

String-to-String Interpretations with Polynomial-Size Output

Mikołaj Bojańczyk

Institute of Informatics,
University of Warsaw, Poland
bojan@mimuw.edu.pl

Sandra Kiefer

Department of Computer Science,
RWTH Aachen University, Germany
kiefer@cs.rwth-aachen.de

Nathan Lhote

Institute of Informatics,
University of Warsaw, Poland
nlhote@mimuw.edu.pl

Abstract

String-to-string MSO interpretations are like Courcelle’s MSO transductions, except that a single output position can be represented using a tuple of input positions instead of just a single input position. In particular, the output length is polynomial in the input length, as opposed to MSO transductions, which have output of linear length. We show that string-to-string MSO interpretations are exactly the polyregular functions. The latter class has various characterizations, one of which is that it consists of the string-to-string functions recognized by pebble transducers.

Our main result implies the surprising fact that string-to-string MSO interpretations are closed under composition.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases MSO, interpretations, pebble transducers, polyregular functions

Acknowledgements The authors would like to thank Benedikt Brütsch for helpful discussions on the topic.

1 Introduction

A string-to-string function is called *regular* if it is computed by a deterministic two-way automaton with output. There are many equivalent models for the same class of functions: string-to-string MSO transductions [10], streaming string transducers [1], and various kinds of combinator-based formalisms [2, 8, 5].

A deterministic two-way automaton can visit each input position at most once in each state, otherwise it would loop forever. This means that the length of the run – and also the size of the output word – is linear in the input string. One way to go beyond linear-sized outputs was proposed by Milo, Suciu, and Vianu [17], following earlier work by Globberman and Harel [12]: equip the automaton with k pebbles which can be used to mark positions in the input word. To avoid making the model Turing-powerful, the pebbles are required to observe a so-called stack discipline: the pebbles are organised in a stack, and only the top-most pebble can be moved. In [3], it is shown that pebble transducers are equivalent to multiple other models: a higher-order functional programming language [3, Section 4], an imperative programming language with for-loops [3, Section 3], combinators [3, end of Section 4], and compositions of certain simple atomic functions [3, Section 1]. Because of the multitude of models and their polynomial size outputs, the class of functions recognised by these models is called *polyregular functions*.

The list of models for polyregular functions described in [3] does not include any logical model. In this paper, we fix that omission. As mentioned above, for the regular functions, which have linear size output, the logical model consists in string-to-string MSO transductions. In an MSO transduction, each position of the output string is interpreted as a single position of the input string. A natural idea to capture polyregular functions is to consider what we call *string-to-string MSO interpretations*, where a position of the output string is represented by a k -tuple of positions in the input string. At first glance, this idea looks suspicious: if string-to-string MSO interpretations were equivalent to polyregular functions, then they would be closed under composition, because the class of polyregular functions is. However, composing two string-to-string MSO interpretations

$$\Sigma^* \xrightarrow{f} \Gamma^* \xrightarrow{g} \Delta^*$$

raises the following issue. Suppose that positions of the intermediate word in Γ^* are represented by k -tuples of positions in the input word from Σ^* . If an MSO formula defining g quantifies over a set of positions in the intermediate word to define a property of the output word in Δ^* , then this corresponds to quantifying over a set of k -tuples of positions in the input word. If we assume dimension $k = 1$, then the problem dissolves, and this is why MSO transductions have dimension $k = 1$, whereas dimension $k > 1$ is never used in the context of MSO (as opposed to first-order logic, where the standard notion of transformation, i.e. first-order interpretation, uses higher dimension).

As our main result, we show that the problems discussed above only invalidate the natural construction for composing MSO interpretations, which uses substitution of formulas. Still, and surprisingly, for structures that represent strings there exists a (less natural) construction. This follows from our main result which states that polyregular functions are exactly the string-to-string MSO interpretations. Indeed, corollaries of the main result are that (a) string-to-string MSO interpretations are closed under composition; and (b) for every regular string language, its inverse image under a string-to-string MSO interpretation is also regular. This is because (a) and (b) are true for polyregular functions. Proving (a) and (b) directly for string-to-string MSO interpretations seems hard; in fact an understandable (but wrong) first reaction to the claims (a) and (b) would be that they are false, for the reasons discussed in the previous paragraph.

It is easy to see that every polyregular function is a special case of a string-to-string MSO interpretation. One argument is that a k -pebble automaton can be simulated using a string-to-string MSO interpretation, where configurations of the pebble automaton are represented using k -tuples of positions in the input word. The difficulty lies in proving the opposite direction and it comes from the stack discipline required in a pebble automaton. A k -tuple of positions used by an MSO interpretation can of course be viewed as a configuration of a pebble automaton, but there does not seem to be any reason why the resulting pebble automaton should observe stack discipline. It turns out – and this is the main technical insight of this paper – that any MSO formula which defines a linear ordering on k -tuples of positions in strings must necessarily observe an implicit stack discipline, which makes it possible to translate a string-to-string MSO interpretation into a pebble automaton.

Outline. After describing string-to-string MSO interpretations in Section 2, we revise polyregular functions via the formalism of for-programs in Section 3. In Section 4, we show that the models are equivalent.

2 Interpretations

In this section, we revise first-order and MSO interpretations, which are transformations of relational structures using formulas.

2.1 Logic and interpretations

Relational vocabularies and logic. A (*relational*) *vocabulary* is a set of relation names, each one associated with a natural number called its *arity*. For short, we refer to relational vocabularies simply as *vocabularies*. A *structure* over a vocabulary σ consists of a set called the *universe* and for each relation name of σ a corresponding relation of the same arity over the universe. To define properties of relational structures, we use monadic second-order logic and its first-order fragment with the usual syntax and semantics [20]. We use the convention that lower-case variables x, y, z range over elements and upper-case variables X, Y, Z range over sets of elements.

Interpretations. Intuitively speaking, an interpretation is a function from relational structures to relational structures where each element of the universe of the output structure is a tuple of elements of the input structure, and the relations of the output structure are defined using formulas evaluated over the input structure.

► **Definition 1** (Interpretations over general structures). *For $k \geq 1$, the syntax of a k -dimensional first-order interpretation consists of:*

1. *two vocabularies, called the input vocabulary and the output vocabulary*
2. *an FO formula over the input vocabulary with k free variables, called the universe formula.*
3. *for each n and each n -ary relation name R of the output vocabulary, an associated FO formula φ_R over the input vocabulary, with $k \cdot n$ free variables.*

MSO interpretations are defined analogously, except that formulas of MSO are used, but the free variables still range over elements and not over sets.

The semantics of an interpretation is a function from structures over the input vocabulary to structures over the output vocabulary, defined as follows.

- The universe of the output structure is the set of k -tuples of elements in the universe of the input structure which satisfy the universe formula from item 2 in Definition 1.
- An n -ary relation name R of the output vocabulary is interpreted as the set of n -tuples of k -tuples from the input structure, for which (a) each k -tuple is in the output universe, and (b) the entire $(n \cdot k)$ -tuple satisfies the formula φ_R in item 3 in Definition 1.

Composition. First-order interpretations are closed under composition [14, p. 218]. Let us recall the proof. Suppose that we want to compose interpretations

$$\text{structures over } \sigma_1 \xrightarrow{\mathcal{I}_1} \text{structures over } \sigma_2 \xrightarrow{\mathcal{I}_2} \text{structures over } \sigma_3$$

of dimensions k_1 and k_2 , respectively. The $(k_1 \cdot k_2)$ -dimensional composition is obtained from \mathcal{I}_2 as follows: (a) quantification over elements of \mathcal{I}_2 is replaced by a quantification over k_1 -tuples of elements; and (b) relation names from σ_2 that appear in the input of \mathcal{I}_2 are replaced by the corresponding formulas from \mathcal{I}_1 . This idea does not work for MSO in general, since set quantification in \mathcal{I}_2 would need to be replaced by quantification over sets of k_1 -tuples. It does work when $k_1 = 1$. This essentially corresponds to Courcelle's transductions, for which closure under composition follows naturally [7, Theorem 7.14]. To recover closure

under composition for $k_1 \geq 2$, one can use (not necessarily monadic) second-order logic, which by Fagin's Theorem [16, Corollary 9.9] corresponds to the polynomial hierarchy of computational complexity and is outside the scope of this paper.

2.2 String-to-string interpretations

We are interested in interpretations that transform structures which represent strings. While there are two natural ways to model strings as relational structures, namely with an order relation or with a successor relation, only the order relation is useful in our context.

► **Definition 2** (String-to-string interpretations). *For a string $w \in \Sigma^*$, its ordered model is defined to be the following relational structure, denoted by \underline{w} :*

- *the universe consists of the positions in the string, i.e., natural numbers;*
 - *there is a binary relation for the natural order on positions;*
 - *for each $a \in \Sigma$ there is a unary relation which is satisfied by every position with label a .*
- A function $f: \Sigma^* \rightarrow \Gamma^*$ is called a first-order string-to-string interpretation if the corresponding transformation on ordered models is a first-order interpretation for strings with length at least two¹. Likewise we define MSO string-to-string interpretations.*

► **Example 3.** Consider the function $f: \{a, b\}^* \rightarrow \{a, b\}^*$ which maps a word to the concatenation of all of its reversed prefixes, as in the following example (with prefixes grouped for better readability):

$$abbb \mapsto \underbrace{a} \underbrace{ba} \underbrace{bba} \underbrace{bbba}.$$

This transformation is the running example in [3]. We show that f can be seen as a string-to-string first-order interpretation. The dimension is 2, i.e. positions in the output word represent pairs of positions in the input word. A pair (x_1, x_2) of positions in the input word is used in the output word if it satisfies the universe formula $x_2 \leq x_1$. The idea is that x_1 represents the length of the prefix, while x_2 is the position in that prefix. The label of a position (x_1, x_2) is inherited from the second coordinate, as expressed by the formulas corresponding to labels on the output structure:

$$\varphi_a(x_1, x_2) = a(x_2) \quad \varphi_b(x_1, x_2) = b(x_2)$$

The order on the positions of the output word is defined by the formula

$$\varphi_{\leq}(\underbrace{x_1, x_2}_{\substack{\text{a position of} \\ \text{the output word}}}, \underbrace{x'_1, x'_2}_{\substack{\text{another position of} \\ \text{the output word}}}) = (x_1 < x'_1) \vee (x_1 = x'_1 \wedge x_2 \geq x'_2).$$

Note that the above formula defines the lexicographic ordering on pairs of positions, with the first coordinate being used in increasing order, and the second coordinate being used in decreasing order. This, as it will turn out, is not a coincidence, since our main technical result says that it is impossible to define a linear order on tuples of positions without implicitly using some kind of lexicographic ordering.

¹ A typical operation we want to model is string duplication. When the input length is at least two, one can represent additional copies of the input string using a higher dimension. For input length $n \leq 1$, the output length will be $n^k \leq 1$ regardless of the dimension k . Another solution to this issue would be to have duplication built into the definition of interpretations.

Successor instead of order. When modelling a string as a relational structure, we use the order on positions. An alternative solution would be to use just the successor relation. The difference between the two solutions is that it is harder to define an order on k -tuples of positions than it is to define a successor relation. It turns out that the difference is crucial, and functions that output strings with successor can be ill-behaved. Note that whether or not the input string is equipped with an order or a successor relation makes no difference, since the order on the position of the input string can be recovered in MSO, which can compute the transitive closure of binary relations on positions.

Define the *successor model* of a string in the same way as the ordered model from Definition 2, except that a binary relation for successor is used instead of the order. Define a *successor-MSO string-to-string interpretation* to be a string-to-string function which is computed by an MSO interpretation, assuming that strings are represented by their successor models. Likewise, we define successor-first-order string-to-string interpretations. Successor-first-order string-to-string interpretations are closed under composition, because first-order interpretations are closed under composition. On the other hand, successor-MSO string-to-string interpretations are not closed under composition and lead to undecidability, as summarised in the following theorem. The proof can be found in Appendix A.

► **Theorem 4.**

1. *The class of successor-MSO string-to-string interpretations is not closed under composition, and strictly contains the class of (order-)MSO string-to-string interpretations.*
2. *The following is undecidable: given a successor-first-order string-to-string interpretation f and a regular language L over the output alphabet, decide if $f^{-1}(L)$ is nonempty.*

3 Polyregular functions

Here we describe the class of polyregular functions. It has several equivalent characterisations, see [3, Theorem 4.4], one of which consists in the aforementioned pebble transducers. For the purposes of this paper, it will be most convenient to use a slightly more abstract characterisation in terms of for-programs, a machine model for string-to-string functions. We just explain the formalism on short examples, for a more detailed description see [3].

<pre> for x in first..last for y in last..first if y ≤ x and a(y) then output a if y ≥ x and b(y) then output b </pre>	<pre> for x in first..last var P : Bool for y in last..first if y ≥ x then P := not P if P and a(x) then output a if P and b(x) then output b </pre>	<pre> for y in first..last if x1 ≤ y and y ≤ x2 and a(y) P := true </pre>
--	--	---

(a) A for-program for the function in Example 3.

(b) A for-program with a Boolean variable P .

(c) A for-program which checks if there is an a between the positions x_1 and x_2 .

■ **Figure 1** Example for-programs.

Most of the syntactic constructions that can be used in a for-program are illustrated in Figure 1a: (1) variables ranging over positions in the input word; (2) for-loops in which a variable iterates over all positions in the input word in increasing or decreasing order; (3) if-statements which depend on the order/labels of variables; (4) instructions which output

6 String-to-String Interpretations

letters. Position variables cannot be declared or written to, they are implicitly declared by for-loops and their only updates are the iterations performed by the for-loops.

The only feature of for-programs that is not used in Figure 1a is (5) Boolean variables. Figure 1b shows a program that outputs only those letters in the input word which have even distance to the last position. In the program, the Boolean variable P is declared in the scope of a for-loop. On each iteration of the loop, the variable is reinitialised to false.

A for-program is called *first-order definable* if Boolean variables can only be updated from false, which is their initial value upon declaration, to true. In other words, the only allowed update for Boolean variables is $P := \text{true}$. For the first-order restriction, it is important that Boolean variables can be declared inside for-loops, and that they are reinitialised to false at each iteration of the loop that they are declared in. The reason for the name “first-order definable” is that one can define in first-order logic the reachability relation on program states of the for-program, see [3, Lemma 5.3].

