A Categorical Approach to Describing Complexity Classes

Baptiste Chanus and Damiano Mazza
CNRS, LIPN, Université Sorbonne Paris Nord

26 January 2024

Abstract

We introduce a categorical framework for describing decision problems and providing machine-independent characterizations of complexity classes, mixing ideas coming from descriptive complexity and categorical logic. The framework is based on the opposite of the category of small Boolean lextensive categories and logical functors. This has nice categorical properties and, crucially, its morphisms may be seen as descriptions of decision problems. The complexity of problems may be controlled by restricting morphisms via suitable algebraic and logical properties. We show the breadth of this approach by capturing several classes of different flavor: recursively enumerable problems, NP, P, NL and P/poly.

1 Introduction

The basic objects of study in computability and complexity live in the category **Set** of sets and functions: a decision problem is a subset of some set Σ of instances; a reduction from a problem $U \subseteq \Sigma$ to a problem $V \subseteq \Delta$ is a (low-complexity) function $r : \Sigma \to \Delta$ such that $r^{-1}(V) = U$, a complexity class is a set of decision problems, and so on. However, the category of sets does not "know" anything about computability and complexity, in the sense that these cannot be defined using categorical structures or properties of **Set**, and one must rely on external notions, typically machines, automata, circuits and so on.

The aim of this paper is to present a well-structured categorical formulation of the external machinery that controls computability and complexity. As a first approximation, the idea is to find a well-behaved (in terms of limits and colimits) category **Data** together with a functor (preserving enough limits and colimits) $\Gamma : \mathbf{Data} \to \mathbf{Set}$ such that:

1. there is a class of objects of **Data**, which we may call *data types*, such that every possible set of instances of interest is equal to ΓS for some data type S;

2. given problems $U \subseteq \Gamma S$ and $V \subseteq \Gamma T$, a reduction (for a general enough notion of reduction) is an arrow $r : S \to T$ of **Data** such that $(\Gamma r)^{-1}(V) = U$;

3. for any complexity class of interest C, there is a class of arrows $\mathcal{K}$ of **Data**, definable purely in terms of **Data** itself, such that a problem $U \subseteq \Gamma S$ is in C iff there exists a morphism $p : X \to S$ in $\mathcal{K}$ such that $U = \operatorname{im} \Gamma p$.

The above description is somewhat vague and technically imprecise and may therefore be realized in uninteresting ways, so the quality of the framework really depends on how **Data** is defined. Our approach draws inspiration from descriptive complexity and uses categorical logic to find well-beahaved structures.

Descriptive complexity [Imm99] is grounded on the idea, originally due to Fagin [Fag74], that the difficulty of a computational problem may be understood in terms of how expressive a logical language must be in order to describe it, rather than how many resources a machine must use in order to solve it. The key insight is to view instances of problems as *finite structures* in the sense of model theory. The "yes" instances of a problem then become finite structures verifying some property, and one may study in which logical languages there are formulas describing exactly that property.

Categorical logic [Jak99, Joh02] originated in Lawvere's work [Law63, Law70] with the observation that the basic set-theoretic operations used to interpret first-order logic (intersection, complement, images…) are available in any category with enough properties. This unveils an intimate connection between logical and categorical language. In particular, classes of theories over a certain logical language exactly correspond to classes of categories with certain properties, shared with **Set**. Moreover, if $\mathcal{C}_{\mathbb{T}}$ is the category corresponding to a theory $\mathbb{T}$, then models of $\mathbb{T}$ are expressed by property-preserving functors $\mathcal{C}_{\mathbb{T}} \to \mathbf{Set}$, and homomorphisms of models by natural transformations. Similarly, *finite* models are functors $\mathcal{C}_{\mathbb{T}} \to \mathbf{FinSet}$, the category of finite sets. This is a direct link with descriptive complexity.

We are therefore led to consider first-order theories which are expressive enough to match the finite structures used in descriptive complexity. These theories correspond to (small) *Boolean lextensive categories*, which themselves form a category **BoolLext**. It turns out that letting $\mathbf{Data} := \mathbf{BoolLext}^{\mathrm{op}}$ gives a very broad context in which all of computability and complexity may be formulated.

Technically, since the morphisms of **BoolLext** are functors, to have nice categorical properties we need to consider natural isomorphisms between them, so **Data** turns out to be a strict $(2,1)$-category, *i.e.*, a strict 2-category in which every 2-arrow is invertible. Accordingly, Γ will be a 2-functor $\mathbf{Data} \to \mathbf{Grpd}$, rather than a functor $\mathbf{Data} \to \mathbf{Set}$ as mentioned in our initial sketch. Here, **Grpd** denotes the $(2,1)$-category of *groupoids*, which are categories in which every arrow is invertible. **Data** has 2-limits and well-behaved finite 2-coproducts, and Γ preserves them.

The intuition is that an object S of **Data**, which we call a *data specification*, has an underlying theory $\mathbb{S}$ and is mapped by Γ to the groupoid whose ob-

jects are finite models of $\mathbb{S}$ and whose arrows are model isomorphisms. *Data types* will be data specifications corresponding to finite theories. These are general enough to address point (1) above. For example, there is a data type Str such that the objects of ΓStr are, as in descriptive complexity, finite structures representing binary strings, and the arrows of ΓStr relate two structures representing the same string.

For what concerns morphisms of **Data**, given $f : S \to T$ with S, T data types, the object part of the functor $\Gamma f : \Gamma S \to \Gamma T$ turns out to be a *quantifier-free query*. Therefore, *quantifier-free reductions* may be expressed by arrows of **Data**, as mentioned in (2).

For what concerns (3), a substantial number of interesting complexity classes may be captured by means of classes of morphisms formulated purely in terms of data specifications (or, dually, Boolean lextensive categories). We chose a compromise between conciseness and variety, and present characterizations of recursively enumerable problems, NP, P, NL and $\mathsf{P/poly}$. However, we currently have characterizations of several other classes, such as AC^0 (also known as FO), L and PH (the polynomial hierarchy), and it is reasonable to believe that basically anything that may be captured by descriptive complexity may also be captured by our framework.

Precisely because our approach borrows so much from descriptive complexity, it is important to understand where it diverges from it. The most important difference is that we formulate everything within one logical framework, that of Boolean lextensive categories, roughly corresponding to (multi-sorted) quantifier-free classical logic. Whereas descriptive complexity changes logical language for each complexity class, we rely on the relative difference between logical theories: in a morphism $X \to Str$ of **Data**, the theory underlying X may be seen as an extension of the theory of strings, and the farthest X is allowed to be from Str, the more problems it will be able to express.

Another difference is that we use finite *models* rather *structures*. This is a minor point but it allows, for example, to express order internally via axioms, whereas, in descriptive complexity, one must impose externally that order predicates in a structure are interpreted as desired. The same holds for arithmetic predicates. Another nice aspect of the categorical approach is that it naturally accounts for isomorphisms of finite models, although in this paper we do not make substantial use of this.

The interface between category theory and computational complexity or automata theory has been growing recently [Abr22, DJR21, CP20, GPR20, ADW17, AS21, AR23]. While technically unrelated, the contributions of this paper fit in this line of work and add a new direction to it, providing a unifying framework for defining and studying complexity classes. We strived to include as much detail as possible to show the breadth of the framework, including, for example, a proof of the Cook-Levin theorem, which gives a glimpse of what may be done besides characterizing complexity classes.

Unfortunately, reading this paper still requires some categorical background: it is written more for the category theorist wanting to learn about complexity than for the complexity theorist wanting to learn about categories. This is solely justified by space constraints, which make it much harder to

write a paper taking the dual approach.

2 Categories and Theories

2.1 Boolean Lextensive Categories

In what follows, we denote terminal objects by 1, binary products by $\times$, initial objects by 0, binary coproducts by $+$ and injections by $\iota_i : A_i \to A_1 + A_2$. Also, we use the following notations for pullback squares:

$$\begin{array}{ccc} B \times_A C & \xrightarrow{g^* f} & C \\ {\scriptstyle f^* g}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[f]{} & A \end{array}$$

In particular, f^*g denotes the pullback of g along f.

Definition 1 (disjointness, pullback-stability) *A coproduct $A + B$ with injections ι_1, ι_2 is said to be*

- disjoint *if $\iota_1^* \iota_2$ and $\iota_2^* \iota_1$ are initial arrows (i.e., $A \times_{A+B} B = 0$);*
- pullback-stable *if, for any $f : C \to A + B$, $f^*\iota_1$ and $f^*\iota_2$ exist and are injections.*

Definition 2 (Boolean lextensive category) *A category is* lextensive *if it has finite limits and finite disjoint pullback-stable coproducts. Lextensive categories are distributive, i.e., $A \times 0 = 0$ and $A \times (B + C) = A \times B + A \times C$ [CLW93]. A* logical functor *between lextensive categories is a functor preserving all finite limits and finite coproducts.*

Let $i : X \to A$ and $j : Y \to A$ be two monomorphisms. They are said to be equivalent *if there exists an iso $k : X \to Y$ such that $i = jk$. A* subobject *of an object A, denoted by $X \rightarrowtail A$, is an equivalence class of monomorphisms into A. A subobject $X \rightarrowtail A$ of a lextensive category has a* complement *if there exists a subobject $\overline{X} \rightarrowtail A$ such that $A = X + \overline{X}$.*

A lextensive category is Boolean *if every subobject has a complement. We denote by* **BoolLext** *the strict $(2,1)$-category of small Boolean lextensive categories, logical functors and natural isomorphisms.*

Lemma 3 ([CLW93]) *A category is Boolean lextensive iff it has finite products, finite coproducts, $1 + 1$ is disjoint and pullback-stable and every monomorphism is an injection. Also, a functor between Boolean lextensive categories is logical as soon as it preserves finite products and finite coproducts.*

The prototypical example of Boolean lextensive category is **Set**, the category of sets and functions. Another important example, which is small, is the skeleton of the category of *finite* sets $\mathcal{F}$, which we define to be the category whose objects are obtained by choosing one set $[n]$ of cardinality n for each

$n \in \mathbb{N}$ and whose arrows are arbitrary functions. An example which is not a Boolean topos is obtained by adding to $\mathcal{F}$ a countably infinite set, again with all functions as arrows. The general construction of Sect. 2.3 will give examples which are not Boolean pretoposes, *i.e.*, in which the image of an arrow does not necessarily exist.

2.2 Theories

Boolean lextensive categories are closely related to a certain kind of first-order theories, which we proceed to describe.

Definition 4 (vocabulary, formula) *A* vocabulary $\mathbb{V}$ *is a pair*

$$(\mathrm{Sort}(\mathbb{V}), \mathrm{Rel}(\mathbb{V}))$$

where $\mathrm{Sort}(\mathbb{V})$ *is a set of* sorts *and* $\mathrm{Rel}(\mathbb{V})$ *is a set of* relation symbols*, which are pairs* $R \rightarrowtail A_1 \times \ldots \times A_k$ *of a syntactic symbol* R *and an ordered list of sorts* $A_1, \ldots, A_k$*, called the* type *of the sort. The case* $k = 0$ *is allowed and written* $R \rightarrowtail 1$*. We may write* A^k *for* $A \times \cdots \times A$ *repeated* k *times.*

The formulas *on a vocabulary* $\mathbb{V}$ *are defined as follows:*

$$\varphi, \psi ::= x =_A y \mid R(x_1, \ldots, x_n) \mid \bot \mid \varphi \wedge \psi \mid \neg\varphi \mid \exists x.\varphi$$

where A *ranges overs sorts and* x, y, x_i *range over a countable set of* variables*. Free and bound variables in formulas are defined as usual, with* $\exists$ *being the only binder. As customary, we consider equal two formulas which differ only by an injective renaming of bound variables (*α*-equivalence). We use the usual abbreviations available in classical logic:* $x \neq_A y := \neg(x =_A y)$*,* $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$*,* $\top := \neg\bot$*, etc.*

We denote by $\varphi[y/x]$ *the formula obtained from* φ *by replacing every free occurrence of* x *with* y*, possibly after having renamed bound variables so that* y *is not captured by a binder.*

In the standard semantics of first-order logic, sorts correspond to sets (the domains for the individuals) and a quantifier-free formula with free variables in $x_1 : A_1, \ldots, x_n : A_n$ defines a subset of $\prod_{i=1}^{k} A_i$: inductively, $x =_A y$ is the diagonal of A; relation symbols, by definition, are subsets; $\bot$ is the empty subset; $\wedge$ is intersection; and $\neg$ is complement.

The key observation underlying categorical logic [Joh02] is that any Boolean lextensive category has enough structure to interpret quantifier-free logic: in the above, just replace "subset" with "subobject" and "intersection" with "pullback" (remember that lextensive categories have finite limits). In fact, Boolean lextensive categories may also interpret a restricted form of existential quantification. If $i : \varphi \to A \times B$ is an injection and $p : A \times B \to B$ is the projection, then the image of $p \circ i$ is precisely $\exists x.\varphi \subseteq B$. Unfortunately, $p \circ i$ is not injective in general, which means that it does not define a subobject. However, if, for each y, there is at most one x such that $(x, y) \in \mathrm{im}\, i$, then $p \circ i$ is still injective. So Boolean lextensive categories can interpret unique existential quantification.

