A proof of Bertrand's postulate

Andrea Asperti, Wilmer Ricciotti
Dipartimento di Scienze dell'Informazione
Università degli Studi di Bologna

We discuss the formalization, in the Matita Interactive Theorem Prover, of some results by Chebyshev concerning the distribution of prime numbers, subsuming, as a corollary, Bertrand's postulate. Even if Chebyshev's result has been later superseded by the stronger prime number theorem, his machinery, and in particular the two functions ψ and θ still play a central role in the modern development of number theory. The proof makes use of most part of the machinery of elementary arithmetics, and in particular of properties of prime numbers, gcd, products and summations, providing a natural benchmark for assessing the actual development of the arithmetical knowledge base.

1. INTRODUCTION

Bertrand's postulate states that for every natural number n larger than 1 there is always at least one prime p such that $n < p \leq 2n$.

The statement was first conjectured by Joseph Bertrand 1945, who also verified his correctness for all numbers up to $3 \cdot 10^6$. A first complete proof was given by Chebyshev in 1850; alternative proofs have been given by Ramanujan in 1919 [17], using properties of the Gamma function, and by Erdös in 1932 [10], exploiting Chebyshev's function θ. So, even if Bertrand's postulate is subsumed by the prime number theorem of Hadamard and la Vallé Poussin (1896), it has an interest in its own, and in particular it makes very precise claims about the distribution of primes for small values of n.

The prime number theorem (PNT) affirms that the number of primes $\pi(n)$ not exceeding n is asymptotically equal to $n/\log(n)$ (see e.g. [16, 20] for a modern introduction to the subject). As a consequence, the number of primes between n and $2n$ grows as $n/log(n)$ when n is large, and hence there are many more primes in this interval than asserted by Bertrand's Postulate.

Chebyshev original proof follows a similar approach, but relying on a weaker property than PNT, namely that the *order of magnitude* of $\pi(n)$ is $n/\log n$ (Chebyshev's theorem), meaning that we can find two positive constants c_1 and c_2 such that, for any n

$$c_1 \frac{n}{\log(n)} \leq \pi(n) \leq c_2 \frac{n}{\log n}$$

Then, if $a > c_2/c_1$.

$$\pi(an) - \pi(n) \geq c_1 \frac{an}{\log(2n)} - c_2 \frac{n}{\log n} \geq \frac{(ac_1 - c2)n}{\log(2n)} > 0$$

so we are sure to have at least a prime between n and an. If (for sufficiently large n) we are able to prove that $c_2 < 2c_1$, then Bertrand's conjecture follows.

Even if Chebyshev's theorem is sensibly simpler than the prime number theorem, already formalized by Avigad et al. in Isabelle [8] and by Harrison in HOL Light [15], it is far from trivial (in Hardy and Wright's famous textbook [13], it takes pages 340-344 of chapter 22). The proof makes use a lot of machinery of elementary arithmetics, and in particular of properties of prime numbers, gcd, products and summations, providing a natural benchmark for assessing the actual development of the arithmetical knowledge base in proof assistants, and for comparing them.

We were also interested in providing a fully arithmetical (and constructive) proof of this theorem. Even if Selberg's proof of the prime number theorem is "elementary", meaning that it requires no sophisticated tools of analysis except for the properties of logarithms, a fully arithmetical proof of this results looks problematics, considering that the statement involves *in an essential way* the Naperian logarithm. On the other side, the logarithm in Chebyshev's theorem can be in any base, and can be also essentially avoided (at least from the statement), asserting the existence of two constants c_1 and c_2 such that, for any n

$$2^{c_1 n} \leq n^{\pi(n)} \leq 2^{c_2 n}$$

that is what we actually proved.

For the proof of Bertrand's postulate we followed instead Erdös argument, that is particularly elegant and allows us to take advantage of the computational facilities of the proof assistant Matita.

This paper is an extended and largely revised version of our previous work [3]. In particular, the proof assistant Matita [5], that we used for the formalization, has meanwhile undergone a substantial evolution, giving us the opportunity to make a detailed comparison between the two versions, and to discuss the many improvements introduced in the new system.

The structure of the paper is the following. In Section 2 we discuss some important preliminaries behind the formal proof, and in particular the new library of big operators of Matita. Section 3 is devoted to some considerations about the factorization of $n!$; in particular, we exploit the decomposition of $(2n)!$ as $n!^2 \cdot B(2n)$, where $B(2n)$ is the binomial coefficient $\binom{2n}{n}$, and study upper and lower bounds for $B(2n)$. Section 4 contains the definition of Chebyshev's Ψ function and the proof of the asymptotic distribution of the prime numbers. Finally, Section 5 is devoted to the proof of Bertrand's postulate, comprising a discussion of Chebyshev's θ function.

2. PRELIMINARIES

In the rest of the paper, all functions are defined on natural numbers. In particular, n/m denotes the integer part of the division between n and m, and $\log_a n$ denotes the maximum i such $a^i \leq n$. We shall use the notation $n!$ for the factorial of n, that is the product of all positive numbers below or equal to n.

$$n! = \prod_{i \leq n} i \tag{1}$$

Chebyshev's approach to the study of the distribution of prime numbers consists in exploiting the decomposition of $n!$ as a product of *prime* numbers. The idea is that the numbers $1, 2, \ldots, n$ include just $\frac{n}{p}$ multiples of p, $\frac{n}{p^2}$ multiples of p^2, an so on. Hence

$$n! = \prod_{p \leq n} \prod_{i < \log_p n} p^{n/p^{i+1}} \tag{2}$$

A *formal* proof of the previous sentence consists in a symbolic manipulation of expressions on "big products", allowing us to pass from equation 1 to equation 2. This kind of manipulations are an easy task for a mathematician, but are relatively complex operations to be performed inside a proof assistant, requiring a large library of lemmas covering not trivial operations such as reindexing and commutation.

It is only in relatively recent times that the importance of a good library of canonical big operators has been duly emphasized as one of the key ingredients behind a really usable library of formal mathematics [9]. In this section, we shall briefly describe the new "bigops" library of Matita, comprising both notations covering in a uniform way different form of indexing, and lemmas encapsulating the most frequent logical steps on these constructs.

