CATEGORICAL COMPLEXITY

SAUGATA BASU AND M. UMUT ISIK

Abstract. We introduce a notion of complexity of diagrams (and in particular of objects and morphisms) in an arbitrary category, as well as a notion of complexity of functors between categories equipped with complexity functions. We discuss several examples of this new definition in categories of wide common interest, such as finite sets, Boolean functions, topological spaces, vector spaces, graded algebras, schemes and modules. We show that on one hand categorical complexity recovers in several settings classical notions of non-uniform computational complexity (such as circuit complexity), while on the other hand it has features which makes it mathematically more natural. We also postulate that studying functor complexity is the categorical analog of classical questions in complexity theory about separating different complexity classes.

1. Introduction

It is usual to associate some measure of complexity to mathematical objects. For example, the complexity of a polynomial is often measured by its degree, or alternatively by the volume of its Newton polytope, or the number the monomials appearing in it with non-zero coefficients, or the least number of operations needed for an algorithm to evaluate the polynomial at a given point. Once a notion of complexity is fixed, one can make quantitative statements about properties of the objects in terms of its complexity. In the case of polynomials for example, there are many results on upper bounds on the topological invariants of the variety that the polynomial defines, the number of steps needed to desingularize the variety, and many other functions defined on the space of polynomials, in terms of the chosen complexity measure.

The notion of complexity also arose in theoretical computer science as a means of studying efficiency of algorithms and also to measure the intrinsic hardness of certain algorithmic problems. The latter led to the development of structural complexity theory and in particular to the famous P versus NP questions for discrete complexity classes that remains unresolved until today. Even though these arose first in the context of decision problems and Boolean functions, there have been subsequent attempts to generalize the scope of computational complexity to other classes of mathematical objects – for example, the Blum-Shub-Smale theory for computations over reals as well as complex numbers [BCSS98], over more general structures [Poi95], for polynomials [Val79a, Val79b, vzG87], for constructible sheaves and functions [Bas15]. Some of these generalizations are motivated by costs of computations in certain models of computations, while others by the desire to have a sound internal notion of complexity for mathematical objects. Remarkably, there exists analogues of the P versus NP question in all the generalizations mentioned above. Thus, it seems that there should be a more fundamental way of looking at questions arising in computational complexity questions which unifies these various viewpoints.

1991 *Mathematics Subject Classification.* Primary 14P10, 14P25; Secondary 68W30.

The first author was supported in part by NSF grants CCF-0915954, CCF-1319080 and DMS-1161629 while working on this paper.

The goal of the current paper is to develop this general theory of complexity via a categorical approach that reconciles the intuitive notion of complexity of mathematical objects with the different notions of computational and circuit complexities used in theoretical computer science.

We start by defining a categorical notion called a *diagram computation*. In an arbitrary category $\mathcal{C}$ with a chosen set of morphisms called *basic morphisms*, diagram computations can be used to construct diagrams, and in particular objects and morphisms in $\mathcal{C}$ as follows. At the first level, one starts with a diagram consisting entirely of basic morphisms, and then successively adds limits and colimits of arbitrary full sub-diagrams, along with the accompanying morphisms from/to those subdiagrams, to construct more and more complex diagrams. Any diagram isomorphic to a subdiagram of the final resulting diagram is said to be computed by the diagram computation. This allows us to associate, to each object, arrow, or diagram in $\mathcal{C}$, a complexity by counting the number of limits, colimits and basic morphisms used its most efficient computation. Diagram computations come in three kinds, the full version described above, called a *mixed computation*, and two more restricted ones where one is either allowed to use only the so-called constructive limits, or only constructive colimits; leading to the notions of *mixed complexity*, *limit complexity*, and *colimit complexity* of objects, arrows or any diagrams in $\mathcal{C}$.

We remark here that connections between computability and logic on one hand and category theory and topos theory on the other hand has a long history (see for example, [LS88, MLM94]). However, our goal is different, and it is to develop a completely general notion of complexity, based on category theory, that is useful in studying basic objects in algebra and geometry from a quantitative point of view. To the best of our knowledge this task has not been undertaken before.

While our notion of complexity bears some similarity with a more classical view of complexity coming from logic, namely descriptive complexity [Imm95], there is one important respect in which our notion of complexity differs significantly from all classical notions. In our categorical world, isomorphic objects should have identical complexity – which is indeed the case with our definition. Thus, we are able to define a good notion of complexity in the category, say, of affine or projective schemes which is independent of embeddings. This is very natural from the mathematical point of view – making our theory completely geometric in those settings – but is sometimes at odd with ordinary complexity theory which deals with embedded objects (like subsets of the Boolean hypercube or subvarieties of $\mathbb{C}^n$). Nevertheless, we will show that, with the appropriate choice of category and basic morphisms, even non-categorical notions of complexity can be meaningfully embedded in categorical ones.

A fundamentally new point of view emerges when one thinks about the categorical analogues of classical complexity questions. For every functor between two categories for which complexity is defined, one can define a natural notion of complexity of the functor. Unlike, the complexity of diagrams, which are numbers, the complexity of a functor is a function $f : \mathbb{N} \to \mathbb{N}$, and one can ask whether such a functor is polynomially bounded. In this way classical questions about separation of complexity classes become, in the categorical world, questions about polynomiality of certain natural functors. With this shift of viewpoint, one can pose many questions about complexities of functors which have no direct analogues in the world of computational complexity. Well-studied properties of functors, such as preservation of limits and colimits, adjointness, etc. are important from this point of view.

The importance of functor complexity was already suggested in [Bas15], where the complexities of adjoint pairs of functors between the categories of semi-algebraically constructible sheaves on finite dimensional real affine spaces were posited as categorical analogues of the classical P versus NP question.

We now give a brief summary of our results. After the basic definitions in Section 2, we look at several basic examples. For sets, the colimit complexity of a set S is $|S|+1$. Infinite sets are "non-computable" in this theory. For topological spaces, colimit computations starting from simplices and face maps define a simplicial complexity for topological spaces. Similarly, mixed computations starting from points and intervals give rise to cubical complexity. These measure how hard it is to make a space from simplices and cubes respectively. Another important basic example one is where we recover monotone Boolean complexity from the categorical complexity in the lattice of subsets of a finite set.

In order to relate the new notion of categorical complexity, with pre-existing notions of (non-uniform) complexity, such as circuit complexity, or lengths of straight-line programs (we refer the reader to the books [BCS97, Bür00] for these notions), we prove certain comparison theorems. The first set of such theorems are about affine varieties, affine schemes, and algebras over a field. We show that the affine zero-set of a polynomial with low circuit complexity has low limit complexity (Theorem 3.2); on the other hand, if X is a variety with low limit complexity, then it is *isomorphic* to the zero-set of a polynomial map with low arithmetic complexity (Theorem 3.4). The same results hold for affine schemes and algebras. For projective schemes in $\mathbb{P}^n$: by building affine pieces with limits and then glueing them using colimits, we show that the mixed complexity of a projective scheme is bounded above by a constant multiple of n^2N, where N is the arithmetic complexity of its defining equations.

The categorical complexites of isomorphic varieties are equal by definition, while circuit complexity, being a non-geometric attribute, does not share this property. In Section 4, we consider two additional categories where circuit complexity of polynomials is, in a sense, embedded into categorical complexity. The first of these is the category of pairs of graded algebras, constructed specifically to make this embedding possible. Still, it remains to be seen how complexity in this category compares to arithmetic complexity of polynomials, or to the complexity of projective varieties discussed in [Isi16]. The second category considered here is the category of modules over polynomial rings, where we prove that the colimit complexity of the morphism diagram $k[x_1,\ldots,x_n] \xrightarrow{1\mapsto f} k[x_1,\ldots,x_n]$ and the arithmetic complexity of f are the same up to a quasi-polynomial function.

Finally, in Section 5, we discuss the behaviour of complexity under functors. Limit and colimit computations are preserved under right and left adjoints respectively. We define the complexity of a functor $F:\mathcal{C}\to\mathcal{D}$ as a function $c(F)(n)$ of n, where $c(F)(n)$ is the supremum of the complexity of $F(D)$, where D runs over all diagrams in $\mathcal{C}$ whose complexity is less than or equal to n. The question of whether the complexity of the image functor on the morphism category $\mathcal{C}^{\bullet\to\bullet}$ is polynomially bounded is the categorical analogue of the P vs NP problem for the category $\mathcal{C}$. Our final result is an analysis of the complexity of the image functor for the category of modules over polynomial rings.

2. Definitions and First Examples

By a directed graph, we mean a pair (V,E) with two maps $s:E\to V$ and $t:E\to V$. Let $\mathcal{C}$ be a category and let $U(\mathcal{C})$ be the underlying directed graph. Let I be any directed graph. By a diagram in $\mathcal{C}$, we mean a directed graph homomorphism $D:I\to U(\mathcal{C})$. The graph I is called the shape of D. Note that I is just a directed graph and does not have a composition

operation on it. As such, there is no a priori assumption of functoriality/commutativity for diagrams.

For $I = (V, E)$, and two diagrams, $D_1 : I \to U(\mathcal{C})$, $D_2 : I \to U(\mathcal{C})$, a morphism between D_1 and D_2 is a collection of morphisms $\varphi = (\varphi_v : D_1(v) \to D_2(v))_{v \in V}$, such that, for all $e \in E$, $D_2(e) \circ \varphi_{D_1(s(e))} = \varphi_{D_1(t(e))} \circ D_1(e)$. This defines the category of diagrams $\mathcal{C}^I$ with shape I.

By a *subdiagram* of a diagram $D : I \to U(\mathcal{C})$, with $I = (V, E)$, we mean the restriction $D_J : J \to U(\mathcal{C})$, with $J = (V', E')$ a full sub-graph of I, i.e. $V' \subset V$, and $E' = \{e \in E \,|\, s(e) \in V', t(e) \in V'\}$. The restrictions to not necessarily full subgraphs will be specified as *not necessarily full subdiagrams*.

Let $\mathcal{A}$ be a set of morphisms in $\mathcal{C}$. These will be called the *basic morphisms* in $\mathcal{C}$. We define a notion of computation in $\mathcal{C}$, called a *limit computation* by starting with these basic morphisms and adding a finite limit at each step; similarly, in a *colimit computation*, we build objects by adding colimits of subdiagrams.

