THE BACK-AND-FORTH METHOD AND COMPUTABILITY WITHOUT DELAY

ALEXANDER G. MELNIKOV AND KENG MENG NG

Abstract. We study the algorithmic content of back-and-forth proofs for graphs and homogeneous structures from the perspective of Turing computations in which unbounded search is forbidden. Quite unexpectedly, we discover subtle differences between the back-nad-forth proofs for the random graph, the dense linear order of the rationals, and the universal countable abelian p-group. We also prove the primitive recursive analog of the Cantor-Bernstein theorem for graphs which says that if there is a "back" isomorphism and the "forth" isomorphism then there is a "back-and-forth" isomorphism. We also show that the fully primitive recursive degrees (to be defined) of the dense linear order of the rationals are downwards dense.

1. Introduction

Every two dense countable linear orders with no end-points are isomorphic. The elementary back-and-forth proof of this fact is a standard example of a "computable" proof in mathematics. Essentially the same algorithmic argument works for the random graph, the atomless countable Boolean algebra, and many other homogeneous structures. Here by "computable" we mean "Turing computable". In particular, we allow our computations to use unbounded resources such as memory, running time, etc. It is quite natural to see if the classical back-and-forth method still works, perhaps after some modifications, when we put a resource bound on our computation. Recall the standard "algorithmic" back-and-forth proof in the case of $(\mathbb{Q}, <)$: We have two (supposedly, algorithmic) presentations of the dense order, say A and B. Suppose $x, y \in B$, assume $\psi(x)$ and $\psi(y)$ have already been defined, and we see $x < z < y$. We *wait* for a w to appear between $\psi(x)$ and $\psi(y)$ and set $\psi(z) = w$. How long should we wait? What if we have to decide "now"? What does "now" mean? To give precise answers to these questions we need formal definitions.

To investigate the algorithmic nature of algebraic back-and-forth arguments we need to choose a suitable definition of an algorithmically presented structure. The following rather general definition of a Turing computable structure goes back to Mal′cev [Mal61] and independently Rabin [Rab60]: *A countably infinite algebraic structure $\mathcal{A}$ is constructive or (Turing) computable if its universe is the set of natural numbers $\mathbb{N}$, and the operations and relations on $\mathcal{A}$ are (Turing) computable.* Note that the definition does not assume any resource bound on the computations. Although the study of constructive structures has been rather fruitful [EG00, AK00], this approach is inadequate for our purposes. We should put some restriction on computations in the definition. Of course, this idea is not new. In the early 1990s, Nerode and Remmel [NR90] started a systematic investigation of polynomial-time computable structures, and Khoussainov and Nerode [KN94] suggested a general definition of algebraic

Date: February 18, 2019.

This work was inspired by many discussions with our former mentor, advisor, and friend Rod Downey. We would like to thank Iskander Kalimullin for many interesting and useful discussions.

The second author is partially supported by grant MOE2015-T2-2-055 and RG131/17.

structures represented by finite automata. Although automatic algebra is a rather appealing and deep topic [KNRS07, ECH+92, NT08, BS11, NS07], automatic structures tend to be very rare; for instance, the additive group of the rationals is not automatic [Tsa11]. The approach via polynomial-time algorithms seems far less restrictive. Remarkably, in many common algebraic classes every constructive (i.e., Turing computable) structure has a polynomial-time computable copy [Gri90, CR, CR92, CDRU09, CR91]. For example, every constructive linear order can be transformed into a polynomial-time computable one [Gri90]. As was noted in [KMN], many known proofs of this sort (e.g., [CR91, CR92, CDRU09, Gri90]) are essentially focused on making the operations and relations on the structure merely *primitive recursive* and then observing that the operations are polynomial-time. On the other hand, to illustrate that a structure has no polynomial time copy, it is sometimes easiest to argue that it does not even have a presentation with primitive recursive operations, see e.g. [CR92]. Such proofs exploit only that there is *some* bound on a computation, and it does not really matter what exactly the bound is. Informally, if an algorithm does not have instructions of the form "wait until some effective process halts", then the algorithm will be primitive recursive.

It follows that primitive recursion plays a rather important intermediate role in such proofs. Primitive recursion here serves as a rather useful *abstraction* that often allows to successfully strip the irrelevant combinatorics away. Based on this crucial observation, Kalimullin, Melnikov and Ng [KMN, KMN17, Mel17] have initiated a systematic study of fully primitive recursive structures: *A countable algebraic structure is fully primitive recursive (fpr) if its domain is* $\mathbb{N}$ *and the operations and predicates of the structure are (uniformly) primitive recursive.* (We note that Alaev [Ala16b, Ala16a] has independently started a closely related research program.)

We adopt the approach via primitive recursion since it emphasises the issues related to the (un)bounded search. We return to the discussion of the back-and-forth proof for $(\mathbb{Q}, <)$. What does "map a point now" mean? In this paper "now" means "without unbounded delay"; more formally, "primitively recursively".

An expert in pure recursion theorist may find our results below counterintuitive, while an expert in (say) complexity theory will perhaps be surprised by the combinatorial depth of our arguments. The former is used to taking unbounded search for granted, the latter may suspect that just removing unbounded search is not enough to get any deep results. Nonetheless, our proofs tend to be combinatorially relatively involved; we conjecture that this complexity is unavoidable.

1.1. **Results.** Note that the inverse of a primitive recursive function does not have to be primitive recursive. This feature makes the situation with fully primitive recursive (fpr) structures very different from computable structure theory, as it leads to a reduction. Let $\mathbf{FPR}(\mathcal{A})$ be the collection of all fpr presentations of a countably infinite structure $\mathcal{A}$. For $\mathcal{A}_1, \mathcal{A}_2 \in \mathbf{FPR}(\mathcal{A})$, write $\mathcal{A}_1 \leq_{pr} \mathcal{A}_2$ if there exists a primitive recursive isomorphism from $\mathcal{A}_1$ onto $\mathcal{A}_2$. Clearly, $\leq_{pr}$ is reflexive and transitive. We write $A_1 \simeq_{pr} A_2$ if $A_1 \leq_{pr} A_2$ and $A_2 \leq_{pr} A_1$. In particular, we can look at the fpr-degrees of a given countably infinite structure $\mathcal{A}$.

Definition 1.1 ([KMN17])**.** The fully primitive recursive (fpr) degrees of a countably infinite algebraic structure $\mathcal{A}$ is the quotient structure $\mathbf{FPR}(\mathcal{A}) = (\mathbf{FPR}(\mathcal{A}), \leq_{\mathbf{pr}})/\simeq_{\mathbf{pr}}$.

Intuitively, the fpr-degrees $\mathbf{FPR}(\mathcal{A})$ is a highly sensitive computability-theoretic invariant of $\mathcal{A}$ that encodes/reflects the primitive recursive back-and-forth content of the isomorphism type of $\mathcal{A}$. We shall use fpr-degrees to study subtle properties of back-and-forth proofs for various structures.

1.1.1. *The first main result.* Recall that a structure $\mathcal{X}$ is homogeneous if every isomorphism $f : F_1 \to F_2$ between any two finitely generated substructures $F_1, F_2 \subseteq \mathcal{X}$ is extendable to an automorphism of $\mathcal{X}$. See [Mac11] for a survey on homogeneous structures.

Example 1.2. The following structures are homogeneous:

- η, the dense linear order without end-points.
- $\mathcal{R}$, the Random Graph.
- $\mathcal{P} \cong \bigoplus_{i\in\omega} \mathbb{Z}_{p^\infty}$, the universal countable abelian p-group (the infinite direct power of the Prüfer group $\mathbb{Z}_{p^\infty}$).

Each structure in Ex. 1.2 is the Fraisse limit of finite structures within the respective class. Also, they do share essentially the same back-and-forth proof of their uniqueness up to isomorphism. The back-and-forth proofs for these structures are identical from the general Turing computability point of view. Remarkably, our first main result below shows that the back-and-forth proofs for these three structures differ from the perspective of primitive recursion.

Theorem 1.3. The fpr-degree structures of the dense linear order η, the random graph $\mathcal{R}$, and the universal divisible abelian p-group $\mathcal{P}$ are pairwise non-isomorphic.

Our proof is essentially degree-theoretic in nature. We note that Alaev [Ala16a] has recently investigated the primitive recursive content of the countable atomless Boolean algebra β. (We note that the use of a polynomial time presentations of β in [Ala16a] does not really make much difference.) We conjecture that $\mathbf{FPR}(\beta) \cong \mathbf{FPR}(\eta)$, but establishing such an isomorphism could be quite tricky (if it exists); the most naive attempt via the interval algebra presentation seems to fail.

1.1.2. *The second main result.* Recall that $A_1 \simeq_{pr} A_2$ stands for $A_1 \leq_{pr} A_2$ and $A_2 \leq_{pr} A_1$. In particular, if $\mathbf{FPR}(\mathcal{A})$ is a singleton, it means that for every two fpr copies of $\mathcal{A}$, say A_1 and A_2, there exist surjective primitive recursive isomorphisms $g : A_1 \to A_2$ and $h : A_2 \to A_1$. Informally, this means that we can always run a primitive recursive "forth" argument as well as a primitive recursive "back" argument, for any pair of presentation of $\mathcal{A}$. Can we always run a back-and-forth for A_1 and A_2 without delay? To make the question formal, we need a definition. Say that a function $f : \mathbb{N} \to \mathbb{N}$ is fully primitive recursive (fpr) if both f and f^{-1} are primitive recursive.

Definition 1.4 ([KMN])**.** A structure $\mathcal{A}$ is *fpr categorical* if it has a unique fpr presentation up to fpr isomorphism.

See [KMN] for a number of results describing fpr categoricity in various common classes of algebraic structures.

So suppose $|\mathbf{FPR}(\mathcal{A})| = \mathbf{1}$. In this case we say that $\mathcal{A}$ is *weakly fpr categorical*. Does a weakly fpr categorical $\mathcal{A}$ have to be fpr-categorical? The right-to-left implication is trivial. The second main result resolves the question in the broad class of undirected graphs.

Theorem 1.5. Suppose $\mathcal{G}$ is a graph. Then $\mathcal{G}$ is weakly fpr categorical iff $\mathcal{G}$ is fpr categorical.

The proof is quite technical, for the following reason. One possible approach to proving that the two notions are equivalent is to prove the *uniform* version. That is, given two fpr graphs which are $\simeq_{pr}$-equivalent, we could try to construct an fpr isomorphism between them. This would clearly work if the language has only unary predicate relations, since there are no dependencies between elements. Unfortunately, already in the language of graphs the situation is more complicated. It is not hard to see that there exist fpr graphs $\mathcal{A}$ and $\mathcal{B}$ and primitive recursive isomorphisms $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{B} \to \mathcal{A}$, but with no fully primitive

recursive isomorphism $h : \mathcal{A} \to \mathcal{B}$. We leave details to the reader. Thus, to show that the two notions of fpr categoricity are equivalent for graphs, one has to use the fact that $\mathcal{A}$ is weakly fpr categorical, instead of merely using the existence of primitive recursive isomorphisms $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{B} \to \mathcal{A}$. Such a proof should involve a construction of some auxiliary copies of $\mathcal{A}$ (other than $\mathcal{A}$ and $\mathcal{B}$).

We leave open whether our second main result (Theorem 1.5) can be extended to arbitrary algebraic structures.

Remark 1.6. We conjecture that there is a counterexample to Theorem 1.5 in a functional language. The theorem would likely be much easier to prove if we had a convenient description of fpr-categorical graphs. Obtaining a description of fpr-categorical graphs seems to be a rather challenging task.

1.1.3. *The third main result.* Recall that (in Theorem 1.3) we discovered that the dense inear order, the random graph, and the universal abelian p-group have non-isomorphic fpr-degree structures. The reader may find Theorem 1.3 counterintuitive; the authors definitely did. In fact, the theorem disproves our initial conjecture that the fpr degrees of these structures should be isomorphic.

The discovery of Theorem 1.3 led us to the conclusion that we do not know enough about the fpr-degrees of even the simplest algebraic structures. It makes sense to pick one familiar and algebraically simple structure and try to understand its fpr-degrees in full depth. The hope is that some of the ideas and techniques can then be applied to other, perhaps more algebraically interesting, structures.

Since η is the standard (and perhaps the simplest natural) example when back-and-forth works, we chose to collect more facts about $\mathbf{FPR}(\eta)$. As the third main result of this paper, we prove:

Theorem 1.7. Let η be the dense linear order without end-points. Then $\mathbf{FPR}(\eta)$ is downwards dense.

The reader should prepare for a non-trivial proof. We leave open whether there is a significantly simpler argument proving the theorem above, and we strongly suspect that the answer is negative. We leave open whether $\mathbf{FPR}(\eta)$ is dense.

2. Homogeneous structures: Proof of Theorem 1.3

Theorem 2.1. *The* **FPR** *degrees of the random graph, the universal abelian p-group, and the dense linear order are all pairwise non-isomorphic.*

Proof. It is clear that the standard copy of the dense ordering of the rationals has the greatest degree in $\mathbf{FPR}(\eta)$. To separate the dense linear order from the random graph, it is sufficient to prove the proposition below.

Proposition 2.2. *The structure of the* **FPR** *degrees of the random graph does not have the greatest element.*

Proof. Given any fpr copy $\mathcal{B}$ of the random graph, we build an fpr copy $\mathcal{A}$ such that $\mathcal{A} \not\leq_{pr} \mathcal{B}$. We build $\mathcal{A}$ satisfying the requirements:

$$R_e : p_e \text{ does not witness } \mathcal{A} \to \mathcal{B},$$

as well as the global requirements

$$Q_{F,G} : \exists x \in \mathcal{A} \text{ such that } x \text{ is connected to } F \text{ and disconnected to } G.$$

Here F, G ranges over pairs of disjoint finite subsets of natural numbers.

Informal discussion. We first describe how to satisfy a single R in isolation together with all the global Q-requirements. Suppose R starts action at some stage s, and we begin by picking any $a_0 \in \mathcal{A}[s]$. We wait for $p(x) \downarrow$ for every $x \in \mathcal{A}[s]$, and wait for some $z^* \in \mathcal{B}$ such that $(z^*, p(a_0)) \notin \mathcal{B}$. While waiting for z^*, we make sure that *every* new element x we put into $\mathcal{A}$ is such that $(x, a_0) \in \mathcal{A}$. This will be our only restriction on the new elements, other than ensuring that $(x, a_0) \in \mathcal{A}$, we do not care if x is connected to other elements of $\mathcal{A}$. Notice that this wait will be finite as $\mathcal{B}$ is guaranteed to be a copy of the random graph.

Now suppose that z^* is found in $\mathcal{B}$. Note that nothing currently in $\mathcal{A}$ can possibly be mapped to z^* under p. Our aim is to force $z^* \notin Rng(p)$. Pick a_1 to be any new element of $\mathcal{A}$ such that $p(a_1) \downarrow$. Now continue enumerating elements into $\mathcal{A}$, but this time, the new elements x enumerated into $\mathcal{A}$ are now required to satisfy $\mathcal{A}(x, a_1) \neq \mathcal{B}(z^*, p(a_1))$. Do this for as long as we like, and eventually pick some new a_2 such that $p(a_2) \downarrow$. Now continue to enumerate more of $\mathcal{A}$, except this time, our restriction on the new elements x of $\mathcal{A}$ are now $\mathcal{A}(x, a_2) \neq \mathcal{B}(z^*, p(a_2))$. Keep going, and it is clear that at the end of the construction, no element of $\mathcal{A}$ can possibly map to z^* under p. Also, since our requirements on new elements are of the form $\mathcal{A}(x, a_i) \neq \mathcal{B}(z^*, p(a_i))$ for different a_i, it is clear that all the global requirements can be met together with R.

Now with two requirements, the restraints on new elements to be enumerated into $\mathcal{A}$ are of the form $\mathcal{A}(x, a_i) \neq \mathcal{B}(z^*, p_0(a_i)) \ \wedge \ \mathcal{A}(x, a'_j) \neq \mathcal{B}(z^{**}, p_1(a'_j))$, but again, since a_i and a'_j increases, all Q requirements can be met together with two R-requirements. As long as a_i and a'_j are picked to be different (in fact a'_j belonging to the lower priority R-requirement will be much smaller than a_i), then there are no conflicts between the wishes of two R-requirements.

