COMPUTABLE LINEAR ORDERS AND PRODUCTS

ANDREY N. FROLOV, STEFFEN LEMPP, KENG MENG NG, AND GUOHUA WU

Abstract. We characterize the linear order types τ with the property that given any countable linear order $\mathcal{L}$, $\tau \cdot \mathcal{L}$ is a computable linear order iff $\mathcal{L}$ is a computable linear order, as exactly the finite nonempty order types.

Contents

1. Introduction

The study of effectiveness in the area of linear orders has a long history; we refer to Downey [Do98] for a summary from the late 1990's. One of the main techniques used in the study of computable linear orders is the use of products by fixed "simple" linear orders (on the left) to bootstrap the complexity by one or more jumps. We will give some examples of this below; but the main purpose of this paper is to study what effect of taking a product has on the computational complexity of linear orders in general.

We thus consider the following general over-arching

2010 *Mathematics Subject Classification.* 03D45.

Key words and phrases. computable linear order, product.

The first author is supported by the grant of RF President MD-2721.2019.1. The second author's research was partially supported by NSF grant DMS-1600228 and AMS-Simons Foundation Collaboration Grant 209087. The third author is partially supported by the grants MOE2015-T2-2-055 and RG131/17. The fourth author is partially supported by M4020333 (MOE2016-T2-1-083), M4011672 (RG32/16) and M4011274 (RG29/14) from Ministry of Education Singapore. The first and second authors wish to thank the New Zealand Mathematics Research Institute, during the 2017 Summer School of which part of this research was carried out. The second author also wishes to thank Nanyang Technological University where part of this research was carried out.

Problem 1.1. *Characterize, for any $n > 0$, the order types τ such that, for any countable linear order $\mathcal{L}$, $\mathcal{L}$ has a $\mathbf{0}^{(n)}$-computable copy iff $\tau \cdot \mathcal{L}$ has a computable copy.*

This problem was mentioned in [Do98] (see the discussion below Question 2.1). For $n = 1$, Downey and Knight [DK92] constructed an example of such a τ. Namely, they proved that, for any $\mathcal{L}$, $\mathcal{L}$ has a $\mathbf{0}'$-computable copy iff $(\eta + 2 + \eta) \cdot \mathcal{L}$ has a computable copy. Frolov showed that, for any $k \in \omega$, $\mathcal{L}$ has a $\mathbf{0}'$-computable copy iff $(\eta + k + 2 + \eta) \cdot \mathcal{L}$ has a computable copy. Indeed, he proved [Fr06, Corollary 1] that if τ has neither a least nor a greatest element, and a linear order $\mathcal{L}$ is $\mathbf{0}'$-computable, then $\tau \cdot \mathcal{L}$ has a computable presentation. Later, Frolov noted in [Fr12] that the latter can be improved to the following

Theorem 1.2. *If τ does not contain a least element, or does not contain a greatest element, and a linear order $\mathcal{L}$ is $\mathbf{0}'$-computable, then $\tau \cdot \mathcal{L}$ has a computable presentation.*

For $n = 2$, Fellner [Fe76] proved that, for any countable linear order $\mathcal{L}$, $\mathcal{L}$ has a $\mathbf{0}''$-computable copy iff $\zeta \cdot \mathcal{L}$ has a computable copy. Ash and Knight [AK00] showed the same for $\omega \cdot \mathcal{L}$ in place of $\zeta \cdot \mathcal{L}$.

Alaev, Thurber and Frolov [ATF09] proved that if τ does not contain a least element, or does not contain a greatest element, and there are no τ_1, τ_2, τ_3 such that $\tau = \tau_1 + \tau_2 + \tau_3$ and $\tau_1, \tau_3 \in \{\eta, 1 + \eta, \eta + 1\}$, then $\tau \cdot \mathcal{L}$ has a computable copy for any $\mathbf{0}''$-computable linear order $\mathcal{L}$. From this, some examples of τ for Problem 1.1 can be deduced for $n = 2$. In particular, for any countable linear order $\mathcal{L}$, $\mathcal{L}$ has a $\mathbf{0}''$-computable copy iff $(\eta + \omega^*) \cdot \mathcal{L}$ has a computable copy iff $(\omega^* + \omega + \omega) \cdot \mathcal{L}$ has a computable copy, among other results.

2. The Main Theorem

In this paper, we give a complete answer to Problem 1.1 for $n = 0$ and discuss for the case $n = 1$. For the case $n = 0$, we prove the following

Theorem 2.1. *The following are equivalent:*

1. *τ is finite and nonempty.*
2. *For any countable linear order $\mathcal{L}$, $\mathcal{L}$ has a computable copy iff $\tau \cdot \mathcal{L}$ has a computable copy.*
3. *For any countably infinite linear order $\mathcal{L}$, $\mathcal{L}$ has a computable copy iff $\tau \cdot \mathcal{L}$ has a computable copy.*

Proof. The direction "(2) $\Rightarrow$ (3)" is obvious. So we need to prove both "(3) $\Rightarrow$ (1)" and "(1) $\Rightarrow$ (2)".

To prove "(3) $\Rightarrow$ (1)", we note first that τ must be computable[1]. Indeed, let $\mathcal{L}$ be an infinite linear order with a least element and a greatest element. Then a computable presentation of $\tau \cdot \mathcal{L}$ contains points t_1, t_2 such that $[t_1, t_2] \cong \tau$. So τ has a computable copy. Obviously, τ is not empty.

For contradiction, we assume that τ is infinite, and we consider three cases.

Case 1: τ fails to contain a least, or a greatest, element.

[1] If $\mathcal{L}$ is arbitrary from Condition(2) then, to prove that τ is computable, it is enough to consider $\mathcal{L} = 1$.

Then if $\mathcal{L}$ is a $\mathbf{0}'$-computable linear order without computable copy, then, by Theorem 1.2, $\tau \cdot \mathcal{L}$ has a computable copy, contradicting Condition(3).

Case 2: τ contains a limit point.

Suppose for definiteness that τ has a left limit point. Thus $\tau = \tau_1 + 1 + \tau_2$, where τ_1 has no greatest element. Let $\mathcal{L} \cong \zeta \cdot \mathcal{L}_0$ be a $\mathbf{0}'$-computable linear order without computable copy. Then, $\tau \cdot \mathcal{L} \cong \tau \cdot (\zeta \cdot \mathcal{L}_0) \cong (\cdots + \tau_1 + 1 + \tau_2 + \tau_1 + 1 + \tau_2 + \cdots) \cdot \mathcal{L}_0 \cong (1 + \tau_2 + \tau_1) \cdot (\zeta \cdot \mathcal{L}_0) \cong \tau_0 \cdot \mathcal{L}$, where $\tau_0 = 1 + \tau_2 + \tau_1$ is computable and has no greatest element. From Theorem 1.2, it follows that $\tau_0 \cdot \mathcal{L}$, and hence that $\tau \cdot \mathcal{L}$, has a computable copy, again contradicting Condition (3).

Case 3: τ is infinite without limit point but with a least element and a greatest element.

Thus τ is discrete and $\tau = \omega + \zeta \cdot \tau_0 + \omega^*$. Again, let $\mathcal{L} \cong \zeta \cdot \mathcal{L}_0$ be a $\mathbf{0}'$-computable linear order without computable copy. Then, $\tau \cdot \mathcal{L} \cong \tau \cdot (\zeta \cdot \mathcal{L}_0) \cong (\cdots + \omega + \zeta \cdot \tau_0 + \omega^* + \omega + \zeta \cdot \tau_0 + \omega^* + \cdots) \cdot \mathcal{L}_0 \cong (\omega^* + \omega + \zeta \cdot \tau_0) \cdot (\zeta \cdot \mathcal{L}_0) \cong (\zeta \cdot (1 + \tau_0)) \cdot \mathcal{L} \cong \tau_1 \cdot \mathcal{L}$, where $\tau_1 \cong \zeta \cdot (1 + \tau_0)$ has no least or greatest element. To apply Theorem 1.2, we need to show that τ_1 has a computable presentation.

It is easy to see that we can build a $\mathbf{0}''$-computable presentation $\mathcal{T}_1$ of τ_1 such that the successor relation and the block relation of $\mathcal{T}_1$ are both $\mathbf{0}''$-computable. By the result of Alaev, Thurber, Frolov in [ATF09], τ_1 has a computable copy. Again by Theorem 1.2, it follows that $\tau_1 \cdot \mathcal{L} \cong \tau \cdot \mathcal{L}$ has a computable copy, contradicting Condition (3).

This concludes the proof of "(3) $\Rightarrow$ (1)".

It remains to prove "(1) $\Rightarrow$ (2)". We will show this in Theorem 3.1 in section 3 below. □

3. The Proof of the Main Theorem

We now finish the proof of Theorem 2.1 by proving the following

Theorem 3.1. *For any $m > 1$ and any countable linear order $\mathcal{A}$, if $\mathcal{M}$ is a computable copy of $m \cdot \mathcal{A}$, then $\mathcal{A}$ is computable.*

Proof. We will prove the theorem for the case $m = 2$ and the general case can be proved in an analogous way.

Fix a computable injective enumeration $\mathcal{M}[s]$ of $\mathcal{M}$. Here, we use $\mathcal{M}[s]$ to denote the enumeration of $\mathcal{M}$ by stage s, and thus, we assume that $\mathcal{M}[0] = \emptyset$, and that at each stage, exactly one new element is enumerated into $\mathcal{M}$. We build a computable linear order $\mathcal{L}$ such that $2 \cdot \mathcal{L} \cong \mathcal{M}$. We do this by defining a computable map $g_s : \mathcal{L}[s] \mapsto \mathcal{M}[s]^{<\omega}$ with the Δ^0_2-limit g such that $\mathrm{dom}(g_s) = \mathcal{L}[s]$ and g_s is order-preserving for all s. Here order-preserving means that if $\max g_s(a) < \min g_s(b)$, then $a < b$. *We will assume that the greatest and least elements of $\mathcal{M}$ (if they exist) are in an infinite block; otherwise we simply remove the finite block(s) from $\mathcal{M}$.*

3.1. **Notation used.** We use x and y to denote points in $\mathcal{M}$, and a and b to denote points in $\mathcal{L}$. For a tuple $\vec{\boldsymbol{x}} \in \mathcal{M}^{<\omega}$, we associate the set

$$g^{-1}(\vec{\boldsymbol{x}}) = \{a \in \mathcal{L} \mid g(a) \subseteq \vec{\boldsymbol{x}}\}.$$

Each element a introduced in $\mathcal{L}$ will receive a *target* $t(a) \in \mathcal{M}$. When a is first introduced, we will define $g(a)$ and keep $t(a)$ undefined. When $R_{g(a)}$ is removed (if ever, i.e., $g(a)$ is not an interval anymore, where $R_{g(a)}$ is a requirement to be

introduced soon), we will redefine $g(a)$ and assign $t(a)$ as a target. Once assigned, $t(a)$ will not be redefined later; hence, t is a partial computable function.

