Furstenberg's Topology And His Proof of the Infinitude of Primes

Manuel Eberl

March 17, 2025

Abstract

This article gives a formal version of Furstenberg's topological proof of the infinitude of primes. He defines a topology on the integers based on arithmetic progressions (or, equivalently, residue classes). Using some fairly obvious properties of this topology, the infinitude of primes is then easily obtained.

Apart from this, this topology is also fairly 'nice' in general: it is second countable, metrizable, and perfect. All of these (well-known) facts are formally proven, including an explicit metric for the topology given by Zulfeqarr.

Contents

1 Furstenberg's topology and his proof of the infinitude of primes

theory *Furstenberg-Topology*
  imports
    HOL−Real-Asymp.Real-Asymp
    HOL−Analysis.Analysis
    HOL−Number-Theory.Number-Theory
begin

This article gives a formal version of Furstenberg's topological proof of the infinitude of primes [2]. He defines a topology on the integers based on arithmetic progressions (or, equivalently, residue classes).

Apart from yielding a short proof of the infinitude of primes, this topology is also fairly 'nice' in general: it is second countable, metrizable, and perfect. All of these (well-known) facts will be formally proven below.

1.1 Arithmetic progressions of integers

We first define 'bidirectional infinite arithmetic progressions' on $\mathbb{Z}$ in the sense that to an integer a and a positive integer b, we associate all the integers x such that $x \equiv a \pmod{b}$, or, equivalently, $\{a + nb \mid n \in \mathbb{Z}\}$.

definition *arith-prog* :: *int* ⇒ *nat* ⇒ *int set* **where**
  arith-prog a b = {*x*. [*x* = *a*] (*mod int b*)}

lemma *arith-prog-0-right* [*simp*]: *arith-prog a 0* = {*a*}
  by (*simp add*: *arith-prog-def*)

lemma *arith-prog-Suc-0-right* [*simp*]: *arith-prog a* (*Suc 0*) = *UNIV*
  by (*auto simp*: *arith-prog-def*)

lemma *in-arith-progI* [*intro*]: [*x* = *a*] (*mod b*) ⟹ *x* ∈ *arith-prog a b*
  by (*auto simp*: *arith-prog-def*)

Two arithmetic progressions with the same period and noncongruent starting points are disjoint.

lemma *arith-prog-disjoint*:
  assumes [*a* ≠ *a'*] (*mod int b*) **and** *b* > *0*
  shows *arith-prog a b* ∩ *arith-prog a' b* = {}
  using *assms* **by** (*auto simp*: *arith-prog-def cong-def*)

Multiplying the period gives us a subset of the original progression.

lemma *arith-prog-dvd-mono*: *b dvd b'* ⟹ *arith-prog a b'* ⊆ *arith-prog a b*
  by (*auto simp*: *arith-prog-def cong-dvd-modulus*)

The following proves the alternative definition mentioned above.

lemma *bij-betw-arith-prog*:
  assumes $b > 0$
  shows *bij-betw* $(\lambda n.\ a + int\ b * n)$ *UNIV* (*arith-prog a b*)
proof (*rule bij-betwI*[*of - - -* $\lambda x.\ (x - a)$ *div int b*], *goal-cases*)
  case *1*
  thus *?case*
   by (*auto simp*: *arith-prog-def cong-add-lcancel-0 cong-mult-self-right mult-of-nat-commute*)
next
  case *4*
  thus *?case*
    by (*auto simp*: *arith-prog-def cong-iff-lin*)
qed (*use* ‹$b > 0$› **in** ‹*auto simp*: *arith-prog-def*›)

lemma *arith-prog-altdef*: *arith-prog a b* = *range* $(\lambda n.\ a + int\ b * n)$
proof (*cases* $b = 0$)
  case *False*
  thus *?thesis*
    using *bij-betw-arith-prog*[*of b*] **by** (*auto simp*: *bij-betw-def*)
qed *auto*

A simple corollary from this is also that any such arithmetic progression is infinite.

lemma *infinite-arith-prog*: $b > 0 \Longrightarrow$ *infinite* (*arith-prog a b*)
  using *bij-betw-finite*[*OF bij-betw-arith-prog*[*of b*]] **by** *simp*

1.2 The Furstenberg topology on $\mathbb{Z}$

The typeclass-based topology is somewhat nicer to use in Isabelle/HOL, but the integers, of course, already have a topology associated to them. We therefore need to introduce a type copy of the integers and furnish them with the new topology. We can easily convert between them and the 'proper' integers using Lifting and Transfer.

typedef *fbint* = *UNIV* :: *int set*
  morphisms *int-of-fbint fbint* **..**

setup-lifting *type-definition-fbint*

lift-definition *arith-prog-fb* :: *int* ⇒ *nat* ⇒ *fbint set* **is** *arith-prog* **.**

instantiation *fbint* :: *topological-space*
begin

Furstenberg defined the topology as the one generated by all arithmetic progressions. We use a slightly more explicit equivalent formulation that exploits the fact that the intersection of two arithmetic progressions is again an arithmetic progression (or empty).

