Computability of the étale Euler-Poincaré characteristic

Jin, J.

1 Effective algebraic geometry

In this chapter we describe the basics of computations in algebraic geometry. We start by explaining in Sections 1.1 to 1.4 the view on computability taken in this dissertation, before treating the basic constructions in algebraic geometry that we will need for the algorithm described in the later chapters.

1.1 Primitive recursive functions and computability

In order to algorithmically compute with mathematical objects, one first needs to be able to present these objects into some computational model. There are a number of classical such models, e.g. that of the *Turing machine*, the *random-access machine* (or *RAM*), and that of the *recursive functions*. We wish to be able to describe a theory of algorithms that are "bounded" in some way; this can be done the most naturally in the theory of recursive functions, in which we have a class of *primitive recursive functions*.

A modern treatment on (primitive) recursive functions can be found in most books on computability; the following treatment is based on that of Moret [31].

We will define the set of primitive recursive functions as a subset of $\coprod_{n=0}^{\infty} \mathbb{N}^{\mathbb{N}^n}$; note that $\mathbb{N}^{\mathbb{N}^0} = \mathbb{N}$.

Definition 1.1. The *base functions* are the following:

- the constant $0 \in \mathbb{N}$;
- the *successor function* $S\colon \mathbb{N} \to \mathbb{N}$, $x \mapsto x + 1$;
- for positive integers n, i such that $i \leq n$, the *projection function* $P_i^n\colon \mathbb{N}^n \to \mathbb{N}$ on the i-th coordinate.

Next, we define the two operations under which we want the set of primitive recursive functions to be closed.

Definition 1.2. Let $m, n \geq 0$ be integers, and let $g\colon \mathbb{N}^m \to \mathbb{N}$, $h_1, \ldots, h_m\colon \mathbb{N}^n \to \mathbb{N}$ be functions. Then the function $\sigma_{m,n}(g, h_1, \ldots, h_m)\colon \mathbb{N}^n \to \mathbb{N}$ given by

$$(x_1, \ldots, x_n) \mapsto g\big(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n)\big)$$

is said to be obtained from $g, h_1, \ldots, h_n$ by *substitution*.

Note that in the edge case $m = 0$, the function $\sigma_{0,n}(g)$ is the constant function (from $\mathbb{N}^n$) with value g; in the other edge case $n = 0$, the function $\sigma_{m,0}(g, h_1, \ldots, h_m)$ is $g(h_1, \ldots, h_m) \in \mathbb{N}$.

Definition 1.3. Let n be a positive integer, and let $g\colon \mathbb{N}^{n-1} \to \mathbb{N}$ and $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ be functions. Then the function $\rho_n(g,h)\colon \mathbb{N}^n \to \mathbb{N}$ given recursively by

$$(x_1,\dots,x_n) \mapsto \begin{cases} g(x_2,x_3,\dots,x_n) & \text{if } x_1 = 0 \\ h\big(x_1-1,\rho_n(g,h)(x_1-1,x_2,\dots,x_n),x_2,\dots,x_n\big) & \text{if } x_1 > 0 \end{cases}$$

is said to be obtained from g,h by *primitive recursion*.

Now we can define the set of primitive recursive functions.

Definition 1.4. The set R_p of *primitive recursive functions* is the smallest subset of the set $\coprod_{n=0}^{\infty} \mathbb{N}^{\mathbb{N}^n}$ that contains the base functions, and such that

- for all non-negative integers m,n, and all $g\colon \mathbb{N}^m \to \mathbb{N}$, $h_1,\dots,h_m\colon \mathbb{N}^n \to \mathbb{N}$ such that $g,h_1,\dots,h_m \in R_p$, we have $\sigma_{m,n}(g,h_1,\dots,h_m) \in R_p$;
- for all positive integers n, and all functions $g\colon \mathbb{N}^{n-1} \to \mathbb{N}$, $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ such that $g,h \in R_p$, we have $\rho_n(g,h) \in R_p$.

An *algorithm* computing a certain primitive recursive function f is in this context an explicit expression of f in terms of the base functions, substitution, and primitive recursion.

Example 1.5. The function $d\colon \mathbb{N} \to \mathbb{N}$ given by $x \mapsto \max(x-1,0)$ is primitive recursive. An algorithm computing it is

$$\rho_1(0,P_1^2).$$

For any primitive recursive function $f\colon \mathbb{N} \to \mathbb{N}$ (together with an algorithm computing it), the function $i_f\colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ defined by $(n,x) \mapsto f^n(x)$ is primitive recursive, and an algorithm computing i_f is given by

$$\rho_2\big(P_1^1,\sigma_{1,3}(f,P_2^3)\big).$$

Note that i_S is the addition map on $\mathbb{N}$.

Note that the primitive recursive functions form a strictly smaller class of functions than what are typically called *recursive* or *computable* functions. We get the usual notion back once we add the *unbounded minimisation operator*, and temporarily also consider partial functions $\mathbb{N}^n \to \mathbb{N}$.

Definition 1.6. Let $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ be a partial function. Then $\mu_f\colon \mathbb{N}^n \to \mathbb{N}$ is the partial function such that $\mu_f(x_1,\dots,x_n)$ is undefined whenever $f(y,x_1,\dots,x_n) \neq 0$ for all $y \in \mathbb{N}$, and such that $\mu_f(x_1,\dots,x_n) = y$ if $y \in \mathbb{N}$ is the minimal number such that $f(y,x_1,\dots,x_n) = 0$. We say that μ_f is obtained from f by *unbounded minimisation*.

This allows us to define the set of recursive functions.

Definition 1.7. The set R' of *partial recursive functions* is the smallest set of partial functions $\mathbb{N}^n \to \mathbb{N}$ (with varying n) that contains the base functions, and such that

- for all integers $m,n \geq 0$, and all partial functions $g\colon \mathbb{N}^m \to \mathbb{N}$ and all partial functions $h_1,\dots,h_m\colon \mathbb{N}^n \to \mathbb{N}$ with $g,h_1,\dots,h_m \in R'$, the partial function $\sigma_{m,n}(g,h_1,\dots,h_m)$ lies in R';

- for all integers $n > 0$, and all partial functions $g\colon \mathbb{N}^{n-1} \to \mathbb{N}$, $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ with $g, h \in R'$, we have $\rho_n(g,h) \in R'$;
- for all integers $n \geq 0$, and all partial functions $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ with $g \in R'$, we have $\mu_g \in R'$.

The set R of *recursive functions* is the subset of R' of total functions.

1.2 Explicitly given sets and maps

The following is essentially the theory of (primitive) recursive sets, as can be found in most books on computability, e.g. Moret [31]. We will view $\mathbb{N}$ as a pointed set with base point 0 in what follows; we will think of 0 as an "error code".

Definition 1.8. A *presentation* of a set X consists of an injective *presentation map* $\pi\colon X \to \mathbb{N} - \{0\}$ together with an algorithm computing the characteristic function $\chi_{\pi(X)}$ of $\pi(X)$. An *explicitly given set* is a pair (X, π_X) of a set and a presentation π_X of X.

