Polyregular Functions on Unordered Trees of Bounded Height

MIKOŁAJ BOJAŃCZYK, University of Warsaw, Poland
BARTEK KLIN, University of Oxford, UK

We consider injective first-order interpretations that input and output trees of bounded height. The corresponding functions have polynomial output size, since a first-order interpretation can use a k-tuple of input nodes to represent a single output node. We prove that the equivalence problem for such functions is decidable, i.e. given two such interpretations, one can decide whether, for every input tree, the two output trees are isomorphic.

We also give a calculus of typed functions and combinators which derives exactly injective first-order interpretations for unordered trees of bounded height. The calculus is based on a type system, where the type constructors are products, coproducts and a monad of multisets. Thanks to our results about tree-to-tree interpretations, the equivalence problem is decidable for this calculus.

As an application, we show that the equivalence problem is decidable for first-order interpretations between classes of graphs that have bounded tree-depth. In all cases studied in this paper, first-order logic and MSO have the same expressive power, and hence all results apply also to MSO interpretations.

CCS Concepts: • **Theory of computation** → **Transducers**; **Tree languages**; **Logic**.

Additional Key Words and Phrases: polyregular functions, first-order interpretation, unordered trees, bounded height

ACM Reference Format:
Mikołaj Bojańczyk and Bartek Klin. 2024. Polyregular Functions on Unordered Trees of Bounded Height. *Proc. ACM Program. Lang.* 8, POPL, Article 45 (January 2024), 26 pages. https://doi.org/10.1145/3632887

1 INTRODUCTION

This paper is about computational models that transform objects such as strings or trees, are powerful enough to describe interesting programs, but are weak enough to have a decidable halting problem. The point of departure is the class of *polyregular functions* [Bojańczyk 2022]. This is a class of string-to-string functions that contains functions such as

$$\underbrace{123 \mapsto 123123123}_{\text{squaring}} \qquad \underbrace{123 \mapsto 321}_{\text{reverse}},$$

and that can be defined by many equivalent models of computation, including: pebble transducers [Milo et al. 2003, Section 3.1], a certain imperative programming language [Bojańczyk 2022, Section 1], several functional programming languages with the same expressive power [Bojańczyk 2022, Sections 3 and 4] and [Bojańczyk 2023], and logical interpretations [Bojańczyk et al. 2019, Theorem 7]. The general idea is that the polyregular functions are those string-to-string functions which have polynomial output size, and which can be computed by devices similar to automata. Due to their similarity to finite automata, certain problems are known to be decidable for polyregular functions (see e.g. [Bojańczyk 2022, Corollary 1.5]). An outstanding open problem about polyregular

Authors' addresses: Mikołaj Bojańczyk, University of Warsaw, Poland; Bartek Klin, University of Oxford, UK.

ACM 2475-1421/2024/1-ART45
https://doi.org/10.1145/3632887

functions is decidability of the equivalence problem, i.e. it is not known if one can decide whether two devices (transducers, programs, interpretations) compute the same polyregular function. In this paper we make progress on this question by answering it in a different setup, where instead of strings the underlying data structure is nested multisets.

Let us begin by explaining the objects studied here. One description (see Section 2) is that these are data types that are constructed from finite sets using the multiset constructor

$$\mathsf{M}X = \text{finite multisets of elements from } X.$$

(We also allow products and coproducts, but it is only the multiset constructor that can create infinite sets.) We use the name *multiset types* for such types[1]. Another description of multiset types is that they describe trees, since nested multisets can be represented as trees, as explained in the following picture:

a multiset of multisets | the corresponding tree | the same tree, again

Importantly, the trees are unordered (there is no order on siblings, since multisets are unordered), and their height is bounded (by the nesting depth of the multiset constructor in a multiset type).

This paper is about functions on multiset types or, equivalently, unordered trees of bounded height. The contributions of this paper are:

(1) In Section 2-4, we introduce a notion of *polyregular function* for such objects. These are functions between multiset types (or, equivalently, unordered trees of bounded height), that have polynomial output size, and that coincide with regular languages in the case of Boolean outputs. One example of such a polyregular function is the product operation $(\mathsf{M}\Sigma) \times (\mathsf{M}\Gamma) \to \mathsf{M}(\Sigma \times \Gamma)$, that inputs two multisets, of cardinalities n and m, respectively, and outputs a single multiset of cardinality $n \cdot m$. This function has quadratic output size. Another example, which has linear output size, is the multiset union operation of type $\mathsf{MM}\Sigma \to \mathsf{M}\Sigma$. In our definition of polyregular functions, we propose two models, and prove that they are equivalent. The first model is first-order interpretations, which are a classical model of describing functions using logical formulas [Hodges 1993, Chapter 5]. The second model uses certain prime functions (such as multiset union) and combinators (such as sequential composition of functions), inspired by a similar system for string-to-string polyregular functions [Bojańczyk 2022, Section 3]. The proof that these models describe the same functions on multiset types is inspired by a similar result for strings [Bojańczyk et al. 2019, Theorem 7], except that in the multiset case we can avoid many technicalities that appear in the string case. One of the useful features of multiset types is that first-order (FO) logic has the same expressive power as monadic second-order (MSO) logic.

(2) In Sections 5-6, we prove that the equivalence problem is decidable for the polyregular functions on multiset types. This means that one can decide if two programs represent the same function, i.e. for every input (which is taken from some multiset type), the two possible outputs are equal. For strings, this problem remains open [Bojańczyk 2022, Section 8], and the decidability result on multisets from this paper is the first significant progress on the problem (Section 6.3, we comment on the connection between the two problems). Our algorithm uses two main ingredients: a quantifier-elimination result, and an algorithm for the quantifier-free case that uses a data structure that involves trees with edge weights taken from a polynomial ring $\mathbb{Z}[X]$. The equivalence algorithm is the main technical contribution of this paper.

[1]The idea to consider regular functions for multiset types was suggested to us by Marcelo Fiore.

(3) Finally, in Section 7 we illustrate the strength of the equivalence result from the previous item by extending it to graphs of bounded tree-depth.

Context. One could place the present paper within the wider context of Hilbert's Entscheidungsproblem [Börger et al. 2001], which is to decide if a first-order formula is true. This problem can be seen as deciding equality on (the semantics) of formulas, i.e whether a formula φ is equal to another formula, such as "true" or "false". The problem is famously undecidable, as shown by Church and Turing. One of the most successful methods of recovering decidability has been to use automata methods (tree automata, MSO logic, tree decompositions, etc). The scope of such methods is now well understood; they will work if and only if one considers structures that are similar to trees, see [Seese 1991, Theorem 8] and [Courcelle and Oum 2007, Theorem 5.6] for formalisations of this statement.

In this paper, we address a functional version of the problem: instead of deciding equality on logically defined languages (functions with Boolean outputs), we want to decide it on logically defined functions (which input and output structures). In the functional version, logical formulas are replaced by logical interpretations, and one asks if for every input, the outputs produced by the two interpretations are isomorphic. This makes the problem nontrivial for output structures with nontrivial isomorphisms, such as graphs, or even unordered trees. This difficulty is not present in the language version, because isomorphism creates difficulties only for the output structures. Making progress on this problem seems to be very hard, even for tame structures such as unordered trees of bounded height or graphs of bounded tree-depth, as considered in this paper.

To our best knowledge, the results of this paper about interpretations on bounded tree-depth gives are the first known decidability results about equivalence for graph-to-graph interpretations. Previous results about equivalence involved trees, with the state of the art being the results on tree-to-tree interpretations from [Seidl et al. 2018, Corollary 8.2] and [Boiret et al. 2018, Section 3]. These papers deal with trees of unbounded depth, but the interpretations are restricted to have linear output size. The techniques are based on Hilbert's Basis Theorem and seem to fail for interpretations of non-linear output size, and also have a certain fragility regarding the type of structures that can be modelled (for example, the results work for linear interpretations on trees with a distinguished root, but the case of trees without distinguished roots remains open [Bojańczyk and Schmude 2020, p. 7]). The only known decidability results for equivalence of non-linear interpretations is for functions that output numbers [Douéneau-Tabot 2021, Corollary 19]; the techniques used there are based on weighted automata and seem to apply only to outputs that are numbers.

Nested multisets have been studied in database theory, in the context of nested relational algebra [Buneman et al. 1995; Cheney et al. 2014; Ricciotti and Cheney 2019]. Calculi of typed functionals are used there to define transformations between nested collections, similarly to what we aim to do here. It appears that our approach is more restricted than most of the calculi considered there, with the advantage of decidable equivalence. Potential applications of our results in nested relational calculi remain to be investigated.

Due to lack of space, some of the proofs have been elided. They can be found in appendices to the full version of this paper [Bojańczyk and Klin 2023].

2 MULTISET TYPES

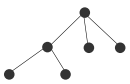
One way of describing trees of bounded height is to view them as values of certain algebraic datatypes. The essential ingredient will be the multiset type constructor, since an unordered tree can be seen as the multiset of its child subtrees. We begin the paper with this approach. Apart from the multiset constructor, we also allow products and coproducts, which will be useful for typing various helper functions.

Definition 2.1 (Multiset types). A multiset type is any type that can be generated using the following type constructors:

$$\underbrace{1}_{\substack{\text{a set with}\\ \text{one element}}} \qquad \underbrace{\Sigma + \Gamma}_{\substack{\text{disjoint union}\\ \text{of sets } \Sigma \text{ and } \Gamma}} \qquad \underbrace{\Sigma \times \Gamma}_{\substack{\text{product of}\\ \text{sets } \Sigma \text{ and } \Gamma}} \qquad \underbrace{\mathsf{M}\Sigma}_{\substack{\text{multisets}\\ \text{over set } \Sigma}}.$$

Strictly speaking, this definition introduces two notions: a syntax of multiset types, and an obvious semantics of each type as a certain set of values. There is no need to be pedantic about this distinction for the moment, but in Section 3 we will introduce a more elaborate semantics of multiset types as sets of relational structures over certain vocabularies.

Example 1. $\mathsf{M}^k 1$ is the type of unordered trees of height at most k. (The *height* of a tree is defined to be the maximal number of edges on a root-to-leaf path.) The idea is that a tree is represented as the multiset of the representations of its child subtrees. For example, consider $k = 2$ and the following tree



Assuming that we use double set brackets to represent multisets, the above tree is represented as the following multiset of multisets:

$$\{\!\{\{\!\{*,*\}\!\},\emptyset,\emptyset\}\!\} \in \mathsf{M}^2 1,$$

where $*$ denotes the unique element of the set 1. □

Example 2. We can extend the trees from the previous example by adding labels from some type Σ. There are two variants of such trees of height at most k:

- $\mathsf{T}^{\text{edge}}_k\Sigma$: edge-labeled trees, and
- $\mathsf{T}^{\text{node}}_k\Sigma$: node-labeled tress.

