1. Introduction

This document provides a natural language companion to the AffineScheme.lean file in Mathlib4. The file defines affine schemes as the essential image of the Spec functor and establishes key properties about affine schemes and affine open sets.

2. Main Definitions

2.1. The Category of Affine Schemes

```
def AffineScheme :=
  Scheme.Spec.EssImageSubcategory
deriving Category
```

Natural Language: The category of affine schemes is defined as the essential image subcategory of the Spec functor. This captures precisely those schemes that are isomorphic to spectra of commutative rings.

2.2. The IsAffine Property

```
class IsAffine (X : Scheme) : Prop where
  affine : IsIso X.toSpecΓ
```

Natural Language: A scheme X is affine if and only if the canonical morphism $X \to \mathrm{Spec}(\Gamma(X, \top))$ is an isomorphism, where $\Gamma(X, \top)$ denotes the global sections of the structure sheaf.

2.3. The Canonical Isomorphism for Affine Schemes

```
def Scheme.isoSpec (X : Scheme) [IsAffine X] :
  X ≅ Spec Γ(X, ⊤) := asIso X.toSpecΓ
```

Natural Language: For any affine scheme X, there exists a canonical isomorphism between X and the spectrum of its global sections.

3. Key Theorems

3.1. Naturality of the Spec Isomorphism

```
theorem Scheme.isoSpec_hom_naturality {X Y : Scheme}
  [IsAffine X] [IsAffine Y] (f : X ⟶ Y) :
  X.isoSpec.hom ≫ Spec.map (f.appTop) = f ≫ Y.isoSpec.hom
```

Natural Language: The canonical isomorphisms to spectra are natural with respect to morphisms between affine schemes. This means the diagram commutes when we map between affine schemes and their corresponding spectra.

3.2. Uniqueness of Morphisms via Global Sections

```
lemma ext_of_isAffine {X Y : Scheme} [IsAffine Y]
  {f g : X ⟶ Y} (e : f.appTop = g.appTop) : f = g
```

Natural Language: Two morphisms into an affine scheme are equal if and only if they induce the same map on global sections. This reflects the fact that morphisms into affine schemes are completely determined by their behavior on global sections.

4. The Equivalence of Categories

4.1. The Spec Functor

```
def Spec : CommRingCatᵒᵖ ⥤ AffineScheme :=
  Scheme.Spec.toEssImage
```

Natural Language: The Spec functor maps from the opposite category of commutative rings to affine schemes. This is the restriction of the usual Spec functor to its essential image.

4.2. The Global Sections Functor

```
def Γ : AffineSchemeᵒᵖ ⥤ CommRingCat :=
  forgetToScheme.op ⋙ Scheme.Γ
```

Natural Language: The global sections functor Γ maps from the opposite category of affine schemes to commutative rings by taking the global sections of the structure sheaf.

4.3. The Main Equivalence

```
def equivCommRingCat : AffineScheme ≌ CommRingCatᵒᵖ :=
  equivEssImageOfReflective.symm
```

Natural Language: The category of affine schemes is equivalent to the opposite category of commutative rings. This is the fundamental duality between algebra and geometry in the affine case.

5. Affine Open Sets

5.1. Definition of Affine Opens

```
def IsAffineOpen {X : Scheme} (U : X.Opens) : Prop :=
  IsAffine U
```

Natural Language: An open subset U of a scheme X is called affine if the corresponding open subscheme is an affine scheme.

5.2. The Set of Affine Opens

```
def Scheme.affineOpens (X : Scheme) : Set X.Opens :=
  {U : X.Opens | IsAffineOpen U}
```

Natural Language: For any scheme X, we can consider the collection of all affine open subsets, which forms a set in the opens of X.

6. Properties of Affine Opens

6.1. Affine Opens Form a Basis

```
theorem isBasis_affine_open (X : Scheme) :
  Opens.IsBasis X.affineOpens
```

Natural Language: The affine open subsets form a topological basis for any scheme. This means every open set can be written as a union of affine opens.

6.2. Coverage by Affine Opens

```
theorem iSup_affineOpens_eq_top (X : Scheme) :
  ⨆ i : X.affineOpens, (i : X.Opens) = ⊤
```

Natural Language: Every scheme can be covered by affine open subsets. The supremum (union) of all affine opens equals the entire scheme.