Definition 1.9. Let $(X, \pi_X), (Y, \pi_Y)$ be explicitly given sets. A *presentation* of a map $f\colon Y \to X$ is an algorithm computing the unique function $\varphi\colon \mathbb{N} \to \mathbb{N}$ such that $\varphi(y) = 0$ for all $y \notin \pi_Y(Y)$, and such that the following diagram commutes.

$$\begin{array}{ccc} Y & \xrightarrow{\pi_Y} & \mathbb{N} \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \varphi} \\ X & \xrightarrow[\pi_X]{} & \mathbb{N} \end{array}$$

An *explicitly given map* $Y \to X$ is a map $Y \to X$ together with a presentation.

We obtain a collection $\mathrm{Set}_!$ of explicitly given sets and explicitly given maps, which only becomes a category after we identify algorithms defining the same map (in other words, we forget the algorithm). There is a forgetful functor $\mathrm{Set}_! \to \mathrm{Set}$, which is faithful by definition, but not full (as not every function is primitive recursive).

Example 1.10. Let, for any non-empty set X and any $x \in X$, the set $\mathrm{Seq}_{\infty,x}(X)$ denote the set of sequences $(x_i)_{i=0}^{\infty}$ such that $x_i = x$ for all but finitely many i. We first give the *Gödel encoding* of $\mathrm{Seq}_{\infty,0}(\mathbb{N})$. Let $p_0 = 2, p_1, \dots$ denote the increasing enumeration of the prime numbers. Then the Gödel encoding $\pi\colon \mathrm{Seq}_{\infty,0}(\mathbb{N}) \to \mathbb{N}$ sending $(a_0, a_1, \dots)$ to $p_0^{a_0} p_1^{a_1} \cdots$ is a presentation of $\mathrm{Seq}_{\infty,0}(\mathbb{N})$.

Now let X be a non-empty explicitly given set, and $x \in X$ an element such that $\pi_X(x) = 1$. The map $X \to \mathbb{N}$, $x \mapsto \pi_X(x) - 1$ then induces a presentation $\mathrm{Seq}_{\infty,x}(X) \to \mathrm{Seq}_{\infty,0}(\mathbb{N}) \to \mathbb{N}$, which sends the constant sequence x to 1. This allows us to iterate this process, obtaining presentations for e.g. $\mathrm{Seq}_{\infty,x}\big(\mathrm{Seq}_{\infty,x}(X)\big)$, etc.

Moreover, let, for any non-empty set X, the set $\mathrm{Seq}(X)$ denote the set of finite sequences in X. We then have an injective map $\mathrm{Seq}(\mathbb{N}) \to \mathrm{Seq}_{\infty,0}(\mathbb{N})$ sending $(a_1, \dots, a_n)$ to $(n, a_1, \dots, a_n)$, which induces a presentation of $\mathrm{Seq}(\mathbb{N})$. If X is an explicitly given set, then as before, the map $X \to \mathbb{N}$, $x \mapsto \pi_X(x) - 1$ induces a presentation $\mathrm{Seq}(X) \to \mathrm{Seq}(\mathbb{N}) \to \mathbb{N}$.

1.3 Explicitly given fields and factorial fields

In this section we will give a definition of a factorial field, cf. e.g. Ayoub [2]. We first give a definition of explicitly given rings and fields.

Definition 1.11. An *explicitly given ring* is an explicitly given set R that is a ring, together with elements $0, 1 \in R$, the characteristic of R, and encodings of the maps $+, \cdot \colon R \times R \to R$, $- \colon R \to R$. An *explicitly given morphism* $R \to S$ of explicitly given rings is an explicitly given map that is also a morphism of rings. An *explicitly given field* is an explicitly given ring k that is a field, together with a presentation of the map $\cdot^{-1} \colon k - \{0\} \to k - \{0\}$.

Remark 1.12. Note that at times, elements of fields are more naturally given as equivalence classes of elements of some set, e.g. the case of a fraction field of an integral domain. Therefore it may be more desirable to accommodate for this and define an explicitly given ring or field as an explicitly given set R together with a primitive recursive equivalence relation on R and the usual operations (which are to satisfy the usual relations only up to equivalence). However, since bounded minimisation is primitive recursive, so is the (characteristic function of the) set of minimal representatives of each equivalence class and the map $R \to R$ sending each x to its corresponding minimal representative. Therefore we can construct from such R an explicitly given ring or field in the sense of the definition above, and we lose no generality.

Example 1.13. The fields $\mathbb{F}_q$ (for q a prime power) and $\mathbb{Q}$ can be given the structure of an explicitly given field. Suppose that k is an explicitly given field. Any finitely generated extension of k can be given the structure of an explicitly given field. The field $k(x_1, x_2, \dots)$ can be given the structure of an explicitly given field.

For an explicitly given ring R, we will give $R[x]$ the structure of an explicitly given ring. First, identify $R[x]$ with $\mathrm{Seq}_{\infty,0}(R)$ by identifying a polynomial $f = \sum_{i=0}^{\infty} a_i x^i$ with the sequence $(a_i)_{i=0}^{\infty}$. Since we have obvious algorithms to compute addition, multiplication, and additive inverse, we get the structure of an explicitly given ring on $R[x]$. By iterating this process, one gets a structure of an explicitly given ring on the polynomial ring $R[x_1, \dots, x_n]$ as well.

Since we now have a presentation of polynomials and therefore also of finite sequences thereof, we can now introduce the notion of a polynomial factorisation algorithm.

Definition 1.14. A *factorial field* is an explicitly given field k, together with a presentation of a map $k[x] - \{0\} \to \mathrm{Seq}(k[x])$ sending f to a tuple $(f_1, \dots, f_n)$ such that $f = f_1 \cdots f_n$ and every f_i is irreducible.

Example 1.15. Any finitely generated extension of $\mathbb{F}_q$ (for q a prime power) or $\mathbb{Q}$ can be given the structure of a factorial field.

There exist explicitly given fields for which polynomial factorisation is not computable, see Fröhlich and Shepherdson [12].

1.4 Remarks on "algorithms" and "complexity"

First note that the notion of algorithm given above is not a very convenient notion to work with. However, there is a different way of describing primitive recursive functions which may be a bit more amenable, namely as so-called *loop programs* (see e.g. Handley and Wainer [20, Sec. 1]). Roughly speaking, these are algorithms using only finite loops of precomputed length (so no recursion is allowed a priori). Of course, as many algorithms use recursion, this is still a bit too restrictive in practice. In practice, we will also allow recursion if the total number of recursive calls for a single instance can be bounded by a precomputed number; it is possible to rewrite such recursively defined functions as a loop of a precomputed length. This allows us to discuss algorithms much more informally, and we will usually do so.

Note moreover that while the notion of explicitly given (or factorial) field described above is suitable for a notion of computability, it doesn't admit a good notion of *arithmetic complexity*, i.e. the number of field operations needed to compute a function (as a function in the input); as the field operations are assumed to be primitive recursive, they can be described in terms of base functions, the notion of a number of field operations isn't even well-defined! While the notion of an arithmetic complexity can be formalised, see e.g. Diem [10, Sec. 1.6.4] for RAMs, we will use the term informally, viewing the field operations of an explicitly given or factorial field as primitive operations.