What we call *theories* in this paper are multisorted first-order theories with equality whose axioms are closed formulas of the form $\forall\vec{x}.\varphi$ where φ

is quantifier-free except for the presence of *provably unique* existential quantifiers. This is formalized as follows:

Definition 5 (theory, model, extension) *Fix a vocabulary* $\mathbb{V}$ *and let* φ *be a formula of* $\mathbb{V}$ *with free variables* $\vec{x}$*. As customary, we refer to the closed first-order formula* $\forall\vec{x}.\varphi$ *as the* universal closure *of* φ*. Let* Ax *be a set of formulas of* $\mathbb{V}$*. We say that a formula* ψ *of* $\mathbb{V}$ *is* provable from Ax *if the universal closure of* ψ *is provable in the first-order theory consisting of the universal closures of the formulas in* Ax*.*

A theory $\mathbb{T}$ *is a triple*

$$(\mathrm{Sort}(\mathbb{T}), \mathrm{Rel}(\mathbb{T}), \mathrm{Ax}(\mathbb{T}))$$

where $(\mathrm{Sort}(\mathbb{T}), \mathrm{Rel}(\mathbb{T}))$ *is a vocabulary, abusively denoted by* $\mathbb{T}$ *as well, and* $\mathrm{Ax}(\mathbb{T})$ *is a set of formulas of* $\mathbb{T}$*, called* axioms*, admitting a well-founded partial ordering such that, for all* $\varphi \in \mathrm{Ax}(\mathbb{T})$ *and for every subformula of the form* $\exists x.\psi$ *of* φ *with* x *of sort* A*, the formula*

$$(\psi \wedge \psi[x'/x]) \Rightarrow x =_A x'$$

is provable from the subset of $\mathrm{Ax}(\mathbb{T})$ *of all axioms strictly below* φ*.*

We say that a formula is provable *in the theory* $\mathbb{T}$ *if it is provable from* $\mathrm{Ax}(\mathbb{T})$*. We say that two formulas* φ *and* φ' *are* provably equivalent *in* $\mathbb{T}$ *if* $\mathbb{T}$ *proves* $\varphi \Leftrightarrow \varphi'$*.*

A model *of a theory* $\mathbb{T}$ *is defined as usual, with each sort* A *being interpreted by a set* $[\![A]\!]$*, so that a relation symbol* $R \rightarrowtail A_1 \times \cdots \times A_k$ *is interpreted by a subset of* $[\![A_1]\!] \times \cdots \times [\![A_k]\!]$*. Isomorphism of models is defined as customary. We denote by* $\mathrm{Mod}(\mathbb{T})$ *the set of* standardized finite models *of a theory* $\mathbb{T}$*, by which we mean that sorts are interpreted by sets of the form* $[n]$*, the same sets we used in our definition of the category* $\mathcal{F}$*, so that* $\mathrm{Mod}(\mathbb{T})$ *is indeed a set.*

A theory $\mathbb{T}'$ *is an* extension *of* $\mathbb{T}$ *if* $\mathrm{Sort}(\mathbb{T}) \subseteq \mathrm{Sort}(\mathbb{T}')$*,* $\mathrm{Rel}(\mathbb{T}) \subseteq \mathrm{Rel}(\mathbb{T}')$ *and* $\mathrm{Ax}(\mathbb{T}) \subseteq \mathrm{Ax}(\mathbb{T}')$*. Notice that, in that case, a model of* $\mathbb{T}'$ *may be seen as a model of* $\mathbb{T}$ *with additional stuff (the individuals of the sets interpreting the additional sorts), structure (the additional relations) and properties (the additional axioms). We write* $\mathbb{T}' = \mathbb{T} + \mathbb{X}$ *to mean that* $\mathbb{X}$ *is the extra sorts, relation symbols and axioms that* $\mathbb{T}'$ *adds to* $\mathbb{T}$*. We say that the extension* $\mathbb{T}'$ *is* finite *if* $\mathbb{X}$ *is finite.*

Observe that function symbols are absent from our definition of vocabulary because they may be defined in terms of relations, plus suitable axioms. For example, a unary function symbol $f : A \to B$ (where A and B are sorts) may be simulated by a relation symbol $F \rightarrowtail A \times B$ together with the axioms

$$F(x,y) \wedge F(x,y') \Rightarrow y =_B y' \qquad\qquad \exists y.F(x,y).$$

The existential quantifier in the second axiom is provably unique because of the first axiom. Then, if one wishes to use a formula of the form $\varphi(f(x))$, one may use instead $\exists y.F(x,y) \wedge \varphi(y)$. This obviously generalizes to n-ary function symbols, including constants.

Every small Boolean lextensive category $\mathcal{B}$ induces a theory $\mathbb{L}(\mathcal{B})$, called its *internal language*, whose sorts are the objects of $\mathcal{B}$ and whose relation symbols are the subobjects of $\mathcal{B}$ (smallness is required so that these are, in fact,

sets). Applying the above-mentioned interpretation of first-order logic in $\mathcal{B}$, any formula φ in this vocabulary whose free variables are $\vec{x} : \vec{A}$ induces a subobject of $\vec{A}$ in $\mathcal{B}$. If such a subobject is isomorphic to $\vec{A}$ itself, then we say that φ is *full*. The axioms of $\mathbb{L}(\mathcal{B})$ are the full formulas.

Let us now look at some concrete examples of theories. The empty theory $\mathbb{E}$ has no sorts, no relation symbols and no axioms. It has exactly one model (the empty model). Any inconsistent theory has no model, an example is the theory $\perp\!\!\!\perp$ extending $\mathbb{E}$ with the axiom $\bot$. The theory of directed graphs $\mathbb{G}\mathrm{rph}$ has one sort N, a relation symbol $E \rightarrowtail N^2$ and no axioms. Another interesting example is the theory $\mathbb{S}\mathrm{tr}$ which has one sort N, two relation symbols $\leq \rightarrowtail N^2$ and $X \rightarrowtail N$, and axioms stating that $\leq$ is a total order:

$$\begin{array}{ll} x \leq x & x \leq y \wedge y \leq z \Rightarrow x \leq z \\ x \leq y \wedge y \leq x \Rightarrow x =_N y & x \leq y \vee y \leq x. \end{array}$$

The elements of $\mathrm{Mod}(\mathbb{S}\mathrm{tr})$ may be identified with binary strings: N is the set of positions in the string, which are totally ordered from left to right, and $X(i)$ holds exactly when the string has a 1 at position i. For example, the model $\{i < j < k\}$ in which $X = \{k\}$ represents the string 001. However, observe that the same string may be represented by different, isomorphic models, such as $\{j < k < i\}$ in which $X = \{i\}$.

Let $\mathbb{O}\mathrm{rd}$ be the theory defined like $\mathbb{S}\mathrm{tr}$, but without the symbol X. Its models are totally ordered sets. Observe that a finite model of $\mathbb{O}\mathrm{rd}$ necessarily has a minimum and a maximum element, but the theory has no access to them. Consider the theory $\mathbb{C}\mathrm{hain}$ extending $\mathbb{O}\mathrm{rd}$ with relation symbols $\mathrm{Z}, \mathrm{Max} \rightarrowtail N$ and $\mathrm{S} \rightarrowtail N^2$, axiomatized as follows:

1. If N is not empty, then Z and Max are singletons (*i.e.*, they define constants):

$$\begin{array}{ll} \mathrm{Z}(x) \wedge \mathrm{Z}(x') \Rightarrow x =_N x' & \mathrm{Max}(x) \wedge \mathrm{Max}(x') \Rightarrow x =_N x' \\ x =_N x \Rightarrow \exists z.\mathrm{Z}(z) & x =_N x \Rightarrow \exists m.\mathrm{Max}(m) \end{array}$$

2. S is an injective partial function (the successor):

$$\mathrm{S}(x,y) \wedge \mathrm{S}(x,y') \Rightarrow y' =_N y \qquad \mathrm{S}(x,y) \wedge \mathrm{S}(x',y) \Rightarrow x' =_N x$$

3. Zero has no predecessor, max has no successor, every non-zero element has a predecessor and every non-max element has a successor:

$$\begin{array}{ll} \mathrm{Z}(z) \wedge \mathrm{S}(x,z) \Rightarrow \bot & \mathrm{Max}(m) \wedge \mathrm{S}(m,y) \Rightarrow \bot \\ \mathrm{Z}(z) \wedge y \neq_N z \Rightarrow \exists x.\mathrm{S}(x,y) & \mathrm{Max}(m) \wedge x \neq_N m \Rightarrow \exists y.\mathrm{S}(x,y) \end{array}$$

4. The chain induced by the successor relation is acyclic:

$$\mathrm{S}(x,y) \Rightarrow x \leq y \wedge x \neq_N y$$

The sets $\mathrm{Mod}(\mathbb{C}\mathrm{hain})$ and $\mathrm{Mod}(\mathbb{O}\mathrm{rd})$ are in canonical bijection: given a finite total order, there is only one way of equipping it with a zero, a max and a

successor function. However, $\mathbb{C}$hain is more expressive. For example, we may state that the size of N is n:

$$\exists x_1 \dots \exists x_n \mathrm{Z}(x_1) \wedge \mathrm{S}(x_1, x_2) \wedge \dots \wedge \mathrm{S}(x_{n-1}, x_n) \wedge \mathrm{Max}(x_n).$$

By contrast, in $\mathbb{O}$rd (as in any theory) we may only state that there are *at most* n elements, via the formula $\bigvee x_i =_N x_j$ ranging over all $1 \leq i \neq j \leq n+1$.

Definition 6 (chain sort, chain object) *We say that a sort of a theory is a* chain sort *if it is equipped with the same relation symbols and axioms as the sort N in* $\mathbb{C}$hain.

Let $\mathcal{B}$ be a Boolean lextensive category. A chain object *of $\mathcal{B}$ is an object A equipped with subobjects $\leq_A, \mathrm{Z}_A, \mathrm{Max}_A, \mathrm{S}_A$ of A^2, A, A, A^2, respectively, such that the axioms of chain sorts are provable in the internal language of $\mathcal{B}$.*

Let P be a polynomial with non-negative integer coefficients. In any Boolean lextensive category the existence of finite products and finite coproducts allows us to evaluate P at any object A, obtaining another object $P(A)$. The following says that, if A is a chain object, then so is $P(A)$:

Lemma 7 *Chain objects are closed under finite products and finite coproducts.*

Proof. The terminal and initial object are trivially chain objects. Let A and B be chain objects of a Boolean lextensive category. The lexicographic order on $A \times B$ is expressible by

$$(x \leq_A x') \vee (x =_A x' \wedge y \leq_B y').$$

Zero is $\mathrm{Z}_A(x) \wedge \mathrm{Z}_B(y)$ and max is $\mathrm{Max}_A(x) \wedge \mathrm{Max}_B(y)$, whereas the successor is

$$(\neg \mathrm{Max}_B(y) \wedge x' =_A x \wedge \mathrm{S}_B(y, y')) \vee (\mathrm{Max}_B(y) \wedge \mathrm{S}_A(x, x') \wedge \mathrm{Z}_B(y')).$$

For what concerns $A + B$, we equip it with the concatenation of $\leq_A$ and $\leq_B$. Since $(A+B)^2 = A^2 + A \times B + B \times A + B^2$, this has four components, which are $\leq_A$, $\top$, $\bot$ and $\leq_B$, respectively. The two components of zero are Z_A and $\bot$, and the two components of max are $\bot$ and Max_B. The successor too has four components: the ones in A^2, B^2 and $B \times A$ are S_A, S_B and $\bot$, respectively; the component in $A \times B$ is expressed by $\mathrm{Max}_A(x) \wedge \mathrm{Z}_B(y)$. □

2.3 Syntactic Categories

Definition 8 ($\mathbb{T}$-set) *Let $\mathbb{T}$ be a theory. A* context *for $\mathbb{T}$ is a finite, possibly empty sequence of variables decorated with sorts of $\mathbb{T}$, of the form $x_1^{A_1} \dots x_n^{A_n}$, and denoted more concisely by $\vec{x} : \vec{A}$ or even just $\vec{x}$ when the sorts are irrelevant.*

A simple $\mathbb{T}$-set *is an expression of the form $\vec{x}.\varphi$ where φ is a formula of $\mathbb{T}$ and $\vec{x}$ a context including the free variables of φ, such that the sorts attributed to variables in $\vec{x}$ are consistent with how they appear in φ. Simple $\mathbb{T}$-sets are considered up to renaming the variables in the context. For example, if $R \rightarrowtail A \times B$, then $x^A y^B.R(x,y)$ and $z^A w^B.R(z,w)$ are identified, whereas $y^B x^A.R(x,y)$ is different.*

A $\mathbb{T}$-set *is a finite (possibly empty) list of simple* $\mathbb{T}$*-sets, concisely denoted by* $(\vec{x}_i.\varphi_i)_{i=1}^n$ *or, even more concisely, when* n *is clear from the context or irrelevant, by* $(\vec{x}_i.\varphi_i)$.

