A Categorical Approach to Descriptive Complexity Theory

Sujet de thèse

Damiano Mazza
CNRS, LIPN, Université Sorbonne Paris Nord

Background

Decision problems. A *decision problem* is a computational question which has a yes/no answer. "Computational" means that the question is about some structure which may be fed as input to a computer program. In particular, such a structure must be finite, albeit arbitrarily large. Examples of decision problems are: the SUBSTRING problem: given two strings s, t, does t occur as substring in s? The CLIQUE problem: Given a graph G and an integer n, does G have n nodes which are all connected with one another?

Answering such yes/no questions has become, perhaps unknowingly at times, part of every day life. For instance, asking our computer whether the word "serendipitous" is found in the present text means solving an instance of the first decision problem mentioned above. Wondering whether there are 100 people on Facebook who are all friends with each other is an instance of the second example of decision problem.

Computational complexity theory. Born in the mid-1960s with the work of Juris Hartmanis and Richard Stearns, complexity theory aims at understanding the *difficulty* of decision problems. The basic observation is that the computational resources (running time, memory space) needed to solve a decision problem usually increase with the size of the instance: for example, it is reasonable to expect a computer program to take less time to tell whether the word "serendipitous" occurs in the present text than in the Bible. So, the resources needed grow as we increase the size of instances; the quicker the resources grow, the more difficult the problem is deemed to be.

Turing machines are the most commonly used model of computation to speak of computational resources. Since Turing machines work on strings, the standard complexity-theoretic definition of decision problem has become simply a subset of $\{0,1\}^*$. Indeed, a Turing machine M working on the smallest non-trivial alphabet $\{0,1\}$ defines a subset of $\{0,1\}^*$, namely the set of strings accepted by M. One may then classify subsets of $\{0,1\}^*$ according to the computational resources required by the machines accepting them. That is, roughly speaking, the complexity of a decision problem is the complexity of the "best" Turing machine solving it.

Classifying decision problems according to their complexity has proved to be a formidably difficult task. To this day, an extremely large number of basic complexity-theoretic questions remain open. For example, it is unknown whether there exists a polynomial-time deterministic Turing machine solving the CLIQUE problem introduced above. Contemporary algorithmic techniques fall way short of achieving such an efficiency (they are exponential at best). At the

same time, proving that no such machine exists seems to be far out of reach of the current theoretical tools.

Descriptive complexity. Confronted with such an imposing barrier, researchers have naturally turned to explore alternative views of complexity theory, in the hope of shedding some light on the field. A particularly well-developed approach is given by *descriptive complexity* [Imm99].

Born with the work of Ronald Fagin, descriptive complexity starts with the observation that any structure that may be given as input to a computer program is a *finite structure* in the sense of model theory. For example, a finite directed graph is just a finite structure on the first-order language $\mathbb{G}\mathrm{rph}$ with exactly one binary relation symbol. A decision problem thus becomes a subset of finite structures of some first-order language.

The key idea of descriptive complexity is that a logical sentence φ defines such a subset, namely the set of finite models of φ, so one may understand the complexity of problems via the expressiveness of the logic needed to define them (*i.e.*, the logic in which φ is formulated). In other words, we determine the difficulty of a decision problem not by how hard it is to solve it but by *how hard it is to describe it*.

Rather than insisting on the multitude of logical characterizations of complexity classes that were achieved following the descriptive complexity approach, this proposal explores a slightly different path, which we proceed to illustrate.

The categorical viewpoint. First of all, instead of structures, one may more generally consider finite models of first-order theories, up to isomorphism: for example a binary string is, up to iso, a finite model of the theory $\mathbb{S}\mathrm{tr}$ whose language contains a binary relation symbol $\leq$ and a unary relation symbol isOne, and whose axioms state that $\leq$ is a total order. "Up to iso" here means that the structures $\{0 < 1 < 2\}$ with $\mathsf{isOne} = \{2\}$ and $\{a < b < c\}$ with $\mathsf{isOne} = \{c\}$ define the same binary string, namely 001.