These constructors can be simulated using multiset types, by induction on k:

$$\mathsf{T}^{\text{edge}}_0\Sigma = 1 \qquad \mathsf{T}^{\text{node}}_0\Sigma = \Sigma$$

$$\mathsf{T}^{\text{edge}}_{k+1}\Sigma = \mathsf{M}\left(\Sigma \times \mathsf{T}^{\text{edge}}_k\Sigma\right) \qquad \mathsf{T}^{\text{node}}_{k+1}\Sigma = \Sigma \times \left(\mathsf{M}\mathsf{T}^{\text{node}}_k\Sigma\right)$$

□

We want to study a class of functions between multiset types which we call the *polyregular functions*, meant to be an analogue of the string-to-string polyregular functions from [Bojańczyk 2018]. There will be two equivalent definitions of this class, one using FO interpretations (Section 3), and one using combinators (Section 4).

3 LOGIC

We describe a logical approach to polyregular functions on multiset types. The idea is to view each multiset type as a class of structures, in the sense of model theory, and to use FO interpretations to transform the structures. FO interpretations are the usual kind of functions from structures to structures, where the universe of the output structure is defined using tuples of elements in the input structure, and the relations of the output structure are defined using FO formulas. Precise definitions are given below.

FO *interpretations.* We assume that the reader is familiar with the basic notions of first-order (FO) logic. We use the following terminology. A *vocabulary* is a finite set of relation names, each one with an associated arity in $\{0, 1, 2, \ldots\}$. A *structure* over such a vocabulary consists of a set, called its *universe*, and for each relation name in the vocabulary a corresponding relation – of same arity – over the universe. We assume that the universe in a structure is nonempty.

When we talk about a *class of structures*, we mean any class that contains structures, over the same vocabulary, which is closed under isomorphism. To transform structures of one class into structures of another class, we use FO interpretations as defined below. This notion is commonly used in model theory (see e.g. [Hodges 1993, Chap. 5.3]). For a simple example, see Ex. 4 below.

Definition 3.1. Let Σ and Γ be classes of structures. A function $f : \Sigma \to \Gamma$ is called an FO interpretation if it can be described as follows: there is a finite set I, called the *components* of the interpretation, with each component $i \in I$ associated to a *dimension* $k_i \in \{0, 1, \ldots\}$, and for every input structure $A \in \Sigma$ the output structure $f(A)$ is defined by:

- **Universe.** For every component i there is an FO formula φ_i over the vocabulary of Σ with k_i free variables, such that the universe of $f(A)$ is
$$\coprod_{i \in I} \{\bar{a} \in A^{k_i} \mid A \models \varphi_i(\bar{a})\}.$$
Elements of the above disjoint union will be written as pairs $(i, \bar{a})$.
- **Relations.** For every relation name R, say of arity n, in the vocabulary of Γ and every tuple of components $i_1, \ldots, i_n \in I$, there is an FO formula φ over the vocabulary of Σ with $k_{i_1} + \cdots + k_{i_n}$ free variables, such that
$$f(A) \models R((i_i, \bar{a}_1), \ldots, (i_n, \bar{a}_n)) \qquad \Leftrightarrow \qquad A \models \varphi(\bar{a}_1 \cdots \bar{a}_n)$$
for all $(i_1, \bar{a}_1), \ldots, (i_n, \bar{a}_n)$ in the universe of $f(A)$.

Remark 3.2. This definition is what is sometimes called an *injective* FO interpretation. In model theory one typically uses a more general notion of interpretation, where the universe of the output structure is a quotient
$$\coprod_{i \in A} \{\bar{a} \in A^{k_i} \mid A \models \varphi_i(\bar{a})\}_{/\sim_i},$$
where each $\sim_i$ is some equivalence relation that is defined by a formula with $2k_i$ free variables. The formulas defining the relations of the output structure are required to be invariant under these equivalences. This is a more powerful model, which we call *non-injective* interpretations, and which will not be discussed in this paper.

Multiset types as classes of structures. A multiset type can be viewed as a class of structures, as explained in the following inductive definition.

Definition 3.3. The type constructors are extended to classes of structures as follows:

1: This is the class of structures over the empty vocabulary, that contains only one structure: a universe with one element.

$\Sigma \times \Gamma$: If Σ and Γ are classes of structures, then $\Sigma \times \Gamma$ is the class over vocabulary
$$voc(\Sigma \times \Gamma) = voc(\Sigma) + voc(\Gamma) + \underbrace{R(x)}_{\text{one extra unary relation name}}$$
(here + means disjoint union of vocabularies) that is defined as follows. A structure in $\Sigma \times \Gamma$ is obtained by taking a structure $A \in \Sigma$ and a structure $B \in \Gamma$, and returning the following

pair structure (A, B): extend both A and B to the vocabulary $\mathrm{voc}(\Sigma) + \mathrm{voc}(\Gamma)$ by interpreting new relation names as empty sets, then take their disjoint union, and finally interpret R as the set of elements that come from A.

$\Sigma + \Gamma$: If Σ and Γ are classes of structures, then $\Sigma + \Gamma$ is the class over vocabulary

$$voc(\Sigma + \Gamma) = voc(\Sigma) + voc(\Gamma) + \underbrace{R}_{\text{one extra 0-ary relation name}}$$

which contains structures that are obtained by either:

(1) taking a structure in Σ, extending it to $\mathrm{voc}(\Sigma) + \mathrm{voc}(\Gamma)$ and interpreting R as true; or
(2) taking a structure in Γ, extending it to $\mathrm{voc}(\Sigma) + \mathrm{voc}(\Gamma)$ and interpreting R as false.

$\mathsf{M}\Sigma$: If Σ is a class of structures, then $\mathrm{voc}(\mathsf{M}\Sigma)$ arises from $\mathrm{voc}(\Sigma)$ by replacing every 0-ary relation by a unary relation, and then adding an extra binary relation $\sim$. A structure in $\mathsf{M}\Sigma$ is obtained by:

(1) taking the disjoint union of some structures $A_1, \ldots, A_n \in \Sigma$,
(2) replacing every 0-ary predicate R from $\mathrm{voc}(\Sigma)$ by a unary relation that holds for all elements of those A_i for which R is true,
(3) interpreting $\sim$ as the equivalence relation whose equivalence classes are the universes of the structures $A_1, \ldots, A_n$,
(4) and adding an extra element (called the *root*) that does not participate in any relations[2].

In this way, every multiset type can be seen as a class of structures. Therefore one can use FO sentences to define subsets of multisets types, and FO interpretations to define functions between multiset types.

Remark 3.4. FO interpretations are closed under composition, and the identity is a FO interpretation. Therefore one can consider the following category: the objects are classes of structures, and the morphisms are FO interpretations modulo equivalence. (This means that two interpretations describe the same morphism if they are equivalent in the sense that for every input structure, the two output structures produced by the two interpretations are isomorphic.) It is not difficult to prove that 1, $\Sigma + \Gamma$ and $\Sigma \times \Gamma$ in Definition 3.3 define respectively the terminal object, coproduct, and product in this category, and that M is a monad. We do not elaborate this categorical perspective in this paper, but it will inform our choice of prime functions and combinators in Definition 4.1.

Example 3. An unordered tree of height at most k can be seen as a relational structure over a vocabulary of k binary relations $\sim_1, \ldots, \sim_k$, with $\sim_i$ interpreted as relating those nodes that have a common ancestor at depth at least i. It is easy to construct mutually inverse interpretations between this representation of trees and the more usual one based on a parent-child relation. By Definition 3.3, trees represented in this way are exactly structures of type $\mathsf{M}^k 1$. □

By Lemma 4.3 below, all multiset types can be viewed as a special case of trees of bounded height. For such structures first-order logic has the same expressive power as monadic second-order logic, see [Elberfeld et al. 2016, Theorem 1.1]. Therefore, MSO and FO logics will define the same kind of interpretations between multiset types. We will therefore simply speak about *interpretations* from now on, without specifying that they are FO interpretations.

Example 4. Assume that we model the natural numbers as multisets

$$\mathbb{N} \stackrel{\text{def}}{=} \mathsf{M}1.$$

[2]The root serves two purposes: (a) it guarantees that the universe is nonempty even if $n = 0$; and (b) it can be uniquely identified by a first-order formula whenever choosing a unique element is needed. This will come useful in Example 6.

Under this representation, which functions $f : \mathbb{N} \to \mathbb{N}$ can be obtained as interpretations? One example is all polynomials with non-negative coefficients. For example, the function

$$f(n) = 3n^4 + n^2 + 7$$

is defined by an interpretation with 3 components of dimension 4, 1 component of dimension 2, and 7 components of dimension 0. In each of these components, the universe formula is "true".

Another example is the function that counts the number of non-repeating k-tuples in the input, the output of this function is

$$n^{(k)} \stackrel{\text{def}}{=} n \cdot (n-1) \cdot (n-2) \cdots (n-k+1).$$

As a polynomial, this function has some negative coefficients. The corresponding interpretation has 1 component of dimension k, with a universe formula that selects non-repeating tuples. As it turns out, non-repeating tuples are essentially the only thing that can be done.

Proposition 3.5. *The following are equivalent for every function $f : \mathbb{N} \to \mathbb{N}$:*

(1) *f is an interpretation, assuming the representation $\mathbb{N} = \mathsf{M}1$;*
(2) *there are non-negative co-efficients $a_0, \ldots, a_k \in \{0, 1, \ldots\}$ such that*

$$f(n) = a_0 \cdot n^{(0)} + a_1 \cdot n^{(1)} + \cdots + a_k \cdot n^{(k)},$$

holds for all sufficiently large n.

Proof. The implication $2 \Rightarrow 1$ is easy to show. For the converse, it is enough to prove the claim for interpretations with one component; since having several components corresponds to taking a sum of functions with non-negative co-efficients. Suppose then that f is defined by an interpretation with one component of dimension k. The corresponding function inputs a number n, and returns the number of k-tuples that satisfy some first-order formula $\varphi(x_1, \ldots, x_k)$ in the structure that has n elements and no nontrivial relations. Over such a structure, quantifiers are not useful, at least as long as n exceeds some threshold. Counting tuples that satisfy a quantifier-free formula gives a function as in item (2). □

As a corollary, e.g. the decrement function $f(n) = \max(n-1, 0)$ is not an interpretation. □

4 DERIVABLE FUNCTIONS

In this section, we give an alternative definition of polyregular functions on multiset types, which uses combinators. Then we prove that this definition is equivalent to the interpretations from Section 3.