The intuition behind $\mathbb{T}$-sets is that they are sets expressible by quantifier-free (except for provably unique existential quantifiers) formulas of $\mathbb{T}$. Every sort A of $\mathbb{T}$ gives a $\mathbb{T}$-set $(x^A.\top)$, which we abusively denote by A. Then, in general, if $\vec{x}_i : A_i^1 \dots A_i^{k_i}$, $(\vec{x}_i.\varphi_i)$ defines a subset of $\sum_i A_i^1 \times \cdots \times A_i^{k_i}$. Finite Cartesian products and disjoint unions of $\mathbb{T}$-sets are also $\mathbb{T}$-sets: $(\vec{x}_i.\varphi_i) \times (\vec{y}_j.\psi_j) := (\vec{x}_i\vec{y}_j.\varphi_i \wedge \psi_j)$ and $(\vec{x}_i.\varphi_i) + (\vec{y}_j.\psi_j)$ is concatenation of sequences, the singleton set is $1 := (\top)$ and the empty set $0 := ()$. This shows, in particular, that finite sets are always $\mathbb{T}$-sets.

For example, in the theory $\mathbb{G}$rph, the unique sort N represents the set of nodes of a generic directed graph, and $(xy.E(x,y)) \subseteq N^2$ is its set of edges. Other examples of $\mathbb{G}$rph-sets are the complement of the set of edges $(xy.\neg E(x,y)) \subseteq N^2$, or

$$(xyz.\neg E(x,x) \wedge \neg E(y,y) \wedge \neg E(z,z) \wedge E(x,y) \wedge E(y,z)) \subseteq N^3,$$

which corresponds to the set of triples of loop-free nodes forming a path of length 2.

Notice how some different $\mathbb{T}$-sets should morally be identified: for example, any $\mathbb{T}$-set $(x^A.\bot)$ expresses the empty set $()$. Logical equivalence is not enough, we need a more general way of saying that two $\mathbb{T}$-sets are in bijection.

Let $S := \vec{x}.\varphi$ and $T := \vec{y}.\psi$ be simple $\mathbb{T}$-sets, with $\vec{y} : \vec{B}$. A *partial* $\mathbb{T}$*-function* $S \rightharpoonup T$ is a simple $\mathbb{T}$-set $\vec{x}\vec{y}.\theta$ such that the following are provable in $\mathbb{T}$:

$$\theta \Rightarrow \varphi \wedge \psi \qquad \text{(domain and codomain)},$$
$$\theta \wedge \theta[\vec{y}'/\vec{y}] \Rightarrow \vec{y} =_{\vec{B}} \vec{y}' \qquad \text{(functionality)}.$$

In other words, $\mathbb{T}$ proves that $\theta \subseteq S \times T$ is a functional relation. A $\mathbb{T}$*-function* $S \to T$ is, additionally, *provably total*, *i.e.*, $\mathbb{T}$ proves $\varphi \Rightarrow \exists\vec{y}.\theta$.

Since $\mathbb{T}$-sets are disjoint unions of simple $\mathbb{T}$-sets, $\mathbb{T}$-functions between them may be defined as matrices of partial functions between the components. For example, a $\mathbb{T}$-function $(S_1, S_2) \to (T_1, T_2)$ will be a 2×2 matrix of partial $\mathbb{T}$-functions $\theta_{ij} : S_j \rightharpoonup T_i$, such that the domains of θ_{1j} and θ_{2j} form a partition of S_j.

Definition 9 ($\mathbb{T}$**-function)** *Fix a theory* $\mathbb{T}$ *and let* $P := (S_j)_{j=1}^n$ *and* $Q := (T_i)_{i=1}^m$ *be* $\mathbb{T}$*-sets, with* $S_j := \vec{x}_j.\varphi_j$, $\vec{x}_j : \vec{A}_j$, $T_i := \vec{y}_i.\psi_i$ *and* $\vec{y}_i : \vec{B}_i$. *We may assume all* $\vec{x}_i$ *and* $\vec{y}_j$ *to be pairwise disjoint, even in case* $P = Q$ *(i.e., we choose different representatives for the same simple* $\mathbb{T}$*-sets). A* $\mathbb{T}$-function $P \to Q$ *is an* $m \times n$ *matrix of partial* $\mathbb{T}$*-functions* $\vec{x}_j\vec{y}_i.\theta_{ij} : S_j \rightharpoonup T_i$ *as defined above, such that, for all* $1 \leq i \leq m$ *and* $1 \leq j \neq j' \leq n$, $\mathbb{T}$ *proves the following:*

$$\theta_{ij} \wedge \theta_{ij'} \Rightarrow \bot \qquad \textit{(disjointness)}$$
$$\varphi_j \Rightarrow \bigvee_{i=1}^m \exists\vec{y}_i.\theta_{ij} \qquad \textit{(joint totality)}$$

$\mathbb{T}$-functions are defined up to equivalence: a matrix $(\vec{x}_j\vec{y}_i.\theta'_{ij})$ such that, for all i and j, θ'_{ij} and θ_{ij} are provably equivalent in $\mathbb{T}$ defines the same $\mathbb{T}$-function as $(\vec{x}_j\vec{y}_i.\theta_{ij})$.

Given $R := (\vec{z}_k.\rho_k)_{k=1}^{p}$ and $\mathbb{T}$-functions $\theta : P \to Q$ and $\chi : R \to P$, we define, for each $1 \leq i \leq m$ and $1 \leq k \leq p$,

$$(\theta \circ \chi)_{ik} \quad := \quad \vec{z}_k\vec{y}_i. \bigvee_{j=1}^{n} \exists \vec{y}_j.\theta_{ij} \wedge \chi_{jk}.$$

One may check that the above defines a $\mathbb{T}$-function $\theta \circ \chi : R \to Q$, which we call composition *of θ and χ.*

We define the identity $\mathbb{T}$-function *on P by letting, for all $1 \leq j, j' \leq n$,*

$$(\mathrm{id}_P)_{jj'} := \begin{cases} \vec{x}_j\vec{x}'_j.\varphi_j \wedge \vec{x}_j =_{\vec{A}_i} \vec{x}'_j & \text{if } j = j', \\ \vec{x}_j\vec{x}'_{j'}.\bot & \text{otherwise.} \end{cases}$$

Notice that we are using the fact that $P = (\vec{x}'_j.\varphi_j[\vec{x}'_j/\vec{x}_j])$ in order to distinguish source and target contexts. One may verify that id_P is indeed a $\mathbb{T}$-function from P to itself.

Let us look at some consequences of the definition. Consider an arbitrary theory $\mathbb{T}$. Observe that, for any $\mathbb{T}$-set $P = (\vec{x}_j.\varphi_j)_{j=1}^{n}$, there is, as expected, exactly one $\mathbb{T}$-function $!_P : 0 \to P$, given by the empty matrix: the conditions are vacuously true because $0 = ()$. Conversely, suppose we have a $\mathbb{T}$-function $f : P \to 0$. By definition, this too must be the empty matrix. However, when $n \neq 0$ the joint totality condition is no longer trivial: it amounts to $\varphi_j \Rightarrow \bot$ for all $1 \leq j \leq n$. In other words, the existence of an arrow into the empty $\mathbb{T}$-set forces each φ_j to be provably equivalent to $\bot$. But, if this is the case, then id_P is the $n \times n$ matrix whose entries are all equivalent to $(\vec{x}_j\vec{x}'_{j'}.\bot)$. This happens to be also the result of the composition $!_p \circ f$: the disjunction in the definition is nullary, hence equal to $\bot$. This proves, as expected, that $P \cong 0$. It also realizes, in particular, the desired identification of $(x^A.\bot)$ with 0.

Definition 10 (syntactic category) *The* syntactic category *of a Boolean theory $\mathbb{T}$, denoted by $\mathcal{F}[\mathbb{T}]$, is defined to be the category whose objects are $\mathbb{T}$-sets and whose arrows are $\mathbb{T}$-functions. Composition and identities are as in Definition 9. Associativity and neutrality are a consequence of the fact that $\mathbb{T}$-functions are defined up to provable equivalence.*

The acquainted reader will have recognized in $\mathcal{F}[\mathbb{T}]$ the standard construction of the syntactic category of an essentially algebraic theory as presented in [Joh02] (where these are called *cartesian theories*), augmented with finite coproducts. Indeed, our theories are essentially algebraic theories over classical logic rather than intuitionistic logic. The following is therefore expected:

Theorem 11 *For every theory $\mathbb{T}$, $\mathcal{F}[\mathbb{T}]$ is Boolean lextensive.*

Let us look at a simple but important example. Recall the empty theory $\mathbb{E}$: since it has no sorts and no relation symbols, up to equivalence there are only

two simple $\mathbb{E}$-sets, $\top$ and $\bot$ (non-empty contexts do not exist because we do not have any sort). Therefore, remembering that $(\bot) \cong 0$, up to iso the objects of $\mathcal{F}[\mathbb{E}]$ are of the form $(\top, \ldots, \top)$, *i.e.*, they may be identified with natural numbers. We invite the reader to check that an $\mathbb{E}$-function $n \to m$ is an $m \times n$ matrix such that each column has exactly one entry equal to $\top$, the rest being $\bot$. This, in turn, is the same thing as a function $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$: given $1 \leq j \leq n$, the position of the $\top$ entry on the j-th column corresponds to $f(j)$. So $\mathcal{F}[\mathbb{E}]$ is equivalent to $\mathcal{F}$, the (skeleton of the) category of finite sets and functions.

When $\mathcal{B}$ is equivalent to $\mathcal{F}[\mathbb{T}]$ in **BoolLext**, we write $\mathcal{B} \simeq \mathcal{F}[\mathbb{T}]$ and say that $\mathbb{T}$ *presents* $\mathcal{B}$. Every small Boolean lextensive category $\mathcal{B}$ is presented by its internal language, *i.e.*, one may prove that $\mathcal{F}[\mathbb{L}(\mathcal{B})] \simeq \mathcal{B}$. Of course, much smaller theories than $\mathbb{L}(\mathcal{B})$ may present $\mathcal{B}$ (think of $\mathcal{F}$, which is presented by the empty theory, while $\mathbb{L}(\mathcal{F})$ has countably many sorts!).

A remarkable property of syntactic categories is that they allow expressing models as functors: a model of a theory $\mathbb{T}$ is the same as a logical functor $\mathcal{F}[\mathbb{T}] \to$ **Set**. This is an idea going all the way back to Lawvere's "functorial semantics" [Law63]. Indeed, the objects of $\mathcal{F}[\mathbb{T}]$ are built using categorical structures which are preserved by logical functors (monomorphisms, diagonals, pullbacks, complements, coproducts), so a logical functor $F : \mathcal{F}[\mathbb{T}] \to$ **Set** is determined by its value on atomic formulas, *i.e.*, simple $\mathbb{T}$-sets of the form $x^A.\top$ or $\vec{x}.R(\vec{x})$. Once these atomic interpretations are fixed, we have a structure in the sense of first-order logic, and $F(\vec{x}.\varphi)$ is the usual set interpreting the (open) formula φ in that structure, because those categorical constructions have the customary meaning in **Set**.

To see that we actually get a model of $\mathbb{T}$, we observe that, if $\alpha \in \mathrm{Ax}(\mathbb{T})$ with free variables $\vec{x} : A_1, \ldots, A_k$, then we have $(\vec{x}.\alpha) \cong \prod_{i=1}^{k} A_i$ in $\mathcal{F}[\mathbb{T}]$. Since functors preserve isomorphisms, the interpretation of the (open) formula α is the whole domain, which means that the structure validates the axiom $\forall\vec{x}.\alpha$, as claimed.

It is worth mentioning that a natural transformation $F \to G$ with F, G logical functors $\mathcal{F}[\mathbb{T}] \to$ **Set** is exactly an *elementary embedding* of the corresponding models [Joh02]. We will only need a consequence of this fact, namely that naturally isomorphic logical functors correspond to isomorphic models.

If logical functors into $\mathcal{F}[\mathbb{T}] \to$ **Set** are models, then logical functors $\mathcal{F}[\mathbb{T}] \to \mathcal{F}$ are standardized finite models of $\mathbb{T}$, in the sense of Definition 5, and the homset **BoolLext**$(\mathcal{F}[\mathbb{T}], \mathcal{F})$ may be identified with $\mathrm{Mod}(\mathbb{T})$. So, for example, an arrow $\mathcal{F}[\mathrm{Str}] \to \mathcal{F}$ of **BoolLext** represents a binary string.

