A NON-DENSITY ASPECT OF THE RATIONALS

HEER TERN KOH, ALEXANDER G. MELNIKOV, AND KENG MENG NG

Abstract. The relation 'being primitively recursively isomorphic' is a reduction (a pre-order) rather than an equivalence relation between presentations of an algebraic structure. It leads to the definition of *punctual degrees* of a given algebraic structure. We show that the punctual degrees $PR(\mathbb{Q})$ of the order of the rationals are not dense.

Contents

1. Introduction 1
2. The setup and the requirements 5
3. An informal description of the strategies 6
4. Labelling and defining subintervals 21
5. Formal Construction 25
6. Verification 33
References 40
Appendix A. Proofs of lemmas about labels 42

1. Introduction

In this paper we investigate the sub-recursive content of the dense linear order $\eta = (\mathbb{Q}, <)$, and we discover that it is remarkably complex. It is well-known that any dense countable linear orders without endpoints is isomorphic to η. The elementary back-and-forth proof of this fact is clearly algorithmic in nature. This intuition can be clarified using the classical notion of *computable categoricity*. We follow Mal'cev [Mal61] and Rabin [Rab60] and say that a structure is *computable* if its domain is ω and the relations and the operations of the structure are computable predicates and functions upon ω. If $\mathcal{A} \cong \mathcal{M}$ and $\mathcal{A}$ is computable, then we say that $\mathcal{A}$ is a computable presentation of $\mathcal{M}$. Following Mal'cev, we say that a countable algebraic structure $\mathcal{M}$ is s *computably categorical* or *autostable* if any two computable presentations of $\mathcal{M}$ are computably isomorphic. Thus, we obtain the elementary folklore fact: *The dense linear order $\eta = (\mathbb{Q}, <)$ is computably categorical.*

The study of computable categoricity has accumulated a lot of intricate techniques and deep results; we cite the books [EG00, AK00, Mon21]. Computably categorical structures can be very complex in general since their index set is Π^1_1-complete [DKL$^+$15]. Remarkably, there also exist structures having exactly two computable presentations up to computable isomorphism [Gon80b, Tur20]. In the terminology

Date: August 2022.

of [KS99, HKSS02], such structures have 'computable dimension' two. However, non-trivial examples of computably categorical structures or structures of finite computable dimension $n > 1$ have to be *specifically constructed*, and furthermore this requires significant effort. Such complex examples can be found among two-step nilpotent groups [Gon81], fields [MPSS18], and in some other classes [HKSS02], but these have to be specifically built using elaborate techniques. It appears that practically all algorithmically presented algebraic structures encountered in mathematical practice will possess a computable dimension of either 1 or ω, and this fact is likely to be apparent and lacking in depth. This assertion certainly holds true for the dense linear order $(\mathbb{Q}, <)$ and for linear orders more generally [Rem81, GD80]. The same can be said about, e.g., Boolean algebras [Gon97, LaR77], broad classes of abelian groups [Smi81, Gon80a, MN18], trees of finite height [LMMS05], ordered abelian groups [GLS03], and finitely generated groups and algebras (folklore).

Why do all naturally occurring algebraic structures generally lack deep effective categoricity properties? It is crucial to emphasize that the discussed above approach to algorithms in algebra does not make any assumptions regarding resource or time restrictions on the algorithms. In particular, the use of *unbounded search* is permitted. In [Ner14], Nerode straightforwardly acknowledges:

> "Unbounded search is a mathematical construction in a mathematical universe, not realizable in the world in which we live."

However, the excessive computational power of the unbounded search is often conveniently assumed throughout the vast mathematical literature, and often without further justification. For example, the classical study of the Word Problem in finitely generated groups [Mil92, LS01] typically assumes that the Word Problem is invariant under the change of group presentation; this is, however, only true if we allow unbounded search. The same can be said about the aforementioned back-and-forth proof illustrating that $(\mathbb{Q}, <)$ is computably categorical. One might start to suspect that permitting unbounded search overlooks many intriguing algorithmic properties and effects that could vary across different presentations of the same structure.

What happens if we forbid unbounded search in computable structure theory? As argued in [BDKM19, KMN17b], the most natural way to attack this question is to use *primitive recursive* procedures. It was proposed in [KMN17b] to systematically investigate structures that admit a presentation with primitive recursive operations, as defined below.

Definition 1.1 ([KMN17b])**.** A countable structure is *fully primitive recursive* or *punctual* if its domain is $\mathbb{N}$ and the operations and predicates of the structure are (uniformly) primitive recursive.

The main intuition is that we need to define more of the structure "without delay". However, in contrast with automatic [KN94] or polynomial-time structures [NR87], the apriori bounds in a punctual algorithm can still be relatively inefficient computationally. So one would perhaps expect that the theory of punctual structures should not be too different from the theory of computable structures since Definition 1.1 is still rather abstract. This naive intuition has proven to be entirely false. Indeed, as explained in [BDKM19, DMN21], punctual structures tend to be more technically related to polynomial-time structures and to online combinatorics [Kie98, BEY98]

rather than to computable structures. We cite the surveys [BDKM19, DMN21] for many counter-intuitive results which required novel techniques to be developed. The main result of this article surprisingly refutes a seemingly natural conjecture which we will discuss shortly.

Following the same pattern as before, fully primitive recursive (i.e. punctual) structures should be studied up to primitive recursive isomorphism. The difference between the primitive recursive and the computable cases becomes immediately apparent. While the inverse of a computable bijective function is also computable, the inverse of a primitive recursive bijective function is not necessarily primitive recursive. To address this issue, we could restrict ourselves to *punctual* isomorphisms; these are the primitive recursive isomorphisms whose inverses are also primitive recursive. This idea was explored in [KMN17b, MN20, DHTK$^+$20, DGM$^+$20, Ala18]. In the present paper we adopt a different, more subtle approach. It was proposed in [KMN17a] to instead view 'being primitively recursively isomorphic' as a *reduction* between presentations, as clarified below.

Suppose $\mathcal{A}$ and $\mathcal{B}$ are punctual structures. We say that $\mathcal{A}$ is *punctually reducible* to $\mathcal{B}$, written $\mathcal{A} \leq_{pr} \mathcal{B}$, if there exists a primitive recursive surjective isomorphism $f : \mathcal{A} \to \mathcal{B}$. When $\mathcal{A} \leq_{pr} \mathcal{B}$, up to a primitive recursive delay we can view $\mathcal{A}$ as a substructure of $\mathcal{B}$ at every stage. We write $\mathcal{A} \cong_{pr} \mathcal{B}$ when $\mathcal{A}$ and $\mathcal{B}$ are punctually reducible to each other. Note that $\mathcal{B} \leq_{pr} \mathcal{A}$ does not have to be witnessed by the inverse of a primitive recursive isomorphism illustrating $\mathcal{A} \leq_{pr} \mathcal{B}$.

Definition 1.2 ([KMN17a])**.** We write $\mathbf{PR}(\mathcal{M})$ to denote the collection of all *punctual degrees* of $\mathcal{M}$, which are the $\cong_{pr}$-classes of its punctual presentations under $\leq_{pr}$.

It turned out that the punctual degrees of algebraically tame structures can possess many counter-intuitive properties. For example, it was illustrated in [MN16] that the punctual degrees of the dense linear order η, the countable atomless Boolean algebra, and the random graph are all pairwise *not* isomorphic. All three structures seem to share essentially the same algorithm witnessing their computable categoricity, with exactly one instance of unbounded search at every stage. However, the result from [MN16] detects the subtle difference in the (truly) unbounded search in the back-and-forth proofs for these homogeneous structures. This is in contrast to the coarser invariants of computable and punctual dimension which fail to distinguish between these three structures. These results show that studying the punctual degree structure $\mathbf{PR}(\mathcal{M})$ of an algebraic structure $\mathcal{M}$ is important beyond just being of mere technical interest. It can reveal the very subtle intricacies of the algorithmic nature of the structure that other methods cannot detect nor explain. This is the main concern of the present paper.

Given a familiar structure $\mathcal{A}$, what kind of properties does $\mathbf{PR}(\mathcal{A})$ exhibit (as a partial ordering)? This question can be remarkably challenging even for some common structures such as, e.g., the random graph [MN16], the ordered semi-ring $(\mathbb{N}, <, +, \times, 0, 1)$ [KMZ23], and the discrete order $(\mathbb{Z}, <)$ [DDH$^+$]. Some progress has been made in the study of the punctual degrees of finitely generated structures [BKMN20, KMZ23], especially in the case of rigid finitely generated structures, but many problems remain unresolved.

Conversely, given some property, can we find a structure whose punctual degrees have this property? Attacking the latter question usually involves constructing some 'artificial' punctual $\mathcal{A}$ so that $\mathbf{PR}(\mathcal{A})$ has the desired property. For example, there exists a structure $\mathcal{A}$ such that the partial order $\mathbf{PR}(\mathcal{A})$ is infinite and has the greatest and the least element [BKMN20]. Also, one can have $|\mathbf{PR}(\mathcal{A})| = 2$ [MN20]; further results can be found in [KMN17a, GHTMT21]. We state one such result that is directly related to the present paper.

Theorem 1.3 ([GHTMT21])**.** *There exists a structure $\mathcal{A}$ such that the punctual degrees of $\mathcal{A}$ are not dense.*

The proof of the result in [GHTMT21] uses a tree of strategies to construct a structure $\mathcal{A}$ and its punctual copies $\mathcal{B} <_{pr} \mathcal{C}$ so that there is no $\mathcal{M}$ with $\mathcal{B} <_{pr} \mathcal{M} <_{pr} \mathcal{C}$. This result (and its relative complexity) suggests that the non-density of $\mathbf{PR}(\mathcal{A})$ is a *pathology*, and that structures with this property have to be specifically constructed. Compare this with structures having computable dimension two; as we discussed earlier, such examples have to be specifically constructed. Also, the punctual degrees are dense for finitely generated structures [BKMN20] and for the order $(\mathbb{Z}, <)$ [DDH$^+$]. These results seemed to further support this intuition.

The principal result of this paper states:

Theorem 1.4. *The punctual degrees of $(\mathbb{Q}, <)$ are not dense.*

The high combinatorial complexity of the proof of Theorem 1.4 is potentially more unexpected than the result itself. Indeed, not all aspects of $(\mathbb{Q}, <)$ are elementary. For instance, the automorphism group of $(\mathbb{Q}, <)$ is a non-trivial object [Mac11, Gla81]. The complexity of our proof is *not* related to the level of injury or guessing in the construction. Rather, it is related to the complexity of $Aut(\mathbb{Q}, <)$. We need to build many automorphisms of $(\mathbb{Q}, <)$ stage-by-stage while simultaneously resolving multiple tensions between different strategies. This is done using an elaborate system of labels which will help us to sort out the combinatorics. In Section 3 we thoroughly explain the main strategies and ideas behind the formal construction. This should help the reader to understand the technical formal proof.

We also remark that there are very few structures $\mathcal{A}$ for which its punctual degrees $\mathbf{PR}(\mathcal{A})$ is known to be not dense. Indeed many early results about the structure of $\mathbf{PR}(\mathcal{A})$ for a given $\mathcal{A}$ focusses on using the 'switching' argument to prove the density of $\mathbf{PR}(\mathcal{A})$ (e.g. [BKMN20, KMZ23, DDH$^+$]). The earliest techniques for controlling the size of (an interval) in $\mathbf{PR}(\mathcal{A})$ comes from the 'pressing' technique developed in [KMN17b], and later adapted to construct various structures $\mathcal{A}$ with different restrictions on $\mathbf{PR}(\mathcal{A})$ (see [MN20, DGM$^+$20]). The example mentioned above [GHTMT21] was also specifically constructed with non-density in mind.

In view of these results in the literature, we believe the result of the present paper is a significant breakthrough in the study of punctual structure theory. All of the known techniques for producing empty intervals in $\mathbf{PR}(\mathcal{A})$ rely on being able to specifically construct the structure $\mathcal{A}$, usually a graph, in which some version of the pressing technique can be employed. Our result is the first example of a natural structure in which its punctual degrees is not dense. Another significant fact is that the example we found was of a homogeneous structure in which there is no way to distinguish between different 'components', the latter being a key thrust in many proofs in computable structure theory. Therefore our proof will have to rely

on developing various novel techniques to overcome the combinatorial difficulties presented.

There are many open questions in the area, some of which are directly related to the study of punctual degrees of common structures; see [BDKM19]. We of course would also like to have some general results and meta-theorems that would relate order-theoretic properties of $\mathbf{PR}(\mathcal{A})$ with algebraic properties of $\mathcal{A}$, in the spirit of the results from [BKMN20, KMZ23] mentioned above. However, it appears crucial to gather more results about common algebraic structures first, as our intuition repeatedly proves to be unreliable and our techniques seem often insufficient.

2. The setup and the requirements

Let $\{A_k\}_{k\in\omega}$ be an effective listing of all primitive recursive presentations of all structures in the language of one binary relation. (We are manly interested in those A_k that are dense linear orders with no endpoints. It is Π^0_2 to tell whether A_k is indeed isomorphic to $\mathbb{Q}$. Since the natural presentation of $\mathbb{Q}$ is primitively recursively universal, each A_k can be viewed as a subset of this presentation.) Let $(\phi_n)_{n\in\omega}$, $(p_i)_{i\in\omega}$, $(q_j)_{i\in\omega}$ be uniformly computable lists of (all) primitive recursive functions; we use three different lists in place of just one for notational convenience. We build two punctual presentations $B \leq_{pr} T$ of the dense order of the rationals that satisfy the following requirements.

$R_{\langle i,j,k\rangle}$: If $p_i : B \to_{iso} A_k$ and $q_j : A_k \to_{iso} T$ then either $\beta_{\langle i,j,k\rangle} : A_k \to_{iso} B$ or $\alpha_{\langle i,j,k\rangle} : T \to_{iso} A_k$,

where '$d : X \to_{iso} Y$' stands for 'd is a (surjective) isomorphism from X onto Y', and

N_m : $\phi_m : T \to B$ is not a surjective isomorphism.

In $R_{\langle i,j,k\rangle}$ the maps $\beta_{\langle i,j,k\rangle}$ and $\alpha_{\langle i,j,k\rangle}$ will be constructed in stages in the proof, thus witnessing that A_k, if it is indeed a punctual presentation of $\mathbb{Q}$, does not lie strictly in-between B and T.

2.1. Notation and conventions.

We also need to fix some notation that will be used throughout the proof, and usually without explicit reference.

Notation 2.1. *For convenience, $R_{\langle i,j,k\rangle}$ will be referred to as R_e, similarly, p, q, β, α and A mentioned in $R_{\langle i,j,k\rangle}$ will all be indexed by e (where e could be thought of as $e = \langle i, j, k\rangle$).*

Notation 2.2. *Define $\mathbb{Q}_{nd} = \mathbb{Q} \setminus \mathbb{Z}[\frac{1}{2}]$ to be the set of all non-dyadic rationals. We also fix the natural presentation of $\mathbb{Q}_{nd}$ via fractions throughout the proof.*

We construct punctual presentations $B = \bigcup_s B_s$ and $T = \bigcup_s T_s$ as subsets of $\mathbb{Q}_{nd}$ via stages, and under the order inherited from $\mathbb{Q}_{nd}$. We ensure that at every stage s, $B_{s+1} \setminus B_s$ and $T_{s+1} \setminus T_s$ is non-empty. Note that we shall treat each element of B, T as a non-dyadic rational.

To better track the 'flow' of the elements entering B and T during the construction, and for the sake of the definitions of certain functions, each interval will be labelled by some finite string. The formal definition of the labels will be introduced

in Section 4, where we also establish and verify several technical properties of such string-labels that will be necessary to run (and then verify) the construction.

2.2. **The global requirements.** In what will follow, we shall monitor various primitive recursive functions (p, q etc.) which are not defined by us. Whenever we see that such a function is not order-preserving we immediately declare the respective requirement met. We will assume that this action is taken in the background, and we will never mention it explicitly in the informal description below. The same can be said about the potential structures A below T and above B: if A is not a linear order then we instantly declare the requirements associated with A met.

At every stage we will have $B \subseteq T \subset \mathbb{Q}_{nd}$. Thus, we have a primitive recursive isomorphism

$$\mathfrak{i} : B \to T$$

which is just the inclusion (identity) map.

Also, we have to ensure that both B and T are dense without end-points. It is crucial that we do *not* have to meet the densification sub-requirements promptly. In other words, we can afford adding such a z witnessing density arbitrarily late in the construction, and thus these actions will effect our construction rather insignificantly. Even if we declare a certain interval 'restrained' we can once in a while still add a fresh point to it, as long as it happens very rarely. Due to these global requirements being rather easy to meet, we shall not make their strategies explicit in the description below. The reader should however keep in mind that these global requirements will be met in the general construction.

3. An informal description of the strategies

3.1. **A global overview of the construction.** We suppress the indices throughout. Recall that $B \leq_{pr} T$ is witnessed by $\mathfrak{i} : B \to T$. Of course $\mathfrak{i}^{-1}$ is not necessarily primitive recursive because of the free elements in $T \setminus B$. In the informal explanation, we imagine that A and the maps $p : B \to A$ and $q : A \to T$ are 'played' by the 'opponent'. In order to construct a structure A such that $B \leq_{pr} A \leq_{pr} T$, the opponent must provide us with primitive recursive functions $p : B \to A$ and $q : A \to T$. In order to satisfy R, we attempt to define primitive recursive functions $\beta : A \to B$ or $\alpha : T \to A$.

According to our notation, the composition $qp : B \to T$ gives an embedding of B onto T, and we shall argue shortly that it should agree with $\mathfrak{i} : B \to T$ (in the sense that $qp = \mathfrak{i}$) otherwise we can diagonalise against either p or q. In other words, we can view A as a punctual substructure of T and a superstructure of B. The most straightforward plan for meeting an N-requirement

$$\phi : T \to B \text{ is not a surjective isomorphism}$$

is to add extra points to T and restrain their ϕ-images from entering B; we will call such elements currently in $T \setminus B$ *free*. The danger is that A may attempt to switch between copying T and B by adding the free points in some irregular way. The overall plan is to build T in a way that it contains as few free points as possible when compared to B. If A 'responds' with a primitive recursive delay by giving its versions of the extra diagonalization points, then we can use this delay to show that $A =_{pr} T$. On the other hand, if there is no primitive recursive upper bound on how slowly A responds, then we can 'copy' A into B and show that $B <_{pr} T$ and $A =_{pr} B$.

Simply put, if A is 'slow' then we will attempt to copy A into B, and this seems consistent with $B <_{pr} T$. If A is 'quick' then, with a primitive recursive delay, it should essentially be T. The obvious tension here is that, of course, we cannot know ahead of time whether A is 'slow' or 'quick'. Thus, we will have to 'press' A to make a choice. We will set up the construction so that, for any given primitive recursive ϕ, either A always responds with delay ϕ, or we diagonalise against ϕ. Then the Π^0_2 outcome corresponds to the situation when we diagonalise against all such ϕ, one-by-one.

The idea here is to make β surjective only if A is 'slow' (Π^0_2), thus witnessing $A \leq_{pr} B$ via β. In this case α will be initialized infinitely often. Every time we initialize α, the strategy will pick the next timestamp function ϕ to measure the speed of convergence of α. On the other hand, if 'A is quick' (as witnessed by some ϕ; this is Σ^0_2) is the true outcome, then we will eventually stick with a stable α primitive recursive in ϕ. Also, in this case β will influence only some part of B leaving enough room for action for the other strategies. The definition of β is not too complex; it will be explained shortly. Recall that β corresponding to A has to be total only under the Π^0_2-outcome (when A is 'slow'). Thus, at every stage of the construction β will not be onto; its range will have 'gaps'. This is explained below.

3.2. **Maintaining the gaps in rng(β).** The most natural (naive) way to define α is to set $\alpha = q^{-1}$, provided that q^{-1} converges with the speed ϕ. If q^{-1} currently seems primitive recursive (via ϕ), then β will stop extending its range, and thus (at least temporarily) fail to be surjective. This idea will be implemented as follows. Pick two dyadic rationals x_0, x_1 and keep (x_0, x_1) out of the range of β. We refer to the interval (x_0, x_1) as the *gap* of β. As long as q^{-1} seems primitive recursive with respect to the current timestamp function ϕ, the gap will remain. While the gap remains, we attempt to define α based on q^{-1} to obtain $T \leq_{pr} A$.

If it is discovered that q^{-1} not primitive recursive via ϕ, we abandon this definition of α and shrink the gap (x_0, x_1). If q^{-1} seems primitive recursive for cofinitely many stages, then the version of α defined past that stage is a primitive recursive isomorphism. If we can ensure that each successive gap is a strict subset of the previous one, with their lengths approaching 0 and so that they converge to a dyadic point, then β will end up being surjective in the Π^0_2 outcome. We never initialise β.

3.3. **One R-strategy and infinitely many N-strategies.** In the following subsections, we provide a more in-depth discussion of the details of the strategies.

3.3.1. *The basic Strategy for R.* The main challenge of the strategy will be defining $\alpha = q^{-1}$ primitively recursively. Of course we cannot always succeed in doing so, but we shall pair each attempt of defining α with the help of $\phi = \phi_m$ provided by N_m, and in this case we say that we *assign* ϕ_m to α.

Ensure that $qp = \mathfrak{i}$, as follows. If $qp \neq \mathfrak{i}$, then fix $b \in B$ such that $qp(b) \neq b$, and (w.l.o.g.) suppose that $qp(b) > b$. Then we make sure that qp fails to be a primitive recursive isomorphism with the strategy below, as illustrated in Fig. 1. Notice that $b' \in (b, \mathfrak{i}^{-1}\, qp(b))$ can always be found as B is a copy of η. Then it must be that $qp(b') > qp(b)$, otherwise qp fails to be order-preserving. When enumerating such a b' into B, the global requirement maintaining $\mathfrak{i}$ only causes elements to enter T on the left of $qp(b)$. As long as we ensure that no elements enter between $qp(b)$

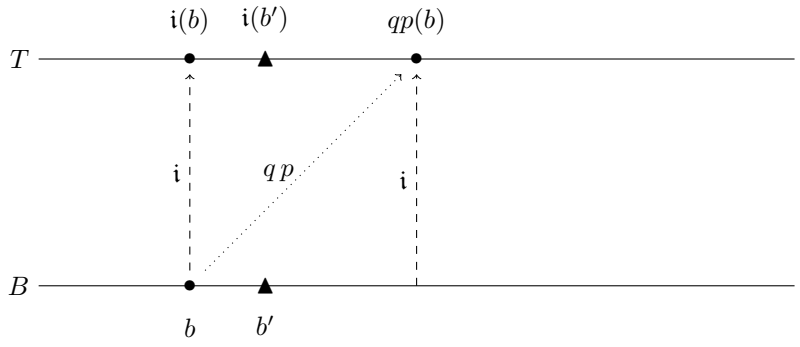

FIGURE 1. Ensuring that $qp = \mathfrak{i}$.

and $qp\,\mathfrak{i}^{-1}\,qp(b)$, qp must fail to be a primitive recursive isomorphism as it has no suitable image for b'. (Note this diagonalization is achieved in finitely many stages.)

