

lemma length-fpc:
  length (build-fpc \( p \ a \ q s \)) = length \( q s \) + 1 \(\langle \text{proof} \rangle\)

lemma div-gt-0:
  fixes \( m \ n :: \text{nat} \)
  assumes \( m \leq n \ 0 < m \)
  shows \( 0 < n \ \text{div } m \) \(\langle \text{proof} \rangle\)

lemma size-pratt-fpc:
  assumes \( a \leq p \ r \leq p \ 0 \leq a \ 0 < r \ 0 < p \ \text{prod-list } q s = r \)
  shows \( \forall x \in \text{set } (\text{build-fpc } p \ a \ q s) . \text{size-pratt } x \leq 3 * \text{log } 2 \ p \) \(\langle \text{proof} \rangle\)

lemma concat-set:
  assumes \( \forall q \in q s . \exists c \in \text{set } c s . \text{Prime } q \in \text{set } c \)
  shows \( \forall q \in q s . \text{Prime } q \in \text{set } (\text{concat } c s) \) \(\langle \text{proof} \rangle\)
lemma \textit{p-in-prime-factorsE}:
\begin{itemize}
\item \textbf{fixes} \( n :: \text{nat} \)
\item \textbf{assumes} \( p \in \text{prime-factors} \; n \) \( 0 < n \)
\item \textbf{obtains} \( 2 \leq p \leq n \) \( p \) \text{div} \( n \) \( \) \text{prime} \( p \)
\end{itemize}
\( \langle \text{proof} \rangle \)

lemma \textit{prime-factors-list-prime}:
\begin{itemize}
\item \textbf{fixes} \( n :: \text{nat} \)
\item \textbf{assumes} \( \text{prime} \; n \)
\item \textbf{shows} \( \exists \; qs. \; \text{prime-factors} \; n = \text{set} \; qs \land \prod \text{list} \; qs = n \land \text{length} \; qs = 1 \)
\end{itemize}
\( \langle \text{proof} \rangle \)

lemma \textit{prime-factors-list}:
\begin{itemize}
\item \textbf{fixes} \( n :: \text{nat} \)
\item \textbf{assumes} \( 3 < n \) \( \neg \) \text{prime} \( n \)
\item \textbf{shows} \( \exists \; qs. \; \text{prime-factors} \; n = \text{set} \; qs \land \prod \text{list} \; qs = n \land \text{length} \; qs \geq 2 \)
\end{itemize}
\( \langle \text{proof} \rangle \)

lemma \textit{prod-list-ge}:
\begin{itemize}
\item \textbf{fixes} \( xs :: \text{nat} \; \text{list} \)
\item \textbf{assumes} \( \forall \; x \in \text{set} \; xs. \; x \geq 1 \)
\item \textbf{shows} \( \prod \text{list} \; xs \geq 1 \)
\end{itemize}
\( \langle \text{proof} \rangle \)

lemma \textit{sum-list-log}:
\begin{itemize}
\item \textbf{fixes} \( b :: \text{real} \)
\item \textbf{fixes} \( xs :: \text{nat} \; \text{list} \)
\item \textbf{assumes} \( b > 0 \) \( b \neq 1 \)
\item \textbf{assumes} \( xs: \forall \; x \in \text{set} \; xs. \; x \geq b \)
\item \textbf{shows} \( \left( \sum x \leftarrow xs. \log b \; x \right) = \log b \left( \prod \text{list} \; xs \right) \)
\end{itemize}
\( \langle \text{proof} \rangle \)

lemma \textit{concat-length-le}:
\begin{itemize}
\item \textbf{fixes} \( g :: \text{nat} \Rightarrow \text{real} \)
\item \textbf{assumes} \( \forall \; x \in \text{set} \; xs. \; \text{real} \; \left( \text{length} \; (f \; x) \right) \leq g \; x \)
\item \textbf{shows} \( \text{length} \; (\text{concat} \; (\text{map} \; f \; xs)) \leq \left( \sum x \leftarrow xs. \; g \; x \right) \)
\end{itemize}
\( \langle \text{proof} \rangle \)

lemma \textit{prime-gt-3-impl-p-minus-one-not-prime}:
\begin{itemize}
\item \textbf{fixes} \( p :: \text{nat} \)
\item \textbf{assumes} \( \text{prime} \; p \)
\item \textbf{shows} \( \neg \) \text{prime} \( (p - 1) \)
\end{itemize}
\( \langle \text{proof} \rangle \)

