${}^{-1-1}2\ {}^{op op}2$
$\mathcal{U}\mathcal{U}1\ \nVdash\nVdash$
$:=:=2$

The Γ-Spec Adjunction in Mathlib4

A Companion to `GammaSpecAdjunction.lean`

1 Introduction

This document provides a natural language companion to the `GammaSpecAdjunction.lean` file in Mathlib4. The file establishes the fundamental adjunction between the global sections functor Γ and the spectrum functor Spec, which is one of the cornerstones of algebraic geometry.

The adjunction $\Gamma \dashv \mathrm{Spec}$ captures the duality between geometry and algebra: geometric objects (locally ringed spaces/schemes) correspond to algebraic objects (commutative rings) via this adjunction. This relationship is at the heart of the philosophy that "geometry is dual to algebra."

2 Mathematical Background

The adjunction establishes that for any locally ringed space X and commutative ring R:

$$\mathrm{Hom}(X, \mathrm{Spec}(R)) \cong \mathrm{Hom}(R, \Gamma(X, \mathcal{O}_X))$$

This means that morphisms from a locally ringed space to an affine scheme are in natural bijection with ring homomorphisms from the ring to the global sections of the space.

Since we're working with contravariant functors, the adjunction is technically between Γ^{op} and Spec, or equivalently $\mathrm{Spec}^{\mathrm{op}} \dashv \Gamma$.

3 The Unit: Canonical Map to Spec of Global Sections

3.1 Construction of the Underlying Map

```
def to SpecFun : X      PrimeSpectrum (  .obj (op X)) := fun x =>
  PrimeSpectrum.map (LocallyRingedSpace.  .map X.to Spec.c.app (op    )) x.1
```

Natural Language: For any locally ringed space X, we construct a continuous map from X to $\mathrm{Spec}(\Gamma(X, \mathcal{O}_X))$. Each point $x \in X$ maps to the prime ideal that is the kernel of the composition of the stalk map $\Gamma(X, \mathcal{O}_X) \to \mathcal{O}_{X,x}$ with the natural map to the residue field.

3.2 Relationship with Stalks and Units

```
theorem notMem_prime_iff_unit_in_stalk (r :   .obj (op X)) (x : X) :
    r     X.to SpecFun x     IsUnit (X.presheaf.germ ( x , trivial  : X.↩
        basicOpen r) r)
```

Natural Language: A global section r is not in the prime ideal corresponding to point x if and only if r becomes a unit in the stalk at x. This captures the fundamental relationship between the algebraic (prime ideal) and geometric (stalk) perspectives.

3.3 Basic Opens and Preimages

```
theorem to Spec_preimage_basicOpen_eq (r :   .obj (op X)) :
    X.to SpecFun      ' PrimeSpectrum.basicOpen r = X.basicOpen r
```

Natural Language: The preimage of a basic open set $D(r)$ in $\mathrm{Spec}(\Gamma(X, \mathcal{O}_X))$ under the canonical map is exactly the basic open set $D(r)$ in X. This shows that the map respects the basic open sets, which are fundamental to the topology.

3.4 Continuity

```
theorem to Spec_continuous : Continuous X.to SpecFun := by
  rw [continuous_iff_isClosed]
  intro S hS
  rw [PrimeSpectrum.isClosed_zeroLocus_iff] at hS
  obtain  T , rfl  := hS
  simp only [to SpecFun, Set.preimage_setOf_eq, PrimeSpectrum.mem_zeroLocus]
  rw [X.to Spec_preimage_zeroLocus_eq T]
  exact X.zeroLocus_isClosed T
```

Natural Language: The canonical map $X \to \mathrm{Spec}(\Gamma(X, \mathcal{O}_X))$ is continuous. This is shown by proving that preimages of closed sets (zero loci) are closed.

4 Sheaf Morphism Construction

4.1 Component Apps on Basic Opens

```
def to SpecCApp :
    (structureSheaf (  .obj (op X))).1.obj (op (basicOpen r))
    X.presheaf.obj (op (X.basicOpen r)) :=
  -- Complex construction involving localizations and restrictions
```

Natural Language: For each basic open $D(r)$ in $\mathrm{Spec}(\Gamma(X, \mathcal{O}_X))$, we need to construct compatible maps from the structure sheaf to the pullback of X's presheaf. This involves carefully handling localizations and the relationship between basic opens.