The idea is to start with certain prime functions, such as multiset union, and close them under certain combinators, such as composition of functions. A brutal approach would be to start with all functions that are interpretations; in this case, the combinators would not be needed. However, in the presence of combinators, only a small set of prime functions is needed. Two of these functions are explained in the following examples.

Example 5. For a multiset type Σ, consider the *de-singleton function* of type

$$\mathsf{M}\Sigma \to 1 + \Sigma$$

that maps singleton multisets to their unique elements, and otherwise returns the unique element of 1. (Like almost all prime functions, this is not a single function but a family of functions indexed by multiset types.) We claim that this is a first-order interpretation. This has a component of dimension zero, used to produce the error value in the case of non-singleton inputs. The universe formula for this component says that the input multiset is not a singleton, i.e. that it either consist

solely of the root or it contains some non-root elements which are not related by the equivalence relation $\sim$ from the definition of $\mathsf{M}\Sigma$:

$$\forall x.\neg(x \sim x) \quad \vee \quad \exists xy.(x \sim x \wedge y \sim y \wedge \neg(x \sim y))$$

For singleton input multisets, the input structure should simply be copied to the output, removing the root. This is done by using an additional component of dimension one, whose universe formula says that the argument is not the root of the multiset. □

Example 6. Let Σ be a multiset type. One of our prime functions, and the only one whose growth is more than linear, is *choices* of the type

$$\mathsf{M}\Sigma \to \mathsf{M}(\Sigma \times \mathsf{M}\Sigma).$$

This function, given a multiset

$$\{\!\{A_1, \ldots, A_n\}\!\} \qquad \text{where } A_i \in \Sigma,$$

outputs the following multiset with the same number of elements:

$$\{\!\{(A_1, B_1), \ldots, (A_n, B_n)\}\!\} \qquad \text{where } B_i = \{\!\{A_1, \ldots, A_n\}\!\} - \{\!\{A_i\}\!\}.$$

The idea is that the output contains all possible multisets that can be obtained from the input multiset by choosing some distinguished element.

Let us prove that this function is a first-order interpretation. To represent an element of A_i in the output structure, we simply use the same element in the input structure. These elements are represented using a component of dimension one. To represent an element of B_i in the output structure, we use a pair (a, b), where a is the *root* of A_i (see below) and b is an element of some A_j in the input structure with $j \neq i$. Formally, the universe formula for the corresponding component of dimension two is:

$$root_\Sigma(a) \wedge \neg(a \sim b),$$

where $\sim$ is the equivalence relation from the definition of $\mathsf{M}\Sigma$, and $root_\Sigma$ is a formula that is satisfied for a unique element of every structure in Σ. Such a formula is defined by induction on Σ, using the root in the induction step for multisets. □

The above examples explain the only two non-obvious prime functions. The remaining prime functions and all the combinators are straightforward enough that we simply give their types and names, and we assume that the reader can guess their definitions.

Definition 4.1. The *derivable* functions are the least class of functions that:

- **Prime functions:** contains the following functions for all types $\Sigma, \Sigma_1, \Sigma_2$:

union:	$\mathsf{M}\mathsf{M}\Sigma \to \mathsf{M}\Sigma$	(multiset union)
add:	$\Sigma \times \mathsf{M}\Sigma \to \mathsf{M}\Sigma$	(add one element)
choices:	$\mathsf{M}\Sigma \to \mathsf{M}(\Sigma \times \mathsf{M}\Sigma)$	(see Ex. 6)
de-singleton:	$\mathsf{M}\Sigma \to 1 + \Sigma$	(see Ex. 5)
empty:	$1 \to \mathsf{M}1$	(constant empty multiset)
id:	$\Sigma \to \Sigma$	(identity)
π_i:	$\Sigma_1 \times \Sigma_2 \to \Sigma_i$	(projection for $i = 1, 2$)
ι_i:	$\Sigma_i \to \Sigma_1 + \Sigma_2$	(coprojection for $i = 1, 2$)
dist:	$\Sigma \times (\Sigma_1 + \Sigma_2) \to \Sigma \times \Sigma_1 + \Sigma \times \Sigma_2$	(distribute $\times$ over $+$)

- **Combinators:** is closed under applying the following combinators:

$$\frac{f_1 : \Sigma \to \Gamma_1 \quad f_2 : \Sigma \to \Gamma_2}{\langle f_1, f_2 \rangle : \Sigma \to \Gamma_1 \times \Gamma_2} \text{ pairing} \qquad \frac{f_1 : \Sigma_1 \to \Gamma \quad f_2 : \Sigma_2 \to \Gamma}{[f_1, f_2] : \Sigma_1 + \Sigma_2 \to \Gamma} \text{ co-pairing}$$

$$\frac{f : \Sigma \to \Gamma}{\mathsf{M}f : \mathsf{M}\Sigma \to \mathsf{M}\Gamma} \text{ mapping} \qquad \frac{f : \Sigma_1 \to \Sigma_2 \quad g : \Sigma_2 \to \Sigma_3}{f ; g : \Sigma_1 \to \Sigma_3} \text{ composition}$$

Remark 4.2. It is easy to prove that all prime functions are interpretations (see Theorem 4.4). In fact, recalling categorical considerations from Remark 3.4, most prime functions are natural transformations between the corresponding functors on the category of FO interpretations. A notable exception is *choices*, whose FO definition in Example 6 relies on *root* formulas that are defined only for multiset types and not for arbitrary structures. The need for *root*s would disappear if we considered non-injective FO interpretations as in Remark 3.2; then *choices* would become a natural transformation. It therefore seems natural to leave the full categorical framework to future work, until we extend our results to non-injective interpretations. For the present, we can use the flexibility afforded by our elementary framework to derive some other non-natural functions by induction on the structure of multiset types.

Example 7. Although the list of prime functions has the identity for every multiset type Σ, it is only really needed in type $1 \to 1$. For the remaining types, it can be derived using combinators. Also, for every type Σ we can derive the unique function $!_\Sigma$ of type $\Sigma \to 1$. The proof is by induction on Σ, in the induction step for $\Sigma = \mathsf{M}\Gamma$ one uses the de-singleton operation.

We can also derive functions of type $1 \to \Sigma$ by induction on Σ. For coproducts, this is done with coprojections; for products, with pairing; for multiset types, with the mapping combinator and the constant empty multiset function. □

Example 8. Define the Boolean type to be

$$\text{Bool} \stackrel{\text{def}}{=} 1 + 1.$$

One can easily derive all Boolean operations, e.g. conjunction $\text{Bool}^2 \to \text{Bool}$. In fact, for every finite types Σ and Γ, i.e. types that are built without the multiset constructor M, one can derive every function of type $\Sigma \to \Gamma$. □

Example 9. The emptiness test

$$\mathit{empty} : \mathsf{M}1 \to \text{Bool},$$

that returns "true" for the empty multiset and "false" for nonempty ones, is derived as the composition of the following operations:

$$\mathsf{M}1 \xrightarrow{\langle !_{\mathsf{M}1}, \mathit{id} \rangle} 1 \times \mathsf{M}1 \xrightarrow{\mathit{add}} \mathsf{M}1 \xrightarrow{\mathit{de\text{-}singleton}} 1 + 1 = \text{Bool}.$$

The idea is that the empty multiset is the only one to which we can add and element to obtain a singleton. □

Example 10. We now derive the function

$$\vee : \mathsf{M}(\text{Bool}) \to \text{Bool}$$

that implements disjunction of unbounded arity: the function checks if the input multiset contains at least one "true" value. This function is derived as follows. To the input, which is a multiset of Booleans, we apply $\mathsf{M}f$, where $f : \text{Bool} \to \mathsf{M}1$ is the function that maps "false" to the empty

multiset and "true" to the singleton multiset. After applying multiset union to the result, we test the resulting value for emptiness.

Using the same idea, for any multiset types Σ and Γ we can derive a function

$$\mathit{drop} : \mathsf{M}(\Sigma + \Gamma) \to \mathsf{M}\Sigma$$

that erases all elements of Γ from the given multiset. □

Example 11. For multiset types Σ and Γ, consider the function *strength* of type

$$\mathsf{M}\Sigma \times \Gamma \to \mathsf{M}(\Sigma \times \Gamma)$$

which, given an argument $B \in \Gamma$ and a multiset

$$\{\!\{A_1, \ldots, A_n\}\!\} \qquad \text{where } A_i \in \Sigma,$$

returns the multiset of pairs:

$$\{\!\{(A_1, B), \ldots, (A_n, B)\}\!\}$$

with the same number of elements.

Let us prove that strength is derivable. Given a structure in $\mathsf{M}\Sigma \times \Gamma$, perform the following steps:

(1) use suitable coprojections to get to $\mathsf{M}(\Sigma + \Gamma) \times (\Sigma + \Gamma)$;
(2) use *add* to get to $\mathsf{M}(\Sigma + \Gamma)$;
(3) use *choices* to get to $\mathsf{M}((\Sigma + \Gamma) \times \mathsf{M}(\Sigma + \Gamma))$;
(4) use *dist* to get to $\mathsf{M}(\Sigma \times \mathsf{M}(\Sigma + \Gamma) + \Gamma \times \mathsf{M}(\Sigma + \Gamma))$;
(5) use *drop* from Example 10 twice, to get to $\mathsf{M}(\Sigma \times \mathsf{M}\Gamma)$;
(6) de-singleton $\mathsf{M}\Gamma$, distribute and use *drop* again to get to $\mathsf{M}(\Sigma \times \Gamma)$.

□

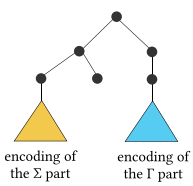
Trees of bounded height. In Examples 1 and 3 we showed how unlabelled trees of height at most k can be modeled as structures of type $\mathsf{M}^k 1$. The following lemma shows that such types are general, i.e. all multiset types can be encoded in them. We use the following notion of encoding: we say that a type Σ *encodes* in a type Γ if there are derivable functions

$$\Sigma \underset{\textit{decode}}{\overset{\textit{encode}}{\rightleftarrows}} \Gamma$$

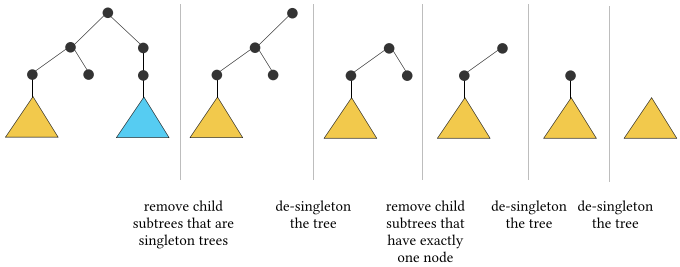
such that *encode;decode* is the identity on Σ.

