1. Introduction

This document provides a natural language companion to the `StructureSheaf.lean` file in Mathlib4. The file constructs the structure sheaf on the prime spectrum $\mathrm{Spec}(R)$ of a commutative ring R, which is fundamental to the definition of affine schemes.

The structure sheaf provides the "functions" on the prime spectrum, generalizing the notion of rational functions on algebraic varieties. The key insight is that sections over an open set should be functions that are "locally fractions" - that is, they can be locally expressed as ratios a/f where $a, f \in R$ and f does not vanish on the open set.

2. Theoretical Background

Following Hartshorne's approach, for an open set $U \subseteq \mathrm{Spec}(A)$, we define $\mathcal{O}(U)$ to be the set of functions $s : U \to \bigsqcup_{\mathfrak{p} \in U} A_{\mathfrak{p}}$, such that:

- $s(\mathfrak{p}) \in A_{\mathfrak{p}}$ for each $\mathfrak{p}$
- s is locally a quotient of elements of A

The modern approach uses dependent functions $\Pi x : U, \mathrm{Localizations}\ R\ x$ instead of functions into disjoint unions.

3. Basic Definitions

3.1. Localizations at Points

```
def PrimeSpectrum.Top : TopCat := TopCat.of (PrimeSpectrum R)

def Localizations (P : PrimeSpectrum.Top R) : Type u :=
  Localization.AtPrime P.asIdeal

instance commRingLocalizations (P : PrimeSpectrum.Top R) : CommRing <| Localizations
R P
instance localRingLocalizations (P : PrimeSpectrum.Top R) : IsLocalRing <|
Localizations R P
```

Natural Language: For each prime ideal $\mathfrak{p} \in \mathrm{Spec}(R)$, we have the localization $R_{\mathfrak{p}}$, which is a local ring. These localizations will serve as the "stalks" of our sheaf.

4. The Fraction Condition

4.1. Basic Fraction Property

```
def IsFraction {U : Opens (PrimeSpectrum.Top R)} (f : ∀ x : U, Localizations R x) :
Prop :=
  ∃ r s : R, ∀ x : U, s ∉ x.1.asIdeal ∧ f x * algebraMap _ _ s = algebraMap _ _ r
```

Natural Language: A dependent function f is a fraction if there exist global elements $r, s \in R$ such that at every point $x \in U$, we have $s \notin x$ (so s is invertible in R_x) and $f(x) = r/s$ in the localization R_x.

4.2. Characterization as Fractions

```
theorem IsFraction.eq_mk' {U : Opens (PrimeSpectrum.Top R)} {f : ∀ x : U,
Localizations R x}
    (hf : IsFraction f) :
    ∃ r s : R,
      ∀ x : U,
```

```
        ∃ hs : s ∉ x.1.asIdeal,
          f x =
            IsLocalization.mk' (Localization.AtPrime _) r
              (⟨s, hs⟩ : (x : PrimeSpectrum.Top R).asIdeal.primeCompl)
```

Natural Language: If a function is a fraction, then it can be explicitly written as r/s using the localization construction at each point, where s is in the prime complement of each prime ideal.

4.3. Prelocal Property

```
def isFractionPrelocal : PrelocalPredicate (Localizations R) where
  pred {_} f := IsFraction f
  res := by rintro V U i f ⟨r, s, w⟩; exact ⟨r, s, fun x => w (i x)⟩
```

Natural Language: The property of being a fraction is "prelocal": if it holds on an open set U, it also holds on any open subset $V \subseteq U$. This is because the same fraction representation works on smaller open sets.

5. Local Fractions

5.1. The Local Fraction Condition

```
def isLocallyFraction : LocalPredicate (Localizations R) :=
  (isFractionPrelocal R).sheafify

theorem isLocallyFraction_pred {U : Opens (PrimeSpectrum.Top R)} (f : ∀ x : U,
Localizations R x) :
    (isLocallyFraction R).pred f =
      ∀ x : U,
        ∃ (V : _) (_ : x.1 ∈ V) (i : V ⟶ U),
          ∃ r s : R,
            ∀ y : V, s ∉ y.1.asIdeal ∧ f (i y : U) * algebraMap _ _ s = algebraMap _
_ r
```

Natural Language: A function is locally a fraction if around every point, there exists a neighborhood where the function can be expressed as a single fraction r/s. This is the sheafification of the basic fraction condition.

