

PRIME-GENERATED LOCALIZATION DESCENT FOR UNIQUE FACTORIZATION IN LEAN

ARTHUR FREITAS RAMOS, DAVID BARROS HULAK, RUY J. G. B. DE QUEIROZ,
AND ANJOLINA GRISI DE OLIVEIRA

ABSTRACT. We give a source-grounded exposition of a Lean 4 formalization of prime-generated Nagata factoriality descent. A commutative noetherian integral domain is a unique factorization domain if a localization is a unique factorization domain and every denominator is exactly a finite product of prime elements belonging to the denominator submonoid. The formal statement works with an arbitrary algebra carrying Mathlib's *IsLocalization* structure. We explain the two cases for an irreducible element, the multiset cancellation and splitting arguments, and the corollary for submonoids generated by arbitrary sets of primes. A registered polynomial corollary illustrates reuse of the descent package; its Laurent-polynomial premise is proved using Mathlib's existing polynomial UFD instance, a dependency made explicit here. The exposition is tied to the registered source and dependency pins, distinguishes historical mechanical verification from manuscript review, and situates the development relative to an earlier Lean preprint and an Isabelle/HOL formalization. No new algebraic theorem or priority claim is asserted.

1. INTRODUCTION

Localization simplifies algebra by turning selected elements into units. The reverse passage is more delicate: factoriality of a localization need not imply factoriality of the original domain. Nagata's criterion identifies a useful situation in which it does. If the elements being inverted are built out of prime elements, their effect on factorization can be tracked; if the base domain also has factorization into irreducibles, primality can be recovered before localization. Noetherianity supplies the latter existence condition in the registered theorem considered here.

The classical mathematical source is Nagata's 1957 note [2], specifically Lemma 2. That lemma assumes denominators are products of prime elements of the ring. The formal artifact uses a prime-generated submonoid condition that additionally requires each chosen prime factor to belong to the denominator submonoid. The elementwise proof discussed below closely follows the irreducibility and primality transfer pattern also presented in the Stacks Project, Tag 0AFU [9]. Nagata's original proof and the present implementation need not have identical intermediate objects to express the same classical descent principle.

Our subject is the public Lean repository [6] at commit `9ebcfa77a4f7eab6fdb663e7f151ceeb48abdf7`, registered as PALOMAR-2026-08-28-000002, version 1 [3]. The main purpose is to make the registered statements and their proof dependencies readable without requiring the reader to reconstruct them from Lean source. We give a mathematical proof of the implemented descent argument, identify the corresponding declarations, and explain the boundaries of the polynomial demonstration.

Date: October 1, 2026.

2020 Mathematics Subject Classification. 13A05, 13B30, 03B35.

Key words and phrases. Nagata factoriality, prime-generated submonoid, localization, unique factorization domain, Lean 4, Mathlib.

Author ORCID identifiers: Arthur Freitas Ramos, 0009-0003-3568-0325; David Barros Hulak, 0009-0002-8056-1774; Ruy J. G. B. de Queiroz, 0000-0003-1482-0977.

Copyright 2026 the authors. This manuscript is licensed under Creative Commons Attribution 4.0 International (CC BY 4.0).

This is a new exposition of an existing formalization. The earlier Lean preprint by Ramos, de Queiroz, and de Oliveira [5] already treats this project. A related Isabelle/HOL development by Ramos, Hulak, and de Queiroz appears in the Archive of Formal Proofs [7]. These works are credited separately; the registered Lean statements are not a verification of the earlier manuscript or of the Isabelle session. Our contribution here is a precise account of the pinned Lean artifact, including a dependency disclosure that matters when interpreting its polynomial result.

2. THE REGISTERED MATHEMATICAL STATEMENTS

Throughout the descent argument, R is a commutative integral domain, S is a multiplicative submonoid of R , and T is an R -algebra that is a localization of R at S . Write $\iota : R \rightarrow T$ for the algebra map. The notation a/s denotes its localization fraction, not division in R . When convenient we write $T = S^{-1}R$; the formal theorem does not require T to be the concrete quotient type bearing that name.