Lemma 4.3. *Every multiset type encodes into* $\mathsf{M}^k 1$ *for some* $k \in \{0, 1, \ldots\}$.

Proof. Induction on the structure of the type. For the type 1, there is nothing to do. For a type $\mathsf{M}\Sigma$, we simply apply the mapping combinator to the encoding for Σ. We are left with products and coproducts. We only describe the construction for products $\Sigma \times \Gamma$; coproducts can be treated in a similar way. By induction assumption, each of the types Σ and Γ encodes into trees of some height. We can assume that the heights are the same, because $\mathsf{M}^k 1$ encodes into $\mathsf{M}^{k+1} 1$. We pair the encodings as follows: a pair in $\Sigma \times \Gamma$ is sent to the following tree



This tree lives in $\mathsf{M}^{k+3}1$, where k is the height of trees needed to encode Σ and Γ. It remains to describe the decoding. We explain how to extract the Σ component. In the argument below, a *singleton tree* is a tree where the root has a single child. The projection to the Σ coordinate is performed in five stages, summarised in the following picture:



We omit the straightforward details of the construction. A similar construction is used for extracting the Γ component. □

4.1 Completeness of the derivable functions

Let us prove that on multiset types, first-order interpretations coincide with derivable functions.

Theorem 4.4. *Let $f : \Sigma \to \Gamma$ be a function between multiset types. Then f is a first-order interpretation if and only if it is derivable.*

The easier direction is *soundness*: every derivable function is also an interpretation. This is proved by a straightforward induction on the derivation: all prime functions are easily seen to be first-order interpretations (see Examples 5-6), and first-order interpretations are easily seen to be closed under applying the combinators.

We now move to the harder part of the theorem, which we call *completeness*: every first-order interpretation can be derived. This is proved first for Boolean outputs, and then for general outputs.

4.1.1 Boolean outputs. We begin by deriving interpretations with Boolean outputs, which is the same as first-order sentences. (Indeed, all structures of type Bool have a one-element universe with a single nullary relation, so the only ingredient of the interpretation is the sentence which is the interpretation of that relation.) This is done by inlining the semantics of FO logic into our type system, with the *choices* function used to simulate a single quantifier.

In the proof it will be convenient to think of structures with distinguished elements as being a multiset type. This is done using the following type constructor, where $k \in \{0, 1, \ldots\}$:

$$\mathsf{V}_k\Sigma \stackrel{\text{def}}{=} \{(A, \bar{a}) \mid A \in \Sigma \text{ and } \bar{a} \text{ is a tuple of } k \text{ distinguished elements}\}.$$

We do not assume that the distinguished elements are pairwise distinct. This type constructor can be encoded using the existing type constructors:

$$\mathsf{V}_0\Sigma \equiv \Sigma \qquad \mathsf{V}_1 1 \equiv 1 \qquad \mathsf{V}_1(\Sigma \times \Gamma) \equiv \mathsf{V}_1\Sigma \times \Gamma + \Sigma \times \mathsf{V}_1\Gamma$$
$$\mathsf{V}_{k+1}\Sigma \equiv V_1(V_k\Sigma) \qquad \mathsf{V}_1(\Sigma + \Gamma) \equiv \mathsf{V}_1\Sigma + \mathsf{V}_1\Gamma \qquad \mathsf{V}_1\mathsf{M}\Sigma \equiv (1 + \mathsf{V}_1\Sigma) \times \mathsf{M}\Sigma$$

(the 1+ in the multiset case is due to the fact that the root element might be distinguished). As a result, for any multiset type Σ, we can view $\mathsf{V}_k\Sigma$ as a multiset type. The semantics of a first-order formula φ over the vocabulary of Σ that has k free variables is an interpretation of type

$$[\![\varphi]\!] : \mathsf{V}_k\Sigma \to \text{Bool}.$$

The following lemma shows that such functions are derivable.

Lemma 4.5. *Let Σ be a multiset type. For every first-order formula φ over the vocabulary of Σ with k free variables, the function $[\![\varphi]\!]$ is derivable.*

Proof. Induction on the size of φ. The induction base, when φ is an atomic formula, is by straightforward induction on Σ. For example, the equality predicate $x = y$ is modeled as a function

$$eq_\Sigma : \mathsf{V}_2\Sigma \to \mathrm{Bool},$$

easily derived by induction on Σ.

For the induction step of negation $\neg\varphi$, we simply compose the function of φ with the negation operation of type $\mathrm{Bool} \to \mathrm{Bool}$. The same works for disjunction and conjunction, except that we use pairing.

The last remaining induction step is that of a quantified formula of the form $\exists x\varphi$ or $\forall x\varphi$, where φ is a formula with $k+1$ free variables. Consider the *extension function* of type:

$$ext_\Sigma : \mathsf{V}_k\Sigma \to \mathsf{M}\mathsf{V}_{k+1}\Sigma$$

that extends the input valuation in all possible ways by adding one element to the tuple. Since by definition $V_{k+1}\Sigma = V_1(V_k\Sigma)$, it is enough to derive this function for $k = 0$. This is done by induction on the type Σ; for $ext_{\mathsf{M}\Sigma}$ we use choices and *strength* from Example 11, and for $ext_{\Sigma\times\Gamma}$ we use *strength* again.

The functions $[\![\exists x\varphi]\!]$ and $[\![\forall x\varphi]\!]$ are derived by first applying the extension function, then applying $[\![\varphi]\!]$ to each element of the resulting multiset, and then applying disjunction/conjunction from Example 10 to the resulting multiset of Booleans. □

4.1.2 General outputs. We now show how to derive first-order interpretations that have outputs which can be any multiset type, not just the Booleans. Thanks to Lemma 4.3, which says that every multiset type encodes into trees, the problem will reduce to functions with tree outputs.

Valuation trees. To deal with tree outputs, we introduce a data structure called valuation trees. For a structure A and a number $k \in \{0, 1, \dots, \}$, the *valuation tree of height k for A* is defined as follows. Nodes of the tree are pairs $(A, \bar{a})$ where $\bar{a}$ is a list of at most k distinguished elements in A; in particular, the structure A is copied in each node in the tree. The ancestor relation is the prefix relation on the tuples of distinguished elements; in particular, the root of the tree uses the empty tuple of distinguished elements. The height of this tree is k, and the leaves are labelled by tuples of k distinguished elements.

For a multiset type Σ, the valuation trees of height k for structures from Σ can be seen as a type:

$$\mathsf{T}^{\mathrm{node}}_k(\mathsf{V}_{\le k}\Sigma), \quad \text{where} \quad \mathsf{V}_{\le k}\Sigma \quad \text{is defined as} \quad \mathsf{V}_0\Sigma + \cdots + \mathsf{V}_k\Sigma$$

and $\mathsf{T}^{\mathrm{node}}_k$ is the constructor of node-labelled trees from Example 2. Not all elements of this type are valuation trees, since a valuation tree requires that the label of a parent node is obtained from the label of any child node by removing the last distinguished element. For every k and multiset type Σ, there is a derivable function:

$$valtree^k_\Sigma : \Sigma \to \mathsf{T}^{\mathrm{node}}_k(\mathsf{V}_{\le k}\Sigma)$$

that maps a structure to its valuation tree. This is by induction on k, with $valtree^{k+1}_\Sigma$ derived by:

$$\Sigma \xrightarrow{\langle id, ext_\Sigma\rangle} \Sigma \times \mathsf{M}\mathsf{V}_1\Sigma \xrightarrow{id\times\mathsf{M}(valtree^k_{\mathsf{V}_1\Sigma})} \Sigma \times \mathsf{M}\mathsf{T}^{\mathrm{node}}_k\mathsf{V}_{\le k}\mathsf{V}_1\Sigma \longrightarrow \mathsf{T}^{\mathrm{node}}_{k+1}\mathsf{V}_{\le k+1}\Sigma$$

with the function on the right derived from obvious coprojections of $\Sigma = \mathsf{V}_0\Sigma$ and $\mathsf{V}_{\le k}\mathsf{V}_1\Sigma$ into $\mathsf{V}_{\le k+1}\Sigma$. The extension function *ext* was derived in the proof of Lemma 4.5.

Completeness proof. Using valuation trees, we finish the completeness proof for Theorem 4.4. Suppose that we want to derive a first-order interpretation $f : \Sigma \to \Gamma$. Apply Lemma 4.3 to the output type, yielding derivable functions

$$\Gamma \underset{decode}{\overset{encode}{\rightleftarrows}} \mathsf{M}^n 1.$$

Since the decoding function is derivable, it is enough to show that the composition

$$\Sigma \xrightarrow{f;encode} \mathsf{M}^n 1 = \mathsf{T}^{\mathrm{node}}_n 1$$

is derivable. This function has tree outputs of bounded height, and it is a first-order interpretation.

For every such interpretation, there is an equivalent interpretation such that:

(a) for every dimension i there is at most one component of this dimension; and
(b) the ancestor ordering on nodes of the output tree is the prefix relation on tuples.

To ensure (b) only, build an interpretation where nodes of the output structure are sequences (paths in the tree) of output nodes in the original interpretation, with new components arising from sequences (of length up to n) of the original components. The original parent-child relation now becomes part of the universe condition, and the new ancestor relation is just the prefix order. To furthermore ensure (a) in this construction, extend the dimension of every component (arising from a sequence of the original components) by adding a certain number of dummy coordinates at the end (and also, necessarily, in the middle, to ensure that the parent-child relation is still a prefix relation).

Assuming that $f; encode$ has properties (a) and (b), and the maximal dimension used in the interpretation is k (typically k will be larger than n), one can decompose $f; encode$ into four steps:

(1) Compute the valuation tree of height k, yielding a result in type $\mathsf{T}^{\mathrm{node}}_k \mathsf{V}_{\leq k} \Sigma$.
(2) Consider the universe formula of the first order-interpretation defining $f; encode$, viewed as a function of type $\mathsf{V}_{\leq k}\Sigma \to \mathrm{Bool}$. Apply this function to the label of each node in the result of the previous step, yielding a result in type $\mathsf{T}^{\mathrm{node}}_k \mathrm{Bool}$.
(3) In the tree computed so far, keep only nodes with label "true", while preserving the ancestor relation, yielding a tree in $\mathsf{T}^{\mathrm{node}}_k 1$.
(4) By the semantic properties of the first-order interpretation, we know that the output from the previous step will have height n. However, we need to explicitly cast it into the output type $\mathsf{T}^{\mathrm{node}}_n 1$. We therefore apply a derivable function that keeps only nodes at distance at most n from the root, yielding a tree in the correct output type $\mathsf{T}^{\mathrm{node}}_n 1$.