► **Definition 5.** *A string-to-string function is called polyregular if it is computed by a for-program. It is called first-order polyregular if it is computed by a first-order definable for-program.*

The class of polyregular functions has other characterisations, including the string-to-string pebble transducers introduced by Milo, Suciú and Vianu [17], as well as a higher-order functional programming language [3, Section 4]. The main result of this paper, Theorem 7 in the next section, adds a logical characterisation, namely string-to-string MSO interpretations.

Evaluating first-order formulas. The for-programs described above take as input strings and also output strings. One can also consider for-programs which input a string with distinguished positions and which output a Boolean value, as in Figure 1c. The distinguished positions are represented by free variables (here x_1 and x_2) while the output value is taken from some distinguished Boolean variable, here P .

► **Lemma 6.** *Let $\varphi(x_1, \dots, x_k)$ be an FO formula over strings. There is a first-order for-program which computes the following.*

- **Input.** *A word $w \in \Sigma^*$ and positions x_1, \dots, x_k in w ;*
- **Output.** *Yes or No, depending on whether w satisfies $\varphi(x_1, \dots, x_k)$.*

Proof. The for-program implements the semantics of an FO formula. For each quantifier, it loops over all possible values for the quantified position, and a Boolean variable is used to remember if some value has already been found which renders the formula true. ◀

A similar result is true for MSO formulas, but the proof for that statement uses automata.

4 Equivalence

We show that the models defined in Sections 2 and 3 are equivalent.

► **Theorem 7.**

1. *String-to-string MSO interpretations are exactly the polyregular functions.*
2. *First-order string-to-string interpretations are exactly the first-order polyregular functions.*

Since the class of polyregular functions is closed under composition², we obtain:

² Closure under composition was proved for pebble transducers in [9, Theorem 11] and for the class of for-programs in [3, Section 8.1] as a step in proving equivalence with the other models of polyregular functions.

► **Corollary 8.** *String-to-string MSO interpretations are closed under composition.*

By using Theorem 7, the proof of the corollary passes through for-programs. We are not aware of any direct proof that does not exploit the equivalence to polyregular functions.

The rest of this paper is dedicated to the proof of Theorem 7. We begin with a reduction of the first to the second item. This reduction illustrates a general phenomenon, namely that results about first-order polyregular functions often imply results about general polyregular functions, despite the latter class being larger. The reason behind this phenomenon is the following lemma, which says that for every polyregular function, all of the behaviour that is not first-order definable can be pushed into a simple preprocessing step. Define a *rational function*, see [4, Section 13.2], to be a string-to-string function which is recognised by a nondeterministic automaton, where every transition is labelled by a pair consisting of a letter from the input alphabet and a string over the output alphabet, and which is unambiguous in the sense that every input string admits exactly one accepting run.

► **Lemma 9.**

1. *A function is polyregular if and only if it is a composition consisting of:*
 - a. *a (letter-to-letter) rational function; followed by*
 - b. *a first-order polyregular function.*
2. *A function is an MSO string-to-string interpretation if and only if it is a composition consisting of:*
 - a. *a (letter-to-letter) rational function; followed by*
 - b. *a first-order string-to-string interpretation.*

The proof of Lemma 9 is based on ideas from [6, 15, 3] and uses factorisation forests.

Proof. The right-to-left implications in items 1 and 2 are proved the same way: both polyregular functions and MSO string-to-string interpretations are closed under pre-composition with rational functions. For the class of polyregular functions, this holds because it is closed under composition and contains all rational functions [3, Theorem 1.6]. For MSO string-to-string interpretations, one observes that rational functions are a special case of MSO string-to-string interpretations of dimension 1 (see [11, Figure 7], where MSO interpretations of dimension 1 are the same as the so-called regular functions), and MSO interpretations are closed under pre-composition with such functions (see the remarks at the end of Section 2.1).

To prove the left-to-right implications in items 1 and 2, namely the decomposition into rational pre-processing and first-order post-processing, we use the following claim.

A *letter-to-letter rational function* is a rational function where every transition in the underlying automaton is labelled with exactly one output letter, in which case the input and output strings have the same set of positions.

▷ **Claim 10.** Let φ be an MSO formula which selects k -tuples of positions in strings over an alphabet Σ . There are a letter-to-letter rational function $f: \Sigma^* \rightarrow \Gamma^*$ and a first-order formula ψ which selects k -tuples of positions in strings over the alphabet Γ such that

$$w \models \varphi(\bar{x}) \quad \text{iff} \quad f(w) \models \psi(\bar{x}) \quad \text{for every } w \in \Sigma^* \text{ and } k\text{-tuple of positions } \bar{x}.$$

The claim is the special case of [6, Theorem 2] for finite strings instead of infinite trees, and its proof uses factorisation forests (see [19]). Another proof of the above claim is in [15, Theorem 3.2]. Using the claim, we immediately get the left-to-right implications in item 2. For item 1, we also use Lemma 9 to obtain a first-order for-program realizing the function. The main idea is that if the reachability relation is first-order definable, then one can define a first-order query which accepts consecutive produced tuples. ◀

With the lemma, we show that item 2 in Theorem 7 implies item 1, i.e. if first-order string-to-string interpretations are exactly the first-order polyregular functions, then MSO interpretations are exactly the polyregular functions:

$$\begin{array}{ll}
 \text{polyregular} = & \text{by Item 1 of Lemma 9} \\
 (\text{first-order polyregular}) \circ \text{rational} = & \text{by Item 2 of Theorem 7} \\
 (\text{first-order interpretations}) \circ \text{rational} = & \text{by Item 2 of Lemma 9} \\
 \text{MSO interpretations} &
 \end{array}$$

It remains to prove item 2 in Theorem 7, i.e. that first-order string-to-string interpretations are exactly the first-order polyregular functions. The right-to-left inclusion follows immediately from [3, Lemma 5.3], which says that a formula in first-order logic can define the reachability relation on program states in first-order for-programs. We are left with the left-to-right-inclusion:

$$\text{first-order string-to-string interpretations} \subseteq \text{first-order definable for-programs} \quad (1)$$

The rest of the paper is devoted to showing the above inclusion. When simulating a first-order interpretation by a for-program, we will mainly be concerned with the universe of the output string (which is a set of k -tuples of positions in the input string) and its ordering. The labelling of the k -tuples can then be recovered using the for-program from Lemma 6. The main result is that every first-order definable linear ordering on tuples of positions can be implemented by a for-program. To be able to speak about this result, we introduce some notation for devices that produce lists of tuples of positions.

Enumerators. Let $k \in \mathbb{N}$. A k -enumerator over an alphabet Σ is a function of the following form:

- **Input.** A string $w \in \Sigma^*$;
- **Output.** A list of k -tuples of positions in w , which is nonrepeating³.

We compare the following two ways of implementing k -enumerators:

1. A k -enumerator is called *definable* if there are two FO formulas: one with k variables, which says when a tuple is part of the output list, and one with $2k$ variables, which defines a total order on the tuples selected by the first formula.
2. A k -enumerator is called *programmable* if its output can be computed by a first-order for-program which instead of outputting letters uses instructions of the form **output** $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ where $\mathbf{x}_1, \dots, \mathbf{x}_k$ are position variables.

For definable k -enumerators, the order on tuples in the output list is given explicitly by the formula φ , while in programmable ones, the order is implicit from the order in which the output instructions are executed during the computation.

► **Example 11.** We present an enumerator based on Example 3. Consider the 2-enumerator which outputs all pairs of positions (x_1, x_2) with $x_2 \leq x_1$, listed in lexicographic order, where x_1 is ordered in increasing order and x_2 is ordered in decreasing order. Here is an example:

$$abbb \mapsto (1, 1), (2, 2), (2, 1), (3, 3), (3, 2), (3, 1), (4, 4), (4, 3), (4, 2), (4, 1)$$

³ Every tuple appears at most once, but positions can appear in multiple tuples. We need this for the existence of the formulas stated in the following definitions.

This enumerator is definable, as witnessed by the formula φ_{\leq} in Example 3. The formula φ_{\leq} is quantifier-free, but in general, quantifiers are allowed. Here is a for-program which computes the same function:

```

for x1 in first..last
  for x2 in last..first
    if x2 ≤ x1 then
      output (x1,x2)

```

The following lemma is the main technical result of this paper.

► **Lemma 12.** *Every definable k -enumerator is also programmable.*

Our proof of Lemma 12 uses two fundamental ingredients. The first is by now standard: this is Simon’s factorisation forest theorem [19], which roughly says that every string can be cut into pieces that are similar to each other. The second ingredient is new: the Domination Lemma, presented in Section 4.1, roughly says that if a string is cut into pieces that are similar to each other, then any first-order definable linear order on tuples of positions must observe an implicit stack discipline. These two results are combined in Section 4.2 to prove Lemma 12. Before we proceed with the proof of Lemma 12, we use it to complete the proof of Theorem 7.

Proof of Theorem 7, second part. The only part of Theorem 7 that has not been proved yet is that every first-order string-to-string interpretation is polyregular. Suppose that f is a k -dimensional first-order string-to-string interpretation. Consider the k -enumerator which inputs a string w and outputs the list of k -tuples of positions in w that are used to represent output positions of $f(w)$, in the appropriate order. Apply Lemma 12 to obtain a first-order for-program g which computes the same list. To compute the original function f , we use a for-program which behaves as g , except that instead of outputting a k -tuple of positions like g , it uses the program described in Lemma 6 as a subroutine to check what is the output letter that should be produced for this tuple, and outputs that letter. ◀

4.1 The Domination Lemma

In this section we present the Domination Lemma, which says that if \prec is a first-order definable linear order on k -tuples of positions in a string, then there is an implicit stack discipline in the following sense. For every type (see below) t of tuples of positions there is a coordinate $d \in \{1, \dots, k\}$ such that for the subset of k -tuples of positions consisting in all of type t , the order \prec is uniquely determined by the order of the d -th coordinates in the string.

We begin by explaining the notions of types. For $r \in \{0, 1, \dots\}$, the *rank r type* of a structure \mathfrak{A} with k distinguished positions $\bar{x} := (x_1, \dots, x_k)$ is defined to be the set of first-order formulas of quantifier rank at most r and k free variables that are true in \mathfrak{A}, \bar{x} . The number k is the *arity* of the type. For arity 0, we talk about the *rank r type of the structure \mathfrak{A}* . If the structure \mathfrak{A} is implicit from the context, then we talk about the *rank r type of the tuple \bar{x}* . For every finite vocabulary, there are finitely many types of given arity and rank. We write \equiv_r for the equivalence relation on structures with distinguished elements of having the same rank r type. For a binary relation R , its *inverse* is the set $\{(v, u) \mid (u, v) \in R\}$. For $p \in \{1, -1\}$, define R^p to be either R or its inverse, depending on the value of p .

► **Lemma 13 (Domination Lemma).** *For all $k, m, r \in \{1, 2, \dots\}$, there is an $\omega \in \{1, 2, \dots\}$ with the following property. Let $n \in \{1, 2, \dots\}$, let w_1, \dots, w_n be strings over some alphabet*

Σ and let \mathfrak{A} be the ordered structure of the concatenation $w_1 \cdots w_n$ extended with the block order defined by

$$x \sqsubset y \quad \text{if} \quad x \text{ is a position in } w_i \text{ and } y \text{ is a position in } w_j \text{ for some } i < j.$$

Let \prec be a linear order on k -tuples in \mathfrak{A} defined by a first-order formula of quantifier rank r , and let t be a k -ary rank ω type over the vocabulary of \mathfrak{A} . If

$$w_i \equiv_{\omega} w_{i+1} \quad \text{holds for all } i \in \{1, \dots, n-1\} \text{ with at most } m \text{ exceptions,}$$

then there is a $d \in \{1, \dots, k\}$, called the dominating coordinate, and a $p \in \{-1, 1\}$, called the polarity, such that

$$x_d \sqsubset^p y_d \quad \text{implies} \quad (x_1, \dots, x_k) \prec (y_1, \dots, y_k) \quad \text{for all } \underbrace{x_1, \dots, x_k}_{\text{of type } t}, \underbrace{y_1, \dots, y_k}_{\text{of type } t} \text{ in } \mathfrak{A}.$$

The Domination Lemma is the technical heart of this paper. The full proof is presented in Appendix C. To explain more intuitively some of the ideas that we use, we treat a special case in detail. In the Domination Lemma, the structure \mathfrak{A} consists of blocks organised in a linear way. A very simple linear order – although infinite – is the natural one on the rational numbers; one reason for its simplicity is that quantifiers can be eliminated (see [13, Section 5.6.2]). Because of this, it is quite easy to prove a version of the Domination Lemma for the rational numbers and still its proof bears some similarity to the proof of the general case.

► **Lemma 14 (Rational Domination Lemma).** *Let \prec be a linear ordering on k -tuples of rational numbers defined by a quantifier-free (equivalently, first-order) formula using only the usual ordering $<$ on rational numbers. Then there is a coordinate $d \in \{1, \dots, k\}$ and a polarity $p \in \{-1, 1\}$ such that*

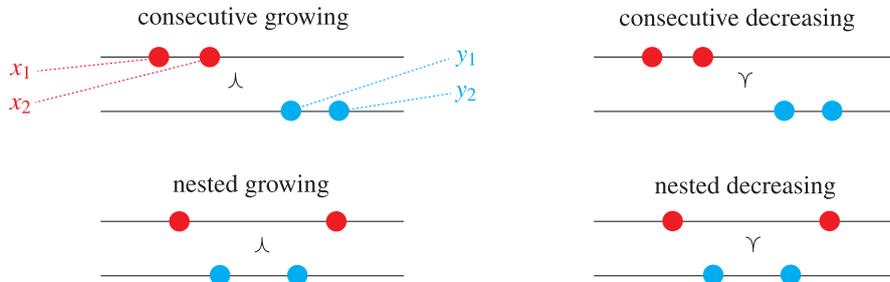
$$x_d <^p y_d \quad \text{implies} \quad (x_1, \dots, x_k) \prec (y_1, \dots, y_k)$$

for all tuples of rational numbers satisfying $x_1 < \dots < x_k$ and $y_1 < \dots < y_k$.