Finally, note that in the definition of a factorial field, we have included a primitive recursive univariate factorisation algorithm, but in practice, some efficient such algorithms use randomisation, but halt with probability 1 and with the correct output, i.e. they are *Las Vegas* algorithms. Therefore algorithms involving factorisation should be viewed as Las Vegas algorithms in general. Other than that, we will usually ignore the difference between Las Vegas and deterministic algorithms.

1.5 Algebra over explicitly given fields

As stated in the previous section, we will be a lot less formal with algorithms from now on.

1.5.1 Vector spaces

We present a finite-dimensional vector space over k (with given basis) by its dimension, and a k-linear map from a vector space of dimension m to a vector space of dimension n by its $n \times m$-matrix with respect to the given bases.

If vector spaces V and V' have bases $(e_1, e_2, \dots, e_m)$ and $(e'_1, e'_2, \dots, e'_{m'})$, respectively, then we will assume their direct sum $V \oplus V'$ to be equipped with the basis $(e_1, e_2, \dots, e_m, e'_1, e'_2, \dots, e'_{m'})$, and their tensor product $V \otimes V'$ to be equipped with the basis $(e_i \otimes e'_{i'})_{i=1,i'=1}^{m,m'}$, with the lexicographical order on the indices. This has the additional advantage that if we have three vector spaces V, V', V'' with given bases, that then the natural isomorphisms $(V \otimes V') \otimes V'' \cong V \otimes (V' \otimes V'')$, $k \otimes V \cong V \cong V \otimes k$, and $(V \oplus V') \otimes V'' \cong (V \otimes V'') \oplus (V' \otimes V'')$ preserve the induced bases.

We present a subspace of dimension m of a given vector space of dimension n by an $n \times m$-matrix in reduced row echelon form. Therefore, by Gaussian elimination,

we can compute kernels and images of linear maps, in a number of field operations polynomially bounded by the dimensions of the source and target. Moreover, we can compute the quotient of a vector space by a subspace, and therefore we can compute cokernels of linear maps as well, also in a number of field operations polynomially bounded by the dimensions of the source and target.

1.5.2 Finitely generated algebras

We present an ideal I of $k[x_1, \ldots, x_m]$ by a finite set of generators $f_1, \ldots, f_s$. An algebra of finite type over k then is given by a non-negative integer m, and an ideal of $k[x_1, \ldots, x_m]$.

We present an element f of $k[x_1, \ldots, x_m]/I$ by an element of $f + I$ in $k[x_1, \ldots, x_m]$; note that sums and products of elements can be computed, and that equality of two elements can be tested using Gröbner basis algorithms. A k-algebra morphism $k[x_1, \ldots, x_m]/I \to k[y_1, \ldots, y_n]/J$ is given by the images of the x_i, under the condition that the generators of I map to 0 (which can be tested by the above). Moreover, compositions of morphisms can be computed, and equality of two morphisms can be tested using Gröbner basis algorithms.

We will consider more properties in Section 1.7.

1.5.3 Finite algebras

We describe two ways to present a finite k-algebra.

One way to present a finite k-algebra A is the *vector space presentation*, namely by its underlying vector space over k, together with the inclusion $\iota\colon k \to A$ and the multiplication map $\mu\colon A \otimes A \to A$; these are to be such that the following diagrams commute:

$$\begin{array}{ccc} A \otimes A \otimes A & \xrightarrow{\mu \otimes \mathrm{id}_A} & A \otimes A \\ {\scriptstyle \mathrm{id}_A \otimes \mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ A \otimes A & \xrightarrow[\mu]{} & A \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{\iota \otimes \mathrm{id}_A} & A \otimes A \\ {\scriptstyle \mathrm{id}_A \otimes \iota}\downarrow & & \downarrow{\scriptstyle \mu} \\ A \otimes A & \xrightarrow[\mu]{} & A \end{array}$$

and we present a morphism $A \to B$ of finite k-algebras by its underlying k-linear map; this map must be such that the following diagrams commute.

$$\begin{array}{ccc} A \otimes A & \xrightarrow{f \otimes f} & B \otimes B \\ {\scriptstyle \mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ A & \xrightarrow[f]{} & B \end{array} \qquad \begin{array}{ccc} & k & \\ {\scriptstyle \iota}\swarrow & & \searrow{\scriptstyle \iota} \\ A & \xrightarrow[f]{} & B \end{array}$$

One other way to present a finite k-algebra is by the quotient of $k[x_1, \ldots, x_m]$ by a zero-dimensional ideal; in this case the morphisms are presented by morphisms of k-algebras. We claim that these two ways are equivalent; i.e. that we can transform one presentation into the other primitive recursively. We will only work this out for the objects, leaving the morphisms to the reader.

First suppose that we are given A as a vector space together with maps $\iota\colon k \to A$ and $\mu\colon A \otimes A \to A$, and let $(t_1, \ldots, t_m)$ be the given basis of A. Then A is isomorphic

to $k[t_1, \ldots, t_m]/I$, where I is generated by $t_i t_j - \mu(t_i \otimes t_j)$ and $1 - \iota(1)$. Note that this is done in a number of field operations polynomially bounded in $\dim_k A$.

Conversely, assume that we are given a zero-dimensional ideal $I \subseteq k[x_1, \ldots, x_m]$ such that $A = k[x_1, \ldots, x_m]/I$. Then we can compute from a Gröbner basis of I a k-basis for A consisting of monomials, and we can compute the multiplication and inclusion maps with respect to this basis using division with remainder with respect to the Gröbner basis. Note that this involves computing Gröbner basis of zero-dimensional ideals, which can theoretically be done in a number of field operations exponentially bounded in the number of given generators of the ideal, see Dickenstein et al. [9].

We now list a number of properties that can be decided algorithmically. We start with the property of being étale over k.

Proposition 1.16. *There exists an algorithm that takes as input an explicitly given field k and a finite k-algebra A, and decides whether A is étale over k, in a number of field operations polynomially bounded in* $\dim_k A$.

Proof. Note that A is étale over k if and only if the *trace form* $A \to \mathrm{Hom}_k(A, k)$ given by $a \mapsto \big(b \mapsto \mathrm{Tr}(ab)\big)$ is invertible. Since we can compute the trace form and the determinant thereof in a number of field operations bounded polynomially in $\dim_k A$, we get the desired result. □

To decide whether a finite k-algebra A is local, we use the following result.

Proposition 1.17 (Khuri-Makdisi [25, Sect. 7])**.** *There exists an algorithm that takes as input a factorial field k and a finite k-algebra A, and returns an isomorphism $\prod_i A_i \cong A$ with all A_i finite local k-algebras, in a number of field operations polynomially bounded in* $\dim_k A$.

Corollary 1.18. *There exists an algorithm that takes as input a factorial field k and a finite k-algebra A, and decides whether A is local, in a number of field operations polynomially bounded in* $\dim_k A$.

Since for a finite k-algebra, being étale and local is equivalent to being a finite separable field extension of k, we also get the following.

Corollary 1.19. *There exists an algorithm that takes as input a factorial field k and a finite k-algebra A, and decides whether A is a finite separable field extension of k, in a number of field operations polynomially bounded in* $\dim_k A$.

Finally, we can decide whether a finite k-algebra is a finite Galois (field) extension of k.

