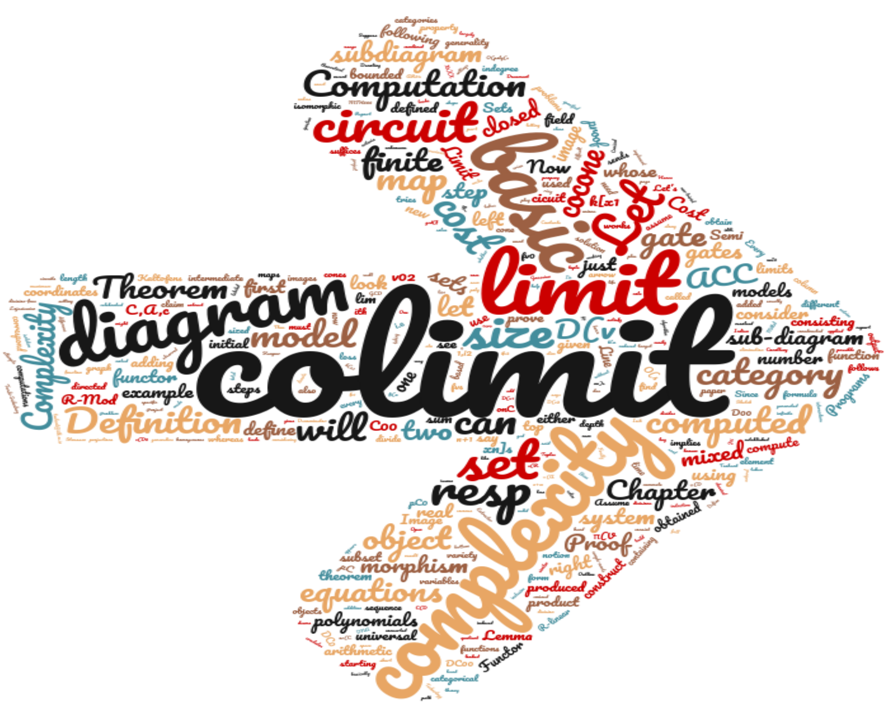
Categorical Complexity

MTH666 : Project Report



Tushant Mittal
Indian Institute of Technology, Kanpur
tushant@iitk.ac.in

Contents

Chapter 1

Introduction

The theory of computational complexity tries to classify problems based on their *complexity* which is the difficulty of solving it using a certain computation model. There are variety of models each of which arise naturally in different settings. For example, the turing machine model is typically used for problems with finite representations but the *real RAM* model is more suited in instances where operations involve exact real numbers. It must be noted, that the computation models are a theoretical construct and might not correspond to real life devices, like say, a quantum computation and DNA compution model.

Each model further tries to analyse the usage of some resource which it deems relevant. For example, in the boolean model time and space are looked at, the cicuit model works with the number of gates in the circuit whereas in the field of communication complexity the total length of messages sent is what matters.

1.1 Attempt at Unification

These various models though have correlations amongst each other are largely studied in isolation depending on the setting.

A recent paper by Basu, Isik [BI17] attempts to genralize the notion of complexity by defining a new notion of categorical complexity. The claim is then that different models can be recovered by working in appropriate categories.

This report is entirely based on this paper and presents the key results of the paper.

Chapter 2

Diagram Computation

Let us now define the categorical model of compuatation which essentialy looks at the the number of steps required to construct a diagram in a category starting from a set of basic morphisms and adding a limit (colimit) of subdiagrams at each step.

