Structure Meets Power Workshop

(Contributed Talks)

4 July 2022

https://www.cst.cam.ac.uk/conference/structure-meets-power-2022

Contents

A sheaf-theoretic approach to (P)CSPs an extended abstract

Adam Ó Conghaile *joint work with*
Samson Abramsky and Anuj Dawar

May 28, 2022

1 Introduction

The Structure and Power programme has introduced compelling compositional semantics for many approximations to homomorphism and isomorphism on finite structures which occur naturally in logic and computer science. This is largely done by the construction of game comonads such as those of Abramsky, Dawar and Wang [2], Abramsky and Shah [4], Abramsky and Marsden [3], and Montacute and Shah [7]. These all create formal links between structural decompositions which appear as coalgebras of the comonad and expressive power in some logical fragment which controls how the Kleilsi morphisms and isomorphisms approximate homomorphism and isomorphism of relational structures.

While these comonads cover a wide range of decompositions and logical fragments, from an algorithmic perspective, the homomorphism and isomorphism approximations related to the known comonads are relatively weak. Indeed, of those mentioned, the strongest is the pebbling comonad whose Kleilsi morphisms and isomorphisms correspond respectively to the success of the k-local consistency and Weisfeiler-Leman algorithms. These are both long known to be unable to capture all efficiently computable instances of CSP and structure isomorphism [5]. Attempts to make game comonads which go beyond this have presented other difficulties. The Hella comonad of Ó Conghaile and Dawar, for example, captures a fragment of logic extended by quantifiers which are not, in general, computable. A game comonad for Dawar and Holm's Invertible Maps equivalance [6] has long been sought but not found.

In this talk, we review the alternative presheaf-theoretic semantics for the pebble games which in as-yet-unpublished work by Abramsky, Ó Conghaile, and Dawar [1, 8, 9] and explore its connections to stronger algorithms for CSP and structure isomorphism and to the theory of promise constraint satisfaction problems.

2 Presheaves for constraint satisfaction and structure isomorphism

Informally, a presheaf is an object which organises local "behaviours" of some space X where the "local" contexts are given by some cover C of X. Global behaviours of X can then be constructed as "global sections" of the presheaf.

In [9], we see that any finite relational structure X can be made into such a space in a natural way using a *relational cover* C, which is defined as a collection of substructures of X whose union is X. This allows the definition of set-valued presheaves $\mathcal{H}^C_A$ and $\mathcal{I}^C_A$ which record the local (as defined in C) solutions to the homomorphism and isomorphism problems between A and X.

Fact 1. *For a structure X with relational cover C and a structure A of the same signature, the presheaf $\mathcal{H}^C_A$ (resp. $\mathcal{I}^C_A$) has a global section if and only if $X \to A$ (resp. $X \cong A$).*

Computing whether one of these presheaves has a global section is clearly just as difficult as the original constraint satisfaction problem so we now look at obstructions to global sections on presheaves which will give rise to efficiently computable approximations of these problems.

Given a set-valued presheaf $\mathcal{F}$ and a semiring $\mathbb{S}$, the functor $\mathbb{S}F$ which composes $\mathcal{F}$ with the free $\mathbb{S}$-semimodule functor is also a presheaf now valued in $\mathbb{S}$-semimodules. We have the following useful fact

Fact 2. *For a semiring $\mathbb{S}$ and a presheaf $\mathcal{F}$ then*

1. *if $\mathbb{S}\mathcal{F}$ has no global section then $\mathcal{F}$ has no global section, and*
2. *given a local section s of $\mathcal{F}$, if $\mathbb{S}\mathcal{F}$ has no global section extending s then $\mathcal{F}$ has no global section extending s.*

When $\mathcal{F}$ is $\mathcal{H}^C_A$ this gives us three tests approximating homomorphism. Test 1 checks if $\mathbb{S}\mathcal{F}$ has a global section, Test 2 checks for given s if it can be extended to a global section of $\mathbb{S}\mathcal{F}$ and Test 2^* repeatedly removes sections which fail Test 2 and succeeds if this process stabilises before removing all sections. When $\mathbb{S}$ admits efficient solving of linear equations then these tests can be done in PTIME in the size of C. We will consider the semirings $(\mathbb{B}, \mathbf{max})$, $(\mathbb{Q}^{\geq 0}, +)$ and $(\mathbb{Z}, +)$ In the next two sections we explore what these tests correspond to for two important choices of cover C.

3 Presheaves and k-consistency

Taking the relational cover of X to be the cover $X^{\leq k}$ of substructures of X of size k or less, the most compelling established connection is that Test 1 succeeds for $\mathbb{S} = \mathbb{B}$ if and only if the k-consistency algorithm accepts the instance (X, A). The cohomological k-consistency of [8] is defined to be Test 2^* of $\mathbb{S} = \mathbb{Z}$.

	Test 1	Test 2^* (singletons)	Test 2^* (all)
$\mathbb{B}$	Arc-consistency	SAC	CAC
$\mathbb{Q}^{\geq 0}$	BLP	SBLP	CBLP
$\mathbb{Z}$	AIP	SAIP	CAIP

Table 1: $C = C(X)$

4 Presheaves and PCSP algorithms

There is another cover where the landscape of algorithms is much more clearly mapped out. This is the cover of X be a substructure for each related tuple and each singleton. We call this cover C_X and note in Table 1 that this captures many algorithms studied in the Promise Constraint Satisfaction community.

References

[1] Abramsky, S. Notes on cohomological width and presheaf representations. Tech. rep., University College London, 2022.

[2] Abramsky, S., Dawar, A., and Wang, P. The pebbling comonad in finite model theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (2017), pp. 1–12.

[3] Abramsky, S., and Marsden, D. Comonadic semantics for guarded fragments. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021* (2021), IEEE, pp. 1–13.

[4] Abramsky, S., and Shah, N. Relating structure and power: Comonadic semantics for computational resources. *Journal of Logic and Computation 31*, 6 (Aug. 2021), 1390–1428.

[5] Cai, J.-Y., Fürer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identification. *Combinatorica 12*, 4 (Dec. 1992), 389–410.

[6] Dawar, A., and Holm, B. Pebble games with algebraic rules. In *Automata, Languages, and Programming* (Berlin, Heidelberg, 2012), A. Czumaj, K. Mehlhorn, A. Pitts, and R. Wattenhofer, Eds., Springer Berlin Heidelberg, pp. 251–262.

[7] Montacute, Y., and Shah, N. The pebble-relation comonad in finite model theory. *CoRR abs/2110.08196* (2021).

[8] Ó C., A. Cohomological k-consistency. Tech. rep., Cambridge, 2022.

[9] Ó C., A. Cohomology in constraint satisfaction and structure isomorphism. Tech. rep., Cambridge, 2022.

Data structures for topologically sound higher-dimensional diagram rewriting

Amar Hadzihasanovic
Tallinn University of Technology
& Quantinuum, Compositional Intelligence
amar@cs.ioc.ee

Diana Kessler
Tallinn University of Technology
diana-maria.kessler@taltech.ee

This work looks at the algorithms and complexity theory of higher-dimensional diagram rewriting. Higher-dimensional rewriting, as emergent from the theory of polygraphs [4] – see [6] for a survey – is founded on an interpretation of *rewrites as directed homotopies*. Proofs by diagrammatic rewriting can be used in the formalisation of homotopical algebra and higher category theory, provided that one has a *topologically sound* formal model, where diagrams admit a functorial interpretation as homotopies in cell complexes.

Beyond the formalisation of mathematics, it is interesting to look at higher-dimensional diagram rewriting as a model of computation in itself. *String diagram rewriting*, a form of 3-dimensional rewriting, is arguably the characteristic computational mechanism of applied category theory. It has been suggested [3] that even "classical" forms of rewriting are more faithfully represented as diagram rewriting: for example, term rewriting implemented as rewriting in monoidal categories with cartesian structure explicitates the "hidden costs" of copying and deleting terms.

Our contribution is to describe a computational implementation for diagrammatic sets [7, 8], a combinatorial alternative to polygraphs which, unlike them, is topologically sound. This is concretely implemented in a Python library for higher-dimensional diagram rewriting, `rewal`, for which a first release is imminent. In addition, we provide a complexity analysis for our implementation, showing that the basic operations in constructing diagrams and rewrites take (low-degree) polynomial time. This is a necessary step towards building diagrammatic models of computation with a reasonable cost model.

In its aims and inspiration, our work is closely related to that of Vicary, Bar, Dorn, and others on quasistrict [2] and later associative [5, 9] n-categories, serving as the foundation of the homotopy.io proof assistant. However, we claim that our model presents both theoretical advantages (proven topological soundness, functorial ties to well-established models of higher categories) and computational advantages (rewrites have better "locality" properties, which is beneficial both in terms of space efficiency and parallelisability).

The shape of a diagram in a diagrammatic set is encoded by its *face poset*, recording whether a cell is located in the boundary of another cell, together with *orientation* data which specifies whether an $(n-1)$-dimensional cell is in the *input* or *output* half of the boundary of an n-dimensional cell. We call the mathematical structure containing these data an *oriented graded poset*.

To represent an oriented face poset, we linearly order its elements in each dimension, so that each face is represented by a pair of integers, its *dimension* and its *position*. We encode the poset structure via an adjacency list view of its Hasse diagram, in the form of an array of arrays of pairs of sets of integers, where the pair of sets at index (n,k) contains the positions of the $(n-1)$-dimensional input, respectively output faces of (n,k). This representation of an oriented graded poset (up to isomorphism) is not unique: any permutation of the linear order on elements in each dimension leads to an equivalent representation.

Sub-diagrams are identified by *(downwards) closed subsets* of an oriented graded poset. Oriented

Submitted to:
ACT2022

graded posets support a purely combinatorial definition of the *input* ($\alpha := -$) and *output* ($\alpha := +$) *k-dimensional boundary* of a closed subset U, denoted by $\partial_n^\alpha U$.

In the theory of diagrammatic sets, shapes of diagrams form an inductively generated class of oriented graded posets, called regular *molecules* after Steiner [10]. We say a closed subset of an oriented graded poset of dimension n is *round* if, for all $k < n$, $\partial_k^+ U \cap \partial_k^- U = \partial_{k-1} U$.

Regular molecules The class of regular molecules is generated by the following clauses.

- (Point). The terminal oriented graded poset $\bullet$ is a regular molecule.
- (Atom). Let U, V be *round* regular molecules such that $\dim(U) = \dim(V)$ and, for all $\alpha \in \{+,-\}$, $\partial^\alpha U$ is isomorphic to $\partial^\alpha V$. Then $U \Rightarrow V$ is a regular molecule, where $U \Rightarrow V$ is the essentially unique oriented graded poset $U \Rightarrow V$ with the property that
  1. $U \Rightarrow V$ has a greatest element, and
  2. $\partial^-(U \Rightarrow V)$ is isomorphic to U, while $\partial^+(U \Rightarrow V)$ is isomorphic to V.
- (Paste). Let U, V be regular molecules and $k < \min(\dim(U), \dim(V))$, such that $\partial_k^+ U$ is isomorphic to $\partial_k^- V$. Then the pushout $U \#_k V$ of the span $\partial_k^+ U \hookrightarrow U$, $\partial_k^+ U \xhookrightarrow{\sim} \partial_k^- V \hookrightarrow V$ is a regular molecule.

A regular molecule is an *atom* if it has a greatest element; these are precisely the molecules whose final generating clause is (Point) or (Atom).

Correspondingly, we implement regular molecules as an inductive subclass of oriented graded posets with a nullary constructor point and partial binary constructors $\mathsf{atom}(-,-)$ and $\mathsf{paste}_k(-,-)$ for $k \in \mathbb{N}$. To implement these constructors, we need to:

1. compute input and output k-boundaries;
2. check if a closed subset is round;
3. determine if two regular molecules are isomorphic;
4. compute the pushout of a span of inclusions.

The first, second, and fourth of these admit straightforward algorithms of low-degree polynomial time complexity, that do not rely on any special properties of regular molecules.

The third task, however, does not have a straightforward solution. The isomorphism problem generalised to all oriented graded posets is equivalent to the *graph isomorphism* problem whose best known algorithm, due to Babai, runs in quasipolynomial time [1]. Our main technical contribution is to provide a polynomial time algorithm for the isomorphism problem restricted to regular molecules.

Our strategy for solving this isomorphism problem is to use a deterministic *traversal algorithm* that, given a regular molecule, outputs a unique ordering of the elements, which depends only on the intrinsic structure of the oriented graded poset and not on its representation. We can then put a regular molecule in "canonical form" by reordering elements in each dimension according to the traversal order. Two regular molecules are then isomorphic if and only if their canonical forms are equal.

For a fixed regular molecule U, let $|E_n|$ be the number of edges between n and $(n-1)$-dimensional elements in the Hasse diagram of U and $|U_n|$ be the number of n-dimensional elements of U. Then, $|U_{\max}| := \max_n |U_n|$, $|E_{\max}| := \max_n |E_n|$.

Theorem — *The traversal algorithm admits an implementation running in time*

$$O\big(|U|^2(|E_{\max}| \cdot \log|E_{\max}| + |U_{\max}| \cdot \log|U_{\max}|)\big).$$

Our last contribution is to provide a type theory $\mathsf{DiagSet}$ whose terms live "on top" of our representation of regular molecules: they are "shapes labelled with variables". This type theory is a formalisation

of our Python implementation, and allows us to prove an adequacy theorem for the intended semantics, in the form of a contravariant equivalence between the syntactic category of our type theory and a full subcategory of the category of diagrammatic sets.

The terms of DiagSet have a trivial equational theory, and in this sense they are "noncomputational": all the computation, which consists exclusively of computing and matching shapes, happens under the hood before a term is even created.

This is intended. Rather than a computational theory in itself, DiagSet is intended as a *substrate for computational theories* according to the paradigm of higher-dimensional rewriting. A term $t : r^- \Rightarrow r^+$ can be seen as a rewrite of the "lower-dimensional" term r^- to the term r^+, and the extension of t via the paste_k rules establishes how the rewrite can happen in a wider context. In this sense, every well-formed context in DiagSet contains its own internal computational theory on terms of each dimension.

For now, we have only scratched the surface of the algorithm and complexity theory of diagram rewriting in higher dimensions. In particular, we have not yet studied the problem of searching for a subdiagram within another diagram, whose solution is essential to any form of fully automated or assisted diagram rewriting. We plan to tackle this problem in future work.

References

[1] L. Babai (2016): *Graph isomorphism in quasipolynomial time [extended abstract]*. In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, ACM, pp. 684–697, doi:10.1145/2897518.2897542. Available at https://doi.org/10.1145%2F2897518.2897542.

[2] K. Bar & J. Vicary (2017): *Data structures for quasistrict higher categories*. In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, doi:10.1109/lics.2017.8005147. Available at https://doi.org/10.1109%2Flics.2017.8005147.

[3] G. Bonfante & Y. Guiraud (2009): *Polygraphic programs and polynomial-time functions*. *Logical Methods in Computer Science* 5(2), doi:10.2168/lmcs-5(2:14)2009. Available at https://doi.org/10.2168%2Flmcs-5%282%3A14%292009.

[4] A. Burroni (1993): *Higher-dimensional word problems with applications to equational logic*. *Theoretical Computer Science* 115(1), pp. 43–62, doi:10.1016/0304-3975(93)90054-w. Available at https://doi.org/10.1016%2F0304-3975%2893%2990054-w.

[5] C. Dorn (2018): *Associative n-categories*. Ph.D. thesis, University of Oxford.

[6] Y. Guiraud (2019): Rewriting methods in higher algebra. Thèse d'habilitation à diriger des recherches, Université Paris 7.

[7] A. Hadzihasanovic (2020): *Diagrammatic sets and rewriting in weak higher categories*. *arXiv preprint arXiv:2007.14505*.

[8] A. Hadzihasanovic (2021): *The smash product of monoidal theories*. In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, doi:10.1109/lics52264.2021.9470575. Available at https://doi.org/10.1109%2Flics52264.2021.9470575.

[9] D. Reutter & J. Vicary (2019): *High-level methods for homotopy construction in associative n-categories*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, doi:10.1109/lics.2019.8785895. Available at https://doi.org/10.1109%2Flics.2019.8785895.

[10] R. Steiner (1993): *The algebra of directed complexes*. *Applied Categorical Structures* 1(3), pp. 247–284, doi:10.1007/bf00873990. Available at https://doi.org/10.1007%2Fbf00873990.

A Categorical Approach to Descriptive Complexity Theory

Damiano Mazza
CNRS, LIPN, Université Sorbonne Paris Nord

Data specifications and complexity. Descriptive complexity [Imm99] teaches us that any structure that may be given as input to a computer program is a *finite structure* in the sense of model theory. More generally, one may consider finite models of first-order theories, up to isomorphism: for example a binary string is, up to iso, a finite model of the theory Str whose language contains a binary relation symbol $\leq$ and a unary relation symbol isOne, and whose axioms state that $\leq$ is a total order. "Up to iso" here means that the structures $\{0 < 1 < 2\}$ with $\mathsf{isOne} = \{2\}$ and $\{a < b < c\}$ with $\mathsf{isOne} = \{c\}$ define the same binary string, namely 001.

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

Introduced by Lawvere, categorical logic [Joh02] posits that logical and categorical structures come hand in hand: logical theories of a given kind (especially subsystems of first-order logic) correspond to categorical structures necessary to interpret them.

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

The category **BoolTh** of Boolean theories and their morphisms and the category **BoolCat** of Boolean categories and logical functors up to natural iso are related by an adjunction $\mathcal{F}[-] \dashv \mathrm{Lang}$, where $\mathrm{Lang} : \mathbf{BoolCat} \to \mathbf{BoolTh}$ associates with a category its *internal language*, whereas $\mathcal{F}[-] : \mathbf{BoolTh} \to \mathbf{BoolCat}$ associates with a theory its *syntactic category*. Intuitively, $\mathcal{F}[\mathbb{T}]$ is the category presented by the theory $\mathbb{T}$, much like $\mathbb{Z}[X_1, \ldots, X_n]/(P_1, \ldots, P_m)$ is the commutative ring presented by n generators and m polynomial equations $P_1, \ldots, P_m \in \mathbb{Z}[X_1, \ldots, X_n]$ (the generators correspond to the sorts and symbols of $\mathbb{T}$, whereas the polynomials to its axioms).

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

We therefore define $\mathcal{B}ool$ to be the category whose objects are finite Boolean theories (meaning with finitely many sorts, relation symbols and axioms) and whose morphisms $\mathbb{T} \to \mathbb{S}$ are logical functors $\mathcal{F}[\mathbb{T}] \to \mathcal{F}[\mathbb{S}]$, modulo natural isomorphism. It turns out that $\mathcal{F}[\mathbb{E}]$, where $\mathbb{E}$ is the empty theory, is equivalent to $\mathcal{F}$, which implies, by what mentioned above,

that a morphism $\mathbb{T} \to \mathbb{E}$ in $\mathcal{B}ool$ is the same thing as a finite model of $\mathbb{T}$ (remember that morphisms are up to iso). Hence, a morphism $f : \mathbb{T} \to \mathbb{S}$ of $\mathcal{B}ool$ induces a function from the finite models of $\mathbb{S}$ to the finite models of $\mathbb{T}$ (notice the contravariance): simply take the image of f via the presheaf $\mathcal{B}ool(-, \mathbb{E})$.