lift-definition *open-fbint* :: *fbint set* ⇒ *bool* **is**

$\lambda U.\ (\forall x \in U.\ \exists b>0.\ \textit{arith-prog}\ x\ b \subseteq U)$ **.**

We now prove that this indeed forms a topology.

instance proof
  show *open* (*UNIV* :: *fbint set*)
    by *transfer auto*
next
  fix *U V* :: *fbint set*
  assume *open U* **and** *open V*
  show *open* $(U \cap V)$
  proof (*use* ‹*open U*› ‹*open V*› **in** *transfer*, *safe*)
    fix *U V* :: *int set* **and** *x* :: *int*
    assume *U*: $\forall x \in U.\ \exists b>0.\ \textit{arith-prog}\ x\ b \subseteq U$ **and** *V*: $\forall x \in V.\ \exists b>0.\ \textit{arith-prog}\ x\ b \subseteq V$
    assume *x*: $x \in U$ $x \in V$
    from *U x* **obtain** *b1* **where** *b1*: $b1 > 0$ $\textit{arith-prog}\ x\ b1 \subseteq U$ **by** *auto*
    from *V x* **obtain** *b2* **where** *b2*: $b2 > 0$ $\textit{arith-prog}\ x\ b2 \subseteq V$ **by** *auto*
    from *b1 b2* **have** $\textit{lcm}\ b1\ b2 > 0$ $\textit{arith-prog}\ x\ (\textit{lcm}\ b1\ b2) \subseteq U \cap V$
      using *arith-prog-dvd-mono*[*of b1 lcm b1 b2 x*] *arith-prog-dvd-mono*[*of b2 lcm b1 b2 x*]
      by (*auto simp*: *lcm-pos-nat*)
    thus $\exists b>0.\ \textit{arith-prog}\ x\ b \subseteq U \cap V$ **by** *blast*
  qed
next
  fix *F* :: *fbint set set*
  assume $*$: $\forall U \in F.\ \textit{open}\ U$
  show *open* $(\bigcup F)$
  proof (*use* $*$ **in** *transfer*, *safe*)
    fix *F* :: *int set set* **and** *U* :: *int set* **and** *x* :: *int*
    assume *F*: $\forall U \in F.\ \forall x \in U.\ \exists b>0.\ \textit{arith-prog}\ x\ b \subseteq U$
    assume $x \in U$ $U \in F$
    with *F* **obtain** *b* **where** *b*: $b > 0$ $\textit{arith-prog}\ x\ b \subseteq U$ **by** *blast*
    with ‹$U \in F$› **show** $\exists b>0.\ \textit{arith-prog}\ x\ b \subseteq \bigcup F$
      by *blast*
  qed
qed

end

Since any non-empty open set contains an arithmetic progression and arithmetic progressions are infinite, we obtain that all nonempty open sets are infinite.

lemma *open-fbint-imp-infinite*:
  fixes *U* :: *fbint set*
  assumes *open U* **and** $U \neq \{\}$
  shows *infinite U*
  using *assms*
proof *transfer*
  fix *U* :: *int set*

assume $*$: $\forall x{\in}U.\ \exists b{>}0.\ \mathit{arith\text{-}prog}\ x\ b \subseteq U$ **and** $U \neq \{\}$
from ‹$U \neq \{\}$› **obtain** x **where** $x \in U$ **by** *auto*
with $*$ **obtain** b **where** b: $b > 0$ $\mathit{arith\text{-}prog}\ x\ b \subseteq U$ **by** *auto*
from b **have** *infinite* (*arith-prog* x b)
using *infinite-arith-prog* **by** *blast*
with b **show** *infinite* U
using *finite-subset* **by** *blast*
qed

lemma *not-open-finite-fbint* [*simp*]:
assumes *finite* (U :: *fbint set*) $U \neq \{\}$
shows $\neg$*open* U
using *open-fbint-imp-infinite assms* **by** *blast*

More or less by definition, any arithmetic progression is open.

lemma *open-arith-prog-fb* [*intro*]:
assumes $b > 0$
shows *open* (*arith-prog-fb* a b)
using *assms*
proof *transfer*
fix a :: *int* **and** b :: *nat*
assume $b > 0$
show $\forall x{\in}\mathit{arith\text{-}prog}\ a\ b.\ \exists b'{>}0.\ \mathit{arith\text{-}prog}\ x\ b' \subseteq \mathit{arith\text{-}prog}\ a\ b$
proof (*intro ballI exI*[*of* - b] *conjI*)
fix x **assume** $x \in \mathit{arith\text{-}prog}\ a\ b$
thus $\mathit{arith\text{-}prog}\ x\ b \subseteq \mathit{arith\text{-}prog}\ a\ b$
using *cong-trans* **by** (*auto simp*: *arith-prog-def*)
qed (*use* ‹$b > 0$› **in** *auto*)
qed

Slightly less obviously, any arithmetic progression is also closed. This can be seen by realising that for a period b, we can partition the integers into b congruence classes and then the complement of each congruence class is the union of the other $b - 1$ classes, and unions of open sets are open.