An element is irreducible when it is a nonunit and every factorization has a unit factor. A prime element is nonzero, is a nonunit, and divides one factor whenever it divides a product. A UFD is a domain in which every nonzero nonunit factors into irreducibles and such factorizations are unique up to associates and permutation. Mathlib packages the multiplicative property as `UniqueFactorizationMonoid`.

2.1. Prime-generated denominators.

Definition 2.1. The submonoid S is *prime-generated* if, for every $s \in S$, there is a finite multiset f of elements of R such that

$$s = \prod_{q \in f} q, \quad q \in S \text{ and } q \text{ is prime for every } q \in f.$$

The empty product is 1. Define

$$\text{Avoids}(S, p) \iff (\forall s \in S) p \nmid s.$$

The multiset product counts occurrences, so repeated prime factors are allowed. The factorization is an equality in R , not merely an equality up to multiplication by a unit. The factors are prime elements, not prime ideals, and their membership in S is part of the condition. There is no bound on the number of factors and no assumption that S has finitely many generators. In particular, the finite product required for each denominator must not be confused with finite generation of the whole submonoid.

Definition 2.1 has a direct set-theoretic interpretation. If $P = \{q \in S : q \text{ is prime}\}$, then

$$\text{PrimeGenerated}(S) \iff S = \langle P \rangle,$$

where $\langle P \rangle$ denotes multiplicative submonoid closure. Indeed, the forward direction expresses each member of S as a product of members of P , and the reverse direction follows by closure induction. This is an explanation of the predicate, rather than an additional registered theorem.

Two consequences are worth keeping in view. First, $0 \notin S$, because every prime element is nonzero and a product of nonzero elements in a domain is nonzero. Second, the only unit that can belong to such an S is 1: a nonempty product of nonunits cannot be a unit in a commutative ring. Thus the predicate is more specific than a convention allowing arbitrary unit factors in each displayed product. This causes no difficulty for submonoids obtained as the closure of a set of primes.

2.2. Descent and closure of prime generators.

Theorem 2.2 (Registered Nagata descent). *Let R be a commutative noetherian integral domain, let S be a prime-generated multiplicative submonoid, and let T be a commutative integral domain with an R -algebra structure making it a localization at S . If T is a UFD, then R is a UFD.*

This is `NagataFactoriality.palomar_nagata_factoriality`. The formal assumptions explicitly include the domain structure on both R and T . The latter is consistent with, and mathematically follows from, localization of a domain at a submonoid avoiding zero; it remains an explicit typeclass assumption in the selected statement. The localization map is injective under these conditions. No embedding into a particular fraction field is an additional input.

Corollary 2.3 (Registered prime-generator specialization). *Let R be a commutative noetherian integral domain, let $P \subseteq R$ consist of prime elements, and let T be a commutative integral-domain localization of R at $\langle P \rangle$. If T is a UFD, then R is a UFD.*

The declaration is `NagataFactoriality.palomar_nagata_factoriality_of_prime_generators`. To obtain it, use closure induction to prove `PrimeGenerated( $\langle P \rangle$ )`. A generator is represented by a singleton multiset, 1 by the empty multiset, and multiplication by multiset addition. Every resulting factor belongs to the closure because it began as a generator. Then apply Theorem 2.2. The set P is arbitrary, including the empty set; in the empty case the localization inverts only 1 and the descent is tautological. The repository also supplies finite-generator and powers-of-a-prime wrappers outside the three selected declarations.

2.3. Where noetherianity enters. The proof of primality of irreducibles in Section 3 needs no noetherian hypothesis. Noetherianity enters when assembling those lemmas into a UFD structure: `Mathlib` provides a `WfDvdMonoid R` instance for a noetherian domain, giving termination of descent by proper divisibility and hence factorization existence. The project wrapper `hasFactorization_of_noetherian` is an `infer_instance` invocation, not a new proof of the noetherian factorization theorem.