All these four steps are derivable. Indeed, as we have already remarked, the valuation tree can be computed using a derivable function. The second step is derivable by Lemma 4.5. Deriving the last two steps is straightforward.

5 DECIDING EQUIVALENCE FOR QUANTIFIER-FREE INTERPRETATIONS ON TREES OF BOUNDED HEIGHT

In the previous section we presented a list of prime functions and combinators that together generate all interpretations that input and output trees of bounded height. It is clear that these functions and combinators satisfy various equalities: pairing of the two projections from a product type is equal to the identity, the mapping combinator preserves function composition, and so on. One wonders whether there is an equational axiomatisation of the equivalence of interpretations presented in this way. Ideally, that equational theory would be decidable. We do not attempt to provide such a theory in this paper. However, we will prove that equivalence of interpretations is

indeed decidable. We shall work directly with the logical presentation of interpretations, without a direct reference to the prime functions and combinators.

The problem is to decide, given two FO interpretations on multiset types, if they produce isomorphic outputs for every input. We first solve a special case of this problem, where the input and output types represent trees of bounded height, and – more importantly – the functions are quantifier-free interpretations. In Section 6 we will reduce the general problem to this special case.

We begin by specifying the exact variant of trees that is used, and their representations as logical structures. We need to be quite specific here, since for quantifier-free interpretations the exact choice of vocabulary plays a role. In Section 6 these details will become unimportant, as quantified formulas allow easy translations between various representations of trees.

For the output trees, we consider unlabelled trees with a height bound n, where a tree is represented as a set of nodes with the parent-child relation. We will denote the class of such trees by $\mathsf{M}^n 1$. This is a slight abuse of notation: in Section 3 this indeed denotes the same class of trees, but under a different relational representation.

For the input, we use trees with edge labels from a finite set Σ and with a height bound k. This corresponds to the type $\mathsf{T}_k^{\text{edge}}\Sigma$ described in Example 2, but again, abusing the notation, we use a representation different from the one arising from Definition 3.3. In this *edge represenation*:

- the universe of the tree is the set of its edges;
- for every label a, there is a unary relation that selects edges with label a;
- there is a parent function, which maps an edge to its parent edge. For edges without parents, i.e. edges that originate in the root of the tree, the parent function loops.

A useful property of this representation is that induced substructures have an intuitive meaning: they correspond to "pruned" trees that arise by removing some subtrees. The empty structure also has a meaning: it represents a root-only tree. This will be convenient when we apply generic results from Section 5.1 to labelled trees.

We will consider functions of type

$$\mathsf{T}_k^{\text{edge}}\Sigma \to \mathsf{M}^n 1,$$

for a finite set Σ of edge labels. The functions will be quantifier-free interpretations, assuming tree representations as above. This is the special case of interpretations as in Definition 3.1, in which all formulas are quantifier-free. (Note that Definition 3.1 makes sense in the presence of functions in the input vocabulary.)

This section is devoted to proving the following theorem.

Theorem 5.1. *The following problem is decidable:*

Instance. *Two quantifier-free interpretations* $f, g : \mathsf{T}_k^{edge}\Sigma \to \mathsf{M}^n 1$.
Question. *Are the functions equivalent in the following sense: for every input, the two outputs are isomorphic as trees?*

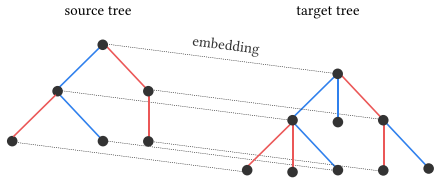
We will even show that the problem is in polynomial time, assuming a suitable representation of quantifier-free interpretations, which we shall call patterns.

5.1 Patterns

In our solution to the equivalence problem, we use a more combinatorial representation of quantifier-free interpretations, which is defined by counting embeddings.

Recall that an *embedding* between structures over the same vocabulary is an injective map from the universe of the source structure into the universe of the target structure, that preserves and reflects all relations. We will use embeddings mainly for trees modelled using the functional edge

representation as described above. Under that representation, an embedding of one tree into another is an injective map from edges in the source tree to edges in the target tree, that preserves the labels and the parent function. Since the parent function loops for edges without parents, such edges must be mapped to edges without parents. In other words, the root node must be mapped to the root node. Here is a picture:

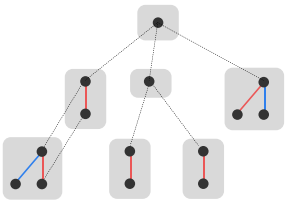


The following definition is the key notion of this section. We will apply it to labeled trees under the edge representation, but the definition makes sense for any class of structures that is closed under induced substructures. Here an induced substructure is obtained by restricting the universe to some (possibly empty) subset that is closed under the functions in the structure.

Definition 5.2 (Pattern). Let $\mathscr{C}$ be a class of structures that is closed under induced substructures. A *pattern for* $\mathscr{C}$ is a finite tree Φ where every node x is labelled by a structure $\Phi(x)$ in $\mathscr{C}$, such that:

- the root node is labelled by the empty structure, and
- if a node x is the parent of a node y, then $\Phi(x)$ is an induced substructure of $\Phi(y)$.

Example 12. Here is a picture of a pattern, where the class $\mathscr{C}$ is trees of height at most one with edges labelled red or blue:



The dotted lines in the picture represent the inclusions of induced substructures between a node of the pattern and its children. □

Two patterns are isomorphic if there is: (i) a bijection between the underlying trees of the two patterns, that preserves the tree structure; and (ii) a family of isomorphisms, one for each node x in the first pattern, that isomorphically maps the structure labelling x in the first pattern to the structure labelling the corresponding node in the second pattern. The family of isomorphisms in (ii) must be consistent with the inclusion of labels, i.e. the isomorphism for a node x must extend the isomorphism for its parent.

Every pattern Φ determines a function which inputs structures from $\mathscr{C}$ and outputs unlabelled trees, and is defined as follows. For an input structure $A \in \mathscr{C}$, the nodes of the output tree are pairs (x, α) where x is a node of the pattern and

$$\alpha : \Phi(x) \hookrightarrow A$$

is an embedding from the structure that labels node x in the pattern to the input structure A. The children of a node (x, α) are nodes (y, β) such that y is a child of x and the embedding β extends α.

By definition, the height of the output tree is bounded by the height of the pattern. Moreover, the function defined by a pattern Φ is easily seen to be a quantifier-free interpretation. Indeed, one

can take a component for each node x in Φ, with dimension equal to the size of the structure $\Phi(x)$, and the universe formula determined by the structure $\Phi(x)$ itself. The parent-child relation is then defined by a suitable sub-tuple formula. All these formulas are quantifier-free.

The following lemma shows that every function of this kind arises from a pattern.

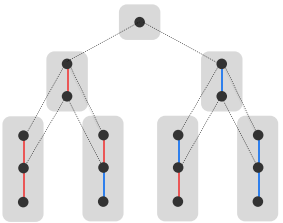
Lemma 5.3. *Let $\mathscr{C}$ be a class of structures that is closed under induced substructures. Then a function from $\mathscr{C}$ to (unlabelled) trees is defined by a pattern if and only if it is a quantifier-free interpretation that produces outputs of bounded height.*

Proof. Consider a quantifier-free interpretation f that, given an input structure in $\mathscr{C}$, produces a tree of height at most n. As we observed in the completeness proof in Section 4.1.2, there exists an equivalent interpretation $\hat{f}$ whose components are sequences of components of f of length up to n; the dimension of each such component is the sum of dimensions of the original components in the sequence. The universe formula for such a component combines the universe formulas for the original components with the parent relation formula of the output tree of f; this is quantifier-free if all the constituent formulas are quantifier-free. Finally, the parent relation is simply the prefix relation of a suitable length; again, this is described by a quantifier-free formula.

Now, $\hat{f}$ is an interpretation of the same type as f, but with the parent relation in the output tree defined as a prefix formula. Every quantifier-free interpretation with this property arises from a pattern. Indeed, the universe formula for each component is (equivalent to) a disjunction of conjunctive formulas that fully describe relations between a finite set of variables. Such a conjunctive formula is essentially a structure from $\mathscr{C}$ together with a valuation of the variables that generates the entire structure. A pattern whose nodes are the conjunctive formulas present in $\hat{f}$, with the parent relation inherited from $\hat{f}$, defines the interpretation $\hat{f}$, which is equivalent to f. □

Importantly, the above proof is constructive: a pattern that defines a quantifier-free interpretation to unlabelled trees can be effectively reconstructed from (the number n and) a logical description of the interpretation.

Example 13. Let $\mathscr{C}$ be the class of trees of height at most two with edges labelled red or blue, and let f be the function that simply returns the input tree, forgetting the colors. This is a quantifier-free interpretation, defined by a single component of dimension one, with the universe formula "true" and an evident parent relation. It is defined by the pattern:



In particular, the root node of this pattern has exactly one embedding into any input tree, and that embedding is the root of the output tree. □

5.2 Patterns with tree inputs

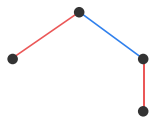
Although patterns make sense for general classes of input structures, we focus on patterns where the input class is trees of bounded height labelled by some finite alphabet, as in Theorem 5.1. By Lemma 5.3, all functions from the decision problem in Theorem 5.1 can be described using patterns, and these patterns can be computed from the original representation in terms of quantifier-free

formulas. Therefore, Theorem 5.1 boils down to deciding equivalence for tree-to-tree functions defined by patterns. We will show that the latter problem is decidable, even in polynomial time, for a very simple reason: different patterns define different tree-to-tree functions.

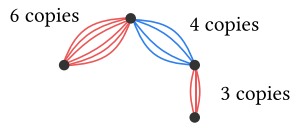
Theorem 5.4. *Consider two patterns Φ_1 and Φ_2 that define tree-to-tree functions. If the patterns are non-isomorphic, then the corresponding tree-to-tree functions are non-equivalent, i.e. for some input tree, the two output trees are non-isomorphic.*

From this we can immediately conclude Theorem 5.1. Indeed, consider two tree-to-tree functions f_1 and f_2, as in Theorem 5.1. By Lemma 5.3, these functions are defined by patterns, say Φ_1 and Φ_2. By Theorem 5.4, the functions are equivalent if and only if the patterns are isomorphic. Isomorphism of patterns can be decided in polynomial time using a straightforward dynamic programming algorithm. Note, however, that there is an exponential overhead in translating a quantifier-free interpretation f_i to a pattern Φ_i, hence the entire algorithm is not polynomial time.