The fact that arrows in $\mathcal{B}ool$ go in the opposite direction with respect to how models are transformed suggests that all arrows should be reversed. Let us then define the category of *data specifications* as $\mathcal{D}ata := \mathcal{B}ool^{\mathrm{op}}$. When seeing a theory $\mathbb{T}$ as a data specification, *i.e.*, as an object of $\mathcal{D}ata$ rather than $\mathcal{B}ool$, we will write $\operatorname{Spec}\mathbb{T}$. $\mathcal{D}ata$ is a small category, with a nice structure: it is itself lextensive and, therefore, for what concerns finite categorical constructions, it behaves in many respects as a category of "spaces".

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

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

Theorem 1 *A subset of $\{0,1\}^*$ is recursively enumerable iff it is the finite image of a morphism $X \to \operatorname{Spec}\mathbb{S}\mathrm{tr}$ of $\mathcal{D}ata$.*

Some interesting connections with complexity may also be found. Call a morphism $f : X \to \operatorname{Spec}\mathbb{S}$ *relational* if $X \cong \operatorname{Spec}\mathbb{T}$ with $\mathbb{T}$ being an extension of $\mathbb{S}$ with no additional sorts and f being equal to the obvious projection $\operatorname{Spec}\mathbb{T} \to \operatorname{Spec}\mathbb{S}$ (dual to the inclusion $\mathbb{S} \to \mathbb{T}$) composed with the above iso. Call f *Horn* if, in addition, every axiom of $\mathbb{T}$ is of the form

$$\varphi \wedge R_1(\vec{x}_1) \wedge \cdots \wedge R_n(\vec{x}_n) \Rightarrow \psi$$

with φ a formula of $\mathbb{S}$, $R_1, \ldots, R_n$ relation symbols of $\mathbb{T}$ which are not of $\mathbb{S}$ (the case $n = 0$ is allowed) and ψ either a formula of $\mathbb{S}$ or of the form $S(\vec{y})$ with S a relation symbol of $\mathbb{T}$ not in $\mathbb{S}$. With these definitions, we have

Theorem 2 *A subset of $\{0,1\}^*$ is*

1. *in* NP *iff it is the finite image of a relational morphism over* $\operatorname{Spec}\mathbb{S}\mathrm{tr}$*;*
2. *in* P *iff it is the finite image of a Horn morphism over* $\operatorname{Spec}\mathbb{S}\mathrm{tr}$*.*

The interesting fact about Theorem 2 is that it does not use known descriptive characterizations of P or NP (such as Fagin's) but, rather, it uses the proof of Theorem 1 and yields those characterizations as a corollary. A similar characterization holds for NL (non-deterministic logspace) but its formulation is a bit more technical.

Reductions as pullback and complete problems via Yoneda. Summing up, a morphism $f : T \to S$ of $\mathcal{D}ata$ may be seen as

- a quantifier-free query from the finite points of T to the finite points of S;
- a recursively enumerable problem on the finite points of S: for any given finite point $x : \operatorname{Spec}\mathbb{E} \to S$ of S, seen as an instance of the problem, the finite points y of T such that $f \circ y = x$ (if any) are solutions to the instance x.

Combining these two viewpoints, we have that, given morphisms/problems $f : X \to S$ and $g : Y \to T$, a morphism/query $r : T \to S$ verifying that the pullback r^*f of f along r has the same finite image as g is exactly a quantifier-free reduction of g to f. If, additionally, $r^*f = g$, then r corresponds to a *parsimonious reduction* [Pap93] (*i.e.*, one that preserves the number of solutions of each instance).

We mentioned that data specifications are "spaces" of some sort. Intuitively, their points are (not necessarily finite) models of finite Boolean theories. Presheaves over $\mathcal{D}ata$ may therefore be seen as "generalized spaces" modeled on top of data specifications. The notions of finite point and finite image apply basically unchanged: a finite point of a presheaf $X : \mathcal{B}ool \to \mathbf{Set}$ is simply an element of $X(\mathbb{E})$, and the finite image of a morphism (natural transformation) $f : X \to Y$ is just the set-theoretic image of $f_{\mathbb{E}}$. The notion of (parsimonious) reduction-as-pullback also makes sense (pullbacks of morphisms of presheaves always exist).

Now, Theorem 2 tells us that, for any data specification S, describing a set of finite points of S of a certain complexity is the same as giving a morphism over S of a certain form. Therefore, it may be interesting to look for "classifying spaces" of such morphisms, that is, an object R such that morphisms on S of the desired form are in bijection with morphism $S \to R$. Although there may not be a classifying space for NP or P in $\mathcal{D}ata$ itself, such classifying spaces exist almost tautologically (thanks to Yoneda) as presheaves:

Theorem 3 *There is a morphism of presheaves $u : R_\bullet \to R$ (resp. $v : H_\bullet \to H$) such that, for every relational (resp. Horn) morphism $f : X \to S$ of $\mathcal{D}ata$, there exists a unique morphism $r_f : S \to R$ (resp. $r_f : S \to H$) such that f is the pullback of u (resp. v) along r_f.*

In other words u (resp. v) is a "generalized problem specification" such that every NP (resp. P) problem specification *uniquely* and parsimoniously quantifier-free-reduces to it. This strong form of completeness seems to be unachievable with "non-generalized" specifications.

Even more interestingly, u and v actually correspond to well-known complete problems:

- $R(\mathbb{E})$ (resp. $H(\mathbb{E})$) is in bijection with the set of CNFs (resp. Horn CNFs);
- $R_\bullet(\mathbb{E})$ (resp. $H_\bullet(\mathbb{E})$) is in bijection with the set of pairs (φ, σ) where φ is a CNF (resp. Horn CNF) and σ is a satisfying assignment for it;
- $u_{\mathbb{E}}$ and $v_{\mathbb{E}}$ are the functions forgetting the assignment.

Therefore, the finite images of u and v are just SAT and HORN SAT, the prototypical NP-complete and P-complete problems, respectively. Also, when repeating the construction with the characterization of NL, we obtain 2-SAT, a well-known NL-complete problem.

Whether this functorial approach is fruitful for descriptive complexity is unclear at present, this note presents the very surface of a rather large theory, whose development is ongoing. An interesting aspect is that Boolean categories present many similarities with commutative rings, so we are working in a context heavily reminiscent of "functorial" algebraic geometry. These analogies are rather encouraging, but too complex to mention here.

References

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

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

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

[Pap93] Christos H. Papadimitriou. *Computational Complexity*. Pearsons, 1993.

Path Predicate Modal Logic and its Comonadic Semantics

Gabriel Goren
Universidad de Buenos Aires & ICC, Argentina

The following work stems from an interest in exploring the applicability of the comonadic formalism [1] to data-aware logics. These are languages that reason on data-graphs studied in Database Theory, both exploring the topology of the structure and manipulating data-values. In particular, (in-)equality comparisons allow the expression of the data join, the most important construct of a query language. This interest lead us to consider simple fragments of CoreDataXPath [4]; in particular, we focused on DataGL, a very simple data-aware logic to reason on data-trees, studied in [2] from a proof-theoretical point of view.

DataGL captures a fragment of CoreDataXPath($\downarrow^+$), and can be seen as a bi-modal logic with the operators $\Diamond_=$ and $\Diamond_\neq$. However, multimodal logic cannot accommodate increasingly sophisticated fragments of CoreDataXPath such as those that express tests on intermediate steps of a path. This lead us to consider DataGL as a particular case of a more general family of logics that we call *Path Predicate Modal Logic* or PPML which, to the best of our knowledge, have not been investigated.

In this joint work with Santiago Figueira[1] we introduce PPML and begin the development of its basic theory from the point of view of comonadic semantics.

Path Predicate Modal Logic. We denote by σ a relational first-order signature with finite symbols. We always assume that σ contains a distinguished binary relation R_0 which will be treated as an accessibility relation, and denote the remaining symbols by $R_1, \ldots, R_m$. For each $i > 0$, R_i has arity $r_i \geq 1$. In any given σ-structure $\mathcal{A}$, we write $a \prec a'$ iff $R^{\mathcal{A}}(a, a')$. $|\mathcal{A}|$ denotes the underlying set of $\mathcal{A}$. We denote by $\mathsf{Struct}(\sigma)$ the category of first-order σ-structures, while $\mathsf{Struct}_*(\sigma)$ refers to the category of σ-structures $\mathcal{A}$ with a distinguished element or basepoint $a \in A$.

The *Path Predicate Modal Logic* (PPML) is defined by the grammar

$$\varphi := \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \Diamond\varphi \mid R_i \qquad (i \in \{1, \ldots, m\}).$$

Just like for Basic Modal Logic, the semantics of a formula φ of PPML is defined relative to a structure $\mathcal{A}$ and a specific point $a \in |\mathcal{A}|$. However, to evaluate φ at a it will be necessary to remember previous 'visited' points, making it natural to define the semantics with respect to a string of points.

Let $N(\sigma) = \max\{2, r_1, \ldots, r_m\}$ be the maximum arity of relations in σ. A *sliding valuation* over $|\mathcal{A}|$ is a string $s \in |\mathcal{A}|^{\leq N}$ of length at most N. The i-th symbol of s is denoted by $s(i)$ and the first k symbols of s by $s \restriction k$ (for $k \leq |s|$ where $|s|$ is the length of s). The concatenation of a symbol $a \in A$ and a word $w \in A^*$ is notated $a.w$. We define the semantics of PPML over a σ-structure $\mathcal{A}$ and a sliding valuation s as follows:

- $\mathcal{A}, s \models R_i$ iff $r_i \leq |s|$ and $(s(r_i), \ldots, s(1)) \in R_i^{\mathcal{A}}$.
- $\mathcal{A}, s \models \Diamond\varphi$ iff there is $a \in A$ such that $(s(1), a) \in R_0^{\mathcal{A}}$ and $\mathcal{A}, (a.(s \restriction n-1)) \models \varphi$.
- The Boolean connectives are defined as usual.

Finally, we say that $\mathcal{A}, a \models \varphi$ iff $\mathcal{A}, s_a \models \varphi$ where s_a is the one letter string (a).

The *positive fragment* of PPML, PPML$^\Diamond$, consists of the subset of negation-free formulas. We denote by PPML$_k$ and PPML$_k^\Diamond$ the fragments of PPML and PPML$^\Diamond$ consisting of formulas of modal depth $\leq k$.

[1] Universidad de Buenos Aires & ICC, Argentina

DataGL as a semantic restriction of PPML. In [2], the semantics of DataGL is given in terms of bi-modal Kripke structures instead of the original data tree semantics. Here we give an alternative semantics in terms of a class of first-order relational structures.

Consider the signature $\sigma_{\mathrm{DGL}} = \{R_0, R_=\} \cup \mathsf{PROP}$ where $R_=$ is binary and PROP is a countable set of unary symbols. A (pointed, first-order) *model of* DataGL is a pointed σ_{DGL}-structure $(\mathcal{A}, a)$ such that (i) $R_0^{\mathcal{A}}$ is the transitive irreflexive closure of the 'child' relation of a tree, and (ii) $R_=^{\mathcal{A}}$ is an equivalence relation. We denote by $\mathsf{Mod}_{\mathrm{DGL}}$ the full subcategory of $\mathsf{Struct}_*(\sigma_{\mathrm{DGL}})$ spanned by models of DataGL.

It is easily seen that any data-tree as in [2] can be encoded as a first-order model of DataGL in the above sense. Moreover there is a truth-preserving, two-way translation between DataGL-formulas and formulas of PPML over σ_{DGL}. Thus, the only essential difference between DataGL and PPML over σ_{DGL} is that the latter admits more general models. Although in this paper we concentrate on the comonadic description of PPML, a 'DataGL comonad' can be recovered as a relative comonad with respect to the inclusion $\mathsf{Mod}_{\mathrm{DGL}} \hookrightarrow \mathsf{Struct}_*(\sigma_{\mathrm{DGL}})$.

The bisimulation game for PPML. We consider the following variation of the Basic Modal Logic bisimulation game [3], played up to k rounds between $(\mathcal{A}, a_0)$ and $(\mathcal{B}, b_0)$. On round $i \in \{0, \dots, k\}$, the chosen positions (a_i, b_i) are checked for agreement on every n-ary relation $R \in \sigma \setminus \{R_0\}$: a_i and b_i agree iff $n \leq i$ and $R^{\mathcal{A}}(a_{i-n+1}, \dots, a_i) \iff R^{\mathcal{B}}(b_{i-n+1}, \dots, b_i)$. Duplicator loses if the positions don't agree in this new sense. This defines the *k-round bisimulation game* $\mathcal{G}_k((\mathcal{A}, a_0), (\mathcal{B}, b_0))$.

We also consider the *k-round simulation game* $\mathcal{G}_k^{\rightarrow}((\mathcal{A}, a_0), (\mathcal{B}, b_0))$, where Spoiler can only play on $\mathcal{A}$ and Duplicator on $\mathcal{B}$. Duplicator wins the i-th round if (a_i, b_i) agree as above except that $\iff$ is replaced by $\implies$.

▶ **Proposition 1.** *Two pointed structures $(\mathcal{A}, a), (\mathcal{B}, b) \in \mathsf{Struct}_*(\sigma)$ satisfy exactly the same $\mathrm{PPML}_k^{\Diamond}$-formulas iff there exist winning strategies for Duplicator both in $\mathcal{G}_k^{\rightarrow}((\mathcal{A}, a), (\mathcal{B}, b))$ and in $\mathcal{G}_k^{\rightarrow}((\mathcal{B}, b), (\mathcal{A}, a))$. They satisfy exactly the same PPML_k-formulas iff there exists a winning strategy for Duplicator in $\mathcal{G}_k((\mathcal{A}, a), (\mathcal{B}, b))$.*

The PPML Comonad. For the sake of brevity, we define the comonad $\mathbb{C}_k$ on $\mathsf{Struct}_*(\sigma)$ by pointing out its relationship to the EF comonad [1]. Let $\mathbb{E}_k : \mathsf{Struct}_*(\sigma) \to \mathsf{Struct}_*(\sigma)$ be the lifting of the Ehrenfeucht-Fraïssé comonad to pointed structures by letting the distinguished point of $\mathbb{E}_k(\mathcal{A}, a)$ be $[a]$. The underlying set of $\mathbb{C}_k(\mathcal{A}, a)$ is defined to be the subset of $\mathbb{E}_k(\mathcal{A}, a)$ consisting of all sequences $[a_1 \dots a_\ell] \in |\mathcal{A}|^{\leq k}$ such that $a_i \prec a_{i+1}\, \forall i \in \{1 \dots \ell - 1\}$; and for each n-ary $R \in \sigma$, $R^{\mathbb{C}_k(\mathcal{A},a)}(s_1, \dots, s_n)$ iff (a) $s_i \prec s_{i+1}$ for all i in range, and (b) $R^{\mathcal{A}}(\varepsilon_{\mathcal{A}}(s_1), \dots, \varepsilon_{\mathcal{A}}(s_n))$ (where ε is the counit of $\mathbb{E}_k$).

▶ **Proposition 2.** *$\mathbb{C}_k$ as above is the action on objects of a sub-comonad of $\mathbb{E}_k$ on $\mathsf{Struct}_*(\sigma)$, which is moreover idempotent*[2]*.*

It can also be shown that $\mathbb{C}_k$ is a sub-comonad of the (lifted) pebbling comonad $\mathbb{P}_N$ [1] with parameter $N = N(\sigma)$ independent of k (in the sense that there is a comonad morphism $\mathbb{C}_k \to \mathbb{P}_N$ with monic components), witnessing the fact that PPML_k translates to a fragment of both bounded quantifier rank and bounded variable number First Order Logic.

[2] It is then standard that if a pointed structure admits a coalgebra structure, it is unique. Thus we refer to pointed structures admitting $\mathbb{C}_k$-coalgebra structure as simply '$\mathbb{C}_k$-coalgebras'.

▶ **Proposition 3.** *$(\mathcal{A}, a) \in \mathsf{Struct}_*(\sigma)$ is a $\mathbb{C}_k$-coalgebra iff (1) $(|\mathcal{A}|, R_0^{\mathcal{A}})$ is a tree of height $\leq k$ rooted in a and (2) for each n-ary $R \in \sigma$, if $a_1, \ldots, a_n \in |\mathcal{A}|$ are such that $R^{\mathcal{A}}(a_1, \ldots, a_n)$, then $a_1 \prec \cdots \prec a_n$, i.e. $[a_1 \ldots a_n]$ is the unique path of length n ending in a_n.*

Expressivity results. We first consider the positive fragment of PPML.

▶ **Proposition 4.** *Given $(\mathcal{A}, a), (\mathcal{B}, b) \in \mathsf{Struct}_*(\sigma)$, there is a bijective correspondence between homomorphisms[3] $\mathbb{C}_k(\mathcal{A}, a) \to (\mathcal{B}, b)$ and the set of winning strategies for Duplicator in the k-round simulation game $\mathcal{G}_k^{\rightarrow}((\mathcal{A}, a), (\mathcal{B}, b))$. Thus, $(\mathcal{A}, a), (\mathcal{B}, b) \in \mathsf{Struct}_*(\sigma)$ satisfy exactly the same $\mathrm{PPML}_k^{\Diamond}$-formulas iff there exist homomorphisms $\mathbb{C}_k(\mathcal{A}, a) \to (\mathcal{B}, b)$ and $\mathbb{C}_k(\mathcal{B}, b) \to (\mathcal{A}, a)$.*

With respect to full PPML, we have the following result which relies on [1, Thm. 10.1].

▶ **Theorem 5.** *Two structures $(\mathcal{A}, a), (\mathcal{B}, b) \in \mathsf{Struct}_*(\sigma)$ satisfy exactly the same PPML_k-formulas iff there exists a span of strong surjective homomorphisms $(\mathcal{P}, p) \to \mathbb{C}_k(\mathcal{A}, a)$ and $(\mathcal{P}, p) \to \mathbb{C}_k(\mathcal{B}, b)$ with some $\mathbb{C}_k$-coalgebra $(\mathcal{P}, p)$ as common domain.*

Relationship between PPML and Basic Modal Logic. Given $\sigma = \{R_0, R_1, \ldots, R_m\}$ as before, we define a new signature $\widetilde{\sigma} = \{R_0, \widetilde{R}_1, \ldots, \widetilde{R}_m\}$ where each symbol $\widetilde{R}_j$ is unary. Given a $\mathbb{C}_k$-coalgebra $(\mathcal{A}, a)$ (in particular it is a σ-structure), let $T(\mathcal{A}, a) \equiv (\widetilde{\mathcal{A}}, a)$ be the $\widetilde{\sigma}$-structure with universe $|\widetilde{\mathcal{A}}| = |\mathcal{A}|$ and the following relations: $R_0^{\widetilde{\mathcal{A}}} = R_0^{\mathcal{A}}$, and for n-ary $R \in \sigma \setminus \{R_0\}$, and $a \in |\mathcal{A}|$, $\widetilde{R}^{\widetilde{\mathcal{A}}}(a)$ iff there exist $a_1, \ldots, a_n = a$ such that $R^{\mathcal{A}}(a_1, \ldots, a_n)$.