Formal construction. Assume $\mathcal{B}$ is an fpr copy of the random graph. We build an fpr copy of the random graph $\mathcal{A}$ an satisfy the requirements R_e and $Q_{F,G}$ for all e, F, G. At stage $s = 0$ do nothing. Assume we are at stage $s = \langle e, k \rangle > 0$. We will attend to R_e at this stage, by taking the appropriate action below, according to the first item that applies, and then perform action (E) below:

(i) If $a_e \uparrow$, we pick it to be a fresh number in $\mathcal{A}[s]$.
(ii) If $a_e \downarrow$ but $a_e^* \uparrow$ (and hence $z_e^* \uparrow$), we check to see if $p_e(x) \downarrow$ for every $x \in \mathcal{A}[s^-]$ where $s^- < s$ is the stage where a_e received definition under (i), and where there is an element $z \in \mathcal{B}[s]$ such that $z \neq p_e(x)$ for every $x \in \mathcal{A}[s^-]$ and $(z, p_e(a_e)) \notin \mathcal{B}$. If so, set a_e^* to be any fresh element of $\mathcal{A}[s]$ and set $z_e^* = z$.
(iii) If $a_e, a_e^* \downarrow$, we check to see if $p_e(a_e^*) \downarrow$. If so, redefine a_e to be the current value of a_e^*, and redefine a_e^* to be any fresh number in $\mathcal{A}[s]$.

Now we perform action (E): We consider the next element s of $\mathcal{A}[s]$ and for every i such that $a_i, a_i^* \downarrow$, declare $\mathcal{A}(s, a_i) = 1 - \mathcal{B}(z_i^*, p_i(a_i))$. For every i such that $a_i \downarrow$ but $a_i^* \uparrow$, we set $\mathcal{A}(s, a_i) = 1$. Let $Q_{F,G}$ be the first Q-requirement not yet satisfied, and check to see if F and G are already present in $\mathcal{A}[s]$, and that $F \cup G$ does not contain any $a_i[s]$ (which are defined). If this is the case, we set $\mathcal{A}(s, x) = 1$ for every $x \in F$ and $\mathcal{A}(s, x) = 0$ for every $x \in G$ (note that this action does not conflict with the previous conditions). Finally, for all remaining $x < s$, we set $\mathcal{A}(s, x) = 0$. This ends stage s of the construction.

2.0.1. *Verification.* First note that $a_e^* \downarrow$ iff $z_e^* \downarrow$, and z_e^* is never redefined. Also note that $a_e^* \downarrow$ implies that $p_e(a_e) \downarrow$, and that a_e, a_e^* for different e are always different, since a_e^* is always redefined fresh. Hence all the instructions in the construction make sense.

Next, we argue that for each e, the requirement R_e will never get stuck waiting at (i), (ii) or (iii) forever. We obviously will not get stuck at (i), and also not at (ii) because p_e is total and $\mathcal{B}$ is a copy of the random graph. (Notice we do not require that $z_e^* \neq z_k^*$ for $e \neq k$). We

also do not get stuck at (iii) because p_e is total. Therefore, for each e, $a_e, a_e^* \to \infty$. Since $a_e, a_e^* > e$, this means that $Q_{F,G}$ is satisfied for every F, G.

Now fix e, and we want to argue that R_e is satisfied. Suppose that p_e is an isomorphism from $\mathcal{A} \to \mathcal{B}$. Let s_0 be the stage where a_e is first defined under (i), s_1 be the stage where a_e^* and z_e^* are defined under (ii), and s_j be the stages where a_e is redefined for $j > 1$. It follows by a simple induction that at every stage s_j for $j > 0$, $p_e(x) \neq z_e^*$ for every $x \in \mathcal{A}[s_j]$. Since $s_j \to \infty$, it follows that p_e is not onto, a contradiction.

Notice that between s_0 and s_1, all new elements x of $\mathcal{A}$ are defined to have $\mathcal{A}(x, a_e) = 1$. At stage s_1 when z_e^* was picked, we have $z_e^* \neq p_e(x)$ for any $x \in \mathcal{A}[s_0]$ and $(z_e^*, p_e(a_e)) \notin \mathcal{B}$. Therefore, $p_e(x) \neq z_e^*$ for any $x \in \mathcal{A}[s_1]$. Now for $j > 1$, observe that every element x introduced into $\mathcal{A}$ between s_{j-1} and s_j must have $\mathcal{A}(x, a_e) = 1 - \mathcal{B}(z_e^*, p_e(a_e))$. Therefore, $p_e(x) \neq z_e^*$ for any such x. □

To separate the dense linear order from the universal abelian p-group, we establish that above any **FPR** degree of the latter there is always a greater one.

Proposition 2.3. *The* **FPR** *degrees of the universal abelian p-group are upwards dense.*

Proof. Given any fpr copy A we produce an fpr copy B and a primitive recursive $h : A \to_{onto} B$ and satisfy $p_e : B \not\to A$.

Diagonalisation. To diagonalise against p_e, choose a fresh witness x_e in B. At stage s set B_s equal to $h(A)[s] \oplus \langle x_e \rangle[s]$ (where $\langle x_e \rangle[s]$ denotes the approximation to the cyclic group generated by x_e at stage s) such that currently the order of x_e looks infinite. Wait for $p_e(x_e) \downarrow = a \in A$. (We wait by copying A into B via h naturally.) Wait for the order of a to be calculated in A. As soon as $O(a) = p^n$ is decided (at stage t), make sure that $O(x_e) > p^n$ by declaring $p^m x_e = 0$, for a large enough m, but keeping $p^{m-1} x_e \notin h(A)$.

The diagonalisation above is trivial, but making h both onto and primitive recursive need care. We give some intuition, and then we describe the construction.

Intuition. Suppose we have diagonalised against p_e and now our goal is to put x_e into the range of $h : B \to A$. The tension is that we need to find the h-image of each $y \in A$ "very soon". In particular, we cannot wait until we see the order of y or whether y satisfies a non-trivial relation with the current domain of h. Although we *know* there is some element of A that will be good enough as a pre-image for x_e, the danger is that we will not be able to see it soon enough and will be forced to map it elsewhere.

Suppose the order of x_e is p^k. We will consequently put elements $p^{k-1} x_e, \ldots, p x_e, x_e$ into the range of h. In fact, it is sufficient to describe how we put $p^{k-1} x_e$ into the domain of h. For that, search for the first found element $y \leq s$ in A outside the current $dom(h)[s]$ such that py is equal to $h^{-1}(p^{l-1} x_e)$, assuming that the latter pre-image has already been found. If no such y exists, copy A to B avoiding $p^l x_e$. If y is found, set $h(y) = p^l x_e$ and proceed to $l + 1$. We will see that, remarkably, this naive idea will work, but it will require a minor modification.

Construction. The diagonalisation requirements will be listed in the natural order and will be met one-by-one, as described above. While some diagonalisation requirement is active (meaning that it follows its instructions), we will be defining h naturally by copying A to B, and we will keep $\langle x_e \rangle$ disjoint with $range(h)$ everywhere except for 0. When the eth requirement is met, we will have constructed partial abelian p-groups $\langle x_e \rangle \oplus B[s]$ and $A[s]$. As soon as this happens the construction will enter *the h-extension phase*, which is described below.

Suppose the order of x_e is p^k. At the beginning of the h-extension phase, set the parameter l equal to 0. The intended meaning is that we wish to extend h to cover $p^{k-l-1}x_e$ assuming that $p^{k-l}x_e$ is in the range, and our first goal is $p^{k-1}x_e$ which has order p. Go through the sequence of substages until $l = k + 1$:

(1) Assuming the current stage is $s \geq k$, for each element $y \leq s$ in A currently not in the domain of h and check if $py = h^{-1}(p^{k-l}x_e)$.
(2) If no such y is found, then extend h naturally to all elements $a \leq s$, by copying A to B and keeping the h-image disjoint with $p^{k-r}x_e$, $r > l$.
(3) If such a y is found, then pick least such and define $h(y) = p^{k-l-1}x_e$, and then extend h accordingly to the rest of $a \leq s$ and to all linear $\mathbb{Z}_{\mid}$-combinations of such elements. Increase l by 1 and goto (1) unless $l = k$. If $l = k$ then stop.

As soon as the h-extension phase is finished, let the next diagonalisation requirement act according to its instructions, meanwhile extending h naturally (avoiding $\langle x_{e+1}\rangle$).

Verification. It is clear that every diagonalisation requirement is met in finite time if it is ever attended in the construction. we need to check that each h-extension phase eventually finishes its work.

The divisibility of H and the fact that its rank is infinite imply that eventually we will find a y with the desired property. Indeed, every element in A, including $h^{-1}(p^{k-l}x_e)$, must be divisible, and furthermore the space $\{x - y : px = h^{-1}(p^{k-l}x_e)\}$ has infinite $\mathbb{Z}_p$-dimension. The first found element y outside of $dom(h)$ such that $py = h^{-1}(p^{k-l})$ will be $\mathbb{Z}_p$-independent of the previously seen elements in A that already have the property $pz = h^{-1}(p^{k-l})$, by (1) and (3). In particular, y cannot already be in the domain of h thus allowing for a consistent extension. The mentioned $\mathbb{Z}_p$-independence implies that any pair of elements of order p in $A/\langle h^{-1}(p^{k-l}x_e)\rangle$ are in fact automorphic in A. In other words, instead of introducing a fresh z such that $pz = p^{k-l}x_e$ and setting $h(y) = z$ we can safely use the already existing $p^{k-l-1}x_e$. There are no further relations between $p^{k-l}x_e$ and $h(y)$ that should be imposed on this stage.

It follows that eventually l will be set equal to $k + 1$ and x_e and the subgroup generated by it will be put into the domain of h. As we have argued, the definition of h differs from the natural copying strategy only by the order in which elements are listed in the image (i.e., instead of using a fresh z we use the already existing power of x_e). Otherwise, the definition is simply the brute-force copying procedure which is clearly an isomorphism of A onto B. It is also clear that B is fully primitive recursive and that h is surjective primitive recursive. □

In particular, it follows that there no maximal elements among the **FPR** degrees of the universal abelian p-group.

Proposition 2.4. *There is a copy $\mathcal{A}$ of the random graph which is maximal in* $\mathbf{FPR}(\mathcal{G})$.

Proof. We shall follow the proof of Proposition 2.2. The global requirements $Q_{F,G}$ are still the same:

$$Q_{F,G} : \exists x \in \mathcal{A} \text{ such that } x \text{ is connected to } F \text{ and disconnected to } G,$$

where F, G ranges over pairs of disjoint finite subsets of natural numbers. We now have to satisfy the requirements

$$R_e : \text{If } p_e : \mathcal{A} \to \mathcal{B}_e \text{ then } p_e^{-1} \text{ is primitive recursive.}$$

2.0.2. *Informal description.* To meet R in isolation is simple. Suppose that $x < s$ is a number such that we have already looked at $p_e(0), p_e(1), \cdots, p_e(x)$, and that these have all converged. Now suppose $z^* \in \mathcal{B}$ such that $p_e^{-1}(z^*) \uparrow$ currently. We shall need to force $p_e^{-1}(z^*)$ to be defined. Call this stage s. Now pick any $a_0 \in \mathcal{A}[s]$, such that $a_0 < x$; in particular, $p_e(a_0) \downarrow$ currently.

From now on, all new elements y introduced into $\mathcal{A}$ are such that $\mathcal{A}(y, a_0) \neq \mathcal{B}(z^*, p_e(a_0))$. We keep doing this until $p_e(y') \downarrow$ for every $y' \in \mathcal{A}[s]$. (Notice that this is a primitive recursive delay from stage s). If by this time we find $p_e^{-1}(z^*) \downarrow$, then we can repeat with a new z^*. Otherwise, if by this time, we still find that $p_e^{-1}(z^*) \uparrow$, then we obviously can no longer ensure that p_e^{-1} is primitive recursive. However if this is the case, notice that no element in the current $\mathcal{A}$ can map to z^*. What we then do for the rest of the construction is to follow Proposition 2.2 to show that $p_e^{-1}(z^*)$ cannot exist, and thus we win the requirement R by showing that p_e is not onto.

Construction. At step $s = 0$ of the construction, do nothing. At step $s = \langle e, k \rangle$ we attend to requirement R_e, where it is either in the *copying phase* (representing that we are still thinking that p_e^{-1} is primitive recursive) or the *diagonalization phase* (where we have switched to killing the isomorphism p_e).

Suppose at step s, R_e is in the copying phase. If $k = 0$ (i.e. this is the first time we are attending to R_e) then set $\hat{z}_e = 0$ and pick $\hat{a}_e$ to be any fresh element of $\mathcal{A}[s]$. Otherwise, $\hat{z}_e$ and $\hat{a}_e$ are already defined. If $\hat{z}_e$ is currently in $Rng(p_e)$ then we increment $\hat{z}_e$ by 1, and set $\hat{a}_e$ to be any fresh element of $\mathcal{A}[s]$. Otherwise, if $p_e(\hat{a}_e) \downarrow$ at some least stage $s^- \leq s$ (we assume that the current value of $\hat{a}_e$ is picked before s^-) and $p_e(x) \downarrow$ for every $x \in \mathcal{A}[s^-]$, we will enter the diagonalization phase by setting $z_e^* = \hat{z}_e$ and $a_e = \hat{a}_e$, and a_e^* to be any fresh element of $\mathcal{A}[s]$.

Suppose at step s, R_e is in the diagonalization phase. In particular, a_e, a_e^* and z_e^* are all defined. Check to see if $p_e(a_e^*) \downarrow$. If so, redefine a_e to be the current value of a_e^*, and redefine a_e^* to be any fresh number in $\mathcal{A}[s]$.

In any case we will now enumerate the next element s of $\mathcal{A}[s]$. For every i such that R_i is in the copying phase and where $p_i(\hat{a}_i) \downarrow$ we set $\mathcal{A}(s, \hat{a}_i) = 1 - \mathcal{B}(\hat{z}_i, p_i(\hat{a}_i))$. For every i such that R_i is in the diagonalization phase, we set $\mathcal{A}(s, a_i) = 1 - \mathcal{B}(z_i^*, p_i(a_i))$. Let $Q_{F,G}$ be the first Q-requirement not yet satisfied, and check to see if F and G are already present in $\mathcal{A}[s]$, and that $F \cup G$ does not contain any a_i or $\hat{a}_i$. If this is the case, we set $\mathcal{A}(s, x) = 1$ for every $x \in F$ and $\mathcal{A}(s, x) = 0$ for every $x \in G$ (note that this action does not conflict with the previous conditions). Finally, for all remaining $x < s$, we set $\mathcal{A}(s, x) = 0$. This ends stage s of the construction.

Verification. First note that it is easy to check that all instructions in the construction make sense, for example the $a_e, \hat{a}_k$ are never the same for different indices.

Next, it is clear that as p_e is total, the requirement R_e will never get stuck waiting and hence $\hat{a}_e \to \infty$ if it is forever in the copying phase, and $a_e, a_e^* \to \infty$ if it ever enters the diagonalization phase. This is sufficient to show that $Q_{F,G}$ is satisfied for every F, G. This implies that $\mathcal{A}$ is a copy of the random graph. Notice that we *do not* assume (and in fact we cannot assume) that $\mathcal{B}_e$ is a copy of the random graph. We only know that it is an fpr graph.