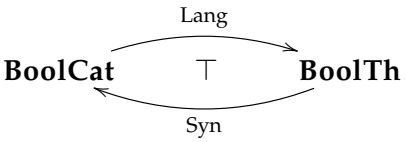
We said "first-order theory" but, for reasons which will be clear later, we actually restrict to what we call *Boolean theories*, which are multisorted relational theories (*i.e.*, no function symbols) with equality whose axioms are all of the form $\forall \vec{x}.\varphi$, where φ contains no quantifiers except *provably unique* existential quantifiers, *i.e.*, of the form $\exists y.\psi$ where the formula $\forall y.\forall y'.\psi(y) \wedge \psi(y') \Rightarrow y = y'$ is provable in the theory.

If finite models of Boolean theories are our instances, it is natural to look for a suitable notion of *transformation* between such instances. In order to get a compositional definition, we turn to categorical logic [Joh02]. Initiated by William Lawvere, categorical logic posits that logical and categorical structures come hand in hand: logical theories of a given kind (especially subsystems of first-order logic) correspond to categorical structures necessary to interpret them.

The categorical structure corresponding to Boolean theories is that of a *Boolean category* [CLW93], which are categories with finite products, disjoint and pullback-stable finite coproducts, and such that the poset of subobjects of every object is a Boolean algebra. Morphisms between Boolean categories, called *logical functors*, are functors preserving finite products and finite coproducts.

The category **BoolTh** of Boolean theories and their morphisms and the category **BoolCat**

of Boolean categories and logical functors are related by an adjunction



The functor Lang associates with a category its *internal language*, whereas the functor Syn associates with a theory its *syntactic category*. Intuitively, $\mathrm{Syn}(\mathbb{T})$ is the category presented by the theory $\mathbb{T}$, much like $\mathbb{Z}[X_1,\ldots,X_n]/(P_1,\ldots,P_m)$ is the commutative ring presented by n generators and m polynomial equations $P_1,\ldots,P_m \in \mathbb{Z}[X_1,\ldots,X_n]$ (the generators correspond to the sorts and symbols of $\mathbb{T}$, whereas the polynomials to its axioms).

A fundamental observation now is that the category $\mathcal{F}in$ of finite sets and functions is Boolean, and finite models of a theory $\mathbb{T}$ are in bijection with natural isomorphism classes of logical functors $\mathrm{Syn}(\mathbb{T}) \to \mathcal{F}in$. Essentially, this was Lawvere's starting point for his development of categorical logic.

We therefore define $\mathcal{B}ool$ to be the category whose objects are finite Boolean theories (meaning with finitely many sorts, relation symbols and axioms) and whose morphisms $\mathbb{T} \to \mathbb{S}$ are logical functors $\mathrm{Syn}(\mathbb{T}) \to \mathrm{Syn}(\mathbb{S})$, modulo natural isomorphism.

The category $\mathcal{B}ool$ is essentially small, has all finite colimits (with the empty theory $\mathbb{E}$ being initial) as well as finite products (the inconsistent theory being the terminal object). Moreover, it turns out that $\mathrm{Syn}(\mathbb{E})$ is equivalent to $\mathcal{F}in$, which implies, by what mentioned above, that a morphism $\mathbb{T} \to \mathbb{E}$ in $\mathcal{B}ool$ is the same thing as a finite model of $\mathbb{T}$ (remember that morphisms are up to iso). Hence, a morphism $f : \mathbb{T} \to \mathbb{S}$ of $\mathcal{B}ool$ induces a function from the finite models of $\mathbb{S}$ to the finite models of $\mathbb{T}$ (notice the contravariance): simply take the image of f via the presheaf $\mathcal{B}ool(-,\mathbb{E})$.

The fact that arrows in $\mathcal{B}ool$ go in the opposite direction with respect to how models are transformed suggests that all arrows should be reversed. Let us then define the category of *data specifications* as $\mathcal{D}ata := \mathcal{B}ool^{\mathrm{op}}$. When seeing a theory $\mathbb{T}$ as a data specification, *i.e.*, as an object of $\mathcal{D}ata$ rather than $\mathcal{B}ool$, we will write $\mathrm{Spec}\,\mathbb{T}$.