Thus, without loss of generality, suppose that $qp = id$. Define $\beta : A \to B$ as follows. Recall that an element $t \in T$ is called free (at the stage) if t is currently not in B. Let

$$I_k \coloneqq (k, k+1),$$

for each $k \in \mathbb{Z}$. Recall that since $B_s \subseteq T_s \subset \mathbb{Q}_{nd}$, then there is a version of I_k in B_s, T_s for each k. Often times we will not explicitly state which I_k we are talking about but will be obvious from the maps. Throughout the construction, free elements can only be introduced in (the interval in T associated with) I_0. In order to keep β primitive recursive, whenever some element x is enumerated into A such that $q(x)$ (in T) is not in I_0, enumerate $q(x) + 1$ into B.

Recall that the strategy was to keep a gap in the range of β while we are attempting to define α. This gap will be some subinterval of I_1 (in B). Roughly speaking, β behaves like the translation map $q(x) \mapsto q(x) + 1$, except for the elements x such that $q(x) \in I_0$. We introduce a map $\Theta : \mathbb{Q}_{nd} \to \mathbb{Q}_{nd}$ which will be maintained for the sake of β. For $x \notin I_0$, $\Theta(x) = x + 1$. The definition for $\Theta(x)$ if $x \in I_0$ is relatively complicated and depends on various factors in the construction. In fact, most of the informal description will be dedicated to explaining certain properties which we want Θ to satisfy, and how they ensure that the construction works. Since $B \subseteq \mathbb{Q}_{nd}$, we can read Θ as a map from B to B, and define $\beta = \Theta\,\mathfrak{i}^{-1}\,q$. In other words, when some element x enters a structure A, we have to enumerate $\Theta\,\mathfrak{i}^{-1}\,q(x)$ into B without delay. We note here that even though the notation uses $\mathfrak{i}^{-1}$, regardless of whether or not $q(x) \in T$ is free (without a $\mathfrak{i}$ preimage), $\Theta\,\mathfrak{i}^{-1}$ is a map on the non-dyadic rationals independent of whether $\mathfrak{i}^{-1}\,q(x)$ is currently in B. Thus we are able to find the correct element to enumerate into B given any x in A. Since the definition of β boils down to a careful definition of Θ, we will also refer to the gap of β as the gap of Θ.

The definition of Θ will be dynamic and relatively intricate; we postpone the details until the later subsections. Here we only outline several key ideas behind this formal definition. As soon as some ϕ is newly assigned to α (to be formally clarified in the following subsection), use the definition of Θ to ensure β is temporarily non-surjective, and set $\alpha = q^{-1}$. Recall that it is assumed that $qp = \mathfrak{i}$, and so $q^{-1}(t) = p\,\mathfrak{i}^{-1}(t)$. Hence, for elements $t \in T$, which are not free, $p\,\mathfrak{i}^{-1}(t)$ must

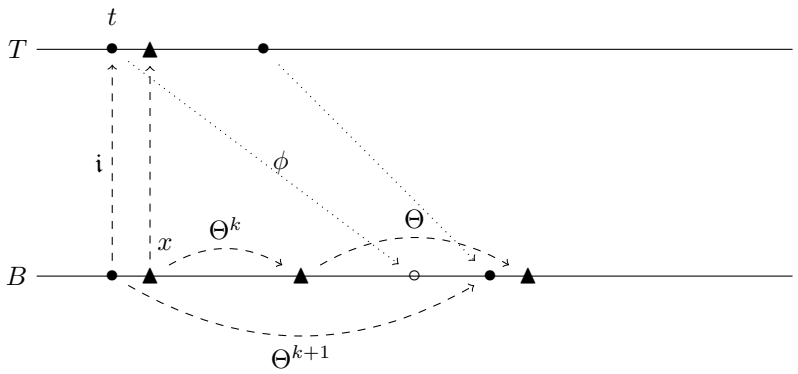

FIGURE 2. Ensuring that $\Theta^k \mathfrak{i}^{-1}$.

show up in A primitively recursively. Thus, unless ϕ_m is wrong, α can always be defined primitively recursively on elements $t \in T$ which are not free. As discussed in Subsection 3.2, if q^{-1} seems to not be primitive recursive (on the free elements), we abandon the current definition of α and remove the gap of Θ, making β 'more surjective'. Furthermore, in such a case, ϕ_m also fails to be a primitive recursive isomorphism as explained in the basic strategy for N_m below.

3.3.2. *The basic strategy for N_m.* Ensure that $\phi : T \to B$ is such that $\phi = \Theta^k \mathfrak{i}^{-1}$ for some $k \geq 0$, as follows. If there exists a $t \in T$ such that

$$\forall k \geq 0 \ \phi(t) \neq \Theta^k \mathfrak{i}^{-1}(t),$$

then let k be such that $\phi(t)$ is strictly between $\Theta^k \mathfrak{i}^{-1}(t)$ and $\Theta^{k+1} \mathfrak{i}^{-1}(t)$. Wait for $\phi^{-1}\Theta^{k+1} \mathfrak{i}^{-1}(t)\downarrow$, which must happen at some finite stage or else ϕ is not surjective. When it does halt, consider the situation illustrated in Fig. 2. Then we do the following. Since $B[s] \subseteq \mathbb{Q}_{nd}$, we can find an element x 'close enough' to $\mathfrak{i}^{-1}(t)$ such that both of the following conditions hold:

- $t < \mathfrak{i}(x) < \phi^{-1}\Theta^{k+1} \mathfrak{i}^{-1}(t)$.
- For all $l \ \Theta^l(x) \notin \big(\phi(t), \Theta^{k+1} \mathfrak{i}^{-1}(t)\big)$, which must exist because $\big(\Theta^k \mathfrak{i}^{-1}(t), \phi(t)\big) \neq \emptyset$.

Then as illustrated in Fig. 2, when such an element x enters B and $\mathfrak{i}(x)$ enters T we can assume that $\Theta^l(x)$ enters B for all $l \in \omega$. However, no new elements enter $\big(\phi(t), \Theta^{k+1} \mathfrak{i}^{-1}(t)\big)$, but in order for ϕ to be order preserving, it must be that $\phi\, \mathfrak{i}(x) \in \big(\phi(t), \Theta^{k+1} \mathfrak{i}^{-1}(t)\big)$. Thus ϕ cannot be a primitive recursive isomorphism.

Thus, we may assume that for some $k \geq 0$ we have $\phi = \Theta^k \mathfrak{i}^{-1}$.

Case 1. If $k = 0$, that is $\phi = \mathfrak{i}^{-1}$, then we enumerate a free element t in T, as illustrated in Fig. 3. Let b_0 and b_1 be currently adjacent elements in B. By enumerating a free element t such that $\mathfrak{i}(b_0) < t < \mathfrak{i}(b_1)$, since ϕ has already chosen to map $(\mathfrak{i}(b_0), \mathfrak{i}(b_1))$ to (b_0, b_1), as long as we are able to keep (b_0, b_1) empty for arbitrarily long, ϕ necessarily fails to be a primitive recursive isomorphism.

Case 2. If $k > 0$, then assume $\phi = \phi_m$ has been assigned to R (or equivalently, N_m has been assigned to α). Introduce a gap into the range of θ. Now R attempts to define α to satisfy

$$\alpha = q^{-1}.$$

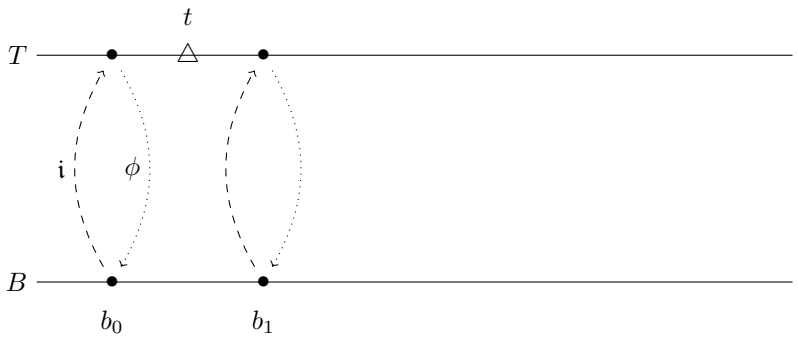

FIGURE 3. Enumerating a free element.

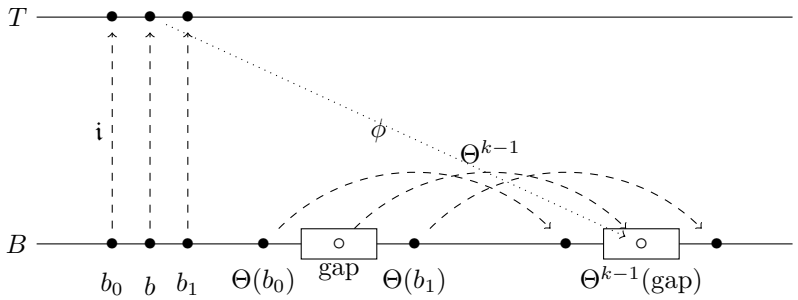

FIGURE 4. If ϕ maps into the gap.

As discussed in Subsections 3.2 and 3.3, there is now some part of I_1 which is currently not in the range of θ. While there is a gap in the range of θ, to ensure that the structure is dense, we need to enumerate elements into the gap. However, the definition of the gap means that there are no elements in I_0 which map to any element in the gap under Θ. Thus, if ϕ ever maps some element into the gap or its subsequent images under Θ, we can make ϕ fail as a primitive recursive isomorphism based on the idea sketched on Fig. 4, as follows

Suppose that $\mathfrak{i}(b)$ is the first element which is mapped by ϕ into $\Theta^{k-1}(\text{gap})$ for some $k \geq 0$. By assumption, we also have that $\phi\,\mathfrak{i}(b_0) = \Theta^k(b_0)$ and $\phi\,\mathfrak{i}(b_1) = \Theta^k(b_1)$, as ϕ must have decided upon the image of some elements before the gap was introduced. Now consider the possible positions of $\Theta^k(b)$. Suppose first that $\Theta^k(b) < \phi\,\mathfrak{i}(b)$. Then there are infinitely many $x > b$ and 'close enough' to b such that $\Theta^k(x) < \phi\,\mathfrak{i}(b)$. In particular, $\Theta^k(x) \notin \big(\phi\,\mathfrak{i}(b), \Theta^k(b_1)\big)$. There can only be finitely many elements currently in $\big(\phi\,\mathfrak{i}(b), \Theta^k(b_1)\big)$, say, n elements. Then by enumerating $n+1$ many such x 'close enough' to b, and temporarily suspending elements from entering the gap and thus $\Theta^{k-1}(\text{gap})$, ϕ must fail to be a primitive recursive isomorphism. So it must be that ϕ truly follows Θ^k even after N_m is assigned to R_e and a gap is introduced. Thus in such a case, ϕ cannot be surjective.

3.3.3. *Putting the strategies together.* Without loss of generality, suppose that $qp = \mathfrak{i}$ and that, for every m, $\phi_m(x) = \Theta^{k_m}\,\mathfrak{i}^{-1}(x)$ for some $k_m \geq 0$. At each stage, pick the least m such that N_m is currently not satisfied and not assigned to R. Then

wait for $\phi_m(t)\downarrow$ for some $t \in I_0$. As explained earlier, either N_m is assigned to R or $\phi_m(t) = \mathfrak{i}^{-1}(t)$. Then we have the following possibilities.

Case 1. No other N-requirement is currently assigned to R. Then if $\phi_m(t) = \Theta^{k_m}(t)$ for some $k_m > 0$, as in the basic strategy, N_m is then assigned to R and R attempts to define $\alpha = q^{-1}$, and puts a gap in the range of θ. If $k_m = 0$, then as explained in the basic strategy for N_m, a free element is enumerated into T and N_m will be satisfied once ϕ_m fails to be a primitive recursive isomorphism.

Case 2. Some other N-requirement is currently assigned to R. If it is some lower priority N-requirement assigned to R we initialise it for the sake of N_m of higher priority. We may thus suppose that $N = N_{m'}$ for some $m' < m$ is assigned to α. Then Θ currently has a gap in its range. There are then two further possibilities for N_m. If $\phi_m = \Theta^k\,\mathfrak{i}^{-1}$ for some $k > 0$, then it cannot possibly be surjective. If instead $\phi_m = \mathfrak{i}^{-1}$, then we enumerate a free element t into T. N_m must be satisfied as long as we keep t free until $\phi_m(t)\downarrow$. Furthermore, since $\phi_{m'}$ is already assigned to α, when t is enumerated free, either $\alpha(t)$ converges (before $\phi_{m'}$), or we obtain a diagonalisation against $\phi_{m'}$. That is, after enumrating t free, we either satisfy $N_{m'}$ of higher priority or we satisfy N_m while maintaining the primitive recursiveness of α.

3.3.4. *The verification (only one R).* Now we provide some explanation as to why R and the infinitely many N_m are satisfied. To see that R is satisfied, observe that it either swaps infinitely often between defining β and α, in which case Θ is eventually surjective, or for cofinitely many stages some ϕ_m is permanently assigned to R. In the latter case q^{-1} is primitive recursive relative to ϕ_M, and hence $\alpha = q^{-1}$ is a primitive recursive isomorphism.

To see that each N_m is satisfied, again consider the two outcomes of R separately. First, suppose R swaps between defining β and α infinitely often, which is the Π^0_2-outcome. Each time it does so, its instructions guarantee that the highest priority N_m currently assigned to R is met. Then every N_m eventually assigned to R must be met. For the remaining N_m requirements never assigned to R, it must be that $\phi_m = \mathfrak{i}^{-1}$. These requirements are then satisfied by the enumeration of free elements. Thus all N_m are satisfied.

In the Σ^0_2-outcome, when R defines α for cofinitely many stages, the ϕ_m assigned to α never becomes surjective. This is because it faithfully follows Θ^{k_m} for some $k_m > 0$ (which is not surjective as long as some N_m is assigned to it), and hence N_m is satisfied. For all other $N_{m'}$, we have the following possibilities. If $\phi_{m'} = \Theta^{k_{m'}}\,\mathfrak{i}^{-1}$ for some $k_{m'} > 0$, then it cannot be surjective as well. Otherwise, $\phi_{m'}$ maps some element into the gap of Θ. In the latter case we eventually diagonalise against $\phi_{m'}$. The only remaining possibility is that $\phi_{m'} = \mathfrak{i}^{-1}$, in which case we enumerate a free element $t \in T$ and make sure $N_{m'}$ is met in finite time in this case too. (Recall that by our assumption, $q^{-1}(t)$ shows up with a primitive recursive delay without us having to enumerate $\mathfrak{i}^{-1}(t)$ into B, so we can diagonalise while keeping the definition of α primitive recursive.)

3.4. **Two R-strategies with infinitely many N-strategies.** Recall that in Subsection 3.3 we noted that in the final definition of β (which will be given later) we

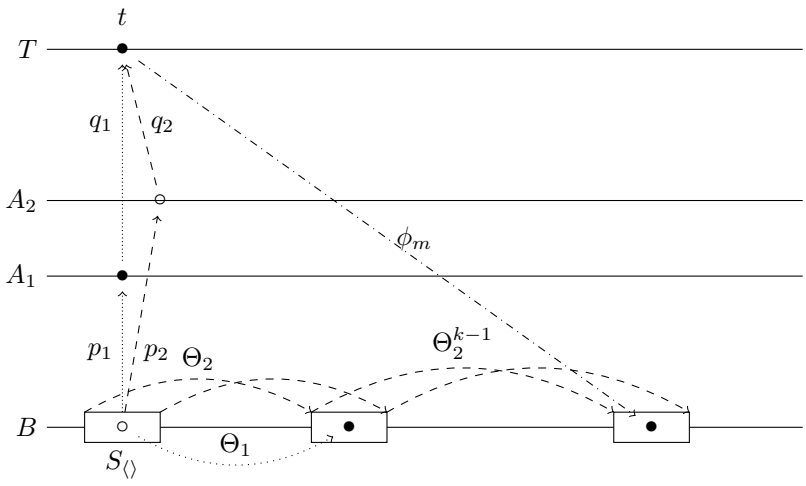

FIGURE 5. Issue 1

will use a certain auxiliary function θ and its extension Θ. The final, formal definition of θ will be relatively complex. However, it is certainly not evident at all from the description of the basic strategy in Subsection 3.3 why some naive definition of β would not be sufficient to run the construction successfully. Unfortunately, further tensions arise already when we have to deal with just two R-requirements; this is informally explained below.

3.4.1. *The two main issues with the (naive) basic strategy.* One of the main challenges in the basic strategy presented in Subsection 3.3 was in the definition of α. In order to do so, we needed to ensure the following:

- In the strategy for R, we ensure $qp = \mathfrak{i}$, which help to define $\alpha(t)$ whenever t is not free.
- In the strategy for N_m, we ensure $\phi_m = \Theta^k \mathfrak{i}^{-1}$ for some $k \geq 0$, in order to define $\alpha(t)$ when t is free. In particular, as long as t remains free, $\Theta^k \mathfrak{i}^{-1}(t)$ does not get enumerated into B provided $q^{-1}(t)\!\uparrow$.

Now consider the case when we have two different R-requirements, say R_1, R_2. We informally explain here why there is a potential issue with ensuring that $q_2 p_2 = \mathfrak{i}$. To see why it can be an issue, recall that the function Θ_1 is maintained for the sake of R_1. Thus, if we follow just the basic strategy (as outlined earlier), we will potentially be unable to diagonalise against $q_2 p_2$ if it 'chooses' to follow the map $\mathfrak{i}\,\Theta_1^k$ for some $k > 0$.

Also, we need to be careful with how exactly we define Θ_1 and Θ_2 on I_0. It would be most convenient if we could keep $\Theta_1(I_0)$ and $\Theta_2(I_0)$ disjoint so that different strategies are maximally independent from each other. Making these ranges independent will not be quite possible due to various tensions in the construction which are perhaps not entirely obvious to the reader at this stage of the informal description. Our (less ambitious) plan is to make the ranges of Θ_1 and Θ_2 disjoint at least on some small subinterval where free elements are enumerated, say $S_{\langle\rangle}$, of I_0. (We use $S_{\langle\rangle}$ here to be consistent with the more complex notation that will be introduced in later subsections.)

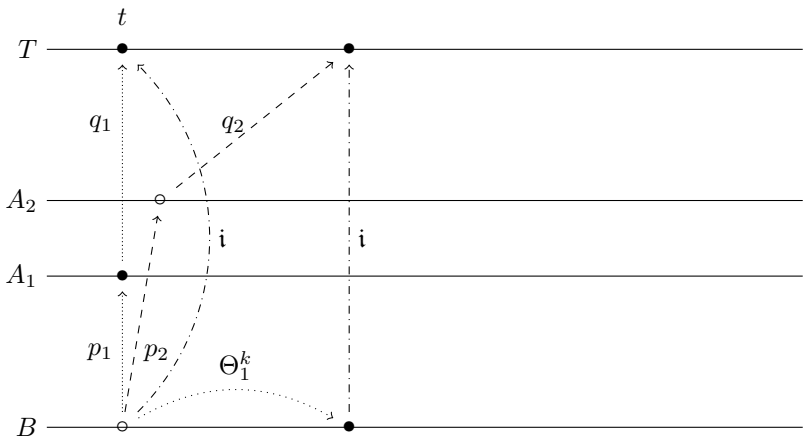

FIGURE 6. Issue 2

For simplicity, first suppose that $q_1 p_1 = q_2 p_2 = \mathfrak{i}$, and that R_2 is currently attempting to define α_2 (using the earlier strategy). The explanation that follows next is illustrated on Fig. 5. Recall that the strategy attempts to define α_2 on a free element t as follows. It waits for A_2 to reveal $q_2^{-1}(t)$ in order to maintain the primitive recursiveness of α_2 *relative to* some ϕ_m currently assigned to α_2. That is, as long as A_2 never enumerates $q_2^{-1}(t)$, no new elements will enter $\Theta_2(S_{\langle\rangle})$ (other than in its gap). In order for ϕ_m to be assigned to α_2, it must be that $\phi_m = \Theta_2^k \, \mathfrak{i}^{-1}$ for some $k > 0$. However, if $\theta_1(S_{\langle\rangle})$ and $\theta_2(S_{\langle\rangle})$ are not disjoint, the opponent could instead show $q_1^{-1}(t)$ in A_1 to force an enumeration of $\Theta_1 \, \mathfrak{i}^{-1}(t)$ into $\Theta_2(S_{\langle\rangle})$. This will force us to enumerate elements into $\Theta_2^k(S_{\langle\rangle})$ which can later serve as an image for t under ϕ_m. It is thus important to keep $\Theta_1(S_{\langle\rangle})$ and $\Theta_2(S_{\langle\rangle})$ disjoint.

Now we return to the issue mentioned earlier caused by $q_2 p_2$ 'following' $\mathfrak{i}\,\Theta_1^k$ for some $k > 0$. Consider the situation described on Fig. 6. Suppose in this case that R_2 is attempting to define $\alpha_2 = q_2^{-1}$, and that a free element t is enumerated into $S_{\langle\rangle}$ for the sake of some other requirement. If A_1 produces the element $q_1^{-1}(t)$, then we have to enumerate $\Theta_1^k \, \mathfrak{i}^{-1}(t)$ into B to keep β_1 primitive recursive. Since $\mathfrak{i}$ is maintained for the sake of the global requirement that $B \leq_{pr} T$, we would also have to enumerate $\mathfrak{i}\,\Theta_1^k \, \mathfrak{i}^{-1}(t)$ into T promptly. Then in order to keep α_2 primitive recursive, we could enumerate $\mathfrak{i}^{-1}(t)$ into B, thus forcing A_2 to enumerate $p_2 \, \mathfrak{i}^{-1}(t)$. This however forces t to no longer be free, while the strategies for satisfying (other) N-requirements depend on the ability to keep t free for arbitrarily long. (We note here that in this specific case it is perfectly fine since both α_1 and α_2 are safe. However we would not like to ever be forced to enumerate $\mathfrak{i}^{-1}(t)$ quickly should we choose t to be enumerated free.)