▶ **Proposition 6.** *T as given above defines the action on objects of a functor $T : \mathsf{EM}(\mathbb{C}_k) \to \mathsf{EM}(\mathbb{M}_k)$, where $\mathbb{M}_k : \mathsf{Struct}_*(\widetilde{\sigma}) \to \mathsf{Struct}_*(\widetilde{\sigma})$ is the Modal Comonad over signature $\widetilde{\sigma}$, which acts on homomorphisms as the identity on the underlying set-functions[4].*

Moreover this functor is fully faithful, and its image is the full subcategory of $\mathsf{EM}(\mathbb{M}_k)$ spanned by $\mathbb{M}_k$-coalgebras $(\mathcal{A}, a)$ that satisfy the folowing condition: for any $a' \in |\mathcal{A}|$ and n-ary relation $R \in \sigma$, $\widetilde{R}^{\mathcal{A}}(a')$ implies that the distance from a to a' is at least $n - 1$.

As a corollary of the idempotency of $\mathbb{C}_k$ and Theorem 5, all $(\mathcal{A}, a) \in \mathsf{Struct}_*(\sigma)$ satisfy exactly the same PPML_k-formulas as their 'unravelings' $\mathbb{C}_k(\mathcal{A}, a)$. Thus, a formula $\varphi \in \mathrm{PPML}_k$ is satisfiable iff it is satisfied by a $\mathbb{C}_k$-coalgebra (we could call this the 'coalgebra-model property', which in Basic Modal Logic specializes to the tree-model property). From this, and using the fact that T preserves and reflects open pathwise embeddings, it follows that satisfiability for PPML_k reduces linearly to satisfiability for the k-bounded modal depth fragment of Basic Modal Logic. In particular, satisfiability for PPML is PSPACE-complete.

References

1 Samson Abramsky and Nihil Shah. Relating structure and power: Comonadic semantics for computational resources. *Journal of Logic and Computation*, 31(6):1390–1428, 2021.

2 David Baelde, Simon Lunel, and Sylvain Schmitz. A sequent calculus for a modal logic on finite data trees. In *CSL*, volume 62 of *LIPIcs*, pages 32:1–32:16, 2016.

3 Patrick Blackburn, Johan van Benthem, and Frank Wolter. *Handbook of modal logic*. Elsevier, 2006.

4 Mikoaj Bojańczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM (JACM)*, 56(3):1–48, 2009.

[3] All homomorphisms are assumed to be basepoint-preserving.
[4] Note that in our context $\mathbb{C}_k$ and $\mathbb{M}_k$ are endofunctors of different categories.

How to Compose Shortest Paths

Jade Master

1 Acknowledgements

Thank you to Benjamin Bumpus, Jules Hedges, and Matteo Capucci who all contributed to this work.

2 A Composition Problem

Divide and conquer is an effective method for reducing the computation time of many algorithms. With this strategy, a problem may be broken up into subdomains, the problem is solved on the subdomains, and then joined together to obtain the final solution. This last step of recombination is the main topic of study for this paper and may be phrased in categorical terms. We work in the following abstract setting

$$\mathsf{Set} \underset{\mathrm{R}}{\overset{\mathrm{L}}{\rightleftarrows}} \mathrm{C} \xrightarrow{\mathrm{F}} \mathrm{D}$$

where C is a category whose objects are systems broadly construed. The left adjoint L is understood as the "discrete system" functor and the right adjoint is the "underlying set functor". The functor $\mathrm{F}\colon \mathrm{C} \to \mathrm{D}$ represents a computational problem with D representing some category of solutions. In this setting, a pushout

$$\begin{array}{ccc} \mathrm{M} +_{\mathrm{LX}} \mathrm{N} & \xleftarrow{\mathrm{i}} & \mathrm{N} \\ {\scriptstyle \mathrm{j}}\uparrow & \lrcorner & \uparrow \\ \mathrm{M} & \longleftarrow & \mathrm{LX} \end{array}$$

in the category C represents the gluing of a system M with a system N along a set of boundaries X. We may now state the crucial step of the divide and conquer method in categorical terms.

Definition 2.1. *The* **composition problem** *for* F *asks: given* F(M) *and* F(N) *as inputs, find* $\mathrm{F}(\mathrm{M} +_{\mathrm{LX}} \mathrm{N})$.

In this paper, C, L, R, and F will be chosen as follows.

Definition 2.2. *Let* $[0,\infty]$ *be the semiring of positive real numbers including infinity. Addition is given by* min *and multiplication is given by* $+$*. A* $[0,\infty]$*-graph with vertices* X *is a function* $\mathrm{M}\colon \mathrm{X} \times \mathrm{X} \to [0,\infty]$*. For a function* $\mathrm{f}\colon \mathrm{X} \to \mathrm{Y}$ *and a matrix* $\mathrm{M}\colon \mathrm{X} \times \mathrm{X} \to [0,\infty]$*, the pushforward matrix* $\mathrm{f}_*(\mathrm{M})\colon \mathrm{Y} \times \mathrm{Y} \to [0,\infty]$ *is defined by*

$$f_*(M)(i,j) = \sum_{(a,b)\in(f\times f)^{-1}(i,j)} M(i,j)$$

Note that confusingly, the zero element for $[0,\infty]$ is ∞ and the one element is 0.

Definition 2.3. *For* $[0,\infty]$*-graphs* $\mathrm{M}\colon \mathrm{X}\times\mathrm{X} \to [0,\infty]$ *and* $\mathrm{N}\colon \mathrm{Y}\times\mathrm{Y} \to [0,\infty]$ *a function* $\mathrm{f}\colon \mathrm{X} \to \mathrm{Y}$ *is a morphism of* $[0,\infty]-$graphs *if the pushforward satisifes* $\mathrm{f}_*(\mathrm{M}) \geq (\mathrm{N})$*. This defines a category* $[0,\infty]\mathsf{Graph}$ *of weighted graphs and their morphisms.*

Proposition 2.1. *There is an adjunction*

$$\mathsf{Set} \underset{\mathrm{R}}{\overset{\mathrm{L}}{\rightleftarrows}} [0,\infty]\mathsf{Graph}$$

whose left adjoint sends a set X *to the* $[0,\infty]$*-matrix with vertices given by* X *and every entry given by* $\mathrm{LX(i,j)} = \infty$*. The right adjoint sends a* $[0,\infty]$*-graph to its underlying set of vertices.*

The functor F in our setup will compute the shortest paths on a weighted graph. This problem can be understood algebraically. Let $\mathsf{Mat}(\mathrm{X},[0,\infty])$ be the semiring of $[0,\infty]$-graphs over a set X. In this semiring, addition is given by pointwise minimum and multiplication is given by

$$M \cdot N(i,j) = \min_{k\in X}\{M(i,k) + N(k,j)\}$$

which is the usual matrix multiplication valued in min-plus semiring.

Proposition 2.2. *The shortest paths in a* $[0,\infty]$*-graph* M*, are given by the matrix exponential*

$$F(M) = \sum_{n\geq 0} M^n$$

in the semiring $\mathsf{Mat}(\mathrm{X},[0,\infty])$.

This formula may be interpreted as follows: The matrix power $\mathrm{M^n}$ has entries given by the shortest paths in exactly n steps. Therefore, summing these values over all $\mathrm{n} \geq 0$ gives the shortest paths in any number of steps.

The algebraic path problem generalizes the shortest path problem by allowing the semiring to vary. For different choices of semiring, the algebraic path problem asks for most likely paths, maximum capacities, connectivity, and more substantially the language of an NFA. As explained in [2], when S is a quantale, there is an adjunction

$$\mathsf{SGraph} \underset{\mathrm{U_S}}{\overset{\mathrm{F_S}}{\rightleftarrows}} \mathsf{SCat}$$

between the category of S-enriched graphs and S-enriched categories. The left adjoint of this adjunction sends an S-enriched graph to the solution of its algebraic path problem. For each left adjoint $\mathrm{F_S}$, there there is an instance of the composition problem. In this paper, we solve the composition problem when $\mathrm{F_S} = \mathrm{F}$ i.e. the special case when S is the semiring $[0, \infty]$. Before doing this, we must state a relevant result.

Proposition 2.3. *The pushout of $[0,\infty]$-graphs is given by the pointwise sum*

$$M +_{LX} N \cong i_*(M) + j_*(N)$$

In general, we use boldface to indicate the pushforward of a weighted graph along an implied function. For example, let $\mathbf{M}$ and $\mathbf{N}$ denote the above pushforwards of M and N. Note that pushforward commutes with F in the sense that $\mathrm{F}(\mathrm{i}_*(\mathrm{M})) = \mathrm{i}_*(\mathrm{F}(\mathrm{M}))$ and $\mathrm{F}(\mathrm{j}_*(\mathrm{N})) = \mathrm{j}_*(\mathrm{F}(\mathrm{N}))$.

3 A Gainful Solution

Because F is a left adjoint we have reason to be optimistic about a solution to its composition problem. Left adjoints preserve pushouts so there is an isomorphism

$$F(M +_{LX} N) \cong F(M) +_{F(LX)} F(N)$$

where the pushout on the right is computed in the category of $[0,\infty]$-enriched categories. In [2], it was shown that this pushout may be computed as $\mathrm{F}(\mathrm{U}(\mathrm{F}(\mathrm{M})) +_{\mathrm{X}} \mathrm{U}(\mathrm{F}(\mathrm{N})))$. Although this gives a solution to the composition problem for F it is not a practical one because the final application of F is very expensive in terms of computation time. The main result of this paper is a more practical expression for this pushout.

Theorem 3.1. *For pushouts and* F *as above we have that*

$$F(M +_{LX} N) \cong$$

$$\sum_{n \le |X|} \underbrace{\mathbf{F}(\mathbf{M})\mathbf{F}(\mathbf{N})\mathbf{F}(\mathbf{M})\dots}_{n \text{ times}} + \underbrace{\mathbf{F}(\mathbf{N})\mathbf{F}(\mathbf{M})\mathbf{F}(\mathbf{N})\dots}_{n \text{ times}}$$

where $\mathbf{F}(\mathbf{M})$ *and* $\mathbf{F}(\mathbf{N})$ *denote the pushforwards* $\mathrm{i}_*(\mathrm{F}(\mathrm{M}))$ *and* $\mathrm{j}_*(\mathrm{F}(\mathrm{N}))$ *respectively.*

Proof. Because $\mathrm{M} +_{\mathrm{LX}} \mathrm{N} = \mathbf{M} + \mathbf{N}$ we have that

$$\begin{aligned} \mathrm{F}(\mathrm{G} +_{\mathrm{K}} \mathrm{H}) &= \sum_{\mathrm{n} \ge 0} (\mathrm{G} +_{\mathrm{K}} \mathrm{H})^{\mathrm{n}} \\ &\cong \sum_{\mathrm{n} \ge 0} (\mathbf{M} + \mathbf{N})^{\mathrm{n}} \\ &= \sum_{\mathrm{n} \ge 0} \sum_{\tilde{\mathrm{v}} \in \mathbf{2}^{\mathrm{n}}} \mathrm{X}_{\mathrm{v}_1} \dots \mathrm{X}_{\mathrm{v}_\mathrm{n}} \end{aligned} \tag{1}$$

where

$$X_{v_i} = \begin{cases} \mathbf{M} & \text{if } \mathrm{v_i} = 0 \\ \mathbf{N} & \text{if } \mathrm{v_i} = 1 \end{cases}$$

and $\mathbf{2}^{\mathrm{n}}$ is the set of boolean vectors $\tilde{\mathrm{v}} = (\mathrm{v}_1, \mathrm{v}_2, \dots, \mathrm{v}_\mathrm{n})$ with length n. The last equality is true in any semiring and is a well-known as the generalization of the binomial theorem for non-commutative elements. We define a function $\gamma\colon \sum_{\mathrm{n} \ge 0} \mathbf{2}^{\mathrm{n}} \to \mathbb{N}$ where sum now indicates the coproduct of sets. $\gamma(\tilde{\mathrm{v}})$ is called the crossing number of $\tilde{\mathrm{v}}$ and it is equal to the number of times $\tilde{\mathrm{v}}$ switches between 0 and 1. It may be defined by induction on the vector length i.e.$\gamma(\tilde{\mathrm{v}}) = 0$ if $\mathrm{length}(\tilde{\mathrm{v}}) = 0$ or 1 and $\gamma((\mathrm{v}_1, \mathrm{v}_2, \dots, \mathrm{v}_\mathrm{n})) =$

$$\gamma((v_1, v_2, \dots, v_{n-1})) + \begin{cases} 1 & \text{if } \mathrm{v_n} = \mathrm{v_{n-1}} \\ 0 & \text{if } \mathrm{v_n} \neq \mathrm{v_{n-1}} \end{cases}$$

The sum of Expression 1 may be repartitioned using the crossing numbers to obtain

$$= \sum_{n \ge 0} \Big[\underbrace{\sum_{\mathrm{i} \ge 1} \mathbf{M}^{\mathrm{i}} \sum_{\mathrm{i} \ge 1} \mathbf{N}^{\mathrm{i}} \dots}_{n \text{ times}} + \underbrace{\sum_{\mathrm{i} \ge 1} \mathbf{N}^{\mathrm{i}} \sum_{\mathrm{i} \ge 1} \mathbf{M}^{\mathrm{i}} \dots}_{n \text{ times}} \Big] \tag{2}$$

The last equality accounts for all terms with crossing number n. They may either start with $\mathbf{M}$ or $\mathbf{N}$ and then continue for any nonzero number of terms. Note that the maximum crossing number which may contribute to this sum is $|\mathrm{X}|$. This is because a shortest path with crossing number greater than $|\mathrm{X}|$ would cross at least one vertex in $|\mathrm{X}|$ more than once and could therefore be shortened by removing a loop. Therefore Expression 2 is equal to

$$= \sum_{n \le |X|} \Big[\underbrace{\sum_{\mathrm{i} \ge 1} \mathbf{M}^{\mathrm{i}} \sum_{\mathrm{i} \ge 1} \mathbf{N}^{\mathrm{i}} \dots}_{n \text{ times}} + \underbrace{\sum_{\mathrm{i} \ge 1} \mathbf{N}^{\mathrm{i}} \sum_{\mathrm{i} \ge 1} \mathbf{M}^{\mathrm{i}} \dots}_{n \text{ times}} \Big] \tag{3}$$

The difference between the above sum and the desired expression is that each $\mathbf{F}(\mathbf{M})$ and $\mathbf{F}(\mathbf{N})$ include paths of length 0. This causes each term of the desired result to also include terms with lower crossing number. However, because $[0,\infty]$ is idempotent adding these terms twice does not affect the sum. □

This theorem gives an algorithm for the composition problem for F: simply plug $\mathbf{F}(\mathbf{M})$ and $\mathbf{F}(\mathbf{N})$ into the isomorphism of Theorem 3.1. Next we use this isomorphism to find single source single target shortest paths.

4 A Compositional Algorithm

$\mathbf{F}(\mathbf{M})$ and $\mathbf{F}(\mathbf{N})$ may be broken into the block matrices

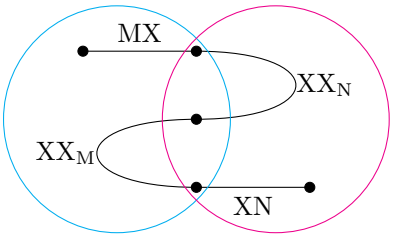
$$\mathbf{F}(\mathbf{M}) = \begin{bmatrix} \mathrm{MM} & \mathrm{MX} & 0 \\ \mathrm{XM} & \mathrm{XX_M} & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{F}(\mathbf{N}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathrm{XX_N} & \mathrm{XN} \\ 0 & \mathrm{NX} & \mathrm{NN} \end{bmatrix}$$

so that each block is labeled by the the edges that it contains. Explicitly, MM consists of the edges going from M to M, MX from M to X, XM from X to M, $\mathrm{XX_M}$ from X to X within M and similarly for the

blocks of $\mathbf{F}(\mathbf{N})$. Taking the blocks as their own variables, we plug $\mathbf{F}(\mathbf{M})$ and $\mathbf{F}(\mathbf{N})$ into the isomorphism of Theorem 3.1 to get the **composition symbol matrices**: $\mathsf{Symbol}(k) = F(M +_{LX} N))$ when $|X| = k$. The **composition symbols**, $\mathsf{Symbol}(k, i, j))$, are the entries of these matrices. For example,

$$\mathsf{Symbol}(4,1,3) = MX \cdot XN + MX \cdot XX_N \cdot XX_M \cdot XN$$

The terms of $\mathsf{Symbol}(4,1,3)$ represent the paths of length less than 4 which start in M and end in N. The second term of this symbol represents the paths which travel between components as drawn below



The single source single target shortest path algorithm has three steps:

1. The composition symbols are computed up to the size of the boundary. This needs to be done only once for each boundary size.

2. Precompilation. In this step, the all pairs shortest paths F(M) and F(N) are computed, pushed forward to $\mathbf{F}(\mathbf{M})$ and $\mathbf{F}(\mathbf{N})$, and broken into blocks as shown above. The computations in this step only need to be done once for each input matrix so their results may be reused in all further computations.

3. Composition. In this step, the source and target nodes s and t are located within the blocks of F(M) and F(N). The appropriate composition symbol is looked up and evaluated on the blocks of F(M) and F(N) using the operations of the min-plus matrix semiring. The first term of this expression is replaced by the row-vector corresponding to s an the last term must be replaced by the column vector corresponding to t.