The library relies on Matita's mechanism of *unification hints* [4, 19] for expressing structural and algebraic properties of indices and operators. This allows rewriting and resolution to infer such properties automatically, which is essential for the practical usability of the library.

In particular, unifications hints are used, similarly to canonical structures in [9], to enrich concrete operations to meet the algebraic requirements (associativity, commutativity, etc.) of generic operators in parametric lemmas.

For instance, consider decomposition laws like the following, where $a \leq b \leq c$

$$\sum_{i \in [a,c]} f i = \sum_{i \in [a,b]} f i + \sum_{i \in [b,c]} f i \qquad \prod_{i \in [a,c]} f i = \prod_{i \in [a,b]} f i \cdot \prod_{i \in [b,c]} f i$$

One would like to express them in a generic way, for an arbitrary operator op to be instantiated with + and $\cdot$, respectively. However, the abstract law is true only if the operator is *associative*. As discussed in [9], the best approach is to define the notion of "associative operation", that is just a record composed by a function and a proof that the function is associative, and to state the abstract property for a generic "associative operation".

Then, we use the mechanism of unification hints to suggest to Matita that, if it meets a sum, and if required by unification, it may lift it up to an associative operation using some laws suggested by the user.

Note that, a priori, this is a *narrowing* operation [11], and it would be a quite complex operation to be performed automatically by the system without some hint from the user).

The basic machinery of big operators was already in the old version of Matita (see e.g. [2]); however, we were lacking the mechanism of proof hints, essentially forcing us to replicate the statement of most lemmas for each specific operator.

2.1 Notation

The notation for a generic big operator must be parametric in the range, in the function F applied on elements in the range, in the operator op used to combine them, and in the value nil to be returned when the range is empty.

The notation is relatively standard ([9]), and has the following shape:

```
\big [ op / nil ]_{ range description } F
```

The range description is responsible for giving the name of the bound variable and stating the set over which this variable is supposed to range. The elements in the range are supposed to be enumerated (that is not a limitation, considering that the range must be finite), hence the range is specified as an interval $i \in [a, b[$ where a is the lower bound (included in the range) and b is the upper bound (not included in the range). In case the lower bound is 0, the simpler notation $i < b$ can also be used. The variable i whose name can obviously be chosen by the user, is bound by the notation, and its scope ranges over F.

Following our old library of generic iterators (see [2]), we add the possibility to filter the range with a boolean predicate, meaning that the big operator takes only the elements of the range that satisfy the predicate. This is simply written by adding `| P` at the end of the index and range description. Again, the scope of the bound variable i ranges over the formula `P`. For instance, the following notation represents the product of all primes less or equal to n (that is essentially Chebyshev θ function).

```
\big[times/1]_{p < S n | primeb p} p
```

The primitive operator is `\big[op/nil]_{i < n | p i} f i`, implemented by the following code, whose reading is straightforward

```
let rec bigop (n:nat) (p:nat → bool) (B:Type[0])
   (nil : B) (op: B → B → B) (f: nat → B) :=
  match n with
   [ O ⇒ nil
   | S k ⇒ match p k with
      [true ⇒ op (f k) (bigop k p B nil op f)
      | false ⇒ bigop k p B nil op f]
   ].
```

We also provide special notation in case the operator satisfies a particular structure. In particular, in case the operator is the sum or the product on natural numbers, the user can use the more readable and comfortable notations $\sum$ (`\sum`) and $\prod$ (`\prod`) as abbreviations for, respectively, `\big[plus\0]` and `\big[times\1]`.

2.2 Main Lemmas

The library of results concerning big operators is naturally organized according to the assumptions on the operator being iterated.

A first collection of lemmas, mostly relative to subsuming equalities on big operators from equalities of its components, do not require any assumption on the operator. The following is a typical example, expressing the fact that we can rewrite in the predicate and the formula parts of a big operation without changing its semantics:

```
lemma same_bigop : ∀k,p1,p2,B,nil,op.∀f,g:nat→ B.
  sameF_upto k bool p1 p2 → sameF_p k p1 B f g →
  \big[op,nil]_{i < k | p1 i}(f i) = \big[op,nil]_{i < k | p2 i}(g i).
```

The first premise `sameF_upto k bool p1 p2` expresses the fact that the two predicates `p1` and `p1` must coincide (pointwise) in the interval $[0,k[$; the second condition `sameF_p k p1 B f g` requires that the two functions `f` and `g` coincide pointwise in the range `{i < k | p1 i}` (that is the same as `{i < k | p2 i}`).

A second collection of lemmas allow to change the length of the range. A typical case is to restrict the bound when it is known that the filtering predicate is false on the eliminated part.

```
theorem pad_bigop1: ∀k,n,p,B,nil,op.∀f:nat→ B. n ≤ k →
  (∀i. n ≤ i → i < k → p i = false) →
  \big[op,nil]_{i < n | p i}(f i) = \big[op,nil]_{i < k | p i}(f i).
```

More interesting decompositions can be obtained in case the operator is associative. For example, we can use the following lemma to decompose an interval in two parts:

```
theorem bigop_sumI: ∀a,b,c,p,B.∀nil.∀op:Aop B nil.∀f:nat→ B.
a ≤ b → b ≤ c →
\big[op,nil]_{i∈[a,c[ | p i}(f i) =
  op (\big[op,nil]_{i ∈ [b,c[ | p i}(f i)) (\big[op,nil]_{i ∈ [a,b[ | p i}(f i)).
```

`Aop` is the type of associative operators, defined in the following way:

```
record Aop (A:Type[0]) (nil:A) : Type[0] :=
  {op :2> A → A → A;
   nill :∀a. op nil a = a;
   nilr :∀a. op a nil = a;
   assoc: ∀a,b,c.op a (op b c) = op (op a b) c
  }.
```

A slightly more complex theorem, that plays a crucial role for the commutation of bigops, is the following one, exploiting the decomposition of an integer i in the range $[0, k_1 \times k_2[$ as a pair $\langle i/k_2, i \bmod k_2\rangle$ where the first component ranges in the interval $[0, k_1[$ and the second one in the interval $[0, k_2[$.

```
theorem bigop_prod: ∀k1,k2,p1,p2,B.∀nil.∀op:Aop B nil.∀f: nat → nat → B.
\big[op,nil]_{x<k1|p1 x}(\big[op,nil]_{i<k2|p2 x i}(f x i)) =
  \big[op,nil]_{i<k1*k2|andb (p1 (i/k2)) (p2 (i/k2) (i \mod k2))}
     (f (i/k2) (i \mod k2)).
```

Things become really interesting when the operator is also commutative, since in this case the order in which elements in the range are processed becomes irrelevant.