We use $a < b$ to denote that $(a, b) \in \mathcal{L}$, similarly $x < y$ to denote that $(x, y) \in \mathcal{M}$. We use $i <_{\mathbb{N}} j$ to denote the ordering of the natural numbers. We define the *distance* between distinct elements x and y at stage s, denoted by $d(x, y)[s]$, as the number of points currently strictly between x and y (hence $d(x, y) = 0$ iff x and y are adjacent). If $y \notin \vec{\boldsymbol{x}}$, then $d(\vec{\boldsymbol{x}}, y) = \min_{z \in \vec{\boldsymbol{x}}} d(z, y)$ (if $y \in \vec{\boldsymbol{x}}$ then $d(\vec{\boldsymbol{x}}, y) = 0$). If $\vec{\boldsymbol{x}} \cap \vec{\boldsymbol{y}} = \emptyset$, then $d(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}) = \min_{z \in \vec{\boldsymbol{x}}, z' \in \vec{\boldsymbol{y}}} d(z, z')$; as we will only apply $d(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}})$ to tuples which are currently intervals, the condition $\vec{\boldsymbol{x}} \cap \vec{\boldsymbol{y}} = \emptyset$ implies that $\vec{\boldsymbol{x}}$ lies completely to the left or completely to the right of $\vec{\boldsymbol{y}}$. If $\vec{\boldsymbol{x}} \cap \vec{\boldsymbol{y}} \neq \emptyset$, then the distance is 0. Given tuples $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{y}}$, we say that $\vec{\boldsymbol{y}}$ is $\vec{\boldsymbol{x}}$-*near* if $d(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}) \leq \langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}$; otherwise we say that $\vec{\boldsymbol{y}}$ is $\vec{\boldsymbol{x}}$-*far*. (Here, we fix an encoding $\langle \cdot \rangle$ of all tuples with the usual properties.) Given tuples $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{y}}$, we will sometimes say informally that $\vec{\boldsymbol{x}}$ *lies to the left of* $\vec{\boldsymbol{y}}$ *if* $\max \vec{\boldsymbol{x}} < \min \vec{\boldsymbol{y}}$.

We will need to satisfy the requirements $R_{\vec{\boldsymbol{x}}}$:

$$R_{\vec{\boldsymbol{x}}}\text{: If } \vec{\boldsymbol{x}} \text{ is a maximal finite block, then } \#g^{-1}(\vec{\boldsymbol{x}}) = \frac{\#\vec{\boldsymbol{x}}}{2},$$

where $\#A$ is the size of A, and $\vec{\boldsymbol{x}}$ ranges over all finite intervals of even length. We will ensure that if $\vec{\boldsymbol{y}} \subset \vec{\boldsymbol{x}}$, then $R_{\vec{\boldsymbol{y}}}$ is of higher priority than $R_{\vec{\boldsymbol{x}}}$ (*hence the construction always attends to all the sub-tuples before attending to a larger tuple*). We will assume here that if $\vec{\boldsymbol{x}}$ is of higher priority than $\vec{\boldsymbol{y}}$, then $\langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle} < \langle \vec{\boldsymbol{y}} \rangle$. This property may imply that the coding of tuples is not a bijection.

As we are assuming that $\mathcal{M}$ receives exactly one new point at every stage, we will generate new tuples $\vec{\boldsymbol{y}}$ for the intervals containing the new point, and add $R_{\vec{\boldsymbol{y}}}$ to the end of the list of requirements. Also the requirement $R_{\vec{\boldsymbol{x}}}$ is removed from the list whenever we see that $\vec{\boldsymbol{x}}$ is no longer an interval.

A tuple $\vec{\boldsymbol{x}}$ is said to be *active* if $R_{\vec{\boldsymbol{x}}}$ has been added to the list of requirements and has not removed from it yet.

Say that $R_{\vec{\boldsymbol{x}}}$ *requires attention* at a stage s, if the following hold.

(i) $\#g^{-1}(\vec{\boldsymbol{x}}) < \frac{\#\vec{\boldsymbol{x}}}{2}$.

(ii) If $\#\vec{\boldsymbol{x}} > 2$, then there are at least $\langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}$ many points to the left and to the right of $\vec{\boldsymbol{x}}$.

(iii) If $\#\vec{\boldsymbol{x}} > 2$, then $d(\vec{\boldsymbol{y}}, \vec{\boldsymbol{x}}) > \langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}$ for every higher-priority $R_{\vec{\boldsymbol{y}}}$ with $\vec{\boldsymbol{y}} \cap \vec{\boldsymbol{x}} = \emptyset$. (In other words, every disjoint tuple with higher-priority is $\vec{\boldsymbol{x}}$-far.)

(iv) For every $b \in \mathcal{L}[s]$, either $g(b) \subsetneq \vec{\boldsymbol{x}}$, or $g(b) \cap \vec{\boldsymbol{x}} = \emptyset$ and

$$d(g(b), \vec{\boldsymbol{x}}) > \begin{cases} \langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}, & \text{if } \#\vec{\boldsymbol{x}} > 2, \\ \min\left\{ \langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}, \langle g(b) \rangle^{\langle g(b) \rangle + 2} \right\}, & \text{if } \#\vec{\boldsymbol{x}} = 2 \text{ and } \#g(b) > 2, \\ 4, & \text{if } \#\vec{\boldsymbol{x}} = \#g(b) = 2. \end{cases}$$

(v) If $\#\vec{\boldsymbol{x}} = 2$, then for each $\vec{\boldsymbol{y}}$ such that $R_{\vec{\boldsymbol{y}}}$ is of higher priority and that $\vec{\boldsymbol{x}}$ is $\vec{\boldsymbol{y}}$-near, either

- there is no a such that $g(a)$ is $\vec{\boldsymbol{y}}$-near and $g(a) \not\subseteq \vec{\boldsymbol{y}}$, or
- there exists some b such that $g(b)$ is $\vec{\boldsymbol{y}}$-near, $g(b) \not\subseteq \vec{\boldsymbol{y}}$ and $g(b)$ is of higher rank than $\vec{\boldsymbol{y}}$.

Here we say that $g(b)$ *is of higher rank than* $\vec{\boldsymbol{y}}$ if one of the following holds:

- $g(b)$ is of higher priority than $\vec{\boldsymbol{y}}$, or
- $\hat{g}(b)$ is of higher priority than $\vec{\boldsymbol{y}}$, or

- at the stage when $\hat{g}(b)$ was defined, there existed some c such that $g(c)$ is of priority higher than $\vec{y}$, $g(c) \not\subseteq \vec{y}$ and $g(c)$ is $\vec{y}$-near, or
- at the stage when $\hat{g}(b)$ was defined, there was no c such that $g(c) \not\subseteq \vec{y}$ and $g(c)$ is $\vec{y}$-near.

Here $\hat{g}(b)$ is the very first definition of g assigned to b. Note that $\hat{g}(b)$ is always defined under Phase 2, and $\hat{g}(b) = g(b)$ if and only if $t(b)$ has no definition.

(vi) If $\#\vec{x} = 2$, then there is no $\vec{y}$ such that $R_{\vec{y}}$ is of higher priority and that $\vec{x}$ is $\vec{y}$-near, and one of the following holds for $\vec{y}$:

- There exists some d such that $g(d) \not\subseteq \vec{y}$, $g(d)$ is $\vec{y}$-near, and every such $g(d)$ is of lower priority than $\vec{y}$. Furthermore, $\max \vec{x} > \max \vec{y}$ and $\max g(a_0) < \min g(a_1)$, where $g(a_0)$ and $g(a_1)$ are the first and the last points, respectively, among $\{g(b) \mid g(b) \subseteq \vec{y}\}$ to be defined.
- There exists some d such that $g(d) \not\subseteq \vec{y}$ and $g(d)$ is $\vec{y}$-near, and every such $g(d)$ is of lower priority than $\vec{y}$. Furthermore, $\min \vec{x} < \min \vec{y}$ and $\min g(a_0) > \max g(a_1)$, where $g(a_0)$ and $g(a_1)$ are the first and the last points, respectively, among $\{g(b) \mid g(b) \subseteq \vec{y}\}$ to be defined.

We remark that clause (vi) will be activated if putting $\vec{x}$ into the range of g causes $\vec{x}$ to be *not* of higher rank than some $\vec{y}$.

3.2. Construction of $\mathcal{L}$ and g_s.

Stage 0: Do nothing at this stage, and there is no requirement on the list (as $\mathcal{M}[0] = \emptyset$).

Stage s: The construction is in two phases.

Phase 1: Remove all the requirements $R_{\vec{x}}$ from the list such that $\vec{x}$ is no longer an interval.

For $a \in \mathcal{L}[s]$, if $R_{g(a)}$ is removed, we will need to redefine $g(a)$, and if $t(a)$ is defined, then we redefine $g(a)$ as an active 2-tuple with the highest priority containing $t(a)$.

For all the remaining a, we will need to find a suitable $t(a)$. Define $t(a)$ as some x (in a compatible position) such that:

(i) x is not in $g(b)$ for any $b \in \mathcal{L}[s]$.
(ii) For $b \in \mathcal{L}[s]$ with $\#g(b) = 2$, $d(x, g(b)) > 4$.
(iii) For $b \in \mathcal{L}[s]$ with $\#g(b) > 2$, $d(x, g(b)) > \langle g(b) \rangle^{\langle g(b) \rangle + 1}$.

If $t(a)$ is found, then define $g(a)$ to be the highest-priority 2-tuple containing $t(a)$ which is still active. Otherwise, keep $t(a)$ and $g(a)$ both undefined.