Generalizing further, a logical functor $F : \mathcal{F}[\mathbb{T}] \to \mathcal{F}[\mathbb{S}]$ is a choice of an $\mathbb{S}$-set for each sort and relation symbol of $\mathbb{T}$, such that, for every $\alpha \in \mathrm{Ax}(\mathbb{T})$ with free variables $\vec{x}$, letting $F(\vec{x}.\alpha) = (\vec{y}_i.\psi_i)$, each ψ_i is provable in $\mathbb{S}$. Since $\mathbb{S}$-sets are, essentially, sequences of formulas of $\mathbb{S}$, the acquainted reader will recognize this as a mildly generalized *quantifier-free query* [Imm99] from the finite models of $\mathbb{S}$ to the finite models of $\mathbb{T}$. Indeed, given a finite model $\mathcal{F}[\mathbb{S}] \to \mathcal{F}$ of $\mathbb{S}$, we obtain a finite model of $\mathbb{T}$ by precomposing F.

2.4 Algebras

Let $\mathcal{A} = \mathcal{F}[\mathbb{S}]$ and $\mathcal{B} = \mathcal{F}[\mathbb{S} + \mathbb{X}]$. Since every formula of $\mathbb{S}$ is a formula of $\mathbb{S} + \mathbb{X}$, there is an inclusion functor $\mathcal{A} \to \mathcal{B}$ sending each object and arrow of $\mathcal{A}$ to "itself". The idea of algebras is to see any logical functor $\mathcal{A} \to \mathcal{B}$ as an extension of a theory.

Definition 12 (algebra) *Let $\mathcal{A}$ be a Boolean lextensive category. An* $\mathcal{A}$-algebra *is a Boolean lextensive category $\mathcal{B}$ equipped with a morphism $\mathcal{A} \to \mathcal{B}$ of* **BoolLext**, *called the* structure map. *Given two $\mathcal{A}$-algebras $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{A} \to \mathcal{C}$, an* algebra morphism $h : \mathcal{B} \to \mathcal{C}$ *is a logical functor such that $hf \cong g$ (in other words, the (2,1)-category of $\mathcal{A}$-algebras is the undercategory $\mathcal{A}/\mathbf{BoolLext}$).*

Referring to the notations used before Definition 12, we write $\mathcal{B} = \mathcal{A}[\mathbb{X}]$ to mean that $\mathcal{B}$ is an $\mathcal{A}$-algebra whose structure map is an inclusion.

The category $\mathcal{F}$ is 2-initial in **BoolLext**: given a Boolean lextensive category $\mathcal{B}$, we may always define a logical functor $F : \mathcal{F} \to \mathcal{B}$ by letting $F([n]) := 1 + \cdots + 1$ n times. Up to iso, this is the only possibility because $[n] = [1] + \cdots + [1]$ n times and F must preserve finite coproducts. So every Boolean lextensive category is an $\mathcal{F}$-algebra in a unique way.

The *tensor product* of $\mathcal{A}$-algebras $\mathcal{B}$ and $\mathcal{C}$ is the 2-pushout $\mathcal{B} \otimes_{\mathcal{A}} \mathcal{C}$ in **BoolLext**, which may be constructed, up to equivalence, as the syntactic category of a theory $\mathbb{U}$ defined as follows:

- $\mathrm{Sort}(\mathbb{U})$ consists of the objects of $\mathcal{B}$ and $\mathcal{C}$ (modulo isomorphism, they may be supposed to have no object in common);
- $\mathrm{Rel}(\mathbb{U})$ consists of the subobjects of $\mathcal{B}$ and $\mathcal{C}$, plus a relation symbol $F_Z : f(Z) \times g(Z)$ for each object or subobject Z of $\mathcal{A}$, where f and g are the structure maps of $\mathcal{B}$ and $\mathcal{C}$, respectively;
- $\mathrm{Ax}(\mathbb{U})$ consists of the axioms of the internal languages of $\mathcal{B}$ and $\mathcal{C}$, plus axioms asserting that F_Z is a bijection $f(Z) \cong g(Z)$.

In the above construction, the internal languages of $\mathcal{B}$ and $\mathcal{C}$ may be replaced (in the sense that we obtain an equivalent category) by any other theories $\mathbb{S}$ and $\mathbb{T}$ presenting $\mathcal{B}$ and $\mathcal{C}$, and we obtain that $\mathbb{U}$ is a common extension of $\mathbb{S}$ and $\mathbb{T}$, which means that there always exist $\mathbb{X}$ and $\mathbb{Y}$ such that $\mathcal{B} \otimes_{\mathcal{A}} \mathcal{C}$ is equivalent to $\mathcal{B}[\mathbb{X}]$ as a $\mathcal{B}$-algebra, or to $\mathcal{C}[\mathbb{Y}]$ as a $\mathcal{C}$-algebra.

Let $\mathbb{S}$ be a theory and $\mathbb{S} + \mathbb{X}$ an extension. Every $f : \mathcal{F}[\mathbb{S}] \to \mathcal{F}[\mathbb{T}]$ induces an extension of $\mathbb{T}$, denoted by $\mathbb{T} + f(\mathbb{X})$, obtained by applying f homomorphically to $\mathbb{X}$. For example, if $\mathbb{X}$ consists of a sort C, a relation symbol $R \rightarrowtail A \times C$ with $A \in \mathrm{Sort}(\mathbb{S})$ and an axiom $R(x, z) \wedge \varphi$ where φ is a formula of $\mathbb{S}$ with free variables $\vec{x}$, and if $f(A) = (\vec{y}_i.\psi_i)$ with $\vec{y}_i : \vec{B}_i$ and $f(\vec{x}.\varphi) = (\vec{y}'_j.\psi'_j)$, then $f(\mathbb{X})$ will consist of a sort C, relation symbols $R_i \rightarrowtail \vec{B}_i \times C$ with axioms $R_i(\vec{y}_i, z) \Rightarrow \psi_i$, and axioms $R_i(\vec{y}_i, z) \wedge \psi'_j$. With these notations, the definition of tensor product directly gives:

Lemma 13 *Let $\mathcal{B}$ be an $\mathcal{A}$-algebra with structure map f. Then, $\mathcal{A}[\mathbb{X}] \otimes_{\mathcal{A}} \mathcal{B} \simeq \mathcal{B}[f(\mathbb{X})]$ (as $\mathcal{B}$-algebras).*

Notice that $\mathcal{A}$ is an $\mathcal{A}$-algebra with structure map $\mathrm{id}_{\mathcal{A}}$, which is neutral for $\otimes_{\mathcal{A}}$ by Lemma 13 (taking $\mathbb{X}$ empty). Therefore, for any $\mathcal{A}$-algebra $\mathcal{B}$, we have $\mathcal{B} = \mathcal{A} \otimes_{\mathcal{A}} \mathcal{B} \simeq \mathcal{A}[\mathbb{X}]$ for some $\mathbb{X}$. So, up to equivalence, the structure map of every $\mathcal{A}$-algebra is an inclusion.

Definition 14 *An $\mathcal{A}$-algebra equivalent to $\mathcal{A}[\mathbb{X}]$ is:*

- of finite presentation *if $\mathbb{X}$ is finite;*
- propositional *if $\mathbb{X}$ contains no sort;*
- CNF *if it propositional and every axiom of $\mathbb{X}$ is of the form*

$$\varphi \vee \bigvee_{i \in I} \alpha_i,$$

  where φ is a formula not containing symbols from $\mathbb{X}$ and the α_i are relation symbols of $\mathbb{X}$, possibly negated ($I = \emptyset$ is allowed);
- Horn *if it is CNF and at most one of the α_i is non-negated;*
- Krom *if it is CNF and I has cardinality at most* 2.

Observe that a finite model of a totally ordered sort N (as in $\mathbb{O}\mathrm{rd}$) may always be pictured as an initial segment of $\mathbb{N}$, *i.e.*, of the form $\{0 < \cdots < m\}$. A relation symbol $R \rightarrowtail N^k$ with N totally ordered is called a *numeric predicate*, because it may be seen as a relation on integers.

In descriptive complexity, numeric predicates are important for expressiveness reasons. An example is $\mathrm{Plus} \rightarrowtail N^3$, whose interpretation is posited to be the (truncated) graph of addition, *i.e.*, $\mathrm{Plus}(x, y, z)$ holds exactly when $x + y = z$. In our setting, Plus may be axiomatized on top of a chain sort (Definition 6) in such a way that finite models are forced to interpret it by standard addition:

$$\begin{aligned}
&\mathrm{Plus}(x, y, z) \wedge \mathrm{Plus}(x, y, z') \Rightarrow z' =_N z \\
&\mathrm{Plus}(x, y, z) \wedge \mathrm{Plus}(x', y, z) \Rightarrow x' =_N x \\
&\mathrm{Plus}(x, y, z) \Leftrightarrow \\
&(\mathrm{Z}(y) \wedge z =_N x) \vee (\exists y'.\exists z'.\mathrm{S}(y', y) \wedge \mathrm{S}(z', z) \wedge \mathrm{Plus}(x, y', z')).
\end{aligned}$$

The proof that finite models are standard is by induction on y.

Something similar may be done for multiplication: we introduce a symbol $\mathrm{TTimes} \rightarrowtail N^4$ which is axiomatized so that, in a model of cardinality n, $\mathrm{TTimes}(x, y, z_1, z_2)$ holds precisely when $xy = nz_1 + z_2$ (we omit the axioms for conciseness).

Let $\mathcal{A}$ be a Boolean lextensive category. A *totally ordered object* of $\mathcal{A}$ is an object T together with a subobject of T^2 satisfying the axioms of total orders (*i.e.*, those of $\mathbb{O}\mathrm{rd}$) in the internal language of $\mathbb{A}$. The following algebras allow to add chain structure and numeric predicates on top of totally ordered objects:

Definition 15 (numeric and arithmetic algebra) *Let $\mathcal{A}$ be a Boolean lextensive category, let $\{T_i\}_{1\leq i\leq n}$ be totally ordered objects of $\mathcal{A}$ and let $\mathbb{T}$ be any presentation of $\mathcal{A}$ containing sorts A_i corresponding to T_i. An $\mathcal{A}$-algebra equivalent to $\mathcal{A}[\mathbb{X}]$ is:*

- numeric *if is it propositional and $\mathbb{X}$ contains finitely many relation symbols of the form $R \rightarrowtail A_i^k$ (i and k arbitrary) and arbitrarily many axioms, and such that the function $\mathrm{Mod}(\mathbb{T}+\mathbb{X}) \to \mathrm{Mod}(\mathbb{T})$ which forgets the extra structure of models of $\mathbb{T}+\mathbb{X}$ is invertible;*
- arithmetic *if it is numeric but relation symbols are limited to those in $\mathrm{Rel}(\mathbb{C}\mathrm{hain}) \cup \{\mathrm{Plus}, \mathrm{TTimes}\}$ and whose axioms are limited to those defining such symbols (as in $\mathbb{C}$hain and above) plus finitely many additional axioms $\{\alpha_j\}_{j\in J}$ which are true in all finite models of $\mathbb{T}+\mathbb{X}\setminus\{\alpha_j\}_{j\in J}$.*

The extra axioms of arithmetic algebras are included to increase expressiveness: in a logical functor $\mathcal{B} \to \mathcal{A}[\mathbb{X}]$ the image of the axioms of any presentation of $\mathcal{B}$ must be provable in $\mathbb{T}+\mathbb{X}$, so more axioms in $\mathbb{X}$ means more such functors.

3 Data Specifications

3.1 Data Specifications and the Gamma Functor

Definition 16 (data specification) *The (2,1)-category of* data specifications *is defined as* **Data** $:=$ **BoolLext**$^{\mathrm{op}}$. *When seeing a Boolean lextensive category $\mathcal{B}$ as a data specification, that is, as an object of* **Data** *rather than* **BoolLext**, *we write* $\mathrm{Spec}\,\mathcal{B}$. *By definition, a morphism of data specifications $\mathrm{Spec}\,\mathcal{B} \to \mathrm{Spec}\,\mathcal{C}$ is just a logical functor $\mathcal{C} \to \mathcal{B}$, and composition in* **Data** *is just composition in* **BoolLext**, *but in reverse order. A 2-morphism in* **Data** *is a natural isomorphism between the underlying logical functors (the direction is irrelevant as these are all invertible).*

Let S be a data specification. A data specification over S *is a data specification X equipped with a morphism $X \to S$, called* structure map, *whereas S is called the* base. *A morphism $X \to Y$ of data specifications over S is a morphism $h : X \to Y$ of* **Data** *such that the structure map of X is isomorphic to the structure map of Y precomposed with h (in other words, the (2,1)-category of data specifications over S is the overcategory* **Data**$/S$*).*

The $(2,1)$-category **Data** is 2-complete and 2-lextensive (as in Definition 2, but in which equalities are replaced by coherent isomorphisms). As such, for what concerns limits and finite coproducts it behaves as a category of "spaces" and "continuous maps", with 2-cells being "homotopy equivalences". Describing these as *bona fide* topological spaces is beyond the scope of this paper. However, it is intuitively useful to think of $S = \mathrm{Spec}\,\mathcal{F}[\mathbb{T}]$ as a space whose points are models of $\mathbb{T}$. In particular, the terminal object $* := \mathrm{Spec}\,\mathcal{F}$ is like a "singleton space" (remember that $\mathcal{F} = \mathcal{F}[\mathbb{E}]$, and $\mathbb{E}$ has exactly one model). However, arrows $* \to S$ do not exhaust all points of S: they are arrows $\mathcal{F}[\mathbb{T}] \to \mathcal{F}$ in **BoolLext**, so they are *finite* models of $\mathbb{T}$. We call them *finite points*.