An implementation in Python for this algorithm may be found at [1]. Figure 4 compares this algorithm to the networkx implementation of Dijkstra's algorithm. Figure 4 was computed using the pushout of two randomly weighted dense graphs with 500 nodes each. For each boundary size, the average computation time for 50 runs of both algorithms with randomly chosen source and target is plotted with the standard deviation of represented by shading. The precompilation time for this plot was 1.34 seconds. As the size of the boundary increases, the computation time for the compositional algorithm increases because the composition symbols grow very large. On the other hand, computation time for Dijkstra's algorithm decreases

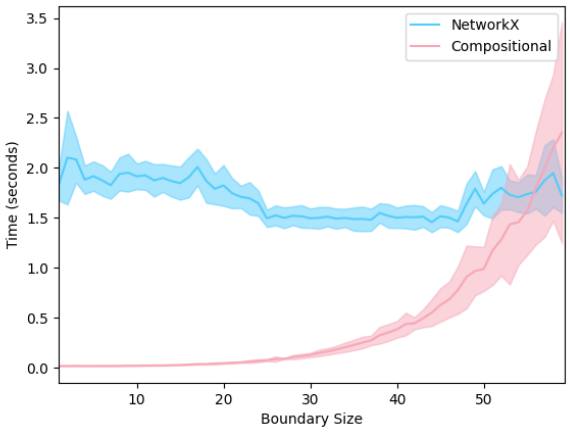


Figure 1: Comparison of Algorithms

slightly as the boundary size increases because identifying more nodes reduces the total size of the graph. The speed-up from using the compositional algorithm is most dramatic when the graph size is large and the boundary size is small. For the composition of two random graphs with 2000 nodes along a boundary of size 5, the times for the compositional algorithm were sampled 50 times for a mean value of 0.1602 seconds and with standard deviation 0.0169. Dijkstra's algorithm was sampled with the same parameters for a mean value of 39.7804 seconds and standard deviation 3.3561. Regardless, there is no free lunch, precompilation for the compositional algorithm took 93.0590 seconds.

5 Conclusion

In this paper we found a formula for composing shortest paths and described an algorithm which implements this formula. We hope that the results and algorithm may be generalized to other instances of the algebraic path problem. In particular we are interested in the case of semiring when the algebraic path problem asks for the language of a nondeterministic finite automaton. Orthogonally, we believe the results of this paper may be extended to more complicated decompositions of weighted graphs and plan on addressing this in future work.

References

[1] J. Master, PathComposer, 2022, Available at https://github.com/Jademaster/pathcomposer. (Referred to on page 3.)

[2] J. Master, The Open Algebraic Path Problem, In *Proceedings of CALCO 2021*, pp 20:1–20:20, Schloss Dagstuhl, 2021, Available at https://arxiv.org/abs/2005.06682. (Referred to on page 1, 2.)

Recent Advances in Homomorphism Indistinguishability

Tim Seppelt

May 27, 2022

In 1967, Lovász [11] proved that two graphs G and H are isomorphic if and only if they are *homomorphism indistinguishable* over the class of all graphs, i.e. for all graphs F the number of homomorphisms $F \to G$ is the same as the number of homomorphisms $F \to H$. This seminal paper sparked a fruitful area of research concerned with homomorphism indistinguishability over various restricted graph classes. In recent years, homomorphism indistinguishability over many natural graph classes has been characterised in terms of equivalence over certain logical fragments [7, 8, 14] and systems of equations [6, 13, 10]. For example, over the class of trees, homomorphism indistinguishability amounts to equivalence over the two-variable fragment of first-order logic with counting quantifier as well as feasibility of the fractional isomorphism system of equations. Other instances are listed in Table 1.

Despite the deepened understanding of various instances of homomorphism indistinguishability, the proofs of the results are often tailored to the specific graph classes and lack a uniform foundation. Towards a universal theory of homomorphism indistinguishability, powerful tools from algebra and category theory have emerged in recent years. This talk gives an overview of these developments and sketches the difficulties arising when attempting to reconcile the various approaches. Moreover, new results [15] attempting such a reconciliation are presented.

Category Theory: Comonads In various papers [1, 3, 5], it has been argued that comonads give rise to a powerful framework for studying homomorphism indistinguishability. In layman's terms, a comonad describes how to wrap a graph into an object which encodes certain information of interest. More precisely, it can be shown [2] that for many natural graph classes there exists a comonad $\mathfrak{C}$ such that a graph belongs to this class if and only if its admits a $\mathfrak{C}$-coalgebra. Furthermore, under mild assumptions, homomorphism indistinguishability of G and H over this class amounts to isomorphism of the cofree coalgebras associated to G and H by $\mathfrak{C}$. This approach has led to uniform proofs for logical characterisations of homomorphism indistinguishability over graphs of bounded treewidth and -depth [5] and to new results concerning graphs of bounded pathwidth [14].

Representation Theory: Homomorphism Tensors Homomorphism indistinguishability results may also be obtained by the means of algebra and representation theory [12, 13, 10]. This is done by considering ℓ-labelled graphs $\boldsymbol{F} = (F, u_1 \dots u_\ell)$ which comprise a graph F and selection of its vertices $u_1, \dots, u_\ell \in V(F)$. The set of all suitably ℓ-labelled graphs from a graph class $\mathcal{F}$ has an algebraic structure induced by parallel composition. One may now attempt to represent this algebraic object in terms of matrices. It turns out that, in many cases, these representations are semisimple and that their characters capture precisely the homomorphism counts of interest. In this way, representation-theoretic arguments yield systems of equations whose feasibility is equivalent to homomorphism indistinguishability over graph classes such as those of bounded pathwidth, treewidth, and -depth.

Graph class	Logical characterisation	System of equations
Cycles	*unknown*	cospectrality of adjacency matrices
Trees	two-variable fragment of first order logic with counting [7]	fractional isomorphism [18]
Tree width $\leq k$	$(k+1)$-variable fragment of first-order logic with counting [7]	non-negative solution to Sherali–Adams relaxation of fractional isomorphism [4, 9, 6]
Path width $\leq k$	[14]	rational solution to Sherali–Adams relaxation of fractional isomorphism [6, 10]
Tree depth $\leq q$	quantifier-depth-q fragment of first-order logic with counting [8]	[10]
Graphs with k-pebble forest cover of depth q	$(k+1)$-variable quantifier-depth-q fragment of first-order logic with counting [1, 3, 5, 17]	[15]
Planar graphs	*unknown*	quantum isomorphism [13]
Trees of bounded degree	*unknown*	[10]

Table 1: Some instances of homomorphism indistinguishability

C*-algebras: Planar Graphs A third approach to the matter has been developed by Mančinska and Roberson [13] who proved that homomorphism indistinguishability over planar graphs amounts to the feasibility of a system of equations with variables ranging over a C*-algebra. Their argument prominently rests on labelled graphs and their representations as homomorphism tensors. Instead of considering only a constant number of labels, they work over all possible finite numbers of labels simultaneously, which leads to the study tensor category with duals. A crucial ingredient here is the Tannaka–Krein duality, which relates quantum groups to their categories of representations. This characterisation of homomorphism indistinguishability has so far eluded comonadic means or more elementary representation-theoretic approaches.

Reconciling the Approaches In order to derive characterisations in terms of logic or systems of equations in a uniform manner, a rich set of tools based on representation and category theory has been developed. So far, none of these approaches is able to explain the results on planar graphs. This talk surveys the encountered difficulties and presents steps towards a reconciliation of the various approaches. To that end, a novel characterisation [15] of equivalence over the k-variable quantifier-depth-q fragment of first-order logic with counting quantifiers is presented. This result builds upon insights from the category-theoretic study of homomorphism indistinguishability and introduces new algebraic tools extending the known representation-theoretic machinery. If time permits, general properties of equivalence relations on the class of graphs stemming from homomorphism indistinguishability as studied in [16] will be discussed.

References

[1] Samson Abramsky, Anuj Dawar, and Pengming Wang. The Pebbling Comonad in Finite Model Theory. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '17. IEEE Press, 2017. event-place: Reykjavík, Iceland. doi:10.1109/LICS.2017.8005129.

[2] Samson Abramsky, Tomáš Jakl, and Thomas Paine. Discrete density comonads and graph parameters, May 2022. URL: http://arxiv.org/abs/2205.06589.

[3] Samson Abramsky and Nihil Shah. Relating Structure and Power: Comonadic Semantics for

Computational Resources. In Corina Cîrstea, editor, *Coalgebraic Methods in Computer Science*, pages 1–5, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-030-00389-0_1.

[4] Albert Atserias and Elitza Maneva. Sherali–Adams Relaxations and Indistinguishability in Counting Logics. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 367–379, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2090236.2090265.

[5] Anuj Dawar, Tomás Jakl, and Luca Reggio. Lovász-type theorems and game comonads. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470609.

[6] Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.40.

[7] Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, August 2010. doi:10.1002/jgt.20461.

[8] Martin Grohe. Counting bounded tree depth homomorphisms. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 507–520. ACM, 2020. doi:10.1145/3373718.3394739.

[9] Martin Grohe and Martin Otto. Pebble Games and Linear Equations. *J. Symb. Log.*, 80(3):797–844, 2015. doi:10.1017/jsl.2015.28.

[10] Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism Tensors and Linear Equations. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. to appear. doi:10.4230/LIPIcs.ICALP.2022.26.

[11] László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3):321–328, September 1967. doi:10.1007/BF02280291.

[12] László Lovász. *Large Networks and Graph Limits*. American Mathematical Society, 2012. doi:10.1090/coll/060.

[13] Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 661–672. IEEE, 2020. doi:10.1109/FOCS46700.2020.00067.

[14] Yoàv Montacute and Nihil Shah. The pebble-relation comonad in finite model theory. *ArXiv*, 2110.08196, 2021. doi:10.48550/arXiv.2110.08196.

[15] Gaurav Rattan and Tim Seppelt. Weisfeiler–Leman and Graph Spectra, 2022. doi:10.48550/ARXIV.2103.02972.

[16] David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree. Private communication, January 2022.

[17] Gian Luca Spitzer. *Characterising Fragments of First-Order Logic by Counting Homomorphisms*. B.Sc. Thesis, RWTH Aachen University, 2022.

[18] Gottfried Tinhofer. Graph isomorphism and theorems of Birkhoff type. *Computing*, 36(4):285–300, December 1986. doi:10.1007/BF02240204.

Coherence, conjectures, and congruential functions

Peter M. Hines – University of York

1. Summary This talk is based on previously unobserved connections between an unresolved conjecture of Lothar Collatz [10], the first two parts of Girard's Geometry of Interaction program [5, 6], categorical coherence, and Richard Thompson's group $\mathcal{F}$. It is based on, but significantly extends, the draft paper https://arxiv.org/abs/2202.04443

We decompose the iterative step from the original Collatz conjecture (Section 2) into a series of more primitive steps. This allows us to express it as a simple game in two distinct ways, that differ by the ordering of these elementary steps. These distinct orderings correspond to a choice between left- and right- associativity for composition.

These distinct games are related by a natural transformation of monoid homomorphisms whose unique component is the successor function, *succ*. We treat this as a member of the symmetric inverse monoid $\mathcal{I}(\mathbb{N})$, in order to use its generalised inverse to define define conjugation and commutators based on the successor.

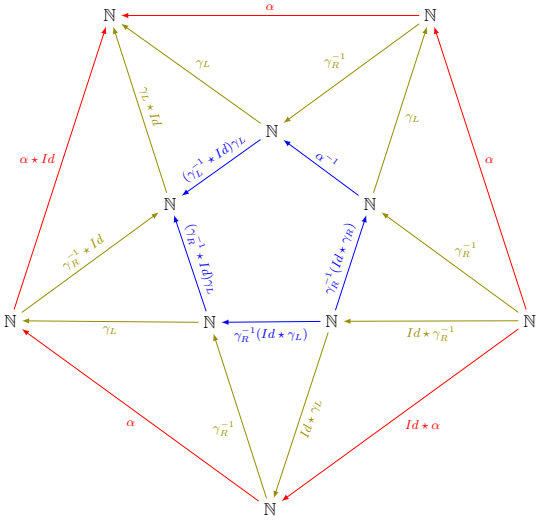
The commutator of the successor function and the arithmetic operation from Collatz's conjecture then gives a function familiar from Girard's first two Geometry of Interaction papers [5, 6], where conjunction is modelled as an injective homomorphism on the symmetric inverse monoid on the natural numbers, $\_\star\_ : \mathcal{I}(\mathbb{N}) \times \mathcal{I}(\mathbb{N}) \to \mathcal{I}(\mathbb{N})$. This is well-known (e.g. [8, 7, 1]) to be a (semi-monoidal) categorical tensor, and the above commutator is its unique canonical associativity isomorphism, satisfying MacLane's pentagon condition.

The long-established folklore that there is a close connection between associativity / MacLane's pentagon, and Thompson's group $\mathcal{F}$, was put on a firm footing by P. Dehornoy [4]. We use this observation to give a group of Conway's congruential functions (Section 2.1) isomorphic to $\mathcal{F}$, based on Collatz's conjecture and Girard's conjunction. In stark contrast to Collatz's problem, questions of finite vs. infinite orbits in this setting are decidedly trivial.

Categorically, the associator for Girard's conjunction is the component of a natural transformation describing re-bracketing; we similarly interpret the two distinct ways of expressing Collatz's conjecture as components of natural transformations between insertions and deletions of matching pairs of brackets. This gives, as one may expect, re-bracketings as combinations of deleting and re-inserting pairs of brackets.

We give this as a commuting diagram of congruential functions that contains, but extends, MacLane's pentagon (Figure 1). The red sub-diagram is (the single-object version of) MacLane's classic Pentagon condition, $\alpha^2 = (\alpha \star Id)\alpha(Id \star \alpha)$. This interprets as mappings between vertices of the fourth associahedron $\mathcal{K}_4$. The inner pentagon describes instead mappings between edges of $\mathcal{K}_4$, and the pentagram figure relating the two gives mappings between vertices and edges.

Figure 1: A commuting diagram of congruential functions



Where :

γ_R is the operator from Collatz's conjecture (Figure 2(b))

$\gamma_L = succ^{-1}.\gamma_R.succ$ is the alternative Collatz operator (Figure 2(a))

$(\_\star\_) : \mathcal{I}(\mathbb{N}) \times \mathcal{I}(\mathbb{N}) \to \mathcal{I}(\mathbb{N})$ is the conjunction from Girard's GoI system (Section 3)

$\alpha = \gamma_L\gamma_R^{-1} = \left[succ, \gamma_R^{-1}\right]$ is the canonical associativity isomorphism for Girard's conjunction $(\_\star\_)$.

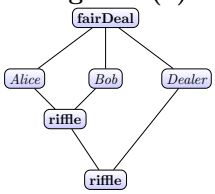
Finally, we put this observation in a general setting. We show how to construct commuting diagrams of congruential functions from arbitrary associahedra, and demonstrate that paths labelled by the function from Collatz's conjecture occur on every associahedron.

2. The original Collatz problem

Although Collatz's $3x+1$ problem is understandably famous, a lesser-known but equally intriguing (& similarly unsolved) problem of his was publicised in [10], where J. Lagarias describes the *'Original Collatz Problem' (OCP)* found in unpublished notebooks of Collatz dated 1^{st} July 1932. This conjectures that the orbit of 8 under the bijection $\gamma_R : \mathbb{N} \to \mathbb{N}$ (given in Figure 2(b)) is unbounded. Collatz described the origins of his $3x+1$ problem in [2], and similarly described the origins of his original conjecture in an (unpublished) letter to J. Lagarias (1985); unfortunately this letter has since been lost [9].

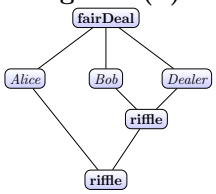
In the absence of Collatz's motivation, we may invent our own story : the above bijection decomposes into more primitive steps, consisting of a (three-player) fair deal of a countably infinite pack of cards, and two (perfectly interleaving) riffle shuffles of two hands of cards. These implement Collatz's bijection as shown in Figure 2(b); his conjecture becomes a question about the paths of cards under repeated rounds of this game. This graphical description also highlights a second alternative ordering of the same primitives; this is shown in Figure 2(a), and implements a distinct bijection γ_L.

Figure 2(a): The left-associated Collatz bijection



$$\gamma_L(n) = \begin{cases} \frac{4n}{3} & n \ (mod\ 3) = 0 \\ \frac{4n+2}{3} & n \ (mod\ 3) = 1 \\ \frac{2n-1}{3} & n \ (mod\ 3) = 2 \end{cases}$$

Figure 2(b): The right-associated Collatz bijection



$$\gamma_R(n) = \begin{cases} \frac{2n}{3} & n \ (mod\ 3) = 0 \\ \frac{4n-1}{3} & n \ (mod\ 3) = 1 \\ \frac{4n+1}{3} & n \ (mod\ 3) = 2 \end{cases}$$

These are related by the identity $1 + \gamma_L^K(\_) = \gamma_R^K(\_ + 1)$ for all $K \in \mathbb{N}$, giving a natural transformation of monoid homomorphisms whose component is the successor function. We may thus express the OCP in terms of either γ_R or γ_L.

2.1 Congruential functions and universal computability

It is entirely possible that the OCP is undecidable, although this could never be proved. Motivated by Collatz's more famous problem, in [3] John Conway considered *congruential functions* – maps on $\mathbb{N}$ defined piecewise-linearly on exact covering systems. He demonstrated – via a reduction to Post production systems – that Turing-complete computation could be implemented via iterative problems such as those posed by Collatz (see also [12]). Deciding whether the orbit of a given natural under a congruential function is finite is exactly as hard as proving that a computer program terminates on some given input.

Despite – or perhaps because of – the link with universal computability, it is still worthwhile to study such problems for their algebraic and categorical interest, even without tackling the problem of whether such conjectures are true. We give a close connection with Girard's Geometry of Interaction series of papers; this shows the bijections $\gamma_R, \gamma_L : \mathbb{N} \to \mathbb{N}$ expressing the OCP to be canonical coherence isomorphisms, closely linked to coherence for associativity, and Richard Thompson's group $\mathcal{F}$.

3. The conjunction of Girard's Geometry of Interaction

In [5, 6], propositions of Multiplicative-Exponential Linear Logic are modelled (up to M. Barr's l_2 functor) as partial injective functions on $\mathbb{N}$. The multiplicative conjunction is then modelled as an injective inverse monoid homomorphism $(a \star b)(n) = \begin{cases} 2a(n/2) & n \text{ even}, \\ 1 + 2b((n-1)/2) & n \text{ odd}. \end{cases}$ This is well-known to be a semi-monoidal tensor (e.g. [8, 1]), associative up to a canonical associativity isomorphism $\alpha : \mathbb{N} \to \mathbb{N}$ satisfying MacLane's pentagon condition (the red sub-diagram of Figure 1). Direct calculation gives this associator to be the congruential function $\alpha = \gamma_L \gamma_R^{-1}$.

Using the natural transformation relating the left and the right Collatz bijections, we may therefore write this in terms of the function written down by Collatz, as $\alpha = \gamma_R(\gamma_R^{-1}(n) + 1) - 1$. Working with partial injective functions, as used by Girard, we may also use generalised inverses to write this as a (inverse semigroup theoretic) commutator $\alpha = \left[succ, \gamma_R^{-1}\right]$.

3.1 A congruential realisation of Richard Thompson's $\mathcal{F}$

Girard's conjunction also has a natural interpretation in terms of shuffles & deals of cards, at least for bijections.