Definition 2.1. Let, $\mathcal{C}$ be a category, $\mathcal{A} \subset \mathrm{Mor}(\mathcal{C})$. A *limit computation* (respectively, a *colimit computation*) in $\mathcal{C}$ is a finite sequence of diagrams $(D_0, \ldots, D_s)$, with $D_i : I_i \to U(\mathcal{C})$, where:

- (i) D_0 consists only of morphisms in $\mathcal{A}$, the basic morphisms.
- (ii) For each $i = 1, \ldots, s$, D_i is obtained from D_{i-1} by adding a limit or colimit cone of a subdiagram. More precisely, there is a sub-diagram $D_{i-1}|_{J_i}$ of D_{i-1} and an object L_i which is a limit (resp C_i which is a colimit) of $D_{i-1}|_{J_i}$ such that the difference between D_i and D_{i-1} are L_i and the limit cone morphisms out of L_i (resp. C_i and the colimit cocone morphisms into C_i).
- (iii) (*Constructivity*) If a limit $L_i = \lim D_{i-1}|_{J_i}$ (resp., colimit C_i) produced in the ith step of the computation is used again in the subdiagram $D_{j-1}|_{J_j}$ used at the jth step of the computation, then $J_i \subset J_j$, i.e. the subdiagram that produced L_i (resp., C_i) must be a sub-diagram of $D_{i-1}|_{J_j}$.

The computation $(D_0, \ldots, D_s)$ is said to compute a diagram D, if D is isomorphic to (a not necessarily full) subdiagram of D_s. In particular, an object in $\mathcal{C}$ is computed by $(D_0, \ldots, D_s)$ if an object isomorphic to it appears in D_s.

Remark 2.2. One could take Parts (i) and (ii) as the definition of limit (respectively, colimit) computation. However, in order that our notion of categorical complexity is closer to the classical notions – such as circuit complexity in certain relevant categories (see Section 3), we also consider the constructivity condition; which roughly means that the limit (or colimit) computation does not forget how an object was constructed. It also prevents objects of exponential rank/size from being constructed; c.f. Example 2.14.

Of course, one may not be able to obtain every object/morphism/diagram from a given set of basic morphisms $\mathcal{A}$ in a category $\mathcal{C}$. We will think of such objects/morphisms/diagrams as non-computable in $\mathcal{C}$ with respect to $\mathcal{A}$.

We now describe a basic syntax for writing down the limit or colimit computations. The computation is described by a set expressions, each expression in a line. The first kind of expression is of the form

`i. source,`f`,target`

and describes objects and/or morphisms that are added to D_0. Here, `i` is an identifier that can be any string. In subsequent lines, the identifier '`i`' is used to refer to the source, and '`i'`' is used to refer to the target of the basic morphism $f \in \mathcal{A}$ that is added to D_0 by this

expression. `source` and `target` are the identifiers of the vertices which are the intended source and target of the new morphism being attached to D_0. If the source is a new vertex that didn't exist in the diagram before, then we write `i.i,`f`,target`, or `i._,`f`,target` for it. If only the target is new, we write `i.source,`f`,i'` or `i.source,`f`,_`; and we write `i.i,`f`,i'` or `i._,`f`,_` if both are new, distinct vertices.

There is no need to list all the morhisms in D_0 at the beginning, so we will have these steps as intermediate steps as well; as long as the morphisms attach only to other vertices in D_0, they be can be considered as part of D_0.

The second kind of expression are those of the form:

```
i.  lim(a,b,...)
```

which describe steps where a limit is added to the subdiagram. The identifiers `a,b,...` describe the vertices in the full sub-diagram whose limit is being taken. In subsequent steps, `i` is used to refer to the limit added. Similarly, we write `colim(a,b,...)` for describing colimit computations. We may use the notation `i->a` to refer to morphisms created during the computation.

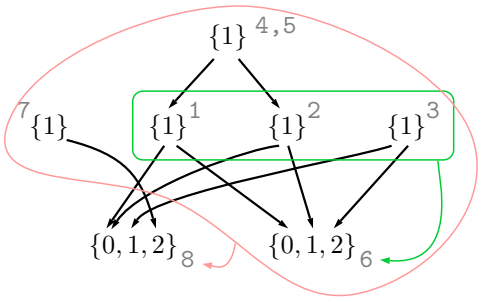
We start with two basic examples about constructions in the category of sets.

Example 2.3. Let $\mathcal{C}$ be the category of sets and let $\mathcal{A}$ consist of a single morphism id : $\{1\} \to \{1\}$. Consider the colimit computation described by

`1.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,1`
`2.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,2`
...
`n.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,n`
`n+1.` `colim(1,2,...,n)`

For each $k \leq n$, the step `k. _,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,k` is adding a new copy of $\{1\}$ to the diagram. In the end, `n+1` is the set with n elements.

Example 2.4. Continuing with the previous example, we now make a colimit computation that produces the morphism $\{0,1,2\} \xrightarrow{f} \{0,1,2\}$ in the category of sets, where $f(0) = 0$, $f(1) = 0$, $f(2) = 1$.



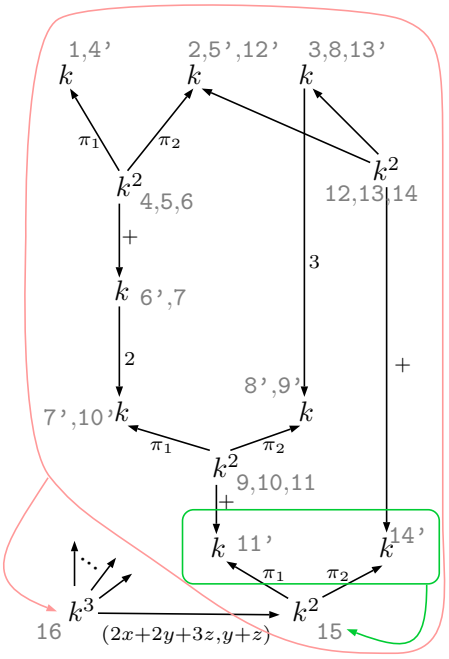
`1.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,1`
`2.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,2`
`3.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,3`
`4.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,1`
`5.` `4,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,2`
`6.` `colim(1,2,3)`
`7.` `_,`$\{1\} \xrightarrow{\text{id}} \{1\}$`,7`
`8.` `colim(1,2,3,4,6,7)`

The morphism `6->8` is f, in the sense that the full-subdiagram containing `6` and `8` is isomorphic to $\{0,1,2\} \xrightarrow{f} \{0,1,2\}$.

We will come back to sets later. We now discuss a more detailed example where we annotated each step in the computation.

Example 2.5. Consider the category k-Vect of vector spaces over a field k. Let $\mathcal{A}$ consist of the scalar multiplication morphisms $k \xrightarrow{c} k$ for each $c \in k$, the addition morphism $k^2 \xrightarrow{+} k$, the two projections $\pi_1, \pi_2 : k^2 \to k$, and morphisms $0 \to k$, $k \to 0$. Say, the characteristic of k is 0 and we wish to compute the morphism $f : k^3 \to k$, $f(x, y, z) = 2x + 2y + 3z$.

We describe the computation as follows.



1. _,$k \xrightarrow{1} k$,1
2. _,$k \xrightarrow{1} k$,2
3. _,$k \xrightarrow{1} k$,3
4. _,$k \times k \xrightarrow{\pi_1} k$,1
5. 4,$k \times k \xrightarrow{\pi_2} k$,2
6. 4,$k \times k \xrightarrow{+} k$,6'
7. 6',$k \xrightarrow{2} k$,7'
8. 3,$k \xrightarrow{3} k$,8'
9. _,$k \times k \xrightarrow{\pi_2} k$,8'
10. 9,$k \times k \xrightarrow{\pi_1} k$,7'
11. 9,$k \times k \xrightarrow{+} k$,11'
12. _,$k \times k \xrightarrow{\pi_1} k$,2
13. 12,$k \times k \xrightarrow{\pi_2} k$,3
14. 12,$k \times k \xrightarrow{+} k$,14'
15. lim(11',14')
16. lim(1,1',2,2',...
    ...,14,14',15)

Remark 2.6. Two facts about limits and colimits are useful in thinking about the above example and other computations. The first is that if $f : X \to Y$ and $g : Y \to Z$ are morphisms, then the limit of the diagram $X \xrightarrow{f} Y \xrightarrow{g} Z$ is (isomorphic to) X, with the induced morphism is (isomorphic, as a diagram, to) $X \to Z$ being the composition. So, compositions are obtained using limits. The second is that if we take the limit L of a diagram $D : I \to \mathcal{C}$, and X is a cone over D, then, the limit of these two diagrams joined together produces an object isomorphic to X. The map $X \to L$ is then the map that would normally come from the universal property of the limit L. So, to get a morphism that would come from the universal property of limits, we just need to take one additional limit.

2.1. **Mixed Limit-Colimit Computations.** We now discuss computations where we can use limits and colimits together, we call these *mixed computations*.

Definition 2.7. A *mixed computation* is a finite sequence $(D_0, \ldots, D_s)$ of diagrams with $D_i : I_i \to U(\mathcal{C})$, where D_0 consists only of morphisms in $\mathcal{A}$, and for each $i = 1, \ldots, s$, D_i is obtained from D_{i-1} by adding either the limit of a subdiagram, with the corresponding cone morphisms or colimit of a subdiagram with the corresponding cocone morphisms.

Note that there is no constructivity assumption for mixed computations. To include it would have been too restrictive and would have prevented natural applications like glueing geometric objects already constructed.

Example 2.8 (Monotone Boolean Circuits)**.** Let $\mathcal{B}_n$ be the lattice of subsets of $\{0,1\}^n$, that is a category whose objects are the subsets of $\{0,1\}^n$, and with $\mathrm{Hom}_{\mathcal{B}_n}(A, B) = \{\iota\}$

if $A \subset B$ where $\iota : A \to B$ is the inclusion, and $\mathrm{Hom}_{\mathcal{B}_n}(A, B) = \emptyset$ otherwise. Let $Z_i = \{(x_1, \ldots, x_n) \in \{0,1\}^n \mid x_i = 1\}$. Let $\mathcal{A}_n$ be the set of basic morphisms $\{\mathrm{id}_{Z_i} \mid i = 1, \ldots, n\}$. Let $\mathcal{B} = \coprod_{n=1}^{\infty} \mathcal{B}_n$ be the disjoint union of these categories and $\mathcal{A} = \coprod_{n=1}^{\infty} \mathcal{A}_n$.

We show that there is a correspondence between multi-output monotone Boolean circuits with n inputs and mixed computations in $\mathcal{B}_n$. Given a monotone Boolean circuit, consider the corresponding straight line program of with Boolean operations. Start a mixed computation in $\mathcal{B}_n$ with a copy each of the subsets Z_i. These correspond to the input variables $z_1, \ldots, z_n$ of the straight line program. Subsequent entries $z_{n+1}, z_{n+2} \ldots$ will correspond to newly constructed objects in the mixed computation. For each operation in the straight line program of the form $z_i = z_j \wedge z_k$, take the limit of the objects corresponding to z_j and z_k; similarly, take the colimit for $z_i = z_j \vee z_k$. To make a straight-line program from a mixed computation, start with the input variables $z_1, \ldots, z_n$ and add $k-1$ new $\wedge$ operations for each limit of k objects and a $k-1$ new $\vee$ operations for each colimit of k objects (ignoring the arrows does not change the limit/colimit).

Thus, mixed computations in $\mathcal{B}_n$ are in direct correspondence with monotone straight-line programs or, equivalently, monotone Boolean circuits.