Remark 3.2. *In Lemma 3.6, we will prove that $t(a)$ can always be found. In fact, Lemma 3.6 will prove that we will generally have many choices for $t(a)$, and we will pick $t(a)$ to avoid $\left[\min \vec{z} - \langle \vec{z} \rangle^{\langle \vec{z} \rangle + 2}, \max \vec{z} + \langle \vec{z} \rangle^{\langle \vec{z} \rangle + 2}\right]$ for as many $\vec{z}$ as possible. The proof of Lemma 3.6 will provide details on how this is done.*

End Phase 1 once all $t(a)$ and $g(a)$ have been (re)defined (if possible).

Phase 2: Find the highest-priority requirement $R_{\vec{x}}$ which is currently active and requires attention.

We will introduce $\dfrac{\#\vec{x}}{2} - \#g^{-1}(\vec{x})$ many new points into $\mathcal{L}$. Let $a_{\min} < a_{\max}$ be the leftmost and rightmost points of $\mathcal{L}[s]$, respectively, such that $g(a) \subset \vec{x}$,

i.e., of $g^{-1}(\vec{\boldsymbol{x}})$. Let $x_{\min}$ and $x_{\max}$ be the leftmost and rightmost points of $\vec{\boldsymbol{x}}$, respectively, that is contained in some $g(a)$. (*We remark here that it is possible for* $\min g(a_{\min}) > x_{\min}$.)

- If $x_{\min} = \min \vec{\boldsymbol{x}}$ and $x_{\max} = \max \vec{\boldsymbol{x}}$, and $\max \vec{\boldsymbol{x}} <_{\mathbb{N}} \min \vec{\boldsymbol{x}}$, then we put all the new points immediately to the left of $a_{\min}$; i.e., put all the $\dfrac{\#\vec{\boldsymbol{x}}}{2} - \#g^{-1}(\vec{\boldsymbol{x}})$ many new points between the current predecessor of $a_{\min}$ and $a_{\min}$. Otherwise, if $\max \vec{\boldsymbol{x}} >_{\mathbb{N}} \min \vec{\boldsymbol{x}}$, we put all the new points immediately to the right of $a_{\max}$.
- If $x_{\min} = \min \vec{\boldsymbol{x}}$ and $x_{\max} < \max \vec{\boldsymbol{x}}$, then put all the new points immediately to the right of $a_{\max}$.
- If $x_{\min} > \min \vec{\boldsymbol{x}}$ and $x_{\max} = \max \vec{\boldsymbol{x}}$, then put all the new points immediately to the left of $a_{\min}$.
- If $x_{\min} > \min \vec{\boldsymbol{x}}$, $x_{\max} < \max \vec{\boldsymbol{x}}$, and $a_{\max} <_{\mathbb{N}} a_{\min}$, then put all the new points immediately to the right of $a_{\max}$. Otherwise, if $a_{\max} >_{\mathbb{N}} a_{\min}$, put all the new points immediately to the left of $a_{\min}$.

For all the new points c introduced at this phrase, set $g(c) = \vec{\boldsymbol{x}}$. Obviously, if one or both of $a_{\min}$ and $a_{\max}$ isn't defined, insert the new points in any compatible position.

If no such a requirement $R_{\vec{\boldsymbol{x}}}$ can be found at this phrase, then do nothing, and go to the next stage.

This ends the construction.

3.3. **Facts about the construction.**

We start with some easily verifiable facts.

Fact 3.3. *(i)* $\lim_s g_s(a)$ *exists for each* a.
(ii) If $t(a)$ *is defined then* $g(a)$ *is defined and is a* 2*-tuple containing* $t(a)$.
(iii) If $t(a)$ *and* $t(b)$ *are both defined then* $t(a) < t(b)$ *if and only if* $a < b$. *Furthermore if* $a \neq b$ *then* $d(t(a), t(b)) > 4$.
(iv) If $g(a)$ *and* $t(b)$ *are both defined and* $\max g(a) < t(b)$ *(and similarly if* $t(b) < \min g(a)$*), then* $a < b$ *and* $d(g(a), t(b)) > 3$.
(v) g *is order preserving.*
(vi) If $g(a)$ *and* $t(b)$ *are both defined and* $t(b) \in g(a)$ *then* $g(b) \subseteq g(a)$.
(vii) If $g(a) = g(b)$ *and* $a \neq b$, *then* $\#g(a) > 2$.

Proof. (i)–(iii): For each a, if $t(a)$ is defined, then the highest-priority 2-tuple containing $t(a)$ must exist, as every finite block has size > 1.

(iv): An obvious induction, applying (iii).

(v): A straightforward induction, applying (iv).

(vi): At the stage when $g(a)$ receives this definition, $t(b)$ must already be defined, otherwise if $t(b)$ is defined later then we certainly cannot have $t(b) \in g(a)$. Since $t(b)$ is already defined, $g(a)$ must receive its current definition under Phase 2 (unless $a = b$), in which case we certainly have $g(b) \subset g(a)$, and $g(b)$ cannot break up unless $g(a)$ breaks up as well.

(vii): Suppose $g(a) = g(b)$ and $a \neq b$ and $\#g(a) = 2$. If $t(a)$ and $t(b)$ are both defined, apply (iii). If $t(a)$ and $t(b)$ are both undefined, then $g(a)$ and $g(b)$ are both set under Phase 2, which is impossible unless they are both defined by the same action. But this is also impossible because a single action in Phase 2 will never introduce more points than necessary. Therefore, without loss of generality, we

have that $t(a)$ is undefined and $t(b)$ is defined. Now consider the two possibilities when $g(a)$ is defined before $t(b)$ or after $t(b)$. □

Lemma 3.4. *Suppose that $g(a)$ and $g(b)$ are both defined at the same time. Then:*

(i) $g(a)$ and $g(b)$ are either disjoint or comparable under $\subseteq$.

(ii) Suppose that $g(a) \cap g(b) = \emptyset$, and $R_{g(b)}$ has lower priority. Then,
- *If $\#g(b) > 2$ then $d(g(a), g(b)) > \langle g(b)\rangle^{\langle g(b)\rangle+2}$.*
- *If $\#g(a) > 2$ and $\#g(b) = 2$ then $d(g(a), g(b)) \geq \langle g(a)\rangle^{\langle g(a)\rangle+1}$.*
- *If $\#g(a) = \#g(b) = 2$ then:*
  - *If $t(a)$ and $t(b)$ are both undefined then $d(g(a), g(b)) > 4$.*
  - *If $t(a)$ and $t(b)$ are both defined then $d(g(a), g(b)) > 2$.*
  - *Otherwise, $d(g(a), g(b)) > 3$.*

(iii) If $g(a) \subsetneq g(b)$ then $g(a)$ is defined before $g(b)$. Furthermore $b > \max\{c : g(c) \subsetneq g(b)\}$ or $b < \min\{c : g(c) \subsetneq g(b)\}$.

Proof. (i): Suppose $g(a)$ is defined before $g(b)$. Apply Fact 3.3 (iv) and (vi) in the case when $g(b)$ is defined under Phase 1.

(ii): Suppose $\#g(b) > 2$. Then $g(b)$ is defined under Phase 2. It is easy to check that no matter which ($g(a)$ or $g(b)$) is defined first, we always have $d(g(a), g(b)) > \langle g(b)\rangle^{\langle g(b)\rangle+2}$.

Suppose $\#g(a) > 2$ and $\#g(b) = 2$. Then $g(a)$ is defined under Phase 2. All cases are obvious except for the case where $t(b)$ is defined and $g(b)$ is defined after $g(a)$. In that case, note that $t(b) \notin g(a)$, and consider the two possibilities when $t(b)$ is defined before $g(a)$, and when $t(b)$ is defined after $g(a)$.

Now assume that $\#g(a) = \#g(b) = 2$. If $t(a)$ and $t(b)$ are both undefined then $g(a)$ and $g(b)$ are both defined under Phase 2 and obviously $d(g(a), g(b)) > 4$. If $t(a)$ and $t(b)$ are both defined, then $d(t(a), t(b)) > 4$. If exactly one of $t(a)$ or $t(b)$ is defined, then observe that $d(g(a), t(b)) > 4$ or $d(g(b), t(a)) > 4$.

(iii): Suppose $g(a)$ is formed after $g(b)$, and at that time, $t(a) \downarrow\in g(b)$. Therefore $t(a)$ must be involved in some 2-tuple $\vec{z}$ which is removed at the beginning of the stage (and $g(a)$ being the new 2-tuple containing $t(a)$ replacing $\vec{z}$). Obviously $t(b)$ is undefined as $\#g(b) > 2$, and so by (i), it follows that $\vec{z} \subset g(b)$. But this is impossible because the removal of $R_{\vec{z}}$ would also cause $R_{g(b)}$ to be removed.

Now since $\#g(b) > 2$, it is obvious that $g(b)$ has to be defined under Phase 2. As $g(a)$ is already defined, the construction will put the new point b on the outside of $\{c : g(c) \subsetneq g(b)\} \neq \emptyset$. After the definition of $g(b)$, it is impossible for $\{c : g(c) \subsetneq g(b)\}$ to increase. □

Lemma 3.5. *For each active $\vec{x}$, $\#g^{-1}(\vec{x}) \leq \frac{\#\vec{x}}{2}$.*

Proof. We prove the lemma by induction on the construction. At the beginning obviously $\#g^{-1}(\vec{x}) = 0$ for every $\vec{x}$. Now consider an action in the construction which defines $g(a)$. This has no effect on $g^{-1}(\vec{x})$ unless $g(a) \subseteq \vec{x}$. Now it is easy to check that if $\vec{x} = g(a)$ then $\#g^{-1}(\vec{x}) > \frac{\#\vec{x}}{2}$ is impossible; apply Fact 3.3(vii) for the case $\#g(a) = 2$. Now consider any active $\vec{x} \supsetneq g(a)$. Fix $a_0 = a, a_1, \cdots, a_k$ such that $g(a_i) \subset \vec{x}$ are all the distinct maximal tuples contained in $\vec{x}$. (By Lemma 3.4(iii) there is no b such that $g(b) \supsetneq g(a)$ and so $g(a)$ is maximal.) By Lemma 3.4(i) these are pairwise disjoint. By the induction hypothesis, it is clear that $\#g^{-1}(\vec{x}) = \sum_{i=0}^{k} \#g^{-1}(g(a_i)) \leq \frac{\#\vec{x}}{2}$. □

Lemma 3.6. *We can always find $t(a)$ in Phase 1 of the construction.*

Proof. We are assuming that at each stage of the construction exactly one new point is introduced into $\mathcal{M}$. If this new point is to the left or to the right of the existing points of $\mathcal{M}$, then Phase 1 of the construction will not do anything and so the Lemma is trivially satisfied. Hence we may assume that exactly one 2-tuple becomes inactive at this stage. We consider the point in the construction when we have removed all requirements which are no longer active, and we have just redefined $g(a^*)$ for some a^* such that $t(a^*)$ is defined. (There can only be at most one a^*, since exactly one 2-tuple is removed from the requirements.) We are about to define $t(a)$ for all the remaining a; call this (substage) s_1. We refer to the beginning of this stage before any action is taken as (substage) s_0. Our goal is to prove that at time s_1, there are enough points in $\mathcal{M}$ in the correct position for us to define $t(a)$ for all the $a \in \mathcal{L}$ which are associated with some requirement removed at this stage. Call these elements *bad*. For each bad a, denote by $\hat{g}(a)$ the value of $g(a)$ at s_0, i.e., $\hat{g}(a) = g(a)[s_0]$; of course, $R_{\hat{g}(a)}$ is now inactive. For the rest of the proof of the lemma, when we write $g(d)$ we mean $g(d)[s_1]$ as measured at s_1.