4.2 Compatibility Conditions

```
theorem to SpecCApp_spec : toOpen _ (basicOpen r)     X.to SpecCApp r = X.↩
    toTo SpecMapBasicOpen r :=
```

Natural Language: The constructed sheaf maps are compatible with the natural inclusions of ring elements into localizations. This ensures that our construction respects the ring structure.

5 The Complete Unit Morphism

5.1 As a Morphism of Locally Ringed Spaces

```
def to Spec : X     Spec.locallyRingedSpaceObj (  .obj (op X)) where
  -- Underlying continuous map
  base := X.to SpecBase
  -- Sheaf morphism
  c := X.to SpecSheafedSpace.c
  -- Proof that stalk maps are local ring homomorphisms
  isLocalAtTarget := -- proof
```

Natural Language: The complete unit morphism $X \to \mathrm{Spec}(\Gamma(X, \mathcal{O}_X))$ consists of the continuous map we constructed plus a compatible sheaf morphism. The additional requirement is that the induced stalk maps are local ring homomorphisms.

5.2 Stalk Map Properties

```
theorem toStalk_stalkMap_to Spec (x : X) :
    toStalk _ (X.to SpecFun x)      X.to Spec.stalkMap x = X.presheaf.germ ↩
        x  , trivial
```

Natural Language: The stalk maps induced by the unit morphism are compatible with the natural germ maps. This ensures that local information is preserved under the unit morphism.

6 Compatibility with Zero Loci

```
lemma to Spec_preimage_zeroLocus_eq {X : LocallyRingedSpace.{u}}
    (S : Set (   .obj (op X))) :
    X.to SpecFun        ' PrimeSpectrum.zeroLocus S = X.zeroLocus S
```

Natural Language: The unit morphism preserves zero loci: the preimage of $V(S)$ in $\mathrm{Spec}(\Gamma(X, \mathcal{O}_X))$ is exactly $V(S)$ in X. This is a crucial compatibility that ensures the morphism respects the closed subspace structure.

7 Triangle Identities

7.1 Left Triangle (Γ-Spec-Γ Identity)

```
theorem    _Spec_left_triangle    : toSpec  (   .obj (op X))      X.to Spec.c.app (↩
     op     ) =        _ := by
  rw [    toOpen_comp_to SpecCApp]
  exact to SpecCApp_spec _ (Set.mem_univ _)        , le_rfl
```

Natural Language: One of the triangle identities for the adjunction: composing the natural isomorphism $R \cong \Gamma(\mathrm{Spec}(R), \mathcal{O})$ with the unit at $\mathrm{Spec}(R)$ gives the identity. This expresses that "Spec undoes Gamma" on affine objects.

7.2 Right Triangle (Spec-Γ-Spec Identity)

```
theorem right_triangle (R : CommRingCat) :
    Spec.locallyRingedSpaceObj R.to Spec
    LocallyRingedSpace.Spec Identity.hom.app R =       _
```

Natural Language: The other triangle identity: composing the unit at a ring R with the counit gives the identity on R. This expresses that "Gamma undoes Spec" on rings.

8 The Adjunctions

8.1 Locally Ringed Space Level

```
def locallyRingedSpaceAdjunction :    .rightOp      Spec.toLocallyRingedSpace.{u}↩
     where
  unit := identityTo Spec
  counit := LocallyRingedSpace.Spec Identity.inv
  left_triangle := left_triangle
  right_triangle := right_triangle
```

Natural Language: The adjunction between Γ^{op} and Spec at the level of locally ringed spaces. The unit is the canonical map we constructed, and the counit is the natural isomorphism between rings and global sections of their spectra.

8.2 Scheme Level

```
def adjunction : Scheme.  .rightOp      Scheme.Spec.{u} where
  unit :=
  { app := fun X           locallyRingedSpaceAdjunction    .{u}.unit.app X.↩
      toLocallyRingedSpace
    naturality := fun X Y f =>
      Scheme.Hom.ext' (locallyRingedSpaceAdjunction.{u}.unit.naturality f.↩
          toLRSHom) }
  counit := locallyRingedSpaceAdjunction.counit
  left_triangle := -- lifting of locally ringed space triangle
  right_triangle := -- lifting of locally ringed space triangle
```

Natural Language: The adjunction lifts to the level of schemes. Since schemes are a full subcategory of locally ringed spaces, the adjunction can be transported, giving us the fundamental $\Gamma \dashv$ Spec adjunction in the category of schemes.