Example 2.9. Consider the category **Top** of topological spaces. Let $I = [0,1]$ be the unit interval, and let the basic morphisms consist of, $I \xrightarrow{\mathrm{id}} I$,$I \to pt$, $pt \xrightarrow{0} I$, and $pt \xrightarrow{1} I$. We can build all cubes using limits, and then can glue these using colimits to construct many topological spaces.

We will reconsider mixed computations when we look at the complexity of projective schemes.

2.2. **Cost and Complexity.** Let $c_0 : \mathcal{A} \to \mathbb{Z}_{\geq 0}$ be any function, considered as the *cost* of the basic morphisms.

Definition 2.10. The *cost* of the computation $(D_0, \ldots, D_s)$ is the the number of steps plus the cost of the initial diagram D_0 consisting of basic morphisms, that is

$$c(D_0, \ldots, D_s) = s + \sum_{f \in \mathrm{edges}(I_0)} c_0(D_0(f)).$$

If c_0 is not specified, then we consider it to be the constant function 1, so every basic morphism will have unit cost. This will be the case in almost every example we consider.

Definition 2.11. The *limit* (resp. *colimit*, resp. *mixed*) complexity $C(D) = C_{\mathcal{C},\mathcal{A}}(D)$, short for $C^{\lim}_{\mathcal{C},\mathcal{A},c}(D)$ (resp., $C^{\mathrm{colim}}_{\mathcal{C},\mathcal{A},c}(D)$, resp., $C^{\mathrm{mixed}}_{\mathcal{C},\mathcal{A},c}(D)$), of a diagram D in a category $\mathcal{C}$ is the cost of the limit (resp., colimit, resp. mixed) computation using basic morphisms $\mathcal{A}$, that has the smallest cost among all such computations that compute D. For a morphism $f : X \to Y$ in $\mathcal{C}$, the complexity $C(f)$ of f is the complexity of the corresponding diagram mapping two objects and a single morphism $X \xrightarrow{f} Y$. For an object X in $\mathcal{C}$, the complexity $C(X)$ of X is the complexity of the diagram with one object, X.

Example 2.12 (Glueing Simplices)**.** Let $\mathcal{C} = \mathbf{Top}$ be the category of topological spaces and let $\mathcal{A}$ be the set of of all face embeddings $\Delta_n \to \Delta_m$, including the identity maps, where Δ_n is the topological n-simplex. Colimit computations correspond to glueing operations between simplices. The colimit complexity then measures how many simplices are needed to construct a given topological space by glueing.

We now go back to considering $\mathcal{C} = \mathbf{Set}$ with the basic morphisms $\mathcal{A}$ consisting of a single morphism $\mathrm{id} : \{1\} \to \{1\}$.

Proposition 2.13 (Colimit complexity of sets)**.** *In the category of sets, let* $\mathcal{A} = \{\mathrm{id} : \{1\} \to \{1\}\}$, $c(\mathrm{id}) = 1$. *Then the colimit complexity of a finite set* S *is* $|S| + 1$.

Proof. Since finite sets of equal size are isomorphic, a computation will compute S if and only if it computes any set of size $|S|$. As in Example 2.3, starting with $|S|$ copies of $\{1\}$ and taking their colimit, we get a set of size $|S|$. So, the complexity is bounded above by $|S| + 1$. To see that this is the most efficient way of producing a set with $|S|$ elements, we use Lemma 2.16 below, which states that if we only care about building a single object, then a colimit computation can be replaced by a single colimit on D_0 consisting of basic morphisms. Since the identity on $\{1\}$ is the only basic morphism in this case, taking the colimit of $|S|$ copies of $\{1\}$ is the most efficient way to obtain an object isomorphic to S. □

Example 2.14 (Non-constructive colimit complexity)**.** The following example shows the difference between colimit computations and non-constructive computations. Consider the computation

1. `_,`$\{1\} \xrightarrow{\mathrm{id}} \{1\}$`,1`
2. `_,`$\{1\} \xrightarrow{\mathrm{id}} \{1\}$`,2`
3. `colim(1,2)`
4. `colim(1,2)`

...

a+2. `colim(1,2)`
a+3. `colim(3,4,...,a+2)`

Which produces a set of size 2^a. Observe that only the step `a+3` is not constructive. To make it constructive, one would need to include `1` and `2` in the colimit in step `a+3`, which would make this colimit be a set with just two elements.

Remark 2.15. We can an example similar to the above for mixed computations by alternating limits and colimits to get double exponential set size starting with no basic morphisms, so the mixed complexity of a finite set $|S|$ is $\mathrm{O}(\log\log|S|)$.

2.3. **Useful facts about limit and colimit computations.** We now collect a few facts that will be useful for proving statements about objects and morphisms computed by limit and colimit computations.

The following lemma, which was already used in the proof of Proposition 2.13 above shows that, if the aim is to produce a specific object, intermediate steps in a limit or colimit computation are unnecessary. The key here is the constructivity assumption.

Lemma 2.16. *Assume* $\mathcal{C}$ *has finite products (resp., coproducts). An object produced in a limit computation (resp., colimit computation) is a limit (resp., colimit) of a diagram consisting only of basic morphisms.*

Proof. Let $(D_0, \ldots, D_s)$ be a limit computation and let X be an object appearing in D_s. The point of the statement is that constructivity ensures that the information that would be added in intermediate limits is also included in the final limit that would produce X.

More precisely, let $L_i = \lim D_{i-1}|_{J_i}$ be the limit added to the diagram at the ith step. Let $J_i' = I_0 \cap J_i$. So we have that $D_{i-1}|_{J_i'}$ is the portion of the sub-diagram of $D_{i-1}|_{J_i}$ which is also in D_0. We claim that $L_i \cong \lim D_{i-1}|_{J_i'}$. Indeed, the universal property of limits and constructivity imply that cones from any object Z to $D_{i-1}|_{J_i'}$ can be uniquely extended to cones from Z to $D_{i-1}|_{J_i}$, and therefore $\lim D_{i-1}|_{J_i'}$ satisfies the same universal property as L_i.

The analogous proof holds for colimits. □

The following remarks are very useful for working with objects and morphisms produced in limit computations.

Remark 2.17. If the category $\mathcal{C}$ has finite products and equalizers, then we can write any limit L as an equalizer, see e.g. [Awo10, 5.4]. Let $J = (V, E)$. If $L \cong \lim D$, for a diagram $D : J \to U(\mathcal{C})$ of basic morphisms. We have that L is isomorphic to the limit of the diagram, (i.e. the equalizer)

$$\prod_{v \in V} D(v) \underset{\phi}{\overset{\psi}{\rightrightarrows}} \prod_{e \in E} D(t(e)). .$$

where $\phi_{D(t(e))} = \pi_{D(t(e))}$ and $\psi_{D(t(e))} = D(e) \circ \pi_{D(s(e))}$.

Remark 2.18. If $X \xrightarrow{f} Y$ is a morphism computed by a limit computation, and neither of X and Y is in D_0, then X must have been computed as a limit of a diagram that contains Y. By constructivity, the diagram whose limit is X must contain the diagram that produced Y as a subdiagram. Therefore, if $X = \lim D$ where $D : (V, E) \to U(\mathcal{C})$ is a diagram of basic morphisms, then $Y = \lim D'$, where D' is the subdiagram corresponding to a full subgraph $(V', E') \subset (V, E)$. We then have a commuting diagram

$$\begin{array}{ccccc} X & \longrightarrow & \prod_{v \in V} D(v) & \underset{\phi}{\overset{\psi}{\rightrightarrows}} & \prod_{e \in E} D(t(e)) \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} \\ Y & \longrightarrow & \prod_{v \in V'} D(v) & \underset{\phi'}{\overset{\psi'}{\rightrightarrows}} & \prod_{e \in E'} D(t(e)) \end{array} .$$

In particular, f is induced by the projection π.

3. Limit Computations, Circuits and Algebraic Varieties

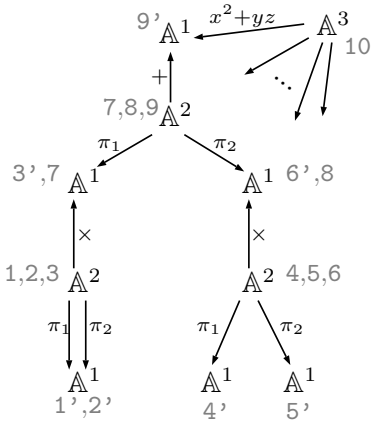
We now make a comparison between the arithmetic complexity of polynomials and the limit-complexity of the varieties which are their zero-sets. For simplicity, we will start with the category of affine algebraic varieties, but what we describe will make sense in other settings like affine schemes and algebras.

Let $\mathcal{C}$ be the category $\mathbf{AffVar}_k$ of affine algebraic varieties over a field k. Let $\mathcal{A}$ consist of the following basic morphisms.

$$\begin{aligned} &\mathbb{A}^1 \xrightarrow{c} \mathbb{A}^1, \text{ for each } c \in k \\ &\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{+} \mathbb{A}^1 \\ &\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{\times} \mathbb{A}^1 \\ &\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{\pi_1, \pi_2} \mathbb{A}^1 \\ &\mathbb{A}^1 \to *, \text{ and } * \xrightarrow{c} \mathbb{A}^1, \text{ for each } c \in k \end{aligned} \tag{3.1}$$

Each of these morphisms is considered to have unit cost.

Example 3.1. As an example, let us make a limit computation of the morphism $\mathbb{A}^3 \xrightarrow{x^2+yz} \mathbb{A}^1$ using these basic morphisms.



1. _,$\mathbb{A}^2 \xrightarrow{\pi_1} \mathbb{A}^1$,1'
2. 1,$\mathbb{A}^2 \xrightarrow{\pi_2} \mathbb{A}^1$,1'
3. 1,$\mathbb{A}^2 \xrightarrow{\times} \mathbb{A}^1$,_
4. _,$\mathbb{A}^2 \xrightarrow{\pi_1} \mathbb{A}^1$,4'
5. 4,$\mathbb{A}^2 \xrightarrow{\pi_2} \mathbb{A}^1$,_
6. 4,$\mathbb{A}^2 \xrightarrow{\times} \mathbb{A}^1$,_
7. _,$\mathbb{A}^2 \xrightarrow{\pi_1} \mathbb{A}^1$,3'
8. 7,$\mathbb{A}^2 \xrightarrow{\pi_2} \mathbb{A}^1$,6'
9. 7,$\mathbb{A}^2 \xrightarrow{+} \mathbb{A}^1$,_
10. lim(1,2,3,4,5,6,7,8,9)

The following shows that a similar computation can be done to compute any polynomial map.