In summary, to address the issues, we need the following.

- $\Theta_1(S_{\langle\rangle})$ and $\Theta_2(S_{\langle\rangle})$ must be disjoint.
- α_2 is primitive recursive on elements outside of I_0.

In order to achieve these goals, we will introduce a system of labels on subintervals of I_0 and I_1, whose purpose is to trace the series of enumerations originating from $S_{\langle\rangle}$ under θ_e for various e. This system shall serve as the basis for all strategies. To

define α_2 primitively recursively on elements outside of I_0, we define $\theta_0 : I_0 \to I_1$ and extend it to $\Theta_0 : B \to B$ (recall that $B \subseteq \mathbb{Q}_{nd}$). Then take $\alpha_2 = p_2\Theta_0^{-k}\mathfrak{i}^{-1}$ for some appropriate k on intervals I_m where $m \neq 0$. Observe that on intervals which are not I_0, the map $\mathfrak{i}^{-1}$ is primitive recursive since free elements are only enumerated into I_0. As long as Θ_0^{-1} is kept primitive recursive, α_2 must also be primitive recursive, at least on elements outside of I_0.

As a consequence of keeping Θ_0^{-1} primitive recursive, the 'flow' of new elements will expand in 'both directions'. This potentially opens up more options for the 'opponent' but the system of labels is robust enough to handle this additional freedom qp has. Since the map Θ_0 (and Θ_0^{-1}) will be maintained globally, i.e., for the sake of all R-requirements, we omit the R_0-strategy for notational convenience. (Our enumeration begins with 1.)

We now depart from the case of only two strategies and explain the system of labels in presence of infinitely many R-strategies. (Of course, the reader is free to restrict themselves only to the case when we only have R_1 and R_2.)

3.5. **The system of labels and colours. An outline.** We now provide a description of the label and colour system. We refer the reader to Section 4 for the formal explanation of the labels. We shall use strings from $\omega^{<\omega}$ for the labels and thus introduce some notation which shall be used throughout.

Notation 3.1. *Let $\sigma \in \omega^{<\omega}$ be given.*

- *Let $\langle\rangle$ denote the empty string.*
- $\max(\sigma) \coloneqq \max\{\sigma(i) \mid i < |\sigma|\}$.
- *If $\sigma \neq \langle\rangle$, then σ^- denotes $\sigma \restriction_{|\sigma|-1}$.*
- *Let $^\frown$ denote concatenation. For convenience, we write $\sigma^\frown x$ to mean $\sigma^\frown\langle x\rangle$. When there is no danger of ambiguity, we opt to drop the $^\frown$ entirely and simply write σx.*

The purpose of the labels are to track the actions of Θ_e and Θ_e^{-1} for various e on I_0 and I_1 respectively. We first consider a simple example of how strings (not necessarily labels) $\sigma \in \omega^{<\omega}$ can be interpreted as compositions of Θ_e and Θ_e^{-1} maps.

Example 3.2.

- *For a string $\langle 1,0,1,2\rangle$, this corresponds to the subinterval as follows.*
$$S_{\langle\rangle} \xrightarrow{\Theta_1} S_{\langle 1\rangle} \xrightarrow{\Theta_0^{-1}} S_{\langle 1,0\rangle} \xrightarrow{\Theta_1} S_{\langle 1,0,1\rangle} \xrightarrow{\Theta_2^{-1}} S_{\langle 1,0,1,2\rangle}.$$
- *For a string $\langle 1,1\rangle$, this would correspond to the subinterval as follows.*
$$S_{\langle\rangle} \xrightarrow{\Theta_1} S_{\langle 1\rangle} \xrightarrow{\Theta_1^{-1}} S_{\langle 1,1\rangle}.$$
A quick observation would then tell us that the strings $\langle 1,1\rangle$ and $\langle\rangle$ should represent the same subinterval.

Thus it is evident that there should be some sort of equivalence between certain strings. This equivalence will be formally defined in Section 4. For each equivalence class, we fix a unique representative called the *reduced* form, and denote the reduced form of σ by σ^*. For instance, $\langle 1,1\rangle^* = \langle\rangle$, and the latter is reduced. The collection of all reduced forms will then form the *labels*. We now introduce some relevant definitions and properties of the labels.

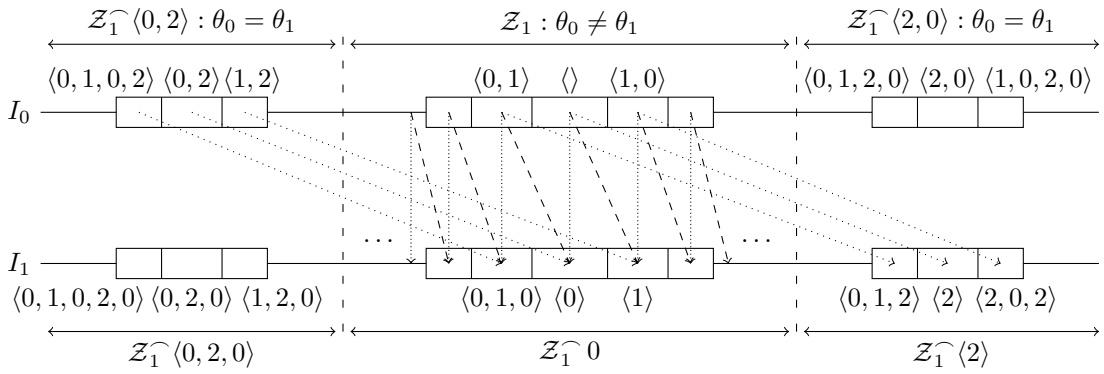

FIGURE 7. Illustration of labels

Definition 3.3. *Let Z_e denote the collection of all even length labels with maximum entry at most e. The set $Z_e^\frown 0$ is defined similarly, but it consists of all labels of odd length with maximum entry at most e. We shall use $\mathcal{Z}_e$ (and $\mathcal{Z}_e^\frown 0$) to denote the union of intervals labelled with elements of Z_e (and $Z_e^\frown 0$).*

To provide some illustration of the labels, we refer the reader to Fig. 7. (Def. 3.3 will be repeated and further clarified in Section 4 but the current ones shall suffice for the informal description.)

The following are some properties of the system of labels, which we lift directly from Section 4. For any label σ, γ, the following properties hold. Recall that every label is assumed to be reduced.

i) $S_\sigma \cap S_\gamma = \emptyset$ whenever $\sigma \neq \gamma$. This property is important in two ways.
   - To avoid the issue as presented in Fig. 5, we need $\Theta_1(S_{\langle\rangle}) = S_{\langle 1\rangle}$ and $\Theta_2(S_{\langle\rangle}) = S_{\langle 2\rangle}$ to be disjoint.
   - To avoid the issue illustrated in Fig. 6, we need $\Theta_0^{-1}\Theta_1(S_{\langle\rangle}) = S_{\langle 1,0\rangle}$ to be disjoint from $S_{\langle\rangle}$.

ii) $\Theta_e : S_\sigma \to S_{(\sigma^\frown e)^*}$ when $S_\sigma \subseteq \mathcal{Z}_e$

iii) $\Theta_e^{-1} : S_\sigma \to S_{(\sigma^\frown e)^*}$ when $S_\sigma \subseteq \mathcal{Z}_e{}^\frown 0$.

Recall that for each $e > 0$, Θ_e is maintained for the sake of R_e. Whenever $q_e^{-1}(t)$ is enumerated into A_e, then we must enumerate $\Theta_e \mathfrak{i}^{-1}(t)$ into B in order to keep β_e primitive recursive. Since the labels of subintervals track the action of the Θ_e maps, we now know exactly where to enumerate elements in order to keep the various β primitive recursive. For example, when some x enters A_e such that $q_e(x) \in S_\sigma$, then we need to enumerate $\Theta_e \mathfrak{i}^{-1} q_e(x)$ into $S_{(\sigma^\frown e)^*}$.

Throughout the construction, the enumeration of new points will originate from either $S_{\langle\rangle}$ or $S_{\langle e\rangle}$ for some $e > 0$. That is, any other action (when it comes to adding new points) will be a result of repeated applications of $\theta_{e'}$ (or $\Theta_{e'}$) or their inverses to elements of $S_{\langle\rangle}$ or $S_{\langle e\rangle}$. Each of these infinitely many cases corresponds to a certain specific scenario in the construction. To distinguish between these cases, we introduce the concept of *colour* in the definition below. Informally, we can think of subintervals with labels starting with e simply as (slightly delayed) copies of $S_{\langle e\rangle}$. Thus, it makes sense to put them together into one group and declare that its 'colour' is e. Recall that all labels are reduced.

Definition 3.4. *Let $\sigma \in \omega^{<\omega}$ be a label and $\sigma \neq \langle\rangle$. Then σ has colour $\sigma(0)$. Also let $\langle\rangle$ be defined as having colour 0.*

We clarify why it is convenient to define the colour of the empty string to be 0, i.e., has the same colour as $\langle 0\rangle$. As we mentioned earlier, both maps Θ_0 and Θ_0^{-1} will be primitive recursive and shared among all requirements R_e. Thus $S_{\langle 0\rangle}$ will simply be a copy of $S_{\langle\rangle}$. Hence, it makes sense to colour $\langle\rangle$ and $\langle 0\rangle$ identically.

The system of labels will be used to 'monitor' the actions of the maps qp and ϕ provided by the opponents. Assuming we know the position of each subinterval (to be defined in Section 4), we are then able to trace the 'cascading-effect' of enumerations. We provide an example illustrating this.

Example 3.5. *Suppose some element is newly enumerated into $S_{\langle 1,0,2\rangle}$. Then we know the following must have happened.*

(1) Some element x must first have been enumerated into $S_{\langle 1\rangle}$.

(2) In order to maintain Θ_0^{-1}, the element $\Theta_0^{-1}(x)$ was enumerated into $S_{\langle 1,0\rangle}$.

(3) $q_2^{-1}\,\mathfrak{i}\,\Theta_0^{-1}(x)\!\downarrow$ to cause an enumeration into $S_{\langle 1,0,2\rangle}$.

Such a property of the labels will turn out to be useful when we are attempting to make qp and ϕ behave in some desired way.

We give a brief explanation now as to why this system of labels resolve the issues raised earlier. Once again, we refer the reader to Fig. 6. The problem was that in order to define $\alpha_2(t')$, where $t' = \mathfrak{i}\,\Theta_1^k\,\mathfrak{i}^{-1}(t)$, we were forced to enumerate $\mathfrak{i}^{-1}(t)$ into B. However, with the new definition of $\alpha_2 = p_2\Theta_0^{-k}\,\mathfrak{i}^{-1}$ on elements outside of I_0, we have

$$\alpha_2(t') = p_2\Theta_0^{-k}\Theta_1^k\,\mathfrak{i}^{-1}(t).$$

Since $\Theta_0 = \Theta_1$ on the intervals that are not I_0, we conclude that $\alpha_2(t') = p_2\Theta_0^{-1}\Theta_1\,\mathfrak{i}^{-1}(t)$. As long as $S_{\langle 1,0\rangle}$ is disjoint from $S_{\langle\rangle}$, we can keep t free while ensuring that $\alpha_2(t)$ is defined.

3.6. **The modified basic strategy of R_e.** As seen earlier, the two essential tasks of the strategy is to make β_e primitive recursive and to define α_e using some ϕ_m that is currently assigned to the strategy. To succeed in the first task we keep Θ_e primitive recursive. To successfully define α_e, we need the following properties:

- For t not free, we must be able to 'force' A_e to show a suitable image for $\alpha_e(t)$ promptly.
- For t free (and as long as it remains free) ϕ_m assigned to α_e has no suitable image for t.

In the earlier strategy, the first condition is satisfied via ensuring that $q_e p_e = \mathfrak{i}$. However, we cannot guarantee this in general. In particular, as long as q_2p_2 only follows maps of higher priority, for example, Θ_1, Θ_0 or Θ_0^{-1}, then we are unable to diagonalise against q_2p_2 without blocking the higher priority maps. Even though we cannot ensure $q_2p_2 = \mathfrak{i}$, we are still able to ensure that it behaves in a certain way. To simplify the explanation that follows, we ignore the possibility of q_2p_2 having a 'large shift'. We only consider the possibilities where $q_2p_2(I_0) = I_k$ for some $k = -1, 0$ or 1. The more general case where $k \in \mathbb{Z}$ is an easy generalisation and will be covered in the later sections.

To avoid diagonalisation, q_2p_2 has to 'follow' only maps of higher priority. This means that q_2p_2 has to copy some composition of maps consisting only of Θ_1, Θ_0

and Θ_0^{-1}. Then q_2p_2 must map $\mathcal{Z}_1$ to $\mathcal{Z}_1 \cup \mathcal{Z}_1^\frown 0$, and as a result $(q_2p_2)^{-1}(\mathcal{Z}_1)$ must be in $\mathcal{Z}_1 \cup \mathcal{Z}_1^\frown 0$. We state this as a lemma below.

Remark 3.6. *During the construction, we will enforce certain conditions on q_ep_e. We simplify the more general statement in Section 5.1 to fit our purposes here. If there is some $\tau \in Z_1$ where $(q_2p_2)^{-1}(S_\tau)$ is not contained in $\mathcal{Z}_1$, $\Theta_0(\mathcal{Z}_1)$, or $\Theta_0^{-1}(\mathcal{Z}_1)$, then we can diagonalise against q_2 or p_2.*

Intuitively, we are able to enforce this property as follows. If q_2p_2 copies only compositions of maps Θ_0, Θ_0^{-1} and Θ_1, then it only has the ability to be different from $\mathfrak{i}$ (or some translation map $x \mapsto x+1$) on intervals in $\mathcal{Z}_1 \cup \mathcal{Z}_1^\frown 0$. Thus, if we discover that q_2p_2 does not satisfy the condition as stated in the lemma, we enumerate elements into an appropriate interval in $\mathcal{Z}_1 \cup \mathcal{Z}_1^\frown 0$ while temporarily blocking all Θ_e maps for $e \geq 2$. In this way, q_2p_2 cannot possibly be a primitive recursive isomorphism.

Recall that we made a simplifying assumption that $q_2p_2 : I_0 \to I_k$ for some $k = -1, 0$ or 1. As a result, we can conclude that $(q_2p_2)^{-1}(I_0)$ is one of I_{-1}, I_0 or I_1. We present the definitions of α_2 in each of these cases below.

(a1) If $q_2p_2 : I_{-1} \mapsto I_0$, then

$$\alpha_2(x) = \begin{cases} q_2^{-1}(x), & \text{if } x \in \mathcal{Z}_1, \\ p_2\Theta_0^{-1}\,\mathfrak{i}^{-1}(x), & \text{otherwise.} \end{cases}$$

(a2) If $q_2p_2 : I_0 \mapsto I_0$, then

$$\alpha_2(x) = \begin{cases} q_2^{-1}(x), & \text{if } x \in \mathcal{Z}_1, \\ p_2\,\mathfrak{i}^{-1}(x), & \text{otherwise.} \end{cases}$$

(a3) If $q_2p_2 : I_1 \mapsto I_0$, then

$$\alpha_2(x) = \begin{cases} q_2^{-1}(x), & \text{if } x \in \mathcal{Z}_1, \\ p_2\Theta_0\,\mathfrak{i}^{-1}(x), & \text{otherwise.} \end{cases}$$

To see why this definition works, we first check that if q_2 and p_2 are isomorphisms, then α_2 defined according to these rules is also an isomorphism (in each case). We explain only the first case; the other cases are similar. Suppose that q_2p_2 is a primitive recursive isomorphism. Applying Remark 3.6, we may assume that $(q_2p_2)^{-1}(\mathcal{Z}_1) = \Theta_0^{-1}\,\mathfrak{i}^{-1}(\mathcal{Z}_1)$. That is, $q_2^{-1}(\mathcal{Z}_1) = p_2\Theta_0^{-1}\,\mathfrak{i}^{-1}(\mathcal{Z}_1)$. If q_2 is an isomorphism, α_2 is also an isomorphism between $\mathcal{Z}_1$ and $p_2\Theta_0^{-1}\,\mathfrak{i}^{-1}(\mathcal{Z}_1)$. For the other line of Definition (a1), as long as p_2 is an isomorphism, α_2 will be isomorphism between $T \setminus \mathcal{Z}_1$ and $p_2\Theta_0^{-1}\,\mathfrak{i}^{-1}(T \setminus \mathcal{Z}_1)$. These ideas will extend to the more general case when $q_ep_e : I_k \to I_0$ (for $k \in \mathbb{Z}$ as in Def. 6.11) to define α_e.

Now we give an example to illustrate how the issues presented earlier in Fig. 5 and 6 are resolved with this new definition of α_2. For simplicity, assume $\phi_m : S_{\langle\rangle} \mapsto S_{\langle 2\rangle}$ and thus ϕ_m is assigned to α_2 (this will be explained in more detail in the strategy for N_m). The explanation that follows is illustrated in Fig. 8.

The first issue is simply resolved by ensuring that $S_{\langle 1\rangle}$ and $S_{\langle 2\rangle}$ are disjoint. Now given a free element t, even if $q_1^{-1}(t)$ is enumerated into A_1, we still need to enumerate $\Theta_1(t)$ into $S_{\langle 1\rangle}$ for the sake of β_1 as before. However, since $S_{\langle 1\rangle}$ and $S_{\langle 2\rangle}$ are disjoint, no new elements are enumerated into $S_{\langle 2\rangle}$, and ϕ_m still does not have a suitable image for the free element t. That is, as long as $q_2^{-1}(t)$ does not show up

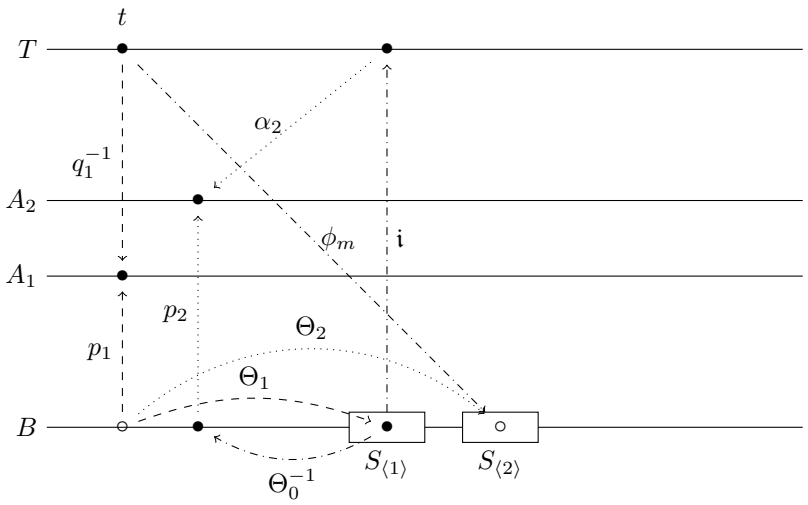

FIGURE 8. Resolving issues 1 and 2

in A_2, no new elements will be enumerated into $S_{\langle 2\rangle}$ (accessible from $S_{\langle\rangle}$ only via Θ_2, as will be formally verified in Corollary A.3).

Recall that the second issue arises when q_2p_2 'copies' Θ_1 (on $S_{\langle\rangle}$). For example, when $q_2p_2 = \mathfrak{i}\,\Theta_1$. Using the naive strategy discussed in Section 3.4, in order to force A_2 to show $\alpha_2\,\mathfrak{i}\,\Theta_1(t)$, we had to enumerate $\mathfrak{i}^{-1}(t)$ into B, thus losing the 'freeness' of t. But now, since $q_2p_2 : S_{\langle\rangle} \to S_1$, α_2 will be defined via $(a1)$. In particular, for elements $t \notin \mathcal{Z}_1$, $\alpha_2\,\mathfrak{i}\,\Theta_1\,\mathfrak{i}^{-1}(t) = p_2\Theta_0^{-1}\Theta_1\,\mathfrak{i}^{-1}(t)$. Thus, in order to force A_2 to show an image for $\alpha_2\,\mathfrak{i}\,\Theta_1\,\mathfrak{i}^{-1}(t)$, we need only enumerate an element into $S_{\langle 1,0\rangle}$. This is done so primitively recursively after an element is enumerated into $S_{\langle 1\rangle}$ as mentioned in Example 3.5. By keeping $S_{\langle\rangle}$ disjoint from $S_{\langle 1,0\rangle}$, we can then maintain the 'freeness' of t while defining $\alpha_2(t)$ primitively recursively.

3.6.1. *The modified basic strategy for N_m.* Recall that in order for the strategy for N_m to succeed, we needed to ensure that $\phi_m = \Theta_e^k\,\mathfrak{i}^{-1}$ for some $k \geq 0$, and in this case we say that ϕ_m has been assigned to α_e. However, that was under the assumption that there is only one R_e, or equivalently, we maintain only one Θ_e. Since we now need to maintain the functions Θ_e for various e, it is entirely possible that ϕ_m could choose to follow some composition of these maps, e.g., $\Theta_2\Theta_0^{-1}\Theta_1$. In this case, we are again unable to diagonalise against ϕ_m as Θ_1, Θ_2 are maintained at the priority of R_1, R_2, and Θ_0^{-1} is maintained for the sake of all α_e. Thus we need a new condition to decide whether or not to assign ϕ_m to α_e.

To start the strategy for N_m, wait for $\phi_m(x)\!\downarrow$ for some $x \in S_{\langle\rangle}$. Employing a similar reasoning as in Remark 3.6, we know that if $\phi_m : S_{\langle\rangle} \mapsto S_\tau$, then $\tau \in \mathcal{Z}_{m-1}$. Thus, we have the following possibilities.

- τ is of colour $e > 0$ (recall Def. 3.4), then we say that ϕ_m is assigned to α_e (or R_e).
- τ is of colour 0, then this corresponds to the case when $\phi_m = \mathfrak{i}^{-1}$.