Proposition 1.20. *There exists an algorithm that takes as input a factorial field k and a finite separable field extension l over k, and outputs the Galois closure of l over k, in a number of field operations exponentially bounded in* $\dim_k l$.

Proof. Decompose $l \otimes l = \prod_i l_i$. Then note that each l_i is a separable field extension of l, and that l is Galois if and only if every l_i is equal to l. Therefore replacing l iteratively by an l_i with maximal dimension (and using that the Galois closure of l over k has degree at most $[l : k]!$ over k) computes the Galois closure of l over k in a number of field operations exponentially bounded in $\dim_k l$. □

Corollary 1.21. *There exists an algorithm that takes as input a factorial field k and a finite k-algebra A, and decides whether A is a finite Galois (field) extension of k, in a number of field operations polynomially bounded in* $\dim_k A$.

Next, we compute the Galois group of a finite Galois extension of k.

Proposition 1.22. *There exists an algorithm that takes as input a factorial field k and a finite Galois extension l of k, and outputs* $\mathrm{Gal}(l/k)$.

Proof. We note that $\mathrm{Gal}(l/k)$ is the set of k-rational points of a finite algebraic subgroup of $\mathrm{GL}_{\dim_k l,k}$, which we can compute using Gröbner bases. □

Finally, given a finite Galois extension l of k with Galois group G, we can make Galois theory effective: given a subgroup H of G, we can compute l^H (as the intersection of the kernels of the k-linear maps $1-h$ for $h \in H$), and vice versa, given a subextension l' of l over k, we can compute $\mathrm{Gal}(l/l')$.

1.5.4 Galois sets

The following treatment is essentially that of Couveignes and Edixhoven [5, p.69–70].

Let G be the absolute Galois group of k; recall that it is a profinite group. There are two natural ways of presenting a finite continuous G-set. For the first one, note that the category of finite continuous G-sets is equivalent to the opposite of that of finite separable k-algebras; so we present a finite continuous G-set by a finite separable k-algebra, and we present a morphism $Y \to X$ of finite continuous G-sets by a morphism of finite separable k-algebras (in the opposite direction).

Alternatively, note that a finite continuous G-set is given by a finite set X, together with a continuous group morphism $G \to S(X)$, where $S(X)$ is the permutation group on X. Its kernel N is a closed subgroup of finite index, which corresponds to a finite Galois extension l over k, and the Galois set X is determined by the action of $\mathrm{Gal}(l/k)$ on X. This shows that we can present a finite continuous G-set by a tuple (l, X, α) of a finite Galois extension l over k, a finite set X, and an action α of $\mathrm{Gal}(l/k)$ on X.

We can extend the above to any finite number of finite continuous G-sets, to see that we can present a finite number of finite continuous G-sets by a tuple $\left(l, (X_i, \alpha_i)_i\right)$, such that every (l, X_i, α_i) presents a finite continuous G-set. In particular, we see that we can present a morphism of finite continuous G-sets by a finite Galois extension l over k, finite sets X, Y, actions α_X, α_Y of $\mathrm{Gal}(l/k)$ on X, Y, respectively, and a $\mathrm{Gal}(l/k)$-equivariant map $f\colon Y \to X$.

Using Section 1.5.3, we see that these two presentations can be converted into one another in a straightforward way; if A is a finite separable k-algebra, and $A = \prod_i l_i$ is a decomposition of A into fields, then a corresponding triple is (l, X, α) where l is the Galois closure of the compositum of the l_i and the set X is $\coprod_i \mathrm{Hom}_k(l_i, l)$ together with the natural $\mathrm{Gal}(l/k)$-action on the $\mathrm{Hom}_k(l_i, l)$; conversely, if (l, X, α) is a presentation of a finite continuous G-set, then decompose X into $\mathrm{Gal}(l/k)$-orbits X_i, compute for each i a stabiliser G_i of a point of X_i (which is well-defined up to inner automorphisms), and set $A = \prod_i l^{G_i}$.

1.5.5 Multivariate and absolute factorisation

Recall that a factorial field is an explicitly given field together with an algorithm for *univariate* polynomial factorisation. However, using a trick attributed to Kronecker in van der Waerden [37, Sec. 42], one can easily reduce multivariate polynomial factorisation to univariate polynomial factorisation; this uses an number of field operations exponential in the degree of the polynomial to be factored.

Next, we consider *absolute factorisation*, i.e. given a polynomial $f \in k[x_1, \ldots, x_m]$, find the factorisation of f over the algebraic closure of k. Note that this factorisation is defined over a finite extension l of k. By Chistov [4, Sec. 1.3], absolute factorisation can be reduced to ordinary multivariate polynomial factorisation in a number of field operations which is polynomial in the degree of the polynomial to be factored.

It follows that any factorial field admits an algorithm computing absolute factorisations of polynomials in $k[x_1, \ldots, x_m]$.

1.6 Curves over explicitly given fields

Let k be an explicitly given field. In this section we describe two ways to describe $\mathbb{P}^1_k$-vector bundles, one of which is more or less classical, essentially going back to Dedekind and Weber [6] (another reference is Diem [10]), and an alternative one better suited for our purposes. We can then describe curves together with a finite locally free morphism to $\mathbb{P}^1_k$ as vector bundles on $\mathbb{P}^1_k$ with an algebra structure.

1.6.1 Vector bundles via function fields

The following is a slight generalisation and alteration of the idea described in Diem [10, Sect. 2.5.4.2].

The basic idea here is to describe a vector bundle $\mathcal{E}$ on $\mathbb{P}^1_k$ by $(\mathcal{E}_\eta, \mathcal{E}(U_0), \mathcal{E}(U_1))$, where $\eta \in \mathbb{P}^1_k$ is the generic point, and U_0, U_1 are the standard affine open subsets of $\mathbb{P}^1_k$. Here we view the $\mathcal{E}(U_i)$ as subsets of $\mathcal{E}_\eta$. The rule attaching to $\mathcal{E}$ a triple as above is a functor, the target category of which we describe below. There, we will identify $\mathcal{O}_{\mathbb{P}^1_k,\eta}, \mathcal{O}_{\mathbb{P}^1_k}(U_0), \mathcal{O}_{\mathbb{P}^1_k}(U_1)$ with $k(x), k[x], k[x^{-1}]$, respectively.

Consider the category $\mathcal{L}(k)$ defined as follows. The objects of $\mathcal{L}(k)$ are tuples (V, V_0, V_1), where V is a finite dimensional vector space over $k(x)$, say of dimension m, and V_0 (resp. V_1) is a free $k[x]$-submodule (resp. $k[x^{-1}]$-submodule) of V of rank m, such that V_0 and V_1 generate the same $k[x, x^{-1}]$-submodule of V. The morphisms $(V, V_0, V_1) \to (W, W_0, W_1)$ in $\mathcal{L}(k)$ are the morphisms $V \to W$ that map V_i into W_i for $i \in \{0, 1\}$.

Note that the functor from the category of vector bundles on $\mathbb{P}^1$ to $\mathcal{L}(k)$ defined by

$$\mathcal{E} \mapsto (\mathcal{E}_\eta, \mathcal{E}(U_0), \mathcal{E}(U_1)).$$

is an equivalence of categories.