Proof. We first prove the statement for $k = 1$ and $k = 2$ and then we deduce the general case.

1. When $k = 1$, then the formula defining \prec must be either $x < y$ or $x > y$.
2. For $k = 2$, we do a case analysis. Note that whether $\bar{x} \prec \bar{y}$ or $\bar{y} \prec \bar{x}$ holds depends only on the order relationship of the positions in \bar{x} and \bar{y} in the rational numbers and not on the precise values in \bar{x} and \bar{y} .

The following picture shows the two possible relationships for two pairs \bar{x} and \bar{y} when they are “consecutive” and the two possible relationships when they are “nested”:

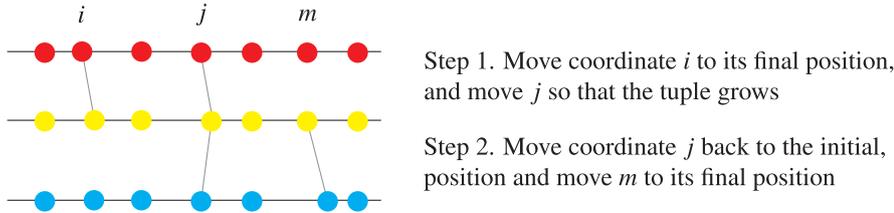


Suppose we are given a pair \bar{x} and without loss of generality, assume the “consecutive growing” case for a second pair \bar{y} . We only show the proof for the case that there is a pair

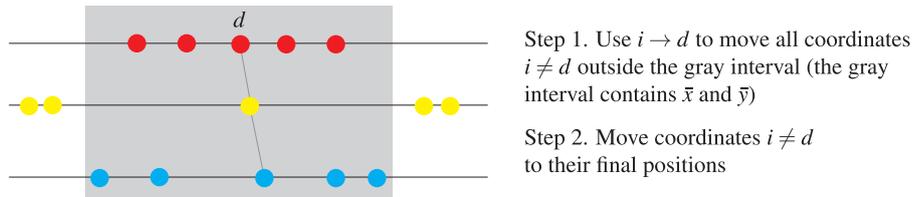
\bar{y}' such that \bar{x} and \bar{y} are “nested growing” (“nested decreasing” works analogously). We prove that $d = 1$ is dominating for \prec with polarity $p = 1$. Consider all three remaining configurations of pairs \bar{x} and \bar{y} with $x_1 < y_1$. In all cases, $\bar{x} \prec \bar{y}$ is proved by finding an intermediate pair (drawn in yellow), whose order with respect to \bar{x} and \bar{y} follows from the assumptions “consecutive/nested growing” (in the pictures below, we assume that lower lines represent bigger tuples in the ordering \prec):



3. Consider the case $k > 2$. Fix a “growing” tuple of k rational numbers, i.e. a tuple \bar{z} such that for $1 \leq i < j \leq k$ it holds that $z_1 \leq z_i < z_j \leq z_k$. Define $\prec_{i,j}^{\bar{z}}$ to be the restriction of \prec to tuples that agree with \bar{z} on coordinates from $\{1, \dots, k\} \setminus \{i, j\}$. Using the reasoning from the previous item, the ordering $\prec_{i,j}^{\bar{z}}$ must admit some dominating coordinate $d \in \{i, j\}$ and one of the cases “growing” or “decreasing”. This must hold for every choice of \bar{z} and i, j . Furthermore, the dominating coordinate d depends only on i and j and not on \bar{z} , likewise for the choice of “growing” or “decreasing”. Let us write $i \rightarrow j$ if j dominates, otherwise we write $j \rightarrow i$. The reasoning in the following picture shows that \rightarrow is transitive, i.e. $i \rightarrow j$ and $j \rightarrow m$ implies $i \rightarrow m$:



Therefore, \rightarrow is in fact a total order on $\{1, \dots, k\}$. Let d be the maximum with respect to this order. The following picture explains why d is the dominating coordinate d from the statement of Lemma 13.



Suppose without loss of generality that we are in the “growing” case for each pair of coordinates. Then we can first move all coordinates apart from d to positions smaller than $\min\{x_1, y_1\}$ or bigger than $\max\{x_k, y_k\}$ and then use the dominations $i \rightarrow d$ to move them, one by one, to their final positions (always increasing the d -th coordinate slightly to a value in the open interval (x_d, y_d)).



4.2 Proof of Lemma 12

We now return to Lemma 12, i.e., we prove that every definable k -enumerator is also programmable. In the proof, we use the following version of the Factorisation Forest Theorem. We use the term *interval* for a connected set of positions in a string.

► **Theorem 15** (Factorisation Forest Theorem, aperiodic variant). *Let $h: \Sigma^+ \rightarrow S$ be a semigroup homomorphism, where S is finite and aperiodic. Then there exists a function which assigns to each string in Σ^+ a partition of the positions into intervals (so-called blocks) such that:*

1. *All blocks are nonempty, and for each string in Σ^+ of length at least 2, there are at least two blocks.*
2. *If a string has at least three blocks, then all of the blocks have the same value under h .*
3. *There exists $M \in \mathbb{N}$ such that all strings have height at most M , where the height of a string is defined as follows: letters have height 1, for other strings the height is the maximum of the heights of its blocks + 1.*
4. *There is a first-order formula φ such that for every string w , the positions satisfying $\varphi(x)$ are exactly the first positions of the blocks of w .*

Apart from the Factorisation Forest Theorem and the Domination Lemma, our proof uses the following straightforward result on combining outputs of two for-programs. As a convention, if ψ is a first-order formula with k free variables and f is a k -enumerator, then $f|\psi$ denotes the k -enumerator where the output list of f is filtered so that it contains only tuples satisfying ψ .

► **Lemma 16** (Merging Lemma). *Let f be a definable k -enumerator. Let Φ be a finite set of FO formulas ψ , each one with k free variables, such that every k -tuple of positions satisfies at least one formula from Φ . Then f is programmable if and only if every $f|\psi$ is programmable.*

Proof. For the left-to-right implication, we observe that the filtering $f|\varphi$ can be implemented by a for-program thanks to Lemma 6. We are left with the right-to-left implication. It suffices to examine the case $|\Phi| = 2$. The general case follows by a straightforward induction.

Suppose that $f|\varphi_1$ and $f|\varphi_2$ are implemented by programs f_1, f_2 . First check whether the first tuple in the output satisfies φ_1 or φ_2 using the result from Lemma 6 and an if-statement, and then run a different program for each of the two outcomes. By symmetry, it suffices to consider inputs where the first tuple in the output of $f|(\varphi_1 \vee \varphi_2)$ satisfies φ_1 . Take the code of f_1 , and after each instruction which outputs a tuple of positions \bar{x} , run a copy of the code for f_2 , with its output restricted to tuples \bar{y} which satisfy:

- \bar{y} is after \bar{x} according to f ; and
- there are no other tuples from the output of f_1 between \bar{x} and \bar{y} .

The first item can be checked by a for-program using the assumption that f is definable and Lemma 6, while the second item can be checked by running a nested copy of f_1 . ◀

We are now ready to prove Lemma 12. Let f be a definable k -enumerator. We need to describe a for-program which outputs the same list of tuples as f . Let r be the maximal quantifier rank of the first-order formulas used in the definition of f . Apply the Domination Lemma to k , $m := 5k$, and r , yielding a constant ω . Define h to be the function which maps a string $w \in \Sigma^+$ to the rank ω type of the corresponding ordered model of w . Compositionality of first-order logic (see [16, Section 3.4]) on strings says that the image of h , the set of rank ω types of strings, is a finite aperiodic semigroup and h is a semigroup homomorphism. Apply the Factorisation Forest Theorem to h , yielding a function which partitions each string into blocks and an upper bound M on heights of strings. By abuse of notation, we lift notions about strings to intervals inside strings: the height of an interval X in a string w is defined to be the height (in the sense of item 3 in Theorem 15) of the infix of w induced by X . Likewise, we define the blocks of X as the blocks of the infix induced by X , viewed as intervals contained in X .

To show that f is also programmable, we use an induction over heights in factorisation forests. More precisely, we prove that for every $i \in \mathbb{N}$ there is a for-program which computes the following:

- **Input.** A string $w \in \Sigma^+$ with distinguished nonempty intervals X_1, \dots, X_k that are pairwise equal or disjoint, and such that the sum of their heights (in the sense of Theorem 15) is at most i . Each interval is represented by its first and its last position.
- **Output.** The list $f(w)$ restricted to tuples in $X_1 \times \dots \times X_k$.

By item 3 in Theorem 15, the for-program with parameter $i := kM$ will work for every choice of pairwise equal or disjoint intervals, in particular when all of the intervals are the entire string. The induction base $i = k$ (where every interval has the height 1) is straightforward: each interval is a singleton, and the for-program simply checks if the unique tuple in $X_1 \times \dots \times X_k$ belongs to the output of f by using the subroutines from Lemma 6. The rest of the proof is devoted to the induction step, more specifically, to producing the correct order of the tuples: whether a tuple belongs to the output or not can again be checked using the subroutines from Lemma 6.

Let X_1, \dots, X_k be intervals in an input string w that are pairwise disjoint or equal. Define \mathcal{X} to be the coarsest partition of the positions in the input string into intervals that satisfies $X_1, \dots, X_k \in \mathcal{X}$. This partition uses at most $2k + 1$ intervals. Consider a factorisation

$$w = w_1 \cdots w_n$$

where each w_j is a block of one of the elements of \mathcal{X} . Define \mathfrak{A} as in the Domination Lemma, i.e. as the ordered structure of w extended with an extra order \sqsubset that describes the partition into factors w_1, \dots, w_n . By item 4 of the Factorisation Forest Theorem, the order \sqsubset can be defined by a first-order formula which uses the input string and the endpoints of the intervals X_1, \dots, X_k . It follows that for every k -ary rank ω type t over the vocabulary of \mathfrak{A} , there is a corresponding first-order formula which selects the k -tuples of positions in w that have type t in \mathfrak{A} . Since there are finitely many choices of t , it follows from the Merging Lemma that it is enough to show that for every t , there is a for-program which outputs the tuples of type t .

Let t be a k -ary rank ω type over the vocabulary of \mathfrak{A} . We show a for-program which outputs all tuples in

$$T := \{\bar{x} \in X_1 \times \dots \times X_k : \bar{x} \text{ has type } t \text{ and is in the output of } f(w)\}$$

according to their order given by $f(w)$, call this order \prec .

If an interval from \mathcal{X} has more than two blocks, then, by item 2 of the Factorisation Forest Theorem, all of these blocks have the same image under h , i.e., the same rank ω type. Since there are at most $2k + 1$ intervals, it follows that with at most $2(2k + 1) - 1 = 4k + 1 < 5k$ exceptions, consecutive strings w_j and w_{j+1} have the same rank ω type. Hence, for the order \prec defined by $f(w)$, the Domination Lemma yields $d \in \{1, \dots, k\}$ and $p \in \{-1, 1\}$ such that

$$x_d \sqsubset^p y_d \quad \text{implies} \quad (x_1, \dots, x_k) \prec (y_1, \dots, y_k) \quad \text{for all } \underbrace{x_1, \dots, x_k}_{\text{of type } t}, \underbrace{y_1, \dots, y_k}_{\text{of type } t} \text{ in } \mathfrak{A}.$$

This means that the tuples in T are \prec -ordered as $T_1 \prec^p T_2 \prec^p \dots \prec^p T_s$, where s is the number of blocks in X_d and T_j consists of the tuples from T where the coordinate x_d is in the j -th block of X_d . Our for-program can simply loop over all the blocks of X_d – in increasing or decreasing order depending on the choice of p – because the endpoints of each block can be identified in first-order logic due to item 4 of the Factorisation Forest Theorem. In each iteration of the loop, the for-program outputs the tuples in the corresponding T_j using the following claim, thus completing the proof of the lemma.

▷ Claim 17. There is a for-program which inputs the i -th block of X_d , given by its endpoints, and outputs the tuples from T_j ordered according to \prec .

Proof of the claim. The general idea is to replace X_d with its j -th block (call this block X) and use the induction assumption. However, if there is an $i \neq d$ such that $X_j = X_d$, then replacing X_d with X would violate the assumption that the intervals are pairwise disjoint or equal (since $X \subsetneq X_j$). To overcome this issue, we use the following simple case disjunction. For each of the 3^k possible values of

$$v \in \{\text{positions before } X, X, \text{positions after } X\}^k$$

construct a for-program that outputs all tuples from $Y_1 \times \dots \times Y_k$, where Y_j is the intersection of X_j with the j -th entry of v . Since each Y_j is a union of blocks of X_j , it is empty or its height is at most the height of X_j . Furthermore, if Y_d is nonempty, then it is X , which is a block of X_d , and therefore its height is strictly smaller than the height of X_d . It follows that the induction assumption can be applied to produce all tuples in $Y_1 \times \dots \times Y_k$, for any given choice of v . These choices can be combined using the Merging Lemma. ◀

References

- 1 Rajeev Alur and Pavol Cerný. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 599–610, 2011.
- 2 Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 9. ACM, 2014.
- 3 Mikołaj Bojańczyk. Polyregular functions, 2018. [arXiv:1810.08760](https://arxiv.org/abs/1810.08760).
- 4 Mikołaj Bojańczyk and Wojciech Czerwiński. Automata toolbox. URL: <https://www.mimuw.edu.pl/~bojan/upload/reduced-may-25.pdf>.
- 5 Mikołaj Bojańczyk, Laure Daviaud, and Shankara Narayanan Krishna. Regular and first-order list functions. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 125–134, 2018.
- 6 Thomas Colcombet. A combinatorial theorem for trees. In *International Colloquium on Automata, Languages, and Programming*, pages 901–912. Springer, 2007.
- 7 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic. A Language-Theoretic Approach*. Cambridge University Press, June 2012.
- 8 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular transducer expressions for regular transformations. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 315–324, 2018. URL: <https://doi.org/10.1145/3209108.3209182>.
- 9 Joost Engelfriet. Two-way pebble transducers for partial functions and their composition. *Acta Informatica*, 52(7-8):559–571, 2015.
- 10 Joost Engelfriet and Hendrik Jan Hoogeboom. Mso definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254, 2001.
- 11 Emmanuel Filiot and Pierre-Alain Reynier. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19, 2016.
- 12 Noa Globberman and David Harel. Complexity Results for Two-Way and Multi-Pebble Automata and their Logics. *Theor. Comput. Sci.*, 169(2):161–184, 1996.