Once we know whether or not to assign ϕ_m to α_e, then the strategy for N_m is the same as before. If it is assigned to some α_e, then ϕ_m cannot possibly be surjective

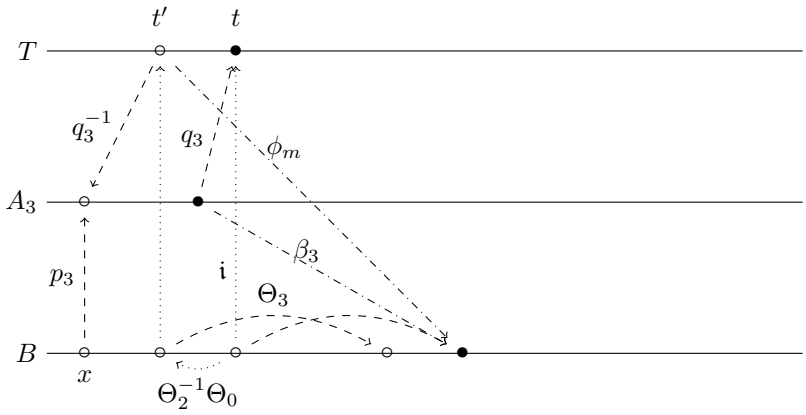

FIGURE 9. Additional complications

(or we can diagonalise against it should it attempt to become surjective). On the other hand, if ϕ_m maps to some interval of colour 0, then by enumerating a free element t into $S_{\langle\rangle}$, no other intervals of colour 0 ever receive new enumerations as long as t remains free. Thus ϕ_m cannot possibly be a primitive recursive isomorphism.

3.7. **Some additional complications.** For a general R_e requirement (of low priority), due to $q_e p_e$ having the option of 'copying' more $\theta_{e'}$ maps, the definition of α_e needs to be generalised as follows. If $q_e p_e : I_k \to I_0$,

$$\alpha_e(x) = \begin{cases} q_e^{-1}(x), & \text{if } x \in \mathcal{Z}_{e-1} \\ p_e \Theta_0^k \, \mathfrak{i}^{-1}(x), & \text{otherwise.} \end{cases}$$

In the definition above, if $k = 0$, we read $p_e \Theta_0^k \, \mathfrak{i}^{-1}(x)$ as $p_e \, \mathfrak{i}^{-1}(x)$.

The main issue that arises is getting $q_e^{-1}(x)$ to show up promptly in A_e for $x \in \mathcal{Z}_{e-1}$. The previous few sections were focused on getting $q_e^{-1}(t)$ to show up in A_e primitively recursively for free elements $t \in S_{\langle\rangle}$. However, $\alpha_e = q_e^{-1}$ for all elements in $\mathcal{Z}_{e-1}$ which contains many more subintervals other than $S_{\langle\rangle}$. In order to force $q_e^{-1}(t)$ to show up in A_e for t in other intervals, we shall utilise two different strategies.

Definition 3.7. *A label σ is good iff for every $i < \frac{|\sigma|}{2}$, $\sigma(2i+1) = 0$. We say a label is bad otherwise.*

On the intervals which are good, we are able to ensure α_e stays defined via pressing A_e with ϕ_m. However, this does not work for elements enumerated into the bad intervals as illustrated in Fig. 9. Suppose that at some stage an element t (possibly free) is enumerated into $S_{\langle\rangle}$. Let $t' = \mathfrak{i} \, \Theta_2^{-1} \Theta_0 \, \mathfrak{i}^{-1}(t)$, which is contained in $S_{\langle 0,2\rangle}$ in T. Suppose further that $\phi_m : S_{\langle\rangle} \mapsto S_{\langle 3,0,3\rangle}$ and $\phi_m : S_{\langle 0,2\rangle} \mapsto S_{\langle 3\rangle}$. Then it could be that $q_3^{-1}(t')\uparrow$ but $\Theta_3 \, \mathfrak{i}^{-1}(t)\downarrow$. This means that $\phi_m(t')\downarrow = \Theta_3 \, \mathfrak{i}^{-1}(t)$ but $\alpha_3(t') = q_3^{-1}(t')\uparrow$; we failed to diagonalise ϕ_m and also failed to keep α_3 primitive recursive.

In order to define α_e primitively recursively on elements in the bad intervals, whenever an element enters some bad interval, all 'related' bad intervals must

also receive elements primitively recursively. Returning to the example illustrated in Fig. 9, if we want to enumerate t' into T, then we must also be prepared to enumerate x into B. But note that as long as we do not enumerate t' into T, we need not enumerate x. The strategy then is to not enumerate any elements into the bad intervals until we are ready to do so.

3.8. **The priority order and injury.** Arrange the requirements in order of priority $N_1, R_1, N_2, R_2, \ldots$. There is a list of conditions which we ensure are maintained during the construction. The specifics can be found in Section 4.2. Roughly speaking, these conditions ensure that each Θ_e is primitive recursive and that $q_e p_e$ and ϕ_e behave 'properly' in the sense of Remark 3.6. Recall that we may need to temporarily make some $\Theta_{e'}$ not primitive recursive in order to diagonalise against $q_e p_e$ or ϕ_e which do not satisfy the desired conditions. This action then *injures* $R_{e'}$, but only for the sake of higher priority R_e or N_e requirements. Thus each $R_{e'}$ can only be injured finitely many times.

At each stage, attend to the highest priority N_m requirement which is not currently assigned to any R_e requirement and not yet satisfied. Then wait for $\phi_m(x)\downarrow$ for some $x \in S_{\langle\rangle}$. Consider the following cases.

Case 1: If $\phi_m(x) \in S_\tau$ for some τ of colour 0, then enumerate a free element t into T and wait for $\phi_m(t)\downarrow$. While waiting, all R_e strategies currently attempting to define α_e are potentially in danger of failing to find a suitable image for $\alpha_e(t)$. However, as explained in the basic strategy for R_e, as long as $q_e^{-1}(t)$ fails to be enumerated into A_e, the $\phi_{m'}$ assigned to α_e will also not have a suitable image for t. That is, $N_{m'}$ will become satisfied as $\phi_{m'}$ fails to be a primitive recursive isomorphism. Since $N_{m'}$ was attended to before N_m, it is also necessarily of higher priority, and thus we can initialise N_m once $N_{m'}$ is satisfied. Once this happens, R_e abandons its current definition of α_e and changes its strategy back to defining β_e by temporarily removing its gap.

Case 2: In this case $\phi_m(x) \in S_\tau$ for some τ of positive colour e. If there are no ϕ currently assigned to R_e, or if there is some lower priority $\phi_{m'}$ currently assigned to R_e, then proceed as follows. Assign ϕ_m to R_e, initialise all lower priority $N_{m'}$ for which $\phi_{m'}$ is assigned to R_e, and introduce a gap into the range of Θ_e (if required). Then as discussed in the basic strategy for N_m, either ϕ_m ends up being not surjective, or it maps some element into the gap (or the image of the gap) of Θ_e. If it does so, we will then be able to diagonalise against ϕ_m.

If there is currently some other higher priority $N_{m'}$ assigned to R_e, then there is already some gap in the range of Θ_e. We also assign N_m to R_e as before but without initialising any $N_{m'}$ of higher priority. Then either N_m is satisfied in the same way as before provided that the gap of Θ_e is never removed. However, it could be that the gap is removed at some stage after N_m has been assigned to R_e thus *injuring* N_m. But recall that the gap of Θ_e is removed only if the highest priority $\phi_{m'}$ assigned to R_e has been diagonalised against. Thus N_m can only be injured this way finitely many times.

By employing a finite injury argument, we obtain that each requirement is satisfied.

4. Labelling and defining subintervals

In preparation for the formal construction, we revisit the system of labels in more detail. The rest of this section will be dedicated to formally defining the labels and some notation regarding the labels which will be required to describe the construction. Recall that the purpose of the labels was to track the behaviour of the Θ_e maps and their inverses on I_0 and I_1. We once again refer the reader to Fig. 7 for an illustration of the labels. An important ingredient in the strategy for R_e was to keep the different subintervals as disjoint as possible. With this in mind, we generate the subintervals in the following way. We first focus only on the maps Θ_0, Θ_1.

(1) Start with the base interval $S_{\langle\rangle}$ in I_0. Applying Θ_0, Θ_1 gives us the following.
$$S_{\langle\rangle} \xrightarrow{\Theta_0} S_{\langle 0\rangle} \quad \text{and} \quad S_{\langle\rangle} \xrightarrow{\Theta_1} S_{\langle 1\rangle}.$$
Recall that these should be disjoint.

(2) Now we apply the inverse maps to the subintervals $S_{\langle 0\rangle}, S_{\langle 1\rangle}$ of I_1. This gives the following.
$$S_{\langle 0\rangle} \sqcup S_{\langle 1\rangle} \xrightarrow{\Theta_0^{-1}} S_{\langle\rangle} \sqcup S_{\langle 1,0\rangle} \quad \text{and} \quad S_{\langle 0\rangle} \sqcup S_{\langle 1\rangle} \xrightarrow{\Theta_1^{-1}} S_{\langle 0,1\rangle} \sqcup S_{\langle\rangle}.$$
In order to keep Θ_0 and Θ_1 order-preserving, we obtain the relative positions as shown in Fig. 7.

(3) Repeat the process recursively to obtain all labels σ with $\max(\sigma) \leq 1$ (recall Notation 3.1). Denote the union of all subintervals contained in I_0 generated this way as $\mathcal{Z}_1$ and the collection of their labels as Z_1. Similarly, let the union of all subintervals contained in I_1 generated by this procedure by $\mathcal{Z}_1{}^\frown 0$ and the collection of their labels be $Z_1{}^\frown 0$.

The idea here is that the maps Θ_0, Θ_1 and their inverses are 'separated' within $\mathcal{Z}_1$. Outside of $\mathcal{Z}_1$, the maps Θ_0 and Θ_1 are now equal. Assuming that $\mathcal{Z}_e$ has been defined, and that $\Theta_{e'}(\mathcal{Z}_e) = \Theta_0(\mathcal{Z}_e)$ for any $e' \leq e$, recursively define $\mathcal{Z}_{e+1}$ as follows.

(1) For each $\sigma \in Z_e$ (contained in I_0), apply the maps Θ_0, Θ_{e+1} to S_σ. This allows us to obtain the following.
$$\mathcal{Z}_e \xrightarrow{\Theta_0} \mathcal{Z}_e{}^\frown 0 \quad \text{and} \quad \mathcal{Z}_e \xrightarrow{\Theta_{e+1}} \mathcal{Z}_e{}^\frown\langle e+1\rangle.$$
We order them by placing $\mathcal{Z}_e{}^\frown 0 < \mathcal{Z}_e{}^\frown\langle e+1\rangle$.

(2) For each $\sigma \in Z_e{}^\frown 0 \sqcup Z_e \frown \langle e+1\rangle$ (contained in I_1), apply the maps Θ_0^{-1} and Θ_{e+1}^{-1} to obtain the following.
$$\mathcal{Z}_e{}^\frown 0 \sqcup \mathcal{Z}_e{}^\frown\langle e+1\rangle \xrightarrow{\Theta_0^{-1}} \mathcal{Z}_e \sqcup \mathcal{Z}_e{}^\frown\langle e+1, 0\rangle \quad \text{and} \quad \mathcal{Z}_e{}^\frown 0 \sqcup \mathcal{Z}_e{}^\frown\langle e+1\rangle \xrightarrow{\Theta_{e+1}^{-1}} \mathcal{Z}_e{}^\frown\langle 0, e+1\rangle \sqcup \mathcal{Z}_e\,.$$
To keep Θ_{e+1}, Θ_0 order-preserving, we order the intervals $\mathcal{Z}_e{}^\frown\langle 0, e+1\rangle < \mathcal{Z}_e < \mathcal{Z}_e{}^\frown\langle e+1, 0\rangle$.

(3) Recursively repeat the previous two steps alternately to generate Z_{e+1} and $Z_{e+1}{}^\frown 0$ in the same way as before. In general, the position of the subintervals of $\mathcal{Z}_{e+1}$ is made up of a ζ-chain[1] of copies of $\mathcal{Z}_e$ arranged as follows.

$$\ldots < \mathcal{Z}_e{}^\frown\langle 0, e{+}1, 0, e{+}1\rangle < \mathcal{Z}_e{}^\frown\langle 0, e{+}1\rangle < \mathcal{Z}_e < \mathcal{Z}_e{}^\frown\langle e{+}1, 0\rangle < \mathcal{Z}_e{}^\frown\langle e{+}1, 0, e{+}1, 0\rangle < \ldots$$

[1]Here and throughout, ζ stands for the usual order on the integers.

The subintervals of $\mathcal{Z}_{e+1}{}^\frown 0$ are arranged in a similar way.

$$\ldots < \mathcal{Z}_e{}^\frown\langle 0, e{+}1, 0, e{+}1, 0\rangle < \mathcal{Z}_e{}^\frown\langle 0, e{+}1, 0\rangle < \mathcal{Z}_e{}^\frown 0 < \mathcal{Z}_e{}^\frown\langle e{+}1\rangle < \mathcal{Z}_e{}^\frown\langle e{+}1, 0, e{+}1\rangle < \ldots$$

Remark 4.1. *Even though the Θ functions seem to be involved in generating the labels, they are there only to provide the intended meaning of the labels. The labels and their relative positions can be generated completely independently of the Θ functions. Thus, when we later define the Θ maps in Section 5 based on the labels, there is no circularity involved.*

In view of the procedure used to generate the subintervals, we introduce a reduction rule on strings $\sigma \in \omega^{<\omega}$ as follows. (Recall Notation 3.1.)

4.1. **Reducing strings.** Recall now that in Example 3.2, some equivalence between different strings was suggested. The purpose of f and * defined below is to formalise this notion. Roughly speaking, the first and second cases in the definition of f reflect that $\Theta_e\Theta_e^{-1}(S_\sigma) = S_\sigma$ and $\Theta_e = \Theta_0$ outside $\mathcal{Z}_e$.

Example 4.2. *Consider the string $\sigma = \langle 1, 2, 0, 1, 2\rangle$. This should correspond to the following subinterval.*

$$\Theta_2\Theta_1^{-1}\Theta_0\Theta_2^{-1}\Theta_1(S_{\langle\rangle}).$$

We can derive the label σ^* using the intended behaviour of Θ maps described before. $S_{\langle\rangle} \xrightarrow{\Theta_1} S_{\langle 1\rangle} \xrightarrow{\Theta_2^{-1}} S_{\langle 1,2\rangle} \xrightarrow{\Theta_0} S_{\langle 1,2,0\rangle} \xrightarrow{\Theta_1^{-1}} S_{\langle 1,2\rangle} \xrightarrow{\Theta_2} S_{\langle 1\rangle}$. The first three steps are easy to obtain, while the fourth follows from the fact that $\langle 1, 2, 0\rangle \notin Z_1$, and hence $\Theta_1^{-1}(S_{\langle 1,2,0\rangle}) = \Theta_0^{-1}(S_{\langle 1,2,0\rangle}) = S_{\langle 1,2\rangle}$.

To define what it means for a string to be reduced, we will define a reduction procedure f on strings. The role of f is to calculate the string naming the range after applying the next Θ-map. We obtain the following computation using the reduction procedure.

$$\sigma_0 = \langle 1\rangle, \quad \sigma_1 = f(\sigma_0{}^\frown 2) = \langle 1, 2\rangle, \quad \sigma_2 = f(\sigma_1{}^\frown 0) = \langle 1, 2, 0\rangle,$$

$$\sigma_3 = f(\sigma_2{}^\frown 1) = \langle 1, 2\rangle, \quad \sigma_4 = f(\sigma_3{}^\frown 2) = \langle 1\rangle.$$

Definition 4.3. *Let $f : \omega^{<\omega} \to \omega^{<\omega}$ be defined as follows.*

- *$f(\langle\rangle) = \langle\rangle$ and for any $e \in \omega$, $f(\langle e\rangle) = \langle e\rangle$.*
- *$f(\sigma{}^\frown e) = \begin{cases} \sigma^-, & \text{if } \sigma(|\sigma|-1) = e \text{ or if } \sigma(|\sigma|-1) = 0 \text{ and } e < \max(\sigma) \\ \sigma{}^\frown 0, & \text{if } \sigma(|\sigma|-1) \neq 0 \text{ and } e < \max(\sigma) \\ \sigma{}^\frown e, & \text{otherwise} \end{cases}$*

The function f will be used as an elementary step in the reduction procedure defined as follows.

Definition 4.4. *We say that a string $\sigma \in \omega^{<\omega}$ is reduced if for every $i \leq |\sigma|$, $f(\sigma \upharpoonright_i) = \sigma \upharpoonright_i$. Given any string σ, we can obtain its reduced form in the following way.*

- *Let $\sigma_0 = \sigma(0)$.*
- *Recursively define $\sigma_{i+1} = f(\sigma_i{}^\frown \sigma(i+1))$.*

Observe that $\sigma_{|\sigma|-1}$ is reduced. For any given string σ, we denote the reduced form (obtained by the described procedure) as σ^*. We also use * as an operator that takes a string to its reduced form.

Given any string σ, in order to compute its reduced form, note that we apply the function f exactly $|\sigma| - 1$ many times. It is also evident that f is primitive recursive. Thus, finding the reduced form σ^* of any string σ is a primitive recursive procedure.

It should follow immediately from the definition of * that for any string, its reduced from is well-defined. Furthermore, one can easily check that (proof in appendix?) every reduced string is contained in $Z_n \cup Z_n^\frown 0$ for some $n \in \omega$. Since we know exactly the structure of $Z_n \cup Z_n^\frown 0$, we arrive at the following observation.

Fact 4.5. *If σ is reduced, then $\sigma = (\sigma_1\sigma_2 \dots \sigma_n\xi)^*$, where $n = \max(\sigma)$, and each σ_i is an even length tuple (possibly empty) of alternating 0's and i's, and $\xi = \langle\rangle$ or 0. Furthermore, such a form $\sigma_1\sigma_2 \dots \sigma_n\xi$ is unique.*

In preparation for the construction and verification, we introduce the following definition as a means to talk about 'adjacency' of labels. This will mainly be used in order to describe how $q_e p_e$ and ϕ_e map different subintervals. Intuitively, if $q_e p_e$ (and ϕ_e) are isomorphisms and they map some S_σ to S_τ, then they should also map the other subintervals 'near' S_σ to the subintervals 'near' S_τ. (See Section 5.1 for details.)

Definition 4.6. *Let $\sigma = (\sigma_1\sigma_2 \dots \sigma_n\xi)^*$ as stated in Fact 4.5, then define*

- $succ_m(\sigma) \coloneqq \begin{cases} (\sigma_1\sigma_2 \dots \sigma_m^\frown m0^\frown \sigma_{m+1} \dots \sigma_n\xi)^*, & \text{if } m \leq n \\ (\sigma_1\sigma_2 \dots \sigma_n^\frown m0^\frown \xi)^*, & \text{otherwise} \end{cases}$
- $succ_m^{-1}(\sigma) \coloneqq \begin{cases} (\sigma_1\sigma_2 \dots \sigma_m^\frown 0m^\frown \sigma_{m+1} \dots \sigma_n\xi)^*, & \text{if } m \leq n \\ (\sigma_1\sigma_2 \dots \sigma_n^\frown 0m^\frown \xi)^*, & \text{otherwise} \end{cases}$

and also

$$Z_m(\sigma) \coloneqq \left\{ succ_i^k(\sigma) \mid \forall 0 < i \leq m, \forall k \in \mathbb{Z} \right\}.$$

In the same fashion as before, we shall use $\mathcal{Z}_m(\sigma)$ to denote the union of subintervals with labels from $Z_m(\sigma)$.

A quick analysis of the definition above allows us to obtain the following fact.

Fact 4.7. *If $\sigma(0) = e$, then for every $n < e$, if $\gamma \in \mathcal{Z}_n(\sigma)$, then $\gamma(0) \leq n$ or $\gamma = \sigma$.*

The reader might have noticed that Z_n corresponds to some ζ^n-chain of labels, where ζ is the order on the integers. The succ functions defined above allow us to navigate this ordering, as illustrated in the example below.

Example 4.8. *Consider the label $\sigma = \langle 1, 3\rangle \in Z_3$. Applying Fact 4.5, we can rewrite $\sigma = \langle 1, 0, 0, 3\rangle^*$. The label that is directly to the right of σ would be given by $succ_1(\sigma) = \langle 1, 0, 1, 3\rangle$. Repeatedly applying $succ_1$ and $succ_1^{-1}$ to σ then generates the ζ-chain of labels around σ, as below.*

$$\dots < \langle 0, 1, 0, 3\rangle < \langle 0, 3\rangle < \langle 1, 3\rangle < \langle 1, 0, 1, 3\rangle < \langle 1, 0, 1, 0, 1, 3\rangle < \dots$$

The ζ-chain that is directly right of the one above would be given by the ζ-chain around $succ_2(\sigma) = \langle 1, 0, 2, 3\rangle$, which we denote as $Z_1(\langle 1, 0, 2, 3\rangle)$. Repeatedly applying $succ_2$ to σ allows us to obtain the labels which form the 'centre' of each successive ζ-chain to the right of $Z_1(\sigma)$. Similarly, applying $succ_2^{-1}$ will allow us to traverse left of $Z_1(\sigma)$. Together, they compose the ζ^2-chain around σ ($Z_2(\sigma)$).

$$\dots < Z_1(\langle 1, 2, 0, 2, 0, 3\rangle) < Z_1(\langle 1, 2, 0, 3\rangle) < Z_1(\langle 1, 3\rangle) < Z_1(\langle 1, 0, 2, 3\rangle) < Z_1(\langle 1, 0, 2, 0, 2, 3\rangle) < \dots$$

In Section 3.7, we briefly mentioned some 'relation' between different bad intervals. Recall that this was in order to ensure that q_e^{-1} converges primitively recursively on elements within bad intervals, so as to keep the definition of α_e safe. The idea was that all 'related' bad intervals should receive elements within primitive recursive delays of each other. The relation is formalised below.

Definition 4.9. *Let σ be a good label. Then define* $\textsc{Bad}(\sigma)$ *as follows.*

$$\gamma \in \textsc{Bad}(\sigma) \iff (\gamma \restriction_i \text{ is good } \implies \gamma \restriction_i \preceq \sigma) \text{ and } \gamma \text{ bad},$$

that is, σ is the maximal good prefix of γ.

Recall that the purpose of the labels is to define the Θ_e maps on I_0. Furthermore, as discussed in the informal description, there are certain properties that we want them to satisfy. We mainly need each Θ_e map to be surjective (Lemma A.5), order-preserving (Lemma A.6) and have 'sufficiently disjoint' images (Corollary A.3). The careful definition of the labels thus far ensure that the desired properties are satisfied. As the proofs that the properties hold are technical and do not contribute to the discussion, we leave them to the Appendix.