By expanding the definition of the objects and morphisms of $\mathcal{L}(k)$ in terms of matrices, we see that $\mathcal{L}(k)$ (hence also the category of vector bundles on $\mathbb{P}^1_k$) is equivalent to the category $\mathcal{P}'(k)$ (of presentations of finite locally free $\mathcal{O}_{\mathbb{P}^1_k}$-modules) defined below.

The objects of $\mathcal{P}'(k)$ are tuples (m, B_0, B_1), where m is a non-negative integer and $B_0, B_1 \colon k(x)^m \to k(x)^m$ are $k(x)$-linear isomorphisms, the columns of the matrices of which generate the same $k[x, x^{-1}]$-submodules of $k(x)^m$, i.e. the matrix of $B_0 B_1^{-1}$ has entries in $k[x, x^{-1}]$. (Matrices are always taken with respect to the standard bases.)

Now consider two objects (m, B_0, B_1) and (n, C_0, C_1) of $\mathcal{P}'_1(k)$. The morphisms in $\mathcal{P}'(k)$ from (m, B_0, B_1) to (n, C_0, C_1) are the $k(x)$-linear maps $f \colon k(x)^m \to k(x)^n$, such that the $k[x]$-submodule generated by the columns of the matrix of B_0 is mapped into that of C_0, and such that the $k[x^{-1}]$-submodule generated by the columns of the matrix of B_1 is mapped into that of C_1. In other words, we have that the matrices of $C_0^{-1} f B_0$, resp. $C_1^{-1} f B_1$ have entries in $k[x]$, resp. $k[x^{-1}]$.

Note that the tensor product $(m, B_0, B_1) \otimes (m', B_0', B_1')$ of two objects in $\mathcal{P}'(k)$ is given by $(mm', B_0 \otimes B_0', B_1 \otimes B_1')$, and that the tensor product of two morphisms $f \colon (m, B_0, B_1) \to (n, C_0, C_1)$ and $f' \colon (m', B_0', B_1') \to (n', C_0', C_1')$ is given by $f \otimes f'$ (as $k(x)$-linear map $k(x)^{mm'} \to k(x)^{nn'}$). Moreover, $\otimes$ is associative and the (identity morphism on) the object $(0, 0, 0)$ is neutral for $\otimes$.

1.6.2 Vector bundles via Dedekind-Weber splitting

There is an alternative way to present vector bundles on $\mathbb{P}^1_k$. The following theorem (commonly attributed to Grothendieck) describes all isomorphism classes of vector bundles on $\mathbb{P}^1_k$.

Theorem 1.23 (Dedekind and Weber [6])**.** *Let k be a field, and let $\mathcal{E}$ be a finite locally free $\mathcal{O}_{\mathbb{P}^1_k}$-module. Then there exists a (up to permutation unique) finite sequence of integers $(a_i)_{i=1}^s$ such that $\mathcal{E} \cong \bigoplus_{i=1}^s \mathcal{O}_{\mathbb{P}^1_k}(a_i)$.*

Write, for a finite sequence $a = (a_i)_{i=1}^s$ of integers, $\mathcal{O}_{\mathbb{P}^1_k}(a)$ for the $\mathcal{O}_{\mathbb{P}^1_k}$-module $\bigoplus_{i=1}^s \mathcal{O}_{\mathbb{P}^1_k}(a_i)$. We will use the "linear algebra" of such objects to describe the category of finite locally free $\mathcal{O}_{\mathbb{P}^1_k}$-algebras. Since for all finite sequences a, b of integers, we have

$$\begin{aligned}\operatorname{Hom}_{\mathcal{O}_{\mathbb{P}^1_k}}\big(\mathcal{O}_{\mathbb{P}^1_k}(a), \mathcal{O}_{\mathbb{P}^1_k}(b)\big) &= \bigoplus_{i,j} \operatorname{Hom}_{\mathcal{O}_{\mathbb{P}^1_k}}\big(\mathcal{O}_{\mathbb{P}^1_k}(a_i), \mathcal{O}_{\mathbb{P}^1_k}(b_j)\big) \\ &= \bigoplus_{i,j} \mathcal{O}_{\mathbb{P}^1_k}(b_j - a_i)(\mathbb{P}^1_k),\end{aligned}$$

we see that giving a morphism $\mathcal{O}_{\mathbb{P}^1_k}(a) \to \mathcal{O}_{\mathbb{P}^1_k}(b)$ is the same as giving an element of

$$\operatorname{Mat}_{b,a}(k) = \{M \in \operatorname{Mat}_{t \times s}\big(k[x, y]\big) : M_{ji} \in k[x, y]_{b_j - a_i}\},$$

where s and t are the respective lengths of the sequences a and b.

Therefore the category of vector bundles on $\mathbb{P}^1_k$ is equivalent to the category $\mathcal{P}(k)$ of which the objects are finite sequences of integers, and in which the set of morphisms from a finite sequence a to a finite sequence b is given by $\operatorname{Mat}_{b,a}(k)$ (with composition given by matrix multiplication).

Next, we describe tensor products in $\mathcal{P}(k)$. For two finite sequences a, a' of integers the sequence $a \oplus a'$ is the set $\{a_i + a'_{i'}\}_{i,i'}$ together with the order on the index

set given by the lexicographical order on pairs (i, i'), so that there is an isomorphism $\mathcal{O}_{\mathbb{P}^1_k}(a \oplus a') \cong \mathcal{O}_{\mathbb{P}^1_k}(a) \otimes \mathcal{O}_{\mathbb{P}^1_k}(a')$. Moreover, the tensor product of two morphisms can be computed by viewing morphisms as matrices with entries in $k(x, y)$.

1.6.3 Converting presentations

We now describe explicit quasi-inverse equivalences between $\mathcal{P}(k)$ and $\mathcal{P}'(k)$.

First we describe the functor $F\colon \mathcal{P}(k) \to \mathcal{P}'(k)$. For an object a of $\mathcal{P}(k)$, i.e. a finite sequence of integers, we set $F(a)$ to be the triple

$$\Big(\bigoplus_i \mathcal{O}_{\mathbb{P}^1_k}(a_i)_\eta, B_0, B_1\Big),$$

where we identify $\mathcal{O}_{\mathbb{P}^1_k}(a_i)_\eta$ with $k(x)$ by identifying y with 1, and where B_0 is the identity matrix, and where B_1 is the diagonal matrix of which the i-th entry is x^{a_i}. The given identification of $\mathcal{O}_{\mathbb{P}^1_k}(a_i)_\eta$ with $k(x)$ also immediately gives a description of F on morphisms.

Next, we describe its quasi-inverse $G\colon \mathcal{P}'(k) \to \mathcal{P}(k)$. Suppose that we have an object (m, B_0, B_1) of $\mathcal{P}'(k)$. Then the method of e.g. Görtz and Wedhorn [16, Lem. 11.50], see also Hess [22, Sec. 4], gives an algorithm to compute a basis $C = \{C_i\}$ of $k(x)^m$ over k such that C generates the same $k[x]$-submodule as B_0, and a sequence of integers a of length s such that $x^{a_i}C_i$ spans the same $k[x^{-1}]$-submodule of $k(x)^m$ as B_1. In this case, we set $G(m, B_1, B_2) = a$.