For each maximal element $\vec{\boldsymbol{x}}$ in the range of $g[s_1]$, denote by a and b the least and greatest elements in $g^{-1}(\vec{\boldsymbol{x}})$, respectively. (Of course, $a = b$ is possible. In fact, $a = b$ iff $\#g(a) = 2$, but we will not need this here.) Since the maximal elements have to be pairwise disjoint, we will write $a_0 < b_0 < a_1 < b_1 < \cdots < a_k < b_k$ to be all the elements associated with some maximal element; $\vec{\boldsymbol{x}}_i$ is associated with a_i and b_i. Notice that $\vec{\boldsymbol{x}}_i = g(a_i)$ or $g(b_i)$. By Lemma 3.4(iii), $g(a^*)$ is maximal at s_1, so let k_0 be such that $a_{k_0} = b_{k_0} = a^*$. We write $\vec{\boldsymbol{x}}_i[s_0]$ to denote the value at s_0: Clearly $\vec{\boldsymbol{x}}_i[s_0] = \vec{\boldsymbol{x}}_i$ for every i except for $\vec{\boldsymbol{x}}_{k_0}$. We distinguish between $\vec{\boldsymbol{x}}_{k_0}$ and $\vec{\boldsymbol{x}}_{k_0}[s_0]$ because it is possible that for some bad element a, we have $g(a^*)[s_0] \subset \hat{g}(a)$, but after the new definition of $g(a^*)$, we end up with $g(a^*) \not\subset \hat{g}(a)$. Obviously every element of $\mathcal{L}[s_1]$ is either bad, or lies between a_i and b_i for some i.

Claim 3.7. *For each bad element $a \in \mathcal{L}$,*

(i) Either $a < a_0$, or $a > b_k$, or $b_i < a < a_{i+1}$ for some i.
(ii) If $a < a_0$ then $\min \hat{g}(a) \leq \min \vec{\boldsymbol{x}}_0[s_0]$. If $a > b_k$ then $\max \hat{g}(a) \geq \max \vec{\boldsymbol{x}}_k[s_0]$.
(iii) For $i = 0, \cdots k-1$, if $b_i < a < a_{i+1}$, then either $\max \vec{\boldsymbol{x}}_i[s_0] < \min \hat{g}(a) \leq \min \vec{\boldsymbol{x}}_{i+1}[s_0]$ or $\max \vec{\boldsymbol{x}}_i[s_0] \leq \max \hat{g}(a) < \min \vec{\boldsymbol{x}}_{i+1}[s_0]$.

Proof of claim. Fix a bad element a. At s_0 it must be that $\hat{g}(a)$ is comparable with or disjoint from $g(d)[s_0]$ for every d. If $\hat{g}(a)$ does not contain $\vec{\boldsymbol{x}}_i[s_0]$ for any i, then it has to lie between $\vec{\boldsymbol{x}}_i[s_0]$ and $\vec{\boldsymbol{x}}_{i+1}[s_0]$ for some i (obviously we are allowing $\pm\infty$ as one of the two tuples). Since $g[s_0]$ is order-preserving, it is easy to verify the claimed properties. Now if $\hat{g}(a)$ contains $\vec{\boldsymbol{x}}_i[s_0]$ for at least one i, the verification of the claimed properties are similar; obviously we have to consider the tuples $\vec{\boldsymbol{x}}_j[s_0]$ which are closest to $\hat{g}(a)$ but disjoint from $\hat{g}(a)$, and apply Lemma 3.4(iii). □

3.4. **Verifying that the construction works.** Putting all the trivialities together, the picture of $\mathcal{L}[s_1]$ is as follows: Every element of $\mathcal{L}[s_1]$ lies between a_i and b_i for some i if and only if it is not a bad element. Now fix i and consider all the bad elements between b_i and a_{i+1}. To ensure that g is order-preserving, clearly $t(a)$ must be picked to be some element x such that $\max \vec{\boldsymbol{x}}_i < x < \min \vec{\boldsymbol{x}}_{i+1}$. Therefore we must argue that there are enough elements x (which we will later call suitable) compared to the number of bad elements between b_i and a_{i+1}.

Towards this end, we fix i and assume that there is some bad element between b_i and a_{i+1}. We wish to show that $d(\mu_0, \mu_1)$ is sufficiently large, where $\mu_0 = \max \vec{\boldsymbol{x}}_i$ and $\mu_1 = \min \vec{\boldsymbol{x}}_{i+1}$. (We will also use $\mu_0[s_0]$ and $\mu_1[s_0]$ with the obvious meaning.) Note that $|d(\mu_0, \mu_1) - d(\mu_0, \mu_1)[s_0]| \leq 2$. Notice that for each bad element a, $t(a)$ is undefined, which means that $\hat{g}(a)$ was previously defined under Phase 2. Now call an element x between μ_0 and μ_1 *suitable for* $\vec{\boldsymbol{y}}$ if $d(x, \vec{\boldsymbol{y}}) > 4$ for $\#\vec{\boldsymbol{y}} = 2$, and $d(x, \vec{\boldsymbol{y}}) > \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 1}$ for $\#\vec{\boldsymbol{y}} > 2$. It is not hard to see that if x is suitable for both μ_0 and μ_1, then x is suitable for $g(b)$ for every $b \in \mathcal{L}[s_1]$, which means that x will be a possible choice for $t(a)$ for a bad element $a \in (b_i, a_{i+1})$.

Firstly, suppose that there is only one bad element a and furthermore that $\#\hat{g}(a) = 2$. In this case, $\hat{g}(a)$ must lie strictly between $\mu_0[s_0]$ and $\mu_1[s_0]$. Now as $\hat{g}(a)$ is the unique tuple of size 2 being made inactive, this means that both μ_0 and μ_1 are unchanged, i.e., $\mu_0 = \mu_0[s_0]$ and $\mu_1 = \mu_1[s_0]$. We only have to find a suitable image for a in the interval (b_i, a_{i+1}). Let x_0 be the new element enumerated into $\mathcal{M}$. We claim that x_0 is a suitable choice for $t(a)$: This is because $d(x_0, \mu_0)[s_1] = d(\hat{g}(a), \mu_0)[s_0] + 1$ and similarly for μ_1. Apply Lemma 3.4(ii) to see that x_0 is suitable for both μ_0 and μ_1 (again, noting that $t(a)$ is undefined).

Now suppose that there are at least two bad elements in (b_i, a_{i+1}), or that $\#\hat{g}(a) > 2$ for the unique bad element a. Fix a bad a such that $\hat{g}(a)$ has the lowest priority where $\#\hat{g}(a) > 2$; this exists since there is exactly one unique tuple of size 2 being made inactive. (Observe that in this case, if $\hat{g}(a')$ is the unique tuple of size 2 being made inactive, we must have $\hat{g}(a') \subset \hat{g}(a)$ and so $\hat{g}(a')$ is of higher priority than $\hat{g}(a)$.)

We will now argue that there are at least $\langle \hat{g}(a) \rangle^3$ many x between μ_0 and μ_1 that are suitable. There are several cases to consider (for each of the below, we apply Lemma 3.4(ii)).

- $\#\mu_0 = \#\mu_1 = 2$**:** Then no matter whether $\hat{g}(a)$ contains $\mu_0[s_0]$, or contains $\mu_1[s_0]$, or is in between, it is obvious that $d(\mu_0[s_0], \mu_1[s_0]) \geq \langle \hat{g}(a) \rangle^{\langle \hat{g}(a) \rangle + 1}$. Hence $d(\mu_0, \mu_1) \geq \langle \hat{g}(a) \rangle^{\langle \hat{g}(a) \rangle + 1} - 2$, which certainly means that there are at least $\langle \hat{g}(a) \rangle^3$ many suitable elements between μ_0 and μ_1.
- $\#\mu_0 = 2$ **and** $\#\mu_1 > 2$**:** Then $\mu_1[s_0] = \mu_1$. If μ_1 has higher priority than $\hat{g}(a)$, it is easy to see that $d(\mu_0, \mu_1) \geq \langle \hat{g}(a) \rangle^{\langle \hat{g}(a) \rangle + 1} - 2$ (no matter what the position of $\hat{g}(a)$ is relative to μ_0 and μ_1). If μ_1 has lower priority than $\hat{g}(a)$, then $\hat{g}(a)$ cannot contain μ_1 (recall our convention that sub-tuples always have higher priority), and so $d(\mu_0, \mu_1) > \langle \mu_1 \rangle^{\langle \mu_1 \rangle + 2} - 2$. In either subcase, it is obvious that there are at least $\langle \hat{g}(a) \rangle^3$ many suitable elements between μ_0 and μ_1.
- $\#\mu_0 > 2$ **and** $\#\mu_1 = 2$**:** Symmetric case.
- $\#\mu_0 > 2$ **and** $\#\mu_1 > 2$**:** Without loss of generality, assume that μ_1 has lower priority than μ_0. If $\hat{g}(a)$ has lower priority than μ_1, then obviously $d(\mu_0, \mu_1) > \langle \hat{g}(a) \rangle^{\langle \hat{g}(a) \rangle + 2} - 2 > \langle \mu_1 \rangle^{\langle \mu_1 \rangle + 2}$. On the other hand, if μ_1 has lower priority than $\hat{g}(a)$, then, of course, $d(\mu_0, \mu_1) > \langle \mu_1 \rangle^{\langle \mu_1 \rangle + 2} - 2$. In either subcase, it is obvious that there are at least $\langle \hat{g}(a) \rangle^3$ many suitable elements between μ_0 and μ_1.