For instance, a very useful lemma is the following one, allowing to single out a specific element i from the range, processing it independently:

```
lemma bigop_diff: ∀p,B.∀nil.∀op:ACop B nil.∀f:nat → B.∀i,n.
  i < n → p i = true →
  \big[op,nil]_{x<n|p x}(f x)=
    op (f i) (\big[op,nil]_{x<n|andb(notb(eqb i x))(p x)}(f x)).
```

Another, more general one states that the semantics of the bigops is the same, provided the ranges are isomorphic.

```
theorem bigop_iso: ∀n1,n2,p1,p2,B.∀nil.∀op:ACop B nil.∀f1,f2.
  iso B (mk_range B f1 n1 p1) (mk_range B f2 n2 p2) →
  \big[op,nil]_{i<n1|p1 i}(f1 i) = \big[op,nil]_{i<n2|p2 i}(f2 i).
```

As a corollary, we obtain the following commutation result, extremely useful for our development.

```
theorem bigop_commute: ∀n,m,p11,p12,p21,p22,B.∀nil.∀op:ACop B nil.∀f.
0 < n → 0 < m →
(∀i,j. i < n → j < m → (p11 i ∧ p12 i j) = (p21 j ∧ p22 i j)) →
\big[op,nil]_{i<n|p11 i}(\big[op,nil]_{j<m|p12 i j}(f i j)) =
   \big[op,nil]_{j<m|p21 j}(\big[op,nil]_{i<n|p22 i j}(f i j)).
```

A final set of lemma is devoted to distributivity. In this case, we expect to have two operators `sum` and `prod`, such that `sum` is associative and commutative, and `prod` distributes over `sum`:

```
record Dop (A:Type[0]) (nil:A): Type[0] :=
  {sum : ACop A nil;
   prod: A → A → A;
   null: \forall a. prod a nil = nil;
   distr: ∀a,b,c:A. prod a (sum b c) = sum (prod a b) (prod a c)
  }.
```

One of the main results is the following one, whose reading is immediate:

```
theorem bigop_distr: ∀n,p,B,nil.∀R:Dop B nil.∀f,a.
  let aop :=sum B nil R in
  let mop :=prod B nil R in
  mop a \big[aop,nil]_{i<n|p i}(f i) =
   \big[aop,nil]_{i<n|p i}(mop a (f i)).
```

3. PRIMES AND THE FACTORIAL FUNCTION

As we already said, the starting point of Chebyshev was the following decomposition of the factorial of n (see e.g. [13], p. 342).

$$n! = \prod_{p \leq n} \prod_{i < \log_p n} p^{n/p^{i+1}} \tag{3}$$

Giving a formal proof of the previous statement requires some work, starting from the definition of the *order* of a prime p in a number n, and the properties of this function.

3.1 Order of a prime

Every integer n may be uniquely decomposed as the product of all its prime factors. For our purposes, we also need to take into account the so called *order* `ord p n` of each prime p in n, that is the multiplicity of p as a factor of n.

This function can be algorithmically defined by iterating the division operator until $(n \bmod p) \neq 0$, increasing a counter at each iteration. However, since Matita only accepts well founded recursive definitions, we also need to provide an upper bound k to the number of iterations (that in is this case can initialized to n itself). It is also convenient to compute, at the same time with the order of p in n, the remainder of n after removing all p-factors.

```
let rec p_ord_aux k n p :=
  match n \mod p with
  [ O ⇒
    match k with
      [ O ⇒ ⟨O,n⟩ (∗ dummy ∗)
      | S k0 ⇒ let ⟨q,r⟩ :=p_ord_aux k0 (n / p) m in ⟨S q,r⟩]
  | S _ ⇒ ⟨O,n⟩].

(∗ p_ord n p = <q,r> if p divides n q times, with remainder r ∗)
definition p_ord :=λn,p:nat.p_ord_aux n n p.

definition ord :nat → nat → nat :=λn,p. fst ?? (p_ord n p).

definition ord_rem :nat → nat → nat :=λn,p. snd ?? (p_ord n p).
```

The behavior of the previous functions is essentially captured by the following theorems, but a small library of about 30 lemmas help to work with them in a comfortable way.

```
theorem exp_ord: ∀p,n. 1 < p → O < n →
  n = pˆ(ord n p) ∗ (ord_rem n p).
```

```
theorem not_divides_ord_rem: ∀m,p.O < m → 1 < p →
  p ∤ (ord_rem m p).
```

The following result characterizes the order of p in n as the number of times p^{i+1} divides n.

```
theorem eq_ord_sigma_p: ∀n,m,p. O < n → prime p →
  pˆm ≤ n → n < pˆ(S m) →
    ord n p = ∑_{i < m | dividesb (pˆ(S i)) n} 1.
```

3.2 Factorization

With the help of the order function, the factorization of a natural number n can now be expressed in the following way:

```
theorem factorization: ∀n. O < n →
  n = ∏_{ p < (S n) | primeb p}(pˆ(ord n p)).
```

The idea for proving the previous result is to work by induction on the upper bound of the product, namely n; the problem is that n appears many times in the statement with different roles, and a brute force approach would not provide the suitable induction hypothesis. We need first to generalize a bit the statement, making the bound more independent from the integer being factorized. In fact, the statement remains true provided the range of the product covers any prime that is a divisor of n, and we can take as an upper bound their maximum:

```
lemma max_to_factorization: ∀q,m. O<m →
  max (S m) (λi.primeb i ∧ dividesb i m)<q →
    m = ∏_{p < q | primeb p ∧ dividesb p m} (pˆ(ord m p)).
```

This result can now be proved by induction on m: the proof is not entirely straightforward (it takes about 100 lines in Matita) but it does not present any major conceptual difficulty.