Mapping a data specification to its set of finite points is actually a 2-functor

$$\Gamma : \mathbf{Data} \longrightarrow \mathbf{Grpd},$$

defined by $\Gamma := \mathbf{Data}(*, -)$, where **Grpd** is the $(2,1)$-category of groupoids. As a hom-functor, Γ preserves limits. It also preserves finite coproducts. If $S = \operatorname{Spec}\mathcal{B}$ and if $\mathbb{S}$ is any theory presenting $\mathcal{B}$, then ΓS is equivalent to the groupoid whose set of objects is $\mathrm{Mod}(\mathbb{S})$ and whose arrows are isomorphisms of models. For example, if we let $Str := \operatorname{Spec}\mathcal{F}[\mathrm{Str}]$, then the objects of ΓStr are representations of binary strings, and two representations are related by an arrow iff they represent the same string. Given a morphism $f : S \to T$ of **Data**, where $T = \operatorname{Spec}\mathcal{C}$ and $\mathcal{C}$ is presented by $\mathbb{T}$, then, on objects, $\Gamma f : \Gamma S \to \Gamma T$ is a quantifier-free query $\mathrm{Mod}(\mathbb{S}) \to \mathrm{Mod}(\mathbb{T})$. This is because Γ acts by postcomposition: given a finite point $x : * \to S$, it yields $f \circ x : * \to T$, and, reversing arrows, we saw that in **BoolLext** this corresponds to applying a quantifier-free query. Functoriality of Γf is guaranteed by the fact that quantifier-free queries preserve isomorphism of models.

The algebraic perspective is interesting here. If T is as above and $p : \operatorname{Spec}\mathcal{C}[\mathbb{X}] \to T$ corresponds to the inclusion $\mathcal{C} \to \mathcal{C}[\mathbb{X}]$ for some $\mathbb{X}$, then Γp is the canonical projection mapping a finite model x of $\mathbb{T} + \mathbb{X}$ to the model of $\mathbb{T}$ obtained from x by forgetting the extra stuff, structure and properties given by $\mathbb{X}$: in **BoolLext**, given a model $x : \mathcal{F}[\mathbb{T} + \mathbb{X}] \to \mathcal{F}$, precomposition with the inclusion $\mathcal{F}[\mathbb{T}] \to \mathcal{F}[\mathbb{T} + \mathbb{X}]$ gives a model which is identical to x except that it is defined only on $\mathcal{F}[\mathbb{T}]$, *i.e.*, it ignores the extra stuff, structure and properties.

Now, a morphism $f : S \to T$ as above exhibits $\mathcal{B}$ as a $\mathcal{C}$-algebra, but we know that $\mathcal{B}$ is always equivalent, as a $\mathcal{C}$-algebra, to $\mathcal{C}[\mathbb{X}_f]$ for some $\mathbb{X}_f$. So, modulo equivalence, every quantifier-free query is a projection! This is the explanation: given the quantifier-free query $\Gamma f : \mathrm{Mod}(\mathbb{S}) \to \mathrm{Mod}(\mathbb{T})$, if $G_{\Gamma f} := \{(x, \Gamma f(x)) \mid x \in \mathrm{Mod}(\mathbb{S})\}$ is the graph of Γf, there is a bijection $b : \mathrm{Mod}(\mathbb{S}) \to G_{\Gamma f}$ mapping x to $(x, \Gamma f(x))$, and $\Gamma f = \Gamma p \circ b$, where $\Gamma p : G_{\Gamma f} \to \mathrm{Mod}(\mathbb{T})$ projects (x, y) to y. It turns out that $\mathbb{X}_f$ is precisely such that $\mathrm{Mod}(\mathbb{T} + \mathbb{X}_f) = G_{\Gamma f}$, and that b too is a quantifier-free query, *i.e.*, $b = \Gamma q$ for an equivalence $q : S \to \operatorname{Spec}\mathcal{C}[\mathbb{X}_f]$ verifying $f \cong p \circ q$.

3.2 Decision Problems

In the following, if S is a data specification, we write $U \subseteq \Gamma S$ to mean that U an isomorphism-closed set of objects of the groupoid ΓS (*i.e.*, if $x \in U$ and $x' \cong x$ in ΓS, then $x' \in U$). Also, if $f : X \to S$ is a morphism of **Data** and $U \subseteq \Gamma S$, we write $(\Gamma f)^{-1}(U)$ for the inverse image of Γf on objects, which is an isomorphism-closed set of objects of ΓX. In this way, we may rephrase familiar computability concepts in terms of data specifications and the Γ functor:

- a *data type* is a data specification $\operatorname{Spec}\mathcal{B}$ with $\mathcal{B}$ of finite presentation (over $\mathcal{F}$);
- a *decision problem* on a data type S is a set $U \subseteq \Gamma S$;

- given decision problems U on S and V on T, a *reduction* from U to V is a morphism $r : S \to T$ such that $(\Gamma r)^{-1}(V) = U$.

The definition of data type adheres to the idea that data manipulated by a program must be finite (our theories being potentially infinite, models based on finite sets may still contain infinite data). Decision problems as we defined them are known in descriptive complexity as *Boolean queries*, and reductions as *quantifier-free reductions*, a kind of low-complexity reductions (below logspace) which are general enough that all "natural" complete problems for all "natural" complexity classes are still complete with respect to them [Imm99].

Consider now a data specification X over a data type S. If p is its structure map, we may consider the set

$$\operatorname{im} X := \{x : * \to S \mid \exists\, v : * \to X \text{ s.t. } x \cong p \circ v\}.$$

By definition, $\operatorname{im} \Gamma p$ is a decision problem on S. The idea is that the finite points of S, which are truly finite objects, are instances of the problem and that the finite points of X are solutions, or witnesses of "yes" instances. The structure map sends each witness to its corresponding instance. The remark at the end of the previous section says that we may think of witnesses as pairs (x, w) where x is the instance, w the actual witness, and Γp the projection.

Definition 17 (expressing problems) *Let L be a decision problem on S. We say that a data specification X over S* expresses *L if $L = \operatorname{im} X$.*

The point of descriptive complexity is that it is possible to isolate, by purely logical means (or, in our approach, algebraic means), data specifications expressing problems of a given complexity.

Definition 18 *Let $f : \operatorname{Spec} \mathcal{B} \to \operatorname{Spec} \mathcal{A}$. Seen as a morphism of* **BoolLext**, *f exhibits $\mathcal{B}$ as an $\mathcal{A}$-algebra. We say that f is* of finite presentation *(or* fp*),* propositional*, etc. if $\mathcal{B}$ is of the same kind (following the terminology of Definitions 14 and 15). Furthermore, we say that a morphism f is*

- pfp *(resp.* CNFfp*) if it is propositional (resp. CNF) of finite presentation;*
- aHfp *(resp.* aKfp*) if $f = a \circ g$ with a arithmetic and g Horn (resp. Krom) of finite presentation;*
- nuHfp *if $f = a \circ g$ with a numeric and g Horn of finite presentation.*

A data specification over another data specification is said to be fp, pfp, etc. if its structure map is. A data specification is fp, pfp, etc. if its terminal arrow is (i.e., we see it as a data specification over $*$).

Observe that a CNFfp data specification $\Phi = \operatorname{Spec} \mathcal{B}$ is presented by a CNF: we have $\mathcal{B} \simeq \mathcal{F}[\mathbb{X}]$ with $\mathbb{X}$ finite and with no sorts, so the relational symbols must be of the form $R \rightarrowtail 1$, which are propositional atoms, and the axioms are disjunctive clauses, the conjunction of which is a CNF ϕ. Moreover, a finite point $* \to \Phi$ is exactly a satisfying assignment for ϕ: in **BoolLext**, it is a logical functor $\mathcal{F}[\mathbb{X}] \to \mathcal{F}$ assigning subobjects of the singleton (that is,

truth values) to the relation symbols of $\mathbb{X}$, in such a way that the axioms are provable in $\mathcal{F}$. But the axioms are the clauses of ϕ, and provable in $\mathcal{F}$ simply means true. Similarly, Horn (resp. Krom) fp data specifications are presented by Horn CNFs (resp. 2-CNFs) and their finite points are satisfying assignments.

Lemma 19 *1. All classes of morphisms of Definition 18 are pullback-stable: if $p : X \to S$ is of a given kind and $f : T \to S$ is arbitrary, $f^*p : X \times_S T \to T$ is of the same kind as p;*

2. Fp, pfp, aHfp, aKfp and nuHfp morphisms are stable under post-composition with arithmetic morphisms.

Proof. (1) First, let $p : X \to S$ be of one of the kinds corresponding to Definition 14 or Definition 15 and let $f : T \to S$ be arbitrary, with $S = \operatorname{Spec} \mathcal{B}$ and $T = \operatorname{Spec} \mathcal{C}$. We know that the category underlying X is equivalent, as a $\mathcal{B}$-algebra, to $\mathcal{B}[\mathbb{X}]$ for some $\mathbb{X}$ whose form depends on the kind of p. By duality (pullbacks in **Data** are pushouts in **BoolLext**) and Lemma 13, f^*p is the structure map of the $\mathcal{C}$-algebra $\mathcal{B}[\mathbb{X}] \otimes_{\mathcal{B}} \mathcal{C} \simeq \mathcal{C}[f(\mathbb{X})]$. We show that f^*p is of the same kind as p by checking, in each case, that $f(\mathbb{X})$ has the same structure as $\mathbb{X}$, because the former is defined (*cf.* paragraph before Lemma 13) by having f act homomorphically on $\mathbb{X}$ and adding Horn/Krom axioms. This suffices for fp and pfp morphisms, and for the rest we conclude by observing that the pullback of a composition is isomorphic to the composition of pullbacks.

(2) It is an easy consequence of the definition that fp, propositional and arithmetic morphisms are stable under composition. Also, arithmetic morphisms are pfp and numeric by definition. The result then follows immediately. □

Pullback-stability is important for so-called *change of base*: if X is a data specification over S and we have a morphism $T \to S$, then we may consider the pullback $Y := X \times_S T$, which is a data specification over T. The idea is that Y expresses a reformulation of the problem expressed by X, over different instances. For example, if $Grph_{\leq}$ is the data specification of directed graphs over a totally ordered set of nodes, and if $Grph_{\leq} \to Str$ is the encoding of graphs as adjacency matrices (*cf.* below), change of base allows any graph problem expressed over graphs-as-strings to be expressed directly over graphs.

Base change may be performed along a single instance $x : * \to S$, yielding a data specification X_x over $*$ (so the structure map gives no information). Now, standard categorical arguments show that the finite points of X_x are in bijection with the solutions of x, *i.e.*, those $v : * \to X$ yielding x when post-composed with the structure map of X. Therefore, if X is CNF (resp. Horn, Krom) fp over S, by pullback-stability and the observations preceding Lemma 19, the solutions of x correspond to solutions of a CNF (resp. Horn CNF, 2-CNF). We start seeing how algebraic restrictions influence the complexity of expressible problems.

Traditionally, computational complexity is based on Turing machines [Pap94, Gol08, AB09], which only take strings as input. Hence, complexity is "officially" defined only for decision problems on Str, and problems on

different data specifications are handled via encodings. These are often left implicit, but we may formalize them, as is done in descriptive complexity.

Encoding a finite structure as a binary string is only possible if the structure is totally ordered (this is well-known in descriptive complexity [Imm99]). We define an *ordered data type* as a data type $\operatorname{Spec}\mathcal{B}$ in which every object of $\mathcal{B}$ is totally ordered (as defined before Definition 15). We saw in Lemma 7 that total orders are stable under products and coproducts. They are stable under subobjects too: if $\varphi \rightarrowtail N$ and $\leq_N$ is a total order on N, then $x \leq_N y \wedge \varphi(x) \wedge \varphi(y)$ defines a total order on φ (by contrast, chains are not stable under subobject: zero, max and successor are not definable in general). Therefore, any finite theory $\mathbb{T}$ whose sorts are totally ordered gives rise to an ordered data type $\operatorname{Spec}\mathcal{F}[\mathbb{T}]$.