Next we show that R_e is satisfied. Suppose that p_e is an isomorphism from $\mathcal{A}$ onto $\mathcal{B}_e$. We claim that R_e never enters the diagonalization phase. Suppose this happens at some stage s_1. At stage s_1 we set $a_e = \hat{a}_e$ and $z_e^* = \hat{z}_e$ and also picked a new value for a_e^*. At s_1 we have $p_e(\hat{a}_e) \downarrow$, so suppose that this is first defined at some $s_0 \leq s_1$. Now all elements x put into the structure $\mathcal{A}$ between s_0 and s_1 must have $\mathcal{A}(x, \hat{a}_e) = 1 - \mathcal{B}(\hat{z}_e, p_e(\hat{a}_e))$. Since at stage s_1, $\hat{z}_e$ is not yet in the range of p_e, it follows that $p_e(x) \neq \hat{z}_e$ for every $x \in \mathcal{A}[s_1]$. Now it just follows by an obvious induction that at stage s_j for all $j \geq 1$, $p_e(x) \neq \hat{z}_e$ for every $x \in \mathcal{A}[s_j]$. Since $s_j \to \infty$ (here s_j ranges over all the stages where a_e is redefined), it follows that p_e is not onto, a contradiction.

Therefore, R_e is forever in the copying phase. Hence $\hat{z}_e$ will eventually range over all of ω. Suppose y is first processed as $\hat{z}_e$ at stage s. Let $t > s$ be the stage where $p_e(\hat{a}_e)$ converges.

By the time p_e converges on all elements of $\mathcal{A}[t]$, we must have seen $y = \hat{z}_e$ in the range of p_e, otherwise we would enter the diagonalization phase for R_e. We will then consider $y+1$ next. Since R_e is attended to at every stage of the form $\langle e, k\rangle$, this means that p_e^{-1} is primitive recursive. □

Thus, we have separated the random graph from the universal abelian p-group. □

3. Back and forth imply back-and-forth: Proof of Theorem 1.5

We prove a bit more than is stated in Theorem 1.5. In order to state the more general technical result we need some definitions.

Definition 3.1. Let $\mathcal{A}$ and $\mathcal{B}$ be fpr graphs and let $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{B} \to \mathcal{A}$ be primitive recursive isomorphisms. For $a \in \mathcal{A}$ we call the *(forward) locus of* a the set

$$\texttt{LOC}(a) = \{(gf)^n(a) \mid n \in \mathbb{N}\}.$$

We let $\texttt{LOC}(a,k) = \{(gf)^n(a) \mid 0 \leq n \leq k\}$. We say that $\texttt{LOC}(a)$ (or $\texttt{LOC}(a,k)$) is *closed* if there exists some n (or $n \leq k$) such that $(gf)^n(a) = a$. Say that $a \sim_{\texttt{LOC}} b$ iff $a \in \texttt{LOC}(b)$ or $b \in \texttt{LOC}(a)$; clearly $\sim_{\texttt{LOC}}$ is a c.e. equivalence relation on $\mathcal{A}$. Given $x, y \in \texttt{LOC}(a)$ we say that $x \prec_{\texttt{LOC}(a)} y$ iff $n < m$ where $n \in \omega$ is the least such that $x = (gf)^n(a)$ and $m \in \omega$ is the least such that $y = (gf)^m(a)$. Note that $\prec_{\texttt{LOC}(a)}$ is well-defined even if $\texttt{LOC}(a)$ is closed. We abuse notation and say that $(a,b) \in \mathcal{A}$ if a and b are connected (by an edge) in $\mathcal{A}$.

Finally, given any two subgraphs $\mathcal{G}_0$ and $\mathcal{G}_1$ of a graph $\mathcal{G}$, we say that $\mathcal{G}_0$ and $\mathcal{G}_1$ are *synchronized* if either $(x,y) \in \mathcal{G}$ for every $x \in \mathcal{G}_0$ and every $y \in \mathcal{G}_1$ where $x \neq y$, or $(x,y) \notin \mathcal{G}$ for every $x \in \mathcal{G}_0$ and every $y \in \mathcal{G}_1$ where $x \neq y$. Notice that if $\texttt{LOC}(a)$ has size 1, then $\texttt{LOC}(a)$ is synchronized with every $\texttt{LOC}(b)$. If $\texttt{LOC}(a)$ and $\texttt{LOC}(b)$ are both finite and have relatively prime cardinalities, then $\texttt{LOC}(a)$ and $\texttt{LOC}(b)$ are synchronized.

We prove that the following are equivalent:

(1) $\mathcal{G}$ is weakly fpr categorical.
(2) $\mathcal{G}$ is fpr categorical.
(3) Given any two fpr copies $\mathcal{A} \cong \mathcal{B}$ of $\mathcal{G}$, there exist primitive recursive isomorphisms $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{B} \to \mathcal{A}$, and a primitive recursive function $t : \mathbb{N} \to \mathbb{N}$ such that given any locus $\texttt{LOC}(a)$ for $a \in \mathcal{A}$, either $\texttt{LOC}(a, t(a))$ is closed, or $\texttt{LOC}(a)$ is synchronized with $\texttt{LOC}(b)$ for every $b \in \mathcal{A}$ (including $b = a$).

Clearly, (2) → (1). We first prove the easy direction (3) → (2). Fix $\mathcal{A} \cong \mathcal{B}$ and f, g, t with the abovementioned properties. We wish to define an fpr isomorphism $h : \mathcal{A} \to \mathcal{B}$. First take $a \in \mathcal{A}$ and assume that $h(a)$ has not yet been defined. First evaluate $t(a)$ and check if $\texttt{LOC}(a, t(a))$ is closed and that $h(x)$ and $h^{-1}(f(x))$ are not yet defined for every $x \in \texttt{LOC}(a, t(a))$. If this holds then we define $h(x) = f(x)$ for every $x \in \texttt{LOC}(a, t(a))$. Otherwise, we pick the first $n \leq 2a$ such that $h^{-1}(fg)^n(f(a))$ has not yet been defined, and we set $h(a) = (fg)^n(f(a))$.

Next, take $a \in \mathcal{B}$ and assume that $h^{-1}(a)$ is not yet defined. First evaluate $t(g(a))$ and check if $\texttt{LOC}(g(a), t(g(a)))$ is closed and that $h(x)$ and $h^{-1}(f(x))$ are not yet defined for every $x \in \texttt{LOC}(g(a), t(g(a)))$. If this holds then we define $h(x) = f(x)$ for every $x \in \texttt{LOC}(g(a), t(g(a)))$ (notice that $h^{-1}(a)$ is defined by this action). Otherwise, we pick the first $n \leq 2a$ such that $h(gf)^n(g(a))$ has not yet been defined, and we set $h^{-1}(a) = (gf)^n(g(a))$.

Now we check that $h : \mathcal{A} \to \mathcal{B}$ is an fpr isomorphism. First of all, notice that for any $x \in \mathcal{A}$, $h(x) \sim_{\texttt{LOC}} f(x)$. Similarly, for any $y \in \mathcal{B}$, $h^{-1}(y) \sim_{\texttt{LOC}} g(y)$. Thus it follows that $dom(h) = \mathcal{A}$, since the first $n \leq 2a$ such that $h^{-1}(fg)^n(f(a))$ has not yet been defined will exist: If $\texttt{LOC}(a, 2a)$ is not yet closed then n exists as there are fewer than $2a$ many elements so far in $dom(h) \cup Rng(h)$. On the other hand if $\texttt{LOC}(a, 2a)$ is closed then n also exists because

$a \notin dom(h)$ but every element of $\mathtt{LOC}(f(a)) \cap Rng(h)$ must correspond to some element of $\mathtt{LOC}(a)$. A symmetric argument applies to show that $Rng(h) = \mathcal{B}$. h is clearly injective since f is injective, and by the construction. Thus h and h^{-1} are primitive recursive bijections.

It remains to verify that h is an isomorphism. Observe that if $\mathtt{LOC}(a)$ and $\mathtt{LOC}(b)$ are synchronized, and $a' \sim_{\mathtt{LOC}} a$ and $b' \sim_{\mathtt{LOC}} b$ then $\mathtt{LOC}(a')$ and $\mathtt{LOC}(b')$ are also synchronized. Now fix $a \neq b \in \mathcal{A}$. If $\mathtt{LOC}(a)$ and $\mathtt{LOC}(b)$ are synchronized, then $\mathtt{LOC}(f(a))$ and $\mathtt{LOC}(f(b))$ are also synchronized, and thus $(a, b) \in \mathcal{A}$ iff $(f(a), f(b)) \in \mathcal{B}$ iff $(h(a), h(b)) \in \mathcal{B}$.

Now take the first element x looked at by the construction in the same locus as either a or $f(a)$. Then $\mathtt{LOC}(x, t(x))$ (or $\mathtt{LOC}\,(g(x), t(g(x)))$ if $x \in \mathcal{B}$) must be closed, because otherwise $\mathtt{LOC}(a)$ and $\mathtt{LOC}(b)$ are synchronized and we are in the above case. This means that when acting for x we would have defined $h(a) = f(a)$. A similar argument for b shows that $h(b) = f(b)$. In that case we still have $(a, b) \in \mathcal{A}$ iff $(f(a), f(b)) \in \mathcal{B}$ iff $(h(a), h(b)) \in \mathcal{B}$.

Now we devote the rest of this proof to showing (1) $\to$ (3). Fix $\mathcal{A} \cong \mathcal{A}_0$ and primitive recursive isomorphisms $f : \mathcal{A} \to \mathcal{A}_0$ and $g : \mathcal{A}_0 \to \mathcal{A}$. We will construct a subgraph $\mathcal{B}$ of $\mathcal{A}$ and infinitely many (attempts at a) primitive recursive function t where $\mathtt{LOC}(a, t(a))$ is not closed implies that $\mathtt{LOC}(a)$ is synchronized with every $\mathcal{A}$-locus. Fix effective listings of all primitive recursive functions $\{p_e\}_{e\in\omega}$, $\{q_e\}_{e\in\omega}$. It is convenient to view each primitive recursive function as a partial computable function φ where $\varphi(x) \downarrow$ in $p(x)$ many steps for some primitive recursive p. We construct the sequence $\{t_e\}$ and ensure that

$$\text{If } p_e : \mathcal{A} \to \mathcal{B} \text{ and } q_e : \mathcal{B} \to \mathcal{A} \text{ are isomorphisms, then } t_e \text{ works.}$$

3.0.1. *Intuitive discussion of the strategies.* We discuss informally the strategy for one pair, that is, fix a pair (p, q) and we discuss how to enumerate a fpr $\mathcal{B} \cong \mathcal{A}$ such that either we diagonalize against the pair (p, q) or successfully build the function t. First pick any element α_s of $\mathcal{A}$. We want to observe $\mathtt{LOC}(\alpha_s)$ by computing more and more of the locus. Notice that the definition of t can depend on p, q, but not the definition of $\mathcal{B}$, which has to be independent of the pair (p, q). Hence at every stage of the construction, while we wait for p, q to converge on various elements of $\mathtt{LOC}(\alpha_s)$, we must continue to enumerate new elements into $\mathcal{B}$.

The fpr graph $\mathcal{B}$ is going to be a primitive recursive subgraph of $\mathcal{A}$. We should note that the domain of every infinite fpr structure is ω, and $\mathcal{B}$ will not be an exception. Thus, by a primitive recursive subgraph we mean a fpr structure algebraically isomorphic to a subgraph of $\mathcal{A}$. The isomorphism between $\mathcal{A}$ and $\mathcal{B}$ will not be explicitly built as a fully primitive recursive one, or at least not obviously so. In fact, we will not make the two even obviously fpr-equivalent.

We begin by enumerating elements of $\mathtt{LOC}(\alpha_s)$ into $\mathcal{B}$ while waiting for elements $y_0 \in \mathtt{LOC}(\alpha_s)$ and $y_1 = gf(y_0)$ such that $p(y_0), p(y_1)$ have both converged to elements in $\mathcal{B}$ which are in $\mathtt{LOC}(\alpha_s)$. Notice that $p(x)$ may take a very long time to converge, and may in general converge to elements of $\mathcal{B}$ which were enumerated for a previous locus $\mathtt{LOC}(\alpha_t)$ for $t < s$. Since p is injective, and our strategy will keep growing $\mathcal{B}$ by adding elements of $\mathtt{LOC}(\alpha_s)$, we must find y_0, y_1 at some stage. We now give more details on why y_0 and y_1 must exist.

Suppose at stage s_0 we begin considering α_{s_0} and $\mathtt{LOC}(\alpha_{s_0})$. At this point in time in our structure $\mathcal{B}$, we have already enumerated at most s_0 many elements. These elements of $\mathcal{B}[s_0]$ will correspond to $\mathtt{LOC}(\alpha_t)$ for previous values of α_t. Let's call this set C, which consists of elements of $\mathcal{A}$ which are currently in a different locus from α_{s_0}. Now we wait for p_e to converge on a suitable element of $\mathtt{LOC}(\alpha_{s_0})$, say z. This may take a long time to converge, after all, p_e is the e-th primitive recursive function. While waiting for $p_e(z)$ to converge, we must continue putting elements into $\mathcal{B}$. We will compute $\mathtt{LOC}(\alpha_{s_0})$, and we can assume that $\mathtt{LOC}(\alpha_{s_0})$ is generated immediately at the rate of one new element per stage. We copy these elements into

$\mathcal{B}$. Suppose p_e finally converges on z after a very long wait. If $p_e(z) \in \mathcal{B} - C$, then we can take $y_0 = z$, and y_1 can then be computed after a further finite delay. If $p_e(z) \in C$, then we pick a different element, say $g(f(z))$ of $\mathtt{LOC}(\alpha_s)$ and wait for p_e to converge on this. As p_e is injective, and because all we do in the meantime is to force $\mathcal{B}$ to copy $\mathtt{LOC}(\alpha_{s_0})$, therefore C does not increase and eventually we will be able to find y_0. Here we point out that if p_e happens to be not injective, then we will see this at some point in the strategy and we can halt this particular procedure (and win the requirement, of course). On the other hand, if p_e is not surjective, then it will also not matter since we will always force $\mathcal{B}$ to copy $\mathtt{LOC}(\alpha_{s_0})$ while waiting. The function q_e will not be directly involved in the diagonalization.

Note that of course, before we *do* find y_0, it might be that $\mathtt{LOC}(\alpha_{s_0})$ closes and we are unable to use this to generate new elements for $\mathcal{B}$ to copy. However, if this happens, we can define $t_e(\alpha_{s_0})$ to be the stage where this happens. This is bounded by the number of stages it takes for p_e to converge on at most s_0 many elements of $\mathtt{LOC}(\alpha_{s_0})$, which is of course primitive recursive relative to p_e. As noted before, the definition of t_e *can* depend on p_e (and of course, f and g).

Now suppose that y_0, y_1 are found, and suppose that $\mathtt{LOC}(\alpha_s)$ has not yet closed. If $\mathtt{LOC}(\alpha_s)$ is currently synchronized with $\mathtt{LOC}(a)$ for every $a \in \mathcal{A}$ (of course, restricted to the part of $\mathcal{A}$ we have enumerated so far), then we have no choice but to define $t(\alpha_s)$ at this point and move on to process the next locus α_{s+1}. Since $\mathtt{LOC}(\alpha_s, t(\alpha_s))$ is not closed, we must ensure that $\mathtt{LOC}(\alpha_s)$ continues to be synchronized with every locus for the rest of the construction. To record the fact that we are continually monitoring this, we declare α_s as *pending*. At any point later when we discover that α_s or any other pending element is not synchronized with some locus, we will enter the diagonalization phase and take a series of actions which will ultimately allow us to diagonalize against (p, q). We discuss the diagonalization phase below.

For now, let us assume that y_0, y_1 are found, $\mathtt{LOC}(\alpha_s)$ has not yet closed, and $\mathtt{LOC}(\alpha_s)$ is not synchronized with some $\mathtt{LOC}(a)$. In this case, we do not yet define $t(\alpha)$, and must continue to do further tests on $\mathtt{LOC}(\alpha_s)$. There are two cases: Either $\mathtt{LOC}(\alpha_s)$ is synchronized with itself (we call this self-synchronized) but unsynchronized with some $a \not\sim_{\mathtt{LOC}} \alpha_s$, or $\mathtt{LOC}(\alpha_s)$ is not self-synchronized. The second case is the good case and we will be able to proceed further with our strategy. If the first case holds, we must take further steps to force either $\mathtt{LOC}(\alpha_s)$ to close or for $\mathtt{LOC}(\alpha_s)$ to become not self-synchronized.