It turns out that a morphism $\mathrm{Spec}\,\mathbb{S} \to \mathrm{Spec}\,\mathbb{T}$ is, essentially, what is known in descriptive complexity as a *quantifier-free query* from the finite structures on $\mathbb{S}$ to the finite structures on $\mathbb{T}$. As for the finite models of $\mathbb{T}$, they become morphisms $\mathrm{Spec}\,\mathbb{E} \to \mathrm{Spec}\,\mathbb{T}$ in $\mathcal{D}ata$, which is reasonable because $\mathrm{Spec}\,\mathbb{E}$ is the terminal object of $\mathcal{D}ata$, and therefore these are, in categorical jargon, the "points" of $\mathrm{Spec}\,\mathbb{T}$, which we will call *finite points* here.

With this perspective, given a morphism $f : \mathrm{Spec}\,\mathbb{T} \to \mathrm{Spec}\,\mathbb{S}$ and a finite point $x : \mathrm{Spec}\,\mathbb{E} \to \mathrm{Spec}\,\mathbb{S}$, we may ask when x factors through f via a finite point $y : \mathrm{Spec}\,\mathbb{E} \to \mathrm{Spec}\,\mathbb{T}$: this corresponds to seeing f as a sort of generalized problem and y as a "solution" for x. We may say that x is in the *finite image* of f if it factors through some y as above.

The category $\mathcal{D}ata$ is related to computability, as follows. Call a morphism $f : \mathrm{Spec}\,\mathbb{T} \to \mathrm{Spec}\,\mathbb{S}$ of *finite presentation* if $\mathbb{T}$ is an extension of $\mathbb{S}$, in the sense that $\mathbb{T}$ is obtained by adding sorts and/or relation symbols and/or axioms to $\mathbb{S}$, and f corresponds to the inclusion of $\mathbb{S}$ in $\mathbb{T}$ in $\mathcal{B}ool$. We then have:

Theorem 1 *A subset of* $\{0,1\}^*$ *is recursively enumerable iff it is the finite image of a morphism of finite presentation over* $\mathrm{Spec}\,\mathbb{S}\mathrm{tr}$.[1]

[1]Notice the similarity with the MRDP theorem (Matiyasevich, Robinson, Davis and Putnam): when formulated

The result generalizes to any finite data type: for instance, if G is the set of finite directed graphs, finitely presented theories over $\mathbb{G}\mathrm{rph}$ define exactly the recursively enumerable subsets of G.

A first link with complexity is given as follows. Call a morphism of finite presentation $f : \operatorname{Spec}\mathbb{T} \to \operatorname{Spec}\mathbb{S}$ *relational* if $\mathbb{T}$ has the same sorts as $\mathbb{S}$ (in other words, $\mathbb{T}$ extends $\mathbb{S}$ by possibly adding relation symbols and axioms, but not sorts). We thus obtain a characterization of NP (the class of problems solvable in polynomial time by a non-deterministic Turing machine):

Theorem 2 *A subset of* $\{0,1\}^*$ *is in* NP *iff it is the finite image of a relational morphism over* $\operatorname{Spec}\mathbb{S}\mathrm{tr}$.

As Theorem 1, this too generalizes to arbitrary finite data types. The interesting fact about Theorem 2 is that it does not use Fagin's descriptive characterization of NP but, rather, it uses Theorem 1 and yields Fagin's theorem as a corollary.

Objectives and Methodology

Objectives. Broadly speaking, the objective of this thesis is to develop as much as possible the categorical approach to descriptive complexity sketched above. In particular, we will address the following questions:

1. which subclasses of NP may be characterized by means of subclasses of relational morphisms of $\mathcal{D}ata$?

2. Pullbacks in $\mathcal{D}ata$, which always exist, are strongly related with *quantifier-free reductions* [Imm99]. This brings forward a "reduction as change of base" perspective. Does this categorical viewpoint have any elucidating consequence in complexity?

3. It is not hard to construct presheaves on $\mathcal{D}ata$ which have remarkable significance in terms of complexity. For instance, one may define a presheaf $R : \mathcal{B}ool \to \mathbf{Set}$ sending each theory $\mathbb{S}$ to the set of relational morphisms on $\operatorname{Spec}\mathbb{S}$, and acting on morphisms by change of base. Thanks to the "reduction as change of base" viewpoint, R may be seen as a "universal" NP-complete problem, in the sense that any NP problem, seen as a relational morphism, reduces to R in a unique way. In fact, R may be shown to be related to SAT (the satisfiablity problem), whose NP-completeness is a landmark complexity result known as Cook's theorem [AB09]. How general is this construction? If repeated with other classes of morphisms characterizing complexity classes, does it yield presheaves related to known complete problems for those classes?

