$^{-1-1}2$ $^{opop}2$
$\mathcal{U}\mathcal{U}1$ ⊮⊮
:=:=2

Restriction of Schemes and Morphisms in Mathlib4

A Companion to `Restrict.lean`

1 Introduction

This document provides a natural language companion to the `Restrict.lean` file in Mathlib4. The file develops the theory of restricting schemes to open subsets and restricting morphisms accordingly. This is fundamental to the local nature of algebraic geometry, where properties are often studied by restricting to smaller open sets.

2 Open Subsets as Schemes

2.1 Coercion to Schemes

```
coe
def toScheme {X : Scheme.{u}} (U : X.Opens) : Scheme.{u} :=
  X.restrict U.isOpenEmbedding

instance : CoeOut X.Opens Scheme :=     toScheme
```

Natural Language: Every open subset U of a scheme X can be viewed as a scheme in its own right through the restriction construction. This creates a coercion that allows us to treat opens as schemes directly.

2.2 The Inclusion Morphism

```
def    :   U       X := X.ofRestrict _
```

Natural Language: For each open subset $U \subseteq X$, there is a canonical inclusion morphism $\iota : U \to X$ that embeds U as an open subscheme of X.

2.3 Open Immersion Property

```
instance : IsOpenImmersion U.
```

Natural Language: The inclusion morphism $U \to X$ is an open immersion, meaning it identifies U with its image as an open subset and preserves the scheme structure.

3 Presheaf Properties

3.1 Presheaf Objects

```
lemma toScheme_presheaf_obj (V) :   (U, V) =   (X, U.   ^U V)
```

Natural Language: Sections over an open set V in the restricted scheme U correspond exactly to sections over the image $U.\iota(V)$ in the original scheme X.

3.2 Presheaf Maps

```
simp
lemma toScheme_presheaf_map {V W} (i : V     W) :
    U.toScheme.presheaf.map i = X.presheaf.map (U.  .opensFunctor.map i.unop).op
```

Natural Language: Restriction maps in the presheaf on U are obtained by pushing forward along ι to get restriction maps in the presheaf on X.

4 Application Maps

4.1 Basic Application

```
simp
lemma  _app  (V) : U.  .app V = X.presheaf.map
    (homOfLE (x := U.   ^U U.   ^{-1}U V) (Set.image_preimage_subset _ _)).op
```

Natural Language: The application of ι to sections is given by the natural restriction map in X's presheaf.

4.2 Top Application

```
simp
lemma  _appTop  :
    U.  .appTop = X.presheaf.map (homOfLE (x := U.   ^U    ) le_top).op
```

Natural Language: Taking global sections of U corresponds to taking sections of X over the image of U.

5 Range Properties

5.1 Opens Range

```
simp
lemma opensRange_  : U.  .opensRange = U
```

Natural Language: The image of ι as an open subset is exactly U itself.

5.2 Topological Range

```
simp
lemma range_  : Set.range U.  .base = U
```

Natural Language: The image of the underlying continuous map is exactly the subset $U \subseteq X$.

6 Global Sections Isomorphism

6.1 Top Isomorphism

```
simps!
def topIso :   (U,    )       (X, U) :=
  X.presheaf.mapIso (eqToIso U.  _image_top  .symm).op
```

Natural Language: The global sections of the restricted scheme U are naturally isomorphic to the sections of X over the open set U. This establishes the fundamental relationship between restriction and sections.

7 Stalks

7.1 Stalk Isomorphism

```
def stalkIso {X : Scheme.{u}} (U : X.Opens) (x : U) :
    U.toScheme.presheaf.stalk x     X.presheaf.stalk x.1
```

Natural Language: The stalk of the structure sheaf of U at a point x is naturally isomorphic to the stalk of X's structure sheaf at the same point. This shows that local properties are preserved under restriction.