4.2. **Assigning labels.** In this section, we assign to each label σ, dyadic rationals for the left and right endpoints of S_σ. Recall that B, T are constructed to be subsets of $\mathbb{Q}_{nd}$. Now we assign dyadic rationals in a primitive recursive way to the endpoints of each subinterval. For each subinterval S_σ, denote the left and right endpoints as σ^0 and σ^1 respectively. At each stage s, and for each σ where $|\sigma| \leq s$ and $\max(\sigma) \leq s$, we will assign dyadic values to σ^0 and σ^1. Recall that we have already decided on the relative positions of the subintervals based on their labels. The procedure here simply assigns a specific dyadic to the endpoints for the sake of defining θ.

Stage 0: Let $S_{\langle\rangle} = \left(\frac{1}{4}, \frac{3}{4}\right)$.

Stage 1: Let $S_{\langle 0\rangle} = \left(\frac{5}{4}, \frac{7}{4}\right)$, and $S_{\langle 1\rangle} = \left(\frac{7}{4}, \frac{15}{8}\right)$.

Stage $s > 1$: We aim to keep the length of each subinterval as 2^{-m} for some $m \in \omega$. Given some interval of length 2^{-m}, we will either split it into two smaller intervals each of length 2^{-m-1} or into three smaller intervals, two of which have length 2^{-m-2} and one of length 2^{-m-1}. In the case that we split it into three, we assume the 'middle' interval is the longest one (though this is not important).

Now let σ be given such that σ^0, σ^1 has not yet been defined. Let $S_\sigma \subseteq I_k = (k, k+1)$ where k is either 0 or 1. Based on the relative position of S_σ (among the other subintervals of I_k), we split into the following cases.

- If $S_\sigma < S_\gamma$ for all $S_\gamma \subseteq I_k$ such that both γ^0 and γ^1 have already been defined, then fix S_γ to be the current left-most subinterval of I_k. If $\mathrm{succ}_1(\sigma) = \gamma$, (recall Def. 4.6) then split (k, γ^0) into two smaller intervals (as described earlier), and let S_σ be the one on the right. Otherwise, split (k, γ^0) into three smaller intervals and let S_σ be the one in the middle.
- If $S_\sigma > S_\gamma$ for all $S_\gamma \subseteq I_k$ such that both γ^0 and γ^1 have already been defined, then we apply a similar procedure as in the case before.
- If there is γ, τ such that the endpoints of $S_\gamma, S_\tau \subseteq I_k$ have already been defined and $S_\gamma < S_\sigma < S_\tau$, then fix S_γ, S_τ to be the right-most and

left-most subinterval satisfying the premise respectively. Now check if $\mathrm{succ}_1(\gamma) = \sigma$ or $\mathrm{succ}_1(\sigma) = \tau$. If both clauses are true, then let $S_\sigma = (\gamma^1, \tau^0)$. If exactly one of the clauses is true, then split (γ^1, τ^0) into two smaller intervals and let S_σ be the one adjacent to the subinterval for which the clause holds true. Finally, if both clauses are false, then split (γ^1, τ^0) into three smaller intervals and let S_σ be the middle one.

It should be evident from the description that each subinterval S_σ has length 2^{-m} for some $m \in \omega$. This property about the length of subintervals will be important in the definition of Θ.

5. Formal Construction

Before presenting the formal construction, we give a brief recap of the notations and objects that have been introduced thus far.

- B, T are structures defined by us to be primitive recursive subsets of $\mathbb{Q}_{nd}$.
- $p_e : B \to A_e$ and $q_e : A_e \to T$ are maps provided by the requirement R_e. We can think of the composition $q_e p_e$ as a map on $\mathbb{Q}_{nd}$.
- $\phi_e : T \to B$ is the map provided by the requirement N_e.
- For each e, Θ_e is a map on $\mathbb{Q}_{nd}$, which is defined by us.
- Z_e is the collection of all labels (reduced strings) with maximum entry e.
- $\mathcal{Z}_e$ is the union of all intervals labelled by labels from Z_e.

5.1. **Diagonalising against $q_e p_e$ and ϕ_e.** In this subsection we restate formally the conditions listed out at the end of Section 3.4 and provide a description of the procedures to be implemented in order to maintain them. The elementary substrategies described here will be used as modules in the formal construction that will be given shortly. As explained earlier, in order for the strategies to work, we need $q_e p_e$ and ϕ_e to 'align' with the various Θ_e maps. In order to specify what we mean, we introduce the following definition.

Definition 5.1. *Let $x \in \mathbb{Q}_{nd}$ be given. Define $\mathcal{O}(x)$ as follows.*

- *$x \in \mathcal{O}(x)$.*
- *If $y \in \mathcal{O}(x)$, then for each $e \in \omega$, $\Theta_e(y)$ and $\Theta_e^{-1}(y)$ (if defined) are both also in $\mathcal{O}(x)$.*

Furthermore, given any x, denote the unique element $y \in S_\sigma \cap \mathcal{O}(x)$ as x^σ.

Recall that the Θ maps are defined on $\mathbb{Q}_{nd}$. Provided that there are currently no gaps in the range of them, for any $x \in \mathbb{Q}_{nd}$ and any σ reduced, x^σ is defined. However, this is not to say that it currently exists in B (or T) as these are constructed stagewise to be primitive recursive substructures of $\mathbb{Q}_{nd}$. In view of this, we will write $x^\sigma\!\downarrow$ to signify that it has already been enumerated (or 'flagged', to be clarified later) into B, and $x^\sigma\!\uparrow$ otherwise. We also note here that $\mathcal{O}(x)$ depends on the definitions of the various Θ_e (and hence ultimately θ_e), which is defined stagewise during the construction. Even though the definition of Θ_e changes, once $x^\tau\!\downarrow$, it will remain the unique element in $\mathcal{O}(x) \cap S_\tau$ regardless of subsequent changes in the definition of Θ_e.

We are now ready to state the conditions which we will enforce on $q_e p_e$ and ϕ_e during the construction. (We only write them for $q_e p_e$ but they shall also be enforced on ϕ_e.)

C1.) For each label σ, and each integer l, there exists a label τ and an integer l', such that for every $b \in \Theta_0^l(S_\sigma)$, $q_e p_e(b) \in \Theta_0^{l'}(S_\tau)$.
  i) Furthermore, $q_e p_e(b) = b^\tau$. In particular, if $b^\tau \uparrow$ and $q_e p_e(b) \downarrow$, then we are able to diagonalise against q_e or p_e.

C2.) If $q_e p_e : \Theta_0^l(S_\sigma) \to \Theta_0^{l'}(S_\tau)$ for some l, l' integers, then all of the following must hold.
  i) If $\tau \in \textsc{Bad}(\gamma)$ for some γ, then $\sigma \in \textsc{Bad}(\gamma)$.
  ii) If τ is of colour k, then σ is either of colour k or 0. (If $k = 0$ then σ must be colour 0.)
  iii) For all n, $q_e p_e : \Theta_0^l(\mathcal{Z}_n(\sigma)) \to \Theta_0^{l'}(\mathcal{Z}_n(\tau))$ (recall Def. 4.6). Furthermore, $(q_e p_e)^{-1}(\mathcal{Z}_{e-1}) = \Theta_0^k(\mathcal{Z}_{e-1})$ for some integer k.

Below are the procedures to diagonalise against $q_e p_e$ should they fail to satisfy one of the conditions above. In what follows, we suppress the Θ_0 maps. Since Θ_0 is maintained to be primitive recursive, any enumeration (or lack thereof) in S_σ and S_τ will be reflected in $\Theta_0^l(S_\sigma)$ and $\Theta_0^l(S_\tau)$ for all integers l respectively. Removing them thus has no real effect on any intricacies of the proof, and we shall do so in order to lighten the notation.

During the construction, every enumeration action is represented by one of A1, A2, A3, A4. The exact details are not important for now but we will put the corresponding actions in parentheses whenever we say to enumerate something in the following discussion. This is to ensure that we do not perform any action which is not allowed by the formal construction.

Condition (C1): Suppose that there is some label σ and two elements $x, y \in S_\sigma$ such that $q_e p_e(x)$ and $q_e p_e(y)$ are in different subintervals. Let $q_e p_e(x) \in S_\tau$. Remember that we suppress the Θ_0. The strategy described below will work even if $q_e p_e(x)$ and $q_e p_e(y)$ are not within the same interval I_k. Without loss of generality, we may assume that $x < y$ and thus $q_e p_e(x) < q_e p_e(y)$. Otherwise one of q_e, p_e must fail to be order preserving.

We consider the following possibilities.

- σ and τ are of different positive colours.
- τ is of colour 0 and σ is not.

In either of these two cases, we are able to enumerate elements (using A3) into S_σ between x, y without having to enumerate any elements into S_τ. This can be done as we are able to keep the subintervals with labels of different colour 'sufficiently disjoint' (see Corollary A.3). Then we can diagonalise against $q_e p_e$ as follows. Wait for some z between x, y such that $q_e p_e(z) \in S_\tau$. This must happen at some finite stage, otherwise $q_e p_e$ cannot be surjective as $q_e p_e(x, y) \cap S_\tau \neq \emptyset$. Once such an element z is found, enumerate sufficiently many elements (using A3) into S_σ between x, z. Then $q_e p_e$ must fail to be a primitive recursive isomorphism since we are able to increase the number of elements between x, z without enumerating any new elements into $q_e p_e(x, z)$.

We may thus assume that one of the following holds.

- σ and τ are both of the same colour.
- σ is of colour 0.

Recall from Def. 5.1 that if σ, τ are of the same colour, or if σ is of colour 0, then for any $z \in S_\sigma$, z^τ must be defined as non-dyadics (but not necessarily

in B). In particular, both x^τ and y^τ are defined. Furthermore, since x, y, have already been enumerated, x^τ, y^τ, can be assumed to exist in B. If $(x^\tau, y^\tau) \cap q_e p_e(x, y) = \emptyset$, then a similar strategy as in the case where σ and τ are of different colours can be employed. This is because enumerations into (x, y) will only cause other enumerations within S_τ to happen in (x^τ, y^τ). We thus assume that $(x^\tau, y^\tau) \cap q_e p_e(x, y) \neq \emptyset$.

Let z be the element x^τ if it is in $q_e p_e(x, y)$ and let $z = y^\tau$ otherwise. By our assumption, $z \in q_e p_e(x, y)$. Wait for $(q_e p_e)^{-1}(z)\downarrow$. As before, this wait must be finite or $q_e p_e$ cannot be surjective. We now have the following.

$$x < (q_e p_e)^{-1}(z) < y \quad \text{and} \quad (q_e p_e(x), z) \subseteq S_\tau.$$

If $z = x^\tau$, then we are able to diagonalise against $q_e p_e$ by enumerating sufficiently many elements (using A1 if σ is of colour 0 and A3 otherwise) into $(x, (q_e p_e)^{-1}(z))$. Doing so would only cause enumerations to the right of x^τ and thus no new elements are enumerated into $(q_e p_e(x), z)$. As a result, $q_e p_e$ cannot be a primitive recursive isomorphism. We may thus assume that $x^\tau \notin q_e p_e(x, y)$, and so $z = y^\tau$. Let $w = (q_e p_e)^{-1}(z)$. Enumerating elements into (x, w) might no longer work now, seeing as $(q_e p_e(x), z) \cap (x^\tau, w^\tau) \neq \emptyset$. We then iterate the idea above once more; wait for $(q_e p_e)^{-1}(w^\tau)\downarrow$, (note that $w^\tau\downarrow$) and provided that $q_e p_e$ is order preserving, we obtain the following.

$$x < (q_e p_e)^{-1}(w^\tau) < w \quad \text{and} \quad q_e p_e((q_e p_e)^{-1}(w^\tau), w) = (w^\tau, z)$$

Now when we enumerate elements (again using either A1 or A3 based on the colour of σ) into $((q_e p_e)^{-1}(w^\tau), w)$, new elements only get enumerated into S_τ to the right of w^τ. Therefore, doing so causes no new enumerations into (w^τ, z) and thus $q_e p_e$ cannot be a primitive recursive isomorphism.

Condition (C1i): We may now assume that $q_e p_e$ satisfies Condition (C1); there exists some τ such that for all $b \in S_\sigma$, $q_e p_e(x) \in S_\tau$. Suppose also that there exists some $b \in S_\sigma$ such that $q_e p_e(b) \neq b^\tau$. Similar to before, we may assume that $b^\tau\downarrow$ (is in B). Otherwise, either σ and τ are of different positive colours, or τ is of colour 0 and σ is not. Then the same strategy used in Condition (C1) can be applied to diagonalise against $q_e p_e$. However, if $q_e p_e(b) \in S_\tau$ and is different from b^τ (which exists), then a similar strategy as the one in Condition (C1) can again be used to diagonalise against $q_e p_e$.

Condition (C2i): Supppose that Condition (C1) is enforced on $q_e p_e$. Let τ be the unique label for which $q_e p_e(b) \in S_\tau$ for each $b \in S_\sigma$. Further suppose that $q_e p_e$ fails to satisfy Condition (C2i); $\tau \in \textsc{Bad}(\gamma)$ but $\sigma \notin \textsc{Bad}(\gamma)$. In order to diagonalise against $q_e p_e$, we enumerate 'sufficiently' many elements into S_σ (using A3), and wait for $q_e p_e$ to converge on them. Since $\sigma \notin \textsc{Bad}(\gamma)$, there is no reason for us to enumerate any new elements into intervals with labels in $\textsc{Bad}(\gamma)$. In other words, we are able to enumerate any number of elements we like into S_σ (using A3), without having to enumerate any elements into S_τ. Then $q_e p_e$ must fail to be a primitive recursive isomorphism.

Condition (C2ii): We use a similar idea as in enforcing Condition (C2i). Observe that if τ is of colour k and σ is of some positive colour $k' \neq k$, then we are able to enumerate elements into S_σ (using A3) without doing so into

S_τ. (Once again this depends on Corollary A.3.) For the same reasons as before, $q_e p_e$ must fail to be a primitive recursive isomorphims.

Condition (C2iii): We proceed inductively on n. The base case is trivial since $\mathcal{Z}_0(\gamma) = S_\gamma$ for any label γ. Suppose inductively that it is true for all $m < n$, that is, if $q_e p_e : S_\sigma \to S_\tau$, then $q_e p_e : \mathcal{Z}_m(\sigma) \to \mathcal{Z}_m(\tau)$.

The aim now is to prove that for all integers l, $q_e p_e : \mathcal{Z}_{n-1}(\mathrm{succ}_n^l(\sigma)) \to \mathcal{Z}_{n-1}(\mathrm{succ}_n^l(\tau))$; recall Def. 4.6. Let $\sigma_l = \mathrm{succ}_n^l(\sigma)$. First consider the case where $l = 1$. Suppose that $q_e p_e : S_{\sigma_1} \to S_{\tau'}$ for some τ'. Applying the inductive hypothesis, we obtain that $q_e p_e : \mathcal{Z}_{n-1}(\sigma_1) \to \mathcal{Z}_{n-1}(\tau')$. Provided that $q_e p_e$ is order preserving and surjective, it must be that $\mathcal{Z}_{n-1}(\tau') = \mathcal{Z}_{n-1}(\mathrm{succ}_n(\tau))$. To see this, recall that $\mathcal{Z}_{n-1}(\mathrm{succ}_n(\tau))$ is the ζ_{n-1}-chain of intervals adjacent to $\mathcal{Z}_{n-1}(\tau)$. Thus, if $q_e p_e$ maps $\mathcal{Z}_{n-1}(\sigma)$ to $\mathcal{Z}_{n-1}(\tau)$ but does not map $\mathcal{Z}_{n-1}(\mathrm{succ}_n(\sigma))$ to $\mathcal{Z}_{n-1}(\mathrm{succ}_n(\tau))$, then it either fails to satisfy Condition (C1) or it is not surjective. An easy induction allows us to conclude that it is true for all integers l. We thus have that if $q_e p_e : S_\sigma \to S_\tau$, then for all n, $q_e p_e : \mathcal{Z}_n(\sigma) \to \mathcal{Z}_n(\tau)$. Observe that this implies that for each k, there exists a unique k' for which $q_e p_e : I_k \to I_{k'}$.

Recall that Θ_e is maintained at the priority of R_e. Thus, $q_e p_e(S_{\langle\rangle}) \in \Theta_0^k(\mathcal{Z}_{e-1})$ for some k. Otherwise, it implies that $q_e p_e$ is copying some composition of maps which include Θ_e or its inverse. Then we would be able to diagonalise against $q_e p_e$ by enumerating elements into $S_{\langle\rangle}$ while preventing enumerations into any interval with labels that have entry e. Therefore, $q_e p_e$ would fail to be a primitive recursive isomorphism. Hence, the only possible preimage for $\mathcal{Z}_{e-1}$ has to be $\Theta_0^k(\mathcal{Z}_{e-1})$ as $q_e p_e$ maps each interval to exactly one interval.

Recall that B, T are constructed as primitive recursive substructures of $\mathbb{Q}_{nd}$. Furthermore, the only difference between B, T is that we sometimes enumerate an element free into T. But observe that the strategies above do not utilise any enumerations of free elements. As such, the same conditions can also be enforced on $\phi_e : T \to B$ using the above strategies.

In addition, during the construction, we will not 'directly' enumerate elements into S_σ for any label σ of length more than 1. However, we remind the reader here that S_σ is a slightly delayed copy of $S_{\sigma(0)}$; any enumeration into $S_{\sigma(0)}$ will be reflected via following Θ maps into S_σ. Therefore, when we say to enumerate an element into S_σ (in the discussion above), we simply enumerate an element into $S_{\sigma(0)}$ and wait for the respective images of the enumerations to show up in S_σ. Note that this wait is not necessarily primitive recursive as we might have to use some Θ_e^{-1} (not primitive recursive for $e > 0$) to reach S_σ from $S_{\sigma(0)}$. However, for the sake of obtaining a permanent satisfaction of some R or N-requirement, the inverse Θ could be made temporarily primitive recursive.

5.2. **The construction.** For convenience, we index the requirements starting from 1. We arrange the requirements in order of priority as follows. $Q_1, R_1, Q_2, R_2, \ldots$. During the construction, each requirement can have one of the following states.

- For R_e:
  (1) Waiting
  (2) Building β
  (3) Building α

(4) Satisfied
- For N_e:
  (1) Waiting
  (2) Prepared
  (3) Satisfied

Recall that we construct B, T as subsets of the non-dyadic rationals. Define $I_k = (k, k+1)$ for each $k \in \mathbb{Z}$, and split I_0 and I_1 into subintervals labelled with reduced strings as described in Section 4.2. Further split $S_{\langle\rangle}$ into even smaller subintervals, each to be assigned to a single R-requirement as follows. Split $S_{\langle\rangle}$ into countably many disjoint subintervals, $S^n_{\langle\rangle}$, where the n^{th} subinterval has length 2^{-n-1}(length of $S_{\langle\rangle}$). For each R_e, where $e > 0$, assign $S^e_{\langle\rangle}$ to it.

Throughout the construction, each non-dyadic will be in one of the following states.

- Not yet flagged to be enumerated.
- Flagged to be enumerated at some stage s.
- Flagged at ∞: to be enumerated at some stage to be decided later. These flags are for elements which we temporarily want to keep out of B, T. But as shall be seen later, during 'catch up' stages, all such elements will be enumerated into the structure.
- Already enumerated into the structure.

An element being flagged for enumeration into the structure is a global requirement and cannot be blocked by any of the R or N-requirements. For convenience whenever an element is flagged with a number less than or equal the current stage, we enumerate the element into the structure immediately. Also, if an element is flagged multiple times by different actions, the flag with the lowest value will always take precedence. During the construction, we keep track of c^e_s and d^e_s for each $e > 0$, defined to be the right-most element in the left half and the left-most element in the right half of $S^e_{\langle\rangle}$, respectively, flagged to be enumerated at stage s.

In order to define actions that will be used during the construction, we will make use of Θ_e, which is defined during the construction. There is no circularity involved as the stagewise definition of Θ_e, as the definition below depends only on the calculus of reduced strings. The exact definition specifying the action of Θ_e on each element will depend on a stage. However, its global behaviour on various intervals is set up in advance, according to the following Definition 5.2. The definition below formalises the idea of the gap (and subsequent images of it) of Θ described in Section 3.2. More specifically, the following definition will be used to describe the part $S^\dagger_{\langle e\rangle}$ of the subinterval $S_{\langle e\rangle}$ which is currently in the range of Θ_e. Note this definition does not actually specify what exactly $S^\dagger_{\langle e\rangle} \subseteq S_{\langle e\rangle}$; this will be defined during the construction.

Definition 5.2. *For σ a label of positive colour, define $S^\dagger_\sigma[s]$ recursively.*

- $S^\dagger_{\langle e\rangle}[s] \coloneqq \Theta_{e,s}(S_{\langle\rangle})$.
- $S^\dagger_{(\sigma e)^*}[s] \coloneqq \Theta_{e,s}(S^\dagger_\sigma[s])$, *where $(\sigma e)^*$ is the reduced form of σe.*

Remark 5.3. *In what follows, we use the notation introduced in Def. 5.1 to identify the elements of $\mathbb{Q}_{nd}$ that will be flagged for enumeration. Since the definition of Θ_e changes based on the stages of the construction, for some fixed x and some*

fixed σ, x^σ could change between different stages of the construction. Thus to avoid ambiguity, we should index x^σ with the stage number. But we also do not wish to be bogged down by the (already) heavy notation, and hence we opted to suppress the index representing stages for x^σ. Instead, whenever we write 'flag x^σ...' we mean flag the current version of x^σ at whatever stage the action is taken. Once an element has been flagged, we never change the definition of Θ_e on them again.

During the construction, there will only be the following types of enumeration actions. Whenever we say 'flag an element in S_σ to be enumerated', it means that the corresponding element in both B and T are flagged except when enumerating free elements. To enumerate free elements, we will use $\hat{S}_{\langle\rangle}$ to emphasize that we flag only some element in T.

Notation 5.4. *To simplify the notation, we will also use* $\textsc{Val}(\sigma)$ *to denote* $\max\{\max(\sigma), |\sigma|\}$ *and refer to this as the* value *of the interval labelled with σ.*

A1. Enumerate an element x in $S_{\langle\rangle}$. Then flag all x^σ for σ of colour 0 with $\textsc{Val}(\sigma)$.

A2. Enumerate a free element x into $\hat{S}_{\langle\rangle}$ (the interval in T corresponding to $S_{\langle\rangle}$). That is, it only enters T and not B. Set the flag for $x \in B$ to be ∞.