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

1. *D_0 consists only of morphisms in A i.e. the basic morphisms*
2. *For each $i = 1, \cdots, s$, 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 $D_{i-1}|J_i$, where $J_i \subset I_i$, of D_{i-1} and L_i is its limit (resp. Ci is colimit) 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).*
3. *(Constructivity) If a limit $L_i = lim \ \ D_{i-1}|J_i$ (resp., colimit C_i) produced in the i^{th} step of the computation is used again in the subdiagram $D_{j-1}|J_j$ used at the j^{th} 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 $D_{j-1}|J_j$*

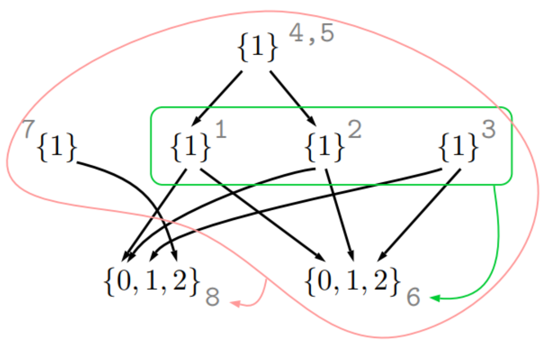
$\diamondsuit$

Definition 2.2. *A mixed computation is similar to the above sonstructuction but D_i is obtained from D_{i-1} by adding either the limit or the colimit of a subdiagram. Moreover, the constructivity condition is dropped.* $\diamondsuit$

The computation $(D_0, \cdots, D_s)$ is said to compute a diagram D, if D is isomorphic to subdiagram of D_s. In particular, an object in $\mathcal{C}$ is computed by $(D_0, \cdots, D_s)$ if an object isomorphic to it appears in D_s.

2.1 A basic example

Let us look at a few concrete examples to understand how the compuation works and to see how the given conditions play a role. We construct the function $f : \{0, 1, 2\} \to \{0, 1, 2\}$ such that $f(0) = 0 = f(1), f(2) = 1$ using colimits.



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)

2.2 Cost of Computation

Let $c_0 : A \to \mathbb{N}$ be a function that assigns a cost to eacho of the basic morphisms.

Definition 2.3 (Cost Computation)**.** *The cost of the computation $(D_0, ..., D_s)$ is the the number of steps plus the cost of the initial diagram D_0 consisting of basic morphisms, that is:*

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

$\diamondsuit$

The cost will be taken to be 1 unless specified. Using this we define the limit/ colimit/ mixed complexity of a diagram D.

Definition 2.4. *The limit (resp. colimit, resp. mixed) complexity $\mathcal{C}(D) = \mathcal{C}_{\mathcal{C},A}(D)$, short for$\mathcal{C}^{lim}_{\mathcal{C},A,c}(D)(resp., \mathcal{C}^{colim}_{\mathcal{C},A,c}(D), resp., \mathcal{C}^{mixed}_{\mathcal{C},A,c}(D))$, of a diagram D in a category $\mathcal{C}$ is the cost* *of the limit (resp., colimit, resp. mixed) computation using basic morphisms A,that has the smallest cost among all such computations that compute D.* ◊

Example Let's calculate the colimit complexity of constructing a finite set starting from just id_1.

Lemma 2.5. *In the category Set, let*

$$A = \{id : \{1\} \to \{1\}\}, c_0(id) = 1$$

. Then, for any set finite set S,

$$c^{colim}_{Set,A}(S) = |S| + 1$$

Proof. Since finite sets of equal size are isomorphic, a computation will compute S if and only if it computes any set of cardinality equal to $|S|$. As in the earlier example, starting with $|S|$ copies of $\{1\}$ and taking their colimit, we get a set of cardinality $|S|$. So, the complexity is bounded from above by $|S| + 1$. To see that this is the most efficient way of producing a set with $|S|$ elements, we use a theorem that is proven in the next chapter, 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. □

Chapter 3

A Useful Theorem

The following lemma, shows that to construct an object, it suffices to just consider the basic morphisms and that the intermediate steps in a limit or colimit computation are unnecessary. The key here is the constructivity assumption.

Lemma 3.1. *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 $(D_0, \cdots, D_s)$ 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 i^{th} step. Let $J_i' = I_0 \cup J_i$. 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 $L_i \cong D_{i-1}|J_i'$. 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. □

Chapter 4

Simulating ACC

4.1 Arithmetic Circuit Complexity

An arithmetic circuit C over a field $\mathbb{F}$ and the set of variables $x_1, \cdots, x_n$ is a directed acyclic graph as follows. Every node in it with indegree zero is called an input gate and is labeled by either a variable x_i or a field element. Every other gate is either a sum $(+)$ or a product $(\times)$ gate. A circuit has two complexity measures associated with it: size and depth. The size of a circuit is the number of gates in it, and the depth of a circuit is the length of the longest directed path in it. The arithmetic circuit complexity of a polynomial f is the least size of a circuit computing it.