Next, if we have a morphism $\varphi\colon (m, B_0, B_1) \to (n, C_0, C_1)$ in $\mathcal{P}'(k)$, we consider their corresponding sequences of integers a, b, and the matrix M of the corresponding k-linear map $k(x)^m \to k(x)^n$ with respect to the $k(x)$-bases given above. By definition of a morphism, the entries of this matrix lie in $k[x]$, and in fact, the degree of the (j, i)-entry is at most $b_j - a_i$. Let M' be the matrix in $k[x, y]$ obtained from M by replacing each entry $M_{ji}(x)$ by $a_{ji}(x/y)y^{b_j - a_i}$. Then $G(\varphi)$ is given by M'.

By construction, the following is now clear.

Proposition 1.24. *The functors F and G defined above are quasi-inverse equivalences.*

1.7 Commutative algebra over explicitly given fields

In this section, we consider certain constructions in commutative algebra. We present k-algebras of finite type as in Section 1.5.2.

1.7.1 Localisations

For an element $f \in k[x_1, \dots, x_m]$ and an ideal I of $k[x_1, \dots, x_m]$, the localisation $\big(k[x_1, \dots, x_m]/I\big)_f$ of $k[x_1, \dots, x_m]$ is given by the morphism

$$k[x_1, \dots, x_m]/I \to k[x_1, \dots, x_m, x_{m+1}]/\big(I + (x_{m+1}f - 1)\big)$$

sending x_i to x_i (for $i = 1, 2, \dots, m$).

1.7.2 Equality of radicals of ideals

Given two ideals I, J of $k[x_1, \ldots, x_m]$, we can test algorithmically whether their radicals are equal using an effective Nullstellensatz, like the following theorem by Kollár.

Theorem 1.25 (Kollár [26, Cor. 1.7])**.** *Let k be a field, and let $f_1, \ldots, f_s \in k[x_1, \ldots, x_m]$ and let d be the maximum of 3 and their degrees. Then for all $h \in \sqrt{(f_1, \ldots, f_s)}$ there exist a positive integer $t \leq 2d^m$ and $g_1, \ldots, g_s \in k[x_1, \ldots, x_m]$ such that*

$$h^t = g_1 f_1 + \cdots + g_s f_s$$

with $\deg g_i f_i \leq (1 + \deg h) 2d^m$.

In fact, if we weaken the condition in the last line to $\deg g_i f_i \leq (1 + 2d^m \deg h) 2d^m$, then we can take $t = 2d^m$. Therefore checking whether a polynomial lies in the radical of some ideal of $k[x_1, \ldots, x_m]$ boils down to solving a large system of linear equations.

1.7.3 Tensor products

Let $A = k[x_1, \ldots, x_m]/I$, $B = k[y_1, \ldots, y_n]/J$, $C = k[z_1, \ldots, z_p]/K$. Let $\varphi\colon A \to B$ and $\psi\colon A \to C$ be morphisms of k-algebras. Then the tensor product $B \otimes_A C$ is given by

$$\frac{k[x_1, \ldots, x_m, y_1, \ldots, y_n, z_1, \ldots, z_p]}{I + J + K + \big(\varphi(x_1) - x_1, \ldots, \varphi(x_m) - x_m, \psi(x_1) - x_1, \ldots, \psi(x_m) - x_m\big)}$$

together with the obvious morphisms $B \to B \otimes_A C$ and $C \to B \otimes_A C$.

1.7.4 Other algorithms

We list some more algorithms we will make use of, namely those for *Noether normalisation* and *primary decomposition*.

Theorem 1.26 (Nagata [32])**.** *There exists an algorithm that takes as input an explicitly given field k and a k-algebra A of finite type, and outputs an injective integral morphism $k[x_1, \ldots, x_m] \to A$ in an effectively bounded number of field operations.*

Theorem 1.27 (Gianni et al. [15])**.** *There exists an algorithm that takes as input a factorial field k and an ideal $I \subseteq k[x_1, \ldots, x_m]$, and outputs a primary decomposition of I in an effectively bounded number of field operations.*

Remark 1.28. We remark that the algorithm by Gianni et al. [15] a priori is not primitive recursive because of the use of an unbounded search at two points, namely Proposition 3.7 and Proposition 8.2. Fortunately, in the case that we need, this can be amended, as explained below.

First, the unbounded search in Proposition 3.7 collapses, as we only need the case that $p = 0$. Moreover, we note that the crucial step in Proposition 8.2 in the case that we need, is the following: given ideals I, J of $k[x_1, \ldots, x_n]$ and $s \in k[x_1, \ldots, x_n]$, compute a positive integer m such that $s^m J \subseteq I$ if one exists. This can be done primitive recursively by first computing $I : J$ (using Gröbner bases) and then checking if $s \in \sqrt{I : J}$, using an effective Nullstellensatz like the one by Kollár [26] mentioned above.

Moreover, replacing every occurrence of a factorisation in their algorithm by an absolute factorisation will give an algorithm computing an *absolute* primary decomposition (i.e. a primary decomposition over $\bar{k}$) instead.

1.8 Schemes of finite type over a field

1.8.1 Affine schemes

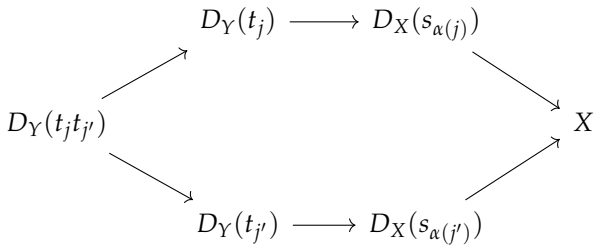
The category of affine schemes of finite type over a field k is just the opposite of the category of k-algebras of finite type, so we present an affine scheme X of finite type over k by its ring $\mathcal{O}(X)$ of global sections, and a morphism $Y \to X$ of affine schemes of finite type over k by the morphism $\mathcal{O}(X) \to \mathcal{O}(Y)$ of k-algebras. For an affine scheme X and $s \in \mathcal{O}(X)$, we will denote by $D_X(s)$ the standard open subscheme of X defined by s.

Note that we can compute fibre products of affine schemes.

1.8.2 Quasi-affine schemes

We present a quasi-affine scheme U of finite type over k by an affine scheme X of finite type over k, together with a finite sequence $s_1, \ldots, s_m \in \mathcal{O}(X)$ of elements generating an ideal defining the complement of U in X. Note that we can view an affine scheme X as a quasi-affine scheme presented by $(X, 1)$. We can test algorithmically whether two such presentations define the same open subscheme of a fixed scheme X, since this boils down to checking that two ideals have the same radical.

Suppose the quasi-affine schemes U and V are presented by tuples $(X, s_1, \ldots, s_m)$ and $(Y, t_1, \ldots, t_n)$, respectively. A morphism $V \to U$ with respect to the given presentations is then given by a map $\alpha\colon \{1, \ldots, n\} \to \{1, \ldots, m\}$ and morphisms $D_Y(t_j) \to D_X(s_{\alpha(j)})$ such that the following diagram commutes for all j and j' in $\{1, \ldots, n\}$.