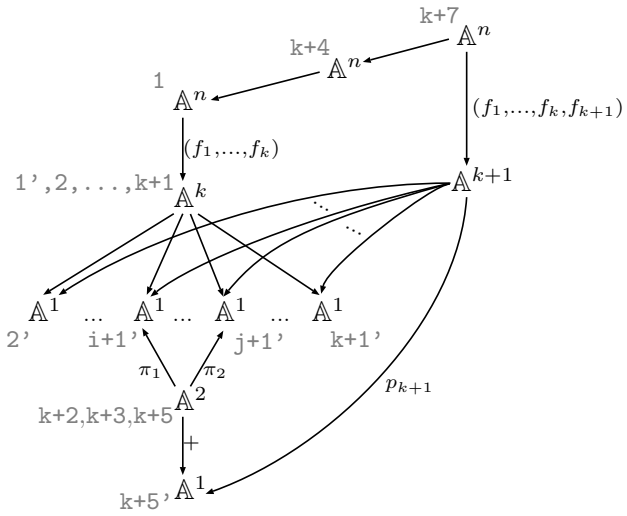
Theorem 3.2. *Let $f \in k[x_1, \ldots, x_n]$ be a polynomial of degree d. Assume that f is computed by a straight line program Γ of length N. Then, the limit-complexity (and therefore the categorical complexity) of the zero-set $X \subset \mathbb{A}^n$ of f is in $\mathrm{O}(N)$.*

Proof. Using the operations in the straight-line program Γ, we will construct a morphism $\mathbb{A}^n \xrightarrow{f} \mathbb{A}^1$. X is then the limit $\mathbb{A}^n \xrightarrow{f} \mathbb{A}^1 \xleftarrow{0} *$.

Associated to each straight-line program of length s, there is a morphism $(f_1, \ldots, f_s) : \mathbb{A}^n \to \mathbb{A}^s$ where $f_1, \ldots, f_s$ are the polynomials computed in each line of the diagram. Let $\Gamma = (\Gamma_1, \ldots, \Gamma_N)$, where Γ_i is the ith instruction in the straight line program Γ. For the first n instructions of Γ, which introduce the variables $x_1, \ldots, x_n$ as polynomials, we have the map $\mathbb{A}^n \xrightarrow{\mathrm{id}} \mathbb{A}^n$. The space $\mathbb{A}^n$ is constructed in a limit computation by taking the limit of n disjoint copies of the diagram consisting of $\mathbb{A}^1$ mapping to itself by 1. To get the map $\mathbb{A}^n \xrightarrow{\mathrm{id}} \mathbb{A}^n$, we take the limit of the whole diagram obtained so far. Costing $n+2$. Note that our diagram also contains all the projections from the source and target of $\mathrm{id}_{\mathbb{A}^n}$ to the n components. This is the base case of the following inductive construction.

Assume that, for $n \leq k \leq N-1$, we have produced, in a limit computation, a morphism $(f_1, \ldots, f_k) : \mathbb{A}^n \to \mathbb{A}^k$ corresponding to the polynomials computed by each line of $(\Gamma_1, \ldots, \Gamma_k)$, together with the projections $p_1, \ldots, p_k$ from $\mathbb{A}^k$ to k copies of $\mathbb{A}^1$, all of which are in D_0.

Assume that Γ_{k+1} is the multiplication of the ith and jth lines of the program, i.e. $f_{k+1} = f_i f_j$. We can then perform the following steps of the computation. For convenience, we have added, as if they were basic morphisms, the portion of the diagram used for the next steps as the first $k+1$ steps in this description:



1. `_,`$\mathbb{A}^n \xrightarrow{(f_1,\ldots,f_k)} \mathbb{A}^k$ `,1'`
2. `1',`$\mathbb{A}^k \xrightarrow{p_1} \mathbb{A}^1$`,2'`
3. `1',`$\mathbb{A}^k \xrightarrow{p_2} \mathbb{A}^1$`,3'`

   ...

k+1. `1',`$\mathbb{A}^k \xrightarrow{p_k} \mathbb{A}^1$`,(k+1)'`
k+2. `_,`$\mathbb{A}^2 \xrightarrow{\pi_1} \mathbb{A}^1$`,(i+1)'`
k+3. `k+2,`$\mathbb{A}^2 \xrightarrow{\pi_2} \mathbb{A}^1$`,(j+1)'`
k+4. `lim((k+2),(i+1)',(j+1)', 1',1)`
k+5. `k+2,`$\mathbb{A}^2 \xrightarrow{\times} \mathbb{A}^1$`,_`
k+6. `lim(2',3',...,(k+1)',(k+4)')`
k+7. `lim((k+4), 1,1',2',3',..., (k+1)', k+2, (k+5)',k+6)`

In the end, the map `(k+7)->(k+6)` is the map $\mathbb{A}^n \xrightarrow{(f_1,\ldots,f_{k+1})} \mathbb{A}^{k+1}$, and the projection maps for the next step are the maps `(k+6)->2',(k+6)->3',...,(k+6)->(k+1)', (k+6)->(k+5)'`. The process for addition steps is similar, with the step `k+5` modified. For scalar multiplication, it is similar with two steps less. For constants appearing in the computation, we add a new variable and take fiber product with $* \xrightarrow{c} \mathbb{A}^1$ to fix the variable to the value c. Repeating this process until, k=N, we see that $(f_1, \ldots, f_N)$ is produced in $O(N)$ steps. We can then compose with the projection to the last coordinate, by taking a limit, to produce $\mathbb{A}^n \xrightarrow{f} \mathbb{A}^1$. To obtain the zero-set of f, we add $* \xrightarrow{0} \mathbb{A}^1$ to this last $\mathbb{A}^1$ and take the limit.

It should also be noted that the way we made a limit computation for $\mathbb{A}^n \xrightarrow{f} \mathbb{A}^1$ was not the most efficient, but this way gives the cleanest inductive argument. For efficiency, the intermediate limit steps can be removed; c.f. Lemma 2.16. □

Remark 3.3. The proof above shows that we can, by starting from a number of $\mathbb{A}^1$ added as basic objects, construct any polynomial functions $f_1 : \mathbb{A}^n \to \mathbb{A}^1$. We will use this fact later.

We now consider a converse for Theorem 3.2 and show that, given a limit computation with low cost, an object X computed by it is *isomorphic to* the zero-set of a polynomial whose arithmetic complexity is low. Since diagram computations produce objects up to

isomorphism, categorical complexity does not reflect the complexity of every polynomial that might be used to cut out X in a larger space. For example, X could be the graph of a polynomial map $f : \mathbb{A}^n \to \mathbb{A}^1$ with very high arithmetic complexity, but since X would be isomorphic to $\mathbb{A}^n$, its limit complexity would be very small, which does not say anything about the arithmetic complexity of f. This is discussed in more detail in Section 4.1.

Theorem 3.4. *Let $(D_0, \ldots, D_s)$ be a limit computation in in $\mathbf{AffVar}_k$, of cost $C = c(D_0, \ldots, D_s)$. Then:*

(*i*) *If X is on object computed by $(D_0, \ldots, D_s)$, then X is isomorphic to the zero-set of a polynomial morphism $\mathbb{A}^{m_1} \to \mathbb{A}^{m_2}$ whose components are polynomials of degree at most 2. The total arithmetic complexity of the map is bounded above by $4C$.*

(*ii*) *Every morphism $f : X \to Y$ in D_s is a the restriction of a projection $\mathbb{A}^{m_1} \to \mathbb{A}^{m_1'}$ where $\mathbb{A}^{m_1}$ and $\mathbb{A}^{m_2}$ are the spaces from part (i) where X and Y are embedded respectively.*

Proof. If $s = 0$, then the statement is true since the basic morphisms are $\mathbb{A}^1 \xrightarrow{c} \mathbb{A}^1$, $\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{+,\times} \mathbb{A}^1$, $\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{\pi_1,\pi_2} \mathbb{A}^1$, $\mathbb{A}^1 \to *$, and $* \xrightarrow{c} \mathbb{A}^1$.

For the general case, constructivity implies that, as far as the isomorphism class of an object X in D_s is concerned, the intermediate limits in a constructive limit computation can be removed (Lemma 2.16). Using the notation of the proof of Lemma 2.16, if $X = L_i$ is produced as a limit in the diagram computation, then $X \cong \lim D_{i-1}|_{J_i'}$ where $D_{i-1}|_{J'}$ is a subdiagram of D_0.

As discussed in Remark 2.17, we can write $X = L_i$ as an equalizer. Let $J_i' = (V, E)$, $L_i \cong \lim D_{i-1}|_{J_i'}$; writing $D = D_{i-1}|_{J_i'}$, L_i is isomorphic to the limit of

$$\prod_{v \in V} D(v) \underset{\phi}{\overset{\psi}{\rightrightarrows}} \prod_{e \in E} D(t(e)).$$

where $\phi_{D(t(e))} = \pi_{D(t(e))}$ and $\psi_{D(t(e))} = D(e) \circ \pi_{D(s(e))}$.

Since $J_i' \subset I_0$, we have that each $D(v)$ or $D(t(e))$ is either $\mathbb{A}^1$, $\mathbb{A}^2$, or a point; and $D(e)$ are addition, multiplication, constant, projection, or multiplication by a constant. So the above equalizer is of the form

$$\mathbb{A}^{m_1} \underset{\phi}{\overset{\psi}{\rightrightarrows}} \mathbb{A}^{m_2},$$

and L_i is isomorphic to the zero locus of $\mathbb{A}^{m_1} \xrightarrow{\phi - \psi} \mathbb{A}^{m_2}$. For the complexity of $\phi - \psi$, observe that $m_1, m_2 \leq 2C$, and that each component of $\phi - \psi$ is a very simple polynomial which can be produced in two steps.

We now consider the second assertion. This follows directly from the discussion in Remark 2.18. We observe that any morphism $f : X \to Y$ appearing in D_s but not in D_0 must be a cone map from a limit $X = L_i = \lim D_{i-1}|_{J_i}$ to an object Y appearing in $D_{i-1}|_{J_i}$. By the above construction, we know that X and Y are zero-loci of morphisms whose sources are products of objects in sub-diagrams of D_0. Moreover constructivity implies that the subdiagram for Y is contained in the sub-diagram for X. The morphism f is then the restriction of the projection from the sub-diagram for X to the sub-diagram for Y.

□

3.1. **Affine Schemes and k-Algebras.** The above arguments can also be considered for the category of affine schemes instead of varieties. Let k be a field and let $\mathbf{AffSch_k}$ be

the category of affine schemes. Consider the same set $\mathcal{A}$ of morphisms as in the affine-variety case: $\mathbb{A}^1 \xrightarrow{c} \mathbb{A}^1$, for each $c \in k$, $\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{+} \mathbb{A}^1$, $\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{\times} \mathbb{A}^1$, $\mathbb{A}^1 \times \mathbb{A}^1 \xrightarrow{\pi_1,\pi_2} \mathbb{A}^1$, $\mathbb{A}^1 \to *$, and $* \xrightarrow{c} \mathbb{A}^1$.