Let $S_\leq = \operatorname{Spec}\mathcal{B}_\leq$ be an ordered data type. By definition, $\mathcal{B}_\leq$ may be presented by a finite theory with m relation symbols $R_i \rightarrowtail \vec{A}_i$. Given such a presentation, we define an encoding morphism $e_{S_\leq} : S_\leq \to Str$ via the logical functor $\mathcal{F}[\mathbb{S}\mathrm{tr}] \to \mathcal{B}_\leq$ mapping N to $\sum_{i=1}^m \vec{A}_i$, $\leq$ to the order on $\sum_{i=1}^m \vec{A}_i$ and X to $\sum_{i=1}^m R_i$. For example, if $\mathbb{G}\mathrm{rph}_\leq$ is the theory of graphs with a total order on their nodes, which has one sort G, two relation symbols $\leq, E \rightarrowtail G^2$ with $\leq$ axiomatized as a total order, and if $Grph_\leq := \operatorname{Spec}\mathcal{F}[\mathbb{G}\mathrm{rph}_\leq]$, then the encoding is such that $\Gamma e_{Grph_\leq}$ maps a graph g with n nodes to the string of length n^2 representing the adjacency matrix of g. The order on the nodes of g is used to order the bits in the string.

For decoding, we would like to send G to $\sqrt{N}$, and this is where arithmetic is useful. Let $\mathbb{S}\mathrm{tr}_+$ be $\mathbb{S}\mathrm{tr}$ augmented with arithmetic. In any model of cardinality n of $\mathbb{S}\mathrm{tr}_+$, the formula

$$\exists z.\exists o.\mathrm{Z}(z) \wedge \mathrm{S}(z,o) \wedge \mathrm{TTimes}(x,x,o,z)$$

defines the empty set if n is not a perfect square, or the set $\{\sqrt{n}\}$ otherwise. This allows us to define a decoding morphism $d_{Grph_\leq} : Str_+ \to Grph_\leq$, where $Str_+ := \operatorname{Spec}\mathcal{F}[\mathbb{S}\mathrm{tr}_+]$ is arithmetic over Str. The idea may be generalized to all ordered data types (we omit the details).

We call *robust* a class of morphisms of **Data** which is stable under pullback and post-composition by arithmetic morphisms. By Lemma 19, fp, pfp, aHfp, aKfp and nuHfp are robust. If $\mathcal{K}$ is robust, then results over Str may be transfered to any other ordered data type $S_\leq$. Indeed, change of base along $e_{S_\leq}$ yields a data specification X' in $\mathcal{K}$ over $S_\leq$ from a data specification X in $\mathcal{K}$ over Str, and X' expresses the same problem as X, but over $S_\leq$. Conversely, a data specification X in $\mathcal{K}$ over $S_\leq$ induces, by change of base along $d_{S_\leq}$, a data specification in $\mathcal{K}$ over Str_+, where $a : Str_+ \to Str$ is some arithmetic morphism, so post-composing with a yields a data specification X' in $\mathcal{K}$ over Str. Notice that, since Γa is invertible on objects, X' expresses the same problem as X on strings.

3.3 Recursively enumerable problems

Lemma 20 *For any non-deterministic Turing machine M with read-only input tape on a binary alphabet and one read/write work tape on an arbitrary alphabet, there*

exists a finite extension $\mathbb{TM}_M$ *of* $\mathbb{S}\mathrm{tr}$ *such that:*

- $\mathbb{TM}_M$ *has two additional sorts* S *and* T*;*
- *the finite models of* $\mathbb{TM}_M$ *are accepting runs of* M *taking time* $|T|$ *and space* $|S|$ *(in the sense that the time and space are the size of the sets interpreting the sorts* T *and* S*, respectively);*
- *the projection of a finite model of* $\mathbb{TM}_M$ *on* $\mathbb{S}\mathrm{tr}$ *is the input of* M *for the run represented by that model.*

Proof. Recall that $\mathrm{Sort}(\mathbb{S}\mathrm{tr}) = \{N\}$. As per the statement of the lemma, we set $\mathrm{Sort}(\mathbb{TM}_M) := \{N, S, T\}$, and we proceed to describe the additional relation symbols and axioms of $\mathbb{TM}_M$ (with respect to $\mathbb{S}\mathrm{tr}$).

First of all, we equip N, S and T with the structure of chain sorts (Definition 6). Recall that $\mathbb{S}\mathrm{tr}$ already has a total order relation symbol $\leq \rightarrowtail N^2$; in the following, we will sometimes denote it by $\leq_N$, to distinguish it from the other total orders on S and T. Then, we add the following relation symbols, where σ ranges over the set Σ of symbols of the work tape of M and q over its set of states Q:

$$\mathrm{State}_q \rightarrowtail T \qquad \mathrm{Symb}_\sigma, \mathrm{wHead} \rightarrowtail T \times S \qquad \mathrm{iHead} \rightarrowtail T \times N$$

The additional axioms of $\mathbb{TM}_M$ (with respect to those already added) are as follows:

1. Axioms stating that the family of relations $(\mathrm{Symb}_\sigma)_{\sigma\in\Sigma}$ is a partial function $T \times S \to \Sigma$:
$$\mathrm{Symb}_\sigma(t, i) \wedge \mathrm{Symb}_{\sigma'}(t, i) \Rightarrow \bot$$
where σ and σ' range over Σ with $\sigma' \neq \sigma$.
2. Similar axioms stating that the family of relations (State_q) is a partial function $T \to Q$.
3. An axiom stating that iHead is a partial function $T \to N$:
$$\mathrm{iHead}(t, i) \wedge \mathrm{iHead}(t, i') \Rightarrow i' =_N i$$
4. A similar axiom stating that wHead is a partial function $T \to S$.
5. An axiom stating that, at time zero, the machine is in the initial state q_0, the work tape is filled with blank symbols $\square \in \Sigma$ and the heads are at the leftmost position of the respective tapes:
$$\begin{aligned}&\mathrm{Z}_T(t_0) \wedge \mathrm{Z}_N(i_0) \wedge \mathrm{Z}_S(j_0) \Rightarrow \\ &\mathrm{State}_{q_0}(t_0) \wedge \mathrm{Symb}_\square(t_0, j) \wedge \mathrm{iHead}(t_0, i_0) \wedge \mathrm{wHead}(t_0, j_0)\end{aligned}$$
6. Axioms encoding the transition function of M. In general, these are implications of the shape
$$\mathrm{State}_q(t) \wedge \mathrm{iHead}(t, i) \wedge \epsilon X(i) \wedge \mathrm{wHead}(t, j) \wedge \mathrm{Symb}_\sigma(t, j) \Rightarrow \bigvee_{i\in I} \varphi_i$$

with ϵ standing for $\neg$ or nothing. The antecedent says that the machine is in state q (which we may suppose to be non-final, otherwise there would be no transition), the input head reads a 0 (if $\epsilon = \neg$) or a 1 (if ϵ is nothing) and the work head reads σ. The consequent describes the non-deterministic choices that the machine takes in that situation: each φ_i corresponds to one such choice.

For example, suppose that the machine, when in state q, upon reading the simbol 0 on the input tape and the symbol σ on the work tape, non-deterministically transitions to state q_1, writes σ_1 and moves both the input and work head right, or transitions to state q_2, writes σ_2, moves the input tape left and keeps the work tape were it is. Then, the above axiom will be instantiated with $\epsilon := \neg$, $I := \{1, 2\}$ and

$$\begin{aligned}
&\varphi_1 := \mathrm{S}_T(t, t_{+1}) \wedge \mathrm{S}_N(i, i_{+1}) \wedge \mathrm{S}_S(j, j_{+1}) \Rightarrow \\
&\quad \mathrm{State}_{q_1}(t_{+1}) \wedge \mathrm{Symb}_{\sigma_1}(t_{+1}, j) \wedge \mathrm{iHead}(t_{+1}, i_{+1}) \wedge \mathrm{wHead}(t_{+1}, j_{+1}), \\
&\varphi_2 := \mathrm{S}_T(t, t_{+1}) \wedge \mathrm{S}_N(i_{-1}, i) \Rightarrow \\
&\quad \mathrm{State}_{q_2}(t_{+1}) \wedge \mathrm{Symb}_{\sigma_2}(t_{+1}, j) \wedge \mathrm{iHead}(t_{+1}, i_{-1}) \wedge \mathrm{wHead}(t_{+1}, j).
\end{aligned}$$

Notice that the general shape above may be amended for the "corner cases" in which the input head is at one of the extremities of the input string (the symbols Z_N and Max_N may be used to handle this) or the work head is at its leftmost position (the symbol Z_S may be used to handle this).

7. Axioms stating that the work tape is unchanged at positions where the head is not found:

$$\mathrm{Symb}_\sigma(t, j) \wedge \mathrm{wHead}(t, j') \wedge j \neq_S j' \wedge \mathrm{S}_T(t, t_{+1}) \Rightarrow \mathrm{Symb}_\sigma(t_{+1}, j)$$

for every $\sigma \in \Sigma$.

8. Axioms stating that the final configuration is in the accepting state q_{accept} (which we may assume to be unique) and that all previous configurations are not in a final state:

$$\mathrm{Max}_T(t) \Rightarrow \mathrm{State}_{q_{\mathrm{accept}}}(t) \qquad \mathrm{S}_T(t, t_{+1}) \wedge \mathrm{State}_{q_{\mathrm{fin}}}(t) \Rightarrow \bot$$

where q_{fin} ranges over all final states of M (including q_{accept}).

Point (1) of the lemma holds by definition. For what concerns points (2) and (3), a finite model of $\mathbb{TM}_M$ consists of a binary string x (the projection on Str) whose set of positions is N (for brevity, in the rest of the proof we will identify the sorts with the sets interpreting them), plus additional structure on it (a first and last element of N, and a successor on N), plus two non-empty finite chains S and T, and four functions

$$\begin{aligned}
\mathrm{State} &: T \to Q & \mathrm{iHead} &: T \to N \\
\mathrm{Symb} &: T \times S \to \Sigma & \mathrm{wHead} &: T \to S
\end{aligned}$$

Actually, axioms 1 through 4 only guarantee these functions to be partial, but totality may be proved by induction on the size of T: if $T = \{t_0\}$, then we conclude by axiom 5; if $T = \{t_0, \dots, t_{n-1}, t_n\}$, then by induction State, iHead and wHead are defined on t_{n-1} and $\mathrm{Symb}(t_{n-1}, j)$ is defined for all $j \in S$. Now, by axioms 8, $\mathrm{State}(t_{n-1})$ cannot be final, so some transition of M applies at time t_{n-1}, and axioms 6 ensure that State, iHead and wHead are defined at t_n. For the same reason, if j is the position of the work head at time t_{n-1}, $\mathrm{Symb}(t_n, j)$ is defined. The fact that it is defined for every other $j' \neq j$ is a consequence of axioms 7.

By definition, the function Symb is the "space-time" of M when run on x, in the sense that $\mathrm{Symb}(t, j)$ is the content of the j-th cell of the work tape at time t. Similarly, $\mathrm{State}(t)$, $\mathrm{iHead}(t)$ and $\mathrm{wHead}(t)$ are the state, position of the input head and position of the work head at time t. This is exactly the definition of a run of M on input x, of length $|T|$ and taking at most space $|S|$. Axioms 8 ensure that such a run is accepting. □

Theorem 21 *A decision problem on Str is recursively enumerable iff it is expressible by an fp data specification over Str.*

Proof. The implication from left to right is given by Lemma 20: since L is recursively enumerable, there exists a Turing machine M accepting it, so we may take $\mathrm{Spec}\,\mathcal{F}[\mathbb{TM}_M]$, which is of finite presentation over *Str* because $\mathbb{TM}_M$ is a finite extension of $\mathbb{S}\mathrm{tr}$.

For the converse, if X is of finite presentation over *Str*, then by definition $X \simeq \mathrm{Spec}\,\mathcal{F}[\mathrm{Str} + \mathbb{X}]$ with $\mathbb{X}$ finite and $L := \mathrm{im}\, X$ is equal to the image of the canonical projection $\mathrm{Mod}(\mathrm{Str} + \mathbb{X}) \to \mathrm{Mod}(\mathrm{Str})$. So $x \in L$ iff there exists finite additional stuff and structure v such that (x, v) verifies the finitely many axioms of $\mathbb{S}\mathrm{tr} + \mathbb{X}$. This is obviously decidable, so L is recursively enumerable. □

3.4 The Class NP

Let us start with an example of pfp data specification. Let $CNF := \mathrm{Spec}\,\mathcal{F}[\mathbb{CNF}]$, where $\mathbb{CNF}$ is the theory with two sorts V, C and two relation symbols $\mathrm{Pos}, \mathrm{Neg} \rightarrowtail V \times C$. A finite model of $\mathbb{CNF}$ represents a CNF: the sets interpreting V and C are the set of variables and clauses of the CNF, respectively; the relation $\mathrm{Pos}(x, c)$ (resp. $\mathrm{Neg}(x, c)$) holds when the variable x occurs positively (resp. negatively) in the clause c.