9 Adjunction Properties and Applications

9.1 Home-Set Bijection

```
theorem adjunction_homEquiv_apply {X : Scheme} {R : CommRingCat }
    (f : X      Spec.obj R) :
     Spec  .adjunction.homEquiv X R f =     locallyRingedSpaceAdjunction    .homEquiv↩
         X.1 R  f

lemma     Spec_adjunction_homEquiv_eq     {X : Scheme.{u}} {B : CommRingCat} (    : B ↩
          (X,    )) :
    (( Spec  .adjunction.homEquiv X (op B))   .op).appTop = (Scheme.  SpecIso   B)↩
        .hom
```

Natural Language: The adjunction provides a natural bijection between morphisms $X \to \mathrm{Spec}(R)$ and ring homomorphisms $R \to \Gamma(X, \mathcal{O}_X)$. The explicit formula shows how to translate between geometric morphisms and algebraic homomorphisms.

9.2 Counit Properties

```
instance isIso_adjunction_counit : IsIso  Spec  .adjunction.counit := by
  apply @NatIso.isIso_of_isIso_app
  intro R
  rw [adjunction_counit_app]
  infer_instance
```

Natural Language: The counit of the adjunction is a natural isomorphism. This means that every commutative ring is naturally isomorphic to the global sections of its spectrum, establishing the equivalence between rings and global sections of affine schemes.

10 Immediate Consequences

10.1 Fully Faithful Spec

```
instance Spec.fullyFaithful : FullyFaithful (LocallyRingedSpace.Spec) :=
  Spec .locallyRingedSpaceAdjunction.fullyFaithfulROfIsIsoCounit

instance Scheme.Spec.fullyFaithful : FullyFaithful Scheme.Spec :=
  Spec .adjunction.fullyFaithful_Of_IsIso_Counit
```

Natural Language: Since the counit is an isomorphism, the Spec functor is fully faithful. This means that the category of commutative rings (with arrows reversed) embeds as a full subcategory of locally ringed spaces/schemes.

10.2 Reflective Subcategory

```
instance : IsReflectiveSubcategory LocallyRingedSpace.Spec := by
  apply IsReflectiveSubcategory.mk
  exact  _ , Spec .locallyRingedSpaceAdjunction

instance : IsReflectiveSubcategory Scheme.Spec := by
  apply IsReflectiveSubcategory.mk
  exact  _ , Spec .adjunction
```

Natural Language: The image of the Spec functor (i.e., affine schemes) forms a reflective subcategory. This means every locally ringed space/scheme has a "best affine approximation" given by Spec($\Gamma(X, \mathcal{O}_X)$).

11 Technical Lemmas and Compatibilities

11.1 Extension Properties

```
theorem comp_ring_hom_ext {X : LocallyRingedSpace.{u}} {R : CommRingCat.{u}} {f↩
     : R        .obj (op X)}
   {g : X      Spec.locallyRingedSpaceObj R} (h : g.c.app (op     ) = f) :
   f     X.to Spec.c.app (op     ) = g.c.app (op     )      Spec.map f.c.app (op↩
         )
```

Natural Language: Technical compatibility results that ensure the adjunction works correctly with composition and various natural transformations. These are essential for proving the triangle identities and other properties.

11.2 Naturality of Constructions

```
def identityTo Spec :       LocallyRingedSpace.{u}       .rightOp     Spec.↩
     toLocallyRingedSpace where
  app := LocallyRingedSpace.to Spec
  naturality := -- proof that this is natural in X
```

Natural Language: The unit of the adjunction is natural: it commutes with morphisms of locally ringed spaces in the expected way. This naturality is crucial for the adjunction to be well-defined.

12 Conclusion

The Γ-Spec adjunction is fundamental to algebraic geometry, establishing the precise relationship between:

- **Geometric objects**: Locally ringed spaces and schemes
- **Algebraic objects**: Commutative rings
- **Geometric morphisms**: Morphisms of locally ringed spaces/schemes
- **Algebraic morphisms**: Ring homomorphisms (in opposite direction)

Key insights:

1. Every locally ringed space has a canonical map to the spectrum of its global sections
2. Every ring is naturally isomorphic to the global sections of its spectrum
3. These relationships are functorial and satisfy compatibility conditions
4. The affine world (spectra of rings) sits inside the general world as a reflective subcategory

This adjunction provides the foundation for understanding how algebraic properties translate to geometric properties and vice versa, making it possible to use algebraic methods to solve geometric problems and geometric intuition to understand algebraic phenomena.