5.2. Subring Structure

```
def sectionsSubring (U : (Opens (PrimeSpectrum.Top R))ᵒᵖ) :
    Subring (∀ x : U.unop, Localizations R x) where
  carrier := { f | (isLocallyFraction R).pred f }
  zero_mem' := -- proof that 0 is locally a fraction
  one_mem' := -- proof that 1 is locally a fraction
  add_mem' := -- proof that sum of local fractions is local fraction
  mul_mem' := -- proof that product of local fractions is local fraction
  neg_mem' := -- proof that negative of local fraction is local fraction
```

Natural Language: The functions that are locally fractions form a subring of all dependent functions. This gives us the ring structure we need for our structure sheaf.

6. The Structure Sheaf Construction

6.1. Sheaf in Types

```
def structureSheafInType : Sheaf (Type u) (PrimeSpectrum.Top R) :=
  subsheafToTypes (isLocallyFraction R)
```

Natural Language: First, we construct the structure sheaf as a sheaf of types, where sections are locally constant fractions.

6.2. Ring Structure on Sections

```
instance commRingStructureSheafInTypeObj (U : (Opens (PrimeSpectrum.Top R))ᵒᵖ) :
    CommRing ((structureSheafInType R).1.obj U) :=
  (sectionsSubring R U).toCommRing

def structurePresheafInCommRing : Presheaf CommRingCat (PrimeSpectrum.Top R) where
  obj U := CommRingCat.of ((structureSheafInType R).1.obj U)
  map i := CommRingCat.ofHom ((structureSheafInType R).1.map i)
```

Natural Language: The sections carry a natural commutative ring structure inherited from the subring of locally constant fractions. This gives us a presheaf of commutative rings.

6.3. The Main Structure Sheaf

```
def Spec.structureSheaf : Sheaf CommRingCat (PrimeSpectrum.Top R) :=
  ⟨structurePresheafInCommRing R, isSheaf_of_isSheaf_comp _ (structureSheafInType
R).2⟩
```

Natural Language: The structure sheaf on Spec(R) is the sheaf of commutative rings whose sections over an open set are functions that are locally expressible as fractions of elements from R.

7. Basic Operations

7.1. Constant Functions

```
def const (f g : R) (U : Opens (PrimeSpectrum.Top R))
    (hu : ∀ x : PrimeSpectrum.Top R, x ∈ U → g ∉ x.asIdeal) :
    (structureSheaf R).1.obj (op U) :=
  ⟨fun x : U => algebraMap _ _ f * (algebraMap _ _ g)⁻¹, ⟨f, g, fun x => ⟨hu x.1 x.2,
by simp⟩⟩⟩
```

Natural Language: For elements $f, g \in R$ where g doesn't vanish on an open set U, we can form the constant function f/g as a section over U. This gives us a way to embed ring elements into the structure sheaf.

7.2. Properties of Constants

```
theorem const_zero (f : R) (U hu) : const R 0 f U hu = 0
theorem const_one (U) : (const R 1 1 U fun _ _ => Submonoid.one_mem _) = 1
theorem const_add (f₁ f₂ g₁ g₂ : R) (U hu₁ hu₂) :
    const R f₁ g₁ U hu₁ + const R f₂ g₂ U hu₂ = const R (f₁ * g₂ + f₂ * g₁) (g₁ * g₂)
U (by ...)
theorem const_mul (f₁ f₂ g₁ g₂ : R) (U hu₁ hu₂) :
    const R f₁ g₁ U hu₁ * const R f₂ g₂ U hu₂ = const R (f₁ * f₂) (g₁ * g₂) U
(by ...)
```

Natural Language: Constant functions behave exactly as expected: zero gives the zero section, the constant 1/1 gives the unit section, and arithmetic operations on constants correspond to the appropriate fraction arithmetic.