One can also consider the category $\mathbf{Alg}_k$ of k-algebras with basic morphisms

$$k[x] \xrightarrow{c} k[x], \text{ for each } c \in k$$
$$k[x] \xrightarrow{x\otimes 1+1\otimes x} k[x] \otimes k[x]$$
$$k[x] \xrightarrow{x\otimes x} k[x] \otimes k[x]$$
$$k[x] \xrightarrow{id\otimes 1,\, 1\otimes id} k[x] \otimes k[x]$$
$$k \xrightarrow{1} k[x], \text{ and}$$
$$k[x] \xrightarrow{x\mapsto c} k, \text{ for each } c \in k$$

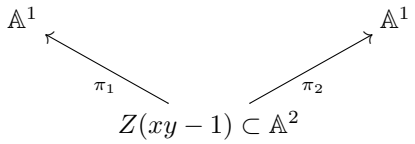
These correspond to the basic morphisms considered for $\mathbf{AffVar}_k$ under the adjoint equivalence

$$\mathbf{AffSch}_k \underset{\text{Spec}}{\overset{k[\cdot]}{\rightleftarrows}} \mathbf{Alg}_k^{\text{op}} .$$

Without any modification to the proofs, Theorems 3.2 and 3.4 hold when $\mathbf{AffVar}_k$ is replaced by $\mathbf{AffSch}_k$ or $\mathbf{Alg}_k$.

3.2. **Mixed computation and projective schemes.** Let $\mathcal{C}$ be the category $\mathbf{Sch}_k$ of all schemes over a field k. Letting $\mathcal{A}$ consist of the morphisms above as in the affine scheme case, we get a definition of complexity in the category of schemes.

The morphisms in $\mathcal{A}$ are actually enough to produce all projective schemes using mixed computations. For example, in order to make $\mathbb{P}^1$, we can produce the following diagram as a subdiagram of a computation and take its colimit:

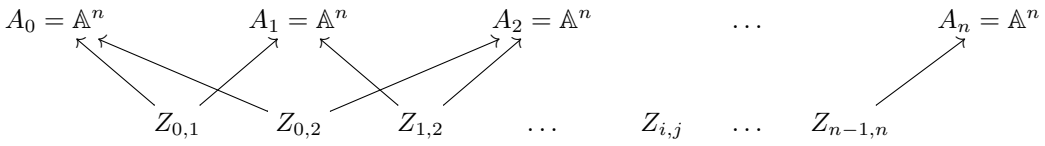


Note that $Z(xy-1)$ is isomorphic to $\mathbb{A}^1 - \{0\}$.

Proposition 3.5. *Let $X \subset \mathbb{P}^n_k$ be the zero-scheme of homogeneous polynomials $f_1, \ldots, f_m \in k[x_0, \ldots, x_n]$. Assume that, for each i, there is an arithmetic circuit of size c_i computing f_i. Then, the categorical complexity of X is in $\mathrm{O}(n^2(c_1 + \cdots + c_s))$.*

Proof. We first show how to construct $\mathbb{P}^n$ by a mixed computation and then modify the construction to make the zero-scheme X of $f_1, \ldots, f_m$.

The last step in making $\mathbb{P}^n$ will be to take the colimit of the diagram



There are $n+1$ copies of $\mathbb{A}^n$ in the first row, which correspond to the standard covering $U_0, \ldots, U_n$ of $\mathbb{P}^n$ by affine opens. In the second row, there is an object $Z_{i,j}$ for each pair

(i,j) with $i<j$; with each $Z_{i,j}$ corresponding to the intersections $U_i \cap U_j$ of the affine opens in the chart, each therefore isomorphic to the complement of a hyperplane in $\mathbb{A}^n$.

We will construct each Z_{ij} as a subscheme of $\mathbb{A}^n \times \mathbb{A}^n = A_i \times A_j$, considered with coordinates $x_1,\ldots,x_n,y_1,\ldots,y_n$, defined by the equation:

$$y_i x_j - 1 = 0, \tag{3.2}$$

and, for each $l \in \{1,\ldots,i-1,i+1,\ldots,n\}$, the equations

$$y_i x_l - y_l = 0. \tag{3.3}$$

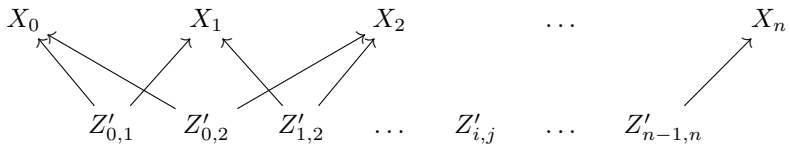
These equations describe the graph of the transition maps

$$(x_1,\ldots,x_n) \mapsto (\frac{x_1}{x_j},\frac{x_2}{x_j},\ldots,\frac{x_{i-1}}{x_j},\frac{1}{x_j},\frac{x_{i+1}}{x_j},\ldots,\frac{x_{j-1}}{x_j},\frac{x_{j+1}}{x_j},\ldots,\frac{x_n}{x_j})$$

between the affine opens in the standard covering of $\mathbb{P}^n$. Here, each affine open U_i, corresponding to points in homogeneous coordinates $[\frac{a_0}{a_i} : \cdots : \frac{a_{i-1}}{a_i} : 1 : \frac{a_{i+1}}{a_i} : \cdots : \frac{a_n}{a_i}]$ is parameterized by simply omitting the ith variable. The maps $Z_{i,j} \to A_i$ and $Z_{i,j} \to A_j$ are the restrictions of the projection maps from $A_i \times A_j$.

To make the computation, start with $(n+1)$ sets of n copies of $\mathbb{A}^1$. Make A_i as the limit of the ith set of n $\mathbb{A}^1$'s. Using the procedure described in the proof of Theorem 3.2 (c.f. Remark 3.3), construct each $Z_{i,j}$ as the zero-set of the equations (3.2) and (3.3). Making the projections to the A_i (c.f. second part of Remark 2.6), we get the diagram above. At this point, the colimit of this diagram can be taken to produce $\mathbb{P}^n$.

To make the zero-scheme X of $f_1,\ldots,f_m$ in $\mathbb{P}^n$, we continue in order to make the following diagram.



where the X_i are isomorphic to $X \cap U_i$ and the $Z'_{i,j}$ are isomorphic to $X \cap U_i \cap U_j$. To make the X_i, de-homogenize $f_1,\ldots,f_m$ for each A_i so that we get the equations for the subscheme of A_i that corresponds to $X \cap U_i$. Using the procedure in Theorem 3.2, take the zero-scheme of these de-homogenized equations in each A_i. To make the $Z'_{i,j}$, make the map $X_i \times X_j \to A_i \times A_j$ and pull back $Z_{i,j}$. Finally, take the colimit of the diagram above to get a scheme isomorphic to X.

□

4. Categorical Complexity of Morphisms vs Circuit Complexity of Polynomials

The aim of this section is to see how closely we can recover arithmetic circuit complexity from categorical complexity in the appropriate choice of category.

We discuss three categories where we can compare the arithmetic circuit complexity of a polynomial $f \in k[x_1,\ldots,x_n]$ with the categorical complexity of a morphism diagram. In the first one, we look at the morphism diagram $\mathrm{D}_f = (\mathbb{A}^n \xrightarrow{f} \mathbb{A}^1)$ in $\mathbf{AffVar}_k$ and see that its complexity can be very different than the arithmetic complexity of f. The second category, the category of graded pairs of algebras, is an attempt at removing this discrepancy. The third category is the category of modules over $k[x_1,\ldots,x_n]$ where we prove concrete

comparison results between the arithmetic complexity of f and the categorical complexity of the morphism diagram $k[x_1,\dots,x_n] \xrightarrow{1\mapsto f} k[x_1,\dots,x_n]$.

4.1. **Complexity of polynomial morphisms in $\mathbf{AffVar}_k$.** We consider the category $\mathbf{AffVar}_k$ with the basic morphisms discussed above (3.1). Given a polynomial $f \in k[x_1,\dots,x_n]$, what is the categorical complexity of the diagram $\mathrm{D}_f = (\mathbb{A}^n \xrightarrow{f} \mathbb{A}^1)$ in $\mathbf{AffVar}_k$?

Categorical computations produce diagrams up to isomorphism, and categorical complexity is defined for isomorphism classes of diagrams. So, the complexity of f is equal to the complexity of any other $\mathbb{A}^n \xrightarrow{g} \mathbb{A}^1$ where there is an automorphism $\phi : \mathbb{A}^n \to \mathbb{A}^n$ such that

$$\begin{array}{ccc} \mathbb{A}^n & \xrightarrow{f} & \mathbb{A}^1 \\ {\scriptstyle\phi}\downarrow & & \downarrow{\scriptstyle 1} \\ \mathbb{A}^n & \xrightarrow{g} & \mathbb{A}^1 \end{array}$$

commutes. Therefore, the complexity of D_f in $\mathbf{AffVar}_k$ is invariant under polynomial automorphisms of $\mathbb{A}^n$, and should capture the 'complexity' of the geometric object which is the zero-set of f rather than the complexity of the polynomial f.

For example, let $p \in k[x_1,\dots,x_n]$ be any polynomial, and let $g \in k[x_1,\dots,x_n,x_{n+1}]$, $g = x_{n+1}+p(x_1,\dots,x_n)$. Let $f = \pi_{n+1}$ and $\phi(x_1,\dots,x_{n+1}) = (x_1,\dots,x_n,x_{n+1}+p(x_1,\dots,x_n))$. Then, we have the following commuting diagram

$$\begin{array}{ccc} \mathbb{A}^{n+1} & \xrightarrow{\pi_{n+1}} & \mathbb{A}^1 \\ {\scriptstyle (x_1,\dots,x_n,x_{n+1}+p(x_1,\dots,x_n))}\uparrow & \nearrow_{x_{n+1}+p(x_1,\dots,x_n)} & \\ \mathbb{A}^{n+1} & & \end{array}$$

So, the diagram $\mathrm{D}_{x_{n+1}+p(x_1,\dots,x_n)}$ is isomorphic to $\mathbb{A}^{n+1} \xrightarrow{x_{n+1}} \mathbb{A}^1$. So, while the circuit complexity of $x_{n+1} + p(x_1,\dots,x_n)$ can be very high (for example, p could be the permanent, or worse, a generic polynomial), the complexity of $\mathrm{D}_{x_{n+1}+p(x_1,\dots,x_n)}$ is trivial (i.e. bounded by a constant independent of n). This is because, geometrically, the zero-set of $x_{n+1}+p(x_1,\dots,x_n)$ is very simple, it is the graph of $-p$, and is therefore isomorphic to $\mathbb{A}^n$.