lemma *closed-arith-prog-fb* [*intro*]:
assumes $b > 0$
shows *closed* (*arith-prog-fb* a b)
proof $-$
have *open* ($-$*arith-prog-fb* a b)
proof $-$
have $-\mathit{arith\text{-}prog\text{-}fb}\ a\ b = (\bigcup i{\in}\{1..{<}b\}.\ \mathit{arith\text{-}prog\text{-}fb}\ (a{+}i)\ b)$
proof (*transfer fixing*: b)
fix a :: *int*
have *disjoint*: $x \notin \mathit{arith\text{-}prog}\ a\ b$ **if** $x \in \mathit{arith\text{-}prog}\ (a + \mathit{int}\ i)\ b$ $i \in \{1..{<}b\}$ **for** x i
proof $-$
have $[a \neq a + \mathit{int}\ i]\ (\mathit{mod}\ \mathit{int}\ b)$
proof
assume $[a = a + \mathit{int}\ i]\ (\mathit{mod}\ \mathit{int}\ b)$

```
        hence [a + 0 = a + int i] (mod int b) by simp
        hence [0 = int i] (mod int b) by (subst (asm) cong-add-lcancel) auto
        with that show False by (auto simp: cong-def)
      qed
     thus ?thesis using arith-prog-disjoint[of a a + int i b] ‹b > 0› that by auto
    qed

    have covering: x ∈ arith-prog a b ∨ x ∈ (⋃ i∈{1..<b}. arith-prog (a + int i)
b) for x
    proof −
      define i where i = nat ((x − a) mod (int b))
      have [a + int i = a + (x − a) mod int b] (mod int b)
        unfolding i-def using ‹b > 0› by simp
      also have [a + (x − a) mod int b = a + (x − a)] (mod int b)
        by (intro cong-add) auto
      finally have [x = a + int i] (mod int b)
        by (simp add: cong-sym-eq)
      hence x ∈ arith-prog (a + int i) b
        using ‹b > 0› by (auto simp: arith-prog-def)
      moreover have i < b using ‹b > 0›
        by (auto simp: i-def nat-less-iff)
      ultimately show ?thesis using ‹b > 0›
        by (cases i = 0) auto
    qed

    from disjoint and covering show − arith-prog a b = (⋃ i∈{1..<b}. arith-prog
(a + int i) b)
      by blast
  qed
  also from ‹b > 0› have open ...
    by auto
  finally show ?thesis .
 qed
 thus ?thesis by (simp add: closed-def)
qed
```

1.3 The infinitude of primes

The infinite of the primes now follows quite obviously: The multiples of any prime form a closed set, so if there were only finitely many primes, the union of all of these would also be open. However, since any number other than ± 1 has a prime divisor, the union of all these sets is simply $\mathbb{Z}\setminus\{\pm 1\}$, which is obviously *not* closed since the finite set $\{\pm 1\}$ is not open.

```
theorem infinite {p::nat. prime p}
proof
  assume fin: finite {p::nat. prime p}
  define A where A = (⋃ p∈{p::nat. prime p}. arith-prog-fb 0 p)
  have closed A
```

```
  unfolding A-def using fin by (intro closed-Union) (auto simp: prime-gt-0-nat)
 hence open (−A)
   by (simp add: closed-def)
 also have A = −{fbint 1, fbint (−1)}
   unfolding A-def
 proof transfer
   show (⋃p∈{p::nat. prime p}. arith-prog 0 p) = − {1, − 1}
   proof (intro equalityI subsetI)
     fix x :: int assume x: x ∈ −{1, −1}
     hence |x| ≠ 1 by auto
     show x ∈ (⋃p∈{p::nat. prime p}. arith-prog 0 p)
     proof (cases x = 0)
       case True
       thus ?thesis
         by (auto simp: A-def intro!: exI[of - 2])
     next
       case [simp]: False
       obtain p where p: prime p p dvd x
         using prime-divisor-exists[of x] and ‹|x| ≠ 1› by auto
       hence x ∈ arith-prog 0 (nat p) using prime-gt-0-int[of p]
         by (auto simp: arith-prog-def cong-0-iff)
       thus ?thesis using p
         by (auto simp: A-def intro!: exI[of - nat p])
     qed
   qed (auto simp: A-def arith-prog-def cong-0-iff)
 qed
 also have −(−{fbint 1, fbint (−1)}) = {fbint 1, fbint (−1)}
   by simp
 finally have open {fbint 1, fbint (−1)} .
 thus False by simp
qed
```