A3. Enumerate an element $x \in S_{\langle k\rangle}$ where $k > 0$. Then flag all x^σ for σ of colour k with ∞.
- For good intervals of colour k, the flags at ∞ could be replaced due to actions as defined later.
- For each bad interval of colour k, it must be in $\textsc{Bad}(\gamma)$ for some good γ (see Def. 3.7 and 4.9). Once x^γ has been enumerated, then we are allowed to also flag x^τ in all labels $\tau \in \textsc{Bad}(\gamma)$ with $\textsc{Val}(\tau)$. Unless mentioned otherwise, flagging of x^τ for $\tau \in \textsc{Bad}(\gamma)$ will be assumed to take place whenever x^γ is enumerated. Recall from Section 3.7 that we can put elements into bad intervals as long as Condition (C2i) is maintained.

A4. Do the following.
- For $x \in S_{\langle\rangle}$ flagged with ∞, enumerate x into B. Note that x must already be in T.
- For $x \in S^\dagger_\sigma$ flagged with ∞, replace the flags with $\textsc{Val}(\sigma)$.
- For each e and each $x \in S^\dagger_{\langle e\rangle}$, enumerate $y = \Theta^{-1}_{e,s}(x)$ into $S_{\langle\rangle}$ (if y has yet to be enumerated into B).
- For each σ, and each $z \in S_{\langle\rangle}$, flag z^σ with $\textsc{Val}(\sigma)$.

The purpose of this action is to ensure that all subintervals have the same number of elements (up to some primitive recursive delay). Furthermore, once this action is applied, we should have $B_s = T_s$.

The actions described above will be the only active actions which we perform and actively control during the construction. All other enumeration actions will be performed in the background 'automatically', as described below. These actions are to ensure that the Θ_e-maps are well-defined and primitive recursive (and so is Θ_0^{-1}). Recall that $I_k = (k, k+1)$ for each $k \in \mathbb{Z}$.

B1. Whenever some x enters I_k for $k > 0$, flag $x + 1$ to be enumerated at the next stage.

B2. Whenever some x enters I_k for $k \leq 0$, flag $x-1$ to be enumerated at the next stage.

B3. Whenever some x enters S_σ at stage s, flag both $\Theta_{0,s}(x)$ and $\Theta_{0,s}^{-1}(x)$ to be enumerated at stage $s+1$.

B4. Whenever some x enters S_σ at stage s, if $|\sigma|$ is even, then do the following.

i) If $q_e^{-1}(x)$ is enumerated into A_e and R_e is active in the construction, flag $\Theta_{e,s}(x)$ to be enumerated at stage $s+1$. As long as $q_e^{-1}(x)$ is not enumerated into A_e, we can delay enumerating $\Theta_{e,s}(x)$ into B.

ii) If R_e is satisfied or building β, then we do not need to wait for $q_e^{-1}(x)$ to be enumerated into A_e. Once x is enumerated, flag $\Theta_{e,s}(x)$ to be enumerated at stage $s+1$ directly.

Throughout the construction, we shall also maintain a queue of pending tasks; let it be empty at stage 0. A task in the construction would correspond to a diagonalisation procedure described in Section 5.1. If we discover some $q_e p_e$ or ϕ_e to fail one of the enforced conditions, then we would add the task to diagonalise against $q_e p_e$ or ϕ_e respectively to the queue of pending tasks. We also say that a task is *ready* if it is not currently waiting for surjectivity of the map ($q_e p_e$ or ϕ_e) it is trying to diagonalise against.

Now we proceed to the formal construction.

Stage 0: Set all R_e and N_e to the waiting state. Define for each e, the function $\Theta_{e,0} : \mathbb{Q}_{nd} \to \mathbb{Q}_{nd}$ as follows. For any $x \notin I_0$, $\Theta_{e,0}(x) = x+1$. In fact, for every s and $x \notin I_0$, $\Theta_{e,s}(x) = x+1$. In order to define $\Theta_{e,0}$ on I_0, we consider each subinterval separately. For each label σ, let $\tau = (\sigma e)^*$, which is the reduced form of σe. (Recall that σ^0, σ^1 are the left and right dyadic endpoints of S_σ respectively.)

$$\Theta_{e,0} : x \in S_\sigma \mapsto \frac{x-\sigma^0}{\sigma^1-\sigma^0}(\tau^1-\tau^0)+\tau^0 \in S_\tau.$$

Note that by the careful choice of endpoints, $\Theta_{e,0}(x)$ is dyadic iff x is dyadic. Pick the non-dyadic with the least index, $x \in S_{\langle\rangle}$, and flag it using action A1.

Stage s: If there is some task currently in progress, then continue attending to the task. Otherwise, pick the non-dyadic with the least index in $S_{\langle\rangle}$ and enumerate it using action A1. Also, for each R_n currently building α, pick the non-dyadic with the least index in the gap of $\Theta_{n,s}$ and enumerate it with action A3. If some pending task is ready, then remove it from the queue and act for it. Otherwise, pick the least e such that one of the following holds.

(1) R_e is waiting.

(2) One of the conditions (C1), (C1i), (C2i), (C2ii) or (C2iii) (see Section 5.1) fails for $q_e p_e$. For convenience we shall henceforth refer to these conditions (C1), (C1i), (C2i), (C2ii), and (C2iii), as the *enforced conditions*.

(3) N_e is waiting.

(4) One of the enforced conditions does not hold for ϕ_e.

In each of the above cases, we do the following.

Case 1: Change the state of R_e to building β. Proceed to the next stage.

Case 2: If $q_e p_e$ fails one of the enforced conditions, then begin the task of diagonalising against $q_e p_e$ according to the strategy in Section 5.1. Recall that in some of the cases, the strategy waits for the preimage of $q_e p_e$ on some element in the structure. If a wait is required, we add this task to the queue of pending tasks and proceed with the construction. If the preimage is found, we say that this task is 'ready'. (Otherwise the task stays pending forever.)

Once the task begins, we issue no new instructions except for the actions dictated by the diagonalisation strategies (of Section 5.1) until either it is complete, or needs to wait again for some preimage of $q_e p_e$ to be revealed. If $q_e p_e$ was observed to fail, then set R_e to satisfied. Otherwise, add the task back to the queue, where it stays pending until it is once again ready. In any case, reset all statuses of $R_{e'}, N_{e'}$ for each $e' > e$ which are not satisfied back to the waiting state. For each $R_{e'}$ in state building α that was initialised, close the gaps in $\Theta_{e',s+1}$ (see Case 4). Once a task is complete, perform action A4 and proceed to the next stage.

Case 3: Pick some element $t \in \hat{S}_{\langle\rangle}$ and compute $\phi_e(t)$. If ϕ_e satisfies all of the desired conditions, particularly Condition (C1), then there exists some τ such that for all $t' \in S_{\langle\rangle}$, $\phi_e(t') \in S_\tau$. Thus computing the image just for t will reveal τ. The possibilities are as follows.

(1) $\phi_e : S_{\langle\rangle} \mapsto S_\tau$ where τ has positive colour n less than e.
(2) $\phi_e : S_{\langle\rangle} \mapsto S_\tau$ where τ has colour 0.

If τ has positive colour n, provided that R_n is active and in state building β, we say that ϕ_e is *assigned* to R_n. Then change the state of R_n to building α and the state of N_e to prepared. Then introduce a gap in the range of $\Theta_{n,s+1}$ as follows. Choose for each $i < 5$, $y_i \in (c_s^n, d_s^n)$ dyadics such that y_2 is the midpoint of $S^n_{\langle\rangle}$, and for some $m > s$, $y_{i+1} - y_i = 2^{-m}$. Also let $z_i = \Theta_{n,s}(y_i)$. Then define

$$\Theta_{n,s+1}(x) \coloneqq \begin{cases} \frac{x-y_0}{y_2-y_0}(z_1 - z_0) + z_0 & \text{if } x \in (y_0, y_2) \\ \frac{y_4-x}{y_4-y_2}(z_4 - z_3) + z_3 & \text{if } x \in (y_2, y_4) \\ \Theta_{n,s}(x) & \text{otherwise} \end{cases}$$

If R_n was already building α, then we also say ϕ_e is assigned to R_n and change the state of N_e to prepared, but do not change the definition of $\Theta_{n,s}$; let $\Theta_{n,s+1} = \Theta_{n,s}$.

If R_n is inactive, then act to diagonalise against ϕ_e as follows. Enumerate sufficiently many elements x into $S^0_{\langle\rangle}$ using action A1, and block action B4i even if $q_n^{-1}(x)$ shows up in A_n. As before, issue no new instructions until either $\phi_e \uparrow$ or it fails one of the conditions being enforced on it. Then apply action A4 and add a new task to the queue if ϕ_e was not witnessed to fail as a primitive recursive isomorphism but instead fails to satisfy one of the desired conditions.

If instead we have that τ is of colour 0, then enumerate an element free into $\hat{S}^0_{\langle\rangle}$ according to action A2 at stage $s + 1$.

For stages t while an element x remains free in S^0_ϵ, maintain only actions B3 and B4. As a result, only good intervals will receive enumerations during such stages where some element remains free. Once $\phi_{e'}(x)$ fails to either be quick or order-preserving for some $e' \leq e$ and $Q_{e'}$ prepared, then we apply action A4, and proceed to the next stage.

Case 4: If ϕ_e fails one of the enforced conditions, then we can diagonalise against ϕ_e as described in Section 5.1. Do the same thing here as in Case 2.

If in addition, ϕ_e is currently the highest priority map assigned to some R_n requirement, then after diagonalising against it, we need to close the gap of $\Theta_{n,s}$. Choose for each $i < 5$, $y'_i \in (c^n_s, d^n_s)$ dyadics such that $y'_2 = y_2$ and for some appropriate large m, $y'_{i+1} - y'_i = 2^{-m}$. Also let $z'_i = \Theta_{n,s}(y'_i)$. Then define

$$\Theta_{n,s+1}(x) \coloneqq \begin{cases} \frac{x-y'_0}{y'_1-y'_0}(z_1 - z'_1) + z'_1 & \text{if } x \in (y'_0, y'_1) \\ \frac{x-y'_1}{y'_3-y'_1}(z_3 - z_1) + z_1 & \text{if } x \in (y'_1, y'_3) \\ \frac{x-y'_3}{y'_4-y'_3}(z'_4 - z_3) + z_3 & \text{if } x \in (y'_3, y'_4) \\ \Theta_{n,s}(x) & \text{otherwise} \end{cases}$$

Then we use action A4 at the beginning of stage $s+1$. In particular, for each $y \in (z_1, z_3)$ flagged to be enumerated, enumerate $\Theta^{-1}_{n,s+1}(y) \in S^n_{\langle\rangle}$, and flag $\left(\Theta^{-1}_{n,s+1}(y)\right)^\sigma$ at stage $s+1$ with $\textsc{Val}(\sigma)$.

Once the gap of Θ_n is closed, set all N-requirements of lower priority that are not in the satisfied state back to the waiting state.

6. Verification

Before we check that the requirements are satisfied, we prove some preliminary lemmas about the enumerations of elements and argue that the injury is finite.

Recall that all labels are reduced strings. For an interval J, we write $|J[s]|$ to denote its cardinality at stage s. (We however usually suppress s if there is no danger of ambiguity.)

Lemma 6.1. *At any stage s, if $\sigma \prec \gamma$ and both σ, γ are labels of the same colour, then $|S_\gamma| \leq |S_\sigma|$. Furthermore, after action A4 is applied, for all σ, $|S^\dagger_\sigma| = |S_{\langle\rangle}|$.*

Proof. First notice that all enumerations are due to either action A1, A2 or A3. Action A4 only replaces flags which are already there while actions B1-B4 flags new elements only when some x previously entered the structure. Since A1, A2 and A3 only ever place elements into $S_{\langle\rangle}$ or S_k, then for any σ where $|\sigma| > 1$, if some element is flagged to be enumerated into S_σ it must be due to one of A4, B3 or B4, as actions B1, B2 only enumerates elements into I_k for $k \neq 0, 1$.

Now let γ such that $|\gamma| > 1$ be given. Further suppose that some element x in S_γ has just been flagged to be enumerated at stage s and that γ is the minimal string such that x^γ has been flagged to be enumerated at some finite stage. However, since $|\gamma| > 1$, x must have been flagged due to either actions B3 or B4. It cannot be action A4, as that would have flagged x^{γ^-} with $\textsc{Val}\,(\gamma^-)$ which cannot be more than $\textsc{Val}(\gamma)$. However, if it is due to actions B3 or B4, then let $x_1 \coloneqq x^{\gamma_1}$ be the

element enumerated which triggered B3 or B4. Note that $\gamma_1 \succ \gamma$, since γ is assumed to be the minimal. Furthermore, it must be that x_1 was enumerated at stage $s-1$ in order for B3 or B4 to flag x at s. Then we repeat the argument to obtain $x_{i+1} \coloneqq x^{\gamma_{i+1}}$ which was enumerated at stage $s-i-1$ to cause the enumeration of x_i. By Corollary A.3, and the fact that γ is minimal, it must be that $\gamma_n \succ \gamma$ for all n. In particular $\gamma_s \succ \gamma$, that is, $|\gamma_s| > 1$ but $x_s \in S_{\gamma_s}$ was flagged to be enumerated at stage 0, which cannot be the case. Therefore, $x \in S_\gamma$ can only be flagged at some finite value if x^{γ^-} was flagged at an earlier stage. Therefore, if γ is of colour k, it only receives an element x after all σ such that $\sigma \prec \gamma$ and also of colour k has received x^σ. We then have that $|S_\gamma| \leq |S_\sigma|$.

Finally, when A4 is applied, for any element $x \in S^\dagger_{\langle n \rangle}$ where n is positive, $\Theta^{-1}_{n,s}(x)$ is enumerated into the structure. And subsequently, $\big(\Theta^{-1}_{n,s}(x)\big)^\sigma$ is also flagged with $\textsc{Val}(\sigma)$ by A4. Thus $|S_{\langle\rangle}| = |S^\dagger_\sigma|$. □

Lemma 6.2. *Each task in the construction, once initiated, only lasts for finitely many stages before the respective requirement is met. If a task stays pending forever, then the requirement corresponding to the task is satisfied.*

Proof. Each task in the construction corresponds to some diagonalisation procedure explained in Section 5.1. If a task is pending, then it means that we are waiting for the preimage of some element under $q_e p_e$ or ϕ_e for some e. Therefore, if a task stays pending forever, then $q_e p_e$ (or ϕ_e) is not surjective, thus satisfying the requirement the task is for.

Suppose at some stage a task becomes ready. A task is added to the queue iff some $q_e p_e$ or ϕ_e fails one of the enforced conditions. It is evident from the description that only finitely many actions are applied to obtain a diagonalisation against $q_e p_e$ (or ϕ_e). It thus remains to check that each individual action takes only a finite amount of time.

During the construction, we only actively control the enumeration of elements into either $S_{\langle\rangle}$ (A1) or one of $S_{\langle e \rangle}$ for some $e > 0$ (A3); the other enumerations (B1)-(B4) happen in the background. All other flags occur as a result of these enumerations. In particular, we show that when we wish to diagonalise against $q_e p_e$ (or ϕ_e), the respective enumerations eventually happen in S_σ, (where σ is as in Section 5.1). The crucial cases then are actions A3 and actions B4i. All other actions provide a finite flag to elements.

In action A3, we are allowed to block enumerations into $\textsc{Bad}(\gamma)$ by leaving the flags at ∞. However, since we are actually attempting to get enumerations into S_σ, if $\sigma \in \textsc{Bad}(\gamma)$, then we would not leave the flags at ∞ for the elements in S_σ, provided enumerations happen in S_γ first. Suppose σ is good. We now proceed by induction on $|\sigma|$ to prove that $x^\sigma\!\downarrow$, i.e., enters B at some stage.

The cases for $|\sigma| = 0, 1$ are trivial. Suppose inductively that after some finite delay from when x is enumerated into $S_{\langle\rangle}$ or $S_{\langle e \rangle}$, $x^\sigma\!\downarrow$ for σ of colour e and length $< n$. If n is even, then the last bit of σ is necessarily 0. Therefore, by action B3, once $x^{\sigma^-}\!\downarrow$ (recall Notation 3.1), $x^\sigma = \Theta_0^{-1}(x^{\sigma^-})$ would be flagged to be enumerated. If n is odd, then the last bit of σ must be k for some $k > 0$. There are two further possibilities. When $x^{\sigma^-}\!\downarrow$, if R_k is not in state building α, then by action B3i, $x^\sigma = \Theta_k(x^{\sigma^-})$ would have been flagged with some finite value. We may thus suppose that R_k is in state building α. In order to maintain α for R_k, we thus need

to wait for $q_k^{-1}(x^{\sigma^-})\downarrow$. But this could potentially never happen. However, we note here that since R_k is in state building α, there must be some ϕ which is currently assigned to it. Referring the reader to Lemma 6.14, if $q_k^{-1}(x^{\sigma^-})\uparrow$, then ϕ fails to be a primitive recursive isomorphism in finite time. Once this happens, R_k changes state back to building β as mentioned in Case 4 of the formal construction. We are then allowed to enumerate $x^\sigma = \Theta_k(x^{\sigma^-})$ into B. □

Lemma 6.3. *B, T are dense linear orders with no endpoints.*

Proof. Since we add the least indexed dyadic into $S_{\langle\rangle}$ and $S_{\langle e\rangle} \setminus S^\dagger_{\langle e\rangle}$ infinitely often, all such intervals are dense. Applying the lemma above with the fact that all other subintervals are simply copies of either $S_{\langle\rangle}$ or $S_{\langle e\rangle}$ for some e, we obtain the lemma. □

Lemma 6.4. *For each $e > 0$, R_e and N_e are initialised at most finitely many times.*

Proof. During the construction, a requirement is only ever initialised if some requirement of higher priority switches it status to satisfied, as in Cases 2 and 4 of the formal construction. Furthermore, once a requirement switches it status to 'satisfied', it never again becomes an active requirement. That is, for each fixed n, there are only at most finitely many times which it can be initialised as there are only finitely many requirements of higher priority. □

Thus for each requirement, there is some large enough finite stage s for which it is never again injured. We are now ready to prove that each requirement is satisfied.

Lemma 6.5. *If N_e changes to state 'satisfied' during the construction, then N_e is satisfied.*

Proof. During the construction, a requirement N_e changes its state to 'satisfied' only if one of the following happens.

- In Case 3 of the formal construction; if $\phi_e(S_{\langle\rangle}) \subseteq S_\tau$ where τ is of colour 0 or n for some inactive R_n.
- In Case 4 of the formal construction where ϕ_e fails one of the enforced conditions.

In Case 3, when N_e picks some inactive R_n, then elements x are enumerated into $S^0_{\langle\rangle}$ while action B4i (for Θ_n) is blocked. Meaning that x^τ for τ of colour n never shows up until A4 is applied, which only happens until after ϕ_e fails. Similarly, in the case that $\phi_e(S_{\langle\rangle}) \subseteq S_\tau$ where τ is of colour 0, action A2 is used to enumerate a free element x into $\hat{S}^0_{\langle\rangle}$, that is to say, x^τ for τ of colour 0 is never flagged to be enumerated at a finite stage until action A4 is executed. Thus ϕ_e must fail one of the enforced conditions or we find some witness that ϕ_e is not a primitive recursive isomorphism.

In Case 4, if ϕ_e fails one of the enforced conditions, then the construction acts according to the strategy as described in Section 5.1 and thus guarantees that ϕ_e fails as a primitive recursive isomorphism. □

Lemma 6.6. *For each e, N_e is satisfied.*

Proof. By Lemma 6.4, after some stage s, N_e is never again initialised. Let stage s be the stage at which N_e is initialised for the last time. Further assume that

during the construction, N_e never changes its state to satisfied, otherwise by the previous lemma, N_e would be satisfied. We may thus assume that ϕ_e satisfies all of the enforced conditions.

At some stage $t \geq s$, the strategy N_e must be the highest priority requirement in the waiting state, and thus the construction must act for it. Then $\phi_e(S_{\langle\rangle}) \subseteq S_\tau$, for some τ. We may assume here that τ is of colour $n > 0$, otherwise N_e will be satisfied as shown in the previous lemma. There are then two possibilities.

Case 1: There are no other N-requirements currently assigned to R_n. Then as described in Case 3 of the formal construction, N_e is assigned to R_n and a gap is introduced in the range of $\Theta_{n,t}$. By assumption, N_e is never again initialised after stage s, and so it must be that Θ_n never again changes its definition, and the gap of Θ_n is not surjective. Since ϕ_e satisfies all of the enforced conditions, it implies that $\phi_e(\hat{S}_{\langle\rangle}) \subseteq S^\dagger_\tau \subsetneq S_\tau$, and so, ϕ_e cannot be surjective.

Case 2: There is some other N-requirement already assigned to R_n, say $N_{e'}$. We claim that $e' < e$. When N_e is initialised, all lower priority N-requirements are also initialised. Since N_e is the current highest priority N-requirement in the waiting state, it must thus be attended to first. Thus any other N-requirement already assigned to R_n after stage s must all be of higher priority than N_e. By assumption, since N_e is never again initialised after stage s, then we can once again assume that the gap in $\Theta_{n,t}$ is never closed, as no higher priority N-requirement changes its state to satisfied after stage s. In particular, the highest priority $N_{e'}$ requirement currently assigned to R_n never changes state after stage s. Then it must be that $\phi_e(\hat{S}_{\langle\rangle}) \subseteq S^\dagger_\tau \subsetneq S_\tau$. Thus ϕ_e cannot be surjective.

Therefore, each N_e requirement either changes its state to satisfied at some finite stage, in which case it must be satisfied, or ϕ_e is assigned to some R_n requirement for cofinitely many stages and fails to be surjective. □

We split the proof that R_e is satisfied into two subsections, addressing the β and α strategy respectively.

6.1. **Defining β_e.**

Definition 6.7. *Let $\Theta_e = \bigcup_s \Theta_{e,s}$, and define $\beta_e : A_e \to B$ as follows.*

$$\beta_e(x) = \Theta_e\, \mathfrak{i}^{-1}\, q_e(x).$$

We now show that β_e is primitive recursive and order preserving.