> A countably infinite pack is dealt out to Alice and Bob, who apply bijections a and b to their respective hands. These are returned to the dealer, who shuffles them back together.

This process implements the conjunction $(a \star b)$.

Given this intuition, it is perhaps unsurprising that the 'conjunction' of two congruential bijections is itself a congruential bijection. We use this, together with the close connection between coherence for associativity / MacLane's pentagon condition with Thompson groups described in [4], to give a group of congruential bijections isomorphic to $\mathcal{F}$. We define $X_0 = \alpha = \left[succ, \gamma_R^{-1}\right]$, and $X_{j+1} = Id \star X_j$, for all $j \in \mathbb{N}$. The set $\{X_j\}_{j \in \mathbb{N}}$ then generates a subgroup of congruential functions isomorphic to Thompson's $\mathcal{F}$. We further show that the problem of characterising orbits is trivial for this set; the orbit of every natural number is either of length zero (i.e. a fixed-point), or unbounded.

4. A categorical explanation We account for all the above results categorically. Girard's conjunction may be seen as the first of an infinite family of *'k-ary elementary conjunctions'* – injective inverse monoid homomorphisms $\mu_k : \mathcal{I}(\mathbb{N})^{\times k} \hookrightarrow \mathcal{I}(\mathbb{N})$ defined by $\mu_k(f_0, f_1, \ldots, f_{k-1})(n) = k.f_r\left(\frac{n-r}{k}\right) + r$, where $r = n \ (mod\ k)$, with the simplest non-trivial example (i.e. $\mu_2 = (\_ \star \_) : \mathcal{I}(\mathbb{N}) \times \mathcal{I}(\mathbb{N}) \hookrightarrow \mathcal{I}(\mathbb{N})$) being Girard's semi-monoidal tensor. These generate a (non-symmetric) operad $\mathcal{GC}$ of 'generalised conjunctions', as a sub-operad of the endomorphism operad of $\mathcal{I}(\mathbb{N})$ in the category of inverse monoids & their homomorphisms. The non-symmetric operad $\mathcal{GC}$ is freely generated by one operation of each arity, and thus isomorphic to the formal operad $\mathbb{RPT}$ of rooted planar trees.

In order to give an interpretation of Collatz's bijections as a form of coherence, we consider natural isomorphisms between generalised conjunctions. Those between the generalised conjunctions of arity three – i.e. the inverse monoid homomorphisms $\mu_2(\mu_2(\_,\_),\_)$, $\mu_2(\_,\mu_2(\_,\_))$, $\mu_2(\_,\mu_2(\_,\_))$ $: \mathcal{I}(\mathbb{N})^{\times 3} \hookrightarrow \mathcal{I}(\mathbb{N})$ have, as unique components, the left- and right- Collatz bijections, and the associator. These homomorphisms, and natural isomorphisms between them, form a posetal functor category.

In the general case, we describe natural isomorphisms between all generalised conjunctions of the same arity. These are congruential bijections, and are the arrows of posetal subcategories of the functor categories of inverse monoid homomorphisms, $\mathbf{Inv}(\mathcal{I}(\mathbb{N})^{\times k}, \mathcal{I}(\mathbb{N}))$, for all $k > 0$. We take the coproduct of these to give a posetal category closed under generalised conjunctions (including Girard's conjunction $\mu_2 = (\_ \star \_)$ as a semi-monoidal tensor). This, of course, contains (a unitless version of) MacLane's posetal category $(\mathcal{W}, \otimes)$ as a special case.

As $\mathcal{GC} \cong \mathbb{RPT}$, it is natural to label facets of the n-th associahedron $\mathcal{K}_n$ with generalised conjunctions of arity n (this is for *all* facets, not simply the 1-skeleton), and paths between them labelled by the above unique natural isomorphisms – congruential functions in the corresponding posetal functor category.

A faithful functor from this posetal category to $\mathcal{I}(\mathbb{N})$ itself gives a class of diagrams, based on associahedra and labelled by congruential functions, guaranteed to commute. We then demonstrate that well-known [11] embeddings $\mathcal{K}_n \hookrightarrow \mathcal{K}_{n+a}$ preserve path-labellings. Hence Collatz's bijections appear as natural transformations between generalised conjunctions of arbitrary arity, and so as labels of paths in commuting diagrams derived from arbitrary dimensional associahedra.

Future directions Our starting point was a close connection between coherence for associativity (for Girard's conjunction) and the OCC. If time permits, we will briefly describe some supporting evidence for the claim that Collatz's $3x+1$ problem can similarly be seen as a question of coherence for associativity & symmetry (again, for Girard's conjunction).

References

[1] S. Abramsky, E. Haghverdi, and P. Scott. Geometry of interaction and linear combinatory algebras. *Mathematical Structures in Computer Science*, 12 (5), 2002.

[2] Lothar Collatz. On the motivation and origin of the $3n + 1$ problem. *J. Qufu Normal University, Natural Science Edition*, 12(3):9–11, 1986.

[3] John Conway. Unpredictable iterations. *Proc. 1972 Number Theory*, pages 49–52, 1972.

[4] P. Dehornoy. The structure group for the associativity identity. *J. Pure Appl. Algebra*, 111, 1-3:59–82, 1996.

[5] J.-Y. Girard. Geometry of interaction 1. In *Proceedings Logic Colloquium '88*, pages 221–260. North-Holland, 1988.

[6] J.-Y. Girard. Geometry of interaction 2: deadlock-free algorithms. In *Conference on Computer Logic*, volume 417 of *Lecture Notes in Computer Science*, pages 76–93. Springer, 1988.

[7] Esfan Haghverdi. *A categorical approach to linear logic, geometry of proofs and full completeness.* PhD thesis, University of Ottawa, 2000.

[8] P. Hines. *The algebra of self-similarity and its applications.* PhD thesis, University of Wales, Bangor, 1997.

[9] J. Lagarias. private communication, 2020. Shared with permission.

[10] Jeffrey Lagarias. The 3x+ 1 problem and its generalisations. *American Mathematical Monthly*, 92, 01 1985.

[11] Jean-Louis Loday. The multiple facets of the associahedron. 2005.

[12] Sergei Ju. Maslov. On e. l. post's "tag problem". *Trudy Matematicheskogo Instituta imeni V.A. Steklova*, 72:5–56, 1964. English translation available as [13].

[13] Sergei Ju. Maslov. On e. l. post's "tag problem". In B. M. Budak, editor, *Eleven Problems on Logic, Algebra, Analysis, and Topology*, A.M.S. Translations. American Mathematical Society, 1971.

Query Algorithms Based on Homomorphism Counts

Wei-Lin Wu*
Department of Computer Science and Engineering,
University of California Santa Cruz
wwu53@ucsc.edu

Abstract

In a recent paper, Chen et al. investigated the expressive power of query algorithms in identifying classes of graphs using a fixed number of left homomorphism counts and compared it with that of such query algorithms using a fixed number of right homomorphism counts. We delve deeper into this comparison regarding first the constraint satisfaction problems and second the graph isomorphism problem together with a proposal of a hybrid (left and right) query algorithm that combines the best features of each.

1 Introduction

Two classical results separately by Lovász [Lov67] and by Chaudhuri and Vardi [CV93] characterize graph isomorphism via homomorphism counts. Writing $\hom(G, H)$ for the number of homomorphisms from G to H, for graphs G, H of size $\leq n$ ($n \geq 1$), the former says that they are isomorphic if and only if $\hom(F, G) = \hom(F, H)$ for all graphs F of size $\leq n$, while the latter states that they are isomorphic if and only if $\hom(G, F) = \hom(H, F)$ for all graphs F of size $\leq n$.

In [CFLX21], Chen et al. studied the question whether a class of graphs admits a left (non-)adaptive algorithm. Let $\mathcal{C}$ be a class of graphs. A *left k-non-adaptive algorithm for $\mathcal{C}$* ($k \geq 1$) consists of a class $\mathcal{F} = \{F_1, \ldots, F_k\}$ of graphs and a decidable set $X \subseteq \mathbb{N}^k$ such that, given any graph G as input, k queries $\hom(F_1, G), \ldots, \hom(F_k, G)$ are made to decide whether $G \in \mathcal{C}$ as output: $G \in \mathcal{C}$ if and only if $(\hom(F_1, G), \ldots, \hom(F_k, G)) \in X$.[1] A *left k-adaptive algorithm for $\mathcal{C}$* differs from a non-adaptive one in that the queries except the first one are adaptive: for input G, each F_i is computed as a function $F_i(n_1, \ldots, n_{i-1})$ of $n_1, \ldots, n_{i-1}$ for $i \in \{2, \ldots, k\}$ (but F_1 is fixed), where $n_1 := \hom(F_1, G), \ldots, n_{k-1} := \hom(F_{k-1}, G)$. We say *$\mathcal{C}$ admits a left k-(non-)adaptive algorithm* if there is such an algorithm for $\mathcal{C}$. Symmetrically, the same can be defined for *right k-(non-)adaptive algorithm for $\mathcal{C}$*, etc., by making the k queries $\hom(G, F_1), \ldots, \hom(G, F_k)$ instead.[2]

Chen et al. showed (i) if a class of graphs is definable by some Boolean combination of universal first-order sentences, then it admits a left non-adaptive algorithm, and (ii) three queries $\hom(F_0, G), \hom(F_1, G), \hom(F_2, G)$ suffice to determine the isomorphism type of any graph G where $F_1 = F_1(n), F_2 = F_2(n)$ are computed as functions of $n := \hom(F_0, G)$,[3] using the aforementioned result by Lovász. They also presented examples of classes admitting a left (non-)adaptive algorithm but not any right (non-)adaptive algorithms.

In this paper, we aim to identify some situations where queries of the form $\hom(G, F)$ with input G are useful: (i) the constraint satisfaction problem with a fixed template H, viewed as a class, admits a right 1-non-adaptive algorithm but not any left non-adaptive algorithms except for trivial cases of H (see Section 2), and (ii) for $n \geq 1$, a graph $F_0 = F_0(n)$ as a function of n can be constructed such that for all graphs G, H of size n, they are isomorphic if and only if $\hom(G, F_0) = \hom(H, F_0)$, using the aforementioned result by Chaudhuri and Vardi (see Section 3).

The conventions, notations and definitions in this paper are given now. Graphs are finite, undirected, loop-free, and do not have multiedges. By a *class* of graphs we mean a collection of graphs in which no two graphs are isomorphic. The set of nodes in a graph G is denoted $V(G)$. The *size of G* is $|V(G)|$. Let $\mathbb{N} := \{0, 1, 2, \ldots\}$ and $\mathbb{N}^+ := \{1, 2, 3, \ldots\}$. The disjoint union operation is $\oplus$. We write $\bigoplus_{i=1}^{n} F_i$ for the disjoint union of $F_1, \ldots, F_n$ ($n \in \mathbb{N}^+$) and write $\bigoplus_n F$ when $F_1 = \cdots = F_n = F$. Let G and H be two graphs. A *homomorphism from G to H* is a mapping $h : V(G) \to V(H)$ that preserves adjacency. We write $\hom(G, H)$, $\mathrm{inj}(G, H)$ and $\mathrm{sur}(G, H)$ for the number of homomorphisms, injective homomorphisms and surjective[4] homomorphisms from G to H, respectively. If $h : V(G) \to V(H)$ is bijective and is a homomorphism from G to H and if its inverse h^{-1} is a homomorphism from H to G, then h is called an *isomorphism from G to H*; in case $H = G$, we call it an *automorphism of G*. We write $\mathrm{aut}(G)$ for the number of automorphisms of G.

*This paper is my joint work with my advisor at UC Santa Cruz, Professor Phokion G. Kolaitis.

[1] Such algorithm has the qualifier *left* because in the k queries $\hom(F_1, G), \ldots, \hom(F_k, G)$ the left argument of $\hom(\cdot, \cdot)$ varies (over a class) while the right argument is fixed to the input graph G.

[2] Occasionally, we say *$\mathcal{C}$ admits a left (or right) (non-)adaptive algorithm* when the number k of queries is irrelevant.

[3] They argued that three queries are necessary in some cases and hence are indeed optimal.

[4] Both node-wise and edge-wise.

2 Constraint Satisfaction Problem and Query Algorithms

The *constraint satisfaction problem* $\mathrm{CSP}(H)$ with a fixed graph H (called the *template*) takes an input graph G and answers as output whether $\hom(G,H)>0$. We will identify $\mathrm{CSP}(H)$ with the class $\{G \mid \hom(G,H)>0\}$. Obviously, for all H, the class $\mathrm{CSP}(H)$ admits a right 1-non-adaptive algorithm: Take $\mathcal{F}=\{H\}$ and $X=\mathbb{N}^+$. However, it turns out that $\mathrm{CSP}(H)$ does not admit any left non-adaptive algorithm unless H contains no edge (i.e., H has chromatic number 1), for which there is a trivial left 1-non-adaptive algorithm with $\mathcal{F}=\{K_2\}$ (K_2 denotes the graph of two nodes connected by an edge) and $X=\{0\}$. Note that to assert a class $\mathcal{C}$ of graphs does not admit a left k-non-adaptive algorithm for any $k\geq 1$ it suffices to show that for every nonempty finite class $\mathcal{F}$ of graphs, there are graphs $G_0\in\mathcal{C}$ and $G_1\notin\mathcal{C}$ such that $\hom(F,G_0)=\hom(F,G_1)$ for all $F\in\mathcal{F}$.

Theorem 1. *Let H be a graph. Then $\mathrm{CSP}(H)$ admits a left non-adaptive algorithm if and only if H contains no edge.*

We prove this theorem in the sequel. Let us begin with the two immediate properties. (i) Additivity of Homomorphism Counts: $\hom(G,\bigoplus_{i=1}^n F_i)=\sum_{i=1}^n \hom(G,F_i)$ for arbitrary graphs $F_1,F_2,\ldots,F_n$ and a *connected* G. (ii) Multiplicativity of Homomorphism Counts: $\hom(\bigoplus_{i=1}^n F_i,G)=\prod_{i=1}^n \hom(F_i,G)$ for arbitrary graphs $F_1,F_2,\ldots,F_n$ and G.

Next, we write C_n for the cycle of n nodes ($n\geq 3$); a graph is *cyclic* if it contains C_n as a subgraph for some $n\geq 3$, otherwise is *acyclic*. We argue that for every nonempty finite class $\mathcal{F}$ of *connected* graphs, there are a 2-colorable graph G_0 and a non-2-colorable graph G_1 such that $\hom(F,G_0)=\hom(F,G_1)$ for all $F\in\mathcal{F}$. Given such $\mathcal{F}$, let s be the maximum size of graphs in $\mathcal{F}$, and denote by $\mathcal{F}'$ the class of all connected graphs of size $\leq s$ (note that $\mathcal{F}\subseteq\mathcal{F}'$). Let $n\geq\max\{3,s+1\}$ be an odd integer, and choose $G_0:=C_{2n}$ and $G_1:=C_n\oplus C_n$. Clearly, G_0 is 2-colorable while G_1 is not. Now, we divide the class $\mathcal{F}'$ into three disjoint subclasses $\mathcal{F}'_1$, $\mathcal{F}'_2$ and $\mathcal{F}'_3$ such that $\mathcal{F}'_1$ consists of cyclic graphs, $\mathcal{F}'_2$ consists of trees of degree >2 and $\mathcal{F}'_3$ consists of paths.[5] Observe that (1) $\mathrm{inj}(F',G_0)=\mathrm{inj}(F',G_1)=0$ for $F'\in\mathcal{F}'_1$ because all cycles appearing in the graphs in $\mathcal{F}'_1$ have length $\leq s<n$, (2) $\mathrm{inj}(F',G_0)=\mathrm{inj}(F',G_1)=0$ for $F'\in\mathcal{F}'_2$ because G_0 and G_1 have degree 2, and (3) $\mathrm{inj}(F',G_0)=\mathrm{inj}(F',G_1)$ for $F'\in\mathcal{F}'_3$ because all paths in $\mathcal{F}'_3$ have length $<s<n$. Thus, $\mathrm{inj}(F',G_0)=\mathrm{inj}(F',G_1)$ for all $F'\in\mathcal{F}'$. It follows that for all $F\in\mathcal{F}$, we have $\hom(F,G_0)=\hom(F,G_1)$ since the image of F under a homomorphism must be in $\mathcal{F}'$ and $\hom(F,G_0)=\sum_{F'\in\mathcal{F}'}\mathrm{sur}(F,F')\cdot\mathrm{inj}(F',G_0)/\mathrm{aut}(F')$, and likewise for $\hom(F,G_1)$.

In fact, for every *arbitrary* nonempty finite class $\mathcal{F}$ in which graphs are not necessarily connected, we can take the class $\mathcal{F}^\circ$ of all connected components of the graphs in $\mathcal{F}$, and then apply the above argument to $\mathcal{F}^\circ$ to get a 2-colorable G_0 and a non-2-colorable G_1 such that $\hom(F^\circ,G_0)=\hom(F^\circ,G_1)$ for all $F^\circ\in\mathcal{F}^\circ$. By multiplicativity of homomorphism counts, it follows that $\hom(F,G_0)=\hom(F,G_1)$ for all $F\in\mathcal{F}$. As a consequence, if H has chromatic number 2, then $\mathrm{CSP}(H)$ is exactly the class of all 2-colorable graphs and hence does not admit any left non-adaptive algorithm.

Finally, it remains to show that if H has chromatic number ≥ 3, then $\mathrm{CSP}(H)$ does not admit any left non-adaptive algorithm. Let H be such a graph and $\mathcal{F}$ be any nonempty finite class, our goal is to prove that there are two graphs $G_0\in\mathrm{CSP}(H)$ and $G_1\notin\mathrm{CSP}(H)$ such that $\hom(F,G_0)=\hom(F,G_1)$ for all $F\in\mathcal{F}$. Let $n\geq 2$ be an integer greater than the maximum treewidth of the graphs in $\mathcal{F}$. Since H is not 2-colorable, the class $\mathrm{CSP}(H)$ is not definable in the counting logic $\mathrm{C}^\omega_{\infty\omega}$ by the Definable H-Coloring Dichotomy Theorem (see Theorem 11 in [AKW21]), which implies the existence of two graphs $G_0\in\mathrm{CSP}(H)$ and $G_1\notin\mathrm{CSP}(H)$ that are indistinguishable by the counting logic C^n and hence, by Dvořák's Theorem (see Theorem 7 in [Dvo10]), $\hom(F,G_0)=\hom(F,G_1)$ for all graphs F of treewidth $\leq n-1$. The last condition yields $\hom(F,G_0)=\hom(F,G_1)$ for all $F\in\mathcal{F}$.