Note that for any morphism $f\colon V \to U$ where $U \subseteq X$ and $V \subseteq Y$ are open subschemes with X and Y affine and of finite type over k, there exist presentations of U and V such that f can be given with respect to those presentations.

We want to be able to compute compositions of composable morphisms and test whether two morphisms are equal. To this end, we first explain how to compute the fibre product of two quasi-affine schemes.

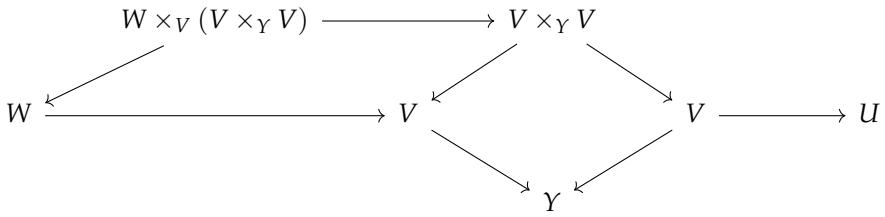
Suppose that the quasi-affine schemes U, V, W are given by tuples $(X, s_1, \ldots, s_m)$, $(Y, t_1, \ldots, t_n)$, $(Z, u_1, \ldots, u_p)$, respectively, and let $V \to U$ and $W \to U$ be morphisms with respect to the given presentations. Then by the classical construction of fibre

products of schemes, we see that $V \times_U W$ is given by

$$(Y \times_X Z, t_j u_k)$$

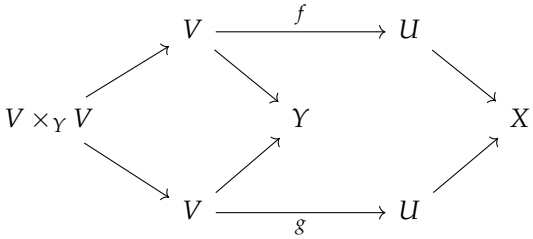
(with (j,k) running through all pairs such that the images of j and k in $\{1,\dots,m\}$ are the same). We have obvious projection morphisms $V \times_U W \to V$ and $V \times_U W \to W$ with respect to the given presentations.

Now let U, V, W be quasi-affine schemes, and let $f\colon V \to U$ and $g\colon W \to V$ be morphisms. Suppose that V is given as an open subscheme of the affine scheme Y, and view the presentations of V used for f and g as morphisms $V \to Y$ of quasi-affine schemes. We can then compute the composition fg using the following diagram, in which we do not simplify the expression $V \times_Y V$ as both factors are in general given by distinct presentations.

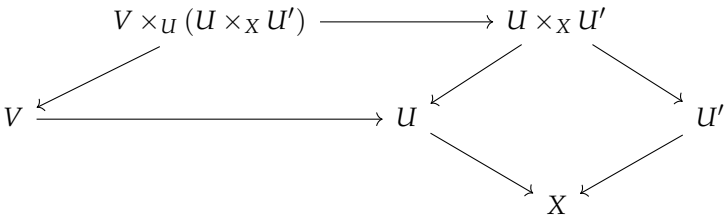


The composition fg then is the morphism $W \times_V (V \times_Y V) \to U$ in the diagram above.

In a similar vein, if U and V are quasi-affine schemes, given as open subschemes, of the affine schemes X and Y, respectively, and $f, g\colon V \to U$ are morphisms, then we can test whether $f = g$ since this is the case if and only if the following diagram commutes.



Finally, suppose that U, U' are both open subschemes of an affine scheme X, that V is an open subscheme of an affine scheme Y, and that $f\colon V \to U$ is a morphism of schemes. We can test whether the image of f is contained in U' by considering the diagram



and testing that $V \times_U (U \times_X U')$ (which is given as an open subscheme of the scheme $Y \times_X X = Y$) is the same as V. Moreover, we see that if the image of f is contained in U', then the diagram above gives a way to compute a presentation of the induced morphism $V \to U'$.

1.8.3 Presentations of schemes

A scheme of finite type over k is presented by gluing data:

- affine schemes $X_1, \dots, X_m$ of finite type over k;
- for all $i, j \in \{1, \dots, m\}$ an open subscheme $X_{ij} \subseteq X_i$ such that $X_{ii} = X_i$ for all $i \in \{1, \dots, m\}$;
- for all $i, j \in \{1, \dots, m\}$ a morphism $\varphi_{ji} \colon X_{ij} \to X_{ji}$ of quasi-affine schemes, such that φ_{ii} is the identity on X_i for all $i \in \{1, \dots, m\}$, and such that the *cocycle condition* holds, i.e.:
  for all $i, j, k \in \{1, \dots, m\}$, the image of $X_{ij} \times_{X_i} X_{ik}$ under φ_{ji} is contained in X_{jk}, and the diagram

$$\begin{array}{ccc} X_{ij} \times_{X_i} X_{ik} & \xrightarrow{\varphi_{ji}} & X_{ji} \times_{X_j} X_{jk} \\ & {\scriptstyle \varphi_{ki}} \searrow \quad \swarrow {\scriptstyle \varphi_{kj}} & \\ & X_{ki} \times_{X_k} X_{kj} & \end{array}$$

  is commutative.

We will denote such a presentation by the shorthand $(X_1, \dots, X_m)$.

Note that the cocycle condition for $i = k$ implies that $\varphi_{ji} = \varphi_{ij}^{-1}$ for all i, j in $\{1, \dots, m\}$ and that the induced morphism $\varphi_{ji} \colon X_{ij} \times_{X_i} X_{ik} \to X_{ji} \times_{X_j} X_{jk}$ is an isomorphism for all $i, j, k \in \{1, \dots, m\}$.

A morphism $(Y_1, \dots, Y_n) \to (X_1, \dots, X_m)$ of presentations of schemes of finite type over k is given by a map $\alpha \colon \{1, \dots, n\} \to \{1, \dots, m\}$, morphisms $f_j \colon Y_j \to X_{\alpha(j)}$, and morphisms $f_{jj'} \colon Y_{jj'} \to X_{\alpha(j)\alpha(j')}$ compatible with gluing data. More precisely, the following diagrams commute for all $j, j' \in \{1, \dots, n\}$.

$$\begin{array}{ccc} Y_{jj'} & \xrightarrow{f_{jj'}} & X_{\alpha(j)\alpha(j')} \\ \downarrow & & \downarrow \\ Y_j & \xrightarrow[f_j]{} & X_{\alpha(j)} \end{array} \qquad \begin{array}{ccc} Y_{jj'} & \xrightarrow{f_{jj'}} & X_{\alpha(j)\alpha(j')} \\ {\scriptstyle \varphi_{j'j}}\downarrow & & \downarrow{\scriptstyle \varphi_{\alpha(j')\alpha(j)}} \\ Y_{j'j} & \xrightarrow[f_{j'j}]{} & X_{\alpha(j')\alpha(j)} \end{array}$$

First note that we can algorithmically determine whether two morphisms of presentations of schemes define the same morphism between the schemes that they present, by the following.