The famous *satisfiability problem* (Sat) asks, given a CNF, whether there exists an assignment to its variables making it true. We may express it by defining $\mathbb{A}\mathrm{sgn}$ as the extension of $\mathbb{CNF}$ adding two relation symbols $\mathrm{True} \rightarrowtail V$ and $\mathrm{Wit} \rightarrowtail C \times V$, plus one axiom saying that Wit is a functional relation and the axiom

$$\exists x.\mathrm{Wit}(c, x) \wedge ((\mathrm{Pos}(x, c) \wedge \mathrm{True}(x)) \vee (\mathrm{Neg}(c, x) \wedge \neg \mathrm{True}(x))).$$

The True predicate should be seen as a selection of which variables are true. The axioms state that Wit is a function assigning a variable x to each clause c, such that either x occurs positively in c, in which case it must be true, or it

occurs negatively in c, in which case it must be false. In other words, x is a witness that the clause c is satisfied. By definition, $\operatorname{Spec}\mathcal{F}[\mathbb{A}\mathrm{sgn}]$ is pfp over CNF, and it obviously expresses satisfiability.

Theorem 22 *A decision problem on Str is in* NP *iff it is expressible by a pfp data specification over Str.*

Proof. Let $L \in \mathsf{NP}$ and let M be the non-deterministic Turing machine deciding L. Lemma 20 gives us a data specification $X := \operatorname{Spec}\mathcal{F}[\mathbb{TM}_M]$ of finite presentation over Str expressing L, but it is not propositional because $\mathbb{TM}_M$ adds two sorts to $\mathbb{S}\mathrm{tr}$. However, this time we know that the running time of M is bounded by a polynomial P, which we may suppose to have non-negative integer coefficients. Hence, still by Lemma 20, it is enough to look for finite models of $\mathbb{TM}_M$ such that T and S are equal to $P(N)$ (remember that the size of N is the input size). But $P(N)$ is expressible within $\mathbb{S}\mathrm{tr}$ itself, so S and T are not needed! Let us formalize this.

Let $\mathbb{T}$ be the extension of $\mathbb{S}\mathrm{tr}$ adding only the chain sorts S, T, and let $U := \operatorname{Spec}\mathcal{F}[\mathbb{T}]$. Notice that the finite models of $\mathbb{T}$ encode triples (x, S, T) where $x \in \{0,1\}^*$ and S, T are finite chains. We have inclusions $\mathcal{F}[\mathbb{S}\mathrm{tr}] \to \mathcal{F}[\mathbb{T}] \to \mathcal{F}[\mathbb{TM}_M]$, so, reversing arrows, the structure map $X \to Str$ factors through U, and $p : X \to U$ (corresponding to the second inclusion) is pfp. Observe that Γp maps a finite model of $\mathbb{TM}_M$, that is, an accepting run of M, to the triple (x, S, T), where x is the input of the run and S, T are chains of length corresponding to the space and time of the run.

Let now $f : \mathcal{F}[\mathbb{T}] \to \mathcal{F}[\mathbb{S}\mathrm{tr}]$ be the logical functor mapping N, $\leq_N$ and X to "themselves" in $\mathcal{F}[\mathbb{S}\mathrm{tr}]$, and mapping S and T to $P(N)$. By Lemma 7, $P(N)$ is still a chain object, so f may map the chain structures of S and T to the chain structure of $P(N)$. Notice that Γf maps a binary string x to $(x, C_{P(|x|)}, C_{P(|x|)})$, where C_n is the chain of length n.

Now, by Lemma 19, base change along the above-defined $f : Str \to U$ yields $Y := X \times_U Str$ which is pfp over Str. We contend that $\operatorname{im} Y = L$. Since Γ preserves pullbacks, we know that an object of ΓY is a pair (v, x) such that $v \in \Gamma X$, $x \in \Gamma Str$ and $\Gamma(p)(v) = \Gamma(f)(x)$, and that $\Gamma(f^* p)$ projects (v, x) to x. But then $x \in \operatorname{im}\Gamma Y$ iff there exists an accepting run of M on input x using time and space $P(|x|)$, which holds exactly when $x \in L$.

For the converse implication, if X is pfp over Str, then by definition $X \simeq \operatorname{Spec}\mathcal{F}[\mathbb{S}\mathrm{tr} + \mathbb{X}]$ with $\mathbb{X}$ finite and adding no sorts, and $L := \operatorname{im} X$ is equal to the image of the canonical projection $\mathrm{Mod}(\mathbb{S}\mathrm{tr} + \mathbb{X}) \to \mathrm{Mod}(\mathbb{S}\mathrm{tr})$. A finite model of $\mathbb{S}\mathrm{tr} + \mathbb{X}$, which has a unique sort N, consists of a binary string of length n plus a constant number, say m, of relations of size $n^{k_1}, \ldots, n^{k_m}$ (where k_i are the arities of the relations), verifying a finite number of properties expressible by first-order formulas. Each of these may be deterministically checked in linear time in the size of the model. So $x \in L$ iff there exists a certificate of size polynomial in the length of x which may be verified in deterministic polynomial time, whence $L \in \mathsf{NP}$. □

Corollary 23 *A decision problem on Str is in* NP *iff it is expressible by a CNFfp data specification over Str.*

PROOF. From left to right, it is enough to scan the proof of Lemma 20 and observe that, once the sorts S and T are replaced by a polynomial in N, every axiom is equivalent to a formula of the required shape. The converse is immediate: CNFfp implies pfp. □

Theorem 24 (Cook [Coo71], Levin [Lev73]) SAT*is* NP*-complete.*

PROOF. By Theorem 22, $\mathrm{Spec}\,\mathcal{F}[\mathbb{A}\mathrm{sgn}]$ shows that $\mathrm{SAT} \in \mathsf{NP}$, so we need to prove that every problem in NP reduces to SAT. For this, we start by observing that every CNFfp data specification X over S induces a morphism $\varphi_X : S \to CNF$, defined as follows.

Let $S = \mathrm{Spec}\,\mathcal{B}$ and pick any $\mathbb{X}$ such that $X \cong \mathrm{Spec}\,\mathcal{B}[\mathbb{X}]$. Let $\{R_i \rightarrowtail \vec{A}_i\}_{i\in I}$ and $\{\psi_j\}_{j\in J}$ (with I and J finite) be the relation symbols and axioms of $\mathbb{X}$, respectively, with $\vec{x}_j : \vec{B}_j$ the free variables of ψ_j. Defining a morphism $S \to CNF$ means defining a logical functor $g : \mathcal{F}[\mathbb{CNF}] \to \mathcal{B}$, which amounts to choosing two objects gV, gC of $\mathcal{B}$ and two subobjects $g\mathrm{Pos}, g\mathrm{Neg} \rightarrowtail fV \times fC$. We first set

$$gV := \sum_{i\in I} \vec{A}_i \qquad\qquad gC := \sum_{j\in J} \vec{B}_j.$$

Now, every occurrence of R_i in ψ_j determines a function from the variable positions of R_i to $\vec{x}_j$. For example, if $R_i = R$ occurs as $R(x, x, y)$ and $\vec{x}_j = x^A, y^B, z^C$, then the function maps the first and second positions to x and the third position to y. Reading this function backwards, we get a morphism $t : \vec{B}_j \to \vec{A}_i$ composed of diagonal, terminal and identity maps. In the above example, the diagonal on A, the identity on B and the terminal map on C. Then, the pairing $\langle t, \mathrm{id}_{B_j}\rangle : \vec{B}_j \rightarrowtail \vec{A}_i \times \vec{B}_j$ is always a monomorphism, because it is composed of diagonal and identity maps (the terminal map on C paired with id_C gives id_C). Let now Pos_{ij} (resp. Neg_{ij}) be the subobject of $\vec{A}_i \times \vec{B}_j$ obtained by taking the copairing of all the subobjects defined above for every positive (resp. negatively) occurrence of R_i in ψ_j (hence absence of occurrences gives empty subobject). By taking the coproduct of all Pos_{ij}, we get

$$g\mathrm{Pos} := \sum_{(i,j)\in I\times J} \mathrm{Pos}_{ij} \rightarrowtail \sum_{(i,j)\in I\times J} \vec{A}_i \times \vec{B}_j = gV \times gC,$$

(by distributivity) and similarly for $g\mathrm{Neg}$, using the Neg_{ij}.

We are left with proving that $(\Gamma\varphi_X)^{-1}(\mathrm{SAT}) = \mathrm{im}\,X$, *i.e.*, that, for all $x \in \Gamma S$, $x \in \mathrm{im}\,X$ iff the CNF $\phi_x := \varphi_X \circ x$ is satisfiable. Given $x : * \to S$, base change of X along x gives a CNFfp data specification Φ_x over $*$. The key fact at this point, which may be shown by a direct calculation, is that ϕ_x is a CNF presenting Φ_x in the sense described before Lemma 19. Then, $x \in \mathrm{im}\,X$ iff x factors through the structure map of X via an arrow $* \to X$ (by definition) iff there is an arrow $* \to \Phi_x$ (by definition of pullback) iff ϕ_x is satisfiable (by the remark before Lemma 19). This suffices because, by Corollary 23, CNFfp data specifications cover all of NP. □

3.5 The Classes P and NL

Theorem 25 *A decision problem on Str is in* P *iff it is expressible by an aHfp data specification over Str.*

Proof. For the implication from left to right, we proceed exactly as in the proof of Theorem 22 and observe that, once the sorts S and T are replaced by $P(N)$, all the additional axioms are Horn. In fact, by inspecting the proof of Lemma 20, we see that the only non-Horn axioms are the totality axioms of the orders on S and T (not explicitly written in the proof) and axioms 6. The first ones disappear because the order on $P(N)$ is defined in terms of the order on N, and the second are Horn in this case because the machine is deterministic, and therefore all disjunctions are unary.

For the converse implication, let $\mathcal{F}[X_1, \dots, X_n]$ be the syntactic category of the theory consisting of n propositional atoms $X_i \rightarrowtail 1$, with no axiom, and let $\mathbb{A}^n := \operatorname{Spec} \mathcal{F}[X_1, \dots, X_n]$. Observe that $\Gamma \mathbb{A}^n \cong \{0,1\}^n$, the discrete category whose objects are binary strings of length n: a finite point $* \to \mathbb{A}^n$ is, dually, a choice of subobject of the singleton (a truth value) for each X_i, so we may regard X_i to be the i-th bit of a string. Now, for any chosen total order $\preceq$ on the object $[n] := 1 + \cdots + 1$ in $\mathcal{F}[X_1, \dots, X_n]$, we may define a logical functor $\mathcal{F}[\mathbb{S}\mathrm{tr}] \to \mathcal{F}[X_1, \dots, X_n]$ by sending N to $[n]$, $\leq$ to $\preceq$ and X to $(X_1 \dots X_n)$. This corresponds to a morphism $\iota_n : \mathbb{A}^n \to \mathit{Str}$ such that $\Gamma \iota_n$ sends $x \in \{0,1\}^n$ to the model of $\mathbb{S}\mathrm{tr}$ representing x with bit positions totally ordered by $\preceq$.

Let now X be aHfp over Str, and let $\Phi_n := X \times_{\mathit{Str}} \mathbb{A}^n$ be the base change along ι_n. By definition, Lemma 19 and because pullback commutes with composition, Y is Horn fp over some arithmetic data specification A_n over $\mathbb{A}^n$. Since the only totally ordered objects of $\mathcal{F}[X_1, \dots, X_n]$ are finite sets, A_n cannot add anything to $\mathbb{A}^n$, so Φ_n is equivalent to a Horn fp data specification over $\mathbb{A}^n$. Then, by the remarks preceding Lemma 19, Φ_n is presented by a Horn CNF among whose variables we have $X_1, \dots, X_n$, and, given x of length n, $x \in \operatorname{im} X$ iff Φ_n is satisfiable. How large is Φ_n? As a data specification over Str, X is equivalent to $\operatorname{Spec} \mathcal{F}[\mathbb{S}\mathrm{tr}][\mathbb{X}]$ for some $\mathbb{X}$. By duality, $\Phi_n = \operatorname{Spec}(\mathcal{F}[\mathbb{S}\mathrm{tr}][\mathbb{X}] \otimes_{\mathcal{F}[\mathbb{S}\mathrm{tr}]} \mathcal{F}[X_1, \dots, X_n])$ which, by Lemma 13, is equivalent to $\operatorname{Spec} \mathcal{F}[\iota_n(\mathbb{X})]$. Now, apart from arithmetic predicates and their axioms, $\mathbb{X}$ contains finitely many relation symbols $R_j \rightarrowtail N^{k_j}$ and axioms of the form

$$\chi_h := \varphi_h \vee \bigvee_{i \in I_h} \alpha_i^h$$

where φ_h is a formula of $\mathbb{S}\mathrm{tr}$ plus arithmetic and α_i^h vary over the R_j, all of them negated, except possibly one. Under ι_n, each R_j yields n^{k_j} propositional variables $R_j^{\vec{a}}$ with $\vec{a} \in [n]^{k_j}$, representing whether the interpretation of R_j in the model x holds or not on $\vec{a}$, and each χ_h yields a formula equivalent to

$$\widehat{\chi}_h := \bigwedge_{\vec{a} \in D_h} \bigvee_{i \in I_h} \alpha_i^h(\vec{a}),$$

where D_h is the set of those $\vec{a}$ not belonging to the interpretation of φ_h in the model x, and $\alpha_i^h(\vec{a})$ denotes $R_j^{\vec{a}}$ (or $\neg R_j^{\vec{a}}$) if $\alpha_i^h = R_j$ (or $\neg R_j$). Notice that, as

expected, each $\widehat{\chi}_h$ is a Horn CNF, and their conjunction, which presents Φ_n, is still a Horn CNF of size polynomial in n.