So we conclude that there are at least $\langle \hat{g}(a) \rangle^3$ many suitable elements between μ_0 and μ_1. How many bad elements are there in the interval (b_i, a_{i+1})? Since $\hat{g}(a)$ has the lowest priority (with the only possible exception of the unique 2-tuple being removed), by the usual convention that $\langle \vec{\boldsymbol{z}} \rangle > \#\vec{\boldsymbol{z}}$ and by Lemma 3.5, we see that

there are at most $1 + \sum_{j \leq \langle \hat{g}(a) \rangle} \frac{j}{2} < 2\langle \hat{g}(a) \rangle^2$ many bad elements in the interval (b_i, a_{i+1}). Since the definition of $t(a)$ for all these elements has to be picked from one of the suitable elements between μ_0 and μ_1, and with a distance of at least 4 between each choice of $t(a)$, we need at least $5 \cdot 2\langle \hat{g}(a) \rangle^2 < \langle \hat{g}(a) \rangle^3$ many suitable elements, which we do have.

Finally, we consider bad elements $a < a_0$ and $a > b_k$. Then we proceed similarly to the above by considering the two cases where we have exactly one bad element with $\#\hat{g}(a) = 2$, and otherwise. This time, however, instead of using Lemma 3.4(ii), we use the fact that when $\vec{z}$ receives attention for $\#\vec{z} > 2$, we have at least $\langle \vec{z} \rangle^{\langle \vec{z} \rangle + 2}$ many points to the left and right of $\vec{z}$.

Before we conclude the proof of the lemma, we give details on how exactly the construction should pick $t(a)$, given that there are usually many more suitable elements than the number of bad elements. In the first case, when $\#\hat{g}(a) = 2$ for the unique bad element a, we set $t(a) = x_0$; recall that x_0 is suitable in this case. In the second case, we conclude that there are at least $\langle \hat{g}(a) \rangle^3$ many suitable elements, and out of the all the different combinations of choices of $t(c)$ for the bad elements c, we pick a combination so that the highest-priority $\vec{z}$ such that $\left[\min \vec{z} - \langle \vec{z} \rangle^{\langle \vec{z} \rangle + 2} - 1, \max \vec{z} + \langle \vec{z} \rangle^{\langle \vec{z} \rangle + 2} + 1\right]$ contains some $g(c)$ has the lowest possible priority. (That is, we wish to avoid as many intervals as possible.) □

Lemma 3.8. *If $\vec{x}$ is a maximal finite block of $\mathcal{M}$ then $g^{-1}(\vec{x})$ is eventually stable with $\frac{\#\vec{x}}{2}$ many elements.*

Proof. That $g^{-1}(\vec{x})$ is eventually stable follows from Lemma 3.5 and the fact that if $g(a)$ is a stable tuple of $\mathcal{M}$ then $g(a)$ is never redefined. First notice that every requirement $R_{\vec{z}}$ will only receive attention at most once (under Phase 2), after which $g^{-1}(\vec{z})$ will contain the correct number of elements. Suppose the stable $g^{-1}(\vec{x})$ has fewer than $\frac{\#\vec{x}}{2}$ many elements. We wish to argue that $R_{\vec{x}}$ must require and eventually receive attention under Phase 2.

Since $\vec{x}$ is a maximal block, $\min \vec{x}$ is a left limit point and $\max \vec{x}$ is a right limit point, and therefore the first three conditions for $R_{\vec{x}}$ to require attention must eventually hold. (Recall our assumption that the greatest and least elements of $\mathcal{M}$ are not included in $\vec{x}$.) Therefore the only obstruction for $R_{\vec{x}}$ to require attention under Phase 2 must eventually come from condition (iv). (Note that conditions (v) and (vi) are not a problem because if $\vec{y}$ is of higher priority then $\vec{x}$ and $\vec{y}$ are incomparable under $\subseteq$.)

Fix a stage s_2 large enough. First of all, observe that there cannot be any b such that $g(b) = \vec{x}$, otherwise $g^{-1}(\vec{x})$ must have the correct number of elements. Next, we also assume that at every stage (just before the start of Phase 2) after s_2, there is some d such that $g(d)$ is $\vec{x}$-near and $g(d) \not\subseteq \vec{x}$; otherwise condition (iv) of Phase 2 will apply and $R_{\vec{x}}$ will receive attention.

We argue by proving the following sequence of statements.

(i) *After s_2 there cannot be a new definition of a $g(c)$, where $g(c)$ is $\vec{x}$-near, $g(c) \cap \vec{x} = \emptyset$ and $\#g(c) > 2$.*

As $g(c)$ is defined under Phase 2 and is of lower priority than $\vec{x}$ (since s_2 is large enough), clause (ii) of Phase 2 will prevent $R_{g(c)}$ from requiring attention.

By the fact that $\vec{x}$ is a maximal block, it is easy to see that we can assume that s_2 is large enough so that at every stage after s_2, and for every c, if $g(c)$ is $\vec{x}$-near, then either $g(c) \subsetneq \vec{x}$ or $g(c)$ is not of higher rank than $\vec{x}$.

(ii) *After s_2 there cannot be a new definition of a $g(c)$ under Phase 2, where $\#g(c) = 2$ and $g(c)$ is $\vec{x}$-near (this, of course, includes the case where $g(c) \cap \vec{x} \neq \emptyset$).*

Consider the first time after s_2 where some $g(c) = \vec{y}$ is defined as in the statement above. As s_2 is large enough we assume that $\vec{y} \not\subseteq \vec{x}$. Under Phase 2 when $g(c)$ is defined, clause (v) ensures that there is no $d \neq c$ such that $g(d)$ is $\vec{x}$-near and $g(d) \not\subseteq \vec{x}$. But this means that the construction at Phase 2 would have attended to $R_{\vec{x}}$ instead of $R_{\vec{y}}$.

By (i) and (ii) and the fact that $\vec{x}$ is a maximal block, we can fix $s_3 > s_2$ large enough so that at every stage after s_3, and for every c, if $g(c)$ is $\vec{x}$-near then either $\#g(c) = 2$ and $t(c)$ is defined, or $\#g(c) > 2$ and $g(c) \cap \vec{x} \neq \emptyset$, or $g(c) \subsetneq \vec{x}$.

(iii) *After s_3 there cannot be a new definition of $t(c)$ under Phase 1 if $d(t(c), \vec{x}) \leq \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} + 1$.*

Again consider the first time after s_3 when some $t(c)$ too close to $\vec{x}$ receives definition. We follow the notation and analysis in the proof of Lemma 3.6. We argue that at $s_1 > s_3$, it is impossible to assign any $t(c)$ such that $d(t(c), \vec{x}) \leq \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} + 1$. At s_1, if there is a unique bad element a and $\#\hat{g}(a) = 2$ then, of course, $d(\hat{g}(a), \vec{x}) > \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2}$. In this case, $t(a)$ is taken to be x_0, which means that $d(t(a), \vec{x}) > \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} + 1$.

On the other hand, if at s_1 we are able to fix a bad a such that $\hat{g}(a)$ has lowest priority with $\#\hat{g}(a) > 2$, then $\hat{g}(a)$ is of lower priority than $\vec{x}$ (as we assume that s_2 is large enough). The proof of Lemma 3.6 tells us that there are at least $\langle \hat{g}(a) \rangle^3$ many suitable elements, and at most $2\langle \hat{g}(a) \rangle^2$ many bad elements competing for these spots. If additionally we wish to avoid the intervals $\left[\min \vec{z} - \langle \vec{z} \rangle^{\langle \vec{z} \rangle + 2} - 1, \max \vec{z} + \langle \vec{z} \rangle^{\langle \vec{z} \rangle + 2} + 1\right]$ for every $\vec{z}$ of priority no lower than that of $\vec{x}$, then this eliminates at most $\sum_{q \leq \langle \vec{x} \rangle} 2q^{q+2} + q + 2 < \sum_{q \leq \langle \vec{x} \rangle} \langle \vec{x} \rangle^{q+3} < \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 4} < \langle \hat{g}(a) \rangle^2$ many suitable elements. Since we need $5 \cdot 2\langle \hat{g}(a) \rangle^2$ many suitable elements, we can (and will) pick an assignment of t so that we do not assign any suitable element in the interval $\left[\min \vec{x} - \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} - 1, \max \vec{x} + \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} + 1\right]$ to a new definition of t.

Now, in view of (iii), we assume $s_4 > s_3$ is large enough so that at every stage after s_4, and for every c, if $g(c)$ is $\vec{x}$-near then either $g(c) \subsetneq \vec{x}$, or $\#g(c) > 2$ and $g(c) \cap \vec{x} \neq \emptyset$.

(iv) *After s_4, if $g(c)$ is defined and $g(c)$ is $\vec{x}$-near then $g(c) \supset g(d)$ for some $g(d)$ defined before s_4, and where $g(d)$ is $\vec{x}$-near and $g(d) \not\subseteq \vec{x}$.*

Consider the definition of some $g(c) = \vec{y}$ after s_4, where $\vec{y}$ is $\vec{x}$-near. Then $\#\vec{y} > 2$ and so $g(c)$ is defined under Phase 2, and of lower priority than $\vec{x}$. Suppose $\vec{y}$ does not contain $g(d)$ for any d such that $g(d)$ is $\vec{x}$-near and $g(d) \not\subseteq \vec{x}$. Since $d(\vec{y}, g(d)) > \langle \vec{y} \rangle^{\langle \vec{y} \rangle + 2}$ for every $g(d)$ disjoint from $\vec{y}$, it then follows that before the definition of $g(c) = \vec{y}$, we must have that $g(d)$ is $\vec{x}$-far for every d such that $g(d) \not\subseteq \vec{x}$, which is a contradiction since $R_{\vec{x}}$ would be attended to instead of the lower-priority $R_{\vec{y}}$. So $\vec{y}$ must (properly) contain $g(d)$ for some d such that $g(d)$ is $\vec{x}$-near and $g(d) \not\subseteq \vec{x}$. By induction, $g(d)$ is either defined before s_4, or contains some $g(d')$ defined before s_4.