3.3 Factorial

The factorial function is defined in matita in a recursive way:

```
let rec fact n :=
  match n with [ O ⇒ 1 | S m ⇒ fact m ∗ S m].
```

For our purposes we need however to exploit its representation in the form of a big product (a theorem that is easily proved by induction on n):

```
theorem eq_fact_pi:∀n.
  fact n = ∏_{i ∈[1,S n[ } i.
qed.
```

We have now the right machinery to prove Chebyshev's decomposition of the factorial function of equation 2. Formally:

```
theorem fact_pi_decomp: ∀n.
fact n = ∏_{ p < S n | primeb p}(∏_{i < log p n} (pˆ(n /(pˆ(S i))))).
```

The proof (130 lines) is summarized by the following chain of equations:

$$\begin{aligned} n! &= \prod_{1\le m\le n} m \\ &= \prod_{1\le m\le n}\prod_{p\le m}\prod_{\substack{i<\log_p m\\ p^{i+1}|m}} p \\ &= \prod_{p\le n}\prod_{p\le m\le n}\prod_{\substack{i<\log_p m\\ p^{i+1}|m}} p \\ &= \prod_{p\le n}\prod_{i<\log_p n}\prod_{\substack{m\le n\\ p^{i+1}|m}} p \\ &= \prod_{p\le n}\prod_{i<\log_p n} p^{n/p^{i+1}} \end{aligned}$$

In particular, for $2n$ we have:

$$(2n)! = \prod_{p\le 2n}\prod_{i<\log_p 2n} p^{2n/p^{i+1}} \qquad (4)$$

Let us decompose $\frac{2n}{p^{i+1}}$ in the following way:

$$\frac{2n}{p^{i+1}} = 2\frac{n}{p^{i+1}} + \left(\frac{2n}{p^{i+1}} \mod 2\right)$$

Moreover, if $n \le p$ or $\log_p n \le i$ we have

$$\frac{n}{p^{i+1}} = 0$$

Hence, if we define

$$B(n) = \prod_{\mathbf{p} \le n} \prod_{\mathbf{i} < \log_p n} p^{(n/p^{i+1} \mod 2)}$$

equation (4) becomes

$$(2n)! = n!^2 B(2n) \tag{5}$$

$B(2n)$ is thus the binomial coefficient $\binom{2n}{n}$.
The formal definition of B in Matita looks as follows:

```
definition B :=λn.
  ∏_{p < S n | primeb p}(∏_{i < log p n} (p^((n /(p^(S i))) \mod 2))).
```

It is worth to remark that the definition is computable; these are some typical values of B, for small integers:

```
example B_3: B 3 = 6. // qed.
example B_4: B 4 = 6. // qed.
example B_5: B 5 = 30. // qed.
example B_6: B 6 = 20. // qed.
example B_7: B 7 = 140. // qed.
example B_8: B 8 = 70. // qed.
```

Equation 4 is expressed by the following theorem:

```
theorem eq_fact_B:∀n. 1 < n → (2∗n)! = n!^2 ∗ B(2∗n).
```

3.4 Upper and lower bounds for B

The next step is to provide upper and lower bounds for B.

It is well known that, for any n, $(2n)! \le 2^{2n-1} n!^2$. For technical reasons, we need however a slightly stronger result, namely,

$$(2n)! \le 2^{2n-2} n!^2$$

that holds for any n larger than 4.

```
theorem lt_4_to_fact: ∀n.4<n → (2∗n)! ≤ 2^((2∗n)−2)∗n!∗n!.
```

The proof is by induction on n.
The base case amounts to check that $10! \le 2^8 5!^2$, that can be proved by a mere computation (after a few simplifications).

In the inductive case

$$\begin{aligned}(2\cdot(n+1))! &= (2n+2)(2n+1)(2n)!\\ &\leq (2n+2)(2n+1)2^{2n-2}n!^2\\ &\leq (2n+2)(2n+2)2^{2n-2}n!^2\\ &= 2^{2n}(n+1)!^2\end{aligned}$$

So, by equation (5), we conclude that, for any n

$$B(2n) \leq 2^{2n-1} \tag{6}$$

and when n is larger than 4,

$$B(2n) \leq 2^{2n-2} \tag{7}$$

```
theorem lt_4_to_le_B_exp: ∀n.4 < n → B (2*n) ≤ 2^((2*n)−2).
```

Similarly, we establish the following lower bound for the factorial function:

```
theorem exp_to_fact2: ∀n.O < n → 2^(2*n)*n!^2 ≤ 2*n*(2*n)!.
```

The proof is by induction on n. For $n=1$ both sides reduce to 4. For $n>1$,

$$\begin{aligned}2^{2n+2}(n+1)!^2 &= 4(n+1)^2 2^{2n} n!\\ &= 4(n+1)^2 2n(2n)!\\ &= 4(n+1)(n+1)2n(2n)!\\ &\leq 4(n+1)(n+1)(2n+1)(2n)!\\ &= 2(n+1)(2n+2)(2n+1)(2n)!\\ &= 2(n+1)(2n+2)!\end{aligned}$$

By equation (5), we finally obtain our lower bound for B:

```
theorem le_exp_B: ∀n. O < n. 2^(2*n) ≤ 2 * n * B (2*n).
```

Since for any n, $2n \leq 2^n$, we also have

$$2^n \leq B(2n) \tag{8}$$

but this is less precise and less useful.

4. CHEBYSHEV'S Ψ FUNCTION

Let `prim n` be the function that counts the number of prime numbers below n (included). A possible definition of `prim` is the following:

```
definition prim :=λn. ∑_{i < S n | primeb i} 1.
```

Let us now consider the following function

$$\Psi(n) = \prod_{p \leq n} p^{\log_p n}$$

where the product ranges over all *primes* less or equal to n.

```
definition Psi: nat → nat :=
  λn.∏_{p < S n | primeb p} (pˆ(log p n)).
```

Actually, Chebyshev's ψ function is the naperian logarithm of our function Ψ, but as we mentioned in the introduction, we prefer to avoid the use of logarithms as far as possible.
As usual, the function Ψ is computable:

```
example Psi_1: Psi 1 = 1. // qed.
example Psi_2: Psi 2 = 2. // qed.
example Psi_3: Psi 3 = 6. // qed.
example Psi_4: Psi 4 = 12. // qed.
```