4.2 ACC and R-Modules

Define R to be the polynomial ring $k[x_1, \cdots, x_n]$. We will look at the colimit computations in $R-Mod$ which is the category of modules over R with the basic of morphisms A containing:

$$R \xrightarrow{x_i} R \;\; i \in [1, n]$$

$$R \xrightarrow{c} R \;\; c \;\in k$$

$$R \xrightarrow{\Delta} R \oplus R$$

$$R \xrightarrow{i_1, i_2} R \oplus R$$

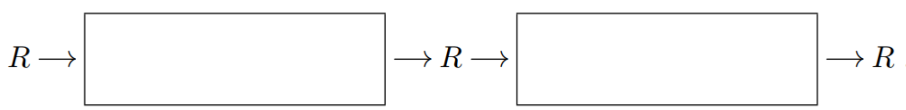
$$R \oplus R \xrightarrow{+} R$$

$$R \to \{0\}$$

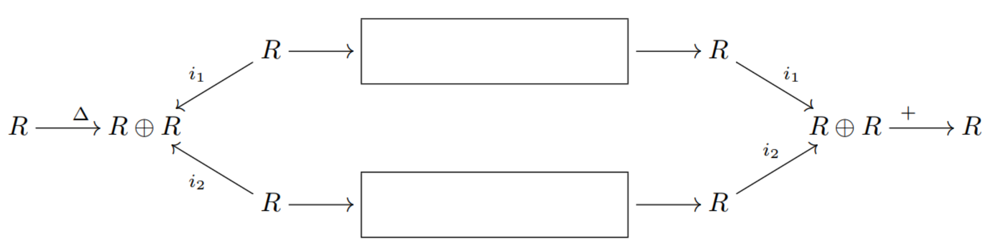
Theorem 4.1. *If a polynomial $f \in R$ is computed by a formula of size s, then the diagram $R \xrightarrow{f} R$ is computed by a colimit computation in R-Mod with cost bounded by $O(s)$.*

Proof. Without loss of generality, we can assume that all sum and product gates have two indegree 2 because such a restriction only increases the size by a polynomial factor. 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

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 $1 \to p_C$.If the output p_C of C is one of the variables x_i let D_C be the diagram R $x_i \to R$. 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 $p_{C'} + p_{C'}$. ∎

Now we prove the converse and show that the existence of a colimit computation in R-Mod producing say about the complexity of f?

Theorem 4.2. *If $R \xrightarrow{f} R$ is computed in a colimit computation with cost c in R-Mod, then there is an arithmetic circuit of size poly(c) with inputs $x_1, \cdots, x_n$ that computes f*

Proof. Consider a diagram $D : I \to R{-}Mod$ consisting only of the basic morphisms. Assume that we have colim $D = R^*$. For each $v \in ob(I)$, 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 $R \xrightarrow{1 \to f_v} R^*$ 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 $(1, 0) \to f_v$ and $(1, 0) \to f_{v'}$. We will prove that each f_v is computed by a polynomially sized circuit. We are considering the f_vs 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:

$$R \xrightarrow{x_i} R \quad f_{v_1} - x_i f_{v_2}$$
$$R \xrightarrow{c} R \quad f_{v_1} - c f_{v_2}$$
$$R \xrightarrow{i_1, i_2} R \oplus R \quad f_{v_1} - f_{v_2} \text{ or } f_{v_1} - f_{v_2'}$$
$$R \xrightarrow{\Delta} R \oplus R \quad f_{v_1} - f_{v_2} - f_{v_2'}$$
$$R \oplus R \xrightarrow{+} R \quad f_{v_1} - f_{v_2} \text{ and } f_{v_1} - f_{v_2'}$$
$$R \to \{0\} \quad f_{v_1} = f_{v_2} = 0$$