1.4 Additional topological properties

Just for fun, let us also show a few more properties of Furstenberg's topology. First, we show the equivalence to the above to Furstenberg's original definition (the topology generated by all arithmetic progressions).

```
theorem topological-basis-fbint: topological-basis {arith-prog-fb a b |a b. b > 0}
  unfolding topological-basis-def
proof safe
  fix a :: int and b :: nat
  assume b > 0
  thus open (arith-prog-fb a b)
    by auto
next
  fix U :: fbint set assume open U
  hence ∀x∈U. ∃b. b > 0 ∧ arith-prog-fb (int-of-fbint x) b ⊆ U
    by transfer
  hence ∃f. ∀x∈U. f x > 0 ∧ arith-prog-fb (int-of-fbint x) (f x) ⊆ U
```

```
    by (subst (asm) bchoice-iff)
  then obtain f where f: ∀x∈U. f x > 0 ∧ arith-prog-fb (int-of-fbint x) (f x) ⊆
U ..
  define B where B = (λx. arith-prog-fb (int-of-fbint x) (f x)) ` U
  have B ⊆ {arith-prog-fb a b |a b. b > 0}
    using f by (auto simp: B-def)
  moreover have ⋃B = U
  proof safe
    fix x assume x ∈ U
    hence x ∈ arith-prog-fb (int-of-fbint x) (f x)
      using f by transfer auto
    with ‹x ∈ U› show x ∈ ⋃B by (auto simp: B-def)
  qed (use f in ‹auto simp: B-def›)
  ultimately show ∃B'⊆{arith-prog-fb a b |a b. 0 < b}. ⋃ B' = U by auto
qed

lemma open-fbint-altdef: open = generate-topology {arith-prog-fb a b |a b. b > 0}
  using topological-basis-imp-subbasis[OF topological-basis-fbint] .
```

From this, we can immediately see that it is second countable:

```
instance fbint :: second-countable-topology
proof
  have countable ((λ(a,b). arith-prog-fb a b) ` (UNIV × {b. b > 0}))
    by (intro countable-image) auto
  also have ... = {arith-prog-fb a b |a b. b > 0}
    by auto
  ultimately show ∃B::fbint set set. countable B ∧ open = generate-topology B
    unfolding open-fbint-altdef by auto
qed
```

A trivial consequence of the fact that nonempty open sets in this topology are infinite is that it is a perfect space:

```
instance fbint :: perfect-space
  by standard auto
```

It is also Hausdorff, since given any two distinct integers, we can easily construct two non-overlapping arithmetic progressions that each contain one of them. We do not *really* have to prove this since we will get it for free later on when we show that it is a metric space, but here is the proof anyway:

```
instance fbint :: t2-space
proof
  fix x y :: fbint
  assume x ≠ y
  define d where d = nat |int-of-fbint x − int-of-fbint y| + 1
  from ‹x ≠ y› have d > 0
    unfolding d-def by transfer auto
  define U where U = arith-prog-fb (int-of-fbint x) d
  define V where V = arith-prog-fb (int-of-fbint y) d
```

```
  have U ∩ V = {} unfolding U-def V-def d-def
  proof (use ‹x ≠ y› in transfer, rule arith-prog-disjoint)
    fix x y :: int
    assume x ≠ y
    show [x ≠ y] (mod int (nat |x − y| + 1))
    proof
      assume [x = y] (mod int (nat |x − y| + 1))
      hence |x − y| + 1 dvd |x − y|
        by (auto simp: cong-iff-dvd-diff algebra-simps)
      hence |x − y| + 1 ≤ |x − y|
        by (rule zdvd-imp-le) (use ‹x ≠ y› in auto)
      thus False by simp
    qed
  qed auto
  moreover have x ∈ U y ∈ V
    unfolding U-def V-def by (use ‹d > 0› in transfer, fastforce)+
  moreover have open U open V
    using ‹d > 0› by (auto simp: U-def V-def)
  ultimately show ∃ U V. open U ∧ open V ∧ x ∈ U ∧ y ∈ V ∧ U ∩ V = {}
by blast
qed
```

Next, we need a small lemma: Given an additional assumption, a T_2 space is also T_3:

```
lemma t2-space-t3-spaceI:
  assumes ⋀(x :: 'a :: t2-space) U. x ∈ U ⟹ open U ⟹
            ∃ V. x ∈ V ∧ open V ∧ closure V ⊆ U
  shows   OFCLASS('a, t3-space-class)
proof
  fix X :: 'a set and z :: 'a
  assume X: closed X z ∉ X
  with assms[of z −X] obtain V where V: z ∈ V open V closure V ⊆ −X
    by auto
  show ∃ U V. open U ∧ open V ∧ z ∈ U ∧ X ⊆ V ∧ U ∩ V = {}
    by (rule exI[of - V], rule exI[of - −closure V])
       (use X V closure-subset[of V] in auto)
qed
```

Since the Furstenberg topology is T_2 and every arithmetic progression is also closed, we can now easily show that it is also T_3 (i. e. regular). Again, we do not really need this proof, but here it is:

```
instance fbint :: t3-space
proof (rule t2-space-t3-spaceI)
  fix x :: fbint and U :: fbint set
  assume x ∈ U and open U
  then obtain b where b: b > 0 arith-prog-fb (int-of-fbint x) b ⊆ U
    by transfer blast
  define V where V = arith-prog-fb (int-of-fbint x) b
```

```
  have x ∈ V
    unfolding V-def by transfer auto
  moreover have open V closed V
    using ‹b > 0› by (auto simp: V-def)
  ultimately show ∃ V. x ∈ V ∧ open V ∧ closure V ⊆ U
    using b by (intro exI[of - V]) (auto simp: V-def)
qed
```

1.5 Metrizability

The metrizability of Furstenberg's topology (i. e. that it is induced by some metric) can be shown from the fact that it is second countable and T_3 using Urysohn's Metrization Theorem, but this is not available in Isabelle yet. Let us therefore give an *explicit* metric, as described by Zulfeqarr [3]. We follow the exposition by Dirmeier [1].