Lemma 6.8. *Given $x \in B_s$, it is primitive recursive to find σ such that $x \in S_\sigma$.*

Proof. Observe that during the construction, for any k and any $x \in S_k$, x is flagged at with some value at least k. Then applying a simple induction and Lemma 6.1 allows us to easily conclude that at each stage s, the only intervals S_σ that has elements enumerated into it are such that $\textsc{Val}(\sigma) \leq s$. Observe that there are at most s^s many such intervals. Furthermore, given any label, it is also primtiive recursive to find its endpoints (see the procedure in Section 4.2). Thus, given any $x \in B_s$, it is primitive recursive to find σ such that $x \in S_\sigma$. □

Lemma 6.9. *If there are infinitely many stages s such that R_e is in state building β at stage s, then β_e is a surjective isomorphism.*

Proof. Suppose that q_e is surjective and order-preserving. Since $\mathfrak{i}^{-1}$ is clearly an isomorphism, then it suffices to check that Θ_e is surjective and order-preserving. By the definition above, $\Theta_e : x \in \mathbb{Q}_{nd} \setminus I_0 \mapsto x + 1 \in \mathbb{Q}_{nd} \setminus I_1$ which is an isomorphism. Thus it remains to check that $\Theta_e \restriction_{I_0}$ is order preserving and surjective.

Since $\Theta_e \restriction_{I_0}$ maps each subinterval to exactly one other subinterval, we can think of it as a map on the labels. By Lemmas A.5 and A.6, $\Theta_e \restriction_{I_0}$ is both order preserving and surjective as a map on the labels. From the definition in Section 5, it is easy to see that $\Theta_{e,0} \restriction_{I_0}$ is order preserving and surjective on each subinterval. Furthermore, for any $x \notin S^e_{\langle\rangle}$, $\Theta_{e,s}(x) = \Theta_{e,0}(x)$. Therefore, we need only check that $\Theta_e \restriction_{S^e_{\langle\rangle}}$ is order preserving and surjective. By the assumption, there are infinitely many stages during which R_e is in state building β. Hence, a gap is placed and removed in the range of Θ_e infinitely often. We refer the reader to Cases 3 and 4 of Section 5 for the exact definitions of placing and removing gaps in the range of Θ_e. Each time a gap is placed, it is an interval centered at a fixed dyadic y_2 (as in the definition), with radius 2^{-m} for some m larger than the current stage number. When the gap is removed (in Case 4) the definition of $\Theta_{e,s}$ becomes surjective. Now for a fixed $y \in S_{\langle e\rangle}$, there must be some t large enough for which $|y - y_2| > 2^{-t}$. Then for any stage $t' > t$, y will never again be in the gap of Θ_e.

Finally, to prove that Θ_e is order preserving, it suffices to check only for elements $x, y \in S_{\langle\rangle}$. We proceed by induction. Since $\Theta_{e,0}$ is order preserving (on elements), then suppose inductively that $\Theta_{e,s}$ is order preserving. We consider the cases when $\Theta_{e,s+1} \neq \Theta_{e,s}$.

Case 1: R_e swapped to building α at stage s (Case 3 of the formal construction). Let $x < y$ be given. If $x, y \notin (y_0, y_4)$ as in the definition of $\Theta_{e,s+1}$, then $\Theta_{e,s+1}(x) = \Theta_{e,s}(x)$ and $\Theta_{e,s+1}(y) = \Theta_{e,s}(y)$. Applying the inductive hypothesis allows us to conclude that $\Theta_{e,s+1}(x) < \Theta_{e,s+1}(y)$.
By definition of $\Theta_{e,s+1}$, $z \in (y_0, y_4)$ iff $\Theta_{e,s+1}(z) \in (z_0, z_4)$. That is, if exactly one of $x, y \in (y_0, y_4)$, then we also have that $\Theta_{e,s+1}(x) < \Theta_{e,s+1}(y)$. Finally, it is easy to check that if both $x, y \in (y_0, y_4)$ we also have the desired property.

Case 2: R_e swapped to building β at stage s (Case 4 of the formal construction). A similar analysis as before can be done to show that $\theta_{e,s+1}$ is also order-preserving.

Thus by induction, for each $s \in \omega$, $\Theta_{e,s}$ is order-preserving. □

As long as action B4i and B4ii is maintained for R_e, β_e must be primitive recursive. During the construction, when B4i or B4ii is blocked, R_e is initialised. However, by Lemma 6.4, there is a final stage where this happens. The version of β_e built after such a stage will thus be primitive recursive. Notice also that the primitive recursiveness of β_e is completely independent of the surjectivity of β_e. We thus obtain the following.

Corollary 6.10. *If there are infinitely many stages for which R_e is in state building β, then β_e is a primitive recursive surjective isomorphism.*

6.2. **Defining α_e.**

Definition 6.11. *Suppose $q_e p_e : I_k \to I_0$ for some $k \in \mathbb{Z}$, then let $\alpha_e : T \to A_e$ be as follows.*

$$\alpha_e(x) = \begin{cases} q_e^{-1}(x) & \text{if } x \in \mathcal{Z}_{e-1} \\ p_e \Theta_0^k \mathfrak{i}^{-1}(x) & \text{otherwise.} \end{cases}$$

Given any $x \in T_s$, applying the same reasoning in Lemma 6.8, it is primitive recursive to tell which $S_\sigma \ni x$. Since $\sigma \in \mathcal{Z}_{e-1}$ iff $\max(\sigma) \leq e-1$, it is thus primitive recursive to tell which case of α_e should be applied to x. To prove that α_e is primitive recursive, it remains to check that $\alpha_e(x)$ is enumerated into A_e within a primitive recursive delay after x is enumerated into T.

Lemma 6.12. *Let $\tau \in \mathcal{Z}_{e-1}$ be arbitrarily chosen, and suppose that $q_e p_e$ satisfies the enforced conditions. Then it is primitive recursive to find σ such that $q_e p_e(S_\sigma) = S_\tau$.*

Proof. For notational convenience, in the following proof, we treat $q_e p_e$ as a map on the labels. Since $q_e p_e$ satisfies all of the enforced conditions, then this induced map is well-defined. Let $q_e p_e(\langle\rangle) = \gamma$. By Condition (C2iii), it must be that for all $k \in \mathbb{Z}$, $q_e p_e\left(\mathrm{succ}_1^k(\langle\rangle)\right) = \mathrm{succ}_1^k(\gamma)$. If $\tau \in Z_1(\gamma)$, since we can always promptly find k such that $\mathrm{succ}_1^k(\gamma) = \tau$, then take $\sigma = \mathrm{succ}_1^k(\langle\rangle)$.

Now suppose that $\tau \in Z_i(\lambda)$, and let δ be such that $q_e p_e(\delta) = \lambda$. We aim to find δ' such that $\tau \in Z_{i-1}((q_e p_e)(\delta'))$. Given λ and τ, using Fact 4.5, rewrite them as follows.

- $\lambda = (\lambda_1 \lambda_2 \dots \lambda_i \dots \lambda_{e-1})^*$
- $\tau = (\tau_1 \tau_2 \dots \tau_i \dots \tau_{e-1})^*$

If $\tau \in Z_{i-1}(\lambda)$, then take $\delta' = \delta$. Otherwise, search for k_i such that $\mathrm{succ}_i^{k_i}(\lambda_i) = \tau_i$, which can always be found promptly. Then take $\delta' = \mathrm{succ}_i^{k_i}(\delta)$, and notice that by Condition (C2iii), $\tau \in Z_{i-1}(q_e p_e(\delta'))$.

Observe that applying the procedure above starting with $\delta = \langle\rangle$ and $\lambda = \gamma$ allows us to obtain σ. Furthermore, since each k_i can be found primitively recursively in τ, γ, the quantity $||\sigma| - |\tau||$ is also primitive recursive in τ, γ. □

Lemma 6.13. *If ϕ_n satisfies the enforced conditions and $\phi_n : S_{\langle\rangle} \mapsto S_\tau$, where τ has colour $e > 0$, then for all $\sigma \in Z_{e-1}$ good, $\phi_n(S_\sigma) = S_{\sigma\tau}$.*

Proof. Let ϕ_n be such that it satisfies the enforced conditions, $\phi_n(S_{\langle\rangle}) = S_\tau$ and $\tau(0) = e > 0$. Then by Condition (C2iii), it must be that $\phi_n : \mathcal{Z}_{e-1} \mapsto \mathcal{Z}_{e-1}(\tau)$. We proceed by induction on the colour of $\sigma \in Z_{e-1}$. Let σ good with colour $e-1$ be given. Then $\sigma = \mathrm{succ}_{e-1}^k(\langle\rangle)$ for some $k > 0$. Again by Condtion (C2iii), $\phi_n : \mathcal{Z}_{e-2}(\sigma) \mapsto \mathcal{Z}_{e-2}\left(\mathrm{succ}_{e-1}^k(\tau)\right)$. Since $\tau(0) = e$, and $k > 0$, then $\gamma = \mathrm{succ}_{e-1}^k(\tau)$ is good and also has colour $e-1$. Applying Fact 4.7 allows us to conclude that γ is the only label in $Z_{e-2}\left(\mathrm{succ}_{e-1}^k(\tau)\right)$ with colour $e-1$. That is, $\phi_n(S_\sigma) = S_\gamma$, otherwise ϕ_n cannot possibly satisfy Condition (C2iii). Furthermore, since $\tau(0) = e$, by the definition of succ (Def. 4.6), it is easy to see that $\gamma = \sigma\tau$ as desired.

Now inductively suppose that $\phi_n(S_\sigma) = S_{\sigma\tau}$ for any good $\sigma \in \mathcal{Z}_{e-1}$ of colour $e-i$. Let σ good of colour $e-i-1$ be given. It must be in some $Z_{e-i-1}(\gamma)$ where γ is $\langle\rangle$ or some good label with colour strictly above $e-i-1$. Then there must be some k such that $\mathrm{succ}_{e-i-1}^k(\gamma) = \sigma$ for some $k > 0$. Since γ is good, by the inductive hypothesis $\phi_n : S_\gamma \mapsto S_{\gamma\tau}$. Furthermore, as ϕ_n is assumed to satisfy Condition (C2iii), then

$\phi_n : \mathcal{Z}_{e-i-2}(\gamma) \mapsto \mathcal{Z}_{e-i-2}(\gamma\tau)$. In fact, $\phi_n : \mathcal{Z}_{e-i-2}(\sigma) \mapsto \mathcal{Z}_{e-i-2}\left(\text{succ}^k_{e-i-1}(\gamma\tau)\right)$. Finally, applying Fact 4.7 allows us to conclude that the only label of colour $e-i-1$ in $Z_{e-i-2}\left(\text{succ}^k_{e-i-1}(\gamma\tau)\right)$ is $\text{succ}^k_{e-i-1}(\gamma\tau)$, which is $\sigma\tau$. Thus, $\phi_n : S_\sigma \mapsto S_{\sigma\tau}$. □

Lemma 6.14. *α_e is a primitive recursive isomorphism if there exists s such that for all $t > s$, R_e is in state 'building α'.*

Remark 6.15. *Observe that the proof below is done without dependence on Lemma 6.2 thus removing any potential circularity. This is because ϕ must first be satisfying the enforced conditions in order for it to be assigned to some R. Once it is assigned, ϕ potentially only fails in some finitary way, thus causing R to change its state back to building β as required in the proof of Lemma 6.2.*

Proof. Suppose that there exists s such that for all $t > s$, R_e is in state building α. We first check that α_e is primitive recursive. In the definition of α_e, the second line utilizes the functions $\mathfrak{i}^{-1}$, Θ_0 or Θ_0^{-1}. Since $x \notin \mathcal{Z}_{e-1}$, $\mathfrak{i}^{-1}(x)$ is primitive recursive. As maintained by actions B1, B2 and B3, Θ_0, Θ_0^{-1} are also both primitive recursive functions. Thus $\alpha_e : T \to A_e$ is then primitive recursive for any $x \notin \mathcal{Z}_{e-1}$ provided p_e is primitive recursive. It remains then to check that for any $x \in \mathcal{Z}_{e-1}$, $q_e^{-1}(x)$ is primitive recursive after stage s. Consider the α_e being built after stage s.

Let $\phi_n : T \to B$ be the highest priority primitive recursive function assigned to α_e; $\phi_n(S_{\langle\rangle}) = S_\tau$ for some good τ of colour e. Suppose to the contrary that there exists $x \in \mathcal{Z}_{e-1}$ for which $q_e^{-1}(x)$ does not show up primitively recursively (in ϕ_n).

Case 1: Suppose that x enters some bad interval, S_γ, where $\gamma \in \mathcal{Z}_{e-1}$. Let $\gamma \in \text{BAD}(\gamma')$ for some good γ'. Since we can assume that $q_e p_e$ satisfies Condition (C2i), if $(q_e p_e)(S_\sigma) = S_\gamma$, then $\sigma \in \text{BAD}(\gamma')$.

If we further have that γ is of positive colour, then x must have entered due to action A3. This same action must have also flagged the elements x^ξ for every $\xi \in \text{BAD}(\gamma')$. In particular, x^σ must be flagged to be enumerated at some finite stage. Furthermore, by Lemma 6.12, $|\sigma|$ can be found primitive recursively in $|\gamma|$. Thus, the element x^σ must enter S_σ within primitively recursively many stages after x enters S_γ. If instead γ is of colour 0, then it must be that $q_e p_e(S_\sigma) = S_\gamma$ for some σ of colour 0 (by Condition (C2ii)). Employing a similar argument, if x enters S_γ, action A1 must have also flagged x^σ to be enumerated at some finite stage. Hence, it must enter S_σ within some primitive recursive delay after x enters S_γ.

Case 2: Suppose that x enters some good interval S_γ where $\gamma \in \mathcal{Z}_{e-1}$. Applying Lemma 6.13 allows us to obtain that $\phi_n(S_\gamma) = S_{\gamma\tau}$, where τ is such that $\phi_n(S_{\langle\rangle}) = S_\tau$.

Case 2a: If x is enumerated due to action A4, then note that x^σ is defined for any σ, as $x \in S^\dagger_\gamma$. Furthermore, since action A4 flags elements according to the value of the interval they are in, by Lemma 6.12, x^σ must enter within primitively recursively many stages after x enters $S^\dagger_\gamma$. After action A4 is applied, we have that $|S^\dagger_\sigma| = |S_{\langle\rangle}| = |S^\dagger_\gamma|$, and thus $q_e^{-1}(x)$ must be enumerated into the structure A_e. Otherwise $p_e q_e$ must fail to satisfy Condition (C1i).

Case 2b: If x enters S_γ not due to action A4, then we prove that as long as $q_e^{-1}(x)\uparrow$, then $\phi_n(x)$ has no suitable image.

We may suppose that γ is of even length (and good) since only the first line in the definition of α is in danger of not being primitive recursive. Recall from Lemma 6.1 that $|S_{\gamma e}| \leq |S_\gamma|$. In fact, when x is enumerated into S_γ, since R_e is in state building α, action B4ii will not be used to flag $\Theta_e(x) \in S_{\gamma e}$. In particular, we now have that $|S_{\gamma e}| < |S_\gamma|$. As long as $q_e^{-1}(x)\!\uparrow$, we never flag the element $x^{\gamma e} = \Theta_e(x)$ to be enumerated; $\phi_n(x)$ has no suitable image in $S_{\gamma\tau}$. This is because $x^{\gamma\tau}$ can only be enumerated if $x^{\gamma e}\!\downarrow$. In other words, $\phi_n : S_\gamma \to S_{\gamma\tau}$ but $|S_\gamma| > |S_{\gamma e}| \geq |S_{\gamma\tau}|$. ϕ_n cannot possibly be a primitive recursive isomorphism while satisfying all of the enforced conditions.

Thus, α_e is primitive recursive or the highest priority ϕ_n assigned to α_e must fail. If the latter case happens, then R_e closes the gap in Θ_e and switches its state back to building β as described in Case 4 of the formal construction, which is a contradiction to the assumption. It remains to check that $\alpha_e : T \to A_e$ is a surjective isomorphism.

Suppose that $q_e : A_e \to T$ and $p_e : B \to A_e$ are both isomorphisms, and that $q_e p_e : I_k \to I_0$ for some $k \in \mathbb{Z}$. Let $x \in I_0 \setminus \mathcal{Z}_{e-1}$ be given. Then $\alpha_e(x) = p_e \Theta_0^k \, \mathfrak{i}^{-1}(x)$, that is to say, $\alpha_e(x) \in p_e\left(I_k \setminus \mathcal{Z}_{e-1}\right)$. In particular, since $\Theta_0^k : I_0 \setminus \mathcal{Z}_{e-1} \to I_k \setminus \mathcal{Z}_{e-1}$ is a surjective isomorphism, then $\alpha_e : I_0 \setminus \mathcal{Z}_{e-1} \to p_e\left(I_k \setminus \mathcal{Z}_{e-1}\right)$ must also be a surjective isomorphism.

Now let $x \in \mathcal{Z}_{e-1}$ be given. Then, $\alpha_e(x) = q_e^{-1}(x)$. With the assumption that q_e is an isomorphism, $\alpha_e : I_0 \setminus \mathcal{Z}_{e-1} \to q_e^{-1}\left(I_0 \setminus \mathcal{Z}_{e-1}\right)$ is also a surjective isomorphism. Using the assumption that $q_e p_e$ satisfies Condition (C2iii), and that $q_e p_e : I_k \mapsto I_0$, we obtain that $q_e^{-1}\left(I_0 \setminus \mathcal{Z}_{e-1}\right)$ and $p_e\left(I_k \setminus \mathcal{Z}_{e-1}\right)$ partition $p_e\left(I_k\right)$. Hence, $\alpha_e : I_0 \to p_e(I_k)$ is a surjective isomorphism. Then for all other intervals I_l where $l \neq 0$, $\alpha_e : x \in I_l \mapsto (p_e \Theta_0^k)(x) \in p_e(I_{l+k})$, which is clearly a surjective isomorphism. Putting it all together gives us that $\alpha_e : T \to A_e$ is an isomorphism. □

Lemma 6.16. *For each e, R_e is satisfied.*

Proof. If R_e changes its state to satisfied at some stage during the construction, it must be because $q_e p_e$ was found to fail one of the enforced conditions. Then the construction acts to diagonalise against $q_e p_e$ and thus R_e must be satisfied. Suppose q_e, p_e are both primitive recursive isomorphisms. Suppose R_e never enters the satisfied state during the construction. By Lemma 6.4, let s be the stage after which R_e is never again initialised. After such a stage s, R_e is either in state building α for cofinitely many stages, or it is in state building β for infinitely many stages. Then α_e is a primitive recursive surjective isomorphism by Lemma 6.14, or β_e is a primitive recursive surjective isomorphism by Corollary 6.10 respectively. □

References

[AK00] C. Ash and J. Knight. *Computable structures and the hyperarithmetical hierarchy*, volume 144 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 2000.

[Ala18] P. E. Alaev. Categoricity for primitively recursive and polynomial Boolean algebras. *Algebra Logika*, 57(4):389–426, 2018.

[BDKM19] Nikolay Bazhenov, Rod Downey, Iskander Kalimullin, and Alexander Melnikov. Foundations of online structure theory. *Bull. Symb. Log.*, 25(2):141–181, 2019.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, 1998.

[BKMN20] Nikolay Bazhenov, Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng. Online presentations of finitely generated structures. *Theoret. Comput. Sci.*, 844:195–216, 2020.

[DDH$^+$] M. Dorzhieva, R. Downey, E. Hammatt, A. Melnikov, and K.M. Ng. Punctually presented structures, ii: comparing presentations. *To appear*.

[DGM$^+$20] Rod Downey, Noam Greenberg, Alexander Melnikov, Keng Meng Ng, and Daniel Turetsky. Punctual categoricity and universality. *J. Symb. Log.*, 85(4):1427–1466, 2020.

[DHTK$^+$20] Rodney Downey, Matthew Harrison-Trainor, Iskander Kalimullin, Alexander Melnikov, and Daniel Turetsky. Graphs are not universal for online computability. *J. Comput. System Sci.*, 112:1–12, 2020.

[DKL$^+$15] Rodney G. Downey, Asher M. Kach, Steffen Lempp, Andrew E. M. Lewis-Pye, Antonio Montalbán, and Daniel D. Turetsky. The complexity of computable categoricity. *Adv. Math.*, 268:423–466, 2015.

[DMN21] Rod Downey, Alexander Melnikov, and Keng Meng Ng. Foundations of online structure theory II: The operator approach. *Log. Methods Comput. Sci.*, 17(3):Paper No. 6, 35, 2021.

[EG00] Y. Ershov and S. Goncharov. *Constructive models*. Siberian School of Algebra and Logic. Consultants Bureau, New York, 2000.

[GD80] S. S. Gončarov and V. D. Dzgoev. Autostability of models. *Algebra i Logika*, 19(1):45–58, 132, 1980.

[GHTMT21] Noam Greenberg, Matthew Harrison-Trainor, Alexander Melnikov, and Dan Turetsky. Non-density in punctual computability. *Ann. Pure Appl. Logic*, 172(9):Paper No. 102985, 17, 2021.

[Gla81] A. M. W. Glass. *Ordered permutation groups*, volume 55 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge-New York, 1981.

[GLS03] S. Goncharov, S. Lempp, and R. Solomon. The computable dimension of ordered abelian groups. *Adv. Math.*, 175(1):102–143, 2003.

[Gon80a] S. Goncharov. Autostability of models and abelian groups. *Algebra i Logika*, 19(1):23–44, 132, 1980.

[Gon80b] S. Goncharov. The problem of the number of nonautoequivalent constructivizations. *Algebra i Logika*, 19(6):621–639, 745, 1980.

[Gon81] S. Goncharov. Groups with a finite number of constructivizations. *Dokl. Akad. Nauk SSSR*, 256(2):269–272, 1981.

[Gon97] S. Goncharov. *Countable Boolean algebras and decidability*. Siberian School of Algebra and Logic. Consultants Bureau, New York, 1997.

[HKSS02] D. Hirschfeldt, B. Khoussainov, R. Shore, and A. Slinko. Degree spectra and computable dimensions in algebraic structures. *Ann. Pure Appl. Logic*, 115(1-3):71–113, 2002.

[Kie98] H. A. Kierstead. On line coloring k-colorable graphs. *Israel J. Math.*, 105(1):93–104, 1998.

[KMN17a] I. Sh. Kalimullin, A. G. Melnikov, and K. M. Ng. The diversity of categoricity without delay. *Algebra Logika*, 56(2):256–266, 2017.

[KMN17b] Iskander Kalimullin, Alexander Melnikov, and Keng Meng Ng. Algebraic structures computable without delay. *Theoret. Comput. Sci.*, 674:73–98, 2017.