The relation between Ψ and π should be clear; since

$$p^{\log_p n} \leq n$$

we immediately get

```
theorem le_Psil: ∀n. Psi n ≤ nˆ(prim n).
```

Moreover, since $n < a^{\log_a n+1}$, we also have $n < a^{2\log_a n}$, and hence

```
theorem lePsi_r2: ∀n. nˆ(prim n) ≤ Psi n ∗ Psi n.
```

Let us now rewrite `Psi n` in the following equivalent form:

```
definition Psi': nat → nat :=
  λn. ∏_{p < S n | primeb p} (∏_{i < log p n} p).
```

We can easily prove

```
theorem eq_Psi_Psi': ∀n.Psi n = Psi' n.
```

and it is then clear, by the definition of B and Psi', that

```
theorem le_B_Psi: ∀n. B n ≤ Psi n.
```

Hence, the lower bound for B immediately gives a lower bound for Ψ, namely

$$2^n \leq 2^{2n}/2n \leq \Psi(2n) \tag{9}$$

For the upper bound, let us first observe that

$$\Psi(2n) = \Psi(n) \prod_{p \leq 2n} \prod_{i < \log_p 2n} p^{j(n,p,i)} \tag{10}$$

where $j(n,p,i)$ is 1 if $n < p^{i+1}$ and 0 otherwise. Indeed

$$\begin{aligned}\Psi(2n) &= \prod_{p\leq 2n}\prod_{i<\log_p 2n} p \\ &= \left(\prod_{p\leq 2n}\prod_{i<\log_p 2n} p^{j(n,p,i)}\right)\left(\prod_{p\leq 2n}\prod_{i<\log_p 2n} p^{1-j(n,p,i)}\right) \\ &= \Psi(n)\prod_{p\leq 2n}\prod_{i<\log_p 2n} p^{j(n,p,i)}\end{aligned}$$

Formally, this is stated by the following result (the formal proof takes about 50 lines).

theorem eq_Psi_2_n: ∀n.O < n →
Psi(2∗n) =
 ∏_{p <S (2∗n) | primeb p}(∏_{i <log p (2∗n)} (p^(leb (S n) (p^(S i))))) ∗ Psi n.

where the boolean expression `(leb (S n) (p^(S i)))` is automatically converted to an integer through an implicit coercion mapping `false` to 0 and `true` to 1. Then, observe that

$$\prod_{\mathbf{p}\leq 2n}\prod_{\mathbf{i}<\log_p 2n} p^{j(n,p,i)} \leq B(2n) = \prod_{\mathbf{p}\leq 2n}\prod_{\mathbf{i}<\log_p 2n} p^{(2n/p^{i+1} \mod 2)} \tag{11}$$

since if $n < p^{i+1}$ then $2n/p^{i+1} \mod 2 = 1$. So we may conclude

theorem le_Psi_BPsi: ∀n. Psi(2∗n) ≤ B(2∗n)∗Psi n.

In particular, for any n

$$\Psi(2n) \leq 2^{2n-1}\Psi(n) \tag{12}$$

and for $4 < n$

$$\Psi(2n) \leq 2^{2n-2}\Psi(n) \tag{13}$$

We may now use inductively these estimates to prove

theorem le_Psi_exp: ∀n. Psi(n) ≤ 2^((2 ∗ n) −3).

For the proof, we need the monotonicity of Ψ, that is easily proved:

$$\Psi(n) = \prod_{\mathbf{p}\leq n} p^{\log_p n} \leq \prod_{\mathbf{p}\leq n} p^{\log_p (n+1)} \leq \prod_{\mathbf{p}\leq n+1} p^{\log_p (n+1)} = \Psi(n+1) \tag{14}$$

Then we check that the property holds for any $n \leq 8$, which can be done by direct computation. If n is larger than 8 we distinguish two cases, according to n is even or odd. We only consider the case $n = 2m+1$ that is the most interesting one. Observe first that $8 < 2m+1$ implies $4 < m$. Then we have:

$$\begin{aligned}\Psi(n) &= \Psi(2m+1) \\ &\leq \Psi(2m+2) \\ &\leq 2^{2m}\Psi(m+1) \\ &\leq 2^{2m}2^{2(m+1)-3} \\ &\leq 2^{2(2m+1)-3}\end{aligned}$$

In conclusion, we have

$$2^{n/2} \leq \Psi(n) \leq n^{\pi(n)} \leq \Psi(n)^2 \leq 2^{4n-6} \leq 2^{4n} \tag{15}$$

5. BERTRAND'S POSTULATE

Our approach to Chebyshev's theorem, as most modern presentations of the subject, essentially follows Chebyshev's original idea, but in a rudimentary form which provides a result that is numerically less precise, though of a similar nature. In particular, Chebyshev was able to prove the asymptotic estimates

$$(c_1 + o(1))\frac{n}{\log n} \leq \pi(n) \leq (c_2 + o(1))\frac{n}{\log n} \qquad (n \to \infty)$$

with

$$\begin{aligned} c_1 &= \log(2^{1/2}3^{1/3}5^{1/5}30^{-1/30}) \approx 0.92129 \\ c_2 &= 6/5c_1 \approx 1.10555 \end{aligned}$$

In particular, since $c_2 < 2c_1$, this implies that

$$\pi(2n) > \pi(n)$$

for all large n. Actually, by direct computation, Chebyshev proved that the inequality remains true for all n, confirming for the first time *Bertrand's postulate.*

With our rough estimates, we could only prove the existence of a prime number between n and $8n$, for n sufficiently large. There exists however an alternative approach to the proof of Bertrand's postulate due to Erdös [10] (see also [13], p. 344) that is well suited to a formal encoding in arithmetics[1].

5.1 Bertrand's predicate

Let us define the following Bertands' predicate:

```
definition bertrand :=λn. ∃p.n < p ∧ p ≤ 2∗n ∧ prime p.
```

Bertrand's postulate is the conjecture that the previous predicate holds for any natural number n.

Erdös proof is essentially a proof by contradiction; supposing the predicate to be false, he derives an absurdity. The negation of Bertrand's predicate is

```
definition not_bertrand :=λn.∀p.n < p → p ≤ 2∗n → ¬ (prime p).
```

Since the quantification is bound, the predicate is *decidable*, that means that, even in a system like Matita which has a *constructive* nature, you can prove

```
theorem not_not_bertrand_to_bertrand: ∀n. ¬ (not_bertrand n) → bertrand n.
```

[1]Erdös argument was already exploited by Théry in his proof of Bertrand postulate [21]; however he failed to provide a fully arithmetical proof, being forced to make use of the (classical, axiomatic) library of Coq reals to solve the remaining inequalities.