First, we define a kind of norm on the integers. The norm depends on a real parameter $q > 1$. The value of q does not matter in the sense that all values induce the same topology (which we will show). For the final definition, we then simply pick $q = 2$.

```
locale fbnorm =
  fixes q :: real
  assumes q-gt-1: q > 1
begin

definition N :: int ⇒ real where
  N n = (∑k. if k = 0 ∨ int k dvd n then 0 else 1 / q ^ k)

lemma N-summable: summable (λk. if k = 0 ∨ int k dvd n then 0 else 1 / q ^ k)
  by (rule summable-comparison-test[OF - summable-geometric[of 1/q]])
     (use q-gt-1 in ‹auto intro!: exI[of - 0] simp: power-divide›)

lemma N-sums: (λk. if k = 0 ∨ int k dvd n then 0 else 1 / q ^ k) sums N n
  using N-summable unfolding N-def by (rule summable-sums)

lemma N-nonneg: N n ≥ 0
  by (rule sums-le[OF - sums-zero N-sums]) (use q-gt-1 in auto)

lemma N-uminus [simp]: N (−n) = N n
  by (simp add: N-def)

lemma N-minus-commute: N (x − y) = N (y − x)
  using N-uminus[of x − y] by (simp del: N-uminus)

lemma N-zero [simp]: N 0 = 0
  by (simp add: N-def)

lemma not-dvd-imp-N-ge:
  assumes ¬n dvd a n > 0
```

```
  shows   N a ≥ 1 / q ^ n
  by (rule sums-le[OF - sums-single[of n] N-sums]) (use q-gt-1 assms in auto)

lemma N-lt-imp-dvd:
  assumes N a < 1 / q ^ n and n > 0
  shows   n dvd a
  using not-dvd-imp-N-ge[of n a] assms by auto

lemma N-pos:
  assumes n ≠ 0
  shows   N n > 0
proof −
  have 0 < 1 / q ^(nat |n|+1)
    using q-gt-1 by simp
  also have ¬1 + |n| dvd |n|
    using zdvd-imp-le[of 1 + |n| |n|] assms by auto
  hence 1 / q ^(nat |n|+1) ≤ N n
    by (intro not-dvd-imp-N-ge) (use assms in auto)
  finally show ?thesis .
qed

lemma N-zero-iff [simp]: N n = 0 ⟷ n = 0
  using N-pos[of n] by (cases n = 0) auto

lemma N-triangle-ineq: N (n + m) ≤ N n + N m
proof (rule sums-le)
  let ?I = λn k. if k = 0 ∨ int k dvd n then 0 else 1 / q ^ k
  show ?I (n + m) sums N (n + m)
    by (rule N-sums)
  show (λk. ?I n k + ?I m k) sums (N n + N m)
    by (intro sums-add N-sums)
qed (use q-gt-1 in auto)

lemma N-1: N 1 = 1 / (q ∗ (q − 1))
proof (rule sums-unique2)
  have (λk. if k = 0 ∨ int k dvd 1 then 0 else 1 / q ^ k) sums N 1
    by (rule N-sums)
  also have (λk. if k = 0 ∨ int k dvd 1 then 0 else 1 / q ^ k) =
             (λk. if k ∈ {0, 1} then 0 else (1 / q) ^ k)
    by (simp add: power-divide cong: if-cong)
  finally show (λk. if k ∈ {0, 1} then 0 else (1 / q) ^ k) sums N 1 .

  have (λk. if k ∈ {0, 1} then 0 else (1 / q) ^ k) sums
              (1 / (1 − 1 / q) + (− (1 / q) − 1))
    by (rule sums-If-finite-set'[OF geometric-sums]) (use q-gt-1 in auto)
  also have . . . = 1 / (q ∗ (q − 1))
    using q-gt-1 by (simp add: field-simps)
  finally show (λk. if k ∈ {0, 1} then 0 else (1 / q) ^ k) sums . . . .
qed
```

It follows directly from the definition that norms fulfil a kind of monotonicity property with respect to divisibility: the norm of a number is at most as large as the norm of any of its factors:

lemma *N-dvd-mono*:
  assumes *m dvd n*
  shows *N n ≤ N m*
proof (*rule sums-le*[*OF - N-sums N-sums*])
  fix *k* :: *nat*
  show (*if k = 0 ∨ int k dvd n then 0 else 1 / q ^ k*) ≤
      (*if k = 0 ∨ int k dvd m then 0 else 1 / q ^ k*)
    using *q-gt-1 assms* **by** *auto*
qed

In particular, this means that 1 and -1 have the greatest norm.