8. Global Sections and Ring Homomorphisms

8.1. From Ring to Opens

```
def toOpen (U : Opens (PrimeSpectrum.Top R)) :
    CommRingCat.of R → (structureSheaf R).1.obj (op U) :=
  ⟨fun f => const R f 1 U fun _ _ => Submonoid.one_mem _, by ring⟩
```

Natural Language: Any ring element $f \in R$ can be viewed as a constant function $f/1$ on any open set. This gives us a canonical ring homomorphism from R to the sections over any open set.

8.2. From Ring to Stalks

```
def toStalk (x : PrimeSpectrum.Top R) : CommRingCat.of R → (structureSheaf
R).presheaf.stalk x :=
  toOpen R ⊤ ≫ (structureSheaf R).presheaf.germ ⟨x, trivial⟩
```

Natural Language: We can also map ring elements to any stalk by first viewing them as global sections, then taking the germ at the desired point.

8.3. Relationship Between Opens and Stalks

```
theorem toOpen_germ (U : Opens (PrimeSpectrum.Top R)) (x : PrimeSpectrum.Top R) (hx :
x ∈ U) :
    toOpen R U ≫ (structureSheaf R).presheaf.germ ⟨x, hx⟩ = toStalk R x

theorem germ_toOpen (U : Opens (PrimeSpectrum.Top R)) (x : PrimeSpectrum.Top R) (hx :
x ∈ U) (f : R) :
    (structureSheaf R).presheaf.germ ⟨x, hx⟩ (toOpen R U f) = toStalk R x f
```

Natural Language: Taking the germ of a constant section is the same as mapping the element directly to the stalk. This shows the consistency of our constructions.

9. Key Isomorphisms

9.1. Stalk Isomorphism

```
def stalkIso (x : PrimeSpectrum.Top R) :
    (structureSheaf R).presheaf.stalk x ≅ CommRingCat.of (Localization.AtPrime
x.asIdeal) :=
  { hom := stalkToFiberRingHom R x
    inv := localizationToStalk R x
    hom_inv_id := -- proof
    inv_hom_id := -- proof }
```

Natural Language: The stalk of the structure sheaf at a prime $\mathfrak{p}$ is canonically isomorphic to the localization $R_{\mathfrak{p}}$. This is the fundamental connection between the geometric (sheaf) and algebraic (localization) perspectives.

9.2. Basic Open Isomorphism

```
def basicOpenIso (f : R) :
    (structureSheaf R).1.obj (op (PrimeSpectrum.basicOpen f)) ≅ CommRingCat.of
(Localization.Away f) :=
  { hom := toBasicOpen R f
    inv := -- inverse map
    hom_inv_id := -- proof
    inv_hom_id := -- proof }
```

Natural Language: The sections of the structure sheaf over a basic open $D(f) = \{x : f \notin x\}$ are canonically isomorphic to the localization $R[f^{-1}]$. This shows that basic opens correspond exactly to localizations.

9.3. Global Sections Isomorphism

```
def globalSectionsIso : CommRingCat.of R ≅ (structureSheaf R).1.obj (op ⊤) :=
  { hom := toOpen R ⊤
    inv := -- inverse map
    hom_inv_id := -- proof
    inv_hom_id := -- proof }
```

Natural Language: The global sections of the structure sheaf are canonically isomorphic to the original ring R. This establishes that $\Gamma(\mathrm{Spec}(R), \mathcal{O}) \simeq R$, a fundamental property of affine schemes.

10. Unit and Invertibility Properties

10.1. Units in Basic Opens

```
theorem isUnit_to_basicOpen_self (f : R) : IsUnit (toOpen R (PrimeSpectrum.basicOpen
f) f) :=
```

Natural Language: An element $f \in R$ becomes a unit when viewed as a section over the basic open $D(f)$. This makes geometric sense: f is invertible exactly where it doesn't vanish.