- 13 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2007.
- 14 Wilfrid Hodges. *Model Theory*. Cambridge University Press, Cambridge, March 1993.
- 15 Wojciech Kazana and Luc Segoufin. Enumeration of monadic second-order queries on trees. *ACM Trans. Comput. Log.*, 14(4):1–12, 2013.
- 16 Leonid Libkin. *Elements of finite model theory*. Springer Science & Business Media, 2013.
- 17 Tova Milo, Dan Suciu, and Victor Vianu. Typechecking for XML transformers. *Journal of Computer and System Sciences*, 66(1):66–97, 2003.
- 18 Andrzej Mostowski. On Direct Products of Theories. *J. Symb. Log.*, 17(01):1–31, 1952.
- 19 Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.
- 20 Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages, Volume 3: Beyond Words.*, pages 389–455. Springer, 1997.

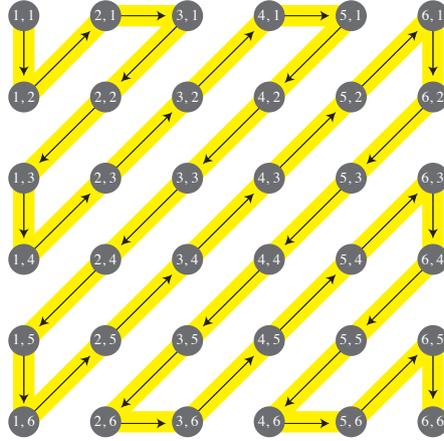
A Successor instead of order

In this appendix, we prove Theorem 4, which says that:

1. The class of successor-MSO string-to-string interpretations is not closed under composition, and strictly contains the class of (order-)MSO string-to-string interpretations.
2. The following is undecidable: given a successor-first-order string-to-string interpretation f and a regular language L over the output alphabet, is $f^{-1}(L)$ empty?

Proof of Theorem 4.

1. We first show item 1. Fix an input alphabet Σ , and consider the function $f: \Sigma^* \rightarrow (\Sigma \times \Sigma)^*$ which inputs a string, and outputs all pairs of positions (with the corresponding pairs of labels) in the order depicted by the following picture:



It is not hard to see that the function f is a successor-MSO string-to-string interpretation (in fact even first-order logic would be enough if the input string was equipped with a labelling indicating the parity of positions). Suppose that the alphabet Σ contains two endmarkers \vdash, \dashv , and consider an input word of the form $\vdash a_1 \cdots a_n \dashv$ where a_1, \dots, a_n are letters that are not endmarkers and the length n is even. In this case, the output contains the letter (\vdash, \dashv) exactly once, it contains the letter (\dashv, \vdash) also exactly once, and the word between these two letters is exactly:

$$(a_1, a_n), \dots, (a_n, a_1).$$

If and only if the word $a_1 \cdots a_n$ is a palindrome, then the above word contains only letters from the diagonal $\{(a, a) : a \in \Sigma\}$. Summing up, there is a regular (and therefore also MSO-definable) language $L \subseteq \Sigma^*$ such that

$$f(w) \in L \quad \text{if and only if} \quad w = \vdash v \dashv \text{ for some palindrome } v \text{ without } \vdash, \dashv. \quad (2)$$

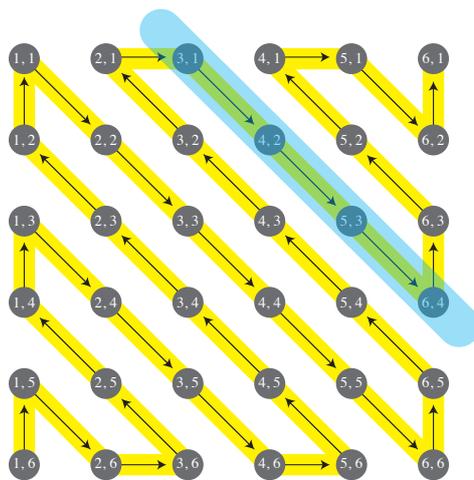
Define χ_L to be the *characteristic function* of L , i.e., the function from Σ^* to $\{0, 1\}$ which outputs 1 or 0 depending on whether the input belongs to L or not. We can view the characteristic function as a string-to-string function, where the output is in $\{0, 1\}^*$ and which happens to only produce outputs with one letter. The following claim is not hard to see.

▷ **Claim 18.** A language $L \subseteq \Sigma^*$ is regular if and only if its characteristic function is a successor-MSO string-to-string function.

From the claim, it follows that the characteristic function of the language L in (2) is in the successor-MSO class. If the class were closed under composition, then also $\chi_L \circ f$, the characteristic function of the palindrome language in (2), would be in successor-MSO, and thus by Claim 18 the palindrome language would be regular, a contradiction.

2. We now show item 2 of Theorem 4, i.e., that for a successor-first-order string-to-string interpretation f and a regular language L over the output alphabet, the emptiness of $f^{-1}(L)$ is undecidable. The proof is a standard reduction from the halting problem for Turing machines.

Let M be a Turing machine. Consider the string-to-string function f defined as in the previous item, except that the order on positions is as follows:



The key observation is that the output $f(a_1 \cdots a_n)$ contains, for every odd $i \in \{1, \dots, n\}$, an infix of the form

$$(a_i, a_1), (a_{i+1}, a_2), \dots, (a_n, a_{n-i+1}).$$

In the picture, the blue colouring indicates this infix for $i = 3$.

The above observation shows that the output of f can be used to compare infixes of w with other infixes; this can be used to check if an input word represents an accepting computation of the fixed Turing machine.

The input will be required to be of the following shape: $|c_1|c_2| \dots |c_n|$, where the c_i s are words that represent the consecutive configurations of an accepting computation of the Turing machine.

We mainly need to enforce two additional properties to obtain the reduction: first, that all the c_i s have the same size and second, that each c_{i+1} is the successor configuration of c_i (and also that c_1 is initial and c_n is final, which are simple regular properties). To enforce these two properties we only need to check properties of the infix $(a_i, a_1), (a_{i+1}, a_2), \dots, (a_n, a_{n-i+1})$ where i is the position of the second $|$ separator symbol of the input word. We can easily enforce that this position is odd by asking that all configurations are of even length.

◀

The proof of item 2 could be improved so that the function f is a successor-first-order string-to-string interpretation, which shows that emptiness of $f^{-1}(L)$ is undecidable already

when f is successor-first-order and L is regular. This shows that the class of successor-first-order string-to-string interpretations is not contained in the class of (ordered) first-order string-to-string interpretations considered in this paper, since by our main theorem, the latter class is contained in the class of polyregular functions, and emptiness of $f^{-1}(L)$ is decidable if L is regular and f is polyregular [3, Theorem 1.7].

B Proof of the Factorisation Forest Theorem

We provide a proof for the aperiodic variant of the Factorisation Forest Theorem (Theorem 15) here. Consider a surjective homomorphism

$$h: \Sigma^+ \rightarrow S.$$

We can assume without loss of generality that Σ is a subset of S . The proof is by induction on (a) the size of S ; (b) the size of Σ . The two parameters are ordered lexicographically, with (a) being more important.

When Σ has one element, then the blocks of a string $w \in \Sigma^+$ are simply its letters; this covers the induction base. The partition of a string into letters is clearly first-order definable.

For the induction step, suppose that Σ has more than one element. Take some $s \in \Sigma$ and consider the functions

$$\begin{array}{ccc} S & \rightarrow & S \\ t & \mapsto & ts \end{array} \quad \text{and} \quad \begin{array}{ccc} S & \rightarrow & S \\ t & \mapsto & st \end{array} \quad (3)$$

If one of these functions is surjective, then it is a permutation, and therefore it has to be the identity by aperiodicity of the semigroup. If both functions are surjective, then s must be the identity element of the semigroup (which might not exist in some semigroups). Since Σ has at least two elements, and there is at most one identity, there must be an $s \in \Sigma$ such that one of the functions in (3) is not surjective. Without loss of generality, assume that $t \mapsto ts$ is not surjective, and therefore $T := Ss$ is a proper subset of the semigroup S .

Consider the following two semigroup homomorphisms: the first one is the product operation

$$h_T: T^+ \rightarrow T$$

in the semigroup T , and the second one is

$$h_{\neq s}: (\Sigma - \{s\})^+ \rightarrow S$$

obtained by restricting h to the smaller alphabet. Both homomorphisms are smaller in our induction order: h_T uses a smaller semigroup, and $h_{\neq s}$ has a smaller alphabet. Therefore, the induction assumption can be applied to obtain both partitions into blocks and bounds M_T and $M_{\neq s}$ on the heights of the corresponding strings.

For a string $w \in \Sigma^+$, we define its partition into blocks with respect to the homomorphism h as follows by case analysis.

1. Suppose that w ends with s and does not begin with s . Decompose w as follows:

$$w = w_1 s^{k_1} \dots w_n s^{k_n} \quad w_1, \dots, w_n \in (\Sigma - \{s\})^+ \quad k_1, \dots, k_n \in \{1, 2, \dots\}.$$

- a. If $n = 1$, then the blocks are w_1 and s^{k_1} . The former word has height at most $M_{\neq s}$ by induction assumption and the latter word has height at most 2 because it uses only the letter s . It follows that w has height at most $M_{\neq s} + 2$.

- b. Otherwise $n > 1$. For $i \in \{1, \dots, n\}$ define t_i to be $h(w_i s^{k_i})$. Note that $t_i \in Ss = T$. Consider the partition into blocks of the word $t_1 \cdots t_n$ with respect to the homomorphism h_T . The blocks of w are the same as the blocks of $t_1 \cdots t_n$, except that in each block, the letter t_i is replaced with the corresponding infix $w_i s^{k_i}$. Since the height of $t_1 \cdots t_n$ is at most M_T from the induction assumption, and each $w_i s^{k_i}$ has height at most $M_{\neq s} + 2$ from item (1a), we obtain a height of at most $M_T + M_{\neq s} + 2$ for the word w .
2. We are left with the case when w either begins with s or does not end with s . In these cases, we simply decompose the word by shaving off the beginning and the end to reduce the decomposition to case (1).
 - a. If $w = usv$ such that u and v do not begin with s , and $v \in (\Sigma - \{s\})^+$ then we split w into two blocks, us and v . According to case (1), us has height at most $M_T + M_{\neq s} + 2$ and by induction assumption, v has height at most $M_{\neq s}$, thus overall, w has height at most $M_T + M_{\neq s} + 3$.
 - b. Finally, let $w = s^k usv$ with $k \in \{1, 2, \dots\}$, u, v not beginning with s , and $v \in (\Sigma - \{s\})^+$. In that case we split w into two parts again: s^k and usv , s^k has height at most 2, and from the previous case, we have a final height of at most $M_T + M_{\neq s} + 4$.

It is not hard to see the partition into blocks described above is first-order definable. This completes the proof of the Factorisation Forest Theorem.

C Proof of the Domination Lemma

This section is devoted to proving the Domination Lemma. The statement of the Domination Lemma in Section 4 was chosen so that it would be most easily applied to strings and their infixes. We begin by stating a more abstract version of the lemma, called the *Product Domination Lemma*, which is adapted to allow for a modular proof and implies the Domination Lemma in the shape in which we use it. Before stating the Product Domination Lemma, we introduce notation for the three kinds of product operations that are relevant to us.

1. Elements of the *direct product* $\prod_{i=1}^k \mathfrak{A}_i := \mathfrak{A}_1 \times \cdots \times \mathfrak{A}_k$ of structures $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ are tuples (a_1, \dots, a_k) with $a_i \in \mathfrak{A}_i$ for every $i \in \{1, \dots, k\}$. For every relation R in some \mathfrak{A}_i , there is a corresponding relation of the same arity in the direct product, which says whether or not R holds after projecting to the i -th coordinate.
2. The *k -th power of a structure \mathfrak{A}* is similar to the k -fold direct product of \mathfrak{A} , except that different coordinates can be compared, i.e., for every two tuples $(a_1, \dots, a_k), (a'_1, \dots, a'_k)$ in the k -th power of \mathfrak{A} and all $i, j \in \{1, \dots, k\}$, we can compare a_i and a'_j . One way of modelling such comparisons is to say that the k -th power is obtained from the k -fold direct product by adding for all $i, j \in \{1, \dots, k\}$ a function which swaps coordinates i and j .
3. The *ordered product* $\mathfrak{A}_1 \cdots \mathfrak{A}_k$ is obtained by taking the disjoint union of the structures $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ and adding an extra binary predicate \sqsubset , called the *block order*, such that $x \sqsubset y$ holds if x comes from an \mathfrak{A}_i and y comes from an \mathfrak{A}_j with $i < j$.

The Product Domination Lemma uses all three kinds of products: it considers a direct product of powers of ordered products.

Recall that we write \equiv_{r+k} for the equivalence relation on structures with distinguished elements of having the same rank $r + k$ type.