The closing characterization says that well-founded divisibility together with primality of every irreducible yields a UFD. In the source, `ufd_of_factorization_and_primes` packages this step through `Mathlib`'s prime-factor characterization. It is important not to replace this explanation by the false equivalence “a domain is a UFD if and only if it is noetherian and every irreducible is prime.” Noetherianity is sufficient here, but not necessary for a domain to be a UFD. The registered theorem retains it as its chosen factorization hypothesis.

3. THE ELEMENTWISE PROOF

We now prove Theorem 2.2 in the form implemented by the prime-generated branch of `Nagata/Lemmas.lean`. All cancellation below takes place in the integral domain R . Products of prime elements are nonzero, so each cancellation used in the argument has an explicit nonzero justification.

3.1. Clearing denominators. The localization interface supplies the following facts:

- (1) every $z \in T$ can be written as a/s , $a \in R, s \in S$,
- (2) $a/s = b/t \iff at = bs$, $s, t \in S$,
- (3) $\iota(a) \mid \iota(b) \iff \exists s \in S (a \mid sb)$.

For the forward implication of (3), write a quotient witness as c/s and cross-multiply to obtain $sb = ac$. Conversely, if $sb = ac$, then c/s is a quotient witness for divisibility in T . Equation (2) uses injectivity of the map from R ; for a general ring localization an additional multiplier from S would occur. The domain and zero-exclusion assumptions are what allow the simplified cross-product equality here.

These statements occur in the abstract helper namespace `NagataFactoriality.IsLocalization`, as `surj`, `mk'_eq_iff`, and `dvd_map_iff`. Elements of S map to units, and fractions with unit numerators are units. The helper interface derives these results from Mathlib's localization API.

3.2. An irreducible divisor of a prime product.

Lemma 3.1. *If p is irreducible and divides a finite product of prime elements, then p is prime.*

Proof. Induct on the number of factors. The empty product is 1, which an irreducible cannot divide. For a product qv , write $qv = pd$, with q prime. Since $q \mid pd$, either $q \mid p$ or $q \mid d$. In the first case, irreducibility of p and of q implies that they are associates, so p is prime. In the second case write $d = qe$ and cancel the nonzero element q to obtain $v = pe$. The induction hypothesis applies to the remaining prime factors. $\square$

The implemented result is `prime_of_irreducible_of_dvd_prime_factors`, and its submonoid specialization is `prime_of_irreducible_of_dvd_mem_primeGenerated`. Consequently, an irreducible p that divides even one denominator is already known to be prime before any appeal to factoriality of T .

3.3. Cancellation of denominators avoiding an irreducible.

Lemma 3.2. *Let p be irreducible, let f be a multiset of prime elements, and suppose $p \nmid q$ for every $q \in f$. If*

$$\left(\prod_{q \in f} q\right)a = pc,$$

then $p \mid a$.

Proof. Induct on f . With no factors the assertion is immediate. Write the product as qv . Primality of q gives $q \mid p$ or $q \mid c$. If $q \mid p$, the two irreducibles are associates, contradicting $p \nmid q$. Therefore $c = qe$. Cancel q from $qva = pqe$ and apply the induction hypothesis to $va = pe$. $\square$

This is `dvd_of_mul_eq_prime_factors`. Notice that the argument does not assume that p is prime: that is precisely what the final descent proof is trying to establish.

Corollary 3.3. *If $\text{PrimeGenerated}(S)$, p is irreducible, and $\text{Avoids}(S, p)$, then*

$$\iota(p) \mid \iota(a) \implies p \mid a.$$

Proof. By (3), $p \mid sa$ for some $s \in S$. Expand s as its prime-factor multiset. No factor can be divisible by p , since every factor belongs to S and p avoids S . Lemma 3.2 removes those factors. $\square$

The corresponding abstract declaration is `dvd_of_localization_dvd_primeGenerated_isLocalization`. The avoidance hypothesis is essential to this reflection result: an element being inverted can divide every localized element without dividing each numerator in the original ring.