5.2 Chebyshev's θ function

In order to use Erdös argument we need to introduce another important function introduced by Chebyshev and largely used in analytic number theory [1]: the so called theta-function.

In particular, we shall work with the following variant (as for the Ψ function, Chebyshev's function is the naperian logarithm of our function)

```
definition theta: nat → nat :=λn. ∏_{p < S n| primeb p} p.
```

It is clear that for any n, $\theta(n) \leq \Psi(n)$, so our upper bound for Ψ is also an upper bound for θ. Since however providing an upper bound for $\theta(n)$ is much simpler, we recall this direct argument here.
Let us split $\theta(2n+1)$ in two parts:

```
theorem theta_split: ∀m. theta (S (2∗m))
= (∏_{p ∈[S(S m)),S (S (2∗m))[ | primeb p} p)∗theta (S m).
```

Let us now consider the following binomial coefficient $M(m) = \binom{2m+1}{m} = \frac{(2m+1)!}{m!m!}$

```
definition M :=λm.bc (S(2∗m)) m.
```

Using the binomial law, it is not difficult to prove that

```
theorem lt_M: ∀m. O < m → M m < 2ˆ(2∗m).
```

Every prime p between $m+2$ and $2m+1$ must be a divisor of $(2m+1)!$ and hence of $M(m)$, so

```
theorem divides_pi_p_M: ∀m.
  ∏_{p ∈[S(S m)),S (S (2∗m))[ | primeb p} p | (M m).
qed.
```

As a corollary,

```
theorem le_pi_p_M: ∀m.
  ∏_{p ∈[S(S m)),S (S (2∗m))[ | primeb p} p ≤ (M m).
```

and also

```
theorem le_theta_M_theta: ∀m.
  theta (S(2∗m)) ≤ (M m)∗theta (S m).
```

Let us also observe that, since a prime larger than 2 cannot be *even*,

```
theorem theta_pred: ∀n. 1 < n → theta (2∗n) = theta (pred (2∗n)).
```

We are now ready to prove the following statement:

```
theorem le_theta: ∀m.theta m ≤ 2ˆ(2∗m).
```

The proof is by (generalized) induction on m. Let us suppose the statement is true for any $n < m$, and let us prove it for m. Either m is even or it is odd (the two cases are similar, and we only consider the first one (the other case is similar, using theorem `theta_pred`). Let us suppose $m = 2a+1$. Then

$$\theta(2a+1) \leq (Ma) * theta(Sa) \leq 2^{2a} \cdot 2^{2(a+1)} = 2^{2(2a+1)}$$

5.3 Erdös argument

Let

$$k(n,p) = \sum_{i<\log_p n} (n/p^{i+1} \mod 2)$$

Then, B can also be written as

$$B(n) = \prod_{p \leq n} p^{k(n,p)}$$

We now split this product in two parts B_1 and B_2, according to $k(n,p) = 1$ or $k(n,p) > 1$.

```
definition k :=λn,p. ∑_{i < log p n}((n/(p^(S i))\mod 2)).

definition Bk :=λn. ∏_{p < S n | primeb p}(p^(k n p)).

theorem eq_B_Bk: ∀n. B n = Bk n.

definition B1 :=λn. ∏_{p < S n | primeb p}(p^(leb (k n p) 1)∗ (k n p)).

definition B2 :=λn. ∏_{p < S n | primeb p}(p^(leb 2 (k n p))∗ (k n p)).

theorem eq_Bk_B1_B2: ∀n. Bk n = B1 n ∗ B2 n.
```

Suppose that Bertrand postulate is $false$, hence there is no prime between n and $2n$. Moreover, if $\frac{2n}{3} < p \leq n$, then $2n/p = 2$ and for $i > 1$ and $n \geq 6$ $2n/p^i = 0$ since

$$2n \leq \left(\frac{2n}{3}\right)^2 \leq p^i$$

so $k(2n,p) = 0$. Summing up, under the assumption that Bertrand postulate is *false*,

$$B_1(2n) = \prod_{\substack{p \leq 2n \\ k(2n,p)=1}} p = \prod_{p \leq 2n/3} p \leq \theta(2n/3) \leq 2^{2(2n/3)} \tag{16}$$

```
le_B1_theta:∀n.6 ≤n →not_bertrand n →B1 (2∗n) ≤theta (2 ∗ n / 3).
```

On the other side, note that $k(n,p) \leq \log_p n$, so if $k(2n,p) \geq 2$ we also have $\log_p 2n \geq 2$ that implies $p \leq \sqrt{2n}$. So

$$B_2(2n) = \prod_{\substack{p \leq 2n \\ 2 \leq k(2n,p)}} p^{k(2n,p)} \leq \prod_{p \leq \sqrt{2n}} 2n = (2n)^{\pi(\sqrt{2n})} \tag{17}$$

For $n \geq 15$, $\pi(n) \leq n/2 - 1$. Hence, for any $n \geq 2^7 > 15^2$, we have

$$B_2(2n) \leq (2n)^{\sqrt{2n}/2-1}$$

Formally:

theorem le_B2_exp: ∀n. 2ˆ7 ≤ n → B2 (2∗n) ≤ (2∗n)ˆ(pred(sqrt(2∗n)/2)).

Putting everything together, supposing Bertrand's postulate is false, we would have, for any $n \geq 2^7$

$$\begin{aligned} 2^{2n} &\leq 2nB(2n) \\ &= 2nB_1(2n)B_2(2n) \\ &\leq 2^{2(2n/3)}(2n)^{\sqrt{2n}/2} \end{aligned}$$

Observe that

$$2^{2n} = 2^{2(2n/3)}2^{2n/3}$$

so, by cancellation,

$$2^{2n/3} \leq (2n)^{\sqrt{2n}/2}$$

and taking logarithms

$$\frac{2n}{3} \leq \frac{\sqrt{2n}}{2}(\log(2n) + 1)$$

theorem not_bertrand_to_le2:
∀n.2ˆ7 ≤ n → not_bertrand n → 2∗n / 3 ≤ (sqrt(2∗n)/2)∗S(log 2 (2∗n)).