► **Lemma 19** (Product Domination Lemma). *Let $k, r \in \{1, 2, \dots\}$. Then there exists an $\omega_* \in \{1, 2, \dots\}$ such that the following holds. Let $I \subset \{0, 1, \dots\}$ be an initial segment of the*

natural numbers and let

$$\mathfrak{A} := \prod_{i \in I} \mathfrak{A}_i^{k_i} \quad \text{such that } k_i \leq k \text{ for all } i,$$

where each \mathfrak{A}_i is an ordered product

$$\mathfrak{A}_i = \mathfrak{A}_{i,1} \cdots \mathfrak{A}_{i,n_i} \quad \text{with } \mathfrak{A}_{i,1} \equiv_{r+k} \cdots \equiv_{r+k} \mathfrak{A}_{i,n_i}.$$

Let \prec be a linear order on \mathfrak{A} defined by a first-order formula of rank r , and let t be a unary rank ω_* type over \mathfrak{A} . Then there exist $d \in I$, $e \in \{1, \dots, k_d\}$, and $p \in \{-1, 1\}$ such that

$$x[d][e] \sqsubset^p y[d][e] \quad \text{implies } x \prec y \quad \text{for all } x, y \in \mathfrak{A} \text{ of type } t.$$

Proof overview. We begin by showing, in Section C.1, that the Product Domination Lemma implies the Domination Lemma in its original form from Section 4. The rest of Section C is then devoted to proving the Product Domination Lemma. This is done in four steps. In Section C.2, we show that if we can prove the Product Domination Lemma for some nonzero polarity other than $\{-1, 1\}$, then we can reduce the polarity down to $\{-1, 1\}$ at the cost of increasing the threshold ω . Next, we prove the Product Domination Lemma in four steps, which deal with special cases of increasing generality, as described below.

- In Section C.3 we prove domination for direct products of linear orders, i.e. structures

$$\mathfrak{A} = \prod_{i \in I} (\{1, \dots, n_i\}, <).$$

This can be viewed as the special case of the Product Domination Lemma when all k_i are 1, and furthermore all structures $\mathfrak{A}_{i,j}$ (called *blocks* in the proof) have size one.

- In Section C.4 we prove domination for powers of linear orders, i.e. structures

$$\mathfrak{A} = (\{1, \dots, n\}, <)^k.$$

This can be viewed as the special case of the Product Domination Lemma when I has size one, and all blocks have size one.

- In Section C.5 we prove the joint generalisation of the results from the two previous sections, i.e. we consider direct products of powers of linear orders:

$$\mathfrak{A} = \prod_{i \in I} (\{1, \dots, n_i\}, <)^{k_i}.$$

This can be viewed as the special case of the Product Domination Lemma when all blocks have size one.

- In Section C.6 we complete the proof of the Product Domination Lemma.

Compositionality. Before continuing with the proof, we state two compositionality properties of first-order logic with respect to products that will be heavily used in the proofs.

► **Theorem 20** ([18]). *The following holds for all $m, n, r \in \{1, 2, \dots\}$.*

1. *Consider structures $\mathfrak{A}_1, \dots, \mathfrak{A}_n, \mathfrak{B}_1, \dots, \mathfrak{B}_m$ over the same vocabulary. The rank r type of the ordered product*

$$\mathfrak{A}_1 \cdots \mathfrak{A}_n \mathfrak{B}_1 \cdots \mathfrak{B}_m$$

is determined by the rank r types of the two ordered products

$$\mathfrak{A}_1 \cdots \mathfrak{A}_n \quad \text{and} \quad \mathfrak{B}_1 \cdots \mathfrak{B}_m.$$

2. Let $\mathfrak{A} := \prod_{i \in I} \mathfrak{A}_i$ be a direct product of structures \mathfrak{A}_i . For every m -ary rank r type t in \mathfrak{A} and every $i \in I$ there is an m -ary rank r type $t[i]$ in the structure \mathfrak{A}_i such that for all $x_1, \dots, x_m \in \mathfrak{A}$

$$(x_1, \dots, x_m) \text{ has rank } r \text{ type } t \text{ in } \mathfrak{A} \quad \text{iff} \quad \text{for every } j, (x_1[j], \dots, x_m[j]) \text{ has type } t[j] \text{ in } \mathfrak{A}_j.$$

Continuous functions. Let $i \in \{0, 1, \dots\}$ and let \mathfrak{A} and \mathfrak{B} be relational structures over possibly different vocabularies. Let f be a function from (the universe of) \mathfrak{A} to (the universe of) \mathfrak{B} and for all $k \in \{1, 2, \dots\}$, denote by f_k the function mapping k -tuples of \mathfrak{A} to k -tuples of \mathfrak{B} by component-wise application of f . Then f is called *i -continuous* if for every $k \in \{1, 2, \dots\}$ and every subset of \mathfrak{B}^k defined by a formula in first-order logic with quantifier rank r , its inverse image under f_k can be defined in first-order logic via a formula with quantifier rank $r + i$. A function is called *continuous* if it is 0-continuous.

An alternative, equivalent characterisation, which we also use in this paper, is that a function is continuous if and only if it is type-preserving, i.e., it maps tuples of the same type to tuples of the same type.

C.1 Proof of the Domination Lemma

We begin by using the Product Domination Lemma to obtain the Domination Lemma in its statement from Section 4. Let $k, m, r \in \{1, 2, \dots\}$. Apply the Product Domination Lemma to k and r yielding some threshold value ω_* . Define

$$\omega := 2\omega_* + r + k + m. \tag{4}$$

We prove that ω satisfies the requirements of the Domination Lemma. Let w_1, \dots, w_n and \mathfrak{A} be as in the assumptions of the Domination Lemma. This means that \mathfrak{A} is the ordered product of the (ordered structures associated with) the strings w_1, \dots, w_n extended with the block order \sqsubset . Furthermore, the strings satisfy $w_i \equiv_{\omega} w_{i+1}$ with at most m exceptions. (The order on the blocks is \sqsubset , while the orders corresponding to the ordered structures are $<$). Let \prec be a linear order on \mathfrak{A}^k defined by a first-order formula of quantifier rank r and let t be a k -ary rank ω type over the vocabulary of \mathfrak{A} . We intend to find a dominating coordinate and a polarity that satisfy

$$x_d \sqsubset^p y_d \quad \text{implies} \quad (x_1, \dots, x_k) \prec (y_1, \dots, y_k) \quad \text{for all } \underbrace{x_1, \dots, x_k}_{\text{of type } t}, \underbrace{y_1, \dots, y_k}_{\text{of type } t} \text{ in } \mathfrak{A}.$$

Define \sim to be the coarsest equivalence relation on $\{1, \dots, n\}$ such that $i \sim i + 1$ holds whenever $w_i \equiv_{\omega_*} w_{i+1}$. Equivalence classes of \sim are intervals. Let \mathcal{I} be the set consisting of these equivalence classes. Since $\omega \geq \omega_*$, we know from the assumptions of the Domination Lemma that \mathcal{I} has at most m elements. For an equivalence class $I \in \mathcal{I}$, define $\mathfrak{B}_I \subseteq \mathfrak{A}$ to be the substructure obtained by restricting \mathfrak{A} to elements that come from w_i with $i \in I$. We can view \mathfrak{B}_I as an ordered product which only uses the words w_i with $i \in I$. By definition, in \mathfrak{B}_I , all blocks (i.e., all w_i) have the same rank ω_* type. For every $I \in \mathcal{I}$, there is a first-order formula which selects the elements from \mathfrak{B}_I inside the structure \mathfrak{A} : the formula counts the number of blocks w_i to the left which satisfy $w_i \not\equiv_{\omega_*} w_{i+1}$, and therefore it has quantifier rank at most $\omega_* + m$. For $\bar{x} \in \mathfrak{A}^k$ of rank ω type t and $I \in \mathcal{I}$, define

$$C_I := \{i \in \{1, \dots, k\} : \bar{x}[i] \text{ is in } \mathfrak{B}_I\}.$$

This set does not depend on \bar{x} once t has been fixed, because, as we have argued above, one can express the containment in \mathfrak{B}_I using a first-order formula with quantifier rank at most ω . Define

$$\iota: \prod_{I \in \mathcal{I}} \overbrace{\mathfrak{B}_I^{C_I}}^{\mathfrak{B}} \rightarrow \mathfrak{A}^k$$

to be the injection that is defined in the following way (where an element in the universe of \mathfrak{B}_I is seen as an element in the universe of \mathfrak{A}):

$$\iota(x)[i] := x[I][i], \quad \text{for } i \in \{1, \dots, k\}, \text{ and } i \in C_I.$$

The image of this injection contains all tuples in \mathfrak{A}^k that have k -ary rank ω type t . Since the injection sends tuple of the same type to tuples of the same type, it is continuous.

▷ **Claim 21.** All elements in the inverse image of t under ι have the same rank ω_* type.

Proof. Note that the continuity of ι is not useful for this result, because it only tells us that the inverse image of t is a union of rank ω types. The image $\iota(\mathfrak{B}) \subseteq \mathfrak{A}^k$ can be defined by a first-order formula of quantifier rank at most $\omega_* + r + k + m$. Therefore, if elements of \mathfrak{B} have different rank ω_* types, then their images under ι have different rank ω types. This proves the claim. ◀

By the above claim, all elements in the inverse image under ι of type t have the same rank ω_* type over \mathfrak{B} , call it $t_{\mathfrak{B}}$. Define $\prec_{\mathfrak{B}}$ to be the linear order on \mathfrak{B} which is the inverse image of \prec under ι , i.e.

$$x \prec_{\mathfrak{B}} y \quad \text{iff} \quad \iota(x) \prec \iota(y).$$

Since ι is continuous, it follows that $\prec_{\mathfrak{B}}$ is defined using a first-order formula of quantifier rank r . By the Product Domination Lemma, there exist $d \in \mathcal{I}$, $e \in C_d$ and $p \in \{-1, 1\}$ such that

$$x[d][e] \square^p y[d][e] \quad \text{implies} \quad x \prec_{\mathfrak{B}} y \quad \text{for all } x, y \text{ of type } t_{\mathfrak{B}} \text{ in } \mathfrak{B}.$$

By pulling this result forward across the injection ι , we get the corresponding conclusion for \bar{x} and \bar{y} in \mathfrak{A}^k of type t .

This finishes the proof of the Domination Lemma, assuming the Product Domination Lemma holds. The rest of this section is devoted to proving the Product Domination Lemma.

C.2 Polarity reduction

In the Product Domination Lemma, we use powers \square^p for polarities $p \in \{-1, 1\}$. This notation also makes sense for other nonzero integers p , for example $x \square^{-3} y$ means that $y \square z_1 \square z_2 \square x$ holds for some z_1, z_2 . (We extend this notation to other binary relations as well.) It would be easier to prove the Product Domination Lemma for polarities p with larger absolute values, since for $s \in \{-1, 1\}$ and for $p' \in \{1, 2, \dots\}$, the implication

$$x[d] \square^{s \cdot p'} y[d] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \in \mathfrak{A} \text{ of type } t$$

has a stronger assumption than $x[d] \square^s y[d]$ and is therefore weaker than

$$x[d] \square^s y[d] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \in \mathfrak{A} \text{ of type } t.$$

The following lemma shows that such weaker versions are indeed enough.

► **Lemma 22** (Polarity Reduction Lemma). *Let \mathfrak{A} be a relational structure, and let R and \prec be binary relations on \mathfrak{A} that are defined by first-order formulas of quantifier rank at most $r \in \{1, 2, \dots\}$. If \prec is transitive and antisymmetric, then for every $p \in \{1, 2, \dots\}$*

$$\begin{array}{ccc}
 R(x, y) \text{ implies } (x \prec y \vee y \prec x) & & \text{for all } x, y \in \mathfrak{A} \\
 & \wedge & \\
 R^p(x, y) \text{ implies } x \prec y & & \text{for all } x, y \in \mathfrak{A} \\
 & \Downarrow & \\
 R(x, y) \text{ implies } x \prec y & & \text{for all } x, y \in \mathfrak{A} \text{ with } x \equiv_{r+p} y.
 \end{array}$$

Proof. Let R and \prec be as in the assumptions and let $p \in \{1, 2, \dots\}$. Suppose the two conditions in the stated implication \Downarrow hold. Let $x, y \in \mathfrak{A}$ be such that $R(x, y)$ and $x \equiv_{r+p} y$ hold. We need to show $x \prec y$. Let t be the binary rank r type that describes the pair (x, y) . Since R is defined using quantifier rank at most r and contains (x, y) , it follows that all pairs of type t are contained in R . Because \prec is defined by a formula of quantifier rank r , our assumptions imply that the set of pairs of type t is contained in either \prec or \succ . To prove the lemma, we need to show that it is contained in \prec . Define a *chain* to be a sequence of elements

$$x_1, \dots, x_i \in \mathfrak{A} \quad \text{where } (x_1, x_2), \dots, (x_{i-1}, x_i) \text{ have type } t,$$

i.e. a walk in the directed graph on the universe of \mathfrak{A} where t is the edge relation. Note that every chain is either growing or decreasing with respect to \prec . We need to rule out the “decreasing” case. The property “there is a chain of length i that begins in x ” can be defined by a first-order formula of quantifier rank $r + i$. It follows inductively from $t(x, y)$ and $x \equiv_{r+p} y$ that there is a chain which begins in x and has length at least p . Indeed, if the maximal length of a chain beginning in y was some value $p' < p$, there would be a chain of length $p' + 1$ beginning in x since (x, y) has type t . This would violate that $x \equiv_{r+p} y$. ◀

As discussed at the beginning of this section, a corollary of the Polarity Reduction Lemma is that it is enough to prove a weaker version of the Product Domination Lemma, where the polarity p from the conclusion is in $\{-\omega_*, \omega_*\}$ instead of $\{-1, 1\}$. To see why, suppose that we have proved the version of the Product Domination Lemma with polarity $p' \in \{-\omega, \omega\}$, and we want to prove the version with polarity $p \in \{-1, 1\}$. Let t be a unary rank w_* type. By the weaker version of the Product Domination Lemma, there is some $p \in \{-1, 1\}$ such that

$$x[d] \sqsubset^{p \cdot \omega_*} y[d] \text{ implies } x \prec y \quad \text{for all } x, y \text{ in } \mathfrak{A} \text{ of type } t.$$

Apply the Polarity Reduction Lemma for $p := \omega_*$, the structure being \mathfrak{A} , and the relation R defined by

$$R(x, y) \text{ iff } x[d] \sqsubset^p y[d].$$

We obtain the conclusion of the Product Domination Lemma in its original form.

Thanks to the above reasoning, in the remaining sections it suffices to show variants of the Product Domination Lemma where the polarity p in the conclusion is some nonzero number with a fixed upper bound, not necessarily 1, on its absolute value.

C.3 Direct products of linear orders

In this section, we show the special case of the Product Domination Lemma for direct products of linear orders. A corollary is going to be that every first-order definable ordering \prec in a product of linear orders coincides with a lexicographic product of the underlying orders, at least when restricted to elements of the direct product that have the same type. The corollary is stated later in this section, but we begin with the underlying result about dominating coordinates.