lemma *N-le-N-1*: *N n ≤ N 1*
  by (*rule N-dvd-mono*) *auto*

Primes have relatively large norms, almost reaching the norm of 1:

lemma *N-prime*:
  assumes *prime p*
  shows *N p = N 1 − 1 / q ^ nat p*
proof (*rule sums-unique2*)
  define *p′* **where** *p′ = nat p*
  have *p*: *p = int p′*
    using *assms* **by** (*auto simp*: *p′-def prime-ge-0-int*)
  have *prime p′*
    using *assms* **by** (*simp add*: *p*)

  have (*λk. if k = 0 ∨ int k dvd p then 0 else 1 / q ^ k*) *sums N p*
    by (*rule N-sums*)
  also have *int k dvd p ⟷ k ∈ {1, p′}* **for** *k*
    using *assms* **by** (*auto simp*: *p prime-nat-iff*)
  hence (*λk. if k = 0 ∨ int k dvd p then 0 else 1 / q ^ k*) =
      (*λk. if k ∈ {0, 1, p′} then 0 else* (*1 / q*) *^ k*)
    using *assms q-gt-1* **by** (*simp add*: *power-divide cong*: *if-cong*)
  finally show ... *sums N p* .

  have (*λk. if k ∈ {0, 1, p′} then 0 else* (*1 / q*) *^ k*) *sums*
        (*1 /* (*1 − 1 / q*) + (− (*1 / q*) − (*1 / q*) *^ p′ − 1*))
    by (*rule sums-If-finite-set′*[*OF geometric-sums*])
      (*use* ‹*prime p′*› *q-gt-1 prime-gt-Suc-0-nat*[*of p′*] **in** ‹*auto simp*: ›)
  also have ... *= N 1 − 1 / q ^ p′*
    using *q-gt-1* **by** (*simp add*: *field-simps N-1*)
  finally show (*λk. if k ∈ {0, 1, p′} then 0 else* (*1 / q*) *^ k*) *sums* ... .
qed

lemma *N-2*: *N 2 = 1 /* (*q ^ 2 ∗* (*q − 1*))
  using *q-gt-1* **by** (*auto simp*: *N-prime N-1 field-simps power2-eq-square*)

```
lemma N-less-N-1:
  assumes n ≠ 1 n ≠ −1
  shows   N n < N 1
proof (cases n = 0)
  case False
  then obtain p where p: prime p p dvd n
    using prime-divisor-exists[of n] assms by force
  hence N n ≤ N p by (intro N-dvd-mono)
  also from p have N p < N 1
    using q-gt-1 by (simp add: N-prime)
  finally show ?thesis .
qed (use q-gt-1 in ‹auto simp: N-1›)
```

Composites, on the other hand, do not achieve this:

```
lemma nonprime-imp-N-lt:
  assumes ¬prime-elem n |n| ≠ 1 n ≠ 0
  shows   N n < N 1 − 1 / q ^ nat |n|
proof −
  obtain p where p: prime p p dvd n
    using prime-divisor-exists[of n] assms by auto
  define p' where p' = nat p
  have p': p = int p'
    using p by (auto simp: p'-def prime-ge-0-int)
  have prime p'
    using p by (simp add: p')

  define n' where n' = nat |n|
  have n' > 1
    using assms by (auto simp: n'-def)

  have N n ≤ 1 / (q ∗ (q − 1)) − 1 / q ^ p' − 1 / q ^ n'
  proof (rule sums-le)
    show (λk. if k = 0 ∨ int k dvd n then 0 else 1 / q ^ k) sums N n
      by (rule N-sums)
  next
    from assms p have n' ≠ p'
      by (auto simp: n'-def p'-def nat-eq-iff)
    hence (λk. if k ∈ {0, 1, p', n'} then 0 else (1 / q) ^ k) sums
              (1 / (1 − 1 / q) + (− (1 / q) − (1 / q) ^ p' − (1 / q) ^ n' − 1))
      by (intro sums-If-finite-set'[OF geometric-sums])
        (use ‹prime p'› q-gt-1 prime-gt-Suc-0-nat[of p'] ‹n' > 1› in ‹auto simp: ›)
    also have ... = 1 / (q ∗ (q − 1)) − 1 / q ^ p' − 1 / q ^ n'
      using q-gt-1 by (simp add: field-simps)
    finally show (λk. if k ∈ {0, 1, p', n'} then 0 else (1 / q) ^ k) sums ... .
  next
    show ⋀k. (if k = 0 ∨ int k dvd n then 0 else 1 / q ^ k)
        ≤ (if k ∈ {0, 1, p', n'} then 0 else (1 / q) ^ k)
      using q-gt-1 p by (auto simp: p'-def n'-def power-divide)
```

```
  qed
  also have ... < 1 / (q ∗ (q − 1)) − 1 / q ^ n′
    using q-gt-1 by simp
  finally show ?thesis by (simp add: n′-def N-1)
qed
```

This implies that one can use the norm as a primality test:

```
lemma prime-iff-N-eq:
  assumes n ≠ 0
  shows   prime-elem n ⟷ N n = N 1 − 1 / q ^ nat |n|
proof −
  have ∗: prime-elem n ⟷ N n = N 1 − 1 / q ^ nat |n| if n > 0 for n
  proof −
    consider n = 1 | prime n | ¬prime n n > 1
      using ‹n > 0› by force
    thus ?thesis
    proof cases
      assume n = 1
      thus ?thesis using q-gt-1
        by (auto simp: N-1)
    next
      assume n: ¬prime n n > 1
      with nonprime-imp-N-lt[of n] show ?thesis by simp
    qed (auto simp: N-prime prime-ge-0-int)
  qed

  show ?thesis
  proof (cases n > 0)
    case True
    with ∗ show ?thesis by blast
  next
    case False
    with ∗[of −n] assms show ?thesis by simp
  qed
qed
```