3.4. Splitting a denominator product between two numerators.

Lemma 3.4. *Suppose f is a multiset of prime elements and*

$$p\left(\prod_{q \in f} q\right) = ab.$$

There are multisets f_1, f_2 and elements $a', b' \in R$ with

$$\begin{aligned} f_1 + f_2 &= f, & a &= \left(\prod_{q \in f_1} q\right)a', \\ p &= a'b', & b &= \left(\prod_{q \in f_2} q\right)b'. \end{aligned}$$

Proof. Induct on f . For the empty multiset use $a' = a, b' = b$. For a leading prime q , the equality shows $q \mid ab$, so q divides a or b . Remove it from that numerator, cancel it from the equality, and use the induction hypothesis for the remaining multiset. Add the removed occurrence of q to the corresponding part of the partition. $\square$

The source theorem `split_prime_factors_of_mul_eq` includes an irreducibility parameter for p , named `_hp`; its induction does not use that parameter. Irreducibility becomes necessary in the next lemma. Multisets record both the partition and the multiplicities, avoiding a choice of ordering of the denominator factors.

Lemma 3.5. *If $\text{PrimeGenerated}(S)$, p is irreducible in R , and $\text{Avoids}(S, p)$, then $\iota(p)$ is irreducible in T .*

Proof. If $\iota(p)$ were a unit, it would divide 1. Equation (3) would give $p \mid s$ for some $s \in S$, contradicting avoidance. Now suppose $\iota(p) = xy$. Write $x = a/s$ and $y = b/t$ using (1). Cross multiplication yields $p(st) = ab$. Express st as a prime-factor multiset and apply Lemma 3.4. Since $p = a'b'$ is irreducible, either a' or b' is a unit. If a' is a unit, then

$$x = \iota\left(\prod_{q \in f_1} q\right)(a'/s)$$

is a product of units: its first factor comes from S , and a'/s has unit numerator. The case of b' is symmetric. Thus every factorization of $\iota(p)$ has a unit factor. $\square$

The implementation is `localization_irreducible_of_irreducible_primeGenerated_isLocalization`. Unlike the divisibility-reflection proof, it explicitly manipulates fraction representatives. The requirement that each prime factor of a denominator belongs to S is used when proving that the two partition products map to units.

3.5. The two cases and the UFD conclusion.

Proof of Theorem 2.2. Let p be irreducible in R . If p divides some $s \in S$, Lemma 3.1 shows that p is prime. Otherwise $\text{Avoids}(S, p)$ holds. Lemma 3.5 makes $\iota(p)$ irreducible in T and hence prime because T is a UFD.

If $p \mid ab$ in R , map the divisibility to T . Primality of $\iota(p)$ gives $\iota(p) \mid \iota(a)$ or $\iota(p) \mid \iota(b)$. Corollary 3.3 brings the selected divisibility back to R . Together with the nonzero and nonunit conditions from irreducibility, this proves that p is prime in the second case too. Finally, noetherianity supplies factorization into irreducibles, and the UFD characterization completes the proof. $\square$

The source separates the last primality reflection into `prime_of_localization_prime_primeGenerated_isLocalization` and assembles the case split in `nagata_key_lemma_primeGenerated_isLocalization`. The top-level descent theorem then supplies noetherian factorization and invokes this key lemma for every irreducible.

4. LEAN INTERFACES AND PROOF ARCHITECTURE

Lean 4 [1] and Mathlib [8] supply the ring, submonoid, localization, and factorization infrastructure. The project organizes its additional proof into a small wrapper layer and the transfer lemmas described above. The three registered declarations use ordinary Mathlib types; their statement module does not import the project-specific definition `PrimeGenerated`.

4.1. The selected statement surface. The following display transcribes the type of the first selected declaration from `Challenge.lean`, with line wrapping adjusted for display. The theorem body is omitted; its proved counterpart occurs in `Solution.lean`.

```
theorem palomar_nagata_factoriality
  {R T : Type*} [CommRing R] [IsDomain R]
  [IsNoetherianRing R]
  (S : Submonoid R) [CommRing T] [Algebra R T]
  [_root_.IsLocalization S T] [IsDomain T]
  (hS : ∀ s : R, s ∈ S →
    ∃ f : Multiset R,
      (∀ q ∈ f, q ∈ S ∧ Prime q) ∧ f.prod = s)
  (hUFD : UniqueFactorizationMonoid T) :
  UniqueFactorizationMonoid R
```

The source is linked above. In particular, the conjunction in hS requires both membership and primality for each occurrence in the multiset.