6.3. Existence of Affine Neighborhoods

```
theorem exists_isAffineOpen_mem_and_subset {X : Scheme.{u}}
  {x : X} {U : X.Opens} (hxU : x ∈ U) :
  ∃ W : X.Opens, IsAffineOpen W ∧ x ∈ W ∧ W.1 ⊆ U
```

Natural Language: For any point x in an open set U of a scheme, there exists an affine open neighborhood W of x contained in U.

7. The IsAffineOpen Structure

7.1. The Canonical Isomorphism for Affine Opens

```
def isoSpec : ↑U ≅ Spec Γ(X, U) :=
  haveI : IsAffine U := hU
  U.toScheme.isoSpec ≪≫ Scheme.Spec.mapIso U.topIso.symm.op
```

Natural Language: For an affine open U of a scheme X, there is a canonical isomorphism between U (viewed as a scheme) and the spectrum of the sections over U.

7.2. The fromSpec Morphism

```
def fromSpec : Spec Γ(X, U) ⟶ X :=
  haveI : IsAffine U := hU
  hU.isoSpec.inv ≫ U.ι
```

Natural Language: For an affine open U, we have a canonical open immersion from $\mathrm{Spec}(\Gamma(X, U))$ into X whose image is precisely U.

7.3. Range of fromSpec

```
theorem range_fromSpec :
  Set.range hU.fromSpec.base = (U : Set X)
```

Natural Language: The image of the fromSpec morphism is exactly the open set U as a subset of X.

8. Preservation of Affine Opens

8.1. Image Under Open Immersions

```
theorem image_of_isOpenImmersion (f : X ⟶ Y)
  [H : IsOpenImmersion f] : IsAffineOpen (f ^U U)
```

Natural Language: The image of an affine open under an open immersion is again affine open.

8.2. Preimage Under Isomorphisms

```
theorem preimage_of_isIso {U : Y.Opens} (hU : IsAffineOpen U)
  (f : X ⟶ Y) [IsIso f] : IsAffineOpen (f ^{-1}U U)
```

Natural Language: The preimage of an affine open under an isomorphism is affine open.

9. Compactness Properties

9.1. Affine Opens are Quasi-Compact

```
protected theorem isCompact : IsCompact (U : Set X)
```

Natural Language: Every affine open subset is quasi-compact (compact in the scheme-theoretic sense).

9.2. Affine Schemes are Quasi-Compact

```
instance Scheme.compactSpace_of_isAffine (X : Scheme)
  [IsAffine X] : CompactSpace X
```

Natural Language: Every affine scheme is quasi-compact as a topological space.

10. Basic Opens in Affine Schemes

```
theorem isBasis_basicOpen (X : Scheme) [IsAffine X] :
  Opens.IsBasis (Set.range (X.basicOpen : Γ(X, ⊤) → X.Opens))
```

Natural Language: In an affine scheme, the basic open sets (corresponding to principal open subsets in the spectrum) form a topological basis.

10.1. Basic Opens are Affine

```
instance [IsAffine X] (r : Γ(X, ⊤)) : IsAffine (X.basicOpen r)
```

Natural Language: If X is an affine scheme and r is a global section, then the basic open set $D(r)$ is also affine. This is the scheme-theoretic analog of the fact that localizations of rings give affine schemes.

11. Localization Properties

11.1. Basic Opens and Localizations

```
theorem isLocalization_basicOpen :
  IsLocalization.Away f Γ(X, X.basicOpen f)
```

Natural Language: The sections over a basic open set $D(f)$ form the localization of the global sections away from f. This establishes the fundamental connection between geometric opens and algebraic localizations.

11.2. Stalk Localization

```
theorem isLocalization_stalk (x : U) :
  IsLocalization.AtPrime
    (X.presheaf.stalk x)
    (hU.primeIdealOf x).asIdeal
```

Natural Language: The stalk at a point x in an affine open U is the localization of $\Gamma(X, U)$ at the corresponding prime ideal. This provides the local-to-global principle for affine opens.

12. The Spec Target Image

12.1. Image Ideal for Morphisms to Spec

```
def specTargetImageIdeal (f : X → Spec A) : Ideal A :=
  Ideal.span (Set.range f.appTop)
```

Natural Language: For a morphism $f : X \to \mathrm{Spec}(A)$, the target image ideal is the ideal generated by the image of the map on global sections.