3 Isomorphism and Query Algorithms

In [CFLX21], Chen et al. showed that for every $n\in\mathbb{N}^+$, two graphs $F_1=F_1(n),F_2=F_2(n)$ as functions of n can be constructed so that for all graphs G and H of size n, they are isomorphic if and only if $\hom(F_1,G)=\hom(F_1,H)$ and $\hom(F_2,G)=\hom(F_2,H)$. As a result, the isomorphism type of any graph G (viewed as a singleton class) admits a left 3-adaptive-algorithm in which, for input H, the first query $\hom(I_1,H)$ decides the size of H, followed by the two queries $\hom(F_1(n),H),\hom(F_2(n),H)$ where $n:=\hom(I_1,H)=|V(H)|$ and I_1 denotes the single-node graph. This is optimal in terms of the number of queries made. They also showed that no right adaptive-algorithm can do the job.

In contrast, however, we discovered:

Theorem 2. *For every $n\in\mathbb{N}^+$, a single graph $F_0=F_0(n)$ as a function of n can be constructed so that for all graphs G and H of size n, they are isomorphic if and only if* $\hom(G,F_0)=\hom(H,F_0)$.

This implies, for any graph G, a *hybrid* 2*-adaptive-algorithm* for the singleton class $\{G\}$ (i.e., the isomorphism type of G) that consists of the construction of F_0 and the set $X:=\{(n,\hom(G,F_0(n))\}$ where $n:=\hom(I_1,G)$. That is, for input

[5] Since $\mathcal{F}'$ consists of connected graphs, the graphs in $\mathcal{F}'\setminus\mathcal{F}'_1$ are trees and the graphs in $\mathcal{F}'\setminus(\mathcal{F}'_1\cup\mathcal{F}'_2)$ are paths. Here paths include the single-node graph I_1.

graph H, make the two queries $m := \hom(I_1, H)$ and $\ell := \hom(H, F_0(m))$ in succession and decide whether $(m, \ell) \in X$. Note that this algorithm is already optimal in terms of the number of queries made.

We prove the theorem as follows. First, observe that given $D \in \mathbb{N}^+$, every sequence $(a_0, \ldots, a_k)$ of fixed length $k+1$ in which $a_i \in \{0, \ldots, D-1\}$ can be encoded by a unique integer $a_0 \times D^0 + \cdots + a_k \times D^k$. Now, let $n \in \mathbb{N}^+$ be given. Then we let $A_1, \ldots, A_s$ enumerate (the isomorphism types of) all graphs of size $\leq n$. Our goal is to construct F_0 such that, for a suitable $D \in \mathbb{N}^+$, certain digits in the D-ary representation of $\hom(G, F_0)$ are $\hom(G, A_1), \ldots, \hom(G, A_s)$ for all graphs G of size n. For this purpose, put $F_0 := \bigoplus_{j=1}^{s} (\bigoplus_{D^{e_j}} A_j)$, where $e_1, \ldots, e_s, D \in \mathbb{N}^+$ will be determined later. We write $\mathcal{E}^+_{\leq n}$ for the set of all integers that are sums of at most n (not necessarily distinct) integers from $\{e_1, \ldots, e_s\}$. Then by additivity and multiplicativity of homomorphism counts, if $G = G_1 \oplus \cdots \oplus G_r$ is an arbitrary graph of size n with r connected components $G_1, \ldots, G_r$ $(1 \leq r \leq n)$, then

$$
\begin{aligned}
\hom(G, F_0) &= \prod_{i=1}^{r} \hom(G_i, F_0) = \prod_{i=1}^{r} \sum_{j=1}^{s} \hom(G_i, A_j) \times D^{e_j} \\
(*) \qquad &= \sum_{e \in \mathcal{E}^+_{\leq n}} \left(\sum_{\substack{j_1, \ldots, j_r \in \{1, \ldots, s\} \\ e_{j_1} + \cdots + e_{j_r} = e}} (\hom(G_1, A_{j_1}) \times \cdots \times \hom(G_r, A_{j_r})) \right) \times D^e.
\end{aligned}
$$

The two properties will be desirable: (1) the outer summation on the last line of $(*)$ is the D-ary representation of $\hom(G, F_0)$, and (2) for all $j \in \{1, \ldots, s\}$, the digit for D^{re_j} in this D-ary representation of $\hom(G, F_0)$ is $\hom(G_1, A_j) \times \cdots \times \hom(G_r, A_j) = \hom(G, A_j)$.

Next, we determine the values of $e_1, \ldots, e_s$ and D to achieve the two desirable properties, based on the (arbitrary) graph G in the previous paragraph. Let $e_1 := 1$ and $e_{j+1} := n^j + n^{j-1} + \cdots + 1$ for all $j \in \{1, \ldots, s-1\}$. It follows that $e_{j+1} > ne_j$ for all $j \in \{1, \ldots, s-1\}$ and that $e_1, \ldots, e_s$ is a strictly increasing sequence. Hence, it is easy to show that every integer $e \in \mathcal{E}^+_{\leq n}$ can be expressed as a unique summation of at most n (not necessarily distinct) integers from $\{e_1, \ldots, e_s\}$ up to permutation of the summands; in particular, for every $j \in \{1, \ldots, s\}$, we have $re_j \in \mathcal{E}^+_{\leq n}$ and the only such summation for re_j is $\underbrace{e_j + \cdots + e_j}_{r\text{-times}}$. Thus, property (2) holds when property (1) also holds, that is, when D is sufficiently large. Writing K_n for the clique of n nodes, we derive an upper bound on the inner summation on the last line of $(*)$ for arbitrary $e \in \mathcal{E}^+_{\leq n}$: $\sum(\hom(G_1, A_{j_1}) \times \cdots \times \hom(G_r, A_{j_r})) \leq \sum(\hom(G_1, K_n) \times \cdots \times \hom(G_r, K_n)) = \sum \hom(G, K_n) \leq \sum n^n \leq r! \times n^n \leq n! n^n$, where $\sum$ is taken over all $j_1, \ldots, j_r \in \{1, \ldots, s\}$ with $e_{j_1} + \cdots + e_{j_r} = e$ and the second last inequality follows from the previous fact about unique summation up to permutation of summands. Set $D := n! n^n + 1$, then property (1) holds.

Finally, it remains to argue that, given two arbitrary graphs G and H of size n, they are isomorphic if and only if $\hom(G, F_0) = \hom(H, F_0)$. The 'only if' direction is trivial. Before we deal with the 'if' direction, let us notice that $\hom(G, F_0) > 0$ because $\hom(G, G) > 0$ and G is among $A_1, \cdots, A_s$, which are subgraphs of F_0; moreover, the exponents of D for the nonzero digits in the D-ary representation of $\hom(G, F_0)$ indicate the number of connected components there are in G, by the previous fact about unique summation up to permutation of summands (cf. the last line of $(*)$). The same also holds for H. Now, for the 'if' direction, assume that $\hom(G, F_0) = \hom(H, F_0)$, then they are identical when expressed in D-ary representation and hence agree on all digits. In particular, they have the same number of connected components. Furthermore, by property (2), we have $\hom(G, A_j) = \hom(H, A_j)$ for all $j \in \{1, \ldots, s\}$. It follows from the characterization result mentioned in Section 1 by Chaudhuri and Vardi that G and H are isomorphic.

References

[AKW21] Albert Atserias, Phokion G Kolaitis, and Wei-Lin Wu. On the expressive power of homomorphism counts. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2021.

[CFLX21] Yijia Chen, Jörg Flum, Mingjun Liu, and Zhiyang Xun. On queries determined by a constant number of homomorphism counts. *arXiv preprint arXiv:2111.13269*, 2021.

[CV93] Surajit Chaudhuri and Moshe Y. Vardi. Optimization of *Real* conjunctive queries. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1993*, pages 59–70, 1993.

[Dvo10] Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, 2010.

[Lov67] László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3-4):321–328, 1967.

DATALOG REDUCTIONS BETWEEN CONSTRAINT SATISFACTION PROBLEMS

JAKUB OPRŠAL

The *constraint satisfaction problem (CSP)* is a prototypical NP-complete problem. Loosely speaking, the goal is given an instance consisting of variables, that attain values over some (usually finite) domain, and constraints, each involving a finite number of variables and given by the set of permissible tuples, decide whether there is an assignment of values to the variables that satisfies all the constraints. A focus of research in the past years has been to classify the complexity of this problem depending on the shape of the constraints allowed. This is better explained when the CSP is expressed as a homomorphism problem: Given two finite structures $\mathbf{A}$ and $\mathbf{B}$ in the same relational language, decide whether there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. Fixing the structure $\mathbf{B}$, we obtain the problem CSP($\mathbf{B}$) whose complexity depends on properties of the structure $\mathbf{B}$, e.g., if $\mathbf{B}$ is the 3-clique, the problem is NP-hard, while if relations of $\mathbf{B}$ are defined by affine equations over a finite field, the problem is in P.

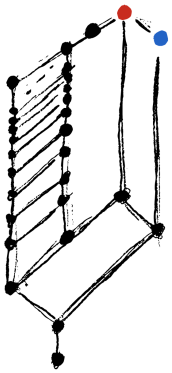
Algebraic approach developed by Jeavons et al. [JCG97, BJK05] was the go to theory for approaching complexity classification of these problems for past few decades, and it played a key role in Bulatov's and Zhuk's celebrated results [Bul17, Zhu20] that showed that, for any finite $\mathbf{B}$, CSP($\mathbf{B}$) is NP-hard or in P. In the core, this theory is usually expressed by claiming that the complexity of the problem CSP($\mathbf{B}$) only depends on certain algebraical structure assigned to the template $\mathbf{B}$ that is called *polymorphisms of* $\mathbf{B}$. In the talk, we would like to shift the focus from this abstract statement to the structure of reductions between different CSPs. In that light, the algebraic approach can be seen as a precise classification of existence of a very clean, structured, and consequently, log-space computable reduction between two CSPs in terms of polymorphisms of the two involved templates. We call the reductions coming from this approach *pp-replacements*.[1]

The notion of a *reduction* is fundamental to computational complexity. A reduction from one problem to another is a (usually efficiently computable) function that on input gets instances of one problem and outputs instances of another problem with equivalent answers: for decision problems we require that positive instances get mapped to positive instances and negative instances get mapped to negative instances.

In the world of CSPs, pp-replacements were sufficient for a long time for many classifications. In particular, pp-replacements are enough to provide all NP-hardness

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

[1] 'pp' stands for *primitive positive*.



(A) pp-replacements

(B) local reductions

FIGURE 1. Boolean CSPs ordered by pp-replacements and local reductions.

arguments for CSP($\mathbf{B}$) for a finite $\mathbf{B}$, and hence the hardness side of the dichotomy conjecture. The need for new and stronger reductions arose from development of a certain structural-approximation of CSPs—called *promise CSPs*—which includes among many other problems *approximate graph colouring*. In approximate graph colouring we fix $c > k$, and ask, given a graph that is *promised* to be k-colourable, to find a c-colouring of such a graph. While it is generally believed that this problem is NP-hard for all $c > k \geq 3$, it has only been shown to be NP-hard for some values of c and k, e.g., $c \geq 3$ and $k = 2c - 1$ [BKO19] (better bounds for $c > 5$ has been shown in [WŽ20]). In the world of promise CSP it is known that pp-replacements do not provide all NP-hardness, i.e., that are NP-hard promise CSPs whose hardness cannot been shown by reducing from, e.g., 3-SAT by a pp-replacement.

In the talk we will talk about a formalism for a wider class of reductions between (promise) CSPs that we call *local reductions*. In short, we extend pp-replacements with (monotone) Datalog interpretations. Apart from the motivation for promise CSPs outlined above, we hope that this formalism could bring better insights in understanding the CSP dichotomy (a.k.a., Bulatov-Zhuk theorem). This is led by the following observation about Boolean CSPs (which were classified by Schaefer [Sch78]). Assuming P $\neq$ NP, there are only three different classes of Boolean CSPs if we identified those that are inter-reducible by local reductions: the class of NP-complete CSPs, the class of XOR-SAT or linear equations over $\mathbb{Z}_2$, and the class of the trivial Boolean CSPs (see Figure 1b). Note that if we considered pp-replacements, there would still be infinitely many classes—even though the lattice of them is well understood (see Figure 1a). We also hope that this better understanding of CSP dichotomy could lead to a reasonable description of expressibility of CSPs in some logics. It is likely (though our current argument is not yet fully rigorous) that local reductions are expressible in Datalog, and hence in fixed point logic. This means that local reductions are likely to preserve many logical properties of (promise) CSPs.

We can show some preliminary results about local reductions including:

- describing a *normal form* of such a reduction which gives hopes of a possible classification similar to the classification of pp-replacements, and
- classification of a subclass of local reductions analogous to the arc-consistency algorithm for CSPs.

Finally, let us finish with a conjecture that is related to several algorithms that have been recently suggested for (promise) CSPs, namely a Sherali-Adams levels of Brakensiek-Guruswami algorithm described and classified in [BGWŽ20] and *cohomological k-consistency* described by Adam Ó Conghaile [Con22]. We conjecture that every finite template CSP that satisfies the (under P $\neq$ NP necessary) condition for tractability by Bulatov-Zhuk theorem is locally reducible to solving systems of linear equations over integers. If this was true, it would immediately give that both above algorithms and also a simple algorithm based on a combination of local consistency and solving affine linear equations would solve every tractable CSP, and it could also lead to showing that tractable finite template CSPs are expressible in fixed point logic with rank operators—assuming more structural properties of those CSPs would be investigated.

Acknowledgements. This is a joint work with Victor Dalmau (UPF, Barcelona) and Marcin Wrochna (University of Warsaw).

References

[BGWŽ20] Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Živný. The power of the combined basic LP and affine relaxation for promise CSPs. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:4, 2020. https://eccc.weizmann.ac.il/report/2020/004.

[BJK05] Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.

[BKO19] Jakub Bulín, Andrei Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19)*, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316300.

[Bul17] Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, Berkeley, CA, USA, October 2017. IEEE. arXiv:1703.03021, doi:10.1109/FOCS.2017.37.

[Con22] Adam Ó Conghaile. Cohomological k-consistency. (manuscript), Jan 2022. https://aconghaile.github.io/cohom_consistency.pdf.

[JCG97] Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997. doi:10.1145/263867.263489.

[Sch78] Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

[WŽ20] Marcin Wrochna and Stanislav Živný. Improved hardness for H-colourings of G-colourable graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1426–1435, Salt Lake City, UT, USA, 2020. SIAM. arXiv:1907.00872, doi:10.1137/1.9781611975994.86.

[Zhu20] Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, August 2020. doi:10.1145/3402029.

University of Oxford (until 31 May), UK
Email address: math@jakub-oprsal.info

Proaperiodic monoids via prime models

Sam van Gool*

May 27, 2022

This talk is on joint work with Benjamin Steinberg[1].

Regular languages of finite words coincide with the sets that are definable in monadic second order logic on finite linear orders [2]. Finite monoids are algebraic structures, similar to deterministic finite automata, that can be used to recognize regular languages. Among the finite monoids, a special role is played by the class **Ap** of finite *aperiodic* monoids, i.e., finite monoids which do not contain any non-trivial subgroups. Finite aperiodic monoids were shown by Schützenberger, McNaughton and Papert to recognize exactly the sets of finite words that are definable in first-order logic.

Profinite monoids are commonly used as limiting structures for classes of finite monoids. Indeed, the category of profinite monoids may be defined as the subcategory of topological monoids consisting of the projective limits of finite monoids, when we view the latter as discrete spaces. Elements of profinite monoids may then be seen as limit points of sequences of finite words with respect to their values in finite monoids. Restricting to the class of finite aperiodic monoids, *proaperiodic monoids* are those which are projective limits of finite aperiodic monoids. They may also be characterized as the profinite monoids in which, for any element x, the unique idempotent power in its orbit-closure (x^ω), is stabilized on the right by x; in an equation: $x^\omega \cdot x = x^\omega$. Thus, we obtain a three-way equivalence:

$$\text{FO-definable} \equiv \mathbf{Ap}\text{-recognizable} \equiv [\![x^\omega = x^{\omega+1}]\!].$$

The aim of the work that I will report on in this talk is to combine these well-known results with methods of first-order model theory, in order to shed new light on free finitely generated proaperiodic monoids.

*IRIF, Université Paris Cité, France.
[1]Department of Mathematics, City College at City University of New York, USA.

First, in the two papers [3, 4], we showed that free finitely generated proaperiodic monoids can be usefully understood as topological monoids of elementary equivalence classes of *pseudofinite words*, i.e., possibly infinite structures that are models of the first order theory of finite words. In particular, we showed there that every such elementary equivalence class contains an ω-saturated member, and that algebraic operations such as concatenation, idempotent power, and in fact more generally any substitutions of structures into others, are well-defined on the ω-saturated models; this closure under substitutions is related to the Feferman-Vaught theorem of first-order logic.

Subsequently to our conference publication [3], Almeida et al. [1] gave an alternative approach to a class of proaperiodic monoids that includes the free finitely generated ones. Their work explicitly associates a particular labeled linear order of so-called 'step points' to any element of a free finitely generated proaperiodic monoid. We will show in this talk that this labeled linear order of step points in fact also admits a model-theoretic interpretation. Indeed, we show that the linear order of step points is isomorphic to the *prime model* for the element, up to a one-point difference.

In the rest of this abstract, we give some more details on an original proof that any element of a free finitely generated proaperiodic monoid admits a prime model, which is independent of the results of [1]. From this, using the general model-theoretic fact that prime models are unique up to isomorphism, we immediately obtain a new proof of one of the main theorems of [1], which was proved there without model theory, but using rather involved combinatorial arguments [1, Sec. 11], that we can now circumvent. For the sake of brevity, we assume some of the notations of [4, 1], and the model-theoretic definitions and notations of [5].

Theorem 1. *Let T be a complete first-order theory extending the first-order theory of finite words. Then T has a prime model.*

Proof (sketch). By model theory (see e.g., [5, Thm. 4.2.10]), it suffices to prove that, for every n, the set of isolated n-types for T is dense in the set of all n-types for T. To this end, assume a formula $\phi(\overline{x})$ is consistent with T. In an ω-saturated model W for T, there is a tuple $\overline{a}$ such that $W, \overline{a} \models \phi(\overline{x}) \wedge \forall \overline{y}(\overline{y} <_{\text{lex}} \overline{x} \rightarrow \neg\phi(\overline{y}))$, i.e., $\overline{a}$ is the lexicographically minimal witness of $\phi(\overline{x})$. Here, the relation $<_{\text{lex}}$ is the lexicographic order on tuples, which can be defined from $<$. The n-type of the tuple $\overline{a}$ is isolated by the formula just given. □

Consider an element w of $\widehat{\mathsf{F}}_{\mathrm{AP}}(A)$, the free pro-aperiodic monoid over A. The *category of transitions* $\mathcal{T}(w)$ of w has as its objects pairs $(u, v) \in \widehat{\mathsf{F}}_{\mathrm{AP}}(A)^2$, and morphisms $t\colon (u, v) \to (u', v')$ are elements $t \in \widehat{\mathsf{F}}_{\mathrm{AP}}(A)$ such that $ut = u'$ and $tv' = v$. The preorder $(u, v) \preceq (u', v')$ is defined by saying there exists a morphism $(u, v) \to (u', v')$ in $\mathcal{T}(w)$, and the structure $\mathcal{L}(w)$ defined in [1, Sec. 4] is the quotient of the objects of $\mathcal{T}(w)$ by the induced equivalence relation $\equiv$ defined as $\preceq \cap \succeq$.