The proof body of this declaration is a single application of `nagata_theorem_isLocalization`. The prime-generator declaration uses `Submonoid.closure s` in its localization assumption and the premise that every member of s is prime. Its solution invokes `nagata_theorem_of_prime_generators_isLocalization`, which uses the closure lemma described in Corollary 2.3. The polynomial declaration is

```
theorem palomar_polynomial_ufd
  {R : Type*} [CommRing R] [IsDomain R]
  [IsNoetherianRing R] [UniqueFactorizationMonoid R] :
  UniqueFactorizationMonoid R[X]
```

Its solution calls `polynomial_uniqueFactorizationMonoid_via_nagata`. The mathematical dependence of that call is discussed in Section 5.

4.2. Abstract and concrete localization. The principal prime-generated transfer lemmas are stated at the abstract `IsLocalization S T` level. This allows the key argument to work directly in a ring such as a Laurent-polynomial ring once its localization instance is supplied. The proof does not first transport everything to the concrete quotient `Localization S`. Concrete theorem names are also provided, but specialize the abstract transfer lemmas.

For the concrete localization, the zero-exclusion fact is installed locally as a `Fact` instance asserting $0 \notin S$, so that typeclass search can recover the domain structure. For the abstract selected theorem, `IsDomain T` is already a hypothesis; zero exclusion is still derived from `PrimeGenerated(S)` when an injectivity or cross-multiplication helper needs it. These are interface choices and should not be read as additional mathematical restrictions hidden outside the displayed statement.

The module boundary follows the argument. `Basic/Noetherian.lean` and `Basic/UFD.lean` supply factorization existence and final UFD assembly. `Localization/IsLocalization.lean` supplies the abstract fraction interface. `Nagata/Lemmas.lean` contains the multiset inductions and transfer chain, while `Nagata/Theorem.lean` exposes abstract, concrete, and generator-based descent statements. The application layer and `Solution.lean` then select the public

result. These module paths are relative to `NagataFactoriality/`, except for the root solution module.

4.3. The prime-or-unit variant. The repository retains another chain under the hypothesis

$$(\forall s \in S) \quad s \text{ is prime or a unit.}$$

This condition is excessively restrictive for a multiplicative submonoid of a domain. In fact it permits no nonunits. If a nonunit $p \in S$ existed, it would be prime; closure would give $p^2 \in S$. The square is not a unit and cannot be prime, because prime implies irreducible and $p^2 = p \cdot p$ has two nonunit factors. This contradicts the condition.

Accordingly this variant does not cover even the powers of one prime. It is logically valid as a descent statement, but inverts units only, so the localization has not changed the ring up to isomorphism. The three registered declarations use prime-generated denominators instead. The powers-of- X application depends on allowing X^2 and higher products, exactly as Definition 2.1 does. The distinction is about the scope of the hypothesis, not a counterexample to the truth of a proved theorem.

5. THE POLYNOMIAL SPECIALIZATION AND ITS DEPENDENCIES

The third registered result says that $R[X]$ is a UFD when R is a commutative noetherian integral domain carrying a UFD structure. The result is classical and already follows from Mathlib’s polynomial factorization instance without the noetherian assumption. Its role in this artifact is to exercise the localization-descent interface.

5.1. Conditional descent from Laurent polynomials. Let $A = R[X]$ and $S = \{X^n : n \geq 0\}$. The polynomial X is prime when R is a domain, and its powers form a prime-generated submonoid: represent X^n by n copies of X . Mathlib identifies the Laurent ring $R[X, X^{-1}]$ as a localization of A away from X . Noetherianity of R supplies noetherianity of A through the polynomial-ring infrastructure. Thus Theorem 2.2 gives the useful conditional implication

$$R[X, X^{-1}] \text{ is a UFD} \implies R[X] \text{ is a UFD.}$$

The source packages this as `polynomial_uniqueFactorizationMonoid_of_laurent`. This step really is an application of the descent theorem and does not assume a UFD structure on $R[X]$.