We are now ready to finish the proof of the lemma. At s_4, for every c such that $g(c)$ is $\vec{x}$-near, we have $g(c) \cap \vec{x} \neq \emptyset$. Now consider a stage $s_5 > s_4$ when every one of these $g(c)$ is inactive (except, of course, those strictly contained in $\vec{x}$). By (iv),

it must be the case that at s_5, $g(d)$ is $\vec{\boldsymbol{x}}$-far whenever $g(d) \not\subseteq \vec{\boldsymbol{x}}$. Since Phase 1 of stage s_5 also cannot define an $\vec{\boldsymbol{x}}$-near $g(d)$, we get a contradiction to our original assumption. Therefore at Phase 2 of s_5, we must attend to $R_{\vec{\boldsymbol{x}}}$. □

Lemma 3.9. *Suppose $A \subset \mathcal{L}$ is an infinite set, and $c \in \mathcal{L}$.*

- *If for every $a, b \in A$ we have $\min g(a) = \min g(b)$ (for the final values of g), and $g(c) \subseteq g(a)$, then $c < b$ for some $b \in A$.*
- *If for every $a, b \in A$ we have $\max g(a) = \max g(b)$ (for the final values of g), and $g(c) \subseteq g(a)$, then $c > b$ for some $b \in A$.*

Proof. We prove the first statement; the second statement follows by a symmetric argument. Suppose $\min g(a) = x$ for every $a \in A$. First pick some $a \in A$ such that $g(a)$ is formed after $g(c)$, and some $b \in A$ such that $g(b)$ is formed after $g(a)$. By Lemma 3.4(iii) we have $g(b) \supsetneq g(a)$ and so $g(b)$ is defined under Phase 2. As A is infinite, we may assume that $\max g(b) >_{\mathbb{N}} x$; if it is not, then we simply consider some $b' \in A$ so that $g(b')$ is formed after $g(b)$. However, by examining the construction, we see that we will put the new point b to the right of c. □

Lemma 3.10. *If B is an infinite block of $\mathcal{M}$ of order type ω or ω^*, then $g^{-1}(B) = \{a \in \mathcal{L} \mid g(a) \subset B\}$ has the same order type as B.*

Proof. Suppose that B has order type ω. We have to prove that $g^{-1}(B)$ is infinite, and that every element of $g^{-1}(B)$ has finitely many elements of $g^{-1}(B)$ on its left. First we assume that $g^{-1}(B)$ is finite.

Fix a block $\vec{\boldsymbol{x}} \subset B$ satisfying the following conditions:

- $\#\vec{\boldsymbol{x}} = 2$.
- For every $a \in g^{-1}(B)$, we want $d(\vec{\boldsymbol{x}}, g(a)) > \langle g(a) \rangle^{\langle g(a) \rangle + 2}$, and that $\vec{\boldsymbol{x}}$ is of lower priority than $g(a)$.
- For every $\vec{\boldsymbol{y}} \subset B$ such that $\vec{\boldsymbol{y}}$ is of higher priority than $g(a)$ for some $a \in g^{-1}(B)$, we want $d(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}) > \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 2}$.
- $d(\vec{\boldsymbol{x}}, \min B) > 4$.

Notice that the argument below will apply even if $g^{-1}(B) = \emptyset$, or if $\min B$ is the least element of $\mathcal{M}$; the reader should keep these possibilities in mind. Also, we remark here that most of the work done below is necessary because it is possible that $\min \vec{\boldsymbol{x}} - \langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2} < \min B$ no matter how we pick $\vec{\boldsymbol{x}}$, and therefore we cannot simply proceed by assuming that the interval $\left[\min \vec{\boldsymbol{x}} - \langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}, \max \vec{\boldsymbol{x}} + \langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}\right]$ has no further activity.

Now we fix $\vec{\boldsymbol{x}}_0 = [\min B, \max \vec{\boldsymbol{x}}]$. We fix a stage s_6 large enough such that at every stage after s_6:

- For every b such that $g(b)$ is $\vec{\boldsymbol{x}}$-near, if $g(b)$ is of higher priority than $\vec{\boldsymbol{x}}$, then $g(b) \subset B$ (and hence $d(g(b), \vec{\boldsymbol{x}}) > \langle g(b) \rangle^{\langle g(b) \rangle + 2}$).
- The interval $\left[\min B, \max \vec{\boldsymbol{x}}_0 + \langle \vec{\boldsymbol{x}}_0 \rangle^{\langle \vec{\boldsymbol{x}}_0 \rangle + 2}\right]$ is stable, i.e., does not receive new points.
- No requirement of priority higher than $R_{\vec{\boldsymbol{x}}_0}$ ever acts again.

We now assume, towards a contradiction, that at every stage (just before Phase 2) after s_6, there is always some $g(b)$ such that $g(b)$ is $\vec{\boldsymbol{x}}_0$-near and $g(b) \not\subseteq \vec{\boldsymbol{x}}_0$.

It is straightforward to see (following (i), (ii) and (iii) in the proof of Lemma 3.8, we omit repeating the details) that we can fix $s_7 > s_6$ such that at every stage after s_7, if $g(c)$ is $\vec{\boldsymbol{x}}_0$-near, then either $g(c) \subsetneq \vec{\boldsymbol{x}}_0$, or $\#g(c) > 2$ and $g(c) \cap \vec{\boldsymbol{x}}_0 \neq \emptyset$.

Now following (iv) in the proof of Lemma 3.8, it is easy to see that if a new definition of $g(c)$ is made after s_7 when $g(c)$ is $\vec{\boldsymbol{x}}_0$-near, then $\#g(c) > 2$ and $g(c) \supset g(d)$ for some $g(d)$ defined before s_7 and $g(d)$ is $\vec{\boldsymbol{x}}_0$-near and $g(d) \not\subseteq \vec{\boldsymbol{x}}_0$. However, each $g(d)$ of this kind (by choice of s_7) must satisfy $\#g(d) > 2$ and $g(d) \cap \vec{\boldsymbol{x}}_0 \neq \emptyset$. Now as $g(d) \not\subseteq \vec{\boldsymbol{x}}_0$, we also have $g(d) \not\subset B$, but as $g(d) \cap B \neq \emptyset$, it follows that $g(d)$ must be eventually made inactive. When all such $g(d)$ defined before s_7 have been made inactive, say at the least $s_8 > s_7$, we see that there cannot be any $g(b)$ such that $g(b)$ is $\vec{\boldsymbol{x}}_0$-near and $g(b) \not\subseteq \vec{\boldsymbol{x}}_0$, a contradiction to our original assumption (note that Phase 1 of s_8 cannot introduce any $\vec{\boldsymbol{x}}_0$-near $g(b)$).

We would now like to proceed as in the proof of Lemma 3.8 and argue that $R_{\vec{\boldsymbol{x}}_0}$ must now receive attention in Phase 2 of stage s_8. Unfortunately, $R_{\vec{\boldsymbol{x}}_0}$ might not be able to do so because condition (ii) might not hold; this depends on how the elements of $\mathcal{M}$ are enumerated and there does not appear to be an obvious way to use $R_{\vec{\boldsymbol{x}}_0}$. Therefore, we are forced to turn to $R_{\vec{\boldsymbol{x}}}$ at stage s_8.

We now argue that at stage s_8, $R_{\vec{\boldsymbol{x}}}$ will receive attention under Phase 2. First of all, $R_{\vec{\boldsymbol{x}}}$ will be chosen to receive attention if all conditions are met, as all higher-priority requirements have stopped acting. Let's examine condition (iv) and fix some $g(b) \not\subseteq \vec{\boldsymbol{x}}$. By the choice of $\vec{\boldsymbol{x}}$ we cannot have $d(g(b), \vec{\boldsymbol{x}}) \leq 4$ so (iv) is obviously true if $\#g(b) = 2$. Suppose $\#g(b) > 2$, then we have to show that $d(g(b), \vec{\boldsymbol{x}}) > \min\left\{\langle \vec{\boldsymbol{x}} \rangle^{\langle \vec{\boldsymbol{x}} \rangle + 2}, \langle g(b) \rangle^{\langle g(b) \rangle + 2}\right\}$. This is true if $g(b)$ is of higher priority than $\vec{\boldsymbol{x}}$ by the choice of stage s_6, and if $g(b)$ is of lower priority, use the properties of stage s_8 (note that if $g(b) \subset B$ then $g(b)$ is of higher priority than $\vec{\boldsymbol{x}}$). Hence condition (iv) for $R_{\vec{\boldsymbol{x}}}$ is met at s_8.

We turn to conditions (v) and (vi). At stage s_8, consider a higher-priority $\vec{\boldsymbol{y}}$ such that $\vec{\boldsymbol{x}}$ is $\vec{\boldsymbol{y}}$-near. This means that $I_{\vec{\boldsymbol{y}}} \subseteq I_{\vec{\boldsymbol{x}}} \subseteq I_{\vec{\boldsymbol{x}}_0}$, where we set $I_{\vec{\boldsymbol{z}}} = \left[\min \vec{\boldsymbol{z}} - \langle \vec{\boldsymbol{z}} \rangle^{\langle \vec{\boldsymbol{z}} \rangle + 2}, \max \vec{\boldsymbol{z}} + \langle \vec{\boldsymbol{z}} \rangle^{\langle \vec{\boldsymbol{z}} \rangle + 2}\right]$. By choosing s_8 large enough, we can assume that $\vec{\boldsymbol{y}} \subset B$. By the choice of $\vec{\boldsymbol{x}}$, we have that $\vec{\boldsymbol{y}}$ is of lower priority than $g(a)$ for every $a \in g^{-1}(B)$. By the properties of s_8, it follows that for any b, if $g(b)$ is $\vec{\boldsymbol{y}}$-near then it is also $\vec{\boldsymbol{x}}_0$-near and hence $b \in g^{-1}(B)$. By considering the two scenarios where there is some b such that $g(b) \not\subseteq \vec{\boldsymbol{y}}$ and $g(b)$ is $\vec{\boldsymbol{y}}$-near, and otherwise, it follows that (v) and (vi) hold for each such $\vec{\boldsymbol{y}}$ (noting that $\vec{\boldsymbol{y}}$ is of lower priority than $g(a)$ for every $a \in g^{-1}(B)$). Hence conditions (v) and (vi) for $R_{\vec{\boldsymbol{x}}}$ are met at s_8, and hence $R_{\vec{\boldsymbol{x}}}$ must receive attention at s_8. This shows that $g^{-1}(B)$ is infinite.