The rest of Section 5 is devoted to proving Theorem 5.4. The general idea is that any two non-isomorphic patterns are distinguished by an input tree where every node, apart from leaves, has a large number of children of every color. To generate a distinguishing input, first a finite labeled tree t of the appropriate depth will be chosen, for example:

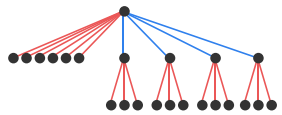


Then every edge in t will be cloned into several copies:



(1)

Finally, the resulting graph will be unfolded into a tree, with the result being:



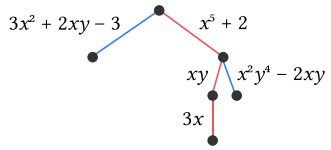
(2)

We will show that, for an appropriate t, cloning every edge into sufficiently many copies will be enough to distinguish any two given patterns. Actually t will not be particularly complicated, it will simply be a large complete tree of the appropriate depth.

To make these ideas precise, we shall consider trees edge-labeled with algebraic expressions that will denote the varying number of copies for every edge.

5.3 Symbolic trees

Our proof will rely on a representation of sets of input trees, which is called *symbolic trees*. Intuitively speaking, a symbolic tree is a tree, where each edge is labelled by a pair consisting of a colour from a finite set and a multivariate polynomial with integer coefficients. In the following picture the colours are red and blue, and the polynomials use variables x and y:

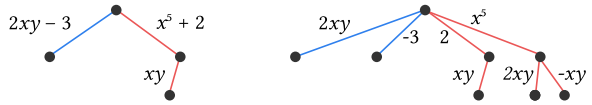


The formal definition of symbolic trees, given below, defines them as the least set closed under taking linear combinations of pairs (colour, smaller symbolic tree). One should think of such a linear combination as representing a tree where the elements of the linear combination are child subtrees, and the coefficients describe the polynomials; in particular, the zero linear combination represents a symbolic tree with a root node and no children.

Definition 5.5. For finite sets Σ and X, the set of *symbolic trees with edge colours in* Σ *and variables* X is the least solution T to the inequality

$$T \supseteq \text{finite formal linear combinations of } \Sigma \times T \text{ with coefficients in } \mathbb{Z}[X].$$

As mentioned before, a symbolic tree can be represented by a tree with edges labelled by pairs (colour, polynomial). This representation is not unique. For example, here are two representations of the same symbolic tree:



The representation does become unique if we require that one cannot have two edges that have the same source, same colour (but possibly different polynomials), and isomorphic subtrees. Symbolic trees are similar to trees modulo bisimulation; the latter would correspond to Definition 5.5 if we used the Boolean semiring instead of the ring of polynomials $\mathbb{Z}[X]$.

A *weighted tree* is the special case of a symbolic tree where the set of variables X is empty. This is a tree with possibly negative integer weights on edges. A weighted tree is called *positive* if all of its weights are positive. Such a tree is the same as an isomorphism class of usual, non-weighted, trees.

The purpose of symbolic trees is to generate weighted trees. This is done by substituting integer values for the variables. If t is a symbolic tree and $\bar{a} \in \mathbb{Z}^X$ is a choice of integer parameters for its variables such that the value of every polynomial in t is positive, then we write $t(\bar{a})$ for the tree that arises by substituting the parameters in each polynomial, and then unfolding the resulting weighted tree as illustrated in (2).

For our proof, we only care about positive weighted trees. One way to generate such trees is to use positive parameters and symbolic trees where all polynomials use only positive coefficients. However, this restriction would be too strong. For example, a typical polynomial that will arise in our proof will be $x^{(k)}$ (see Example 4), which counts non-repeating k-tuples. Although this polynomial has negative coefficients, it is *ultimately positive*: there is some N such that if all parameters are larger than N then the value of the polynomial is positive. This is the same as saying that all monomials of maximal degree have positive coefficients. In the proof below, we will only work with symbolic trees that are ultimately positive, i.e. all polynomials used in them are ultimately positive. For such symbolic trees, all sufficiently large parameters generate positive weighted trees.

Our first observation is that different symbolic trees will generate non-isomorphic trees for many choices of parameters.

Lemma 5.6. *If two symbolic trees s and t over the same variables are not equal, then for every integer N there is a combination $\bar{a}$ of parameters larger than N such that $s(\bar{a}) \neq t(\bar{a})$.*

Proof. Let $p_1, \ldots, p_n$ be all the polynomials present in s and t, listed without repetition. For any N there is a combination $\bar{a}$ of parameters larger than N such that all values $p_i(\bar{a})$ are pairwise different. Then s can be reconstructed from $s(\bar{a})$ by looking at degrees of all nodes, starting from the leaves, and similarly for t and $t(\bar{a})$. □

From the well-known Schwartz-Zippel lemma a stronger statement follows: if the parameters $\bar{a}$ are randomly chosen from $\{1, \ldots, M\}^X$ then the probability that $s(\bar{a}) \neq t(\bar{a})$ approaches 1 with $M \to \infty$. However, in the following we will only need the crude Lemma 5.6.

5.4 Applying patterns to symbolic trees

Consider a symbolic tree t, and a pattern Φ. If we choose some parameters $\bar{a}$ sufficiently large for all polynomials in t being positive, and apply Φ to $t(\bar{a})$, then we get some output tree. The following lemma shows that the dependence of the output tree on the parameter $\bar{a}$ can be described using a symbolic tree. In other words, the lemma gives a way to apply patterns to symbolic trees.

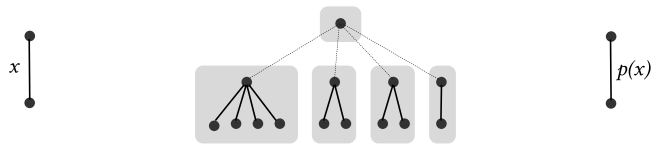
Lemma 5.7. *Let Φ be a pattern and let t be a symbolic tree with variables X. There exists a symbolic tree s with the same variables X, such that*

$$\Phi(t(\bar{a})) = s(\bar{a})$$

for all sufficiently large parameters $\bar{a}$.

Proof. See Appendix A of [Bojańczyk and Klin 2023]. Note that thanks to Lemma 5.6, the symbolic tree s is unique. □

Example 14. Consider a simple scenario where the input trees and the patterns are of height one, and there is only one input label (i.e., input trees are effectively unlabelled). For the very simple symbolic input tree t and the pattern Φ as seen below, the symbolic tree that arises from Lemma 5.7:



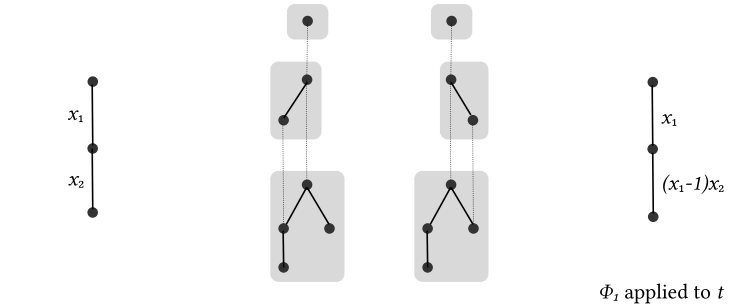
symbolic input tree t pattern Φ Φ applied to t

has only one edge labelled with the polynomial

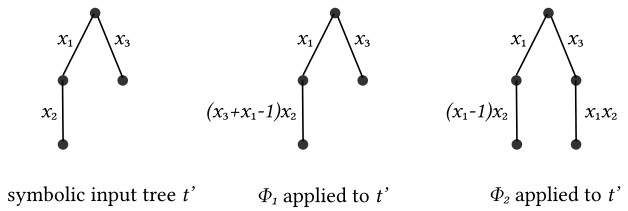
$$p(x) = x^4 - 6x^3 + 13x^2 - 7x = x(x-1)(x-2)(x-3) + x(x-1) + x(x-1) + x.$$

Indeed, for $x > 3$, this polynomial counts embeddings of the trees that label the nodes of Φ into the tree of height one with x children of the root. Note that the polynomial retains full information about the pattern, assuming that both the pattern and all labels of its nodes are of height one. In other words, the single-edge input symbolic tree distinguishes all patterns of this shape. Lemma 5.9 below says that this is a general phenomenon, although more complex input trees will be needed to distinguish more complex patterns. □

Example 15. Remaining in the unlabelled setting, consider the following symbolic input tree t and patterns Φ_1 and Φ_2 which are identical except for the embedding between the two bottom nodes:

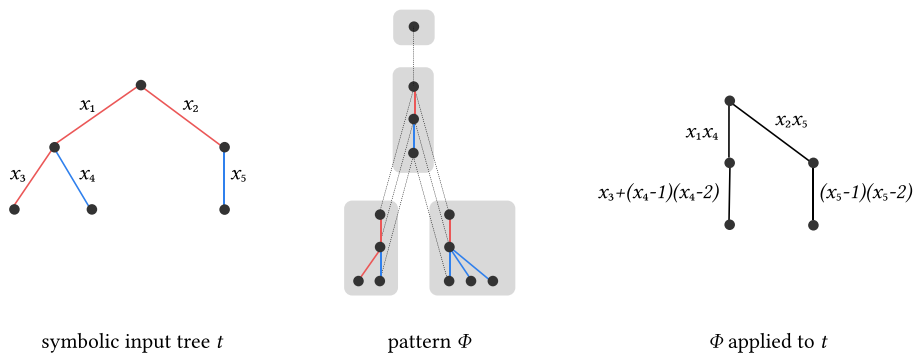


The symbolic output trees that arise from Lemma 5.7 are the same for both patterns. In other words, the symbolic input t does not distinguish the patterns. The slightly larger tree t' below does:



□

Example 16. Consider a setting with two labels for edges of input trees. For the symbolic input tree t and the pattern Φ as seen below, the tree that arises from Lemma 5.7 is shown on the right:



Indeed, substituting any positive numbers for the x_i's (with $x_4, x_5 > 2$), the output tree instance is the result of applying Φ to the input tree instance. □

We will only apply the Lemma 5.7 to symbolic trees which are *free*, meaning that all polynomials in them are pairwise distinct variables, as in Examples 14-16. We will also require the symbolic trees to be sufficiently large for Φ, in the following sense:

Definition 5.8. A tree t is *large* for a class of trees S if for every tree $s \in S$ and its substructure s' (that is, a "pruned" tree that arises from s by removing some subtrees), every embedding of s' into t extends to an embedding of s into t.

Every finite set of trees admits some large tree; it is enough to take a complete tree of large height and large degree.

We now state the main technical result in this section, which is that if we apply a pattern to a symbolic input tree that is free and large, then the output symbolic tree uniquely identifies the pattern. Using the notation from the statement of Lemma 5.7, this lemma says that Φ can be reconstructed from s.