We want to find an integer m such that for all values larger than m the previous equation is false; moreover, the integer m must be sufficiently small to allow to check the remaining cases automatically in a feasible time. The trouble is that, in our case, all operations are discrete, and it is not so easy to reason about inequalities in arithmetics.
We must prove

$$\frac{\sqrt{2n}}{2}(\log(2n) + 1) < \frac{2n}{3}$$

The strict inequality is the first source of trouble, so we prove instead

$$\frac{\sqrt{2n}}{2}(\log(2n) + 1) \leq \frac{2n}{4}$$

using the fact that

$$\frac{n}{m+1} < \frac{n}{m}$$

for any $n \geq m^2$ (in our case, $n \geq 8$). By means of simple manipulations, it is easy to transform the last equation in the following simpler form

$$2(\log(2n) + 1) \leq \sqrt{2n}$$

or equivalently

$$2(\log n + 2)^2 \leq n$$

We now use the fact that for any $a > 0$ and any $n \geq 4a$

$$2^a n^2 \leq 2^n$$

to get, for any $n \geq 2^8$

$$2(\log n + 2)^2 \leq 4(\log n)^2 = 2^2(\log n)^2 \leq 2^{\log n} \leq n$$

In conlcusion, we are able to prove

```
lemma sqrt_bound: ∀n. 2ˆ8 ≤ n → 2∗(S(log 2 (2∗n))) ≤ sqrt (2∗n).
```

and hence, as a simple corollary

```
theorem bertrand_up: ∀n. 2ˆ8 ≤ n → bertrand n.
```

5.4 Automatic check

To complete the proof, we have still to check that Bertrand's postulate remains true for all integers less then 2^8. This is very simple in principle: it is sufficient to

(1) generate the list of all primes up to the first prime larger than 2^8 (in reverse order)
(2) check that for any pair p_i, p_{i+1} of consecutive primes in such list, $p_i < 2p_{i+1}$

Both the generation of the list and its check can be performed automatically. All we have to do is to prove that our algorithm for generating primes is correct and complete, and that the previous check is equivalent to Bertrand's postulate, on the given interval.

With respect to the version discussed in [3], this part of the proof has been completely rewritten and substantially simplified.

5.5 The list of primes

To compute the list of prime numbers, we use the following simple algorithm. We first define a function `list_divides l n` that returns `true` if `l` contains at least a divisor of `n` and `false` otherwise.

```
let rec list_divides l n :=
  match l with
  [ nil ⇒ false
  | cons (m:nat) (tl: list nat) ⇒ orb (dividesb m n) (list_divides tl n) ].
```

```
let rec lprim m i acc :=
  match m with
   [ O ⇒ acc
   | S m1 ⇒ match (list_divides acc i) with
      [ true ⇒ lprim m1 (S i) acc
      | false ⇒ lprim m1 (S i) (acc@[i]) ]].

definition list_of_primes :=λn. lprim n 2 [].
```

For instance,

```
example lprim_ex: lprim 8 2 [ ] = [2; 3 ; 5 ; 7 ]. // qed.
```

Let us define a predicate `primes_below l n` stating that the list `l` is the list of *all* primes strictly below `n`.

```
definition all_primes :=λl.∀p. mem nat p l → prime p.
definition all_below :=λl,n.∀p. mem nat p l → p < n.
definition primes_all :=λl,n. ∀p. prime p → p < n → mem nat p l.

definition primes_below :=λl,n. all_primes l ∧ all_below l n ∧ primes_all l n.
```

Then, by the definition of prime number, it is easy to prove that

```
lemma ld_to_prime: ∀i,acc. 1 < i →
  primes_below acc i → list_divides acc i = false → prime i.
```

Using the previous property we can then prove the main invariant of the `lprim` function, namely:

```
lemma lprim_invariant: ∀n,i,acc. 1 < i →
  primes_below acc i → primes_below (lprim n i acc) (n+i).
```

Since moreover

```
lemma primes_below2: primes_below [] 2.
```

we obtain

```
lemma primes_below_lop: ∀n. primes_below (list_of_primes n) (n+2).
```

5.6 Checking

We perform the following test on the list:

```
let rec checker l :=
  match l with
  [ nil ⇒ true
  | cons hd tl ⇒ match tl with
     [ nil ⇒ true
     | cons hd1 tl1 ⇒ leb (S hd) hd1 ∧ leb hd1 (2∗hd) ∧ checker tl
    ]
  ].
```

Then, working by induction on the list, it is easy to prove the following property:

```
theorem checker_spec: ∀tl,a,l. checker l = true → l = a:: tl →
  ∀p. mem ? p tl → ∃pp. mem nat pp l ∧ pp < p ∧ p ≤ 2∗pp.
```

The last step consists in proving that the previous condition is enough to entail Bertrand's property for all n below the maximum prime in the list:

```
theorem primes_below_to_bertrand:
∀pm,l.prime pm → primes_below l (S pm) →
  (∀p. mem ? p l → 2 < p → ∃pp. mem ? pp l ∧ pp < p ∧ p ≤ 2∗pp)
    → ∀n.0 < n → n < pm → bertrand n.
```

Finally we have just to compute the list of primes below $2^8 + 1$ (that is prime), and apply the previous results to conclude

lemma bertrand_down : $\forall$n.O < n $\rightarrow$ n $\leq$ 2ˆ8 $\rightarrow$ bertrand n.

The generation of the list of primes and its check takes less than 5 seconds. Putting together this result with theorem `bertrand_up`, we finally get

theorem bertrand : $\forall$n.O < n $\rightarrow$ bertrand n.

6. CONCLUSIONS

In this paper we presented the formalization, in the Matita interactive theorem prover, of some results about Chebyshev's functions ψ and θ, comprising a proof of Bertrand's postulate following the approach of Erdös [10]. The subject, even from the point of view of formalization, is not completely original (see e.g. [21, 18]). For us, it was mostly an opportunity to present and discuss the many novelties recently introduced in Matita, comparing the old development relative to version 0.5.2 of the system [5] and described in [3], with the current development relative to version 0.99 of the system.