Recall that the *step points* of $\mathcal{L}(w)$ are by definition the points that are either the minimum, maximum, or have a predecessor or successor in the order. It follows from the proof of [1, Prop. 7.5] that $[(u, v)] \in \mathcal{L}(w)$ is a step point if, and only if, the endomorphism monoid of (u, v) in $\mathcal{T}(w)$ is trivial. One may show that the latter happens if, and only if, the type of (u, v) is isolated. We then obtain the following theorem. Let us denote by $\mathcal{L}'(w)$ the total order $\mathcal{L}(w)$ minus its maximum point, $(w, 1)$.

Theorem 2. *Let $w \in \widehat{\mathsf{F}}_{\mathrm{AP}}(A)$ and let T be the corresponding complete theory extending the theory of pseudo-finite words. The prime model of T is isomorphic to the step points of $\mathcal{L}'(w)$.*

A key result of [1], Theorem 8.7, is that the cluster words $\mathcal{L}_c(u)$ and $\mathcal{L}_c(v)$ are isomorphic if and only if $u = v$. Note that the proof of Theorem 8.7 in [1] relies on the intricate analysis in Sections 9–11. But this is now an easy consequence of Theorem 2. Indeed, if $\mathcal{L}_c(u)$ and $\mathcal{L}_c(v)$ are isomorphic, then this means in particular by Theorem 2 that the prime models for the theories of u and v are isomorphic, but then the theories are in fact the same, so $u = v$.

References

[1] J. Almeida, A. Costa, J. C. Costa, and M. Zeitoun, *The linear nature of pseudowords*, Publ. Mat. **63** (2019), no. 2, 361–422.

[2] J. R. Büchi, *Weak second-order arithmetic and finite automata*, Z. Math. Logik Grundlag. Math. **6** (1960), no. 1-6, 66–92.

[3] S. J. v. Gool and B. Steinberg, *Pro-aperiodic monoids via saturated models (conference version)*, in: STACS, 2017, pp. 39:1–39:14.

[4] ———, *Proaperiodic monoids via saturated models*, Israel J. Math. **234** (2019), 451–498.

[5] D. Marker, *Model Theory: An Introduction*, Graduate texts in mathematics, vol. 217, Springer-Verlag New York, 2002.

Parsing as a lifting problem and the Chomsky-Schützenberger representation theorem

Paul-André Melliès
IRIF, Université Paris Cité
CNRS, Inria, Paris, France

Noam Zeilberger
LIX, École Polytechnique
Palaiseau, France

Keywords: context-free languages, parsing, finite state automata, category theory, operads, representation theorem

Building on our work on type refinement systems, we continue developing the thesis that many kinds of deductive systems may be usefully modelled as functors and derivability as a lifting problem, focusing in this work on derivability in context-free grammars.

We begin by explaining how derivations in any context-free grammar G may be naturally encoded by a functor of operads

$$p : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[\Sigma]$$

from a freely generated operad into a certain "operad of spliced words". This motivates the introduction of a more general notion of context-free grammar over any category C, defined as a finite species $\mathcal{S}$ equipped with a color denoting the start symbol, and a map of species

$$\phi : \mathcal{S} \to \mathcal{W}[C]$$

into the *operad of spliced arrows* in C, which induces a unique functor

$$p : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[C]$$

generating a context-free language of arrows in C. Many standard properties of context-free grammars can be usefully formulated as properties of either the species $\mathcal{S}$, the map of species ϕ, or the functor of operads p. We also show that usual closure properties of context-free languages generalize to context-free languages of arrows.

One advantage of considering parsing as a lifting problem is that it enables a dual fibrational perspective on the functor p via the notion of "displayed" operad p^{-1} defined as a lax functor of operads $\mathcal{W}[C] \to \mathrm{Span}(\mathrm{Set})$. We show that displayed free operads admit an explicit inductive definition and use this to give a

reconstruction of Leermakers' generalization of the CYK parsing algorithm. We then turn to the Chomsky-Schützenberger Representation Theorem. We start by explaining that a non-deterministic finite state automaton over words, or more generally over arrows of a category, can be seen as a functor of categories satisfying the unique lifting of factorizations (ULF) property and the finite fiber property, and how every pair (q_0, q_f) of initial and final states induces a regular language of arrows. Then, we explain how to extend this notion of automaton to functors of operads, which generalize tree automata, allowing us to lift an automaton over a category to an automaton over its operad of spliced arrows. We show that every context-free grammar over a category can be pulled back along a non-deterministic finite state automaton over the same category, and hence that context-free languages are closed under intersection with regular languages.

The last and important ingredient is the identification of a left adjoint

$$C[-] : \text{Operad} \to \text{Cat}$$

to the operad of spliced arrows functor

$$\mathcal{W}[-] : \text{Cat} \to \text{Operad}$$

This construction builds the contour category $C[O]$ of any operad O, whose arrows have a geometric interpretation as "oriented contours" of the operations of O. A direct consequence of the contour / splicing adjunction is that every finite species equipped with a color induces a universal context-free grammar, generating a language of tree contour words. Finally, we prove a generalization of the Chomsky-Schützenberger Representation Theorem, establishing that any context-free language of arrows over a category C is the functorial image of the intersection of a C-chromatic tree contour language and a regular language.

An extended abstract has been submitted to the MFPS 2022 conference.

Monoidal Width
Extended Abstract

Elena Di Lavore and Paweł Sobociński

Tallinn University of Technology

Introduction. In applied category theory, monoidal categories are used as algebras of processes [11, 18, 7]. The assignment of semantics is, typically, a monoidal functor. The semantics of a process can thus be computed compositionally, but the efficiency of this computation often depends on how a process is decomposed, in terms of alternations between compositions and monoidal products. Different decomposition can have very different computational costs: in fact, sequential compositions typically involve some synchronization or resource sharing along the common boundary, which translates into additional computational effort, while monoidal products do not involve communication between the two processes, thus they do not require extra computations (Figure 1).

Figure 1: Two monoidal decompositions of the same morphism, the right one being the cheapest.

A second motivation comes from the graph theory literature. Motivated by algorithmic results [5, 6, 12], several measures of complexity for graphs have been defined [3, 19, 22, 21, 23, 20, 13, 2, 10]. Among these, we will be concerned with tree width [3, 19, 22], path width [21], branch width [23] and rank width [20]. All of these works rely on various *implicit* algebras of graph decomposition. Our goal is to make them explicit as particular monoidal categories.

We define monoidal decompositions (Definition 1) and the notion of monoidal width (Definition 3). We then report four results relating monoidal width and two variations with path width, tree width, branch width and rank width (Table 1). The details can be found in [17, 16].

Monoidal Width is the cost of the most efficient decomposition of a morphism into its atomic components, thus capturing—intuitively—its intrinsic structural complexity.

Definition 1 (Monoidal decomposition)**.** Let C be a monoidal category and $\mathcal{A}$ be a subset of its morphisms referred to as *atomic*. The *monoidal decompositions* D_f of $f\colon A \to B$ in C are:

$$
\begin{array}{rcll}
D_f & ::= & (f) & \text{if } f \in \mathcal{A} \\
 & | & (d_1, \otimes, d_2) & \text{if } d_1 \in D_{f_1},\ d_2 \in D_{f_2} \text{ and } f =_{\mathsf{C}} f_1 \otimes f_2 \\
 & | & (d_1, ;_X, d_2) & \text{if } d_1 \in D_{f_1\colon A \to X},\ d_2 \in D_{f_2\colon X \to B} \text{ and } f =_{\mathsf{C}} f_1 \,;\, f_2
\end{array}
$$

To calculate the cost of a decomposition, each operation and each atomic morphism is associated with a number, which we call weight. Roughly speaking, sequential composition is priced according to the size of the object the composition occurs over, while monoidal products are free. Finally, the weight of an atom is the application-specific cost of computing its semantics.

Definition 2. Let C be a monoidal category and let $\mathcal{A}$ be a set of atoms. A *weight function* is a function $\mathsf{w}\colon \mathcal{A} \cup \{\otimes\} \cup \mathsf{Obj}(\mathsf{C}) \to \mathbb{N}$ such that $\mathsf{w}(X \otimes Y) = \mathsf{w}(X) + \mathsf{w}(Y)$, and $\mathsf{w}(\otimes) = 0$.

Definition 3 (Monoidal width [17])**.** Let w be a weight function for $(\mathsf{C}, \mathcal{A})$. Let f be in C and $d \in D_f$. The width of d is defined recursively as follows:

$$\mathsf{wd}(d) \coloneqq \begin{cases} \mathsf{w}(f) & \text{if } d = (f) \\ \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2)\} & \text{if } d = (d_1, \otimes, d_2) \\ \max\{\mathsf{wd}(d_1), \mathsf{w}(X), \mathsf{wd}(d_2)\} & \text{if } d = (d_1, ;_X, d_2) \end{cases}$$

The *monoidal width* of f is $\mathsf{mwd}(f) \coloneqq \min_{d \in D_f} \mathsf{wd}(d)$. Restricting the operations allowed in decompositions to (i) precompositions with atoms and monoidal products or (ii) only compositions, we obtain: (i) monoidal tree width, mtwd, and (ii) monoidal path width, mpwd.

Monoidal Width Meets Graph Widths. Path width [21] and tree width [22] intuitively measure the difficulty of decomposing a graph into subgraphs forming a path and tree shape, respectively. On the other hand, branch width [23] and rank width [20] intuitively measure the difficulty of decomposing a graph into one-edge and one-vertex subgraphs, respectively.

In the case of path, tree and branch width, the subgraphs are obtained by "cutting" the original graph along its vertices. The right algebra to capture these decompositions is that given by cospans of graphs [24, 14], where the composition of two cospans is by pushout, which glues graphs along the vertices in their common boundary.

In the case of rank width, instead, the subgraphs are obtained by "cutting" the original graph along its edges. The correct algebra to capture rank width is, then, that given by a prop of graphs with dangling edges [9, 15]. This prop is more linear algebraic in nature, with adjacency matrices and matrix algebra playing a central role.

To back these claims, we established the results in Table 1, where $\bullet\!\!-\!\!G\!\!-\!\!\bullet \coloneqq \emptyset \to G \leftarrow \emptyset$ is the cospan associated to a graph G and $[G]$ is its corresponding graph with dangling edges.

$\mathsf{pwd}(G)$	$=$	$\mathsf{mpwd}(\bullet\!\!-\!\!G\!\!-\!\!\bullet)$	$=$	$\mathsf{pwd}(G)$	([17], Theorem 45)
$\mathsf{twd}(G)$	$\leq$	$\mathsf{mtwd}(\bullet\!\!-\!\!G\!\!-\!\!\bullet)$	$\leq$	$2 \cdot \mathsf{twd}(G)$	([17], Theorem 41)
$\frac{1}{2} \cdot \mathsf{bwd}(G)$	$\leq$	$\mathsf{mwd}(\bullet\!\!-\!\!G\!\!-\!\!\bullet)$	$\leq$	$\mathsf{bwd}(G) + 1$	([17], Theorem 49)
$\frac{1}{2} \cdot \mathsf{rwd}(G)$	$\leq$	$\mathsf{mwd}([G])$	$\leq$	$2 \cdot \mathsf{rwd}(G)$	([16], Theorem 5.12)

Table 1: Summary of the results

Conclusions and future work. Monoidal width measures the complexity of decomposing morphisms in monoidal categories. By identifying the correct algebras of decomposition, this framework allows us to capture path width, tree width, branch width and rank width.

We want to establish a result similar to Courcelle's theorem [12], relating logical expressibility with efficient recognisability in families of morphisms with bounded monoidal width. We are also working on capturing other graph widths such as clique [13], twin [8] and cut width [2, 10], and generalisations of tree width to directed graphs [4] and relational structures [1].

References

[1] Samson Abramsky, Anuj Dawar, and Pengming Wang. The pebbling comonad in finite model theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

[2] D. Adolphson and T. C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

[3] Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973.

[4] Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer, and Jan Obdržálek. The dag-width of directed graphs. *Journal of Combinatorial Theory, Series B*, 102(4):900–923, 2012.

[5] Hans L Bodlaender. A tourist guide through treewidth. *Technical report*, 1992.

[6] Hans L Bodlaender and Arie MCA Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

[7] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A survey of compositional signal flow theory. In Michael Goedicke, Erich J. Neuhold, and Kai Rannenberg, editors, *Advancing Research in Information and Communication Technology*, volume 600 of *IFIP Advances in Information and Communication Technology*, pages 29–56. Springer, 2021. doi:10.1007/978-3-030-81701-5_2.

[8] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width i: tractable fo model checking. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 601–612. IEEE, 2020.

[9] Apiwat Chantawibul and Paweł Sobociński. Towards compositional graph theory. *Electronic Notes in Theoretical Computer Science*, 319:121–136, 2015.

[10] Maria Chudnovsky and Paul Seymour. A well-quasi-order for tournaments. *Journal of Combinatorial Theory, Series B*, 101(1):47–53, 2011.

[11] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes - A first course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[12] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990.

[13] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[14] Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Paweł Sobociński. A canonical algebra of open transition systems. In Gwen Salaün and Anton Wijs, editors, *Formal Aspects of Component Software*, pages 63–81, Cham, 2021. Springer International Publishing.

[15] Elena Di Lavore, Jules Hedges, and Paweł Sobociński. Compositional modelling of network games. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[16] Elena Di Lavore and Paweł Sobociński. Monoidal Width: Capturing Rank Width, 2022. arXiv:2205.08916.

[17] Elena Di Lavore and Paweł Sobociński. Monoidal Width: Unifying Tree Width, Path Width and Branch Width, 2022. arXiv:2202.07582.

[18] Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020.

[19] Rudolf Halin. S-functions for graphs. *Journal of geometry*, 8(1-2):171–186, 1976.

[20] Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[21] Neil Robertson and Paul D. Seymour. Graph minors. I. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983.

[22] Neil Robertson and Paul D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

[23] Neil Robertson and Paul D. Seymour. Graph minors. X. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

[24] Robert Rosebrugh, Nicoletta Sabadini, and Robert FC Walters. Generic commutative separable algebras and cospans of graphs. *Theory and applications of categories*, 15(6):164–177, 2005.

From profinite words to profinite λ-terms

Vincent Moreau, IRIF, Université Paris Cité

This is joint work with Sam van Gool and Paul-André Melliès.

The aim of this work is to combine methods from profinite algebra and models of the λ-calculus to obtain a notion of *profinite λ-term*. Profinite algebraic structures have been used in automata theory for giving classifications of regular languages, see e.g. [2] for a survey. In particular, elements of the free profinite monoid are known as *profinite words*. They provide a way to speak about limiting behavior of finite words with respect to deterministic automata. In order to connect this notion of profiniteness to the λ-calculus, we generalize from usual automata models to the computation model of *higher-order automata* [4, 1]. Indeed, higher-order automata generalize automata of words and trees, in that they can process arbitrary simply-typed λ-terms as their input. Existing notions of automata on words and trees are then obtained as a special case through Church encoding.

Our main contribution here is the study of the notion of parametric λ-term, parametric in the sense of Reynolds [3] and relatively to any model of the simply-typed λ-calculus, and the establishment of a link with profinite words. We first consider a model constructed from finite sets and functions, which corresponds to deterministic automata. We prove that in that setting, the notion of parametric λ-term essentially coincides with the classical notion of profinite word. However, when we move for example from functions to relations, which corresponds to the move from deterministic to non-deterministic automata, we will obtain a new notion of 'non-deterministically profinite' λ-term. The abstract point of view that we introduce here will thus allow us, in future work, to investigate parametric λ-terms for other models, obtaining a new landscape of generalizations of profinite words.

In the remainder of this abstract, we will describe our work in more technical detail.

Let Σ be a finite alphabet. A *profinite word* over Σ is a family $u = (u_p)$, where p ranges over the surjective monoid homomorphisms $p\colon \Sigma^* \twoheadrightarrow M$, with M a finite monoid, such that, for every p, u_p is an element of the range of p, and for any monoid homomorphism $f : M \to N$, with N a finite monoid,

$$u_{f\circ p} = f(u_p).$$

Every finite word $w \in \Sigma^*$ yields a profinite word (w_p) by defining w_p to be $p(w)$, for each $p\colon \Sigma^* \twoheadrightarrow M$. However, there exist many profinite words that do not come from finite words: for example, for any letter $a \in \Sigma$, denote by u_p is the unique idempotent power of $p(a)$ in the finite monoid M, for each $p\colon \Sigma^* \twoheadrightarrow M$. Then (u_p) is a profinite word, which is usually denoted by "a^ω" and does not arise from a finite word.

The first step towards the definition of parametric λ-terms is the following alternative characterization of profinite words. Let us define a *monoidal relation* from M to N to be a submonoid of $M \times N$. It can be shown that, for any $p : \Sigma^* \twoheadrightarrow M$, $q : \Sigma^* \twoheadrightarrow N$, R a monoidal relation from M to N, and u a profinite word,

$$\text{if, for all } w \in \Sigma^*,\ p(w)\ R\ q(w),\ \text{ then } u_p\ R\ u_q.$$

Conversely, any family (u_p) which verifies this condition is a profinite word.

The second step is to see how words may be encoded within the simply-typed λ-calculus. Recall that the untypes λ-terms are those generated by the grammar

$$M, N ::= x \mid \lambda x.M \mid MN.$$

A simple type (with one type variable $\mathfrak{o}$) is a term generated by the grammar

$$A, B ::= \mathfrak{o} \mid A \Rightarrow B.$$

Let Γ be a context, namely a set of pairs $x : A$ consisting of a variable and of a simple type. A λ-term M has type A in context Γ iff the judgment $\Gamma \vdash M : A$ can be derived in the system sith the three following typing rules

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash M : B \Rightarrow A \quad \Gamma \vdash N : B}{\Gamma \vdash MN : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \Rightarrow B}$$

When a λ-term M is closed, i.e. when every occurring variable is bound by a λ-abstraction, we say that M has type A if $\varnothing \vdash M : A$ is derivable.