[KMZ23] Iskander Kalimullin, Alexander Melnikov, and Maxim Zubkov. *Punctual Degrees and Lattice Embeddings*, pages 315–334. 2023.

[KN94] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, pages 367–392, 1994.

[KS99] Bakhadyr Khoussainov and Richard A. Shore. Effective model theory: the number of models and their complexity. In *Models and computability (Leeds, 1997)*, volume 259 of *London Math. Soc. Lecture Note Ser.*, pages 193–239. Cambridge Univ. Press, Cambridge, 1999.

[LaR77] P. LaRoche. Recursively presented boolean algebras. *Notices AMS*, 24:552–553, 1977.

[LMMS05] Steffen Lempp, Charles McCoy, Russell Miller, and Reed Solomon. Computable categoricity of trees of finite height. *J. Symbolic Logic*, 70(1):151–215, 2005.

[LS01] Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*. Classics in Mathematics. Springer-Verlag, Berlin, 2001. Reprint of the 1977 edition.

[Mac11] Dugald Macpherson. A survey of homogeneous structures. *Discrete Mathematics*, 311(15):1599–1634, 2011. Infinite Graphs: Introductions, Connections, Surveys.

[Mal61] A. Mal′cev. Constructive algebras. I. *Uspehi Mat. Nauk*, 16(3 (99)):3–60, 1961.

[Mil92] C. Miller, III. Decision problems for groups—survey and reflections. In *Algorithms and classification in combinatorial group theory (Berkeley, CA, 1989)*, volume 23 of *Math. Sci. Res. Inst. Publ.*, pages 1–59. Springer, New York, 1992.

[MN16] Alexander G. Melnikov and Keng Meng Ng. Computable structures and operations on the space of continuous functions. *Fund. Math.*, 233(2):101–141, 2016.

[MN18] Alexander G. Melnikov and Keng Meng Ng. Computable torsion abelian groups. *Adv. Math.*, 325:864–907, 2018.

[MN20] Alexander Melnikov and Keng Meng Ng. A structure of punctual dimension two. *Proc. Amer. Math. Soc.*, 148(7):3113–3128, 2020.

[Mon21] Antonio Montalbán. *Computable structure theory—within the arithmetic*. Perspectives in Logic. Cambridge University Press, Cambridge; Association for Symbolic Logic, Ithaca, NY, 2021.

[MPSS18] RUSSELL MILLER, BJORN POONEN, HANS SCHOUTENS, and ALEXANDRA SHLAPENTOKH. A computable functor from graphs to fields. *The Journal of Symbolic Logic*, 83(1):326–348, 2018.

[Ner14] Anil Nerode. Musings on Turing's thesis. In *Turing's legacy: developments from Turing's ideas in logic*, volume 42 of *Lect. Notes Log.*, pages 386–396. Assoc. Symbol. Logic, La Jolla, CA, 2014.

[NR87] Anil Nerode and J. B. Remmel. Complexity theoretic algebra. I. Vector spaces over finite fields. pages 1–10. 1987. Conference on Logic and Computer Science: New Trends and Applications (Turin, 1986).

[Rab60] M. Rabin. Computable algebra, general theory and theory of computable fields. *Trans. Amer. Math. Soc.*, 95:341–360, 1960.

[Rem81] J. B. Remmel. Recursively categorical linear orderings. *Proc. Amer. Math. Soc.*, 83(2):387–391, 1981.

[Smi81] R. Smith. Two theorems on autostability in p-groups. In *Logic Year 1979–80 (Proc. Seminars and Conf. Math. Logic, Univ. Connecticut, Storrs, Conn., 1979/80)*, volume 859 of *Lecture Notes in Math.*, pages 302–311. Springer, Berlin, 1981.

[Tur20] Dan Turetsky. Coding in the automorphism group of a computably categorical structure. *J. Math. Log.*, 20(3):2050016, 24, 2020.

Appendix A. Proofs of lemmas about labels

Here we establish certain technical lemmas regarding the labels used in the proof of the main theorem. We begin by listing some observations about the labels.

Fact A.1. *The following are true for any string σ:*

- *If σ is reduced, then for any $x, n \in \omega$, $f(\sigma^\frown x) = f^{n+1}(\sigma^\frown x)$.*
- *If σ is reduced, then for any $i \leq |\sigma|$, $\sigma \upharpoonright_i$ is also reduced.*
- *For any string $\tau \in \omega^{<\omega}$ and any decomposition $\tau = \sigma^\frown\gamma$, $\tau^* = (\sigma^{*\frown}\gamma)^*$.*
- *If σ is reduced, then for all $x \in \omega$, $(\sigma^\frown x)^* = f(\sigma^\frown x)$.*
- *If σ is reduced, then for any $e \in \omega$, $\hat{\theta}_e(\sigma) = (\sigma^\frown e)^*$.*

The following lemma and corollary illustrate that the intervals of different colour are 'sufficiently disjoint'. For example, when some element x is enumerated into $S_{\langle 1\rangle}$, the only way for this to have an effect on enumerations in $S_{\langle 2\rangle}$ is for $\theta_1^{-1}(x)$ to first be enumerated into $S_{\langle\rangle}$.

Lemma A.2. *If $\sigma^\frown k$ is reduced, $(\sigma^\frown k^\frown\gamma)^* = \sigma^\frown k'$ where $\sigma^\frown k'$ is also reduced, and $k \neq k'$, then there exists some $i < |\gamma|$ such that $(\sigma^\frown k^\frown(\gamma\restriction_i))^* = \sigma$.*

Proof. Let σk be reduced and γ be such that for all $i < |\gamma|$, $(\sigma k^\frown(\gamma\restriction_i))^* \neq \sigma$. We aim to show that for all $i < |\gamma|$, σk is a prefix of $(\sigma k^\frown(\gamma\restriction_i))^*$. Suppose not, let $j < |\gamma|$ be the least index for which $\sigma k \not\subseteq (\sigma k^\frown(\gamma\restriction_j))^*$. Applying Fact A.1, we arrive at

$$(\sigma k^\frown(\gamma\restriction_j))^* = \left((\sigma k^\frown(\gamma\restriction_{j-1}))^\frown\gamma(j-1)\right)^* = f\left((\sigma k^\frown(\gamma\restriction_{j-1}))^\frown\gamma(j-1)\right).$$

By choice of j, $\sigma k \subseteq (\sigma k^\frown(\gamma\restriction_{j-1}))^*$. Thus in order for $\sigma k \not\subseteq (\sigma k^\frown(\gamma\restriction_j))^*$ to hold, it must be that $(\sigma k^\frown(\gamma\restriction_{j-1}))^* = \sigma k$, as f only affects at most the last two entries of its input. In fact, the only possibility is that f removes the last two entries of $\sigma k^\frown\gamma(j-1)$, which implies that $(\sigma k^\frown(\gamma\restriction_j))^* = \sigma$, a contradiction. Finally, since for each $i < |\gamma|$, $\sigma k \subseteq (\sigma k^\frown(\gamma\restriction_i))^*$ and σk is reduced, then $(\sigma k\gamma)^*$ either also has σk as a prefix or is equal to exactly σ. In either case, $(\sigma k\gamma)^* \neq \sigma k'$ for any $k' \neq k$. □

Corollary A.3. *If $(k^\frown\gamma)^* = k'$ where $k \neq k'$, then there is some $i < |\gamma|$ such that $(k^\frown(\gamma\restriction_i))^* = \langle\rangle$.*

Lemma A.4. *For all $\gamma \in \omega^{<\omega}$, there is a unique σ such that $\gamma^* = \sigma$. Furthermore, $\sigma \in Z_n \cup Z_n^\frown 0$ for some n.*

Proof. Existence of σ is trivial, and since the reduction procedure is well-defined, the uniqueness is guaranteed. For the second part of the statement, we prove that if $\max(\sigma) \leq n$, σ is reduced, and $|\sigma|$ is even (or odd), then $\sigma \in Z_n$ (or $\sigma \in Z_n^\frown 0$).

Base case: $\max(\sigma) = 0$ or $\sigma = \langle\rangle$. It is easy to see that $\langle\rangle \in Z_0$ and $0 \in Z_0^\frown 0$. Note that if σ is reduced and $\max(\sigma) = 0$, then it can only be the case that $\sigma = \langle 0\rangle$.

Suppose true for σ where $\max(\sigma) < n$. Now consider σ (reduced) such that $\max(\sigma) = n$. We write $\sigma = \gamma^\frown n^\frown\tau$, where $\max(\gamma) < n$. By the definition of the reduction procedure, it is clear that τ is a string of alternating 0's and n's, where $\tau(0) = 0$, otherwise, $\sigma^* \neq \sigma$. We now go through the possible cases.

Case 1a: $|\gamma|$ is even, $|\tau|$ is odd. By the inductive hypothesis, $\gamma \in \mathcal{Z}_m$ for some $m < n$. In particular, $\gamma \in \mathcal{Z}_{n-1}$. Then $\sigma = \gamma^\frown n^\frown\tau = \gamma^\frown n^\frown\langle 0,n\rangle^k{}^\frown 0 = \gamma^\frown\langle n,0\rangle^{k+1}$ for some $k \geq 0$. Thus $\sigma \in \mathcal{Z}_{n-1}{}^\frown\langle n,0\rangle^{k+1}$. And since $\sigma^* = \sigma$, we have that $\sigma \in \mathcal{Z}_{n-1}{}^\frown\langle n,0\rangle^{k+1}$.

Case 1b: $|\gamma|$ is odd, $|\tau|$ is even. By the inductive hypothesis, $\gamma \in Z_m^\frown 0$ for some $m < n$, and so $\gamma = (\xi^\frown 0)^*$ for some $\xi \in Z_m$. In particular, $\xi \in Z_{n-1}$, hence $\xi^\frown\langle 0,n\rangle^\frown\tau \in Z_{n-1}{}^\frown\langle 0,n\rangle^{k+1}$ for some $k \geq 0$. This gives $\sigma = \gamma^\frown n^\frown\tau = (\xi^\frown 0)^\frown n^\frown\tau = \left(\xi^\frown\langle 0,n\rangle^{k+1}\right)^*$, because $\max(\xi) < n$. That is, $\sigma \in Z_{n-1}{}^\frown 0$.

Case 2a: $|\gamma|$ and $|\tau|$ are both even. Consider $\sigma^\frown 0 = \gamma^\frown n^\frown\tau^\frown 0$ and see that $(\sigma^\frown 0)^* = \sigma^\frown 0$. Furthermore, since $|\gamma|$ is even and $|\tau^\frown 0|$ is odd, by Case 1a, we have that $\sigma^\frown 0 \in \mathcal{Z}_n$. Therefore, $\sigma = (\sigma^\frown\langle 0,0\rangle)^* \in Z_n^\frown 0$.

Case 2b: $|\gamma|$ and $|\tau|$ are both odd. Then write τ as $\tau'^\frown 0$. By Case 1b, $\sigma' = \gamma^\frown\tau' \in \mathcal{Z}_n$, as σ' is also reduced. Hence, $\sigma = \gamma^\frown n^\frown\tau'^\frown 0 \in Z_n^\frown 0$.

Thus we have that if $|\sigma| = 2m$ and $\max(\sigma) = n$ then $\sigma \in Z_n$, and also, if $|\sigma| = 2m+1$ and $\max(\sigma) = n$ then $\sigma \in Z_n^\frown 0$. □

It is evident from the definition of Θ_x in the formal construction that for each σ reduced, $\Theta_x : S_\sigma \to S_{(\sigma x)^*}$. We can thus think of Θ_x as a map on labels defined

by $\Theta_x : \sigma \mapsto (\sigma x)^*$. We now verify that as a map on labels, for all $x \in \omega$, Θ_x is surjective (Lemma A.5) and order preserving (Lemma A.6).

Lemma A.5. *For all σ reduced and for all $x \in \omega$, there exists γ reduced such that $(\gamma^\frown x)^* = \sigma$. In particular, if σ is reduced, then $((\sigma x)^{*\frown} x)^* = \sigma$.*

Proof. We prove that for all σ reduced, $((\sigma x)^{*\frown} x)^* = \sigma$. From Def. 4.3 and Fact A.1, we know that

$$(\sigma x)^* = \begin{cases} \sigma^-, & \text{if } \sigma(|\sigma|-1) = x \text{ or if } \sigma(|\sigma|-1) = 0 \text{ and } x < \max(\sigma) \\ \sigma 0, & \text{if } \sigma(|\sigma|-1) \neq 0 \text{ and } x < \max(\sigma) \\ \sigma x, & \text{otherwise.} \end{cases}$$

Then it remains to check through each possibility.

Case 1: $(\sigma x)^* = \sigma^-$. If $\sigma(|\sigma|-1) = x$, then $\sigma^-(|\sigma|-2) \neq x$. In this case, it must also be that $x \geq \max(\sigma) \geq \max(\sigma^-)$. That is, $(\sigma^- {}^\frown x)^* = \sigma^- {}^\frown x = \sigma$. In the other case, if $\sigma(|\sigma|-1) = 0$, then $\sigma^-(|\sigma|-2) \neq 0$. Additionally, since $x < \max(\sigma)$ in this case, then $x < \max(\sigma^-)$, since $\sigma(|\sigma|-1) = 0$. Then $(\sigma^- {}^\frown x)^* = \sigma^- {}^\frown 0 = \sigma$.

Case 2: $(\sigma x)^* = \sigma 0$. In this case, we have that $x < \max(\sigma)$, that is, $(\sigma 0^\frown x)^* = f(\sigma 0^\frown x) = \sigma$.

Case 3: $(\sigma x)^* = \sigma x$, then $(\sigma x^\frown x)^* = \sigma$.

In any case, $((\sigma x)^{*\frown} x)^* = \sigma$. Thus for any σ reduced and for any $x \in \omega$, there exists $\gamma = (\sigma x)^*$ which gives the desired result. □

We now prove that each Θ_x is order preserving as a map on the labels.

Lemma A.6. *For all σ, γ reduced and even length and for all $x \in \omega$, if $\sigma < \gamma$ then $(\sigma x)^* < (\gamma x)^*$.*

Proof. Fix some $x \in \omega$. Since each label of odd length is in $Z_n{}^\frown 0$ for some n, and the ordering there is inherited from the ordering in $\mathcal{Z}_n$, we want to find for each reduced τ, some τ' such that $(\tau x)^* = (\tau' 0)^*$. We claim that given τ, the τ' are determined exactly as below.

	$\tau \in Z_{n-1}{}^\frown (0n)^k$	$\tau \in Z_{n-1}$	$\tau \in Z_{n-1}{}^\frown (n0)^k$
$x < n$	$\tau' = \tau$	$\tau' = (\tau x 0)^*$	$\tau' = \tau$
$x = n$	$\tau' = (\tau x 0)^* = (\tau n 0)^*$	$\tau' = \tau x 0 = \tau n 0$	$\tau' = \tau x 0 = \tau n 0$
$x > n$	$\tau' = \tau x 0$	$\tau' = \tau x 0$	$\tau' = \tau x 0$

Proof of claim. Fix $n, x \in \omega$, and an arbitrary $\tau \in \mathcal{Z}_n$. We run through the computations.

For $x > n$, since τ is reduced, then τx is also reduced, i.e. $(\tau x)^* = \tau x$. For the other side of the equality, $(\tau' 0)^* = (\tau x 00)^* = f(\tau x 00) = \tau x$.

For $x < n$,

- $\tau \in Z_{n-1}{}^\frown (0n)$: τ ends with n. Since $x < n$, then $(\tau x)^* = f(\tau x) = \tau 0 = \tau' 0 = (\tau' 0)^*$.
- $\tau \in Z_{n-1}$: $(\tau' 0)^* = ((\tau x 0)^* 0)^*$. Applying Fact A.1, we know that $(\tau x 0)^* = ((\tau x)^* 0)^*$. Since $(\tau x)^*$ is reduced, thus by Lemma A.5, $((\tau x 0)^* 0)^* = (\tau x)^*$.
- $\tau \in Z_{n-1}{}^\frown (n0)^k$: $(\tau x)^* = f(\tau x) = \tau^-$. But we also have that $f(\tau' 0) = f(\tau 0) = \tau^-$.

Finally, for $x = n$,

- $\tau' = (\tau x0)^*$: $(\tau'0)^* = ((\tau x0)^*0)^* = (((\tau x)^*0)^*0)^* = (\tau x)^*$.
- $\tau' = \tau x0$: Since $\tau x0$ is reduced, then $(\tau'0)^* = (\tau x00)^* = f(\tau x00) = \tau x = (\tau x)^*$.

□

We now return to the proof of the lemma. Fix $\sigma < \gamma$ (of even length) and $x \in \omega$. Let $\max\{\sigma, \gamma\} = n$ and apply the claim above to obtain σ', γ' such that $(\sigma'0)^* = (\sigma x)^*$ and $(\gamma'0)^* = (\gamma x)^*$. It then suffices to show that if $\sigma < \gamma$, $\sigma' < \gamma'$ as this would then imply that $(\sigma'0)^* < (\gamma'0)^*$. (Recall that the ordering in $Z_m^\frown 0$ is inherited from Z_m.)

Case 1: $x > n$. Then $\sigma' = \sigma x0$ and $\gamma' = \gamma x0$. Since $\max\{\sigma, \gamma\} = n < x$, then $\sigma, \gamma \in Z_{x-1}$. Thus $\sigma x0, \gamma x0 \in Z_{x-1}^\frown x0$ where the ordering is inherited from Z_{x-1}. Thus $\sigma' = \sigma x0 < \gamma x0 = \gamma'$ as desired.

Case 2: $x < n$. For the subcases here, if $\sigma' = \sigma$ and $\gamma' = \gamma$, the argument is trivial, as $\sigma < \gamma$ by assumption, giving $\sigma' < \gamma'$. So we check only the cases where at least one of σ, γ is in Z_{n-1}.

- $\sigma < Z_{n-1}$ and $\gamma \in Z_{n-1}$: Then $\sigma' = \sigma$ and $\gamma' = (\gamma x0)^*$ which we again note is in Z_{n-1}, thus $\sigma' < \gamma'$ as desired.
- $\sigma, \gamma \in Z_{n-1}$: In this case, $\max\{\sigma, \gamma\} < n$ which cannot be.
- $\sigma \in \mathcal{Z}_{n-1}$ and $\gamma > Z_{n-1}$: Then $\sigma' = (\sigma x0)^*$ and $\gamma' = \gamma$. Since $\sigma' \in Z_{n-1}$ then $\sigma' < \gamma'$.

Case 3: $x = n$. In this case, the choice of τ' is always $(\tau x0)^* = (\tau n0)^*$, for τ either σ or γ. We then show that τ' is always exactly one block to the right of τ. More specifically, in the ordering

$$\cdots < Z_{n-1}{}^\frown(0n)^2 < Z_{n-1}{}^\frown(0n) < Z_{n-1} < Z_{n-1}{}^\frown(n0) < Z_{n-1}{}^\frown(n0)^2 < \dots$$

if $\tau \in Z_{n-1}{}^\frown(0n)^k$ for some $k > 0$, then $\tau' \in Z_{n-1}{}^\frown(0n)^{k-1}$, and if $\tau \in Z_{n-1}^\frown(n0)^k$ for some $k \geq 0$, then $\tau' \in Z_{n-1}{}^\frown(n0)^{k+1}$. In the latter case, since $\tau \in Z_{n-1}{}^\frown(n0)^k$ must be of the form $\xi^\frown(n0)^k$ where $\xi \in Z_{n-1}$, then $\tau' = \xi^\frown(n0)^k{}^\frown n0 = \xi^\frown(n0)^{k+1}$, thus $\tau' \in Z_{n-1}{}^\frown(n0)^{k+1}$.

If $\tau \in Z_{n-1}{}^\frown(0n)^k$ for some $k > 0$, then $\tau = \left(\xi^\frown(0n)^k\right)^*$ for some $\xi \in Z_{n-1}$. If $\xi(|\xi| - 1) = 0$, then $\tau = (\xi^-)^\frown n(0n)^{k-1}$. If we further have that $k - 1 > 0$, then

$$\begin{aligned}
\tau' &= \left((\xi^-)^\frown n(0n)^{k-1}{}^\frown n0\right)^* \\
&= (\xi^-)^\frown n(0n)^{k-2} \\
&= (\xi^\frown(0n)^\frown(0n)^{k-2})^* \\
&= (\xi^\frown(0n)^{k-1})^*
\end{aligned}$$

giving that $\tau' \in Z_{n-1}{}^\frown(0n)^{k-1}$ as desired. Otherwise, suppose that $k - 1 = 0$, then

$$\begin{aligned}
\tau' &= ((\xi^-)^\frown n^\frown n0)^* \\
&= f(f((\xi^-)^\frown n^\frown n)^\frown 0) \\
&= f((\xi^-)^\frown 0) \\
&= \xi
\end{aligned}$$

That is, if $\tau \in \mathcal{Z}_{n-1}{}^\frown(0n)$, then $\tau' = \xi \in \mathcal{Z}_{n-1}$.

Finally, if $\xi(|\xi|-1) \neq 0$ then $\tau = \xi^\frown(0n)^k$ for some $k > 0$. Then

$$\begin{aligned}\tau' &= \left(\xi^\frown(0n)^k{}^\frown n0\right)^* \\ &= f\left(f\left(\xi^\frown(0n)^k{}^\frown n\right)^\frown 0\right) \\ &= f\left(\xi^\frown(0n)^{k-1}{}^\frown 0^\frown 0\right) \\ &= \xi^\frown(0n)^{k-1}\end{aligned}$$

Thus $\tau' \in Z^\frown_{n-1}(0n)^{k-1}$ as desired. It is then easy to see that if σ is in a block strictly left of γ, σ' must also be in a block strictly left of γ', giving that $\sigma' < \gamma'$. If however, σ, γ are in the same block, then σ', γ' will also be in the same block. Then the order between them comes down to their initial segments which lie in Z_{n-1}. However, since $\sigma < \gamma$, and they lie in the same block, it must be that the initial segment of σ which lies in Z_{n-1} is strictly to the left of the initial segment of γ which lies in Z_{n-1}. But the initial segment of σ which lies in Z_{n-1} is exactly the initial segment of σ' which lies in Z_{n-1}, and the same holds for γ and γ', thus $\sigma' < \gamma'$ as desired. □

Nanyang Technological University, Singapore
Email address: heertern001@e.ntu.edu.sg

Victoria University of Wellington, Wellington, New Zealand
Email address: alexander.g.melnikov@gmail.com

Nanyang Technological University, Singapore
Email address: kmng@ntu.edu.sg