12.2. Factorization Through the Image

```
def specTargetImageFactorization (f : X ⟶ Spec A) :
  X ⟶ Spec (specTargetImage f)
```

Natural Language: Any morphism to a spectrum factors through the spectrum of its target image ring, which is the quotient by the kernel of the induced ring homomorphism.

13. Lifting and Quotient Properties

13.1. Lifting Morphisms Through Quotients

```
def Scheme.Hom.liftQuotient (f : X.Hom (Spec A)) (I : Ideal A)
  (h : ∀ x : X, f.base x ∈ (PrimeSpectrum.zeroLocus I : Set)) :
  X.Hom (Spec (A ⧸ I))
```

Natural Language: A morphism $f : X \to \mathrm{Spec}(A)$ whose image lies in the zero locus of an ideal I can be lifted to a morphism $X \to \mathrm{Spec}(A/I)$.

14. Zero Locus and Closed Sets

14.1. Characterization of Closed Sets in Affine Schemes

```
lemma eq_zeroLocus_of_isClosed_of_isAffine [IsAffine X] (s : Set X) :
  IsClosed s ↔ ∃ I : Ideal Γ(X, ⊤), s = X.zeroLocus I
```

Natural Language: In an affine scheme, every closed set is the zero locus of some ideal in the global sections. This establishes the correspondence between closed sets and radical ideals.

14.2. Preimage of Zero Locus

```
lemma toSpecΓ_preimage_zeroLocus (s : Set Γ(X, ⊤)) :
  X.toSpecΓ.base ⁻¹' PrimeSpectrum.zeroLocus s =
  X.zeroLocus (Ideal.span s)
```

Natural Language: The preimage of a zero locus under the canonical morphism to the spectrum is the zero locus of the ideal generated by the corresponding sections.

15. Union and Intersection Properties

15.1. Basic Opens Generate the Topology

```
theorem basicOpen_union_eq_self_iff (s : Set Γ(X, U)) :
  ⨆ f : s, X.basicOpen f.1 = U ↔
  Ideal.span s = ⊤
```

Natural Language: A collection of basic opens covers an affine open U if and only if the corresponding sections generate the unit ideal. This is the geometric manifestation of the fact that elements generate the unit ideal if and only if they have no common zeros.

15.2. Supremum of Basic Opens

```
lemma iSup_basicOpen_of_span_eq_top {X : Scheme} (U) (s : Set Γ(X, U))
  (hs : Ideal.span s = ⊤) : ⨆ f : s, X.basicOpen f.1 = U
```

Natural Language: If sections generate the unit ideal, then their corresponding basic opens cover the entire affine open.

16. Properties of Affine Open Covers

16.1. Local Properties on Affine Opens

```
theorem of_affine_open_cover {X : Scheme} {P : X.affineOpens → Prop}
  (hP : ∀ (U : X.affineOpens) (f : Γ(X, U)) (hf : X.basicOpen f ≤ U),
    P ⟨X.basicOpen f, (U : X.Opens).isAffineOpen.basicOpen f⟩ →
    P U)
  (hP' : ∀ (U : X.affineOpens) (s : Finset Γ(X, U))
    (hs : Ideal.span (s : Set Γ(X, U)) = ⊤),
    (∀ f : s, P ⟨X.basicOpen f.1, (U : X.Opens).isAffineOpen.basicOpen f⟩) →
    P U)
  (U : X.affineOpens) : P U
```

Natural Language: Properties of affine opens can be established by checking them on basic opens and using the fact that basic opens form a basis. This provides a powerful induction principle for proving statements about all affine opens.

17. Categorical Properties

17.1. Limits and Colimits

```
instance hasColimits : HasColimits AffineScheme.{u}
instance hasLimits : HasLimits AffineScheme.{u}
```

Natural Language: The category of affine schemes has all limits and colimits. These are computed via the equivalence with the opposite category of commutative rings.

17.2. Fullness and Faithfulness

```
instance Spec_full : Spec.Full
instance Spec_faithful : Spec.Faithful
instance Spec_essSurj : Spec.EssSurj
```

Natural Language: The Spec functor is fully faithful and essentially surjective, establishing that it gives an equivalence of categories between commutative rings (with reversed arrows) and affine schemes.