7.2 Germ Compatibility

```
reassoc (attr := simp)
lemma germ_stalkIso_hom {X : Scheme.{u}} (U : X.Opens)
    {V : U.toScheme.Opens} (x : U) (hx : x     V) :
      U.toScheme.presheaf.germ V x hx     (U.stalkIso x).hom =
        X.presheaf.germ (U.   ^U V) x.1   x  , hx,  r f l
```

Natural Language: Germs (local sections) behave compatibly with the stalk isomorphism, ensuring that the restriction preserves all local algebraic structure.

8 Restriction Functors and Equivalences

8.1 Opens Restriction

```
simps!
def opensRestrict :
    Scheme.Opens U     { V : X.Opens // V     U }
```

Natural Language: Open subsets of the restricted scheme U correspond bijectively to open subsets of X that are contained in U.

8.2 Restrict Functor

```
simps! obj_left obj_hom map_left
def Scheme.restrictFunctor : X.Opens     Over X where
  obj U := Over.mk U.
  map {U V} i := Over.homMk (X.homOfLE i.le) (by simp)
```

Natural Language: There is a functor from the category of opens of X to schemes over X, sending each open U to the restricted scheme $U \to X$.

9 Morphisms Between Opens

9.1 Inclusion of Smaller Opens

```
protected noncomputable
def Scheme.homOfLE (X : Scheme.{u}) {U V : X.Opens} (e : U     V) : (U : Scheme.{
    u})     V
```

Natural Language: If $U \subseteq V$ are open subsets of X, there is a natural morphism from the scheme U to the scheme V.

9.2 Compatibility with Inclusion

```
reassoc (attr := simp)
lemma Scheme.homOfLE_  (X : Scheme.{u}) {U V : X.Opens} (e : U     V) :
    X.homOfLE e     V.   = U.
```

Natural Language: The composition $U \to V \to X$ equals the direct inclusion $U \to X$.

10 Basic Opens and Localizations

10.1 Basic Open Mapping

```
lemma Scheme.map_basicOpen (r :   (U,    )) :
    U.   ^U U.toScheme.basicOpen r = X.basicOpen
      (X.presheaf.map (eqToHom U.isOpenEmbedding_obj_top.symm).op r)
```

Natural Language: The image under ι of a basic open $D(r)$ in U equals a basic open in X determined by the corresponding section.

10.2 Basic Open Isomorphism to Localization

```
def basicOpenIsoSpecAway {R : CommRingCat.{u}} (f : R) :
    Scheme.Opens.toScheme (X := Spec R) (PrimeSpectrum.basicOpen f)     Spec(
        Localization.Away f)
```

Natural Language: For an affine scheme $\mathrm{Spec}(R)$, the basic open $D(f)$ is isomorphic to $\mathrm{Spec}(R[1/f])$, establishing the connection between geometric opens and ring-theoretic localizations.

11 Morphism Restriction

11.1 Pullback-Restriction Isomorphism

```
def pullbackRestrictIsoRestrict {X Y : Scheme.{u}} (f : X     Y) (U : Y.Opens) :
    pullback f (U.  )     f ^{-1}U U
```

Natural Language: The pullback of a morphism $f : X \to Y$ along the inclusion $U \to Y$ is isomorphic to the restriction of X to the preimage $f^{-1}(U)$.

11.2 Morphism Restriction Definition

```
def morphismRestrict {X Y : Scheme.{u}} (f : X     Y) (U : Y.Opens) : (f ^{-1}U U
    ).toScheme     U :=
  (pullbackRestrictIsoRestrict f U).inv     pullback.snd _ _

infixl:85 "   _   " => morphismRestrict
```

Natural Language: Given a morphism $f : X \to Y$ and an open $U \subseteq Y$, we can restrict f to get a morphism $f|_U : f^{-1}(U) \to U$. This uses the notation $f\,|_U$.

12 Properties of Morphism Restriction

12.1 Compatibility with Inclusions

```
reassoc (attr := simp)
theorem morphismRestrict_ {X Y : Scheme.{u}} (f : X Y) (U : Y.Opens) :
    (f _ U) U. = (f ^{-1}U U). f
```

Natural Language: The restricted morphism is compatible with the inclusion maps in the sense that the diagram commutes.