Factorials, on the other hand, have very small norms:

```
lemma N-fact-le: N (fact m) ≤ 1 / (q − 1) ∗ 1 / q ^ m
proof (rule sums-le[OF - N-sums])
  have (λk. 1 / q ^ k / q ^ Suc m) sums (q / (q − 1) / q ^ Suc m)
    using geometric-sums[of 1 / q] q-gt-1
    by (intro sums-divide) (auto simp: field-simps)
  also have (q / (q − 1) / q ^ Suc m) = 1 / (q − 1) ∗ 1 / q ^ m
    using q-gt-1 by (simp add: field-simps)
  also have (λk. 1 / q ^ k / q ^ Suc m) = (λk. 1 / q ^ (k + Suc m))
    using q-gt-1 by (simp add: field-simps power-add)
  also have ... = (λk. if k + Suc m ≤ m then 0 else 1 / q ^ (k + Suc m))
    by auto
  finally have ... sums (1 / (q − 1) ∗ 1 / q ^ m) .
```

also have *?this* $\longleftrightarrow$ (λk. *if* $k \leq m$ *then 0 else 1* / q ^ k) *sums* (*1* / (q − *1*) ∗ *1* / q ^ m)
  by (*rule sums-zero-iff-shift*) *auto*
 finally show . . . **.**
next
 fix k :: *nat*
 have *int k dvd fact m* **if** $k > 0$ $k \leq m$
 proof −
  have *int k dvd int* (*fact m*)
   unfolding *int-dvd-int-iff* **using** *that* **by** (*simp add*: *dvd-fact*)
  thus *int k dvd fact m*
   unfolding *of-nat-fact* **by** *simp*
 qed
 thus (*if* $k = 0 \vee$ *int k dvd fact m then 0 else 1* / q ^ k) $\leq$
   (*if* $k \leq m$ *then 0 else 1* / q ^ k) **using** *q-gt-1* **by** *auto*
qed

lemma *N-prime-mono*:
 assumes *prime p prime* p' $p \leq p'$
 shows $N\ p \leq N\ p'$
 using *assms q-gt-1* **by** (*auto simp add*: *N-prime field-simps nat-le-iff prime-ge-0-int*)

lemma *N-prime-ge*:
 assumes *prime p*
 shows $N\ p \geq 1\ /\ (q^2 * (q - 1))$
proof −
 have *1* / (q ^ *2* ∗ (q − *1*)) = *N 2*
  using *q-gt-1* **by** (*auto simp*: *N-prime N-1 field-simps power2-eq-square*)
 also have . . . $\leq N\ p$
  using *assms* **by** (*intro N-prime-mono*) (*auto simp*: *prime-ge-2-int*)
 finally show *?thesis* **.**
qed

lemma *N-prime-elem-ge*:
 assumes *prime-elem p*
 shows $N\ p \geq 1\ /\ (q^2 * (q - 1))$
proof (*cases* $p \geq 0$)
 case *True*
 with *assms N-prime-ge* **show** *?thesis* **by** *auto*
next
 case *False*
 with *assms N-prime-ge*[*of* −*p*] **show** *?thesis* **by** *auto*
qed

Next, we use this norm to derive a metric:

lift-definition *dist* :: *fbint* ⇒ *fbint* ⇒ *real* **is**
 $\lambda x\ y.\ N\ (x - y)$ **.**

lemma *dist-self* [*simp*]: *dist x x* = *0*

by *transfer simp*

lemma *dist-sym* [*simp*]: *dist x y* = *dist y x*
by *transfer* (*simp add*: *N-minus-commute*)

lemma *dist-pos*: $x \neq y \Longrightarrow$ *dist x y* > *0*
by *transfer* (*use N-pos* **in** *simp*)

lemma *dist-eq-0-iff* [*simp*]: *dist x y* = *0* $\longleftrightarrow$ *x* = *y*
using *dist-pos*[*of x y*] **by** (*cases x* = *y*) *auto*

lemma *dist-triangle-ineq*: *dist x z* $\leq$ *dist x y* + *dist y z*
proof *transfer*
fix *x y z* :: *int*
show *N* (*x* − *z*) $\leq$ *N* (*x* − *y*) + *N* (*y* − *z*)
using *N-triangle-ineq*[*of x* − *y y* − *z*] **by** *simp*
qed

Lastly, we show that the metric we defined indeed induces the Furstenberg topology.