Lemma 5.9. *Let Φ_1 and Φ_2 be patterns, and let t be a tree that is large for the set of trees that are present as node labels in either of the two patterns. View t as a free symbolic tree, with the variables being the edges of t, and consider the two output symbolic trees that arise by applying Φ_1 and Φ_2 to it. If these output symbolic trees are equal, then the patterns Φ_1 and Φ_2 are isomorphic.*

Proof. See Appendix B of [Bojańczyk and Klin 2023]. □

To complete the proof of Theorem 5.4, suppose that Φ_1 and Φ_2 are non-isomorphic patterns. Choose some large finite tree t and apply Lemma 5.9. Since the symbolic trees $\Phi_1(t)$ and $\Phi_2(t)$ are not equal, by Lemma 5.6, there must be some choice of parameters $\bar{a}$ such that applying Φ_1 and Φ_2 to the input tree $t(\bar{a})$ yields non-isomorphic outputs, thus proving Theorem 5.4.

6 DECIDING EQUIVALENCE FOR POLYREGULAR FUNCTIONS ON MULTISET TYPES

So far we have solved the equivalence problem for quantifier-free tree-to-tree interpretations on trees of bounded height. We shall now generalize that to polyregular functions on arbitrary multiset types. The more important part of the generalization is the ability of polyregular functions to use quantifiers; in this sense, the proof of the theorem below will amount to a quantifier elimination procedure that will reduce the problem to the quantifier-free case from Section 5.

Theorem 6.1. *The following problem is decidable:*

Instance. *Two polyregular functions between multiset types $f, g : \Sigma \to \Gamma$ given as first-order interpretations, or as derivations.*

Question. *Are the functions equivalent in the following sense: for every input, the two outputs are isomorphic?*

Since the proof of Theorem 4.4 is effective, i.e. for every first-order interpretation we can compute a corresponding derivation and *vice versa*, the representation of the input is not important as long as decidability is concerned.

We will prove Theorem 6.1 by reducing it to the equivalence problem for quantifier-free tree-to-tree interpretations that was shown decidable in Theorem 5.1. The reduction is non-elementary, i.e. both its running time and the size of the output instance are towers of exponentials whose height is polynomial in the input size.

6.1 Quantifier elimination

The most important part of the reduction is a quantifier-elimination result, given in Lemma 6.2 below. The idea is that every structure of a multiset type can be labelled with extra information so that first-order formulas can be replaced by quantifier-free formulas.

To formalise the notion of labelling, we use *non-copying interpretations*. These are interpretations in which there is only one component, and it has dimension one. For such an interpretation, the universe of the output structure can be regarded as a subset of the universe of the input structure.

Recall that the *quantifier rank* of a first-order formula is the maximal nesting depth of its quantifiers. For a rank $r \in \{0, 1, \ldots\}$, the *r-theory* of a structure A with distinguished elements $\bar{a} \in A^n$ is the set of all first-order rank-r formulas with n free variables that are satisfied in it. (This is often called the r-type, but we should not overload the word "type".)

Lemma 6.2. *For every multiset type Σ and numbers $r, n \geq 0$, there is a multiset type Γ and a surjective*[3] *non-copying interpretation $f : \Gamma \to \Sigma$, such that for every rank-r formula $\varphi(x_1, \ldots, x_n)$ over the vocabulary of Σ there is a quantifier-free formula $\psi(x_1, \ldots, x_n)$ over the vocabulary of Γ such that for every structure $A \in \Gamma$ and every tuple $\bar{a} \in A^n$:*

$$A, \bar{a} \models \psi \quad \text{iff} \quad \text{all elements of } \bar{a} \text{ are in the universe of } f(A) \text{ and } f(A), \bar{a} \models \varphi.$$

Proof. See Appendix **??** in [Bojańczyk and Klin 2023]. □

6.2 Reduction to Theorem 5.1

We now use quantifier elimination from Lemma 6.2 to reduce the problem in Theorem 6.1, about equivalence for polyregular functions on multiset types, to the already solved problem from Theorem 5.1, about equivalence for quantifier-free interpretations on tree types.

Consider an instance of the equivalence problem in Theorem 6.1. Apply Lemma 4.3 to the output type Γ, yielding some encoding:

$$\Sigma \underset{f_2}{\overset{f_1}{\rightrightarrows}} \Gamma \xrightarrow[\text{encode}]{} \mathsf{M}^k 1$$

The encoding is injective, since it admits an inverse decoding. As a result, appending it to both functions does not change the answer to the equivalence problem.

Let r be the maximal quantifier rank among all of the first-order formulas used by the above two interpretations. Apply Lemma 6.2 yielding some surjective function g as in the following diagram:

$$\Gamma' \xrightarrow{g} \Sigma \underset{f_2}{\overset{f_1}{\rightrightarrows}} \Gamma \xrightarrow[\text{encode}]{} \mathsf{M}^k 1$$

Because g is surjective, prepending it to the diagram does not change the answer to the equivalence problem. By the quantifier-elimination properties in Lemma 6.2, the two functions of type $\Gamma' \to \mathsf{M}^k 1$ in the above diagram are quantifier-free interpretations.

So far, we have managed to reduce the equivalence problem to the special case of quantifier-free interpretations that output unlabelled trees. The last remaining step in the reduction to Theorem 5.1 is turning the input type into trees. This works thanks to the following fact:

Lemma 6.3. *For every multiset type Γ there is a finite multiset type Δ, numbers $k, n \in \{0, 1, \ldots\}$ and a surjective quantifier-free interpretation*

$$\underbrace{\mathsf{T}_k^{edge}\Delta + \cdots + \mathsf{T}_k^{edge}\Delta}_{n \text{ times, with trees using edge representation}} \to \Gamma.$$

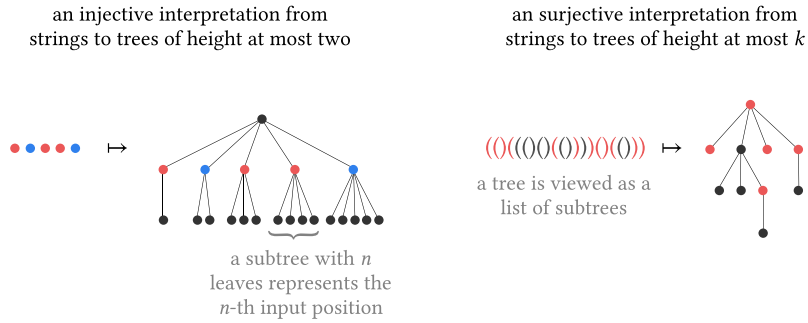
Proof. Induction on Γ. For types of the form $\mathsf{M}\Sigma$, use the fact that if an interpretation f is surjective and quantifier-free then so is $\mathsf{M}f$. □

6.3 Connections with the string-to-string case

In Theorem 6.1, we proved decidability of equivalence for tree-to-tree interpretations, with the trees being unordered and of bounded height. Decidability of the equivalence problem for string-to-string interpretations remains open. We finish this section with some comments about the connection between these two problems. As we explain below, these two problems are incomparable, because (1) as inputs, strings are more general than trees of bounded height; and (2) as outputs, trees of bounded

[3] Meaning that every structure in Σ arises as $f(A)$ for some $A \in \Gamma$, up to isomorphism.

height (in fact, height two) are more general than strings. In particular, a common generalisation of both problems would be interpretations that input strings and output trees of bounded height. Below, we give informal explanations for (1) and (2) above. Consider two interpretations:



The injective interpretation shows that trees of height two are harder than strings as outputs. The surjective one shows that strings are harder than trees of bounded height as inputs; the height bound on trees is needed there so that the interpretation can match brackets.

We believe that the ideas developed in this paper, together with above representation of output strings as trees, might be useful in the solution of the string-to-string problem.

7 APPLICATIONS TO GRAPHS OF BOUNDED TREE DEPTH

In this section, we apply the results on tree-to-tree transducers to graph-to-graph transducers. The graphs in this section are undirected and without parallel edges. We use a straightforward representation of a graph as a structure: the universe is the vertices, and there is one binary relation describing the edges. Choosing another representation, such as having edges in the universe, would give the same results. We use FO interpretations to transform graphs into other graphs.

We would like to decide equivalence for FO interpretations that describe graph-to-graph functions. The problem is undecidable in general, even for interpretations with Boolean outputs. This is because the following *satisfiability problem* is undecidable: given a first-order sentence, decide if it is true in some finite graph [Courcelle and Engelfriet 2012, Theorem 5.5]. A classical approach to working around this limitation is to consider graphs that are similar to trees. Several graph parameters can be used to formalize similarity to trees, with three important parameters being: tree-depth [Nešetřil and de Mendez 2012, Section 6.2], tree-width [Courcelle and Engelfriet 2012, Section 4.1] and clique-width [Courcelle and Engelfriet 2012, Section 4.3]. The parameters are related as follows: every class of graphs satisfies the following implications

$$\text{bounded tree-depth} \quad \Rightarrow \quad \text{bounded tree-width} \quad \Rightarrow \quad \text{bounded clique-width.}$$

For graphs of bounded clique-width (and therefore also for graphs of bounded tree-depth or tree-width), satisfiability is decidable, even if FO logic is extended to MSO logic [Courcelle and Engelfriet 2012, Theorem 5.80]. It is not known if one can decide equivalence for FO interpretations between graphs of bounded clique-width. In fact, the problem is open already for tree-width, even for tree-width 1 and dimension 1, see [Bojańczyk and Schmude 2020, p.7]. We present the first progress on this problem: we show that the equivalence problem is decidable for bounded tree-depth, for interpretations of arbitrary dimension. We begin by defining tree-depth.

Definition 7.1. A graph has tree-depth $k = 1$ if it has no edges. A graph has tree-depth at most $k > 1$ if there is a vertex, such that after removing that vertex, all connected components have tree depth at most $k - 1$.

Theorem 7.2. *The following problem is decidable:*

Instance. *Two* FO *interpretations, which input graphs of tree-depth at most k and output graphs of tree-depth at most ℓ.*

Question. *Are the interpretations equivalent in the following sense: for every input, the two output graphs are isomorphic?*

Before proving the theorem, let us make two remarks. First, the above problem would remain decidable if MSO logic was used instead of FO logic. This is because, as we have mentioned before, FO and MSO have the same expressive power on graphs of bounded tree-depth [Elberfeld et al. 2016, Theorem 1.1]. The second remark is about how the instances of the decision problem are represented. We use the following representation: we are given the bounds k and ℓ on the tree-depth, as well as a first-order interpretation that uses the vocabulary of graphs. The interpretation should have the property that if the input graph has tree-depth at most k, then the output graph has tree-depth at most ℓ. This property is decidable, since FO logic is decidable on graphs of tree-depth at most k, and there is a FO formula that checks if a graph has tree-depth at most ℓ.