Since computing Φ_n may be done in deterministic polynomial time (in fact, even in deterministic logarithmic space) in n, and since solving a Horn CNF may be done in time linear in the size of the CNF [DG84], we have im $X \in \mathsf{P}$. □

The acquainted reader will have recognized in the above proof a rephrasing, using categorical language, of Grädel's argument for his descriptive characterization of P in terms of second-order Horn formulas [Grä92]. In light of this, the following is not surprising:

Theorem 26 *A decision problem on Str is in* coNL *iff it is expressible by an aKfp data specification over Str.*

Proof. The implication from left to right is quite technical and we omit it for conciseness. Instead, we show how to express the archetypal coNL-complete problem, namely the complement of reachability for directed graphs. This asks, given a non-empty directed graph and two nodes s and t, whether there is no path from s to t. Directed graphs with two singled-out nodes may be expressed by the theory stGrph extending Grph with two relation symbols $S, T \rightarrowtail N$ which are axiomatized to be singletons. Let $stGrph :=$ Spec $\mathcal{F}$[stGrph] and let stFlow be the theory extending stGrph with one relation symbol $C \rightarrowtail N$ and axioms

$$\neg S(s) \vee C(s) \qquad \neg E(x, x') \vee \neg C(x) \vee C(x') \qquad \neg T(t) \vee \neg C(t).$$

Spec $\mathcal{F}$[stFlow] is obviously Krom over $stGrph$. The finite models of stFlow are 2-colored graphs, with C specifying the color. The axioms force the coloring to have the following properties: node s is colored; if a node x is colored and there is an edge $x \to x'$, then x' too is colored; node t is not colored. It follows that a graph is colorable iff there is no path from s to t.

For the converse, we apply the same argument as in the proof of the right-to-left implication of Theorem 25, but this time the CNF Φ_n obtained (in deterministic logarithmic space) by change of base along $\iota_n : \mathbb{A}^n \to Str$ is a 2-CNF, the satisfiability of which is well-known to be in coNL [Pap94]. □

3.6 Non-Uniform Complexity Classes

Lemma 27 *Let Ord* := Spec $\mathcal{F}$[Ord]. *Every numeric morphism $S' \to S$ is the pullback of some numeric morphism $X \to Ord^k$ along some morphism $S \to Ord^k$.*

Proof. Let us first describe Ord^k. Let $\mathcal{O} := \mathcal{F}$[Ord]. By duality, the product of k copies of Ord in **Data** is the coproduct of k copies of $\mathcal{O}$ in **BoolLext**, *i.e.*, $\mathcal{O} \otimes_{\mathcal{F}} \cdots \otimes_{\mathcal{F}} \mathcal{O}$. By definition, this is $\mathcal{F}[\mathbb{T}]$ where $\mathbb{T}$ is the theory containing k totally ordered sorts $N_1, \ldots, N_k$ and nothing else.

Let S' be numeric over S, with $S =$ Spec $\mathcal{B}$ and $S' =$ Spec $\mathcal{B}[\mathbb{X}]$. By definition, $\mathbb{X}$ adds relations on powers of finitely many totally ordered objects $A_1, \ldots, A_k$ of $\mathcal{B}$. We may therefore define a morphism $S \to Ord^k$ as the dual of the logical functor $f : \mathcal{F}[\mathbb{T}] \to \mathcal{B}$ mapping N_i and its order to A_i and its

order. We then define $\mathbb{Y}$ to have the same relations as $\mathbb{X}$ but over N_i instead of A_i, and the same axioms, inducing a numeric $\mathcal{F}[\mathbb{T}]$-algebra $\mathcal{F}[\mathbb{T}][\mathbb{Y}]$. By construction, $\mathbb{Y} = f(\mathbb{X})$, so we conclude by Lemma 13. □

Theorem 28 *A decision problem on Str is in* P/poly *iff it is expressible by a nuHfp data specification over Str.*

Proof. Let M be a deterministic polynomial-time Turing machine with advice deciding a language in P/poly. We may see M as a machine with two input tapes, one for the actual input and one for the advice. We already know from Theorem 25 how to build an Hfp morphism encoding a deterministic polynomial-time Turing machine with a single input tape. Extending this to two input tapes is straightforward, so we only need to encode the advice. Suppose that this is of size n^k where n is the input size. We define $\mathbb{S}\mathrm{tr}'$ extending $\mathbb{S}\mathrm{tr}$ with one relation symbol $\mathrm{Adv} \rightarrowtail N^k$, axiomatized as follows. Let $\gamma_n(x_1, \ldots, x_n) := \mathrm{Z}(x_1) \wedge \mathrm{S}(x_1, x_2) \wedge \cdots \wedge \mathrm{S}(x_{n-1}, x_n) \wedge \mathrm{Max}(x_n)$. Then, for each $n \in \mathbb{N}$ we include in $\mathbb{S}\mathrm{tr}'$ the following axiom:

$$(\exists \vec{z}.\gamma_n(\vec{z})) \Rightarrow \bigwedge_{\vec{y} \in \{x_1, \ldots, x_n\}^k} (\gamma_n(\vec{x}) \Rightarrow \epsilon_{\vec{y}} \mathrm{Adv}(\vec{y})),$$

with $\epsilon_{\vec{y}}$ standing for either $\neg$ or nothing, according to whether the bit of the advice at position $\vec{y}$ is 0 or 1. For example, if we want to encode the fact that the advice for length n starts with 1, we use the clause $\gamma_n(\vec{x}) \Rightarrow \mathrm{Adv}(x_1, \ldots, x_1)$.

A finite model of $\mathbb{S}\mathrm{tr}'$ represents a pair of binary strings (x, a), with x of arbitrary length n and a of length n^k: the bits of x are specified by X, those of a by Adv. Observe that $\exists \vec{x}.\gamma_m$ holds exactly when $m = n$, and the consequent of the m-th axiom specifies the bits of a for length m. By definition, $Str' := \operatorname{Spec} \mathcal{F}[\mathbb{S}\mathrm{tr}']$ is numeric over Str. We precompose its structure map with the Hfp morphism $X \to Str'$ encoding M and we are done.

Let now X be nuHfp over Str. By definition, the structure map of X decomposes as $b \circ h$ with $h : X \to S$ Hfp and $b : S \to Str$ numeric. Let L be the problem on S expressed by h. Given $x : * \to Str$ of length n, $x \in \operatorname{im} X$ iff $x = b \circ v$ for some $v : * \to S$ such that $v \in L$. We claim that $v = (x, a)$ with a of size polynomial in n and depending only on n (not on x). This is enough to conclude that $\operatorname{im} X \in \mathsf{P/poly}$, because we already have $L \in \mathsf{P}$ from Theorem 25.

By Lemma 27, b is the pullback of some numeric $b_0 : Y \to Ord^k$ along some $f : Str \to Ord^k$. Let us write $Ord^k = \operatorname{Spec} \mathcal{O}_k$, so that $Y \simeq \operatorname{Spec} \mathcal{O}_k[\mathbb{X}]$ for some $\mathbb{X}$ as required by numeric morphisms. Since Γ preserves pullbacks, the objects of ΓX (which are the morphisms like v above) are pairs (x, a) with x an object of ΓStr and a an object of ΓY, such that $\Gamma f(x) \cong \Gamma b_0(a)$. Now, $\Gamma f(x)$ is a tuple of totally ordered sets $(A_1, \ldots, A_k)$, and the isomorphism says that these are in bijection with the totally ordered sets on which the relations of $\mathbb{X}$ are defined. The A_i are definable in $\mathbb{S}\mathrm{tr}$, so, if n is the length of x, they are sets of tuples of positions of x of size bounded by n^m, for some fixed m depending on f. On the other hand, a consists of a finite number of subsets $R \subseteq A_{i_R}^{q_R}$, one for each relation symbol of $\mathbb{X}$. The fact that Γb_0 is invertible,

as required by numeric morphisms, means that these R depend only on the sets A_i, not on x, because a must be reconstructed from $(A_1, \ldots, A_k)$. So a consists of a finite number of subsets of tuples of positions of x whose size is bounded by n^{mq}, where q is the maximum arity of the relation symbols of $\mathbb{X}$. This information may be encoded in a binary string of length polynomial in n, and, as we observed, it depends only on n and not on x, as claimed. □

It is not hard to see that the same argument works with propositional and Krom morphisms in place of Horn morphisms, yielding characterizations of NP/poly and NL/poly.

4 Perspectives

Characterizing complexity classes is an important first step but we should go further. Descriptive complexity has well-known tools, taking the form of various pebble games [Imm99], allowing one to establish lower bound results. These have been reformulated categorically [ADW17, DJR21, AR23]. Knowing if and how these reformulations interface with our work is an interesting question for the future.

The most intriguing aspect of our work, however, is that **Data** is defined as the dual of a category of "algebraic" objects. Dualities of the form algebra/geometry or logic/topology permeate mathematics, from Stone duality to the duality underlying algebraic geometry [Awo21, AJ21]. The latter is surprisingly related to our approach. Indeed, acquainted readers may have noticed that our terminology purposefully hints to the idea that we are doing algebraic geometry with Boolean lextensive categories in place of commutative rings. Ongoing work is showing that this is not just an analogy: we found a wider class of categories, which we call *ultrarings*, in which both commutative rings and Boolean lextensive categories appear as special cases. Remarkably, the (embedding of the) category $\mathcal{F}$ is the initial ultraring and corresponds to Durov's definition of the "field with one element" $\mathbb{F}_1$ [Dur07]. In light of this, this paper may be said to be working in the context of "idempotent affine geometry over $\mathbb{F}_1$"... but this is a story to be developed in another paper.

Acknowledgments

The second author would like to thank Tom Hirschowitz, Morgan Rogers and Carlos Simpson for stimulating and insightful discussions concerning this work.

References

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

[Abr22] Samson Abramsky. Structure and power: an emerging landscape. *Fundam. Informaticae*, 186(1-4):1–26, 2022.

[ADW17] Samson Abramsky, Anuj Dawar, and Pengming Wang. The pebbling comonad in finite model theory. In *Proceedings of LICS*, pages 1–12. IEEE Computer Society, 2017.

[AJ21] Mathieu Anel and André Joyal. Topo-logie. In *New Spaces for Mathematics and Physics*, pages 155–257. Cambridge University Press, 2021.

[AR23] Samson Abramsky and Luca Reggio. Arboreal categories: An axiomatic theory of resources. *Log. Methods Comput. Sci.*, 19(3), 2023.

[AS21] Samson Abramsky and Nihil Shah. Relating structure and power: Comonadic semantics for computational resources. *J. Log. Comput.*, 31(6):1390–1428, 2021.

[Awo21] Steven Awodey. Sheaf representations and duality in logic. In *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, pages 39–57. Springer, 2021.

[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.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158. ACM, 1971.

[CP20] Thomas Colcombet and Daniela Petrisan. Automata minimization: a functorial approach. *Log. Methods Comput. Sci.*, 16(1), 2020.

[DG84] William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.

[DJR21] Anuj Dawar, Tomás Jakl, and Luca Reggio. Lovász-type theorems and game comonads. In *Proceedings of LICS*, pages 1–13. IEEE, 2021.

[Dur07] Nikolai Durov. New approach to arakelov geometry, 2007.

[Fag74] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation, SIAM–AMS Proceedings*, volume 7, pages 43–73, 1974.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GPR20] Mai Gehrke, Daniela Petrisan, and Luca Reggio. Quantifiers on languages and codensity monads. *Math. Struct. Comput. Sci.*, 30(10):1054–1088, 2020.

[Grä92] Erich Grädel. Capturing complexity classes by fragments of second-order logic. *Theor. Comput. Sci.*, 101(1):35–57, 1992.

[Imm99] Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[Jak99] Bart Jakobs. *Categorical Logic and Type Theory*. Elsevier, 1999.

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

[Law63] William Lawvere. *Functorial Semantics of Algebraic Theories*. Ph.d. thesis, Columbia University, 1963.

[Law70] William Lawvere. Quantifiers and sheaves. In *Actes du Congres International Des Mathématiciens, Tome 1*, pages 329–334, 1970.

[Lev73] Leonid Levin. Universal search problems. *Problems of Information Transmission*, 9(3):115–116, 1973. (In Russian).

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.