5.2. How the registered proof supplies the premise. To prove the Laurent premise from a UFD structure on R , the source uses `laurentPolynomial_uniqueFactorizationMonoid`. Its first line installs the existing Mathlib instance:

```
letI : UniqueFactorizationMonoid R[X] := inferInstance
```

This is a substantive mathematical dependency. The remainder represents a nonzero Laurent polynomial, after multiplication by a unit monomial, as an ordinary polynomial p . It removes the largest power of X dividing p , leaving a nonzero polynomial q not divisible by X . It then factors q using the UFD structure on $R[X]$ just installed. Each prime factor not associated to X remains prime in the Laurent ring, using localization of its principal prime ideal. The monomial factors become units, and the prime factorization gives the Laurent UFD structure.

Finally, `polynomial_uniqueFactorizationMonoid_via_nagata` uses this Laurent structure and calls the conditional descent theorem. The full dependency path is therefore

$$\begin{aligned} & \text{Mathlib polynomial UFD instance for } R[X] \\ \implies & \text{the project's Laurent UFD construction} \\ \implies & \text{Nagata descent back to } R[X]. \end{aligned}$$

This is a sound proof term, because the initial UFD instance is already proved in the imported library. It is not an independent derivation of polynomial factoriality avoiding that theorem. The distinction does not affect the type or validity of the registered corollary; it does affect what methodological contribution should be attributed to its proof. The pinned Mathlib file `Mathlib/RingTheory/Polynomial/UniqueFactorization.lean` contains `Polynomial.uniqueFactorizationMonoid`, with no noetherian-ring premise. The pinned library also provides `UniqueFactorizationMonoid.of_isLocalization`, a general forward transfer of UFD structure to a localization. Thus Laurent factoriality is not an absent library capability; the project gives an explicit construction using available factorization and localization infrastructure.

5.3. A separate constant-prime construction. The repository also includes `Applications/FractionField.lean`, which is not the proof selected by `palomar_polynomial_ufd`. It localizes $R[X]$ at the closure of the constant polynomials $C(r)$ with r prime in R . These constant polynomials are prime, so the denominator submonoid is prime-generated. Since R is a UFD, every nonzero coefficient is associated to a finite product of primes. Its constant image therefore becomes a unit in this localization.

This lets the source identify the localization with $\text{Frac}(R)[X]$, by comparing both as localizations at the larger submonoid of nonzero constant polynomials. The polynomial ring over the fraction field obtains a UFD instance from Mathlib, and that instance is transported across the algebra equivalence before descent. This path has a different dependency structure from the Laurent demonstration: it invokes polynomial factoriality over a field in the localized ring. We describe it as a companion construction in the source, not as an additional registered declaration or a new algebraic result. Likewise, the iterated polynomial wrapper in `Applications/Laurent.lean` reuses the selected construction; it does not remove its library dependency.

6. PRIOR WORK AND PROVENANCE

Nagata’s note [2] establishes the classical factoriality criterion; the registered artifact’s metadata also names *Local Rings* and Samuel’s *Lectures on Unique Factorization Domains* as background. Our elementwise presentation has been reconstructed from the pinned Lean proof and is not a reproduction of textbook prose. The Stacks Project [9] makes explicit that factorization existence can be separated from localization transfer, matching the logical division used here. Its criterion is broader than the noetherian selected statement in the factorization hypothesis.

The April 2026 preprint [5] discusses the same Lean project under an earlier toolchain. Its authors are Arthur F. Ramos, Ruy J. G. B. de Queiroz, and Anjolina Grisi de Oliveira. It is prior exposition of this formalization, not evidence that the August registered source has precisely the same declarations, hypotheses, or dependency graph. The present account uses the August pin and explicitly states the polynomial overlap documented in Section 5. It does not retain any first-public- formalization claim.