Let's assume for now that the first case holds, i.e. $\mathtt{LOC}(\alpha_s)$ is synchronized with itself but unsynchronized with some $a \not\sim_{\mathtt{LOC}} \alpha_s$. Suppose that $\mathtt{LOC}(a)$ is currently open. Since $\mathtt{LOC}(\alpha_s)$ is currently not synchronized with $\mathtt{LOC}(a)$, it is not hard to see that there exist elements $e' \in \mathtt{LOC}(\alpha_s)$ and $d' \in \mathtt{LOC}(a)$ such that $\mathcal{A}(e', d') \neq \mathcal{A}(gf(e'), d')$. We continue enumerating elements of $\mathtt{LOC}(\alpha_s)$ into $\mathcal{B}$, and watch $\mathtt{LOC}(\alpha_s)$ and $\mathtt{LOC}(a)$. If these two loci remain open, and since $\mathcal{A}(e', d') \neq \mathcal{A}(gf(e'), d')$, we will be able to get distinct elements $e', e'', e''', \cdots \in \mathtt{LOC}(\alpha_s)$ and $d', d'', d''', \cdots \in \mathtt{LOC}(a)$ such that $\mathcal{A}(e'', d'') \neq \mathcal{A}(gf(e''), d'')$ and $\mathcal{A}(e''', d''') \neq \mathcal{A}(gf(e'''), d''')$ and so on. Now since p is injective, eventually p must map some triple $p(e)$, $p(gf(e))$ and $p(d)$ to elements of $\mathtt{LOC}(\alpha_s)$. Since $\mathcal{A}(e, d) \neq \mathcal{A}(gf(e), d)$, either p is not an isomorphism, or we get $\mathcal{A}(p(e), p(d)) \neq \mathcal{A}(p(gf(e)), p(d))$ which implies that $\mathtt{LOC}(\alpha_s)$ is not self-synchronized, and we are in the good case, the second case, as desired.

Therefore, in the first case, we may assume that the locus $\mathtt{LOC}(a)$ must eventually close. We next wait for $p(a)$ to converge. Unfortunately we have no control over which locus $p(a)$ may land in, it may be the case that $p(a) \sim_{\mathtt{LOC}} a$ or $p(a) \sim_{\mathtt{LOC}} \alpha_s$ or even neither. However, we *can* conclude that $\mathtt{LOC}(p(a))$ is closed: If $\mathtt{LOC}(p(a))$ isn't closed, then as $p(a)$ is an element of $\mathcal{B}$ we previously enumerated, it must be the case that either $p(a) \in \mathtt{LOC}(\alpha_s)$, or $p(a) \in \mathtt{LOC}(\pi)$ for some pending element π. (The locus of every other element of $\mathcal{B}$ must be closed). If

$p(a) \in \mathtt{LOC}(\alpha_s)$ then we see that $\mathtt{LOC}(\alpha_s)$ is not self-synchronized, putting us in the good second case.

If $p(a) \in \mathtt{LOC}(\pi)$ for some pending element π, then π isn't synchronized with α_s, which means that we can enter the diagonalization phase with π. Therefore we may assume that $\mathtt{LOC}(p(a))$ is also closed.

At this point, we have that $\mathtt{LOC}(\alpha_s)$ and $\mathtt{LOC}(a)$ are not synchronized, and both $\mathtt{LOC}(a)$ and $\mathtt{LOC}(p(a))$ are closed, though $\mathtt{LOC}(\alpha_s)$ remains open. Now our strategy for enumerating elements in $\mathcal{B}$ will switch to the following. Let m be the size of $\mathtt{LOC}(p(a))$. We will now enumerate into $\mathcal{B}$ elements $z, (gf)^m(z), (gf)^{2m}(z), (gf)^{3m}(z) \cdots$ for some starting element $z \in \mathtt{LOC}(\alpha_s)$. Since $\mathtt{LOC}(\alpha_s)$ and $\mathtt{LOC}(a)$ are not synchronized, and that $\mathtt{LOC}(a)$ is closed, we can deduce the existence of elements $e_0, e_1, \cdots \in \mathtt{LOC}(\alpha_s)$ such that $\mathcal{A}(e_u, a) \neq \mathcal{A}((gf)(e_u), a)$ for every u. On the other hand, for the elements we are enumerating into $\mathcal{B}$ of the form $(gf)^{jm}(z)$, we see that for any two such elements, $\mathcal{A}\left((gf)^{jm}(z), p(a)\right) = \mathcal{A}\left((gf)^{j'm}(z), p(a)\right)$, since the size of $\mathtt{LOC}(p(a))$ is m. Eventually there must be some u such that $p(e_u)$ and $p((gf)(e_u))$ are both mapped to elements of the form $(gf)^{jm}(z)$. This means that $\mathcal{A}\left(p(e_u), p(a)\right) = \mathcal{A}\left(p((gf)(e_u)), p(a)\right)$, which means that if p is an isomorphism, we have $\mathcal{A}\left(e_u, a\right) = \mathcal{A}\left((gf)(e_u), a\right)$, contradicting the choice of e_u.

Now we may assume that we are in the good case, the second case, where $\mathtt{LOC}(\alpha_s)$ is not self-synchronized. We have the existence of elements $y_0 \in \mathtt{LOC}(\alpha_s)$ and $y_1 = gf(y_0)$ such that $p(y_0)$ and $p(y_1)$ are both in $\mathtt{LOC}(\alpha_s)$. Let m be such that $p(y_1) = (gf)^m\left(p(y_0)\right)$ (or vice versa). As $\mathtt{LOC}(\alpha_s)$ is not self-synchronized there will exist an element $z_0 \in \mathtt{LOC}(\alpha_s)$ such that $\mathcal{A}(y_0, z_0) \neq \mathcal{A}(y_0, (gf)(z_0))$. Without loss of generality, assume for this discussion that $(y_0, z_0) \in \mathcal{A}$ and $(y_0, (gf)(z_0)) \notin \mathcal{A}$.

Now our strategy for enumerating elements in $\mathcal{B}$ could be the following. We test each new element x of $\mathtt{LOC}(\alpha_s)$ to see if $\left((gf)^i(x), p(y_0)\right) \in \mathcal{A}$ for some $i \leq 3m$. If this holds for some i, then we enumerate $(gf)^i(x)$ as our new element of $\mathcal{B}$. If $\left((gf)^i(x), p(y_0)\right) \notin \mathcal{A}$ for every $i \leq 3m$ then we enumerate $(gf)^{2m}(x)$ as our new element of $\mathcal{B}$. Now we claim that $p\left((gf)(z_0)\right)$ cannot be equal to any element of this form: If $p\left((gf)(z_0)\right) = (gf)^i(x)$ where $\left((gf)^i(x), p(y_0)\right) \in \mathcal{A}$, then $((gf)(z_0), y_0) \in \mathcal{A}$ (unless p is not an isomorphism) contradicting the choice of z_0 above. On the other hand if $p\left((gf)(z_0)\right) = (gf)^{2m}(x)$, then as $\left((gf)^i(x), p(y_0)\right) \notin \mathcal{A}$ for every $i \leq 3m$, it follows that $\left((gf)^{\pm m}\left(p\left((gf)(z_0)\right)\right), p(y_0)\right) \notin \mathcal{A}$, and hence $\left(p\left((gf)(z_0)\right), (gf)^{\pm m}\left(p(y_0)\right)\right) \notin \mathcal{A}$. But since $p(y_1) = (gf)^m\left(p(y_0)\right)$ or $(gf)^{-m}\left(p(y_0)\right)$ we see that $\left(p\left((gf)(z_0)\right), p(y_1)\right) \notin \mathcal{A}$, which means (unless p is not an isomorphism) that $((gf)(z_0), y_1) \notin \mathcal{A}$. But since $y_1 = gf(y_0)$ we have that $(z_0, y_0) \notin \mathcal{A}$, again contradicting our choice of z_0 above. Thus we conclude that $p((gf)(z_0))$ cannot be equal to any new element we enumerate into $\mathcal{B}$.

However it could still be possible that $p((gf)(z_0))$ is equal to an element of $\mathcal{B}$ we enumerated previously. If this is the case then we will be unable to force $\mathtt{LOC}(\alpha_s)$ to close quickly using this strategy. We need to guarantee that $\mathtt{LOC}(\alpha_s)$ contains many different elements $z_0 \prec_{\mathtt{LOC}(\alpha_s)} z_1 \prec_{\mathtt{LOC}(\alpha_s)} z_2 \prec_{\mathtt{LOC}(\alpha_s)} \cdots$ all of which behave like z_0, i.e. $(y_0, z_u) \in \mathcal{A}$ and $(y_0, (gf)(z_u)) \notin \mathcal{A}$ for every u. If we can guarantee that these elements exist (and that they can be found quickly), then we can get a contradiction by waiting for $p((gf)(z_u))$ to converge on enough u.

How can we force this sequence? If this sequence does not exist, it means that there is some index i_0 such that $(\alpha_s, (gf)^i(\alpha_s)) \in \mathcal{A}$ for every $i > i_0$ (or the symmetric case that $(\alpha_s, (gf)^i(\alpha_s)) \notin \mathcal{A}$ for every $i > i_0$). Now if this *were* the case, our choice for new elements of $\mathcal{B}$ could exploit this by enumerating the elements $(gf)^{i_0}(\alpha_s), (gf)^{2i_0}(\alpha_s), (gf)^{3i_0}(\alpha_s), \cdots$ in $\mathcal{B}$. In that case, note that given any $j < j'$ we have $\mathcal{A}\left((gf)^{ji_0}(\alpha_s), (gf)^{j'i_0}(\alpha_s)\right) = \mathcal{A}\left(\alpha_s, (gf)^{(j'-j)i_0}(\alpha_s)\right)$, but as $(j'-j)i_0 > i_0$ we see that $\left(\alpha_s, (gf)^{(j'-j)i_0}(\alpha_s)\right) \in \mathcal{A}$. This

means that the elements we enumerate in $\mathcal{B}$ of this form $(gf)^{i_0}(\alpha_s), (gf)^{2i_0}(\alpha_s), (gf)^{3i_0}(\alpha_s), \cdots$ forms a complete subgraph of $\mathcal{B}$. (In the symmetric case that $(\alpha_s, (gf)^i(\alpha_s)) \notin \mathcal{A}$ for every $i > i_0$ we will be able to enumerate a subgraph of $\mathcal{B}$ with all vertices isolated). Now recall that as $(y_0, (gf)(z_0)) \notin \mathcal{A}$ we have that $\left((gf)^i(y_0), (gf)^{i+1}(z_0)\right) \notin \mathcal{A}$ for every i, and so eventually for a large enough i, $p\left((gf)^i(y_0)\right)$ and $p\left((gf)^{i+1}(z_0)\right)$ must both end up in this complete subgraph of $\mathcal{B}$, which means that p is not an isomorphism. (For the symmetric case where the subgraph is completely isolated, use (y_0, z_0) instead of $(y_0, (gf)(z_0))$). The only way to avoid this contradiction is for $\texttt{LOC}(\alpha_s)$ to close before all these events described above, and we will be able to define $t(\alpha_s)$.

Now we need to describe the actions in the diagonalization phase. Recall that the construction will enter the diagonalization phase immediately when we detect that some pending element π is not synchronized with $\texttt{LOC}(a)$ for some a. Our actions for the diagonalization phase is almost identical with the actions we described above for the good case. In $\texttt{LOC}(\pi)$ we have elements y_0 and $y_1 = (gf)(y_0)$ and such that $p(y_0), p(y_1)$ have already converged to elements in $\texttt{LOC}(\pi)$. Since this is the first stage where we detect that $\texttt{LOC}(\pi)$ is not synchronized with any locus, we shall be able to show that given any element $b \in \mathcal{B}$ so far, $\mathcal{A}(p(y_0), b) = \mathcal{A}(p(y_1), b)$. Now similar to the good case described above, we have some $z_0 \in \texttt{LOC}(a)$ such that $\mathcal{A}(y_0, z_0) \neq \mathcal{A}(y_0, (gf)(z_0))$. Our strategy for enumerating elements in $\mathcal{B}$ in the diagonalization phase will also follow the good case: Enumerate w into $\mathcal{B}$ if $(w, p(y_0)) \in \mathcal{A}$, otherwise enumerate w if $((gf)^{\pm m}(w), p(y_0)) \notin \mathcal{A}$. When $p((gf)(z_0))$ converges, it cannot be equal to any of these new elements we enumerate in $\mathcal{B}$, for the same reasons as before. However in this case, $p((gf)(z_0))$ cannot be an element of $\mathcal{B}$ when we started the diagonalization phase, since $\mathcal{A}(p(y_0), b) = \mathcal{A}(p(y_1), b)$ for all such b. Hence this contradiction leads us to conclude that the diagonalization phase must conclude after finitely many steps. Notice that in the diagonalization phase we no longer care about extending the definition of t; this phase will always conclude with the diagonalization of (p, q).

3.0.2. *The overall plan of the construction.* The construction will proceed in stages. At each stage s we assume that we are given access to the subgraph generated by $\texttt{LOC}(a, 2^s)$ for $a \leq 2t$, where t was the largest element considered at the previous construction stage. Denote this subgraph by $\mathcal{A}[s]$. At stage s we will decide the next element b_s to enumerate into $\mathcal{B}$, by setting b_s to be some element of $\mathcal{A}[s] - \{b_0, \cdots, b_{s-1}\}$. When we enumerate b_s into $\mathcal{B}$ we also connect b_s to b_j iff they are connected in $\mathcal{A}$. Clearly this makes $\mathcal{B}$ a primitive recursive graph.

The construction will then go through each pair (p_e, q_e) one at a time, and for each e, build more of the structure $\mathcal{B}$ and the function t_e. Either at some point we discover that (p_e, q_e) are not isomorphisms (and we then move to the next pair (p_{e+1}, q_{e+1})), or we stick it out to the end and successfully build t_e to do what we want. Finally we will ensure that every time we kill off a pair (p_e, q_e) and move to the next pair, we copy more of $\mathcal{A}$ into $\mathcal{B}$ so that at the end, if we manage to kill off every pair, then we succeed in making $\mathcal{A} \cong \mathcal{B}$ and thus obtain a contradiction to the fact that $\mathcal{A}$ is weakly fpr categorical.

Suppose that we have killed off (p_i, q_i) for $i < e$ and this is the first stage we look at (p_e, q_e). Suppose we are now at stage s of the construction, and we have defined $\mathcal{B}_s = \{b_0, \cdots, b_{s-2}\}$. Enumerate the least element b_{s-1} (ordered by index) of $\mathcal{A} - \{b_0, \cdots, b_{s-2}\}$ into $\mathcal{B}$ (this choice of b_{s-1} is only at this first stage; at subsequent stages we attend to e we will decide the choice of b in a more elaborate way). We now describe the construction until (if ever) we succeed in showing that (p_e, q_e) is wrong. The construction for the pair (p_e, q_e) (in short, we refer to this as the construction for e) will begin in the *copying phase*. During this phase there will always be a currently *active element* which we denote as $\alpha_s \in \mathcal{A}$. There will also be a collection of *pending elements*. Roughly speaking, the pending elements are the previously

active elements whose loci did not close in time. At each stage during the copying phase, we will enumerate some element of the locus $\texttt{LOC}(\alpha_s)$ into $\mathcal{B}$, until either $\texttt{LOC}(\alpha_s)$ closes, or we are able to diagonalize with a pending element. In the former situation, we will extend the definition of $t(\alpha_s)$ and take α_{s+1} to be the next available element of $\mathcal{A}$. In the latter situation we will enter the *diagonalization phase* and begin a sequence of actions which will force (p_e, q_e) to be incorrect, via the locus of the diagonalizing element. If the copying phase is always active, then we succeed in defining t_e correctly. If we ever enter the diagonalization phase, we will be able to kill off (p_e, q_e) and move on to the next $e+1$.