It should also be noted that reductions in circuit complexity do not immediately lead to reductions in the complexity in $\mathbf{AffVar}_k$. Even though $p(x_1,\dots,x_n)$ reduces to $x_{n+1} + p(x_1,\dots,x_n)$ in Valiant complexity, this does not lead to an easy diagram computation of $D_{p(x_1,\dots,x_n)}$ from a computation of $D_{x_{n+1}+p(x_1,\dots,x_n)}$. Still, one can ask whether polynomials which are believed to be hard to compute in Valiant's model also have high categorical complexity. For example:

Question 4.1. *Is the limit/mixed complexity of $\mathbb{A}^{n^2} \xrightarrow{\mathrm{perm}_n} \mathbb{A}^1$ polynomially bounded in n?*

4.2. **Pairs of Graded Algebras.** It is possible that the difference between complexity of D_f in $\mathbf{AffVar}_k$ and the Valiant complexity of f is caused by the large number of automorphisms of $\mathbb{A}^n$. The aim in this section is to consider a category where Valiant complexity is possibly close to categorical (colimit) complexity.

Since projective space has relatively few automorphisms, the complexity of a projective hypersurface $Z(f)$ should be closer to the Valiant complexity of its defining polynomial f. However, since the basic objects are not projective, it is better to consider the corresponding setup in the category of graded algebras.

To this end, we could consider the category $\mathbf{GrAlg}_k$ of graded algebras considered with the basic morphisms.

$$\begin{aligned}
k[z^n] &\xrightarrow{+} k[x^n, y^n],\ z^n \mapsto x^n + y^n,\\
k[z^{2n}] &\xrightarrow{\cdot} k[x^n, y^n],\ z^{2n} \mapsto x^n y^n,\\
k[z^n] &\xrightarrow{i_1} k[x^n, y^n],\ z^n \mapsto x^n,\\
k[z^n] &\xrightarrow{i_2} k[x^n, y^n],\ z^n \mapsto y^n,\\
k[z^n] &\xrightarrow{c\times} k[z^n],\ z^n \mapsto cz^n, c \in k,\\
k[z^n] &\to k,\ z^n \mapsto 0,
\end{aligned}$$

and consider, for a homogeneous polynomial $f \in k[x_1, \ldots, x_n]$, the diagram

$$k[x] \xrightarrow{z \mapsto f} k[x_1, \ldots, x_n]$$

denoted also by D_f. However, the complexity of D_f would still be different than the Valiant complexity of f. Indeed, we could have $k[x_1, \ldots, x_n]$ be the coordinate algebra of an embedded $\mathbb{P}^{n-1} \subset \mathbb{P}^{N-1} = \operatorname{Proj} k[z_1, \ldots, z_n]$, whose defining equations are easy to compute categorically. As a result, we would have the induced morphisms $k[z_j] \xrightarrow{z_j \mapsto g_j} k[x_1, \ldots, x_n]$ easy to compute. But the polynomial g_j are the result of elimination and could therefore have high complexity.

To prevent this, we consider embedded spaces $X \subset \mathbb{P}^n$. Or rather, we consider the corresponding morphisms of the coordinate algebras. We denote by $\mathbf{GrAlgPairs}_k$ the category whose objects are surjective morphisms $(A \xrightarrow{f} B)$, where A is isomorphic to some polynomial ring $k[x_0, \ldots, x_n]$ graded by degree. For example, suppose that $A = k[x]$ and $B = k[x]/(x^2)$ both graded by degree, and $A \xrightarrow{f} B$ the canonical surjection.

In order to define categorical complexity in $\mathbf{GrAlgPairs}_k$ we define the basic morphisms as follows. For each $n \geq 1$, we include in the set of basic morphisms the following set of morphisms (all polynomial rings appearing below are graded by degrees):

$$\begin{aligned}
(k[z^n] \xrightarrow{\sim} k[z^n]) &\xrightarrow{+} (k[x^n, y^n] \xrightarrow{\sim} k[x^n, y^n]),\ z^n \mapsto x^n + y^n,\\
(k[z^{2n}] \xrightarrow{\sim} k[z^{2n}]) &\xrightarrow{\cdot} (k[x^n, y^n] \xrightarrow{\sim} k[x^n, y^n]),\ z^{2n} \mapsto x^n y^n,\\
(k[z^n] \xrightarrow{\sim} k[z^n]) &\xrightarrow{i_1} (k[x^n, y^n] \xrightarrow{\sim} k[x^n, y^n]),\ z^n \mapsto x^n,\\
(k[z^n] \xrightarrow{\sim} k[z^n]) &\xrightarrow{i_2} (k[x^n, y^n] \xrightarrow{\sim} k[x^n, y^n]),\ z^n \mapsto y^n,\\
(k[z^n] \xrightarrow{\sim} k[z^n]) &\xrightarrow{c\times} (k[z^n] \xrightarrow{\sim} k[z^n]),\ z^n \mapsto cz^n, c \in k,\\
(k[z^n] \xrightarrow{\sim} k[z^n]) &\to (k[x^n] \xrightarrow{(x \mapsto 0)} k),\ z \mapsto 0.
\end{aligned}$$

Notice that we can obtain the morphism

$$(k[z^n] \xrightarrow{(z^n \mapsto 0)} k) \xrightarrow{(z^n \mapsto 0)} (k[x^n] \xrightarrow{\sim} k[x^n])$$

as the colimit diagram:

$$\begin{array}{ccccc}
(k[t^n] \xrightarrow{\sim} k[t^n]) & \xrightarrow{t^n \mapsto 0, t^n \mapsto 0} & (k[y^n] \xrightarrow{\sim} k[y^n]) \\
\downarrow{\scriptstyle t^n \mapsto z^n, t^n \mapsto 0} & & \downarrow \\
(k[z^n] \xrightarrow{(z^n \mapsto 0)} k) & \xrightarrow{(z^n \mapsto 0)} & (k[x^n] \xrightarrow{\sim} k[x^n])
\end{array}$$

Given a homogeneous polynomial $f \in k[x_1, \ldots, x_n]$, we consider the categorical complexity of the diagram M_f defined as,

$$(k[z] \xrightarrow{\sim} k(z)) \xrightarrow{M_f} (k[x_0, \ldots, x_n] \xrightarrow{\sim} k[x_0, \ldots, x_n]), z \mapsto f.$$

Following the same proof as Theorem 3.2, we have:

Proposition 4.2. *Given a homogeneous polynomial $f \in k[x_1, \ldots, x_n]$ of arithmetic complexity N. The colimit complexity of M_f is in $\mathrm{O}(N)$.*

We ask whether the converse is true.

Question 4.3. *For a homogeneous polynomial $f \in k[x_0, \ldots, x_n]$ for which the colimit complexity of the morphism diagram M_f in $\mathbf{GrAlgPairs}_k$ is N, is the arithmetic circuit complexity of f polynomially bounded in N.*

4.3. **Modules over Polynomial Rings.** We now consider the relationship between arithmetic complexity and categorical complexity in categories of modules.

Let $R = k[x_1, \ldots, x_n]$. We consider the category R-**Mod** be the category of R-modules. We consider colimit computations in R-**Mod** with these basic objects.

$$\begin{aligned}
&R \xrightarrow{x_i} R, \text{ for each } i = 1, \ldots, n \\
&R \xrightarrow{c} R, \text{for each } c \in k \\
&R \xrightarrow{i_1, i_2} R \oplus R \\
&R \xrightarrow{\Delta} R \oplus R \\
&R \oplus R \xrightarrow{+} R \\
&R \to \{0\}
\end{aligned}$$

For a polynomial $f \in k[x_1, \ldots, x_n]$, we consider the corresponding morphism $R \xrightarrow{f} R$ that sends 1 to f. Recall that a formula is an arithmetic circuit or straight line program where past intermediate computations cannot be re-used.

Proposition 4.4. *If a polynomial $f \in k[x_1, \ldots, x_n]$ is computed by a formula of size s, then the diagram $R \xrightarrow{f} R$ is computed by a colimit computation in R-**Mod** of cost in $\mathrm{O}(s)$.*

Proof. Without loss of generality, assume that all sum and product gates have fan-in at most two. We will build, for each formula C, a diagram D_C whose colimit will contain $R \xrightarrow{p_c} R$ where p_C is the output polynomial of C. This will be done inductively on the size of C.

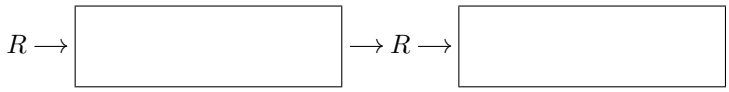
Each D_C will be a diagram of the form

$$R \longrightarrow \boxed{} \longrightarrow R\,.$$

whose colimit is R with the morphism from the R on the right to the colimit being id_R and the morphism from the R on the left to the colimit being defined by $1 \mapsto p_C$.

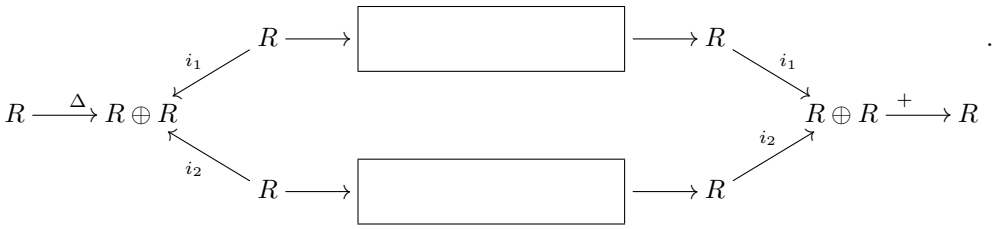
If the output p_C of C is one of the variables x_i let D_C be the diagram $R \xrightarrow{x_i} R$. If it just a constant, then D_C is $R \xrightarrow{c} R$.

If the top gate of C is a product gate with C' and C'' as the left and right sub-circuits, then we set D_C by chaining together $D_{C'}$ and $D_{C''}$:



The map from the left-most R to the colimit is the composition $R \xrightarrow{p_{C'}} R \xrightarrow{p_{C''}} R$, which is $R \xrightarrow{p_{C'} p_{C''}} R$.

If the top gate of C is a sum gate with C' and C'' as the left and right sub-circuits, then we define D_C as



where the top and bottom rows are $D_{C'}$ and $D_{C''}$. The colimit of this diagram is again R with the map from the left-most R to the colimit being $p_C + p_{C'}$. □

What about a converse? What does the existence of a colimit computation in R-**Mod** that produces $R \xrightarrow{1 \mapsto f} R$ say about the complexity of f?

Theorem 4.5. *Let $R = k[x_1, \ldots, x_n]$. If $R \xrightarrow{f} R$ is computed in a colimit computation of cost c in R-**Mod**, then there is an arithmetic circuit of size* $\mathrm{poly}(c)$ *with inputs $x_1, \ldots, x_n$, that computes f.*

Proof. Consider a diagram $D : I \to R$-**Mod** consisting only of the basic morphisms described above. Assume that we have $\operatorname{colim} D = R$.