As an example, we consider the two letter alphabet $\{a, b\}$. For any word $w = a_1 \dots a_n \in \{a, b\}^*$, one can derive the typing judgment in the simply-typed λ-calculus

$$a : \mathfrak{o} \Rightarrow \mathfrak{o}, b : \mathfrak{o} \Rightarrow \mathfrak{o}, c : \mathfrak{o} \vdash a_1(\dots(a_n c)) : \mathfrak{o}.$$

It follows that the closed term $\lambda a.\lambda b.\lambda c.a_1(\dots(a_n c))$ has type

$$(\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow (\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow (\mathfrak{o} \Rightarrow \mathfrak{o}).$$

In general, any word $w \in \Sigma^*$ can be encoded in this way as a closed term having type

$$\mathbf{Church}_\Sigma := \underbrace{(\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow \dots \Rightarrow (\mathfrak{o} \Rightarrow \mathfrak{o})}_{|\Sigma| \text{ times}} \Rightarrow (\mathfrak{o} \Rightarrow \mathfrak{o}).$$

The third step is to consider logical relations in the sense of Reynolds [3]. As the category **FinSet** of finite sets and functions is cartesian closed, we can interpret

the simply-typed λ-calculus in it. Given a set P and defining $[\![\mathtt{o}]\!]_P := P$, structural induction on simple types yields a set $[\![A]\!]_P$ for every simple type A. We extend this interpretation of types by defining, for every binary relation R between P and Q, the binary relation $[\![A]\!]_R$ between $[\![A]\!]_P$ and $[\![A]\!]_Q$ in the following way: we let $[\![\mathtt{o}]\!]_R$ be R itself and define, for any simple types A and B, the set $[\![A \Rightarrow B]\!]_R$ to be

$$\{(f,g) \in [\![A \Rightarrow B]\!]_P \times [\![A \Rightarrow B]\!]_Q \mid \forall (p,q) \in [\![A]\!]_R,\ (f(p), g(q)) \in [\![B]\!]_R\}.$$

We now state the central definition of this work that we investigate in the case of finite models.

Definition. A *parametric λ-term of type A* is a family of elements (θ_Q), where Q ranges over all finite sets, such that $\theta_Q \in [\![A]\!]_Q$, and for any binary relation R between finite sets P and Q, we have $\theta_P \ [\![A]\!]_R \ \theta_Q$.

We note that we have only given the definition for the category of finite sets and relations, but this same definition applies in any cartesian closed double category $\mathbf{D}$.

When A is taken to be $\mathbf{Church}_\Sigma$, θ_Q can be seen as a function from $[\![(\mathtt{o} \Rightarrow \mathtt{o})^{|\Sigma|}]\!]_Q$ to $[\![\mathtt{o} \Rightarrow \mathtt{o}]\!]_Q$, and the parametricity condition then becomes, for all families $(p_a)_{a\in\Sigma} : P \to P$ and $(q_a)_{a\in\Sigma} : Q \to Q$

$$\text{if, for all } a \in \Sigma,\ (p_a, q_a) \in [\![\mathtt{o} \Rightarrow \mathtt{o}]\!]_R, \text{ then } (\theta_P, \theta_Q) \in [\![\mathtt{o} \Rightarrow \mathtt{o}]\!]_R.$$

Now, any profinite word induces a parametric λ-term in **FinSet** by choosing the monoid M to be $Q \to Q$ with the composition. On the other hand, any parametric λ-term θ in **FinSet** induces a profinite word defined as

$$\begin{array}{ccc} [\Sigma^*, M] & \longrightarrow & M \\ p & \longmapsto & \theta_{(i_M \circ p)}(e_m) \end{array}$$

where i_M is the Cayley injection of M into $M \to M$. These two functions indeed yield profinite words and parametric λ-terms. Moreover, we prove the following, by making crucial use of the notion of parametricity with respect to relations.

Theorem. *The functions between profinite words and parametric λ-term in* **FinSet** *are mutually inverse.*

References

[1] Paul-André Melliès. "Higher-order parity automata". In: LICS 2017.

[2] Jean-Eric Pin. "Profinite Methods in Automata Theory". In: STACS 2009.

[3] John C. Reynolds. "Types, Abstraction and Parametric Polymorphism". In: IFIP Congress 1983.

[4] Sylvain Salvati. "Recognizability in the Simply Typed Lambda-Calculus". In: *Logic, Language, Information and Computation*. Springer Berlin Heidelberg, 2009, pp. 48–60.

Indexed complexity classes

Siddharth Bhaskar

Structure meets Power 2022—Extended Abstract

1 Structure, power, and indexings

Abstractly, a *programming language* is a set P of programs, a set D of a data, and a ternary semantics relation of type $P \times D \times D$, where $[\![p]\!](x) = w$ is interpreted as "program p terminates on input x with output w." Following [5], a programming language is *grounded* if programs can be interpreted as data, namely $P \subseteq D$. Henceforth, we restrict ourselves to grounded programming languages.

From this vantage, the *power* of a programming language can be identified with the class of functions (or relations, or partial functions ...) it computes, whereas its *structure* can be thought of as its theory in the first order language of $[\![]\!]$. No doubt this is a very coarse conception of structure; however, it suffices to express nontrivial structural properties of the programming language, such as whether it admits *syntactic composition*.[1]

For *Turing complete* languages, power and structure are almost completely independent. That is to say, all reasonable Turing complete programming languages have the same $[\![]\!]$-theory.[2] The situation for *non-Turing complete* programming languages is much better (or worse, depending on your perspective). Here, there is the possibility that different indexings of the same complexity class possess genuinely different structural properties. This affords the possibility of interplay between the structure and expressive power of a programming language.

There is one significant qualitative difference between indexings of partial recursive functions and of any complexity class: namely, the existence of a *universal program* u which satisfies the property $[\![u]\!](p, x) = [\![p]\!](x)$ for any program p and datum x.[3] Any non-pathological indexing of the partial recursive functions must contain a universal function, however, an indexing of a complexity class cannot, by a standard diagonalization argument.

On the other hand, there are natural axioms which an indexing of a complexity class *can* satisfy. Kozen [7] identifies three such simple axioms with short formal statements[4] Kozen shows that these axioms have certain familiar consequences; for example, any indexing satisfying them must satisfy Kleene's second recursion theorem.[5]

It is already an interesting question to ask: *which complexity classes satisfy these three axioms?* But more suggestively, Kozen suggests that indexings of some complexity classes obey *conservation of structure*: the more "synthesizing" programs a language admits (such as

[1] I.e., whether there is a program which takes as input the codes of two programs computing functions f and g and returns the code of a program computing $f \circ g$.

[2] In recursion theoretic lingo, this is simply to say that all acceptable universal indexings of partial recursive functions are recursively isomorphic.

[3] We silently assume a bit more structure on our data, viz., the existence of a *pairing function* $\langle \cdot, \cdot \rangle : D^2 \to D$.

[4] Two of which are the admission of parallel and sequential syntactic composition.

[5] Namely, $(\forall p \in P)(\exists q \in P)(\forall d \in D)\ [\![p]\!](q, d) = [\![q]\!](d)$. Note that in the absence of a universal program, Kleene's recursion theorem is different from *Roger's fixpoint theorem*, which will not hold in any indexed complexity class.

a syntactic composer), the more complex it must be to uniformly simulate a program in that language. He then proves it for a particular family of indexings of PTIME.

We feel that this is a fascinating open question, squarely in the realm of "structure meets power," viz., *to be powerful, must a programming language be structurally complex?* But besides raising it, we have nothing to say about this matter. Rather, the point of this example is that studying indexings of complexity classes can be a powerful tool for examining structure-power relationships.

2 Subrecursion theory meets ICC

One criticism of these questions is that they are no longer in vogue: recursion-theoretic techniques are not as big a part of complexity theory as they once were, and the study of indexings of complexity classes (or, to use the older terminology, *subrecursive indexings*), mostly fizzled out by the 1980's.

We believe that, on the contrary it is high time to revisit the theory of subrecursive indexings in light of the proliferation of new examples from the field of *implicit computational complexity*, or ICC, from the 1990's onward. Before then, most non-Turing complete languages captured primitive recursive functions, or at least an initial segment of the Grzegorczyk hierarchy. Indexings of complexity classes lower than ELEMENTARY were usually obtained by families of clocked Turing machines.

Now, by contrast, we have a slew of indexings of PTIME, PSPACE, and other "small" classes, obtained by techniques like tiered recursion, linear typing discipline, cons-free computation, among others [4, 6, 8, 9]. What indexings of their respective complexity classes do they induce? Are these indexings isomorphic to those obtained by clocked Turing machines, or genuinely new? Can we quantify their complexity?

Not only are these new examples from the perspective of subrecursion theory, but these are (we believe) new questions from the perspective of ICC.

3 A test case: speedup phenomena

We conclude by describing one specific technical question which we have taken some preliminary steps towards.

The $[\![]\!]$-theory of a programming language can be refined by expanding it with another ternary predicate, namely the *Blum relation* $\Phi_p(x) \leq t$, defined by "program p on input x terminates in time (or space, or other resource) at most t."[6] The resulting theory captures yet more structural properties of the original programming language than its $[\![]\!]$-theory.

Blum complexity theory is an extension of classical recursion theory which studies indexings of partial recursive functions expanded by a recursive Blum relation [3]. This framework is general enough to describe almost any model of computation with almost any resource bound. From these simple axioms, we get powerful results like the *speedup theorem*, which asserts, for any recursive function f, the existence of problems X which can always be sped up by a factor of f.[7] Crucially, speedup phenomena imply lower bounds, e.g., if a problem can be sped up by a quadratic factor, then it cannot be solved in polynomial time.

Blum's theory relies on the machinery of primitive recursive indexings, for example the existence of universal programs. Is there any way to recover a Blum-like theory for an indexed

[6] Of course, this requires that we identify numbers, or whatever we use to measure cost, as a subset of D.

[7] I.e., for any program solving X, there is a an f-faster program solving X.

complexity class which lacks such machinery? Surprisingly, the answer is yes. Alton [1] shows that sometimes all we need is a *simulation function*, namely a program *Sim* satisfying

$$[\![Sim]\!](p, x, t) = [\![p]\!](t) \text{ if } \Phi_p(x) \leq t.$$

In other words, we can evaluate a given program on a given input *if* we are also given the upper bound for the running time as a parameter.

The program we would like to suggest is to *look for speedup phenomena within non-Turing complete programming languages.* To get off the ground we need to find indexings of complexity classes which admit efficient simulation functions. By studying cons-free programs, we have found an indexing of Ptime with a Pspace simulating function; this shows that such indexings exist (or at least that their existence is hard to refute).

It also suggests that we can locate speedup phenomena within cons-free programs. Since cons-free running time is closely related to boolean circuit depth [2], which is the basis for many interesting classes between Logspace and Ptime we might hope to prove something like the following.

Conjecture. If Ptime=Pspace, then the NC hierarchy is strict.

The path to showing speedup and related Blum phenomena involves *structural* work; namely, showing the existence of various efficient synthesizing functions for cons-free programs.

References

[1] D. A. Alton. *"Natural" Programming Languages and Complexity Measures for Subrecursive Programming Languages: An Abstract Approach*, page 248–285. London Mathematical Society Lecture Note Series. Cambridge University Press, 1980.

[2] S. Bhaskar, C. Kop, and J. G. Simonsen. Subclasses of ptime interpreted by programming languages. *Theory of Computing Systems*, 2022. (Forthcoming).

[3] M. Blum. A machine-independent theory of the complexity of recursive functions. *J. ACM*, 14(2):322–336, Apr. 1967.

[4] M. Hofmann. The strength of non-size increasing computation. In *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '02, page 260–269. Association for Computing Machinery, 2002. ISBN 1581134509.

[5] N. D. Jones. *Computability and complexity from a programming perspective*. London: MIT Press, 1997. ISBN 0-262-10064-9.

[6] N. D. Jones. Logspace and ptime characterized by programming languages. *Theoretical Computer Science*, 228(1):151 – 174, 1999.

[7] D. Kozen. Indexing of subrecursive classes. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 287–295, New York, NY, USA, 1978. Association for Computing Machinery.

[8] D. Leivant. Ramified recurrence and computational complexity i: Word recurrence and poly-time. In *Feasible Mathematics II*, pages 320–343. Birkhäuser Boston, 1995.

[9] J. Marion. A type system for complexity flow analysis. In *2011 IEEE 26th Annual Symposium on Logic in Computer Science (LICS 2011)*, pages 123–132, 2011.

Compositionality and Proof Complexity *

Gabriel Istrate†

May 25, 2022

1 Introduction

Proofs in propositional proof systems are often *compositional*: if P, Q, R are propositions such that $P \models Q$ and $Q \models R$, π_1 is a propositional proof for $P \models Q$ and π_2 is a propositional proof of $Q \models R$ then one can often "compose" π_1 and π_2 to obtain a proof $\pi_1 \circ \pi_2$ for $P \models R$. For instance, for resolution composition refers simply to the concatenation of the two proofs.

This simple fact, combined with the simple fact that many complexity measures c are also compositional, that is for any two proofs π_1, π_2, $c(\pi_1 \circ \pi_2)$ depends in a well behaved manner on $c(\pi_1), c(\pi_2)$[1], can be used to prove the existence of efficient propositional proofs of unsatisfiability for various propositional encodings of combinatorial statements.

This strategy (that we have first employed in [2] to prove the existence of quasi-polynomial size Frege proofs for the Kneser-Lovász theorem) was further exploited in [1] to prove the existence of polynomial (or quasi-polynomial, in some cases) Frege and extended Frege proofs for a variety of propositional encodings of various combinatorial principles, such as

- Schrijver's theorem [3], a generalization of the Kneser-Lovász theorem[2]
- vertex coloring.
- dual graph coloring.
- edge clique cover
- hitting set
- the Arrow and Gibbard-Satterthwaite theorems from the theory of social choice.

*Based on work [1] presented at ICALP 2021 (with Cosmin Bonchiş and Adrian Crăciun) and related, subsequent work-in-progress by the author of this presentation

†West University of Timişoara and the e-Austria Research Institute, Timişoara, Romania, email: gabrielistrate@acm.org

[1] e.g., for resolution size, $c(\pi_1 \circ \pi_2) \leq c(\pi_1) + c(\pi_2)$

[2] For some recent applications of our proof strategy for this result see [4]

Our strategy was to use the existence of a chain of reductions provided by the concepts of *data reduction/kernelization* from parameterized complexity [5]:

Definition 1 *A* parametrized problem *over alphabet* Σ *is a set* $L \subseteq \Sigma^* \times \mathbb{N}$.

Let L *be a parametrized problem. A* ***data reduction rule for*** L *is an algorithm* A *that maps (in time polynomial in* $|x| + k$*) an instance* (x, k) *of* L *to an instance* (x', k') *such that* $(x, k) \in L$ *iff* $(x', k') \in L$ *(we say that the two instances are* equivalent*, or that the reduction rule is* safe*), and* $|x'| \leq |x|$.

A ***kernelization algorithm*** *(or, shortly, kernelization) Ker for the problem* L *is an algorithm that works as follows: on input* (x, k)*, Ker outputs (in time polynomial in* $|(x, k)|$ *) a pair* (x', k')*, such that the following are true:* $(x, k) \in L$ *iff* $(x', k') \in L$*, and* $|x'|, k' \leq g(k)$*, where* g *is a computable function. Pair* (x', k') *is called* the kernel of (x, k)*, while* $g(k)$ *is called* the size of the kernel.

The length of a kernelization provided by a data reduction matters for obtaining applications to proof complexity. This is encapsulated by the following

Metatheorem 1 *Let* L *be a parameterized problem that is kernelizable via a finite number of data reduction rules* $(A_1, A_2, \ldots, A_r)$ *with kernel size* $g(\cdot)$.

1. *Assume that negative instance* (x, k) *of* L *has a data reduction chains of length* $C(x, k)$*, and that the soundness of each reduction rule* $A_1, A_2, \ldots, A_r$ *can be witnessed using extended Frege proofs of size at most* $h(|\Phi(x, k)|)$*, for some function* $h(\cdot)$*. Then* L *has extended Frege proofs of size* $O((\sum_{i=0}^{C(x,k)} R^i)[h(|\Phi(x, k)|) + 2^{O(poly(g(k)))}])$*. In particular, if* $R = 1$ *and for every fixed* k *we have* $C(x, k) = O(poly(|\Phi(x, k)|))$ *then, for every fixed* k*, negative instances* $\Phi(x, k)$ *of* L *have extended Frege proofs of size polynomial in* $|\Phi(x, k)|$.

2. *Assume that negative instances* (x, k) *of* L *have data reduction chains of length* $C(x, k) = O(1)$ *(*$O(log(|\Phi(x, k)|))$*, respectively), where the constant may depend on* k*, and that the safety of each reduction* $\Phi(x_i, k_i) \vdash \Phi(x_{i+1}, k_{i+1})$ *is witnessed by Frege proofs of size* $\leq p(|\Phi(x, k)|)$*, for some fixed polynomial* $p(\cdot)$ *Then for every fixed* k*, negative instances* $(\Phi(x, k), k)$ *of* L *have Frege proofs of size polynomial (quasipolynomial) in* $|\Phi(x, k)|$.

In the present talk:

- We aim to explain some of the technical details of our results in [1], including the novel data reductions of small length for the problems above.

- Time permitting, we will present subsequent work-in-progress. This work encompasses two directions:

    1. First, the application of a different technique from the theory of parameterized complexity, that of *iterative compression* [6].

2. Second, we discuss applications to proof system other than Frege and extended Frege. Our main target is the class of proof systems based on clause redundancy recently introduced in the SAT solving community [7, 8], specifically the proof system SPR^{-}.[9]. These are proof systems for which the logical equivalence of formulas generated through the proof is **not** guaranteed. Instead, these formulas are only *equisatisfiable* with original formulas. Such proof systems have important practical advantages, and the issue of guaranteeing efficient proofs in them is an important one.

References

[1] Gabriel Istrate, Cosmin Bonchiş, and Adrian Crăciun. Kernelization, proof complexity and social choice. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 135:1–135:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[2] James Aisenberg, Maria Luisa Bonet, Sam Buss, Adrian Crăciun, and Gabriel Istrate. Short proofs of the Kneser–Lovász coloring principle. *Information and Computation*, 261:296–310, 2018.

[3] A. Schrijver. Vertex-critical subgraphs of Kneser graphs. *Nieuw Arch. Wiskd., III. Ser.*, 26:454–461, 1978.

[4] Ishay Haviv. A fixed-parameter algorithm for the Schrijver problem. *arXiv preprint arXiv:2204.09009*, 2022.

[5] Fedor Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing.* Cambridge University Press, 2019.

[6] Fedor Fomin, Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Saket Saurabh. Iterative compression and exact algorithms. *Theoretical Computer Science*, 411(7-9):1045–1053, 2010.

[7] Marijn Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. *Journal of Automated Reasoning*, 64(3):533–554, 2020.

[8] Marijn Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. PRuning through satisfaction. In *Haifa Verification Conference*, pages 179–194. Springer, 2017.

[9] Neil Thapen and Sam Buss. DRAT and propagation redundancy proofs without new variables. *Logical Methods in Computer Science*, vol. 17, issue 2, 2021.