3.0.3. *The formal construction for e - Copying phase.* At the first stage where we look at e, set α_s to be the first element of $\mathcal{A}[s]$ (elements are always ordered by their indices). Otherwise assume inductively we have the currently active element α_s and a set of pending elements $\pi < \alpha_s$ with the following properties for each π:

- (π.1) There exist elements $\pi_0 = (gf)^n(\pi)$ and $\pi_1 = (gf)^{n+1}(\pi)$ for some n, such that $p_e(\pi_0), p_e(\pi_1)$ have both converged by this stage and $p_e(\pi_0), p_e(\pi_1) \in \texttt{LOC}(\pi)$. Here we identify elements of $\mathcal{B}$ with the corresponding elements of $\mathcal{A}$.
- (π.2) For every $b \in \mathcal{B}[s] - \{p_e(\pi_0), p_e(\pi_1)\}$, $(b, p_e(\pi_0)) \in \mathcal{B} \Leftrightarrow (b, p_e(\pi_1)) \in \mathcal{B}$.
- (π.3) When restricted to the subgraph $\mathcal{A}[s-1]$, $\texttt{LOC}(\pi)$ is synchronized with $\texttt{LOC}(a)$ for every $a \in \mathcal{A}[s-1]$.

Now first of all check if (p_e, q_e) are partial isomorphisms on $A[s], \mathcal{B}[s]$, and if no, conclude the construction for e. Proceed to the construction for $e+1$. Otherwise go through each of the following item in the list and take the actions corresponding to the first item which applies:

- (C.1) First check if (π.3) still holds for every pending element π in the subgraph $\mathcal{A}[s]$. If this no longer holds, immediately enter the diagonalization phase (described below) without enumerating anything into $\mathcal{B}$ at this stage.
- (C.2) Next check if $\texttt{LOC}(\alpha_s)$ has closed in $\mathcal{A}[s]$, and if so, enumerate the least element of $\texttt{LOC}(\alpha_s) - \mathcal{B}[s-1]$ into $\mathcal{B}[s]$, if it exists. Define $t(\alpha_s) = \max \mathcal{A}[s]$. Let α_{s+1} be the least element of $\mathcal{A}[s] - \{\alpha_0, \cdots, \alpha_s\}$. Go to stage $s+1$.
- (C.3) Check if there exists some n such that the following hold:
  - $y_0 = (gf)^n(\alpha_s), y_1 = (gf)^{n+1}(\alpha_s) \in \mathcal{A}[s]$.
  - $p_e(y_0), p_e(y_1)$ have both converged by this stage, and are both in $\texttt{LOC}(\alpha_s)$. (Note that the definition of $\mathcal{B}$ cannot be delayed by waiting for p_e to converge).
  - $\texttt{LOC}(\alpha_s)$ is either synchronized with $\texttt{LOC}(a)$ for every $a \in \mathcal{A}[s]$ (restricted to members of $\mathcal{A}[s]$), or is not synchronized with itself.

  Suppose this n cannot be found at this time. We take the following actions. Check if there is some $a \in \mathcal{A}[s]$ such that $\texttt{LOC}(a)$ is already closed, $\texttt{LOC}(\alpha_s)$ and $\texttt{LOC}(a)$ are not synchronized, and $p_e(a)$ has already converged such that $\texttt{LOC}(p_e(a))$ is also closed. (Note that there is no requirement for $p_e(a) \sim_{\texttt{LOC}} a$). If a does not exist at this time, enumerate the least element of $\texttt{LOC}(\alpha_s) - \mathcal{B}[s-1]$ into $\mathcal{B}[s]$. Go to stage $s+1$.

  Otherwise fix a and let m be the size of $\texttt{LOC}(p_e(a))$. Let z be the previous element from $\texttt{LOC}(\alpha_s)$ to be enumerated in $\mathcal{B}$. We enumerate $(gf)^m(z)$ as our next element of $\mathcal{B}[s]$. Go to stage $s+1$.
- (C.4) If $\texttt{LOC}(\alpha_s)$ is synchronized with $\texttt{LOC}(a)$ for every $a \in \mathcal{A}[s]$, then we define $t(\alpha_s), \alpha_{s+1}$ as above, and enumerate the least element of $\texttt{LOC}(\alpha_s) - \mathcal{B}[s-1]$ into $\mathcal{B}[s]$. Declare α_s as pending, and go to stage $s+1$.
- (C.5) If we are here, it means that $y_0 = (gf)^n(\alpha_s), y_1 = (gf)^{n+1}(\alpha_s) \in \mathcal{A}[s]$ have been found such that $p_e(y_0), p_e(y_1)$ have converged and are in $\texttt{LOC}(\alpha_s)$, and $\texttt{LOC}(\alpha_s)$ is not synchronized with itself. Suppose $m > 0$ is the number such that $p_e(y_1) = (gf)^m(p_e(y_0))$ or vice versa.

Fix k such that $(gf)^k(\alpha_s)$ is the element enumerated into $\mathcal{B}$ at the previous stage, if the previous stage visited (C.5), and if this is the first visit to (C.5) for α_s, we let $k > m$ to be the largest considered so far by the construction. Take the first subcase below which applies:

(C.5.1) $\big((gf)^{2k}(\alpha_s), p_e(y_0)\big) \in \mathcal{A}$: In this case we enumerate $(gf)^{2k}(\alpha_s)$ as our next element of $\mathcal{B}[s]$.

(C.5.2) $\big((gf)^{2k}(\alpha_s), p_e(y_0)\big) \notin \mathcal{A}$ but there exists some j such that $k < j < 3k$ such that $\big((gf)^{j}(\alpha_s), p_e(y_0)\big) \in \mathcal{A}$: In this case enumerate $(gf)^{j}(\alpha_s)$ as our next element of $\mathcal{B}[s]$.

(C.5.3) $\big((gf)^{j}(\alpha_s), p_e(y_0)\big) \notin \mathcal{A}$ for every $k < j < 3k$: In this case enumerate $(gf)^{2k}(\alpha_s)$ as our next element of $\mathcal{B}[s]$.

Note that the element enumerated in $\mathcal{B}$ is new since $\texttt{LOC}(\alpha_s)$ is not yet closed. Now go to the next stage $s+1$. This ends the description of the copying phase for e.

3.0.4. *The formal construction for e - Diagonalizing phase.* The first time we enter this phase, by Lemma 3.4, we have the existence of elements a, c, y_0, y_1 such that $\{c, y_0, y_1\}$ are pairwise distinct, $y_0 = (gf)^n(a), y_1 = (gf)^{n+1}(a) \in \mathcal{A}[s]$ and $p_e(y_0), p_e(y_1)$ have both converged by this stage, and are both in $\texttt{LOC}(a)$, and $p_e(c)$ has not yet converged. Furthermore $\mathcal{A}(y_0, c) \neq \mathcal{A}(y_1, c)$, and for every $b \in \mathcal{B}[s]$, if $p_e^{-1}(b)$ has not yet been found at this stage then $\mathcal{A}(p_e(y_0), b) = \mathcal{A}(p_e(y_1), b)$. Without loss of generality assume that $(y_0, c) \notin \mathcal{A}$ and $(y_1, c) \in \mathcal{A}$. Also suppose $m > 0$ is the number such that $p_e(y_1) = (gf)^m(p_e(y_0))$ or vice versa.

Now extend the construction by taking the following steps until $p_e(c)$ is found (this is similar to step (C.5) above). Let $z \in \mathcal{A}[s]$ be any element such that $(gf)^j(z) \notin \mathcal{B}[s]$ for every $0 \leq j \leq 3m$; note that it's fine even if $\texttt{LOC}(z)$ closes in fewer than $3m$ steps. Note that z will always exist. Now if $\big(p_e(y_0), (gf)^j(z)\big) \in \mathcal{A}$ for some $0 \leq j \leq 3m$, enumerate $(gf)^j(z)$ as the new element of $\mathcal{B}$. Otherwise, enumerate $(gf)^{2m}(z)$ as the new element of $\mathcal{B}$. Repeat this at every stage until $p_e(c)$ is found. By Lemma 3.5 we will conclude the diagonalization phase with a successful diagonalization for the pair (p_e, q_e). Conclude the construction for e, and proceed to the construction for $e+1$.

3.0.5. *Verification.* We begin with a remark contrasting construction step (C.5) and the diagonalization phase. In both steps the construction wants to enumerate new elements b into $\mathcal{B}$ such that either $(p_e(y_0), b) \in \mathcal{A}$ or $\mathcal{A}(p_e(y_0), b) = \mathcal{A}(p_e(y_1), b) = 0$. In fact, only elements of this type are enumerated in $\mathcal{B}$ under these two steps. This ensures that the element c cannot have a p_e-image equal to a new element b. Under step (C.5), $p_e(c)$ is allowed to converge to an old element $b \in \mathcal{B}$ belonging to a previous locus, but there are only finitely many possible choices for $p_e(c)$. Hence, we need the locus $\texttt{LOC}(\alpha_s)$ to stay open and apply the non-synchronization of $\texttt{LOC}(\alpha_s)$ to produce enough elements of type c. Once $\texttt{LOC}(\alpha_s)$ closes we have to abandon this strategy since we are no longer guaranteed to have any more elements c with the property $\mathcal{A}(y_0, c) \neq \mathcal{A}(y_1, c)$. In contrast, under the diagonalization phase, $p_e(c)$ cannot converge even to an element belonging to a previous locus, because of property $(\pi.2)$. Thus we are able to kill p_e once $p_e(c)$ converges. Therefore the diagonalization phase will proceed even if the current locus $\texttt{LOC}(\alpha_s)$ closes.

We begin the verification by checking some easy properties of the construction. First of all, notice that any element enumerated into $\mathcal{B}$ during the copying phase must be a member of $\texttt{LOC}(\alpha_s)$.

Lemma 3.2. *For each e, and at the end of every stage in the copying phase for e, the properties* $(\pi.1), (\pi.2)$ *and* $(\pi.3)$ *hold for every pending element* π.

Proof. Suppose these properties hold for a pending element π at the end of $s-1$. (π.1) obviously holds at the end of s. (π.3) holds unless (C.1) applies at stage s, in which case we immediately enter the diagonalization phase. For (π.2), simply observe that (C.1) did not apply at stage s. Now suppose α_s is declared as pending at a stage s under (C.4). Then (α_s.1) holds since (C.3) did not apply at s. (α_s.3) holds by condition (C.4). Lastly (α_s.2) holds since $p_e(y_0)$ and $p_e(y_1)$ are both in $\mathtt{LOC}(\alpha_s)$. □

Lemma 3.3. *$\mathcal{B}$ is a primitive recursive subgraph of $\mathcal{A}$.*

Proof. At every stage in the diagonalizing phase we always enumerate a new element in $\mathcal{B}$. In the copying phase we will be able to enumerate a new element into $\mathcal{B}$ as long as the current locus $\mathtt{LOC}(\alpha_s)$ is still open, so the only item to check is (C.2). It suffices to show that any consecutive run of stages in which we enumerate nothing into $\mathcal{B}$ is primitive recursively computable. Therefore we may restrict ourselves to the situation where $\mathtt{LOC}(\alpha_s)$ closes at the first stage immediately after it is defined. However if this is the case, then α_s itself must have already been enumerated into $\mathcal{B}$ previously. Thus any consecutive run of stages in which we enumerate nothing into $\mathcal{B}$ is primitive recursively bounded. Finally observe that each construction stage speeds up the enumeration of $\mathcal{A}$ by a primitive recursive factor. □

Lemma 3.4. *Suppose we first enter the diagonalization phase for e. Then (by extending the construction by finitely many more stages if necessary) we have the existence of elements a, c, y_0, y_1 such that $\{c, y_0, y_1\}$ are pairwise distinct, $y_0 = (gf)^n(a), y_1 = (gf)^{n+1}(a) \in \mathcal{A}[s]$ and $p_e(y_0), p_e(y_1)$ have both converged by this stage, and are both in $\mathtt{LOC}(a)$, and $p_e(c)$ has not yet converged. Furthermore $\mathcal{A}(y_0, c) \neq \mathcal{A}(y_1, c)$, and for every $b \in \mathcal{B}[s]$, if $p_e^{-1}(b)$ has not yet been found at this stage then $\mathcal{A}(p_e(y_0), b) = \mathcal{A}(p_e(y_1), b)$.*

Proof. Suppose we enter the diagonalization phase via (C.1) at some stage s. Fix a pending π and $d \in \mathcal{A}[s]$ such that $\mathtt{LOC}(\pi)$ and $\mathtt{LOC}(d)$ are not synchronized. By Lemma 3.2, (π.1) and (π.2) holds, so we can take a to be π, and y_0, y_1 appropriately. We certainly have $\mathcal{A}(p_e(y_0), b) = \mathcal{A}(p_e(y_1), b)$ for every $b \in \mathcal{B}[s] - \{p_e(y_0), p_e(y_1)\}$ by (π.2). Now it remains to find $c \in \mathtt{LOC}(d)$.

Since $\mathtt{LOC}(\pi)$ and $\mathtt{LOC}(d)$ are not synchronized, fix n_0, n_1, m_0, m_1 such that $\mathcal{A}\left((gf)^{n_0}(\pi), (gf)^{m_0}(d)\right) \neq \mathcal{A}\left((gf)^{n_1}(\pi), (gf)^{m_1}(d)\right)$, where $(gf)^{n_0}(\pi) \neq (gf)^{m_0}(d)$ and $(gf)^{n_1}(\pi) \neq (gf)^{m_1}(d)$. Suppose that $\mathtt{LOC}(\pi)$ is closed. Then there exist i_0, i_1 such that $(gf)^{n_0+i_0}(\pi) = (gf)^{n_1+i_1}(\pi) = y_0$, and thus $\mathcal{A}\left(y_0, (gf)^{m_0+i_0}(d)\right) = \mathcal{A}\left((gf)^{n_0}(\pi), (gf)^{m_0}(d)\right) \neq \mathcal{A}\left((gf)^{n_1}(\pi), (gf)^{m_1}(d)\right) = \mathcal{A}\left(y_0, (gf)^{m_1+i_1}(d)\right)$. Clearly $\left\{y_0, (gf)^{m_0+i_0}(d), (gf)^{m_1+i_1}(d)\right\}$ are pairwise distinct.

We claim that there exists a number i such that $\mathcal{A}\left(y_0, (gf)^i(d)\right) \neq \mathcal{A}\left(y_0, (gf)^{i+1}(d)\right)$ and that $\left\{y_0, (gf)^i(d), (gf)^{i+1}(d)\right\}$ are pairwise distinct. If $d \notin \mathtt{LOC}(\pi)$ then this is obvious, by taking i to be some number between $n_0 + i_0$ and $n_1 + i_1$. Otherwise if $d \in \mathtt{LOC}(\pi)$, and since the locus has closed, we can write $(gf)^{m_0+i_0}(d) = (gf)^{j_0}(y_0)$ and $(gf)^{m_1+i_1}(d) = (gf)^{j_1}(y_0)$ for some $j_0 \neq j_1$, both strictly positive, and then taking $(gf)^i(d) = (gf)^j(y_0)$ for some $j_0 \leq j < j_1$.

Now we can take $c = (gf)^{i+1}(d)$ we see that $\mathcal{A}(y_1, c) = \mathcal{A}(y_0, (gf)^i(d)) \neq \mathcal{A}(y_0, c)$, where $\{c, y_0, y_1\}$ are pairwise distinct. Note that this holds whether or not $\mathtt{LOC}(d)$ is closed. Now if $p_e(c)$ has converged then as $c \neq y_0, y_1$ we see that by (π.2) we have $\mathcal{A}(p_e(y_0), p_e(c)) = \mathcal{A}(p_e(y_1), p_e(c))$, which means that p_e is not an isomorphism, and we would not have entered the diagonalization phase.