For each vertex $v \in I$, we have that $D(v)$ is R, $R \oplus R$ or $\{0\}$. For each v such that $D(v) = R$, let f_v be the image of 1 under the morphism $R \xrightarrow{1 \mapsto f_v} R$ from $D(v)$ to the colimit R. If $D(v) = \{0\}$, then we set $f_v = 0$. If $D(v) = R \oplus R$, then we set two polynomials f_v and $f_{v'}$ so that the map $R \oplus R \to R$ to the colimit is given by $(1, 0) \mapsto f_v$ and $(0, 1) \mapsto f_{v'}$. We will prove that each f_v is computed by a polynomially sized circuit.

We are considering the f_v's as unknowns in a system of equations. For each arrow in D, we consider one or two R-linear equations. For an arrow $D(v_1) \to D(v_2)$ of the form given in the left column, we add the equations in the right column:

$$\begin{array}{cc}
R \xrightarrow{x_i} R & f_{v_1} - x_i f_{v_2} \\
R \xrightarrow{c} R, & f_{v_1} - c f_{v_2} \\
R \xrightarrow{i_1, i_2} R \oplus R & f_{v_1} - f_{v_2} \text{ or } f_{v_1} - f_{v_2'} \\
R \xrightarrow{\Delta} R \oplus R & f_{v_1} - f_{v_2} - f_{v_2'} \\
R \oplus R \xrightarrow{+} R & f_{v_1} - f_{v_2} \text{ and } f_{v_1'} - f_{v_2} \\
R \to \{0\} & f_{v_1} = 0 \text{ and } f_{v_2} = 0.
\end{array}$$

In this way, we obtain a homogeneous system $k[x_1, \ldots, x_n]$-linear equations; $A\vec{f} = 0$, $A \in \mathrm{Mat}_{n_2 \times s}(R)$. Tuples $\vec{f} = (f_{v_1}, f_{v_2}, f_{v_3}, \ldots, f_{v_s}) \in k[x_1, \ldots, x_n]^s$ that satisfy this system of equations correspond to a cocones of the diagram D with target R.

Since the colimit of D is R, for any such cocone corresponding to $(f_{v_1}, \ldots, f_{v_s})$, there will be a map $(\mathrm{colim}\, D = R) \xrightarrow{1 \mapsto g} R$ making the diagram containing the new cocone, the colimit cocone and the map $R \xrightarrow{1 \mapsto g} R$ commute. This implies that g divides each f_{v_j}. Since the colimit is the initial cocone, we can find the tuple of polynomials corresponding to the colimit cocone by taking $(\frac{f_{v_1}}{h}, \frac{f_{v_2}}{h}, \ldots, \frac{f_{v_s}}{h})$ where $h = \gcd(f_{v_1}, \ldots, f_{v_s})$. Thus, assuming the colimit is R, to compute the map from every $D(v)$ to the colimit, it suffices to: (i) find a solution to the above system of equations for D, and (ii) divide by $\gcd(f_{v_1}, \ldots, f_{v_s})$.

To solve the system $A\vec{f} = 0$, we use Gaussian elimination over the field $k(x_1, \ldots, x_n)$. Let R and C be square matrices with entries in $k(x_1, \ldots, x_n)$ such that RAC is a diagonal matrix, in the sense that it contains an $r \times r$ minor which is I_r, with $r < s$, and all other entries are 0. Following the steps of Gaussian elimination, there exist circuits (or straight-line programs) with division that produce each entry of R and C in time $\mathrm{poly}(s)$. Also, observe that the entries of R and C have degree at most s. Without loss of generality, assume $RACe_1 = 0$. Then

$$Ce_1 = \left(\frac{p_1}{q_1}, \ldots, \frac{p_s}{q_s}\right)^T \in k(x_1, \ldots, x_n)^s$$

is a solution to $A\vec{f} = 0$.

Before we proceed further and get this solution into $k[x_1, \ldots, x_n]^s$, we make a small digression into algorithms about straight-line programs. We consider the following, which are both proven by the algorithms in Kaltofen's [Kal88]:

(GCD) Greatest common divisor: Given polynomials $f_1, \ldots, f_q \in k[x_1, \ldots, x_n]$ which are the output of a circuit of size s', there is a circuit of size $\mathrm{poly}(s')$ that produces the greatest common divisor of $f_1, \ldots, f_q$.

(DE) Denominator Extraction: Given a reduced rational function $\frac{p(x)}{q(x)} \in k(x_1, \ldots, x_n)$, produced by a circuit with division of size s', there is a circuit of size $\mathrm{poly}(s')$ that produces q.

In *loc. cit.*, randomized algorithms that produce the output circuits for both DE and GCD are presented. But the circuits that output the gcd and the denominator are themselves not randomized. This will be enough in our non-uniform setting; we only use the existence of the of the polynomial size circuits that produce the gcd and the denominator and do not use the algorithm that produces the circuits themselves.

Going back to the proof, we made a system of equations $A\vec{f} = 0$ from the diagram of basic morphisms and got a solution of the form $Ce_1 = (p_1/q_1, \ldots, p_s/q_s)^T \in k(x_1, \ldots, x_n)^s$ and wanted to produce a solution in $k[x_1, \ldots, x_n]^s$ instead. To do this, first use Kaltofen's GCD to assume, without loss of generality that each $\frac{p_i}{q_i}$ is reduced. Then use Kaltofen's DE to extract the denominators q_i from each fraction. The element $q_1 q_2 \ldots q_s Ce_1$ is in $k[x_1, \ldots, x_n]^s$ and is a solution to $A\vec{f} = 0$. Now divide $q_1 q_2 \ldots q_s Ce_1$ by the gcd of all of its entries to obtain $(f_{v_1}, \ldots, f_{v_s}) \in k[x_1, \ldots, x_n]^s$. The polynomials $f_1, \ldots, f_s$ correspond to the morphisms from D to the colimit R. Each f_v is produced by a circuit with division, but since the degrees of f_v are polynomially bounded, each such circuit can be turned into a circuit without division using Strassen's method [Str73]. This concludes the proof that for any diagram of basic morphisms with R as a colimit, every colimit cocone morphism from an object in D sends 1 to an f_v which is computed by a circuit of size polynomial in s.

We now prove the theorem. Let $R \xrightarrow{1\mapsto f} R$ be a sub-diagram of a colimit computation with initial step D_0. Call the source R_1 and the target R_2. By Lemma 2.16, there is a sub-diagram $D_0' \subset D_0$ of basic morphisms whose colimit is R_1; and (c.f. Remark 2.18) there is a subdiagram $D_0'' \subset D_0$, which contains D_0', and whose colimit is R_2, with the induced map $R_1 \to R_2$ being a map that sends $1 \mapsto f$. This implies, combined with the first part of this proof applied to both D_0' and D_0'', that f is the quotient of two polynomials computed by polynomially sized circuits. Hence, by Strassen's method, f is computed by a circuit of cost polynomial in the size of D_0. □

5. Functors

Here we discuss phenomena of categorical complexity related to functors between categories.

5.1. **Preservation Under Functors.** Let $F : \mathcal{C} \to \mathcal{D}$ be a functor. If D is a diagram in $\mathcal{C}$, the image diagram $F(D)$ is defined in the obvious way. If F preserves finite limits, then, for every every limit computation $(D_0, \ldots, D_s)$ in $\mathcal{C}$, starting with a set of basic morphisms $\mathcal{A}$ will correspond to a limit computation $(F(D_0), \ldots, F(D_s))$ in $\mathcal{D}$ which starts with basic morphisms in $F(\mathcal{A})$. Therefore, in this case, we have

$$c^{\lim}_{\mathcal{D},F(\mathcal{A})}(F(D)) \leq c^{\lim}_{\mathcal{C},\mathcal{A}}(D).$$

We have the analogous statement for colimit computations if F preserves colimits.

Since equivalences preserve limits and colimits, categorical complexity, limit complexity, colimit complexity and mixed complexity are all invariant under equivalences of categories. A more general case is that of adjoint functors. Let $R : \mathcal{C} \to \mathcal{D}$ be a functor right-adjoint to a functor $L : \mathcal{D} \to \mathcal{C}$. Then, since right-adjoints preserve limits and left adjoints preserve colimits. We then have the following.

Lemma 5.1. *Let $R : \mathcal{C} \to \mathcal{D}$ and $L : \mathcal{D} \to \mathcal{C}$ be a pair of adjoint functors. Let $\mathcal{A}$ be a set of basic morhisms in $\mathcal{C}$, and $\mathcal{A}'$ be a set of basic morphisms in $\mathcal{D}$, such that $R(\mathcal{A}) \subset \mathcal{A}'$ and $L(\mathcal{A}') \subset \mathcal{A}$. Then we have inequalities of complexities*

$$c^{\lim}_{\mathcal{D},\mathcal{A}'}(R(D)) \leq c^{\lim}_{\mathcal{C},\mathcal{A}}(D),$$
$$c^{\mathrm{colim}}_{\mathcal{C},\mathcal{A}}(L(D')) \leq c^{\mathrm{colim}}_{\mathcal{D},\mathcal{A}}(D'),$$

for every diagram D in $\mathcal{C}$ and D' in $\mathcal{D}$; similarly for constructive limit and colimit complexity.

Example 5.2. Let $\mathbf{Vect}_k$ be the category of vector spaces over a field k. As the free vector space functor $\mathrm{Fr} : \mathbf{Set} \to \mathbf{Vect}_k$ is left adjoint to the forgetful functor $\mathrm{Fo} : \mathbf{Vect}_k \to \mathbf{Set}$, we have that Fr preserves colimits. In particular, the constructive colimit complexity of a vector space is bounded above by its dimension.

5.2. **Complexity of Functors.** We discussed in Section 5.1 above, how adjoint functors preserve limits or colimits. Here, we take a different point of view and consider complexity directly as a function on a category. We can then define the complexity of a functor as a function of n.

Definition 5.3 (Complexity of functors)**.** Let $\mathcal{C}, \mathcal{D}$ be two categories with complexity functions, $C_{\mathcal{C}}, C_{\mathcal{D}}$, and let $F : \mathcal{C} \to \mathcal{D}$ be a functor. We define the complexity, $c(F) : \mathbb{N} \to \mathbb{N}$ by

$$c(F)(n) = \sup \left\{ C_{\mathcal{D}}(D) \mid I \text{ a finite directed graph, } D \in \mathcal{C}^I,\ C_{\mathcal{C}}(D) \leq n \right\}.$$

The most interesting instance of this definition is the image functor. Consider (limit, colimit or mixed) complexity $C_{\mathcal{C},\mathcal{A}}$ as a function on the objects of the morphism category $\mathcal{C}^{\bullet\to\bullet}$.

In the diagram category $\mathcal{C}^{\bullet\to\bullet}$, let $\mathrm{Mon}_{\mathcal{C}}$ be the full subcategory of monomorphisms. Let $\mathrm{i}_{\mathcal{C}} : \mathrm{Mon}_{\mathcal{C}} \to \mathcal{C}^{\bullet\to\bullet}$ be the inclusion functor. We say that the category $\mathcal{C}$ *has images* if there is a left-adjoint $\mathrm{im}_{\mathcal{C}}$ to $\mathrm{i}_{\mathcal{C}}$,

$$\mathrm{Mon}_{\mathcal{C}} \underset{\mathrm{i}_{\mathcal{C}}}{\overset{\mathrm{im}_{\mathcal{C}}}{\rightleftarrows}} \mathcal{C}^{\bullet\to\bullet}\ .$$

It is a natural question to ask: how does complexity behave under the image functor? We consider the following as the analogue of the P vs NP problem for a category $\mathcal{C}$.