theorem *dist-induces-open*:
open U $\longleftrightarrow$ ($\forall x \in U.\ \exists e>0.\ \forall y.$ *dist x y* < *e* $\longrightarrow$ *y* $\in$ *U*)
proof (*transfer*, *safe*)
fix *U* :: *int set* **and** *x* :: *int*
assume $*$: $\forall x \in U.\ \exists b>0.$ *arith-prog x b* $\subseteq$ *U*
assume *x* $\in$ *U*
with $*$ **obtain** *b* **where** *b*: *b* > *0 arith-prog x b* $\subseteq$ *U* **by** *blast*
define *e* **where** *e* = *1* / *q* ^ *b*

show $\exists e>0.\ \forall y.$ *N* (*x* − *y*) < *e* $\longrightarrow$ *y* $\in$ *U*
proof (*rule exI*; *safe?*)
show *e* > *0* **using** *q-gt-1* **by** (*simp add*: *e-def*)
next
fix *y* **assume** *N* (*x* − *y*) < *e*
also have ... = *1* / *q* ^ *b* **by** *fact*
finally have *b dvd* (*x* − *y*)
by (*rule N-lt-imp-dvd*) *fact*
hence *y* $\in$ *arith-prog x b*
by (*auto simp*: *arith-prog-def cong-iff-dvd-diff dvd-diff-commute*)
with *b* **show** *y* $\in$ *U* **by** *blast*
qed

next

fix *U* :: *int set* **and** *x* :: *int*
assume $*$: $\forall x \in U.\ \exists e>0.\ \forall y.$ *N* (*x* − *y*) < *e* $\longrightarrow$ *y* $\in$ *U*
assume *x* $\in$ *U*
with $*$ **obtain** *e* **where** *e*: *e* > *0* $\forall y.$ *N* (*x* − *y*) < *e* $\longrightarrow$ *y* $\in$ *U* **by** *blast*
have *eventually* ($\lambda N.$ *1* / (*q* − *1*) $*$ *1* / *q* ^ *N* < *e*) *at-top*

```
    using q-gt-1 ‹e > 0› by real-asymp
  then obtain m where m: 1 / (q − 1) ∗ 1 / q ^ m < e
    by (auto simp: eventually-at-top-linorder)
  define b :: nat where b = fact m

  have arith-prog x b ⊆ U
  proof
    fix y assume y ∈ arith-prog x b
    show y ∈ U
    proof (cases y = x)
      case False
      from ‹y ∈ arith-prog x b› obtain n where y: y = x + int b ∗ n
        by (auto simp: arith-prog-altdef)
      from y and ‹y ≠ x› have [simp]: n ≠ 0 by auto
      have N (x − y) = N (int b ∗ n) by (simp add: y)
      also have . . . ≤ N (int b)
        by (rule N-dvd-mono) auto
      also have . . . ≤ 1 / (q − 1) ∗ 1 / q ^ m
        using N-fact-le by (simp add: b-def)
      also have . . . < e by fact
      finally show y ∈ U using e by auto
    qed (use ‹x ∈ U› in auto)
  qed
  moreover have b > 0 by (auto simp: b-def)
  ultimately show ∃ b>0. arith-prog x b ⊆ U
    by blast
qed

end
```

We now show that the Furstenberg space is a metric space with this metric (with $q = 2$), which essentially only amounts to plugging together all the results from above.

```
interpretation fb: fbnorm 2
  by standard auto


instantiation fbint :: dist
begin

definition dist-fbint where dist-fbint = fb.dist

instance ..

end


instantiation fbint :: uniformity-dist
begin
```

```
definition uniformity-fbint :: (fbint × fbint) filter where
  uniformity-fbint = (INF e∈{0 <..}. principal {(x, y). dist x y < e})

instance by standard (simp add: uniformity-fbint-def)

end


instance fbint :: open-uniformity
proof
  fix U :: fbint set
  show open U = (∀x∈U. eventually (λ(x′,y). x′ = x ⟶ y ∈ U) uniformity)
    unfolding eventually-uniformity-metric dist-fbint-def
    using fb.dist-induces-open by simp
qed


instance fbint :: metric-space
  by standard (use fb.dist-triangle-ineq in ‹auto simp: dist-fbint-def›)
```

In particular, we can now show that the sequence $n!$ tends to 0 in the Furstenberg topology:

```
lemma tendsto-fbint-fact: (λn. fbint (fact n)) ⟶ fbint 0
proof –
  have (λn. dist (fbint (fact n)) (fbint 0)) ⟶ 0
  proof (rule tendsto-sandwich[OF always-eventually always-eventually]; safe?)
    fix n :: nat
    show dist (fbint (fact n)) (fbint 0) ≤ 1 / 2 ^ n
     unfolding dist-fbint-def by (transfer fixing: n) (use fb.N-fact-le[of n] in simp)
    show dist (fbint (fact n)) (fbint 0) ≥ 0
      by simp
    show (λn. 1 / 2 ^ n :: real) ⟶ 0
      by real-asymp
  qed simp-all
  thus ?thesis
    using tendsto-dist-iff by metis
qed

end
```

References

[1] A. Dirmeier. On metrics inducing the Fürstenberg topology on the integers. https://arxiv.org/abs/1912.11663, 2019.

[2] H. Furstenberg. On the infinitude of primes. *The American Mathematical Monthly*, 62(5):353, May 1955.

[3] F. Zulfeqarr. Some interesting consequences of Furstenberg topology. *Resonance*, 24(7):755–765, July 2019.