We now proceed to prove Theorem 7.2. The difficulty in the theorem is that there is an implicit isomorphism check, i.e. the two interpretations might produce isomorphic graphs in different ways. This difficulty was already present in Theorems 5.1 and 6.1, because bounded-height trees also have non-trivial isomorphisms. As it turns out, that difficulty has already been addressed: Theorem 7.2 is proved by a relatively straightforward reduction to the case of multiset types from Theorem 6.1. The key step is to represent graphs of bounded tree-depth using multiset types:

Lemma 7.3. *For every $k \in \{1, 2, \ldots\}$ there are:*

(1) *a surjective interpretation from some multiset type to graphs of tree-depth at most k;*
(2) *an injective interpretation from graph of tree-depth at most k to some multiset type.*

Before proving the lemma, we use it to complete the proof of Theorem 7.2. Consider an instance of the problem in the theorem. Apply Lemma 7.3, yielding a surjective interpretation onto the input graphs, and an injective interpretation from the output graphs, as shown in the following diagram:

$$\Sigma \twoheadrightarrow \mathrm{G}_k \underset{f_2}{\overset{f_1}{\rightleftarrows}} \mathrm{G}_\ell \hookrightarrow \Gamma$$

where G_k (and G_ℓ) denote the class of graphs of tree-depth at most k (and ℓ). The equivalence of f_1 and f_2 is the same as the equivalence of the two paths in the diagram that go from Σ to Γ, and these paths represent interpretations between multiset types. For such interpretations, equivalence is decidable by Theorem 6.1. It remains to prove the lemma.

Proof of Lemma 7.3. In the proof, it will be convenient to work with vertex-labelled graphs. For a finite set Σ, let us write $\mathrm{G}_k\Sigma$ for the class of graphs that have tree-depth at most k, with vertices labelled by elements of Σ. Such graphs are viewed as structures in the same way as unlabelled graphs, with an additional unary predicate $a(x)$ for each possible label $a \in \Sigma$. Ultimately we care about unlabelled graphs, i.e. the case of $\Sigma = 1$, but the induction proofs will use other choices of Σ.

Surjective. We begin describing a surjective interpretation from some multiset type to $\mathrm{G}_k\Sigma$, for every finite set Σ. The interpretation is defined by induction on k. For $k = 1$, there is nothing to do, since a graph of tree-depth 1 is nothing but a multiset of vertices, and therefore $\mathrm{G}_1\Sigma = \mathrm{M}\Sigma$.

Consider now the induction step, where we already have a surjective interpretation for tree-depth k, and we now want to define it for $k + 1$. By definition, a graph of tree-depth at most $k + 1$ has some distinguished vertex v, such that after removing this vertex, every connected component has

strictly smaller tree-depth. Therefore, in order to represent such a graph, it is enough to give: (a) the label of the removed vertex; (b) the multiset of graphs that represent the connected components after removing the vertex; and (c) for each vertex that is not removed, one bit of information that says if it was connected to the removed vertex by an edge. This is formalized in the following claim:

Claim 7.4. *For every finite set Σ of labels, there is a surjective interpretation*

$$\Sigma \times \quad \mathrm{M}(\mathrm{G}_k(\Sigma \times \mathrm{Bool})) \qquad \twoheadrightarrow \qquad \mathrm{G}_{k+1}\Sigma.$$

Proof. The interpretation maps an element of the input type to a graph in the natural way: it takes the disjoint union of the graphs in the multiset from the second coordinate, adds a new vertex with the label from the first coordinate, connects this new vertex to the vertices in the disjoint union that have a 1 value on the extra bit, and removes this extra bit. This function is surjective by definition of tree-depth, and it is easily seen to be a first-order interpretation. □

Combining the above claim with an inductively defined interpretation from some multiset type to $\mathrm{G}_k(\Sigma \times \{0,1\})$, we get the desired surjective interpretation from a multiset type to $\mathrm{G}_{k+1}\Sigma$.

Injective. We now present an injective interpretation from graphs of tree-depth at most k to a multiset type. The rough idea is to use the inverse of the surjective interpretation described above. However, the rough idea needs some work, since there can be several choices for the removed vertex, and there might be no way of choosing a unique removed vertex. The solution is to output all possible choices, aggregated using a multiset. This is expressed in the following claim.

Claim 7.5. *For every finite set Σ of labels, there is an injective interpretation*

$$\mathrm{G}_{k+1}\Sigma \qquad \hookrightarrow \qquad \mathrm{M}(\Sigma \times \mathrm{M}(\mathrm{G}_k(\Sigma \times \mathrm{Bool})))$$

Proof. This interpretation is defined as follows. Consider a vertex v of an input graph G, and let H be a connected component of the graph obtained by deleting v in G. Similarly to Claim 7.4, define H^v to be the graph with labels from $\Sigma \times \mathrm{Bool}$ that is obtained from H by adding to the label of each vertex one bit of information that says if that vertex is connected to the deleted vertex v by an edge. The interpretation in the claim maps a graph G to

$$\left\{\!\!\left\{ \left(\text{label of } v, \left\{\!\!\left\{ H^v \;\middle|\; \begin{array}{l} H \text{ is a connected} \\ \text{component of } G \text{ with} \\ \text{vertex } v \text{ removed} \end{array} \right\}\!\!\right\} \right) \;\middle|\; \begin{array}{l} v \text{ is a vertex of } G \text{ such that after} \\ \text{removing it, every connected} \\ \text{component has tree-depth at most } k \end{array} \right\}\!\!\right\}$$

This interpretation is injective, because it inverses the function from Claim 7.4 in the following sense: if we apply the function from this claim to a graph in $\mathrm{G}_{k+1}\Sigma$, and then apply the interpretation from Claim 7.4 to each element of the resulting multiset, then we get a multiset that contains several copies of the original input graph.

It remains to prove that the function described above is indeed a FO interpretation. The main observation is that the criterion for v in the outermost multiset is definable in FO logic assuming that the input graph has bounded tree-depth. The reason is that, although FO logic cannot define graph connectivity in general, it can do so for graphs of bounded tree-depth, because in graphs of tree-depth k, paths have length at most 2^k. Therefore, the operation described in this claim can be defined using a FO interpretation. This interpretation has dimension two, because elements of the output multiset of multisets are represented using pairs (v, w) such that v is a deleted vertex and w is one of the remaining vertices. □

Similarly to the surjective case, the injective case is proved by induction on k using the above claim in the induction step. This completes the proof of Lemma 7.3. □

ACKNOWLEDGMENTS

The idea to consider regular functions on multisets was suggested to us by Marcelo Fiore. We are very grateful to the anonymous reviewers for their numerous insightful comments and corrections.

Mikołaj Bojańczyk was supported by the Polish National Science Centre (NCN) grant "Polynomial finite state computation" (2022/46/A/ST6/00072).

REFERENCES

Adrien Boiret, Radoslaw Piórkowski, and Janusz Schmude. 2018. Reducing Transducer Equivalence to Register Automata Problems Solved by "Hilbert Method". In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, Ahmedabad, India (LIPIcs, Vol. 122)*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 48:1–48:16.

Mikołaj Bojańczyk. 2018. Polyregular Functions. *CoRR* abs/1810.08760 (2018).

Mikołaj Bojańczyk. 2022. Transducers of Polynomial Growth. In *Logic in Computer Science (LICS '22)*. Association for Computing Machinery, New York, NY, USA.

Mikołaj Bojańczyk. 2023. Folding Interpretations. In *Logic in Computer Science (LICS '22)*. Association for Computing Machinery.

Mikolaj Bojańczyk, Sandra Kiefer, and Nathan Lhote. 2019. String-to-String Interpretations With Polynomial-Size Output. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. 106:1–106:14.

Mikołaj Bojańczyk and Janusz Schmude. 2020. Some Remarks on Deciding Equivalence for Graph-To-Graph Transducers. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic (LIPIcs, Vol. 170)*, Javier Esparza and Daniel Král' (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 19:1–19:14.

Mikołaj Bojańczyk and Bartek Klin. 2023. Polyregular functions on unordered trees of bounded height. *CoRR* abs/2311.04180 (2023). https://doi.org/10.48550/arXiv.2311.04180

Egon Börger, Erich Grädel, and Yuri Gurevich. 2001. *The classical decision problem*. Springer Science & Business Media.

Peter Buneman, Shamim Naqvi, Val Tannen, and Limsson Wong. 1995. Principles of programming with complex objects and collection types. *Theoretical Computer Science* 149, 1 (1995), 3–48. https://doi.org/10.1016/0304-3975(95)00024-Q Fourth International Conference on Database Theory (ICDT '92).

James Cheney, Sam Lindley, and Philip Wadler. 2014. Query Shredding: Efficient Relational Evaluation of Queries over Nested Multisets. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. 1027–1038. https://doi.org/10.1145/2588555.2612186

Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*. Encyclopedia of Mathematics and Its Applications, Vol. 138. Cambridge University Press.

Bruno Courcelle and Sang-il Oum. 2007. Vertex-minors, monadic second-order logic, and a conjecture by Seese. *Journal of Combinatorial Theory, Series B* 97, 1 (2007), 91–126.

Gaëtan Douéneau-Tabot. 2021. Pebble Transducers with Unary Output. In *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia (LIPIcs, Vol. 202)*, Filippo Bonchi and Simon J. Puglisi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 40:1–40:17.

Michael Elberfeld, Martin Grohe, and Till Tantau. 2016. Where first-order and monadic second-order logic coincide. *ACM Transactions on Computational Logic (TOCL)* 17, 4 (2016), 1–18.

Wilfrid Hodges. 1993. *Model Theory*. Cambridge University Press.

Tova Milo, Dan Suciu, and Victor Vianu. 2003. Typechecking for XML transformers. *J. Comput. Syst. Sci.* 66, 1 (2003), 66–97.

Jaroslav Nešetřil and Patrice Ossona de Mendez. 2012. Sparsity. In *Algorithms and Combinatorics*. Vol. 28. Springer, xxiv+–457.

Wilmer Ricciotti and James Cheney. 2019. Mixing Set and Bag Semantics. In *Proceedings of the 17th ACM SIGPLAN International Symposium on Database Programming Languages (DBPL 2019)*. 70–73. https://doi.org/10.1145/3315507.3330202

Detlef Seese. 1991. The structure of the models of decidable monadic theories of graphs. *Annals of pure and applied logic* 53, 2 (1991), 169–195.

Helmut Seidl, Sebastian Maneth, and Gregor Kemper. 2018. Equivalence of Deterministic Top-Down Tree-to-String Transducers Is Decidable. *J. ACM* 65, 4 (April 2018).

Received 2023-07-11; accepted 2023-11-07