10.2. Units in Stalks

```
theorem isUnit_toStalk (x : PrimeSpectrum.Top R) (f : x.asIdeal.primeCompl) :
    IsUnit (toStalk R x f) :=
```

Natural Language: Elements that are not in a prime ideal $\mathfrak{p}$ become units in the stalk at $\mathfrak{p}$. This reflects the definition of localization.

11. Localization Maps and Compatibility

11.1. Localization to Stalk

```
def localizationToStalk (x : PrimeSpectrum.Top R) :
    CommRingCat.of (Localization.AtPrime x.asIdeal) → (structureSheaf
R).presheaf.stalk x :=

theorem localizationToStalk_of (x : PrimeSpectrum.Top R) (f : R) :
    localizationToStalk R x (algebraMap _ _ f) = toStalk R x f

theorem localizationToStalk_mk' (x : PrimeSpectrum.Top R) (f : R) (s :
x.asIdeal.primeCompl) :
    localizationToStalk R x (IsLocalization.mk' _ f s) =
    (structureSheaf R).presheaf.germ ⟨x, trivial⟩ (toOpen R ⊤ f) *
    (toStalk R x s)⁻¹
```

Natural Language: There are canonical maps from localizations to stalks that respect the algebraic operations. The map sends f/s to the germ of f times the inverse of the germ of s.

11.2. Open to Localization

```
def openToLocalization (U : Opens (PrimeSpectrum.Top R)) (x : PrimeSpectrum.Top R)
(hx : x ∈ U) :
    (structureSheaf R).1.obj (op U) → CommRingCat.of (Localization.AtPrime
```

```
x.asIdeal) :=
  (structureSheaf R).presheaf.germ ⟨x, hx⟩ ≫ stalkToFiberRingHom R x
```

Natural Language: Sections over an open set containing a point can be evaluated at that point, giving an element of the localization at that point's prime ideal.

12. Computational Aspects

12.1. Explicit Section Formulas

```
theorem toOpen_apply (U : Opens (PrimeSpectrum.Top R)) (f : R) (x : U) :
    (toOpen R U f).val x = algebraMap _ _ f

theorem res_apply (U V : Opens (PrimeSpectrum.Top R)) (i : V → U)
    (s : (structureSheaf R).1.obj (op U)) (x : V) :
    ((structureSheaf R).1.map i.op s).val x = s.val (i x)
```

Natural Language: The explicit formulas show how sections behave: constant sections have the expected values, and restriction simply composes with the inclusion map.

13. Advanced Properties

13.1. Existence of Local Representations

```
theorem exists_const (U) (s : (structureSheaf R).1.obj (op U)) (x : PrimeSpectrum.Top
R)
    (hx : x ∈ U) : ∃ (V) (_ : x ∈ V) (i : V → U) (f g : R)
    (_ : ∀ y ∈ V, g ∉ y.asIdeal),
    (structureSheaf R).1.map i.op s = const R f g V _
```

Natural Language: Every section can be locally represented as a constant fraction. This is the key property that characterizes the structure sheaf: all sections are locally fractions.

13.2. Uniqueness and Extension

The structure sheaf construction ensures that:

- Sections are determined by their local behavior
- Local data can be glued to global sections when compatible
- The resulting sheaf has the correct stalks (localizations)
- Basic opens correspond to localizations

14. Conclusion

The structure sheaf construction provides the algebraic foundation for defining schemes. Key insights:

- **Local-Global Principle**: Functions are defined locally as fractions and glued globally
- **Stalks are Localizations**: The local rings of the sheaf are exactly the algebraic localizations
- **Basic Opens and Localization**: Geometric opens correspond to algebraic localizations
- **Global Sections Recovery**: The original ring is recovered as global sections

This construction bridges the gap between commutative algebra (rings, localizations, prime ideals) and algebraic geometry (spaces, sheaves, local rings), providing the foundation for the theory of schemes in Mathlib4.