In this way, we obtain a homogeneous system R-linear equations; $Af = 0, A \in Mat_{n \times s}(R)$. Tuples 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 $f_{v_1}, \cdots, f_{v_s})$, by the universal propery there will be a map $R \xrightarrow{1 \to g} R*$ making the diagram containing the new cocone, the colimit cocone and the map $R \xrightarrow{1 \to g} R$ 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 $(\frac{f_{v_1}}{h}, \cdots, \frac{f_{v_s}}{h})$ where $h = gcd(f_{v_1}, \cdots f_{v_s})$. Thus, to compute the map from every $D(v)$ to R^*, it suffices to: (i) find a solution to the above system of equations for D, and (ii) divide by h.

It can be easily seen that Gaussian elimination gives a poly time cicuit (using divisions) to solve the system of equations.

But there are 2 issues to be resolved here. One, the solution obtained $say, S = (\frac{p_1}{q_1}, \cdots, \frac{p_s}{q_s})$

lies in $k(x_1, \cdots, x_n)^s$ whereas we want one in $k[x_1, \cdots, x_n]^s$.

To do this, first use Kaltofens GCD algorithm [Kal88] to assume, without loss of generality that each p_i, q_i is reduced. Then use Kaltofens Denominator Extractor [Kal88] to extract the denominators q_i from each fraction. The element $\prod_i q_i S \in k[x_1, \cdots, x_n]^s$. Now divide $\prod_i q_i S$ by the gcd of all of its entries to obtain $(f_{v_1}, \cdots, f_{v_s}) \in k[x_1, \cdots, x_n]^s$.

The other is that we need to implement these in a division-free circuit. This is made possible due to a classical result by Strassen [Str73] which says that any cicuit of size $O(s)$ which uses division gates can be converted to one of size $O(poly(s))$ which has just addition and multiplication gates.

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 $R_1 \xrightarrow{1 \to f} R^*$ be a sub-diagram of a colimit computation with initial step D_0. By Lemma 3.1, there is a sub-diagram $D_0' \subset D_0$ of basic morphisms whose colimit is R_1; and from the constructivity criteria there is a subdiagram $D_0' \subset D_0'' \subset D_0$ whose colimit is R^*, with the induced map $R_1 \to R^*$ being a map that sends $1 \to f$. 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.

□

Chapter 5

Image Functor and P vs NP

Definition 5.1. *A complexity function on $\mathcal{C}$ is a function that takes (finite) diagrams of C to $\mathbb{N} \cup \{\infty\}$* ◊

Definition 5.2. *Let $\mathcal{C}, \mathcal{D}$ be two categories with complexity functions, ϕ, ψ, and let $F : \mathcal{C} \to \mathcal{D}$ be a functor. We define the complexity, $c_{\phi,\psi}(F) : \mathbb{N} \to \mathbb{N}$ by*

$$\mathcal{C}_{\phi,\psi}(F)(n) = sup\{\psi(F(D)) \mid I is\ a\ finite\ shape, D \in [I, \mathcal{C}],\ \phi(D) \leq n\}$$

◊

Lemma 5.3. *Suppose that $\mathcal{C}$ is a category that has pull-backs and images. Then in $\mathcal{C}^{\to}$, letting $Mon_{\mathcal{C}}$ denote full subcategory of monomorphisms, and $i_{\mathcal{C}} : Mon_{\mathcal{C}} \to \mathcal{C}^{\to}$ the inclusion functor has a left-adjoint .*

If $\mathcal{C}$ has pullbacks, we have discussed in class if $\mathcal{C}$ has images then as

The complexity of a functor F is basically the maximum complexity of the image of a diagram of complexity at most n under F.

5.1 Semi Algebraic Sets

Definition 5.4. *The basic (closed) semialgebraic set defined by polynomials $f_1, \cdots, f_n$ is*

$$\{\ x \in \mathbb{R}^m \mid f_i(x) \geq 0\ \forall i \in [1, n]\ \}$$

◊

Definition 5.5. *A set generated by a finite sequence of unions, intersections and complements on basic semialgebraic sets is called a semialgebraic set.* ♢

Let's look at the specific category SA of the semi-algebraic sets. The category has as its objects semi-algebraic sets and the morphisms are polynomial maps. That it is a valid morphism is non-trivial and follows from the following celebrated theorem.