The AFP entry [7], dated April 20, 2026, credits Arthur Freitas Ramos, David Barros Hulak, and Ruy Jose Guerra Barretto de Queiroz. It formalizes prime-generated factoriality descent in Isabelle/HOL, using record-based ring and localization interfaces and depending on the AFP entry *The Localization of a Commutative Ring*. It also packages closure-based corollaries and polynomial-localization applications. This is related work with a different prover and interface, rather than an outcome checked by the Lean registration. The Lean repository includes Isabelle files, but those files are outside the registered Lean statement surface and the verification scope discussed next.

The attribution of a manuscript, a formalization artifact, and a registry submission are separate matters. The pinned Palomar record lists Arthur Freitas Ramos, David Barros Hulak, and Ruy J. G. B. de Queiroz for the registered artifact, with Arthur as responsible

maintainer. Credit to the earlier Lean manuscript includes de Oliveira as indicated above. No inference about which author proved an individual lemma is made from these lists.

7. REGISTERED VERIFICATION AND REPRODUCIBILITY

7.1. Exact source identity. The record [3] fixes Lean `leanprover/lean4:v4.33.0` and Mathlib commit `db584cd6d46c92f209a44c0f1c829460d327499d`. The repository commit is `9ebcfa77a4f7eab6fdb663e7f151ceeb48abdf7`. These identities distinguish the registered artifact from the Lean 4.24.0 snapshot described by the earlier preprint. Reproducing the selected result means checking the pinned source and lock file, not the current repository branch or a PDF alone.

The registry records a successful mechanical verification at August 27, 2026, 23:55:51 UTC, with registration on August 28, 2026, 01:12:18 UTC. Its linked workflow is run `33127679460`. The archived mechanical report [4] has status `pass`, no report-level errors or warnings, and SHA-256 `dc1fae6a07a45cf6e5c65a6777635a6266302ec10d35809d8d0bd6bbe6ffbc57`. The retained compilation log does contain expected statement-placeholder warnings and style-linter warnings; a report-level pass is not a claim that the historical Lean compiler output was warning-free.

7.2. Statement and proof separation. The statement-only module `Challenge.lean` imports four Mathlib modules and contains three intentional `sorry` placeholders. It fixes the target types using only ordinary library definitions. The separate module `Solution.lean` provides proved declarations with the same names by invoking the project theorems. The registration compares this pair; the statement placeholders are not evidence that the selected solution proofs rely on an admitted result. Conversely, searching source text for placeholders is not a substitute for the statement comparison and axiom check.

The recorded permitted axioms are `propext`, `Quot.sound`, and `Classical.choice`. These are the standard classical and quotient foundations of this Lean development. Thus “no additional project axioms” is the appropriate interpretation; the result is not an axiom-free constructive proof. The record’s preservation metadata also archives the project and pinned dependencies.

The solution and challenge SHA-256 values in the record are, respectively, `e3a673c72a1935a2ebe45651ec24b4596f5886e98a314555ee345ea8d221bc3b` and `76dd7103a3fc14ea71e25a6f494930dd56217971e2ed83cb6f89b66680181d25`. They provide byte-level identifiers in addition to the repository commit. The verification is historical evidence for those selected statements; the preparation of this manuscript did not rerun Lean, Comparator, or NanoDa.

7.3. A reproduction boundary. A reader can retrieve the pinned repository, install the recorded Lean toolchain, retain its `lake-manifest.json`, fetch the Mathlib cache with `lake exe cache get`, and compile the selected proof module and its dependencies with `lake build Solution`. A project-wide `lake build` checks the configured library targets. Replaying the registration’s statement comparison and exported-term verification additionally requires the recorded external tool versions; ordinary project compilation should not be represented as the same operation.

Neither the mechanical registration nor this exposition establishes that a manuscript is free of every explanatory error, that a theorem is new, or that a human referee has reviewed it. The formal target types still need to match the intended mathematical claim. This is why the precise denominator condition and the polynomial dependency have been emphasized independently of the pass result.

7.4. AI assistance and author understanding. The pinned formalization metadata describes manual mathematical development and OpenAI Codex assistance for repository preparation, statement/proof separation, and reproducibility checks. It does not identify a

precise historical model version for every proof task. This historical description is separate from the present manuscript.