Now we wish to see that every element of $g^{-1}(B)$ has finitely many elements from $g^{-1}(B)$ to its left. We fix $a \in g^{-1}(B)$ and assume that a has infinitely many elements $a_0, a_1, \cdots \in g^{-1}(B)$ to the left of a. Since g is order preserving, it must be that $\min g(a_i) < \max g(a)$ for every i, and hence $g(a)$ and $g(a_i)$ have to be comparable. By Lemma 3.5, for all but finitely many i we must have $g(a_i) \supseteq g(a)$. Therefore, for an infinite subset $X \subseteq g^{-1}(B)$ we must have $\min g(a_i) = \min g(a_j)$ for every $a_i, a_j \in X$. Apply Lemma 3.9 for a contradiction. Thus $g^{-1}(B)$ has order type ω.

If B has order type ω^* then a symmetric argument applies. □

Lemma 3.11. *If B is an infinite block of $\mathcal{M}$ of order type $\omega^* + \omega$, then $g^{-1}(B) = \{a \in \mathcal{L} \mid g(a) \subset B\}$ has also order type $\omega^* + \omega$.*

Proof. We now assume that B has order type $\omega^* + \omega$. We first show that $g^{-1}(B)$ is infinite. Suppose the contrary. Fix $\vec{\boldsymbol{x}} \subset B$ such that $\#\vec{\boldsymbol{x}} = 2$ and for every $\vec{\boldsymbol{y}} \subset B$ such that $\vec{\boldsymbol{y}}$ is of higher or equal priority than $g(a)$ for some $a \in g^{-1}(B)$, we have

$d(\vec{x}, \vec{y}) > \langle \vec{y} \rangle^{\langle \vec{y} \rangle + 2}$. Then it is straightforward to check that $R_{\vec{x}}$ must eventually receive attention under Phase 2 of the construction (follow the argument at stage s_8 in the proof of Lemma 3.10). Thus we assume that $g^{-1}(B)$ is infinite.

Fix $x \in B$. We next wish to show that there is some $a \in g^{-1}(B)$ such that $\max g(a) > x$. Suppose no such a exists. We first argue that for every $a \in g^{-1}(B)$, there is some $b \in g^{-1}(B)$ such that $g(a) \subseteq g(b)$ and $g(b)$ is maximal. If $g(a)$ is not included in some maximal $g(b)$, then we can fix an infinite sequence $b_0, b_1, \cdots$ such that $g(a) \subsetneq g(b_0) \subsetneq g(b_1) \subsetneq \cdots \subsetneq B$. As $\max g(b_i) \leq x$ we may assume that $\max g(b_i) = \max g(b_j)$ for every i, j. Consider $\vec{x}$ where $\#\vec{x} = 2$ consisting of the next two elements of B after $\max g(b_0)$; hence $d(g(b_i), \vec{x}) = 0$ but $g(b_i) \cap \vec{x} = \emptyset$. Since $\#g(b_i) > 2$, we have that $g(b_i)$ is defined under Phase 2, so we see that this is impossible if $g(b_i)$ is of lower priority than $\vec{x}$. (Note that this phenomenon is impossible unless $\max g(b_0)$ or its successor is a right limit point.)

Hence, for every $a \in g^{-1}(B)$, there is some $b \in g^{-1}(B)$ such that $g(a) \subseteq g(b)$ and $g(b)$ is maximal. Fix $a_0 \in g^{-1}(B)$ such that $g(a_0)$ is maximal and contains the first $g(d) \subset B$ to be defined. Since $g^{-1}(B)$ is infinite, fix $a_1 \in g^{-1}(B)$ such that $g(a_1)$ is maximal and lies to the left of $g(a_0)$, and which is formed after $g(b) \subset B$ for every $g(b)$ to the right of $g(a_0)$, as well as $g(a_0)$ itself. Denote $C_0 = \big\{b \in g^{-1}(B) \mid g(b) \subseteq g(a_1)$ or $g(b)$ is to the right of $g(a_1)\big\}$. Also denote $C = \big\{b \in g^{-1}(B) \mid g(b) \subseteq g(a_2)$ or $g(b)$ is to the right of $g(a_2)\big\}$ for some $a_2 \in g^{-1}(B)$ where $g(a_2)$ is maximal and lies to the left of $g(a_1)$, and with the property that if any $z \in B$ is enumerated into $\mathcal{M}$ before some $g(c)$ with $c \in C_0$ is defined, then $z > \max g(a_2)$. Note that $C_0 \subset C$ are both finite.

Now fix $\vec{x} \subset B$ with the following properties:

- $\#\vec{x} = 2$ and $\min \vec{x} > x$.
- $\vec{x}$ is of lower priority than $g(c)$ for every $c \in C$.
- For every $\vec{y} \subset B$ such that $\vec{y}$ is of equal or higher priority than $g(c)$ for some $c \in C$, we have $d(\vec{x}, \vec{y}) > \langle \vec{y} \rangle^{\langle \vec{y} \rangle + 2}$.

Now we fix a stage s_0 large enough such that at every stage after s_0:

- For every b such that $g(b)$ is $\vec{x}$-near, if $g(b)$ is of higher priority than $\vec{x}$, then $g(b) \subset B$.
- The interval
$$\left[\min_{c \in C} g(c), \max \vec{x} + \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} + 1\right] \cup$$
$$\left[\min \vec{x} - \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} - 1, \max \vec{x} + \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} + 1\right] \subset B$$
is stable, i.e., does not receive new points, and no more g-definitions in them.
- No requirement of priority higher than $R_{\vec{x}}$ ever acts again.

We now assume for a contradiction that at every stage (just before Phase 2) after s_0, there is always some $g(b)$ such that $g(b)$ is $\vec{x}$-near and $g(b) \not\subset B$. After s_0 suppose we define $g(c)$ where $g(c)$ is $\vec{x}$-near. Then $\#g(c) > 2$ since the interval $\left[\min \vec{x} - \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} - 1, \max \vec{x} + \langle \vec{x} \rangle^{\langle \vec{x} \rangle + 2} + 1\right]$ is already stable. Thus $g(c)$ is defined under Phase 2, and hence $g(c) \supset g(d)$ for some $g(d)$ defined before s_0 and where $g(d)$ is $\vec{x}$-near and $g(d) \not\subset B$.

However, note that if $g(d)$ is defined before s_0 and where $g(d)$ is $\vec{x}$-near and $g(d) \not\subset B$, then $g(d)$ contains some element of the interval $I_{\vec{x}} \subset B$ which is stable

at s_0, hence $g(d)$ must be made inactive after s_0. At the least stage $s_1 > s_0$ when every such $g(d)$ is made inactive, there cannot be any $g(c)$ which is $\vec{\boldsymbol{x}}$-near and $g(c) \not\subset B$. Since Phase 1 of stage s_1 only introduces $g(c)$ where $\#g(c) = 2$, it cannot introduce a new $g(c)$ which is $\vec{\boldsymbol{x}}$-near. Hence we get a contradiction to our assumption.

We claim that at stage s_1, $R_{\vec{\boldsymbol{x}}}$ must receive attention under Phase 2. Recall that s_1 at Phase 2 has the property that for every c, if $g(c)$ is $\vec{\boldsymbol{x}}$-near then $g(c) \subset B$.

First consider condition (iv), and fix $g(b) \not\subseteq \vec{\boldsymbol{x}}$. If $g(b)$ is $\vec{\boldsymbol{x}}$-far then we are done, so assume that $g(b)$ is $\vec{\boldsymbol{x}}$-near. By the properties of s_1, we have $g(b) \subset B$. Now if $\#g(b) = 2$ then we are also done since $d(g(b), \vec{\boldsymbol{x}}) > 4$. Thus assume that $\#g(b) > 2$. Let $a_3 \in g^{-1}(B)$ be such that $g(a_3)$ is maximal and the rightmost such, that is, $g(b) \subseteq g(a_3)$ or $g(b)$ lies to the left of $g(a_3)$. Since $a_3 \in C$ we have that $d(\vec{\boldsymbol{x}}, g(a_3)) > \langle g(a_3)\rangle^{\langle g(a_3)\rangle+2}$, and so we are also fine if $g(b) \subseteq g(a_3)$. Suppose $g(b)$ is to the left of $g(a_3)$. If $g(b)$ is of lower priority than $g(a_3)$ then, by Lemma 3.4, $d(g(b), \vec{\boldsymbol{x}}) > d(g(b), g(a_3)) > \langle g(b)\rangle^{\langle g(b)\rangle+2}$. On the other hand, if $g(b)$ is of higher priority than $g(a_3)$, then $d(g(b), \vec{\boldsymbol{x}}) > d(g(a_3), \vec{\boldsymbol{x}}) > \langle g(a_3)\rangle^{\langle g(a_3)\rangle+2} > \langle g(b)\rangle^{\langle g(b)\rangle+2}$. Hence condition (iv) for $R_{\vec{\boldsymbol{x}}}$ is satisfied at s_3.

We now turn to conditions (v) and (vi) for $R_{\vec{\boldsymbol{x}}}$. We fix $\vec{\boldsymbol{y}}$ such that $R_{\vec{\boldsymbol{y}}}$ is of higher priority than $R_{\vec{\boldsymbol{x}}}$ and such that $\vec{\boldsymbol{x}}$ is $\vec{\boldsymbol{y}}$-near. Note that at s_1, we have $I_{\vec{\boldsymbol{y}}} \subset I_{\vec{\boldsymbol{x}}} \subset B$. We claim that one of the following holds for $\vec{\boldsymbol{y}}$:

- There is no a such that $g(a)$ is $\vec{\boldsymbol{y}}$-near and $g(a) \not\subseteq \vec{\boldsymbol{y}}$, or
- such a exists where $a \in C$, or
- $g(a) \subseteq \vec{\boldsymbol{y}}$ for every $a \in C$.