We now prove that Pratt’s proof system is complete and derive upper bounds for the length and the size of the entries of a minimal certificate.

theorem \textit{pratt-complete}:
\begin{itemize}
\item \textbf{assumes} \( \text{prime} \; p \)
\item \textbf{shows} \( 3 \; \text{c.} \; \text{Prime} \; p \in \text{set} \; c \land \text{valid-cert} \; c \land \text{length} \; c \leq 6 \ast \log 2 \; p - 4 \land \forall \; x \in \text{set} \; c. \; \text{size-pratt} \; x \leq 3 \ast \log 2 \; p \)
\end{itemize}
\( \langle \text{proof} \rangle \)
We now recapitulate our results. A number $p$ is prime if and only if there is a certificate for $p$. Moreover, for a prime $p$ there always is a certificate whose size is polynomially bounded in the logarithm of $p$.

**corollary pratt:**

prime $p \iff (\exists c. \text{Prime } p \in \text{set } c \land \text{valid-cert } c)$

**corollary pratt-size:**

assumes prime $p$

shows $\exists c. \text{Prime } p \in \text{set } c \land \text{valid-cert } c \land \text{size-cert } c \leq (6 \ast \log 2 p - 4) \ast (1 + 3 \ast \log 2 p)$

**1.3 Proof Method Setup**

Using the following ‘lazy disjunction’, we can force the simplifier to fully simplify the first operand of the disjunction first and not evaluate the second one at all if the first one simplifies to True. This improved performance three-fold in our simple test.

**definition LAZY-DISJ where**

LAZY-DISJ $a \; b \iff a \lor b$

**lemma eval-LAZY-DISJ [simp]:** LAZY-DISJ False $b = b$ LAZY-DISJ True $b = True$

**lemma LAZY-DISJ-cong [cong]:** $a = a' \Longrightarrow$ LAZY-DISJ $a \; b = LAZY-DISJ a' \; b$

The following alternative definitions of valid certificates are better suited for evaluation by the simplifier than the original ones.

**context**

**begin**

**lemma valid-cert-Cons1:**

valid-cert (Prime $p \# x$s) $\iff$

$p > 1 \land (\exists t \in \text{set } x$s. case $t$ of Prime - $\Rightarrow$ False | Triple $p' \; a \; x \Rightarrow p' = p \land x = p - 1 \land \text{mod-exp } a (p-1) p = 1 ) \land \text{valid-cert } x$s

(is ?lhs = ?rhs)

**lemma Suc-0-mod-eq-Suc-0-iff:**

Suc 0 $\mod$ n = Suc 0 $\iff$ n $\neq$ Suc 0

**lemma Suc-0-eq-Suc-0-mod-iff:**

Suc 0 = Suc 0 $\mod$ n $\iff$ n $\neq$ Suc 0

**lemma valid-cert-Cons2:**
valid-cert \((\text{Triple } p \ a \ x \ # \ xs) \leftrightarrow x > 0 \land p > 1 \land (\text{LAZY-DISJ } (x = 1) \ (\exists t \in \text{set } xs, \ \text{case } t \text{ of } \text{Prime } - \Rightarrow \text{False} \ | \ \text{Triple } p' a' y \Rightarrow p' = p \land a' = a \land y \mid x \land \text{(let } q = x \div y \text{ in } \text{Prime } q \in \text{set } xs \land \text{mod-exp } a ((p-1) \div q) p \neq 1)))
\land \text{valid-cert } xs
\langle \text{proof} \rangle
\langle \text{ML} \rangle
\end{ML}

The following two theorems serve as regression tests – the first one computes the certificate in ML automatically, whereas the second one uses a pre-computed certificate. The first example should not take more than a few seconds; the second one no more than 30 seconds or so.

\begin{lemma}
\text{prime } (2503 :: \text{nat})
\langle \text{proof} \rangle
\end{lemma}

\begin{lemma}
\text{prime } (131059 :: \text{nat})
\langle \text{proof} \rangle
\end{lemma}

\begin{ML}
\end{ML}

References