While porting it to the new version of the system, the arithmetical library has undergone a deep revisitation, integration and restructuring, comprising e.g. the new library of big operators discussed in section 2.

The table in Figure 2 makes a comparison between the two versions of the proof: for each file the first dimension is the number of lines, while the second one is the number of theorems.

file	Matita 0.5.2		Matita 0.99	
logarithms	413	(20)	223	(21)
square root	217	(13)	221	(19)
binomial coeff.	259	(9)	192	(12)
order of primes	656	(33)	411	(37)
big operators	978	(30)	425	(27)
sigma and pi	526	(26)	188	(9)
factorial	325	(14)	145	(12)
chebyshev's theta	486	(13)	213	(13)
chebishev's psi	294	(11)	143	(13)
factorization	927	(25)	629	(32)
psi bounds	1123	(37)	507	(30)
bertrand (up)	683	(18)	446	(27)
bertrand (down)	526	(22)	240	(19)
total	7413	(271)	3983	(271)

Fig. 1. Size of the developments

In passing from version 0.5.2 to version 0.99 the size of the development has been essentially reduced to one half, passing from an average of 28 lines per theorem to less than 15.

This is essentially due to two factors: the first one is the new compact syntax for tactics, in the spirit of the SSReflect language [12]; the second one is the support for *small scale automation* [7, 6] that relieves the user from spelling out a lot of trivial steps. The difference can be appreciated by the following table, summarizing the number of invocations of the most frequent tactics in the two versions of the above files.

	Matita 0.5.2		Matita 0.99	
tactic	name	no.	name	no.
application	apply	2203	@	1792
	assumption	779		
rewriting	rewrite	1110	< / >	984
	reflexivity	244		
simplification	simplify	255	normalize	122
			whd	76
introduction	intro/intros	435	#	1904
elimination	cases	306	cases	190
	elim	131	elim	92
			*	62
cut	cut	89	cut	148
automation	auto	10	//	943

Fig. 2. Number of tactics invocations

The 943 invocations of the auto tactic in version 0.99 not only replace the 779 invocations of "assumption" and the 244 invocations of "reflexivity", but are also responsible of the sensible reduction in the number of applications and rewritings. The increase in the number of invocations of the introduction tactic is merely due to the fact that the old tactic performed multiple introductions, delegating to the system the choice of names, while the new tactic is nominal, and handles a single variable at a time.

In Hardy's book [13], the proof of Bertrand's postulate takes 42 lines, while Chebyshev's theorem takes precisely three pages (90 lines). With the new version of the proof, this gives a de Brujin factor between 8 and 10, that is in line with other formalization case studies in the realm of arithmetics [8, 15, 14].

References

[1] T. M. Apostol. *Introduction to Analytic Number Theory.* Springer Verlag, 1976.

[2] Andrea Asperti and Cristian Armentano. A page in number theory. *Journal of Formalized Reasoning*, 1:1–23, 2008.

[3] Andrea Asperti and Wilmer Ricciotti. About the formalization of some results by Chebyshev in number theory. In *Proc. of TYPES'08*, volume 5497 of *LNCS*, pages 19–31. Springer-Verlag, 2009.

[4] Andrea Asperti, Wilmer Ricciotti, Claudio Sacerdoti Coen, and Enrico Tassi. Hints in unification. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *TPHOLs*, volume 5674 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 2009.

[5] Andrea Asperti, Wilmer Ricciotti, Claudio Sacerdoti Coen, and Enrico Tassi. The Matita interactive theorem prover. In *Proceedings of the 23rd International Conference on Automated Deduction (CADE-2011), Wroclaw, Poland*, volume 6803 of *LNCS*, 2011.

[6] Andrea Asperti and Enrico Tassi. Smart matching. In *Intelligent Computer Mathematics, 10th International Conference, AISC 2010, 17th Symposium, Calculemus 2010, and 9th International Conference, MKM 2010, Paris, France, July 5-10, 2010. Proceedings*, volume 6167 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2010.

[7] Andrea Asperti and Enrico Tassi. Superposition as a logical glue. *EPTCS*, 53:1–15, 2011.

[8] Jeremy Avigad, Kevin Donnelly, David Gray, and Paul Raff. A formally verified proof of the prime number theorem. *ACM Trans. Comput. Log.*, 9(1), 2007.

[9] Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Pasca. Canonical big operators. In *TPHOLs*, pages 86–101, 2008.

[10] P .Erdös. Beweis eines satzes von tschebyschef. *Acta Scientifica Mathematica*, 5:194–198, 1932.

[11] Santiago Escobar, José Meseguer, and Prasanna Thati. Narrowing and rewriting logic: from foundations to applications. *Electr. Notes Theor. Comput. Sci.*, 177:5–33, 2007.

[12] Georges Gonthier and Assia Mahboubi. An introduction to small scale reflection in coq. *Journal of Formalized Reasoning*, 3(2):95–152, 2010.

[13] G. H. Hardy. *An introduction to the theory of numbers.* Oxford University Press, 1938.

[14] John Harrison. A formalized proof of dirichlet's theorem on primes in arithmetic progression. *Journal of Formalized Reasoning*, 2:63–83, 2009.

[15] John Harrison. Formalizing an analytic proof of the prime number theorem. *J. Autom. Reasoning*, 43(3):243–261, 2009.

[16] G. J. O. Jameson. *The Prime Number Theorem*, volume 53 of *London Mathematical Society Student Texts.* Cambridge University Press, 2003.

[17] Srinivasa Ramanujan. A proof of bertrand's postulate. *Journal of the Indian Mathematical Society*, 11:181–182, 1919.

[18] Riccardi. Pocklington's theorem and bertrand's postulate. *Formalized Mathematics*, 14(2):47–52, 2006.

[19] Claudio Sacerdoti Coen and Enrico Tassi. Nonuniform coercions via unification hints. volume 53 of *EPTCS*, pages 16–29, 2011.

[20] G. Tenenbaum and M. Mendès France. *The Prime Numbers and Their Distribution.* American Mathematical Society, 2000.

[21] Laurent Théry. Proving pearl: Knuth's algorithm for prime numbers. In *Proceedings of TPHOLs'03*, LNCS, pages 304–318. Springer Verlag, 2003.