Now suppose that $\mathtt{LOC}(\pi)$ is not yet closed. We might not be able to obtain the element c immediately if n_0 or n_1 is larger than n; since $\mathtt{LOC}(\pi)$ isn't yet closed, we cannot argue by moving backwards in the locus, since $(gf)^{m_0-n_0+n}(d)$ might not currently exist. In this case, we check if d can be taken to be in $\mathtt{LOC}(\pi)$ (equivalently, if $\mathtt{LOC}(\pi)$ is currently not synchronized

with itself). Suppose we can take $d \in \mathtt{LOC}(\pi)$, then the argument proceeds exactly as above (in the case $\mathtt{LOC}(\pi)$ is closed) to find c. If d currently cannot be found in $\mathtt{LOC}(\pi)$ (equivalently, if $\mathtt{LOC}(\pi)$ is currently self-synchronized), then we extend the construction by finitely many more stages, by waiting for the elements $(gf)^{m_0-n_0+n}(d)$ and $(gf)^{m_1-n_1+n}(d)$ to be found. Note that these elements must exist, but may take a very long time to appear. While waiting and extending the construction we cannot delay the definition of $\mathcal{B}$ (see Lemma 3.3) and so we must enumerate elements of $\mathtt{LOC}(\pi)$ into $\mathcal{B}$. As $\mathtt{LOC}(\pi)$ is synchronized with itself we still have $(\pi.2)$ holds. This continues until either $\mathtt{LOC}(\pi)$ closes, or until the first stage found such that $\mathtt{LOC}(\pi)$ is no longer self-synchronized, or until we find the elements above. If $\mathtt{LOC}(\pi)$ closes, we are back in the earlier discussed case; note that $(\pi.2)$ still holds after this extension. If $\mathtt{LOC}(\pi)$ is no longer self-synchronized, then we are in the above case where d can be found in $\mathtt{LOC}(\pi)$; again $(\pi.2)$ still holds after this extension. Otherwise we successfully find $(gf)^{m_0-n_0+n}(d)$ and $(gf)^{m_1-n_1+n}(d)$, and we may repeat the same argument as above to obtain c, noting that $(\pi.2)$ still holds.

Note that extending the construction this way has no effect on Lemma 3.3, since we grow $\mathcal{B}$ at each stage of this extension. However, this may take many more stages (and cannot be primitive recursively bounded), but we do not care as we are no longer concerned with the correctness of t_e if we enter the diagonalization phase (see Lemma 3.5). □

Lemma 3.5. *If we enter the diagonalization phase for e, $p_e : \mathcal{A} \to \mathcal{B}$ is not an isomorphism.*

Proof. When we start the diagonalization phase we have the elements a, c, y_0 and y_1 given by Lemma 3.4, assume this starts at stage $s_0 + 1$. The construction then proceeds in the diagonalization phase for finitely many more stages, until $p_e(c)$ is found. If $p_e(c) \in \mathcal{B}[s_0]$, then by the properties of a, c, y_0 and y_1, we have $\mathcal{A}(y_0, c) \neq \mathcal{A}(y_1, c)$ but $\mathcal{A}(p_e(y_0), p_e(c)) = \mathcal{A}(p_e(y_1), p_e(c))$, which means that p_e is not an isomorphism. So we may assume that $p_e(c)$ has been enumerated into $\mathcal{B}$ after s_0.

By the construction, there are two possibilities. First, if $(p_e(y_0), p_e(c)) \in \mathcal{A}$, then we have $(y_0, c) \notin \mathcal{A}$ and $(y_1, c) \in \mathcal{A}$ (this was the assumption in the construction; of course if $(y_0, c) \in \mathcal{A}$ but $(y_1, c) \notin \mathcal{A}$ then the construction will take the corresponding symmetrical actions). This means that p_e is not an isomorphism. Thus we must have $(p_e(y_0), p_e(c)) \notin \mathcal{A}$. But if this the case, then we in fact have $(gf)^{-m}(p_e(c))$ is defined, and $(p_e(y_0), (gf)^{-m}(p_e(c))) \notin \mathcal{A}$ and $(p_e(y_0), (gf)^{m}(p_e(c))) \notin \mathcal{A}$. Since $p_e(y_1) = (gf)^{\pm m}(p_e(y_0))$ this implies that $(p_e(y_1), p_e(c)) \notin \mathcal{A}$. Hence p_e is not an isomorphism since $(y_1, c) \in \mathcal{A}$. □

Lemma 3.6. *There exists some e such that the construction is forever stuck at e.*

Proof. Suppose the construction finishes every e. Then for every e, the construction either enters the diagonalization phase, or is concluded in the copying phase. By Lemma 3.5 this means that no (p_e, q_e) is a pair of isomorphisms between $\mathcal{A}$ and $\mathcal{B}$. However every time we first begin a new e we copy the least element of $\mathcal{A}$ (not yet in $\mathcal{B}$) in $\mathcal{B}$. This means that $\mathcal{A} \cong \mathcal{B}$. By Lemma 3.3 we get a contradiction to the weak fpr categoricity of $\mathcal{A}$. □

Now for the rest of the verification we fix e to be the final e the construction attends to. For this e the construction will never enter the diagonalization phase. Our task now is to show that t_e is a primitive recursive function, and is total; note that t_e can use the pair (p_e, q_e). Assume inductively that we have defined $t(\alpha_{s-1})$ and a new α_s, and we are also given the stage of the construction s_0 where this happens. Our aim is to compute a primitive recursive bound on the stage where we define $t(\alpha_s)$ and assign a new α; in other words, when either (C.2) or (C.4) applies. The life cycle of α_s can be summarized as follows:

- The construction will keep acting under (C.3), until either $\mathtt{LOC}(\alpha_s)$ closes, or y_0, y_1 are found.

- If $\mathtt{LOC}(\alpha_s)$ closes *before* y_0, y_1 are found, take (C.2) and define $t(\alpha_s)$.
- If y_0, y_1 are found *before* $\mathtt{LOC}(\alpha_s)$ closes, and if $\mathtt{LOC}(\alpha_s)$ is synchronized with every other locus, then take (C.4) and define $t(\alpha_s)$.
- Otherwise we will keep acting under (C.5) until $\mathtt{LOC}(\alpha_s)$ closes and we define $t(\alpha_s)$ under (C.2).

Therefore, it suffices to show that we can primitive recursively obtain a bound on the number of stages the construction can act under (C.3), and under (C.5).

First, let's analyze how many stages the construction can act under (C.3) before y_0, y_1 are found. Define $\mathtt{LOC}(\alpha_s, m, m') = \left\{(gf)^j(\alpha_s) \mid m \leq j \leq m'\right\}$. Note that in any interval of length $2s_0$, that is, given any $\mathtt{LOC}(\alpha_s, m, m+2s_0)$, there is always some $z \in \mathtt{LOC}(\alpha_s, m, m+2s_0)$ such that $p_e(z)$ and $p_e(gf(z))$ are both in $\mathtt{LOC}(\alpha_s)$. Therefore, in the first interval $\mathtt{LOC}(\alpha_s, 2s_0)$ we can already find candidates for y_0, y_1. Wait for p_e to converge on these candidates in $\mathtt{LOC}(\alpha_s, 2s_0)$. Recall that we view each p_e as a (partial) recursive function φ where there is a primitive recursive bound on the number of stages it takes for φ to converge, so waiting for p_e to converge on some or even all elements of $\mathtt{LOC}(\alpha_s, 2s_0)$ is certainly primitive recursively bounded. When p_e converges, we may assume that $\mathtt{LOC}(\alpha_s)$ is self-synchronized but not synchronized with some d which is currently $\not\sim_{\mathtt{LOC}} \alpha_s$. Pick the least such d currently in $\mathcal{A}$.

Assume that $\mathtt{LOC}(d)$ is currently not closed. There is some $d' \in \mathtt{LOC}(d)$ and some $e' \in \mathtt{LOC}(\alpha_s)$ such that $\mathcal{A}(e', d') \neq \mathcal{A}(gf(e'), d')$. But this means that $\mathcal{A}((gf)^i(e'), (gf)^i(d')) \neq \mathcal{A}\left((gf)^{i+1}(e'), (gf)^i(d')\right)$ for every i. So if $\mathtt{LOC}(d', 2s_0^2)$ is not closed, then its elements are all distinct and we can find a sub-interval $\mathtt{LOC}(d', m', m'+2s_0) \subset \mathtt{LOC}(d', 2s_0^2)$ such that for every $z \in \mathtt{LOC}(d', m', m'+2s_0)$, $p_e(z) \in \mathtt{LOC}(\alpha_s)$. This means that there exists some i_0 such that $\mathcal{A}((gf)^{i_0}(e'), (gf)^{i_0}(d')) \neq \mathcal{A}\left((gf)^{i_0+1}(e'), (gf)^{i_0}(d')\right)$ and $p_e\left((gf)^{i_0}(e')\right), p_e\left((gf)^{i_0+1}(e')\right)$ and $p_e((gf)^{i_0}(d'))$ are all in $\mathtt{LOC}(\alpha_s)$. But since p_e is not diagonalized, this means that $\mathtt{LOC}(\alpha_s)$ is not self-synchronized, and so y_0, y_1 can be found by looking far ahead (at these elements just mentioned).

Therefore we may assume that $\mathtt{LOC}(d', 2s_0^2)$ is closed. Let i_1 be the first such that $p_e\left((gf)^{i_1}(e')\right), p_e\left((gf)^{i_1+1}(e')\right) \in \mathtt{LOC}(\alpha_s)$. Then we have $\mathtt{LOC}((gf)^{i_1}(d')) = \mathtt{LOC}(d')$ is also closed, and furthermore if $\mathtt{LOC}\left(p_e\left((gf)^{i_1}(d')\right)\right)$ isn't closed then $p_e\left((gf)^{i_1}(d')\right) \in \mathtt{LOC}(\alpha_s)$ or $p_e\left((gf)^{i_1}(d')\right) \in \mathtt{LOC}(\pi)$ for some pending element π; this is because $p_e\left((gf)^{i_1}(d')\right)$ is an element we previously enumerated in $\mathcal{B}$. Since we fail to diagonalize p_e, if $p_e\left((gf)^{i_1}(d')\right) \in \mathtt{LOC}(\alpha_s)$ then $\mathtt{LOC}(\alpha_s)$ cannot be self-synchronized, and if $p_e\left((gf)^{i_1}(d')\right) \in \mathtt{LOC}(\pi)$ then (π.3) must fail for some pending π.

This means that by the time we find these elements afore-mentioned, the construction under (C.3) would have found and fixed some a such that $\mathtt{LOC}(a)$ is already closed, $\mathtt{LOC}(\alpha_s)$ and $\mathtt{LOC}(a)$ are not synchronized, and $p_e(a)$ has converged such that $\mathtt{LOC}(p_e(a))$ is already closed. Let m be the size of $\mathtt{LOC}(p_e(a))$. The construction henceforth will repeatedly enumerate into $\mathcal{B}$ elements of the form $(gf)^{jm}(z)$ for $j = 1, 2, \cdots$ where $z \in \mathtt{LOC}(\alpha_s)$. Since $\mathtt{LOC}(\alpha_s)$ and $\mathtt{LOC}(a)$ are not synchronized and $\mathtt{LOC}(a)$ is closed, this implies that there exist integers $k_0 < k_1 < \cdots$ such that for every i,

- $\mathcal{A}\left((gf)^{k_i}(\alpha_s), a\right) \neq \mathcal{A}\left((gf)^{k_i+1}(\alpha_s), a\right)$ and
- $p_e\left((gf)^{k_i}(\alpha_s)\right), p_e\left((gf)^{k_i+1}(\alpha_s)\right) \in \mathtt{LOC}(\alpha_s)$.

The sequence $\{k_i\}$ can be obtained primitive recursively. However, for every j, we have $\mathcal{A}(z, p_e(a)) = \mathcal{A}\left((gf)^{jm}(z), p_e(a)\right)$, since $\mathtt{LOC}(p_e(a))$ has size m, so for a large enough i, we must have both $p_e\left((gf)^{k_i}(\alpha_s)\right)$ and $p_e\left((gf)^{k_i+1}(\alpha_s)\right)$ equal to elements of this form. This implies that $\mathcal{A}\left(p_e\left((gf)^{k_i}(\alpha_s)\right), p_e(a)\right) = \mathcal{A}\left(p_e\left((gf)^{k_i+1}(\alpha_s)\right), p_e(a)\right)$, but since p_e is not diagonalized, this implies that $\mathcal{A}\left((gf)^{k_i}(\alpha_s), a\right) = \mathcal{A}\left((gf)^{k_i+1}(\alpha_s), a\right)$, contradicting the choice

of k_i above. Thus, y_0 and y_1 must be found by the construction (or $\texttt{LOC}(\alpha_s)$ closes) before we find all of these elements.

Now we analyze how many stages the construction can act under (C.5) before $\texttt{LOC}(\alpha_s)$ closes. When the construction first gets to (C.5) we have elements y_0, y_1 such that $p_e(y_0), p_e(y_1) \in \texttt{LOC}(\alpha_s)$ and such that $\texttt{LOC}(\alpha_s)$ is not self-synchronized. We set m to be the locus-distance between $p_e(y_0)$ and $p_e(y_1)$.

Since $\texttt{LOC}(\alpha_s)$ is not synchronized with itself, there exists some $z, z' \in \texttt{LOC}(\alpha_s)$ such that $\mathcal{A}(z, z') \neq \mathcal{A}(gf(z), z')$, and these three elements are pairwise distinct. By shifting along $\texttt{LOC}(\alpha_s)$, there exists $y_2 \succ_{\texttt{LOC}(\alpha_s)} y_1$ such that $\mathcal{A}(y_0, y_2) \neq \mathcal{A}(y_1, y_2)$. Without loss of generality, assume that $(y_0, y_2) \notin \mathcal{A}$ and $(y_1, y_2) \in \mathcal{A}$.

Now suppose that there exists some i_0 such that for every $i > i_0$, $\left((gf)^i(\alpha_s), y_0\right) \in \mathcal{A}$. As $y_0 \in \texttt{LOC}(\alpha_s)$, this is equivalent to saying that there exists some i_0 such that $\left((gf)^i(\alpha_s), \alpha_s\right) \in \mathcal{A}$ for every $i > i_0$. The construction would, after a while, only enumerate elements into $\mathcal{B}$ under (C.5.1). In other words, the construction would eventually enumerate elements of the form $(gf)^{2^j k}(\alpha_s)$ for $j = 0, 1, 2, \cdots$ and some fixed k. We may assume $k > i_0$ by choosing the starting term large enough. Now for any $j' > j$, we have $\mathcal{A}\left((gf)^{2^j k}(\alpha_s), (gf)^{2^{j'} k}(\alpha_s)\right) = \mathcal{A}\left(\alpha_s, (gf)^{(2^{j'} - 2^j)k}(\alpha_s)\right)$. Since $(2^{j'} - 2^j)k > k > i_0$ it follows that $\left((gf)^{2^j k}(\alpha_s), (gf)^{2^{j'} k}(\alpha_s)\right) \in \mathcal{A}$ for every j, j'. But now since $(y_0, y_2) \notin \mathcal{A}$, hence $\left((gf)^i(y_0), (gf)^i(y_2)\right) \notin \mathcal{A}$ for all i, and so for large enough i, $p_e\left((gf)^i(y_0)\right)$ and $p_e\left((gf)^i(y_2)\right)$ must both be of the form $(gf)^{2^j k}(\alpha_s)$. This means that $\left(p_e\left((gf)^i(y_0)\right), p_e\left((gf)^i(y_2)\right)\right) \in \mathcal{A}$, but it's impossible since p_e is not diagonalized. An exact symmetric argument works with (y_1, y_2) in place of (y_0, y_2), and (C.5.3) in place of (C.5.1) to show that it is impossible to have cofinitely many i such that $\left((gf)^i(\alpha_s), y_0\right) \notin \mathcal{A}$. The argument above in fact allows us to compute primitive recursively, given any i_u, a bound below which the next element $i_{u+1} > i_u$ such that $\left((gf)^{i_{u+1}}(\alpha_s), y_0\right) \in \mathcal{A}$ and $\left((gf)^{i_{u+1}+1}(\alpha_s), y_0\right) \notin \mathcal{A}$ can be found.