Lemma 1.29. *Let $f, g \colon (Y_1, \dots, Y_n) \to (X_1, \dots, X_m)$ be morphisms between presentations of schemes (with α, β the corresponding maps on indices). Then they define the same morphism of schemes if and only if the following holds: for all $j \in \{1, \dots, n\}$, the projections*

$Y_j \times_{X_{\alpha(j)}} X_{\alpha(j)\beta(j)} \to Y_j$ and $Y_j \times_{X_{\beta(j)}} X_{\alpha(j)\beta(j)} \to Y_j$ are isomorphisms and the compositions $Y_j \to Y_j \times_{X_{\alpha(j)}} X_{\alpha(j)\beta(j)} \to X_{\alpha(j)\beta(j)}$ and $Y_j \to Y_j \times_{X_{\beta(j)}} X_{\alpha(j)\beta(j)} \to X_{\alpha(j)\beta(j)}$ of the respective inverses and the projections are equal.

Proof. In this proof, we will identify the $X_i, X_{ii'}, Y_j, Y_{jj'}$ with the open subschemes of the schemes they define.

First suppose that f, g define the same morphism of schemes. Then, for all j, the image of Y_j lies in $X_{\alpha(j)}$ and $X_{\beta(j)}$, and therefore in $X_{\alpha(j)\beta(j)}$. Hence the projections $Y_j \times_{X_{\alpha(j)}} X_{\alpha(j)\beta(j)} \to Y_j$ and $Y_j \times_{X_{\beta(j)}} X_{\alpha(j)\beta(j)} \to Y_j$ are isomorphisms. Moreover, the morphism $Y_j \to X_{\alpha(j)\beta(j)}$ induced by f_j now is the composition

$$Y_j \to Y_j \times_{X_{\alpha(j)}} X_{\alpha(j)\beta(j)} \to X_{\alpha(j)\beta(j)},$$

and the one induced by g_j is the composition

$$Y_j \to Y_j \times_{X_{\beta(j)}} X_{\alpha(j)\beta(j)} \to X_{\alpha(j)\beta(j)},$$

so these two must be equal.

Conversely, suppose that the projection morphisms $Y_j \times_{X_{\alpha(j)}} X_{\alpha(j)\beta(j)} \to Y_j$ and $Y_j \times_{X_{\beta(j)}} X_{\alpha(j)\beta(j)} \to Y_j$ are isomorphisms for all $j \in \{1, \dots, n\}$, and that the compositions

$$Y_j \to Y_j \times_{X_{\alpha(j)}} X_{\alpha(j)\beta(j)} \to X_{\alpha(j)\beta(j)}$$

and

$$Y_j \to Y_j \times_{X_{\beta(j)}} X_{\alpha(j)\beta(j)} \to X_{\alpha(j)\beta(j)}$$

of the respective inverses and the projections are equal. The first condition implies that the image of every Y_j under both f and g lies in $X_{\alpha(j)\beta(j)}$, and the second condition then implies that the resulting morphisms $Y_j \to X_{\alpha(j)\beta(j)}$ are equal. Since the Y_j cover Y, it follows that f and g define the same morphism of schemes. □

Note that we can also compute algorithmically compositions of morphisms of presentations of schemes. Moreover, if $(X_1, \dots, X_m)$, $(Y_1, \dots, Y_n)$, $(Z_1, \dots, Z_p)$ are presentations of schemes X, Y, Z, respectively, and $(Y_1, \dots, Y_n) \to (X_1, \dots, X_m)$ and $(Z_1, \dots, Z_p) \to (X_1, \dots, X_m)$ are morphisms of presentations, then the construction of fibre products of schemes gives a way to compute a presentation of the fibre product $Y \times_X Z$.

Next, we describe presentations of open subschemes of schemes. Let X be a scheme, presented as $(X_1, \dots, X_m)$. Then an open subscheme U of X is presented by a tuple $(U_1, \dots, U_n)$ together with a map $\alpha\colon \{1, \dots, n\} \to \{1, \dots, m\}$ and standard open subschemes $U_j \to X_{\alpha(j)}$ for all $j \in \{1, \dots, n\}$.

The corresponding description of U as a scheme is then given by the additional data of the intersections, which can be computed as the open subscheme

$$U_{jj'} = (U_j \times_{X_{\alpha(j)}} X_{\alpha(j)\alpha(j')}) \times_{U_j} \varphi_{j'j}^{-1}(U_{j'} \times_{X_{\alpha(j')}} X_{\alpha(j')\alpha(j)})$$

of U_j, together with the isomorphisms $U_{jj'} \to U_{j'j}$ induced by $\varphi_{j'j}^{-1}$. We then have an obvious morphism $U \to X$ given with respect to the given presentations.

Note that we can test algorithmically whether $U = X$ by testing that for all i in $\{1, \ldots, m\}$, the union in the affine scheme X_i over all $j \in \{1, \ldots, n\}$ of the open subschemes $\varphi^{-1}_{\alpha(j)i}(U_j \times_{X_{\alpha(j)}} X_{\alpha(j)i})$ is X_i itself.

Let us call a presentation of such an open subscheme $U = X$ of X a *refinement*; these present the identity morphism on X. Now note that with the same methods as in the case of quasi-affine schemes, if two morphisms $Z \to Y$ and $Y \to X$ are given, together with a chain of refinements connecting the target of the former to the source of the latter, one can compute the composition $Z \to X$. Moreover, again with the same methods as in the case of quasi-affine schemes, if two morphisms $Y \to X$ are given, together with a chain of refinements connecting the targets, and one connecting the sources, one can test whether these two morphisms are equal.

1.8.4 Finite étale morphisms of schemes

Let $(X_1, \ldots, X_m)$ be a presentation of a scheme X. Then we present a finite étale X-scheme (or equivalently, a finite locally constant sheaf on $X_{\text{ét}}$) by a descent datum w.r.t. the open cover $\{X_i \to X\}$; more precisely, by the following data:

- for all $i \in \{1, \ldots, m\}$ a finite étale morphism $Y_i \to X_i$;
- for all $i, j \in \{1, \ldots, m\}$ a morphism $\psi_{ji} \colon Y_i \times_{X_i} X_{ij} \to Y_j \times_{X_j} X_{ji}$ lying over the morphism $\varphi_{ji} \colon X_{ij} \to X_{ji}$,

such that ψ_{ii} is the identity on Y_i for all i and such that the *cocycle condition* holds, i.e. for all $i, j, k \in \{1, \ldots, m\}$ the following diagram commutes

$$\begin{array}{ccc} Y_i \times_{X_i} X_{ij} \times_{X_i} X_{ik} & \xrightarrow{\psi_{ji}} & Y_j \times_{X_j} X_{ji} \times_{X_j} X_{jk} \\ & {\scriptstyle \psi_{ki}} \searrow \quad \swarrow {\scriptstyle \psi_{kj}} & \\ & Y_k \times_{X_k} X_{ki} \times_{X_k} X_{kj} & \end{array}$$

Note that a presentation as above in particular defines a (presentation) of a scheme Y, and a morphism $Y \to X$ with respect to the given presentations. A morphism $Z \to Y$ of finite étale X-schemes is therefore simply a commutative triangle

$$\begin{array}{ccc} Z & \longrightarrow & Y \\ & \searrow \quad \swarrow & \\ & X & \end{array}$$