4. In light of the above, if subclasses of relational morphisms in $\mathcal{D}ata$ do not suffice to characterize certain interesting subclasses of NP, is it possible to find characterizations using classes of morphisms of presheaves on $\mathcal{D}ata$ instead? For instance, there is some evidence that a characterization of NL (non-deterministic logarithmic space) may be obtained in this way.

5. Arguably, the most important open problem of descriptive complexity is finding a characterization of "order independent" P, that is, the class of languages on unordered

in the language of schemes, this states exactly that a subset of $\mathbb{Z}$ is recursively enumerable iff it is the integral image (defined as the finite image but with $\operatorname{Spec}\mathbb{Z}$ instead of $\operatorname{Spec}\mathbb{E}$) of a morphism of finite presentation over $\operatorname{Spec}\mathbb{Z}[X]$.

structures which are decidable in polynomial time by a deterministic Turing machine operating on an arbitrary ordered version of them (since Turing machines only accept strings as input, one must introduce a total order: for example, for representing a graph as a string, one necessarily introduces a total order on the nodes of the graph, corresponding to the "left-to-right" order of the characters of a string). Can the categorical perspective shed any light on this problem, or at least its difficulty?

These technical questions are complemented by considerations of more philosophical nature, which might nevertheless be interesting to develop. For example, there is a canonical functor $\Gamma : \mathcal{D}ata \to \mathbf{Set}$, known as the "global section functor", sending a data specification $\operatorname{Spec}\mathbb{S}$ to the set of finite models of $\mathbb{S}$, and sending $f : \operatorname{Spec}\mathbb{S} \to \operatorname{Spec}\mathbb{T}$ to the computable function corresponding to f. Being a quantifier-free query, this is in fact an extremely low-complexity function (for example, well below logspace-computable—an exact description is too technical to be given here). So the arrows of $\mathcal{D}ata$ may be seen as low-complexity programs, typed by data specifications, and the global section functor is the "semantics" of such types and programs, that is, it associates with each program the set-theoretic function that it computes. Complexity theory is interested in answering questions in **Set**, but what we actually have access to is the programs in $\mathcal{D}ata$, and the properties of the global section functor give an idea, at least conceptually, of the difficulty of complexity-theoretic questions.

For instance, the fact that the global section functor preserves monomorphisms but does not reflect them means that a decision problem, which is a subset of $\{0,1\}^*$ and therefore just a monomorphism of the form $X \rightarrowtail \{0,1\}^*$ of **Set**, is usually implemented by complex morphisms of the form $\operatorname{Spec}\mathbb{X} \to \operatorname{Spec}\mathbb{S}\mathrm{tr}$ in $\mathcal{D}ata$, which are potentially much harder to reason about than mere subobjects. More generally, this corresponds to the fundamental distinction in computer science between what is "intensional" (the category $\mathcal{D}ata$) and what is "extensional" (the category **Set**), and the technical framework suggested here might give an interesting philosophical viewpoint on this.

The methodology will be to address simpler questions first and move to progressively more complex material as the work advances. Indeed, the above objectives are listed more or less in order of difficulty, and they are connected so as to facilitate a smooth development.

Plan. We expect the work to be developed along roughly the following plan:

Year 1: bibliography, introduction to the theory; first steps towards objectives (1) and (2); philosophical considerations.

Year 2: more work on objective (1) and philosophical considerations; objectives (3) and (4).

Year 3: finalizing all objectives; reflection about objective (5); writing the manuscript.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[CLW93] Aurelio Carboni, Stephen Lack, and R.F.C. Walters. Introduction to extensive and distributive categories. *Journal of Pure and Applied Algebra*, 84(2):145–198, 1993.

[Imm99] Neil Immerman. *Descriptive Complexity*. Springer, 1999.

[Joh02] Peter T. Johnstone. *Sketches of en Elephant. A Topos Theory Compendium. Volume 2*. Oxford University Press, 2002.