This manuscript was drafted primarily with GPT-6.1 assistance, including source cross-checking and typesetting. The submitting author, Arthur Freitas Ramos, reports understanding some parts of the work. This statement does not assert full understanding by him or any level of understanding by the other named manuscript authors. AI-assisted mathematical and source review is not independent human peer review. The Lean source is licensed Apache-2.0; the manuscript and its TeX source are licensed CC BY 4.0. The manuscript license does not relicense the formalization, its dependencies, or the cited prior works.

8. CONCLUSION

The registered development separates Nagata descent into a factorization-existence input and an elementwise prime-transfer argument. Finite prime-factor multisets serve two different purposes: they cancel denominators in divisibility reflection and partition denominator factors in irreducibility preservation. The abstract localization interface lets the same chain apply to different representations of a localization, while closure induction supplies the prime-generator corollary without a finite-generation restriction.

The polynomial declaration is best understood as a demonstration of this interface within an existing algebra library. Its selected Laurent proof depends on Mathlib’s polynomial UFD instance and makes no independent-proof claim. Broader denominator conventions, nonnoetherian generalizations, and ideal-theoretic class-group formulations are outside the three selected statements. An extension in any of these directions would need its own explicit statement, proof, and verification evidence.

REFERENCES

1. Leonardo de Moura and Sebastian Ullrich, *The Lean 4 theorem prover and programming language*, Automated Deduction—CADE 28, Lecture Notes in Computer Science, vol. 12699, Springer, 2021, https://doi.org/10.1007/978-3-030-79876-5_37, pp. 625–635.
2. Masayoshi Nagata, *A remark on the unique factorization theorem*, Journal of the Mathematical Society of Japan **9** (1957), no. 1, 143–145, <https://doi.org/10.2969/jmsj/00910143>.
3. Palomar Registry, *Arthur742Ramos/NagataFactoriality*, 2026, PALOMAR-2026-08-28-000002, version 1; registered August 28, 2026. <https://palomar-registry.org/entry?id=PALOMAR-2026-08-28-000002&version=1>.
4. ———, *Mechanical verification report for PALOMAR-2026-08-28-000002 version 1*, 2026, Checked August 27, 2026, 23:55:51 UTC; historical verification evidence. <https://data.palomar-registry.org/evidence/PALOMAR-2026-08-28-000002-v1/8ba8ccd66d9c217c4a795335c444c43f0ac06896ce37016e2262784d45dd4638/mechanical-report.json>.
5. Arthur F. Ramos, Ruy J. G. B. de Queiroz, and Anjolina G. de Oliveira, *A prime-generated formalization of Nagata’s factoriality theorem in Lean 4*, arXiv:2604.05238v1, 2026, Submitted April 6, 2026. <https://arxiv.org/abs/2604.05238v1>.
6. Arthur Freitas Ramos, David Barros Hulak, and Ruy J. G. B. de Queiroz, *NagataFactoriality: registered Lean source artifact*, 2026, Commit 9ebcfa77a4f7eab6fdb663e7f151ceeb48abddf7; Apache-2.0. <https://github.com/Arthur742Ramos/NagataFactoriality/tree/9ebcfa77a4f7eab6fdb663e7f151ceeb48abddf7>.
7. Arthur Freitas Ramos, David Barros Hulak, and Ruy Jose Guerra Barretto de Queiroz, *Nagata factoriality*, Archive of Formal Proofs, 2026, April 20, 2026. Isabelle/HOL formal proof development. <https://isa-afp.org/entries/Nagata-Factoriality.html>.
8. The mathlib Community, *The Lean mathematical library*, Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, 2020, <https://doi.org/10.1145/3372885.3373824>, pp. 367–381.
9. The Stacks Project Authors, *Nagata’s criterion for factoriality*, The Stacks Project, Lemma 10.120.7, Tag 0AFU, 2026, Accessed October 1, 2026. <https://stacks.math.columbia.edu/tag/0AFU>.