Theorem 5.6 (Tarski-Seidenberg)**.** *Let A be a semialgebraic subset of $\mathbb{R}^{n+1}$ and $\pi : \mathbb{R}^{n+1} \to \mathbb{R}^n$, the projection on the first n coordinates. Then $\pi(A)$ is a semialgebraic subset of $\mathbb{R}^n$.*

Corollary 5.7. *Let A be a semialgebraic subset of $\mathbb{R}^n$ and $F : \mathbb{R}^n \to \mathbb{R}^m$. Then $F(A)$ is a semialgebraic.*

Proof Sketch. Firstly, we can use induction to generalize the theorem to $\mathbb{R}^{n+m} \to \mathbb{R}^n$. Now, consider the graph of F $\{(a, F(a)) \mid a \in A\}$ and then project it. □

Now that we have established the category of semialgebraic sets let us look at the limit complexity of the image functor of SA.

5.1.1 Limit Complexity in SA

Let the basic morphisms A consist of the following morphisms

$$\mathbb{R} \xrightarrow{c} \mathbb{R} \quad c \in \mathbb{R}$$
$$\mathbb{R}^2 \xrightarrow{+} \mathbb{R}$$
$$\mathbb{R}^2 \xrightarrow{\times} \mathbb{R}$$
$$[0, \infty) \hookrightarrow \mathbb{R}$$
$$\mathbb{R} \to \{0\}$$

Theorem 5.8. *$\mathcal{C}_{c^{lim}_{SA^{\to},A}}(im_{SA})$ is not polynomially bounded.*

Proof. It is not difficult to see that the objects of SA that can be constructed using a limit computation are exactly the basic closed semi-algebraic sets. On the other hand, it is well known that the image under polynomial maps (for example, projections along some coordinates) of a basic closed semi-algebraic set need not be a basic closed semi-algebraic set. For example, consider the real variety V defined by

$$(X_1 - X_3^2)(X_2 - X_4^2) = 0$$

Denoting by $\pi : \mathbb{R}^4 \to \mathbb{R}^2$ the projection to first 2 coordinates, $\pi(V) = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 \geq 0 \vee x_2 \geq 0\}$ which is not a basic closed semi-algebraic set (as observed by Lojasiewicz, see [AR94]), and hence $\pi(V)$ has infinite limit complexity. $\square$

Similarly we can look at the the category of semilinear sets with affine maps as morphisms and it turns out that again, $\mathcal{C}_{c^{lim}_{SL^{\cdot \to \cdot},A}}(im_{SL})$ is not polynomially bounded. Proof can be found here [BI17]

This leads us to the natural question i.e. whether having mixed limits can help us?

Open Problem 5.1 [Categorical P vs NP]

Are the functions

$$\mathcal{C}_{c^{mixed}_{SL^{\cdot \to \cdot},A}}(im_{SL}), \quad \mathcal{C}_{c^{mixed}_{SA^{\cdot \to \cdot},A}}(im_{SA})$$

polynomially bounded ?

The paper claims that this is a categorical analogue of the famous P vs NP question.

Not just this we can look at the image functor in a variety of categories and maybe even draw up simialar analogues for VP vs VNP or other such related questions.

Bibliography

[AR94] Carlos Andradas and Jess M. Ruiz. Ubiquity of ojasiewicz's example of a nonbasic semialgebraic set. *Michigan Math. J.*, 41(3):465–472, 1994.

[BI17] Saugata Basu and M. Umut Isik. Categorical Complexity. 2017. arXiv, Preprint.

[Kal88] Erich Kaltofen. Greatest Common Divisors of Polynomials Given by Straight-line Programs. *J. ACM*, 35(1):231–264, jan 1988.

[Str73] Volker Strassen. Vermeidung von Divisionen. *Journal fr die reine und angewandte Mathematik*, 264:184–202, 1973.