► **Lemma 23.** *Let $r \in \{1, 2, \dots\}$ and let \prec be a linear ordering on a direct product*

$$\mathfrak{A} = \prod_{i \in I} (\{1, \dots, n_i\}, <) \quad \text{where } n_i > 6 \cdot 2^r \text{ for every } i \in I$$

such that \prec is defined by a first-order formula of quantifier rank r . Then for every unary rank r type t in \mathfrak{A} , there is a dominating coordinate $d \in I$ and $p \in \{-2 \cdot 2^r, 2 \cdot 2^r\}$ such that

$$x[d] <^p y[d] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \in \mathfrak{A} \text{ of type } t.$$

Note that in the above lemma, the polarity p can have a value not contained in $\{-1, 1\}$. As explained in Section C.2, at the cost of increasing the quantifier rank of the type t , the polarity can be reduced to values in $\{-1, 1\}$.

To prove Lemma 23, we use the following observation, which expresses that first-order logic formulas with quantifier rank r can only measure distances up to 2^r in a linear order. Its proof is the same as for [16, Theorem 3.6].

► **Lemma 24 (Threshold Lemma).** *Let $r \in \{1, 2, \dots\}$ and consider two k -tuples x and y in a linearly ordered set*

$$(\{1, \dots, n\}, <)$$

Then x and y have the same rank r type if and only if they have the same quantifier-free type in the structure extended with relations $<^i$ for $i \in \{1, \dots, 2^r\}$ and unary relations \min and \max .

Proof of Lemma 23. The proof proceeds by induction on the size of the set I , i.e. on the dimension of the product. Let t be a unary rank r type over the vocabulary of \mathfrak{A} . Consider its projections $t[i]$ for $i \in I$ as in Theorem 20. By the Threshold Lemma, if an $|I|$ -tuple $x \in \mathfrak{A}$ has type t , then $t[i]$ determines the distance of $x[i]$ from the first and last positions in $\{1, \dots, n_i\}$, measured up to threshold 2^r . If the distance from either the first or last position is $< 2^r$, then the value of $x[i]$ is fixed by the type t . For such types, we can eliminate one coordinate, and obtain the result by using the induction assumption. We are left with the case when t expresses that all coordinates are at least 2^r positions away from both the first and last positions.

Choose a tuple $x \in \mathfrak{A}$ such that for every $i \in I$, coordinate $x[i]$ is at least $3 \cdot 2^r$ positions away from the first and last positions in $\{1, \dots, n_i\}$, which can be achieved by the assumption that $n_i \geq 6 \cdot 2^r$ for all $i \in I$. We say that $\delta \in \mathbb{Z}^I$ is *small* if for every $i \in I$, the absolute value of $\delta[i]$ is between 2^r and $2 \cdot 2^r$. By the choice of x , we know that if δ is small, then $x + \delta$ has the same type as x . Define the *sign vector* of δ to be the subset of I which contains the coordinates on which δ is positive, and define

$$\mathcal{I} := \{\text{sign vector of } \delta : \delta \text{ is small and } x \prec x + \delta\} \subseteq 2^I.$$

It is not hard to see that the family \mathcal{I} is closed under set union, and the same is true for its complement.

▷ **Claim 25.** Consider a partition of the powerset 2^I into two families of sets, both of which are closed under union. Then there is a $d \in I$ such that one of the families is $\{J \subseteq I : d \in J\}$.

Proof. Since both families are closed under union, it follows from De Morgan's law that both families are closed under intersection. One of the families does not contain the empty set, call this family \mathcal{I} . Since \mathcal{I} is closed under intersection, it follows that the intersection $\cap \mathcal{I}$ of all sets in \mathcal{I} is nonempty. The intersection $\cap \mathcal{I}$ cannot have more than one element, because otherwise it could be decomposed as a union of two sets outside \mathcal{I} , and therefore it would be outside \mathcal{I} . Hence, the intersection of all sets in \mathcal{I} is a singleton $\{d\}$ for some $d \in I$. This means that all sets in \mathcal{I} contain d . It follows that for every $i \neq d$, the singleton $\{i\}$ must belong to the complement of \mathcal{I} . Since this complement is closed under taking unions, every set that does not contain d belongs to the complement of \mathcal{I} . ◀

An application of the claim yields a coordinate $d \in I$ such that either \mathcal{I} or its complement consists in exactly the sets that contain d . By symmetry, we may assume the first case. By unfolding the definition of \mathcal{I} , it follows that incrementing $x[d]$ by at least 2^r and at most $2 \cdot 2^r$ and modifying all other coordinates by any number with absolute value between 2^r and $2 \cdot 2^r$ yields a bigger tuple. Performing this procedure twice allows us to modify the coordinates other than d by any value in $\{-2^r, \dots, 2^r\}$, and therefore the result follows using the Threshold Lemma. ◀

We end this section with an interesting consequence of Lemma 23. Define a *lexicographic ordering* on

$$\mathfrak{A} = \prod_{i \in I} (\{1, \dots, n_i\}, <)$$

to be an ordering that is the lexicographic product, under some ordering of I , of orderings in the coordinates that are either $<$ or $>$. If I has n elements, then there are $n! \cdot 2^n$ lexicographic orderings, since the coordinates can be ordered in $n!$ ways and for each ordering one can use $<$ or $>$ for each of the n coordinates. By iteratively applying Lemma 23 and then using the Polarity Reduction Lemma, we can infer the following result.

▶ **Corollary 26.** *For every $r \in \{1, 2, \dots\}$, there is a threshold $\omega \in \{1, 2, \dots\}$ with the following property. Let \prec be a linear ordering on the product*

$$\mathfrak{A} = \prod_{i \in I} (\{1, \dots, n_i\}, <)$$

such that \prec is defined by a first-order formula of quantifier rank r . For every unary rank r type t in \mathfrak{A} , the order \prec restricted to tuples of type t coincides with one of the lexicographic orderings on \mathfrak{A} .

C.4 Powers of linear orders

In this section, we prove a version of the Domination Lemma that considers powers of finite linear orders. The difference to the scenario treated in Section C.3 is that here we can compare different coordinates.

▶ **Lemma 27 (Linear Domination Lemma).** *For all $k, r \in \{1, 2, \dots\}$, there exists a threshold $\omega \in \{1, 2, \dots\}$ such that the following holds. If \prec is a linear ordering on \mathfrak{A}^k with*

$$\mathfrak{A} = (\{1, \dots, n\}, <) \quad \text{and} \quad n > \omega$$

such that \prec is defined by a first-order formula of rank r , then for every k -ary rank ω type t , there are $d \in \{1, \dots, k\}$ and $p \in \{-\omega, \omega\}$ such that

$$x[d] <^p y[d] \text{ implies } x \prec y \quad \text{for all } x, y \in \mathfrak{A}^k \text{ of type } t.$$

In the proof, we do a detailed case analysis of the expressive power of first-order logic on linear orderings, which relies on the Threshold Lemma.

For $m \in \{1, 2, \dots\}$, a tuple $x \in \mathfrak{A}^k$ is called m -separated if for all $i \neq j$ in $\{0, \dots, k+1\}$ it holds that $x[i] <^m x[j]$ or $x[i] <^{-m} x[j]$, with the convention that $x[0] = \min$ and $x[k+1] = \max$. We can extend the notion of being separated to types. A rank r type t is called m -separated if every tuple of type t is m -separated, and *separated* if every tuple of type t is 2^r -separated.

▷ **Claim 28 (Linear Domination Lemma - Separated Case).** For all $k, r \in \{1, 2, \dots\}$, there exists a threshold $\omega \in \{1, 2, \dots\}$ such that the following holds. If \prec is a linear order on \mathfrak{A}^k , where

$$\mathfrak{A} = (\{1, \dots, n\}, <) \quad \text{and} \quad n > \omega$$

such that \prec is defined by a first-order formula of rank r , then for every separated rank ω type t , there are $d \in \{1, \dots, k\}$ and $p \in \{-\omega, \omega\}$ such that

$$x[d] <^p y[d] \text{ implies } x \prec y \quad \text{for all } x, y \in \mathfrak{A}^k \text{ of type } t.$$

We first show that the separated version of the lemma implies the general version.

Proof of the Linear Domination Lemma. The proof proceeds by induction on the dimension k . For $k = 1$, the statement holds by the Threshold Lemma. Let $k \geq 2$ and $r \in \{1, \dots\}$ and assume that the Linear Domination Lemma holds for dimension $k - 1$. Let ω_1 be the value obtained using Claim 28 for dimension k and quantifier rank r . Let ω_2 be the value obtained using the induction hypothesis for arity $k - 1$ and quantifier rank $\omega_1 + 2$. Let $\omega := \max\{\omega_1 + 2, \omega_2\}$.

Let $\mathfrak{A} := (\{1, \dots, n\}, <)$ for an $n > \omega$ and consider a rank ω type t of arity k . Let s be the rank ω_1 type associated with t . Note that all tuples of type t have type s but the converse does not necessarily hold. By the Threshold Lemma, the type s is entirely given by quantifier-free formulas using $<^i$ for $i \in \{1, \dots, 2^{\omega_1}\}$. If s is separated, by Claim 28, the result holds for tuples of type s , and thus in particular for tuples of type t . Otherwise, if there is a $\delta \in \{0, \dots, 2^{\omega_1} - 1\}$ such that $x[0] = \min$ and $x[k+1] = \max$ are exactly δ apart, then all coordinates are pairwise at most δ apart and thus, s (and therefore also t) fixes all coordinates, such that the conclusion from the lemma trivially holds. Thus, assume that there are $i \in \{1, \dots, k\}$, $j \in \{0, \dots, k+1\} \setminus \{i\}$ and $\delta \in \{0, \dots, 2^{\omega_1} - 1\}$ such that $x[i]$ and $x[j]$ are exactly δ apart. Without loss of generality, we assume $i = k$, $x[j] <^\delta x[k]$ and $x[j] \not<^{\delta+1} x[k]$.

Let $\pi: \mathfrak{A}^k \rightarrow \mathfrak{A}^{k-1}$ be the projection to the first $k-1$ coordinates. Define a linear ordering \prec_{k-1} over the tuples of \mathfrak{A}^{k-1} which are images of tuples of type s with respect to π , and such that $x \prec y \Leftrightarrow \pi(x) \prec_{k-1} \pi(y)$ for all x and y of type s . This order can be defined using a formula of quantifier rank $\omega_1 + 2$, simply by existentially quantifying over the missing coordinates.

Moreover, by continuity of π , all tuples of ω -type t are mapped to tuples of ω -type t' . By the induction hypothesis, we know that there are $d \in \{1, \dots, k-1\}$ and $p \in \{-\omega_2, +\omega_2\}$ such that:

$$x[d] <^p y[d] \text{ implies } x \prec_{k-1} y \quad \text{for all } x, y \in \mathfrak{A}^{k-1} \text{ of type } t'.$$

Thus, we have in particular:

$$x[d] <^p y[d] \text{ implies } x \prec y \quad \text{for all } x, y \in \mathfrak{A}^k \text{ of type } t,$$

which concludes the proof. \blacktriangleleft

The proof of Claim 28 has three stages, depending on whether the arity k is 1, 2 or bigger. The case $k = 1$ is actually trivial, and the most interesting case is arity $k = 2$.

Arity two

We first prove Claim 28 for $k = 2$. We need to show that there are $d \in \{1, 2\}$, $\omega > 0$ and $p \in \{-\omega, \omega\}$ such that for every separated type t of arity two (we use the name *binary* from now on) and rank r , we have:

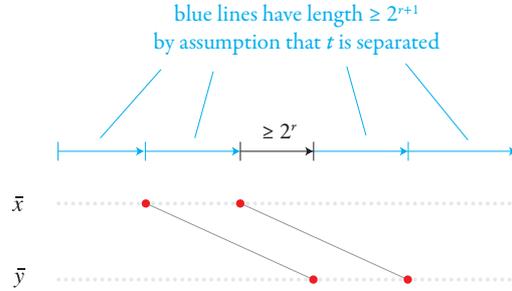
$$x[d] <^p y[d] \text{ implies } \bar{x} \prec \bar{y} \quad \text{for all } \bar{x}, \bar{y} \in \{1, \dots, n\}^2 \text{ of type } t.$$

For two pairs x, y , we say that they are ω -distant if for $i \in \{1, 2\}$, it holds that $x[i] <^{\pm\omega} y[i]$ or $x[i] = y[i]$. We first show the following claim.

\triangleright **Claim 29.** If the statement from Claim 28 holds in the case of ω -distant tuples, then it holds for all tuples.

Proof. This is shown in exactly the same way as the Polarity Reduction Lemma. If we have two tuples x, y of the same type of sufficiently high rank, then we can ensure that there is a sufficiently long sequence x_0, \dots, x_ℓ , such that the r -type of (x, y) is the same as the one of (x_{i-1}, x_i) , for $i \in \{1, \dots, \ell\}$. If the sequence is long enough, then x_0, x_ℓ are ω -distant. \blacktriangleleft

Let $\omega := 2^r$. Consider tuples x (first row) and y (second row) where the first coordinate of y is at least 2^r larger than the second coordinate of x , as in the following picture.



By the Threshold Lemma, the order relationship $x \prec y$ does not depend on the choice of x and y (subject to the requirements in the picture above). There are two cases, namely

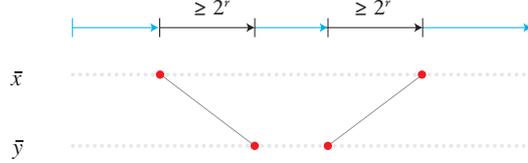
$$\underbrace{x \prec y}_{A1} \quad \underbrace{x \succ y}_{A2}.$$

The cases are symmetric, we assume A1 without loss of generality.

Consider the case as above with $x[1] <^\omega x[2] \leq y[1] <^\omega y[2]$, but the distance between $x[2]$ and $y[1]$ is $\delta < 2^r$. We use a simplified version of the Polarity Reduction Lemma in the case of a linear order. Let $z := (y[2] + \delta, y[2] + \delta + \omega)$, hence the r -type of (x, y) is equal to the one of (y, z) . Then x and z are in situation A1, which means by the transitivity of \prec that $x \prec y$.