Fix such a sequence $i_0 < i_1 < i_2, \cdots$. We claim that for every u, $p_e\left((gf)^{i_u+1}(\alpha_s)\right)$ cannot be an element enumerated into $\mathcal{B}$ under (C.5). Suppose $p_e\left((gf)^{i_u+1}(\alpha_s)\right)$ is enumerated into $\mathcal{B}$ under (C.5.1) or (C.5.2), then $\left(p_e\left((gf)^{i_u+1}(\alpha_s)\right), p_e(y_0)\right) \in \mathcal{A}$, which means that $\left((gf)^{i_u+1}(\alpha_s), y_0\right) \in \mathcal{A}$, which is a contradiction to the choice of i_u. Thus we must have that $p_e\left((gf)^{i_u+1}(\alpha_s)\right)$ is enumerated into $\mathcal{B}$ under (C.5.3). But in this case, $\left((gf)^{\pm m}\left(p_e\left((gf)^{i_u+1}(\alpha_s)\right)\right), p_e(y_0)\right) \notin \mathcal{A}$, which means that $\left(p_e\left((gf)^{i_u+1}(\alpha_s)\right), (gf)^{\pm m}\left(p_e(y_0)\right)\right) \notin \mathcal{A}$. Since $p_e(y_1) = (gf)^{\pm m}\left(p_e(y_0)\right)$ this means that $\left((gf)^{i_u}(\alpha_s), y_0\right) = \left((gf)^{i_u+1}(\alpha_s), y_1\right) \notin \mathcal{A}$. But this also contradicts the choice of i_u. Thus $\texttt{LOC}(\alpha_s)$ must close before these elements are found. This allows us to compute primitive recursively a bound for the stage where $t(\alpha_s)$ is defined.

We have shown that t_e is primitive recursive. If $t(\alpha_s)$ is defined under (C.2) then $\texttt{LOC}(\alpha_s, t(\alpha_s))$ is closed. Otherwise if $t_e(\alpha_s)$ is defined under (C.4), then α_s is declared to be pending. Since we never enter the diagonalization phase for e, it follows that (α_s.3) holds at every stage after $t(\alpha_s)$ is defined. Hence $\texttt{LOC}(\alpha_s)$ is synchronized with $\texttt{LOC}(b)$ for every $b \in \mathcal{A}$. So, t_e is the function we need.

4. Is the dense linear order dense enough? Proof of Theorem 1.7

We prove that $\mathbf{FPR}(\eta)$ is downwards dense. Fix a copy $\mathcal{A}$ of η. We shall build $\mathcal{B}$ and infinitely many $\mathcal{B}_e$ (each $\mathcal{B}_e$ uses p_e). In fact, we build the pairs $(\mathcal{B}, q)$ and $(\mathcal{B}_e, q_e)$ such that $q : \mathcal{B} \to \mathcal{A}$ and $q_e : \mathcal{B}_e \to \mathcal{A}$. We need to meet the requirements

$$R_{e,i} \ : \ \text{Ensure that } p_e : \mathcal{A} \to \mathcal{B} \text{ fails or } p_i : \mathcal{A} \to \mathcal{B}_e \text{ fails.}$$

The proof is obviously a non uniform argument; if for every e there exists an i such that $R_{e,i}$ is met via the first disjunct, then the structure $\mathcal{B} <_{pr} A$. Otherwise for some e we meet every $R_{e,i}$ via the second disjunct, then $\mathcal{B}_e <_{pr} A$.

Informal description of strategies. We now describe the strategy of a single $R_{e,i}$. First pick $a_0 \in \mathcal{A}$ and keep it out of $Rng(q_e)$. (Note: this conflicts with making q_e total, of course, but this "restraint" is only for finitely many stages). We wait for $p_i(a_0)\downarrow$. Meanwhile, grow $(\mathcal{B}, q)$ and $(\mathcal{B}_e, q_e)$ appropriately.

Suppose that $p_i(a_0)\downarrow$. Clearly $q_e(p_i(a_0)) \neq a_0$, so WLOG assume that $x = q_e(p_i(a_0)) > a_0$. We release the restraint on a_0 and now restrain $\frac{a_0+x}{2}$. Wait for $p_i\left(\frac{a_0+x}{2}\right)\downarrow$. Now release the restraint on $\frac{a_0+x}{2}$. Now let $b_0 = \frac{a_0+x}{2}$ we now have $(a_0, b_0) \cap (a_1, b_1) = \emptyset$ where $a_1 = q_e(p_i(a_0))$ and $b_1 = q_e(p_i(b_0))$.

Now it is easy to see that we can go on further and arrange for (a_2, b_2) so that $(a_0, b_0), (a_1, b_1)$ and (a_2, b_2) are pairwise disjoint such that $q_e(p_i(a_1)) = a_2$ and $q_e(p_i(b_1)) = b_2$. (Here we may not succeed if p_i isn't onto, and we may have to shrink the intervals (a_0, b_0) and (a_1, b_1) further). The picture we are aiming for is in Figure 1.

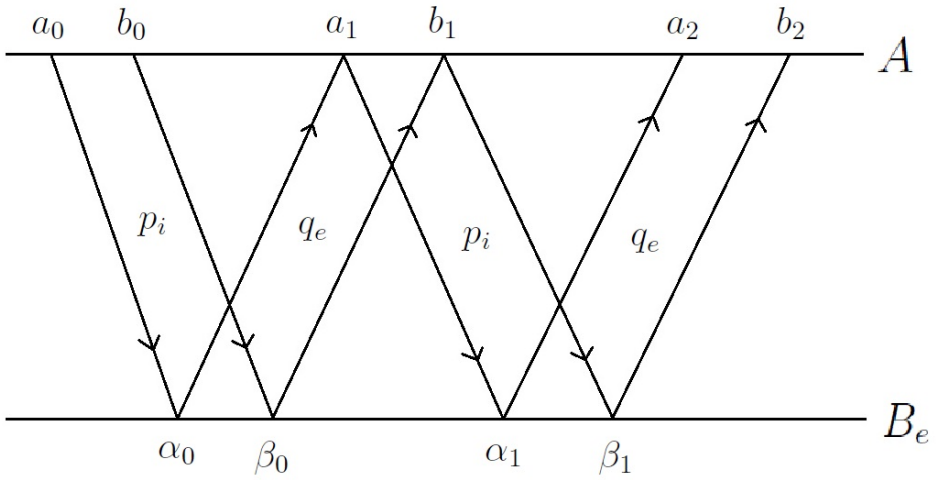

FIGURE 1. The setup

Now next we wait for $\mathcal{B}$ to set up. By restricting the intervals down, we may assume that we also have $(a_1, b_1) \cap (a_3, b_3) = \emptyset$ where $a_3 = q(p_e(a_1))$ and $b_3 = q(p_e(b_1))$ and also $(a_1, b_1) \cap (a_4, b_4) = \emptyset$ where $a_4 = q(p_e(a_2))$ and $b_4 = q(p_e(b_2))$. Finally when all these intervals are set up as desired, we enter the *diagonalization phase*.

Note that during the *waiting phase* the requirement simply sits around and waits, and imposes no restrictions on the enumeration of elements into $\mathcal{B}$ and $\mathcal{B}_e$. (The only restraints in the waiting phase is restraining elements from being put into the range of q and q_e, which are finitary). We may wait forever in the waiting phase for a certain requirement because p_e or p_i isn't onto, but then we win the entire requirement this way with no other effect on the rest of the construction. However, once a requirement enters the diagonalization phase, then control of the construction, as well as the control of the enumeration of $\mathcal{B}$ and $\mathcal{B}_e$, is handed over to the requirement $R_{e,i}$. We will see that diagonalization phase only lasts for finitely many stages.

We now describe what happens when $R_{e,i}$ is in the diagonalization phase. We monitor the enumeration of elements into $\mathcal{A}$. If elements enter outside of (a_1, b_1) or (a_2, b_2), then we

enumerate promptly into $\mathcal{B}_e$; note that this does not cause us to increase (α_0, β_0) or (α_1, β_1). In the diagonalization phase, we are waiting for a stage s^* such that either

- $\#(a_0, b_0) > \#(\alpha_0, \beta_0)$, or
- $\#(a_1, b_1) > \#(\alpha_1, \beta_1)$.

Claim 4.1. *Before stage s^* we will not enumerate any elements inside (α_0, β_0).*

Proof. We proceed inductively. If elements enter $\mathcal{A}$ outside of the interval (a_1, b_1), then we will not be forced to increase (α_0, β_0). Otherwise if an element enters (a_1, b_1), then we temporarily delay enumerating anything into $\mathcal{B}_e$ until $\#(a_1, b_1)$ catches up with $\#(\alpha_1, \beta_1)$. If this happens then we have reached stage s^*. Otherwise $\#(\alpha_1, \beta_1)$ must increase and so we do not need to enumerate anything else into (α_0, β_0). In any case, the enumeration of $\mathcal{B}_e$ is only delayed by a primitive recursive amount. □

Claim 4.2. *s^* exists.*

Proof. Otherwise by the above claim, $\#(\alpha_0, \beta_0) < \infty$. However, as s^* is never found, this means that $\#(a_0, b_0) \leq \#(\alpha_0, \beta_0) < \infty$, but this is impossible as $\mathcal{A}$ is a copy of $\mathbb{Q}$. □

Now we assume that we have reached stage s^* after a primitive recursive delay on $\mathcal{B}_e$. Assume WLOG that $\#(a_0, b_0) > \#(\alpha_0, \beta_0)$. Now we would like to restrain the interval (α_0, β_0) of $\mathcal{B}_e$ from growing, while waiting for p_i to converge on every element of (a_0, b_0). Once this happens, we will be able to kill p_i. Unfortunately, we cannot do this if the interval (a_1, b_1) grows quickly, because we will have to delay enumerating any new element into $\mathcal{B}$ until p_i have converged on all elements of (a_0, b_0). This requires a delay determined by p_i, which unfortunately is too much for the structure $\mathcal{B}_e$. The trick is to observe that if all subsequent elements of $\mathcal{A}$ appears in the interval (a_1, b_1), then we will be able to kill p_e.

Delay enumerating the next element of $\mathcal{B}_e$ and wait for either

(a) enough elements to be enumerated into (a_1, b_1) consecutively, with nothing else enumerated into $\mathcal{A}$ outside of (a_1, b_1), and for p_e to converge on all these elements, or
(b) some number to be enumerated in $\mathcal{A} - (a_1, b_1)$ interrupting (a).

Note that this additional wait adds a further primitive recursive delay to the enumeration of $\mathcal{B}_e$ by a factor of p_e, which the enumeration of $\mathcal{B}_e$ is allowed access to.

If case (a) happens, then enough elements has entered (a_1, b_1) consecutively, which means that $\#(a_1, b_1) > \#(p_e(a_1), p_e(b_1))$. Notice that $\#(p_e(a_1), p_e(b_1))$ did not increase because nothing has entered (a_3, b_3) in the mean time. Furthermore, as p_e has converged on all these elements in (a_1, b_1), this means that $p_e : \mathcal{A} \to \mathcal{B}$ is killed permanently.

If case (b) happens, then we can now finally enumerate the next element of $\mathcal{B}_e$, but outside of the interval (α_0, β_0), since case (b) says that we have just discovered a new addition to $\mathcal{A} - (a_1, b_1)$. Then repeat the procedure above, delaying the next element of $\mathcal{B}_e$ this way until (a) or (b) applies. If we ever hit case (a) then we satisfy the requirement $R_{e,i}$ via killing off $p_e : \mathcal{A} \to \mathcal{B}$. Otherwise we keep having case (b) apply. However this means that we will never increase $\#(\alpha_0, \beta_0)$ at all. After finitely many stages, as $\#(a_0, b_0) > \#(\alpha_0, \beta_0)$, when p_i has converged on enough elements of (a_0, b_0), we will be able to kill off $p_i : \mathcal{A} \to \mathcal{B}_e$ permanently.

Note that the diagonalization phase will only run for finitely many stages. Also, note that the enumeration of $(\mathcal{B}, q)$ is never delayed, and when some requirement in the diagonalization phase is delaying the definition of $\mathcal{B}_e$, all other structures $\mathcal{B}_k$ for $k \neq e$ are still enumerated quickly.

Formal construction for Theorem 1.7. As described above we fix a copy $\mathcal{A}$ of the countable dense linear order η. We shall build the fpr structure $\mathcal{B}$ and infinitely many computable copies

$\mathcal{B}_e$ out of which some (possibly all) will be fpr. We shall build primitive recursive isomorphisms $q : \mathcal{B} \to \mathcal{A}$ and $q_e : \mathcal{B}_e \to \mathcal{A}$, and meet the requirements $R_{e,i}$ above.

The requirement $R_{e,i}$ will begin by defining the parameters $a_0^{e,i}$ and $b_0^{e,i}$. Given these two numbers, we let

- $\alpha_0^{e,i} = p_i(a_0^{e,i})$ and $\beta_0^{e,i} = p_i(b_0^{e,i})$,
- $a_1^{e,i} = q_e(\alpha_0^{e,i})$ and $b_1^{e,i} = q_e(\beta_0^{e,i})$,
- $\alpha_1^{e,i} = p_i(a_1^{e,i})$ and $\beta_1^{e,i} = p_i(b_1^{e,i})$,
- $a_2^{e,i} = q_e(\alpha_1^{e,i})$ and $b_2^{e,i} = q_e(\beta_1^{e,i})$,

To simplify notations, we write (x, y) to be the interval of all points strictly between x and y, even if $y < x$.

The construction will proceed in the following way. At a certain stage we will pick a requirement $R_{e,i}$ and give control of the construction to $R_{e,i}$. We will then keep acting for the sake of $R_{e,i}$ for finitely many stages, after which $R_{e,i}$ relinquishes control of the construction. We will then pick another (possibly the same) requirement and give control to the requirement, and so on. When we say that we *extend* $\mathcal{B}$ *with* $x \in \mathcal{A}$ *restrained* at a stage s, we mean that we pick an element $r \in \mathcal{A}[s]$ with the least index such that $r \neq x$ and $r \notin Rng(q)$, and we enumerate the next element y into $\mathcal{B}$ in the appropriate location and define $q(y) = r$ such that q is order-preserving. If r does not yet exist, speed up the enumeration of $\mathcal{A}$ by two elements, where we must have r. The same goes for $\mathcal{B}_e$ and q_e. To extend a structure *freely* means to extend it with no element restrained.

At a stage s, we say that $R_{e,i}$ requires attention if it is not yet satisfied, and if we either have never acted for $R_{e,i}$ before, or the re-computation of the parameters (according to the basic strategy) are now complete. We say that $R_{e,i}$ is satisfied at a stage if either p_e or p_i is not order-preserving. At stage s if no requirement is currently in control, we extend every structure freely. Pick the highest priority requirement $R_{e,i}$ requiring attention, and hand control to $R_{e,i}$. What this means is the following. In the following we drop the superscripts e, i from all parameters.

(i) If $R_{e,i}$ has never before acted, then all parameters associated with $R_{e,i}$ are undefined. We perform the following actions.
- First pick a fresh value $a_0 \in \mathcal{A}$ different from all other parameters in the construction, and such that $a_0 \notin Rng(q_e)$.
- Next we wait for α_0, α_1, a_1 and a_2 to be computed. These must show up after a finite delay (note that q and q_e are always defined on every element currently present in $\mathcal{B}$ and $\mathcal{B}_e$). While waiting we extend $\mathcal{B}_e$ restraining a_0, and extend $\mathcal{B}$ and all other $\mathcal{B}_k$ freely. Notice that as we restrained a_0, we must have $a_1 \neq a_0$, and therefore we have either $a_0 < a_1 < a_2$ and $\alpha_0 < \alpha_1$, or we have $a_0 > a_1 > a_2$ and $\alpha_0 > \alpha_1$.
- Now pick b_0 to be any fresh value strictly between a_0 and a_1. (Of course there may be currently no choice for b_0, in which case we simply wait for finitely many stages, extending all structures freely during the wait). Once b_0 is defined, we wait until β_0, β_1, b_1 and b_2 to be computed. While waiting we extend all structures freely. Now we must have $a_0 < b_0 < a_1 < b_1 < a_2 < b_2$ and $\alpha_0 < \beta_0 < \alpha_1 < \beta_1$, or the other way round.
- Now pick a fresh element $\hat{a}_2$ strictly between a_2 and b_2 such that $\hat{a}_2 \notin Rng(q)$. While waiting we extend all structures freely. Once $\hat{a}_2$ is found, we compute $q(p_e(\hat{a}_2))$, and while waiting we extend $\mathcal{B}$ with $\hat{a}_2$ restrained, and all other structures freely. We must have $\hat{a}_2 \neq q(p_e(\hat{a}_2))$.