Question 5.4 (Complexity of Images in $\mathcal{C}$)**.** *Given a sequence $(X_n \xrightarrow{f_n} Y_n)$ of diagrams in $\mathcal{C}$ whose complexities are polynomially bounded in n, is the complexity of $\mathrm{i}_{\mathcal{C}} \circ \mathrm{im}_{\mathcal{C}}(X_n \xrightarrow{f_n} Y_n)$ polynomially bounded in n?*

Natural places where this question makes sense are the category of projective varieties and the category of projective schemes. These categories are considered as subcategories of the categories of all varieties and schemes respectively and inherit the (mixed) complexity functions from them. Another important possibility is the discussion of the complexity of semi-algebraic spaces over $\mathbb{R}$ or constructible spaces over $\mathbb{C}$. In these cases, we must also consider mixed complexity.

5.3. **The Image Functor in $k[x_1, \ldots, x_n]$-Mod.** We now consider Question 5.4 for the union of the the categories of modules over polynomial rings discussed above in Section 4.3; and look for an upper bound on the complexity of the image functor. Let $R = k[x_1, \ldots x_n]$ with the basic morphisms defined in that section.

We describe a method for writing a colimit computation to compute the image of morphism of modules. A careful analysis of the Gröbner basis portion of the proof below would give an upper bound for the complexity of the image functor in R-**Mod**. With a naive analysis, we can only obtain an unnecessarily high bound, so we leave the complexity analysis out of the following statement.

Proposition 5.5. *Let $M \xrightarrow{\varphi} N$ be a morphism diagram in R-**Mod**, computed by a colimit computation of size s. Then, there is a colimit computation that computes $\mathrm{im}(\varphi) \to N$.*

Proof. By Lemma 2.16 and Remark 2.18, the colimit computation that produces $M \xrightarrow{\varphi} N$ can be simplified to produce (only) $M \xrightarrow{\varphi} N$ in two colimit steps. The first one is taking the colimit of diagram D_M of basic morphisms which produces M. The second one is the colimit of a larger diagram $D_N \supset D_M$ together with M and all the cocone morphisms from the objects in D_M to M. Replacing colimits with co-equalizers and coproducts, we have that M is isomorphic to the quotient

$$\bigoplus_{\rho \in \mathrm{s}(D_M)} R^{j_\rho} \xrightarrow{A_M} \bigoplus_{\gamma \in \mathrm{v}(D_M)} R^{j_\gamma} \longrightarrow M,$$

where γ runs over all the vertices in D_M and ρ runs over all sources of arrows in D_M; and j_ρ and j_γ are 0, 1 or 2, based on whether the corresponding basic object is $\{0\}$, R, or R^2. Similarly N we can write N as a quotient,

$$\bigoplus_{\rho \in \mathrm{s}(D_N)} R^{j_\rho} \oplus \bigoplus_{v \in \mathrm{v}(D_M)} R^{j_v} \longrightarrow \bigoplus_{\gamma \in \mathrm{v}(D_N)} R^{j_\gamma} \oplus M \longrightarrow N\ . \tag{5.1}$$

Since M is the colimit of D_M, we can remove the extra sum on the left and the M summand in middle. So N is the quotient:

$$\bigoplus_{\rho\in \mathrm{s}(D_N)} R^{j_\rho} \xrightarrow{A} \bigoplus_{\gamma\in \mathrm{v}(D_N)} R^{j_\gamma} \longrightarrow N$$

Combining the direct sums, we get a commuting diagram:

$$\begin{array}{ccccc} R^{m_2} & \xrightarrow{A_M} & R^{m_1} & \longrightarrow & M \\ \downarrow{\scriptstyle i_2} & & \downarrow{\scriptstyle i_1} & & \downarrow{\scriptstyle \varphi} \\ R^{n_2} & \xrightarrow{A} & R^{n_1} & \longrightarrow & N \end{array}$$

Where the maps i_1 and i_2 are the inclusion of the first m_1 and m_2 coordinate spaces respectively. Written this way, the map φ from M to N is induced by the inclusion of the generators of M into the generators of N.

To construct the image $\mathrm{im}(\varphi)$, we need to find the relations among the images of the generators of M, i.e. the first m_1 generators of N. To do this, let A' be an $(n_1 - m_1) \times n_2$ matrix consisting of the the lower $n_1 - m_1$ rows of A. We will use use Gröbner basis methods to find generators for the kernel of A', i.e. syzygies for the columns of A', and apply these syzygies to the full columns of A to get the relations among the generators of M mapped to N.

More precisely, consider the map $R^{n_2} \xrightarrow{A'} R^{n_1-m_1}$. Using the position over lexicographical ordering (or another ordering) on monomial times basis elements in R^n, extend the columns $\{f_1, \ldots, f_{n_2}\}$ of A' into a Gröbner basis $\{f_1, \ldots, f_s\}$, $f_i \in R^{n_1-m_1}$, for the image of A'. Let $R^s \xrightarrow{B} R^{n_2}$ be the map sending the standard basis element e_i to the vector $(a_{i1}, \ldots, a_{in_2})$ of coefficients so that $f_i = \sum_{j=1}^{n_2} a_{ij} f_j$. Since the f_i form a Gröbner basis, we have, say s', relations of the form

$$m_{ji} f_i - m_{ij} f_j - \sum_{u=1}^{s} g_u^{(ij)} f_u = 0$$

where the m_{ij} and m_{ji} are the smallest monomials set so that the top order terms of the first two terms cancel each other, and where the $g_u^{(ij)}$ are computed using the division algorithm. If we store the coefficients of the $f's$ for each relation in a vector in R^s, then, the image of the corresponding map $R^{s'} \xrightarrow{Z} R^s$ is the kernel of the map $R^s \xrightarrow{A'B} R^{n_1-m_1}$, c.f. [Sch80, Sch91]. Moreover, the image of the map $R^{s'} \xrightarrow{BZ} R^{n_2}$ is the kernel of the map, c.f. [Wal89, Buc85].

Now take $R^{s'} \xrightarrow{ABZ} R^{n_1}$. Since the image of BZ was the kernel of A' which was the lower portion of the matrix A, we have that the image of ABZ is the intersection of the image of A with R^{m_1} sitting in the first m_1 coordinates. Hence we have a map $R^{s'} \xrightarrow{ABZ} R^{m_1}$ whose image is the submodule of relations among images of the generators of M mapped in N by φ.

Finally, construct $\mathrm{im}(\varphi) \xhookrightarrow{\iota} N$ in a colimit computation as follows. Construct

$$0 \leftarrow R^{s'} \xrightarrow{ABZ} R^{m_1}$$

and take the colimit to get $\mathrm{im}(\varphi)$; then, construct N again by taking the basic object R's in the computation of $\mathrm{im}(\varphi)$ corresponding to the generators of M and constructing any remaining generators and relations for N from 5.1 above. □

References

[Awo10] Steve Awodey, *Category theory*, Oxford University Press, 2010. 9

[Bas15] Saugata Basu, *A complexity theory of constructible functions and sheaves*, Foundations of Computational Mathematics **15** (2015), no. 1, 199–279. 1, 3

[BCS97] Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi, *Algebraic complexity theory*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 315, Springer-Verlag, Berlin, 1997, With the collaboration of Thomas Lickteig. MR 1440179 3

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and real computation*, Springer-Verlag, New York, 1998, With a foreword by Richard M. Karp. MR 1479636 (99a:68070) 1

[Buc85] Bruno Buchberger, *Grobner bases: An algorithmic method in polynomial ideal theory*, Multidimensional systems theory (1985), 184–232. 22

[Bür00] Peter Bürgisser, *Completeness and reduction in algebraic complexity theory*, Algorithms and Computation in Mathematics, vol. 7, Springer-Verlag, Berlin, 2000. MR 1771845 3

[Imm95] N. Immerman, *Descriptive complexity: a logician's approach to computation*, Notices Amer. Math. Soc. **42** (1995), no. 10, 1127–1133. MR 1350010 2

[Isi16] M Umut Isik, *Complexity classes and completeness in algebraic geometry*, arXiv preprint arXiv:1609.02562 (2016). 3

[Kal88] Erich Kaltofen, *Greatest common divisors of polynomials given by straight-line programs*, Journal of the ACM (JACM) **35** (1988), no. 1, 231–264. 19

[LS88] J. Lambek and P. J. Scott, *Introduction to higher order categorical logic*, Cambridge Studies in Advanced Mathematics, vol. 7, Cambridge University Press, Cambridge, 1988, Reprint of the 1986 original. MR 939612 2

[MLM94] Saunders Mac Lane and Ieke Moerdijk, *Sheaves in geometry and logic*, Universitext, Springer-Verlag, New York, 1994, A first introduction to topos theory, Corrected reprint of the 1992 edition. MR 1300636 2

[Poi95] Bruno Poizat, *Les petits cailloux*, Nur al-Mantiq wal-Ma'rifah [Light of Logic and Knowledge], 3, Aléas, Lyon, 1995, Une approche modèle-théorique de l'algorithmie. [A model-theoretic approach to algorithms]. MR 1333892 1

[Sch80] Frank-Olaf Schreyer, *Die berechnung von syzygien mit dem verallgemeinerten weierstraßschen divisionssatz und eine anwendung auf analytische cohen-macaulay stellenalgebren minimaler multiplizität*, Ph.D. thesis, 1980. 22

[Sch91] ———, *A standard basis approach to syzygies of canonical curves*, J. reine angew. Math **421** (1991), 83–123. 22

[Str73] Volker Strassen, *Vermeidung von divisionen.*, Journal für die reine und angewandte Mathematik **264** (1973), 184–202. 19

[Val79a] Leslie G Valiant, *Completeness classes in algebra*, Proceedings of the eleventh annual ACM symposium on Theory of computing, ACM, 1979, pp. 249–261. 1

[Val79b] ———, *The complexity of computing the permanent*, Theoretical computer science **8** (1979), no. 2, 189–201. 1

[vzG87] Joachim von zur Gathen, *Feasible arithmetic computations: Valiant's hypothesis*, Journal of Symbolic Computation **4** (1987), no. 2, 137–172. 1

[Wal89] Bernhard Wall, *On the computation of syzygies*, ACM SIGSAM Bulletin **23** (1989), no. 4, 5–14. 22

Department of Mathematics, Purdue University, West Lafayette, IN 47906, U.S.A.
E-mail address: sbasu@math.purdue.edu

Department of Mathematics, University of California, Irvine, Irvine, CA 92697