Consider the case with $x[1] <^\omega y[1] < x[2] <^\omega y[2]$. By the Threshold Lemma, we can assume that the distance δ between $x[2]$ and $y[1]$ is at most 2^r . Let $z := (y[2] - \delta, y[2] - \delta + \omega)$, hence the r -type of (x, y) is equal to the one of (y, z) . By transitivity of \prec , we have $x \prec y \Leftrightarrow x \prec z$. Now we have that $z[1] - x[2] = y[2] - \delta - x[2] \geq 0$ since $x[2] <^\omega y[2]$. Using the Threshold Lemma, we can assume that $x[1] <^\omega x[2] \leq z[1] <^\omega z[2]$, which means, according to the previous paragraph, that $x \prec y$.

Since we only compare distant separated tuples, the only remaining case is the one illustrated below. Consider now two tuples x and y that are related as follows:



Again there are two cases, namely

$$\underbrace{x \prec y}_{\text{B1}} \quad \text{and} \quad \underbrace{x \succ y}_{\text{B2}}.$$

Case B1 implies that the dominating coordinate is the first one and Case B2 the second.

General arity

In this section we complete the proof of the Separated Linear Domination Lemma. The main idea is that it suffices to compare tuples which differ in at most two coordinates.

Let \prec be a linear order on k -tuples in $\{1, \dots, n\}$ that is defined by a first-order formula of quantifier rank r . Let t be a separated k -ary rank r type and let $\omega = 2^r$. We prove that there is a dominating coordinate $d \in \{1, \dots, k\}$ and a $p \in \{-\omega, \omega\}$ such that

$$x[d] <^p y[d] \quad \text{implies} \quad x \prec y \quad \text{for all } \bar{x}, \bar{y} \in \{1, \dots, n\}^k \text{ of type } t. \quad (5)$$

Choose distinct coordinates $i, j \in \{1, \dots, k\}$ and let z be a tuple of type t . Define

$$T_{ij}^z := \{x \in \{1, \dots, n\}^k : x \text{ has type } t \text{ and agrees with } z \text{ on all coordinates except for possibly } i, j\}.$$

By the case of arity two, we have the following result:

▷ **Claim 30.** For every z and distinct coordinates i, j there is a dominating coordinate $d \in \{i, j\}$ and a polarity $p \in \{-\omega, \omega\}$ such that

$$x[d] <^p y[d] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \in T_{ij}^z \text{ of type } t.$$

Proof. Without loss of generality, we assume $z[1] \leq \dots \leq z[k]$. Let $i < j$. Since $T_{ij}^z = T_{ji}^z$, we can assume without loss of generality that $i < j$. We make a case analysis depending on whether $i+1 = j$ holds or not. Assume $i+1 = j$, and let $\mathfrak{B} := \{z[i-1] + 1, \dots, z[j+1] - 1\}$. We define the partial function π from tuples of type t of \mathfrak{A}^k to pairs of elements in the universe of \mathfrak{B} by keeping only the coordinates i and j . We also consider the function $\sigma : \mathfrak{B}^2 \rightarrow \mathfrak{A}^k$, which just fills in the missing coordinates by the coordinates of z . Note that the function σ is continuous, hence $\prec_{\mathfrak{B}} = \sigma^{-1}(\prec)$ can be defined by a formula of quantifier rank r . Moreover,

π is also continuous, thus the tuples in the image of π have the same r -type t' . Therefore, applying the Linear Domination Lemma to the case of dimension 2, we obtain that there are $d \in \{1, 2\}$, $\omega \in \{1, 2, \dots\}$, and $p \in \{-\omega, +\omega\}$ such that:

$$x[d+1] <^p y[d+1] \quad \text{implies} \quad x \prec_{k-1} y \quad \text{for all } x, y \in \mathfrak{B}^2 \text{ of type } t'.$$

Thus, we have in particular:

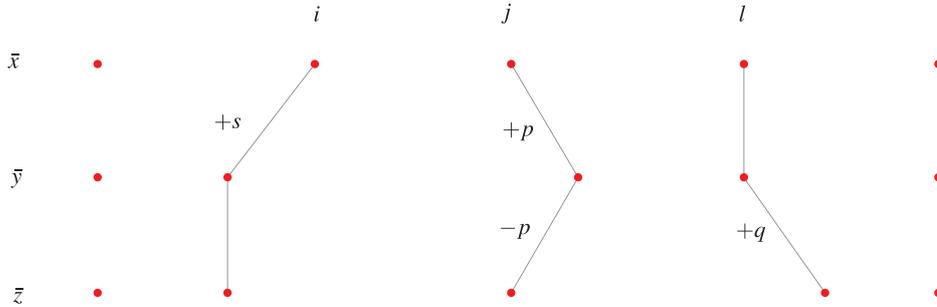
$$x[d+i] <^p y[d+i] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \in T_{ij}^z \text{ of type } t.$$

Similarly, if $i+1 < j$, we define the structure $\mathfrak{B} := \{z[i-1]+1, \dots, z[i+1]-1\} \times \{z[j-1]+1, \dots, z[j+1]-1\}$. Using the same arguments and Lemma 23, we obtain the result. \blacktriangleleft

It is not hard to see that there is exactly one possibility for d and p – once z , i and j have been fixed – since otherwise we would get a cycle for the order \prec . By the Threshold Lemma, the dominating coordinate d depends only on i, j and not on the choice of z , and therefore we can write d_{ij} for the dominating coordinate that is appropriate to coordinates i, j . Also the polarity p depends only on i and j . Let us write $i \xrightarrow{p} j$ if, whenever the values at coordinates i, j are distinct, then j is the dominating coordinate for i, j and the associated polarity is p .

\triangleright **Claim 31.** If i, j, ℓ are distinct, then $i \xrightarrow{p} j \xrightarrow{q} \ell$ implies $i \xrightarrow{q} \ell$.

Proof. Choose $s \in \{-\omega, \omega\}$ arbitrarily. Consider a tuple x which is (2^{r+1}) -separated. This tuple has type t , and shifting any coordinate by offset in $\{-\omega, \omega\}$ still leads to a tuple that has type t , because the quantifier rank of t is r . Define y to be the tuple obtained from x by adding s to coordinate i and adding p to coordinate i , and define z to be the tuple obtained from x by adding s to coordinate i and adding q to coordinate ℓ . Here is a picture where $p = q = \omega$ and $s = -\omega$.



From the assumption of the claim it follows that $x \prec y \prec z$, and this holds regardless of the choice of s . It follows that $i \xrightarrow{q} \ell$. \blacktriangleleft

From the above lemma it follows that the relation $i \rightarrow j$ defined by

$$i = j \quad \text{or} \quad i \xrightarrow{p} j \text{ for some } p \in \{-\omega, \omega\}$$

is a linear order on the coordinates $\{1, \dots, k\}$. Let d be the maximal element according to this total order. Let d' be the second-to-maximal element in the total order, and let p be such that $d' \xrightarrow{p} d$ holds. From Claim 31 it follows that $i \xrightarrow{p} d$ holds for all $i \in \{1, \dots, k\} \setminus \{d\}$.

We show that coordinate d dominates when comparing tuples where the values of coordinate d are sufficiently far apart.

To finish the proof of the Linear Domination Lemma, we prove below (5) for polarity $2pk$, i.e., we show

$$x[d] <^{2pk} y[d] \text{ implies } x \prec y \quad \text{for } x, y \in \mathfrak{A}^k \text{ of type } t.$$

Assume without loss of generality that the coordinates in some (equivalently, every) tuple of type t are ordered so that they are strictly increasing. Let x and y be tuples as in the assumptions of the claim.

Let x and y be such that $x[d] <^{2pk} y[d]$. We need to show $x \prec y$. By the Threshold Lemma, we can assume that all coordinates in the tuples x and y avoid the first and last $k \cdot 2^\omega$ positions. To prove $x \prec y$, we will find a chain of $2k$ tuples that begins in x , ends in y , and is growing with respect to \prec . We do the proof in the case where $k = 7$ and $d = 4$, but the general case works the same. Define

$$z_1 = 2^\omega \quad z_2 = 2 \cdot 2^\omega \quad z_3 = 3 \cdot 2^\omega \quad z_5 = n - 3 \cdot 2^\omega \quad z_6 = n - 2 \cdot 2^\omega \quad z_7 = n - 2^\omega.$$

In general, z_i is defined as $i \cdot 2^\omega$ when $i \neq d$ and otherwise it is defined as $n - (k - i + 1) \cdot 2^\omega$. The choice of coordinates z_i is made so that they are far apart, and furthermore z_i is to the left/right of the tuples x, y , depending on whether $i < d$ or $i > d$. The chain that witnesses $x \prec z$ is given below:

$$\begin{aligned} x_1 &= (x_1, x_2, x_3, x_4, x_5, x_6, x_7) \\ x_2 &= (z_1, x_2, x_3, x_4 + p, x_5, x_6, x_7) \\ x_3 &= (z_1, z_2, x_3, x_4 + 2p, x_5, x_6, x_7) \\ x_4 &= (z_1, z_2, z_3, x_4 + 3p, x_5, x_6, x_7) \\ x_5 &= (z_1, z_2, z_3, x_4 + 4p, z_5, x_6, x_7) \\ x_6 &= (z_1, z_2, z_3, x_4 + 5p, z_5, z_6, x_7) \\ x_7 &= (z_1, z_2, z_3, x_4 + 6p, z_5, z_6, z_7) \\ y_7 &= (z_1, z_2, z_3, y_4 - 6p, z_5, z_6, z_7) \\ y_6 &= (z_1, z_2, y_3, y_4 - 5p, z_5, z_6, z_7) \\ y_5 &= (z_1, y_2, y_3, y_4 - 4p, z_5, z_6, z_7) \\ y_4 &= (y_1, y_2, y_3, y_4 - 3p, z_5, z_6, z_7) \\ y_3 &= (y_1, y_2, y_3, y_4 - 2p, y_5, z_6, z_7) \\ y_2 &= (y_1, y_2, y_3, y_4 - p, y_5, y_6, z_7) \\ y_1 &= (y_1, y_2, y_3, y_4, y_5, y_6, y_7) \end{aligned}$$

All tuples in the above chain have type t , by the assumption that coordinates z_i are far apart and to the left/right of the tuples x, y . Since the dominating coordinate is incremented as the chain progresses, and at most two coordinates change in each step, we can use the assumption that coordinate d dominates when only two coordinates change to conclude that each consecutive step yields a tuple that is bigger with respect to \prec . By transitivity, it follows that $x = x_1 \prec y_1 = y$.

C.5 Direct products of powers of linear orders

In this section we prove the most general version of the Product Domination Lemma for linear orderings, namely the case of direct products of powers of linear orderings.

► **Lemma 32.** *For all $k, r \in \{1, 2, \dots\}$, there exists a threshold $\omega \in \{1, 2, \dots\}$ such that the following holds. If \prec is a linear order on*

$$\mathfrak{A} = \prod_{i \in I} (\{1, \dots, n_i\}, <)^{k_i}$$

such that \prec is defined by a first-order formula of rank r , then for every unary rank ω type t over \mathfrak{A} , there are $d \in I$, $e \in \{1, \dots, k_i\}$ and $p \in \{-\omega, \omega\}$ such that

$$x[d][e] <^p y[d][e] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \in \mathfrak{A} \text{ of type } t.$$

Let ω be a threshold that is large enough – we will specify the required bounds during the proof. Fix for the rest of this proof a unary rank ω type t in \mathfrak{A} . Our goal is to find a dominating coordinate (d, e) and a polarity p as in the statement of Lemma 32.

The general strategy is as follows. We begin in Section C.5.0.1 by looking, for every $i \in I$, at the projection

$$\pi_i: \mathfrak{A} \rightarrow \overbrace{(\{1, \dots, n_i\}, <)^{k_i}}^{\mathfrak{A}_i} \quad x \mapsto x[i].$$

We show that there exists $\sigma_i: \mathfrak{A}_i \rightarrow \mathfrak{A}$ which is a *section* of π_i in the sense that $\pi_i \circ \sigma_i$ is the identity on \mathfrak{A}_i . By applying the results from Section C.4 about powers of linear orders, we show that there is a dominating coordinate d_i which works for elements in the image of the section. Next, in Section C.5.0.2, we consider the projection

$$\pi: \mathfrak{A} \rightarrow \prod_{i \in I} \overbrace{\{1, \dots, n_i\}}^{\mathfrak{B}} \quad x \mapsto (x[d_i])_i,$$

which is defined in terms of the dominating coordinates $\{d_i\}_{i \in I}$ that were found in Section C.5.0.1. Again, we find a section $\sigma: \mathfrak{B} \rightarrow \mathfrak{A}$. By applying the results from Section C.3 about direct products of linear orders, we find a dominating coordinate $d \in I$ which works for elements in the image of the section. Finally, in Section C.5.0.3, we combine the results about the sections σ_i and σ to prove the conclusion of Lemma 32 for $e = d_i$ where $i = d$.

C.5.0.1 Sections of π_i

For each $i \in I$, apply Lemma 27 about domination for powers of linear orders to r and k_i , leading to some threshold ω_i . Define

$$\omega_* := \max(\{r\} \cup \{\omega_i\}_{i \in I}).$$

Our first condition on the threshold ω is that

$$\omega \geq \omega_*. \tag{6}$$

Let t_* be the information of rank ω_* that is stored in the rank ω type t , i.e. t_* is the unique rank ω_* type contained in type t .