- Now pick a fresh element $\hat{b}_2$ strictly between a_2 and b_2 and also strictly between $\hat{a}_2$ and $q(p_e(\hat{a}_2))$. Wait for $q(p_e(\hat{b}_2))$ to be computed, extending all structures freely. Therefore we must have $\left(\hat{a}_2, \hat{b}_2\right)$ and $\left(q(p_e(\hat{a}_2)), q(p_e(\hat{b}_2))\right)$ non-empty and disjoint.
- Now release control of the construction, and wait for a new pair $\hat{a}_0, \hat{b}_0$ such that $q_e(p_i(q_e(p_i(\hat{a}_0)))) = \hat{a}_2$ and $q_e(p_i(q_e(p_i(\hat{b}_0)))) = \hat{b}_2$.

(ii) If $R_{e,i}$ was waiting for $\hat{a}_0$ and $\hat{b}_0$ to be found and has now found them, we do the following. Note that as p_i is assumed to be order preserving (q and q_e certainly are), the "hat" intervals have to be nested within the "unhatted" intervals. Thus we must have $\left(\hat{a}_0, \hat{b}_0\right), \left(q_e(p_i(\hat{a}_0)), q_e(p_i(\hat{b}_0))\right)$ and $\left(\hat{a}_2, \hat{b}_2\right)$ non-empty and mutually disjoint. The same goes for the $\mathcal{B}_e$-intervals $\left(p_i(\hat{a}_0), p_i(\hat{b}_0)\right)$ and $\left(p_i(q_e(p_i(\hat{a}_0))), p_i(q_e(p_i(\hat{b}_0)))\right)$.

- Our next action is to pick a fresh element $\tilde{a}_1$ strictly between $\hat{a}_1$ and $\hat{b}_1$ such that $\tilde{a}_1 \notin Rng(q)$. While waiting we extend all structures freely. Once $\tilde{a}_1$ is found, we compute $q(p_e(\tilde{a}_1))$, and while waiting we extend $\mathcal{B}$ with $\tilde{a}_1$ restrained, and all other structures freely. We must have $\tilde{a}_1 \neq q(p_e(\tilde{a}_1))$.
- Now pick a fresh element $\tilde{b}_1$ strictly between $\hat{a}_1$ and $\hat{b}_1$ and also strictly between $\tilde{a}_1$ and $q(p_e(\tilde{a}_1))$. Wait for $q(p_e(\tilde{b}_1))$ to be computed, extending all structures freely. Therefore we must have $\left(\tilde{a}_1, \tilde{b}_1\right)$ and $\left(q(p_e(\tilde{a}_1)), q(p_e(\tilde{b}_1))\right)$ non-empty and disjoint.
- Now release control of the construction, and wait for a new pair $\tilde{a}_0, \tilde{b}_0$ such that $q_e(p_i(\tilde{a}_0)) = \tilde{a}_1$ and $q_e(p_i(\tilde{b}_0)) = \tilde{b}_1$.

(iii) Suppose $R_{e,i}$ was waiting for $\tilde{a}_0$ and $\tilde{b}_0$ to be found and has now found them. Note that the "tilde" intervals have to be nested within the "hat" intervals. Thus we must have $\left(\tilde{a}_0, \tilde{b}_0\right), \left(\tilde{a}_1, \tilde{b}_1\right)$ and $\left(q_e(p_i(\tilde{a}_1)), q_e(p_i(\tilde{b}_1))\right)$ non-empty and mutually disjoint. The same goes for the $\mathcal{B}_e$-intervals $\left(p_i(\tilde{a}_0), p_i(\tilde{b}_0)\right)$ and $\left(p_i(\tilde{a}_1), p_i(\tilde{b}_1)\right)$.

Now we call $\tilde{a}_2 = q_e(p_i(\tilde{a}_1)), \tilde{b}_2 = q_e(p_i(\tilde{b}_1)), \tilde{\alpha}_0 = p_i(\tilde{a}_0), \tilde{\beta}_0 = p_i(\tilde{b}_0), \tilde{\alpha}_1 = p_i(\tilde{a}_1), \tilde{\beta}_1 = p_i(\tilde{b}_1)$. We will also call $\tilde{a}_3 = q(p_e(\tilde{a}_1))$ and $\tilde{b}_3 = q(p_e(\tilde{b}_1))$. Do the following.

- Compute $\tilde{a}_4 = q(p_e(\tilde{a}_2))$ and $\tilde{b}_4 = q(p_e(\tilde{b}_2))$, and while waiting, extend all structures freely.
- Suppose now we find $\tilde{a}_4$ and $\tilde{b}_4$. We are now ready to begin diagonalization for $R_{e,i}$. Diagonalization will take only finitely many stages, after which control will be released. To summarise, if we are here, we have successfully computed the following classes of intervals. Each class contains intervals which are pairwise disjoint:
  - $(\tilde{a}_0, \tilde{b}_0), (\tilde{a}_1, \tilde{b}_1), (\tilde{a}_2, \tilde{b}_2)$.
  - $(\tilde{a}_1, \tilde{b}_1), (\tilde{a}_3, \tilde{b}_3)$.
  - $(\tilde{a}_2, \tilde{b}_2), (\tilde{a}_4, \tilde{b}_4)$.

  Now we begin diagonalization for $R_{e,i}$. We wait for a stage s^* such that either $\#(\tilde{a}_0, \tilde{b}_0) > \#(\tilde{\alpha}_0, \tilde{\beta}_0)$, or $\#(\tilde{a}_1, \tilde{b}_1) > \#(\tilde{\alpha}_1, \tilde{\beta}_1)$. While we wait for the stage s^*, we extend all structures freely except for $\mathcal{B}_e$. For $\mathcal{B}_e$ we will extend it restraining every element in the interval $(\tilde{a}_1, \tilde{b}_1)$.

  Note that we will always find s^* after a finite delay, because while waiting we extend $\mathcal{B}_e$ by restraining every element in the interval $(\tilde{a}_1, \tilde{b}_1)$, which means that we will never enumerate any additional element into $\mathcal{B}_e$ in the interval $(\tilde{\alpha}_0, \tilde{\beta}_0)$, but $\mathcal{A}$ is a copy of η which means that the interval $(\tilde{a}_0, \tilde{b}_0)$ has to be infinite, and so certainly we must eventually see $\#(\tilde{a}_0, \tilde{b}_0) > \#(\tilde{\alpha}_0, \tilde{\beta}_0)$.

- Suppose we have now found s^*. Without loss of generality, assume that $\#(\tilde{a}_0, \tilde{b}_0) > \#(\tilde{\alpha}_0, \tilde{\beta}_0)$. If instead $\#(\tilde{a}_1, \tilde{b}_1) > \#(\tilde{\alpha}_1, \tilde{\beta}_1)$ holds, we will repeat the steps below with $(\tilde{a}_2, \tilde{b}_2)$ instead of $(\tilde{a}_1, \tilde{b}_1)$.
  Now we wait for a stage $s^{**} > s^*$ such that either p_i or p_e is found to be not order preserving. While waiting for s^{**} we extend all structures freely except for $\mathcal{B}_e$. For $\mathcal{B}_e$ we will extend it restraining every element in the interval $(\tilde{a}_1, \tilde{b}_1)$.
- Again, it is clear that s^{**} will be found after a finite delay, as we never enumerate anything into $\mathcal{B}$ in the interval $(\tilde{\alpha}_0, \tilde{\beta}_0)$. Once s^{**} is found, we release control of the construction and note that $R_{e,i}$ is now satisfied.

This ends the description of the construction.

Verification. It is easy to check the construction to see that every time a requirement is given control, it will release control after finitely many steps. The only facts we use here are that $\mathcal{A}$ is a copy of η and all functions are total. Therefore, there are infinitely many stages where all structures are extended freely (namely, at those stages where we give control to a different requirement). Since the functions q and q_e are clearly injective at every stage, this means that q and q_e are onto isomorphisms. These functions will be primitive recursive if the structures $\mathcal{B}$ and $\mathcal{B}_e$ can be computed with a primitive recursive delay. By examining the construction, it is clear that $\mathcal{B}$ is extended at every stage with at most a single element being restrained. Therefore, the next element of $\mathcal{B}$ is enumerated by looking ahead at the enumeration of $\mathcal{A}$ by at most two stages. Therefore $\mathcal{B}$ is an fpr structure and $q : \mathcal{B} \to \mathcal{A}$ is a primitive recursive isomorphism.

Now suppose that for every e, p_e is not order-preserving. This means that $\mathcal{B} <_{pr} \mathcal{A}$ and we are done. Therefore we fix some e such that p_e is order-preserving. (Note that even if p_e is not onto, we are still in this case). This means that for every i, either $R_{e,i}$ is never satisfied, or it is satisfied by discovering that p_i is not order-preserving. First we argue that $\mathcal{B}_e$ can be computed with a primitive delay, using p_e as a parameter. The only stages in which this might be a problem are the stages where we are diagonalizing for some $R_{e,i}$ and waiting for s^* or s^{**}.

Fix an i and a point in the construction where we are diagonalizing for $R_{e,i}$ and waiting for s^*. Before s^* is found we must extend $\mathcal{B}_e$ while restraining $(\tilde{a}_1, \tilde{b}_1)$. At any stage t where we are doing this, we only have to look ahead at the next t many elements of $\mathcal{A}$. It cannot be that the next t many elements of $\mathcal{A}$ all appear in the interval $(\tilde{a}_1, \tilde{b}_1)$, because if this were the case, then $\mathcal{B}_e$ will have at most t many elements at stage $2t$. Since at stage $2t$, we have $\#(\tilde{a}_1, \tilde{b}_1) \geq t > \#(\tilde{\alpha}_1, \tilde{\beta}_1)$, which means that s^* would be found at or before stage $2t$, and we would have progressed to the next step. Therefore at each stage t, either we find s^* by stage $2t$, or we must find some element enumerated into $\mathcal{A}$ outside the interval $(\tilde{a}_1, \tilde{b}_1)$. Such an element is not restrained and so we will be able to extend $\mathcal{B}$ by stage $2t$.

Suppose at stage t we are waiting for s^{**}. We wish to compute a primitive recursive bound on the delay on the next element enumerated in $\mathcal{B}_e$. Suppose we have $\#(\tilde{a}_0, \tilde{b}_0) > \#(\tilde{\alpha}_0, \tilde{\beta}_0)$. Now we claim that at stage u we must either find s^{**} or discover an element enumerated into $\mathcal{A}$ outside the interval $(\tilde{a}_1, \tilde{b}_1)$, where u is the number of stages required for $p_e(x)$ to converge on every element in $\mathcal{A}[2t]$. Suppose this is not the case; then every element enumerated into $\mathcal{A}$ between t and u all appear in the interval $(\tilde{a}_1, \tilde{b}_1)$. As $(\tilde{a}_1, \tilde{b}_1)$ and $(\tilde{a}_3, \tilde{b}_3)$ are disjoint, this means that at stage u, $\#(\tilde{a}_3, \tilde{b}_3) < t$. Therefore, at stage u, we also have $\#(p_e(\tilde{a}_1), p_e(\tilde{b}_1)) < t$. However, at stage $2t$, the size $\#(\tilde{a}_1, \tilde{b}_1) \geq t$, and at stage u, p_e would have converged on all these elements. Therefore p_e cannot be order preserving, a contradiction to our case assumption. This means that by stage u we must either find s^{**} or discover an element enumerated into

$\mathcal{A}$ outside the interval $(\tilde{a}_1, \tilde{b}_1)$, and in either case, we are able to enumerate the next element of $\mathcal{B}_e$. Hence $\mathcal{B}_e$ is only delayed by a primitive recursive amount (with parameter p_e).

Therefore $\mathcal{B}_e$ is an fpr structure and $q_e : \mathcal{B}_e \to \mathcal{A}$ is a primitive recursive isomorphism. Thus $\mathcal{B}_e \leq_{pr} \mathcal{A}$. We now show that for every i, either $R_{e,i}$ is satisfied or p_i is not onto. Fix i such that $R_{e,i}$ is never satisfied. This means that we wait forever for $\hat{a}_0$ and $\hat{b}_0$ or for $\tilde{a}_0$ and $\tilde{b}_0$. Since q_e is onto, this means that p_i is not onto. Therefore $\mathcal{A} \not\leq_{pr} \mathcal{B}_e$.

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.

[Ala16a] P. E. Alaev. Atomless Boolean algebras computable in polynomial time. *Sib. Èlektron. Mat. Izv.*, 13:1035–1039, 2016.

[Ala16b] P. E. Alaev. Existence and uniqueness of structures computable in polynomial time. *Algebra and Logic*, 55(1):72–76, Mar 2016.

[BS11] Gábor Braun and Lutz Strüngmann. Breaking up finite automata presentable torsion-free abelian groups. *Internat. J. Algebra Comput.*, 21(8):1463–1472, 2011.

[CDRU09] Douglas Cenzer, Rodney G. Downey, Jeffrey B. Remmel, and Zia Uddin. Space complexity of abelian groups. *Arch. Math. Log.*, 48(1):115–140, 2009.

[CR] D. Cenzer and J.B. Remmel. Polynomial time versus computable boolean algebras. *Recursion Theory and Complexity, Proceedings 1997 Kazan Workshop (M. Arslanov and S. Lempp eds.), de Gruyter (1999), 15-53.*

[CR91] Douglas A. Cenzer and Jeffrey B. Remmel. Polynomial-time versus recursive models. *Ann. Pure Appl. Logic*, 54(1):17–58, 1991.

[CR92] Douglas A. Cenzer and Jeffrey B. Remmel. Polynomial-time abelian groups. *Ann. Pure Appl. Logic*, 56(1-3):313–363, 1992.

[ECH+92] David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston. *Word processing in groups*. Jones and Bartlett Publishers, Boston, MA, 1992.

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

[Gri90] Serge Grigorieff. Every recursive linear ordering has a copy in dtime-space(n, log(n)). *J. Symb. Log.*, 55(1):260–276, 1990.

[KMN] I. Kalimullin, A. Melnikov, and K.M. Ng. Algebraic structures computable without delay. *Submitted.*

[KMN17] I. S. Kalimullin, A. G. Melnikov, and K. M. Ng. The diversity of categoricity without delay. *Algebra and Logic*, 56(2):171–177, May 2017.

[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.

[KNRS07] Bakhadyr Khoussainov, André Nies, Sasha Rubin, and Frank Stephan. Automatic structures: richness and limitations. *Log. Methods Comput. Sci.*, 3(2):2:2, 18, 2007.

[Mac11] Dugald Macpherson. A survey of homogeneous structures. *Discrete Math.*, 311(15):1599–1634, 2011.

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

[Mel17] Alexander G. Melnikov. *Eliminating Unbounded Search in Computable Algebra*, pages 77–87. Springer International Publishing, Cham, 2017.

[NR90] A. Nerode and J. B. Remmel. Polynomial time equivalence types. In *Logic and computation (Pittsburgh, PA, 1987)*, volume 106 of *Contemp. Math.*, pages 221–249. Amer. Math. Soc., Providence, RI, 1990.

[NS07] André Nies and Pavel Semukhin. Finite automata presentable abelian groups. In *Logical foundations of computer science*, volume 4514 of *Lecture Notes in Comput. Sci.*, pages 422–436. Springer, Berlin, 2007.

[NT08] André Nies and Richard M. Thomas. FA-presentable groups and rings. *J. Algebra*, 320(2):569–585, 2008.

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

[Tsa11] T. Tsankov. The additive group of the rationals does not have an automatic presentation. *J. Symbolic Logic*, 76(4):1341–1351, 2011.