12.2 Identity Restriction

```
simp
lemma morphismRestrict_id {X : Scheme.{u}} (U : X.Opens) : X _ U = _
    := by
  rw [ cancel_mono U. , morphismRestrict_ , Category.comp_id, Category.
      id_comp]
```

Natural Language: Restricting the identity morphism gives the identity on the restricted scheme.

12.3 Composition Property

```
theorem morphismRestrict_comp {X Y Z : Scheme.{u}} (f : X Y) (g : Y Z) (U
    : Opens Z) :
    (f g) _ U = f _ g ^{-1}U U g _ U
```

Natural Language: Restricting a composition of morphisms equals the composition of the appropriate restrictions.

13 Base Change Properties

13.1 Continuous Map Base

```
theorem morphismRestrict_base {X Y : Scheme.{u}} (f : X Y) (U : Y.Opens) :
      (f _ U).base = U.1.restrictPreimage f.base
```

Natural Language: The underlying continuous map of the restricted morphism is the restriction of the original continuous map to the appropriate preimage.

13.2 Pullback Isomorphism

```
theorem isPullback_morphismRestrict {X Y : Scheme.{u}} (f : X Y) (U : Y.Opens
    ) :
    IsPullback (f _ U) (f ^{-1}U U). U. f
```

Natural Language: The restriction construction gives a genuine pullback square, confirming that it captures the correct universal property.

14 Stalk Maps and Localization

14.1 Stalk Map Compatibility

```
def morphismRestrictStalkMap {X Y : Scheme.{u}} (f : X     Y) (U : Y.Opens) (x) :
    Arrow.mk ((f   _   U).stalkMap x)     Arrow.mk (f.stalkMap x.1)
```

Natural Language: The stalk map of a restricted morphism is naturally isomorphic to the stalk map of the original morphism, showing that local algebraic properties are preserved.

15 Advanced Restriction Constructions

15.1 Double Restriction

```
def morphismRestrictRestrict {X Y : Scheme.{u}} (f : X     Y) (U : Y.Opens) (V :
    U.toScheme.Opens) :
    Arrow.mk (f   _   U   _   V)     Arrow.mk (f   _   U.   ^U V)
```

Natural Language: Restricting twice is isomorphic to a single restriction to the appropriate open subset.

15.2 Basic Open Restriction

```
def morphismRestrictRestrictBasicOpen {X Y : Scheme.{u}} (f : X     Y) (U : Y.
    Opens) (r :   (Y, U)) :
    Arrow.mk (f   _   U   _   U.toScheme.basicOpen (Y.presheaf.map (eqToHom U.
        isOpenEmbedding_obj_top).op r))
      Arrow.mk (f   _   Y.basicOpen r)
```

Natural Language: Restricting first to an open U and then to a basic open determined by a section on U is isomorphic to restricting directly to the corresponding basic open in Y.

16 Applications to Covers

16.1 Open Cover Restriction

```
simps! J obj map
noncomputable
def Scheme.OpenCover.restrict {X : Scheme.{u}} (     : X.OpenCover) (U : Opens X)
     :
    U.toScheme.OpenCover
```

Natural Language: Given an open cover of X, we can restrict it to get an open cover of any open subset $U \subseteq X$.

17 Conclusion

The restriction theory developed in `Restrict.lean` provides:

17.1 Foundational Results

- Every open subset naturally carries a scheme structure
- Inclusion morphisms are open immersions
- Local properties (stalks, germs) are preserved under restriction

17.2 Functorial Properties

- Restriction gives functors between appropriate categories
- Global sections of restricted schemes relate naturally to sections on opens
- Morphisms can be restricted in a way that preserves composition and identities

17.3 Connections to Commutative Algebra

- Basic opens in affine schemes correspond to localizations
- Stalks of restricted schemes are unchanged
- Restriction preserves the local ring structure

17.4 Geometric Applications

- Provides the foundation for studying local properties of schemes
- Enables the use of open covers to reduce global problems to local ones
- Essential for constructions like gluing and descent theory

This restriction theory is fundamental to algebraic geometry, enabling the systematic study of local properties and the construction of schemes via gluing local pieces.