Let $i \in I$. The general idea in this part of the proof is to study the order \prec when comparing elements of \mathfrak{A} that agree on all coordinates other than i . Consider the projection

$$\pi_i: \mathfrak{A} \rightarrow \mathfrak{A}_i.$$

For $z \in \mathfrak{A}$ of type t_* define

$$\sigma_i^z: \mathfrak{A}_i \rightarrow \mathfrak{A}$$

to be the function which fills in the missing coordinates $j \in I - \{i\}$ by the values used in z . This function is a section of π_i in the sense that $\pi_i \circ \sigma_i^z$ is the identity on \mathfrak{A}_i . Consider the preimage of \prec under this section, i.e. the relation \prec_i^z defined by

$$x \prec_i^z y \quad \text{if} \quad \sigma_i^z(x) \prec \sigma_i^z(y).$$

By Theorem 20, σ_i^z is continuous, and therefore \prec_i^z is defined by a first-order formula of same quantifier rank as \prec , namely r . Therefore, since the type t_* has rank at least ω_i as obtained from Lemma 27, it follows that there are a polarity $p_i \in \{-\omega_i, \omega_i\}$ and a dominating coordinate $d_i \in \{1, \dots, k_i\}$ such that

$$x[d_i] \prec^{p_i} y[d_i] \quad \text{implies} \quad x \prec_i^z y \quad \text{for all } x, y \in \mathfrak{A}_i \text{ of type } t_*[i].$$

By Theorem 20, if z and z' have the same type of rank r then \prec_i^z and $\prec_i^{z'}$ are the same order. Therefore, since the quantifier rank of t_* is at least r , it follows that the dominating coordinate d_i and polarity p_i do not depend on z , as long as it has type t_* . Because σ_i^z is a section of π_i and it preserves the appropriate orderings, it follows that

$$x[i][d_i] \prec^{p_i} y[i][d_i] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \text{ in the image of } \sigma_i^z \text{ with type } t_*[i].$$

By applying the above to $z = x$, we see that coordinate d_i dominates whenever x, y agree on coordinates other than i :

$$x[i][d_i] \prec^{p_i} y[i][d_i] \quad \text{implies} \quad x \prec y \quad \begin{array}{l} \text{for all } x, y \in \mathfrak{A} \text{ of type } t_* \\ \text{such that } x[j] = y[j] \text{ for } j \neq i. \end{array} \quad (7)$$

C.5.0.2 Section of π

As announced in the proof strategy, we now consider the projection which uses only the dominating coordinates d_i that were found in Section C.5.0.1:

$$\pi: \mathfrak{A} \rightarrow \overbrace{\prod_{i \in I} (\{1, \dots, n_i\}, \prec)}^{\mathfrak{B}} \quad x \mapsto (x[d_i])_i.$$

We will find a suitable section σ of π and prove that there is a dominating coordinate for the image of that section.

Recall the type t_* discussed in Section C.5.0.1, which is obtained by keeping only the rank ω_* information from the type t . Define

$$\sigma: \mathfrak{B} \rightarrow \mathfrak{A}$$

to be the section of π that is defined by

$$x \in \mathfrak{B} \quad \mapsto \quad \begin{cases} \prec\text{-least element of } \pi^{-1}(x) \text{ with type } t_* & \text{if there is such an element} \\ \prec\text{-least element of } \pi^{-1}(x) & \text{otherwise.} \end{cases}$$

We argue why the section σ is $(\omega_* + 2k + r)$ -continuous. Consider a subset S of the universe of structure \mathfrak{A} defined by a unary formula of some quantifier rank q . We want to show

that the set of elements of \mathfrak{B} that are sent to S satisfy some formula of rank $q + \omega_* + 2k + r$. This amounts to showing that, for a unary rank q type t of \mathfrak{A} , there is a unary formula ϕ over \mathfrak{B} which selects elements whose image with respect to σ has type t . Using Theorem 20, the formula ϕ can be defined by quantifying over the missing coordinates and then checking that the whole tuple satisfies t , t_* if possible, and that it is the minimal one to do so. Thus we obtain a formula of quantifier rank $q + \omega_* + 2k + r$ (the 2 comes from the fact that we need to check minimality).

Define $\prec_{\mathfrak{B}}$ as the inverse image of \prec with respect to σ . From continuity, it follows that $\prec_{\mathfrak{B}}$ is defined by a first-order formula of quantifier rank at most $\omega_* + 2k + 2r$. Apply Lemma 23 to this quantifier rank, yielding some threshold $\omega_{\mathfrak{B}}$. We assume that

$$\omega \geq \omega_{\mathfrak{B}}. \quad (8)$$

Since the projection π is continuous, it follows that all elements of type t in \mathfrak{A} are mapped by π to elements of the same rank ω type, call it $t_{\mathfrak{B}}$. By Lemma 23 and the assumption (8), there are a dominating coordinate $d \in I$ and a polarity $p_{\mathfrak{B}} \in \{-\omega_{\mathfrak{B}}, \omega_{\mathfrak{B}}\}$ such that

$$x[d] \prec^{p_{\mathfrak{B}}} y[d] \quad \text{implies} \quad x \prec_{\mathfrak{B}} y \quad \text{for all } x, y \in \mathfrak{B} \text{ of type } t_{\mathfrak{B}}.$$

Define e to be d_i for $i = d$. Because σ is a section and it preserves the appropriate orderings, it follows that

$$x[d][e] \prec^{p_{\mathfrak{B}}} y[d][e] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \text{ of type } t \text{ in the image of } \sigma. \quad (9)$$

C.5.0.3 Proof of Lemma 32

We now complete the proof of Lemma 32. Define $p := 2p_i + p_{\mathfrak{B}}$. Let $x, y \in \mathfrak{A}$ have type t and assume that

$$x[d][e] \prec^p y[d][e]. \quad (10)$$

To prove $x \prec y$, as required in the conclusion of Lemma 32, we will find an \prec -ascending chain which begins in x , ends in y , and such that each step in the ascending chain is proved using the results from Sections C.5.0.1 and C.5.0.2.

▷ **Claim 33.** There is an x' of type t in the image of σ such that $x \prec x'$ and

$$x'[i][d_i] = x[i][d_i] + p_i \quad \text{for all } i \in I.$$

Proof. We say that x is *canonical on coordinate* $i \in I$ if $x[i] = z[i]$ for some z in the image of σ . By Theorem 20, if x is canonical on all coordinates $i \in I$, then it is in the image of σ . Therefore, we can prove the claim by induction on the number of coordinates $i \in I$ on which x is canonical, and in the induction step we can make a single coordinate i canonical, at the cost of shifting $x[i][d_i]$ by p_i positions, thanks to (7). ◀

Apply Claim 33 to x , yielding some x' of type t with $x \prec x'$. Apply a symmetric result to y , yielding some y' of type t with $y' \prec y$ and

$$y'[i][d_i] = y[i][d_i] - p_i \quad \text{for all } i \in I.$$

By definition of p and the assumption (10), we see that

$$x'[i][d_i] \prec^{p_{\mathfrak{B}}} y'[i][d_i] \quad \text{for } i = d$$

and therefore (9) can be applied to conclude $x' \prec y'$, and thus also $x \prec y$.

C.6 Proof of the Product Domination Lemma

In this section, we complete the proof of the Product Domination Lemma, and therefore also of the Domination Lemma.

Let ω be a threshold that is high enough, we will specify the lower bounds on ω throughout the proof. Let t be a unary rank ω type in \mathfrak{A} . Consider the projection

$$\pi: \mathfrak{A} \rightarrow \prod_{i \in I} \overbrace{\{1, \dots, n_i\}, < \}^{k_i}}^{\mathfrak{B}}$$

which maps each element of \mathfrak{A} to the appropriate tuple of block numbers. This function is continuous. Let ω_* be the threshold obtained by applying Lemma 32 to quantifier rank r and the product \mathfrak{B} . We assume that

$$\omega \geq \omega_*. \quad (11)$$

Let t_* be the type of rank ω_* which stores the quantifier rank ω_* information of type t .

We say that $x, x' \in \mathfrak{A}$ *overlap* if there is a block which intersects both x and x' . More formally,

$$(\pi(x))[i][j] = (\pi(x'))[i][j'] \quad \text{for some } i \in I \text{ and } j, j' \in \{1, \dots, k_i\}.$$

Note that overlapping is defined purely in terms of the image under π . The first step in the proof is the following claim, which shows that there is a dominating coordinate when only comparing non-overlapping elements.

▷ **Claim 34.** There exist $d \in I$, $e \in \{1, \dots, k_d\}$ and $q \in \{-\omega_*, \omega_*\}$ such that

$$x[d][e] \sqsubset^{\mathfrak{B}} y[d][e] \quad \text{implies} \quad x \prec y \quad \text{for all non-overlapping } x, y \in \mathfrak{A} \text{ of type } t_*.$$

Proof. One can show that there is a continuous section σ of π such that $\sigma \circ \pi$ preserves the type t_* , i.e., it maps t_* to a subset of itself. To define the section, one only needs to choose for each coordinate (i, j) and each block of \mathfrak{A}_i an appropriate representative such that tuples of type t_* are mapped to tuples of type t . Using compositionality, we thus have that σ maps tuples of the same type to tuples of the same type.

Define $\prec_{\mathfrak{B}}$ to be the pre-image of \prec under this section. Since σ is continuous, the order $\prec_{\mathfrak{B}}$ is definable using quantifier rank r . By Lemma 32 and the definition of ω_* , there exist dominating coordinates $d \in I$, $e \in \{1, \dots, k_d\}$ and a polarity $q \in \{-\omega_*, \omega_*\}$ such that

$$x[d][e] \sqsubset^q y[d][e] \quad \text{implies} \quad x \prec y \quad \text{for all } x, y \in \mathfrak{A} \text{ of type } t_* \text{ in the image of } \sigma. \quad (12)$$

We now extend the above result to non-overlapping elements of type t_* , as required in the statement of the claim. Using compositionality, one can show that if x and y are non-overlapping, then the binary rank r type of the pair (x, y) in \mathfrak{A} is uniquely determined by the unary rank r types of x and y as well as the binary rank r type of $\pi(x, y)$. Since $\sigma \circ \pi$ does not change the value under π , it follows that x and y overlap if and only if their images with respect to $\sigma \circ \pi$ overlap. As we have argued at the beginning of this proof, $\sigma \circ \pi$ maps the set of tuples of type t_* to a subset of itself, and therefore if x and y have type t_* , then also their images under $\sigma \circ \pi$ have type t_* . It follows that

$$(x, y) \equiv_{\omega_*} (\sigma \circ \pi)(x, y) \quad \text{for all non-overlapping } x, y \in \mathfrak{A} \text{ of type } t_*$$

By combining this observation with (12), we obtain the conclusion of the claim. ◀

The general idea in the rest of the proof is to show that if $x, y \in \mathfrak{A}$ of type t are possibly overlapping, then one can find a z which overlaps neither with x nor with y and where $x \prec z \prec y$ can be shown using Claim 34. To find this z , we will shift x (or y) by several blocks to the left or right, as explained below.

For $b \in \mathfrak{B}$ and a possibly negative integer δ , define $b + \delta$ to be the result of adding δ to all coordinates of b . Note that $x + \delta$ might fall out of \mathfrak{B} , e.g. because some coordinate might become negative. For $x \in \mathfrak{A}$, define Δ_x to be the set of integers δ such that

$$\pi(y) = \pi(x) + \delta \quad \text{for some } y \in \mathfrak{A} \text{ of type } t_*.$$

The key observation is the following claim, which says that either Δ_x is big for all $x \in \mathfrak{A}$ of type t , or one can trivially find a dominating coordinate, because there is a choice of coordinates the values at which always lie in the same block.

▷ **Claim 35.** One of the following holds:

1. There are $i \in I$ and $j \in \{1, \dots, k_i\}$ such that

$$x[i][j] \not\sqsubset y[i][j] \quad \text{for all } x, y \in \mathfrak{A} \text{ of type } t.$$

2. For every $x \in \mathfrak{A}$ of type t , the set Δ_x contains $\{-p, \dots, p\}$.

Proof. For $s \in \{0, 1, \dots\}$ define t^s to be rank s information stored in type t , i.e. this is the unique rank s type contained in t . By continuity of π , the image of t^s under π is a rank s type in the structure \mathfrak{B} , call it $t_{\mathfrak{B}}^s$. By the Threshold Lemma, every rank s type in \mathfrak{B} amounts to measuring distances between coordinates and minimal and maximal elements, up to threshold 2^s . We say that $t_{\mathfrak{B}}^s$ is *anchored* if it fixes the distance of some coordinate to either the minimal or maximal element, i.e. some (equivalently, every) x in \mathfrak{B} of type $t_{\mathfrak{B}}^s$ is such that $x[i][j]$ has distance $< 2^s$ from either 1 or n_i for some $i \in I$ and $j \in \{1, \dots, n_i\}$. If $t_{\mathfrak{B}}^s$ is anchored, then item 1 in the claim holds. Suppose that $t_{\mathfrak{B}}^{s+1}$ is not anchored. It follows from the Threshold Lemma that for every b of type $t_{\mathfrak{B}}^{s+1}$ and every $\delta \in \{-2^s, \dots, 2^s\}$, the shifted value $b + \delta$ has type $t_{\mathfrak{B}}^s$. In particular, if $s \geq \omega_*$ and $t_{\mathfrak{B}}^{s+1}$ is not anchored, then Δ_x has size at least 2^{s+1} for every x of type $t_{\mathfrak{B}}^{s+1}$. Thus, with the assumption that

$$\omega > \omega_* + \log p,$$

the result follows. ◀

Apply Claim 35. If the first case holds, then i and j are dominating coordinates by vacuous truth. We are left with the other case. We will show that d and e , as in Claim 34, are dominating coordinates. Assume that $x, y \in \mathfrak{A}$ have type t and assume $x \sqsupset^p y$. We will show $x \prec y$. By the assumption on Δ_x , we know that for every integer δ with $0 < \delta < p$, there is a x_δ of type t_* such that

$$\pi(x_\delta) = \pi(x) + \delta.$$

For every $i \in I$ and $j \in \{1, \dots, k_i\}$, there are at most two choices of δ such that

$$(\pi(x))[i][j] = (\pi(x_\delta))[i][j] \quad \text{or} \quad (\pi(y))[i][j] = (\pi(x_\delta))[i][j].$$

By a counting argument and thanks to the definition of p , it follows that there is some $p_{\mathfrak{B}}$ with

$$p_{\mathfrak{B}} < \delta < p - p_{\mathfrak{B}}$$

such that x_δ overlaps neither with x nor with y . Therefore, we can use Claim 34 to get $x \prec x_\delta \prec y$.