This is certainly true if $g(a)$ is $\vec{\boldsymbol{y}}$-near for every $a \in C$ (via the second or third option). Suppose $g(a) \cap I_{\vec{\boldsymbol{y}}} = \emptyset$ for some $a \in C$. Since $\max I_{\vec{\boldsymbol{y}}} > x$, this means that $I_{\vec{\boldsymbol{y}}}$ lies to the right of $g(a)$. Now for any $g(b)$, if $g(b)$ is $\vec{\boldsymbol{y}}$-near then $g(b) \subset B$ (by s_1) and so $b \in C$ (since C is "right-closed"). This implies that the first or second option holds.

Now, in particular, we examine condition (v) for $R_{\vec{\boldsymbol{x}}}$ and $\vec{\boldsymbol{y}}$. By the properties of $\vec{\boldsymbol{x}}$, we see that $\vec{\boldsymbol{y}}$ is of lower priority than $g(c)$ for every $c \in C$. Hence (v) holds if the either of the first two options for $\vec{\boldsymbol{y}}$ holds. Therefore, let us assume that $g(a) \subseteq \vec{\boldsymbol{y}}$ for every $a \in C$, and assume that (v) fails for this $\vec{\boldsymbol{y}}$. In particular, fix b_0 such that $g(b_0)$ is $\vec{\boldsymbol{y}}$-near and $g(b_0) \not\subseteq \vec{\boldsymbol{y}}$ and $g(b_0)$ is not of higher rank than $\vec{\boldsymbol{y}}$. There are now several cases to consider:

$\hat{g}(b_0) = g(b_0)$ **and** $\#\hat{g}(b_0) > 2$**:** At stage s_1, as $g(b_0)$ is $\vec{\boldsymbol{y}}$-near, it is also $\vec{\boldsymbol{x}}$-near and hence $g(b_0) \subset B$. As $g(b_0) \not\subseteq \vec{\boldsymbol{y}}$, but $g(a_0) \subseteq \vec{\boldsymbol{y}}$, we have that $g(b_0) \cap g(a_0) = \emptyset$ (as $g(a_0)$ is maximal). When $g(b_0)$ earlier receives its definition under Phase 2, $g(a_0')$ must already exist for some $g(a_0') \subseteq g(a_0)$. However, at that time, condition (iv) demands that $d(g(a_0'), g(b_0)) > \langle g(b_0)\rangle^{\langle g(b_0)\rangle+2}$, which is impossible as $g(a_0')$ and $g(b_0)$ are both $\vec{\boldsymbol{y}}$-near and $\vec{\boldsymbol{y}}$ is of higher priority than $g(b_0)$ (by the "not of higher rank" assumption).

$\hat{g}(b_0) = g(b_0)$ **and** $\#\hat{g}(b_0) = 2$**:** At the earlier stage, say t_0 when $g(b_0)$ receives definition under Phase 2, there must be some d such that $g(d) \not\subseteq \vec{\boldsymbol{y}}$ and $g(d)$ is $\vec{\boldsymbol{y}}$-near, and every such $g(d)$ is of lower priority than $\vec{\boldsymbol{y}}$ (this follows by the "not of higher rank" assumption). As $\vec{\boldsymbol{y}} \supseteq g(a_3)$ and $g(b_0) \subset B$ and $g(b_0) \not\subseteq \vec{\boldsymbol{y}}$, it follows that $\min g(b_0) < \min \vec{\boldsymbol{y}}$. Since $g(b_0)$ lies to the left of $g(a_2)$, this means that at stage t_0, $g(d)$ must already be defined for every $d \in C_0$. But this means that when evaluating condition (vi) at stage t_0,

the first $g(d_0) \subseteq \vec{\boldsymbol{y}}$ to be defined must lie to the right of the last $g(d_1) \subseteq \vec{\boldsymbol{y}}$ to be defined. So condition (vi) must fail via $\vec{\boldsymbol{y}}$, and so $g(b_0)$ cannot receive definition under Phase 2 at t_0 after all.

$\hat{g}(b_0) \neq g(b_0)$ **and** $\#\hat{g}(b_0) > 2$**:** Then $t(b_0)[s_1]$ is defined. Let $t_1 < s_1$ be the stage when $t(b_0)$ receives definition; thus at the beginning of t_1, $\hat{g}(b_0)$ is made inactive and $t(b_0)$ is found during Phase 1 of t_1. Looking back at the proof of Lemma 3.6, since b_0 is a bad element at stage t_1 and $\#\hat{g}(b_0) > 2$, we must be in the second case where there are at least $\langle \hat{g}(b_0) \rangle^3$ many suitable elements in a compatible position for $t(b_0)$. Since $\hat{g}(b_0)$ is of lower priority than $\vec{\boldsymbol{y}}$ (again by the "not of higher rank" assumption), by the same computations in Lemma 3.8(iii), the action of Phase 1 at t_1 will not assign $t(b_0)$ inside the interval $\left[\min \vec{\boldsymbol{y}} - \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 2} - 1, \max \vec{\boldsymbol{y}} + \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 2} + 1\right]$ (notice also that $\vec{\boldsymbol{y}}$ is of higher priority than $\hat{g}(b_0)$ and so $R_{\vec{\boldsymbol{y}}}$ is certainly active at t_1). Thus $g(b_0)[s_1]$ (at a stage after t_1) cannot possibly be $\vec{\boldsymbol{y}}$-near, a contradiction.

$\hat{g}(b_0) \neq g(b_0)$ **and** $\#\hat{g}(b_0) = 2$**:** We also have $t(b_0)[s_1]$ is defined. Let t_2 be the stage when $t(b_0)$ receives definition. At t_2 if we are in the second case and we find at least $\langle \hat{g}(b_0) \rangle^3$ many suitable elements in a compatible position for $t(b_0)$, then $t(b_0)$ will not be assigned to something inside the interval $\left[\min \vec{\boldsymbol{y}} - \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 2} - 1, \max \vec{\boldsymbol{y}} + \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 2} + 1\right]$, as in the previous case. So let's assume that when $t(b_0)$ is defined at t_2, we are in the first case, where b_0 is the unique bad element in the corresponding interval (again refer to Lemma 3.6). In this case we pick $t(b_0)$ to be the new element enumerated into $\mathcal{M}[t_2]$. More specifically, if $\hat{g}(b_0) = (z_0, z_1)$ with $z_0 < z_1$, then $t(b_0)$ is the (new) element such that $z_0 < t(b_0) < z_1$.

Clearly, at stage s_1 we must have

$$t(b_0) \in \left[\min \vec{\boldsymbol{y}} - \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 2} - 1, \max \vec{\boldsymbol{y}} + \langle \vec{\boldsymbol{y}} \rangle^{\langle \vec{\boldsymbol{y}} \rangle + 2} + 1\right],$$

as $g(b_0)[s_1]$ is $\vec{\boldsymbol{y}}$-near, hence $t(b_0) \in B$. In fact, as $g(b_0)[s_1]$ is $\vec{\boldsymbol{y}}$-near, hence $g(b_0)[s_1] \subset B$ and recall also that $g(b_0)[s_1] \not\subseteq \vec{\boldsymbol{y}}$. Thus $g(b_0)[s_0]$ lies to the left of $g(a_2)$, which means that $t(b_0) \leq \max \vec{\boldsymbol{y}}$ (this inequality will suffice for our purpose, but in fact, $t(b_0)$ must be much closer to $\min \vec{\boldsymbol{y}}$). This is also true when measured at t_2 and so we conclude that $z_1 \leq \max \vec{\boldsymbol{y}}$. But now we have $t(b_0) < z_1 \leq \max \vec{\boldsymbol{y}}$ where $t(b_0), \max \vec{\boldsymbol{y}} \in B$. Hence $z_1 \in B$.

Now let $t_3 < t_2$ be the stage when $\hat{g}(b_0)$ receives definition (under Phase 2). We wish to follow the argument in the previous case (where $\hat{g}(b_0) = g(b_0)$ and $\#\hat{g}(b_0) = 2$) to obtain a contradiction, by examining the construction in Phase 2 at stage t_3. As before we certainly have some d such that $g(d) \not\subseteq \vec{\boldsymbol{y}}$ and $g(d)$ is $\vec{\boldsymbol{y}}$-near, and every such $g(d)$ is of lower priority than $\vec{\boldsymbol{y}}$ (this only uses the "not of higher rank" assumption). Since $\hat{g}(b_0)$ is of lower priority than $\vec{\boldsymbol{y}}$, we cannot have $\hat{g}(b_0) \subseteq \vec{\boldsymbol{y}}$, so they have to be incomparable. Since $z_1 \leq \max \vec{\boldsymbol{y}}$ we must still have $\min \hat{g}(b_0) = z_0 < \min \vec{\boldsymbol{y}}$, which means that $z_1 \leq \min \vec{\boldsymbol{y}}$. Thus $z_1 \not> \max g(a_2)$ which means that at t_3, $g(d)$ must be already be defined for every $d \in C_0$. Thus as above, condition (vi) must fail via $\vec{\boldsymbol{y}}$ at stage t_3, which means that $\hat{g}(b_0)$ cannot receive definition at t_3 after all.

This contradiction shows that b_0 cannot exist and hence (v) holds for this $\vec{\boldsymbol{y}}$.

Now we turn to condition (vi) for $R_{\vec{x}}$ at stage s_1. Consider again $\vec{y}$ such that $R_{\vec{y}}$ is of higher priority than $R_{\vec{x}}$ and such that $\vec{x}$ is $\vec{y}$-near. Recall from above that at s_1 we have $I_{\vec{y}} \subset I_{\vec{x}} \subset B$, and that one of the following holds for $\vec{y}$:

- There is no a such that $g(a)$ is $\vec{y}$-near and $g(a) \not\subseteq \vec{y}$, or
- such a exists where $a \in C$, or
- $g(a) \subseteq \vec{y}$ for every $a \in C$.

Since $\vec{y}$ is of lower priority than $g(c)$ for every $c \in C$, it follows that (vi) is not a problem for $\vec{y}$ if the first or second option above holds. So, let's assume the third option for $\vec{y}$ holds. As $\vec{y}$ is of higher priority, we certainly have $\vec{x} \not\subseteq \vec{y}$. Hence $\max \vec{y} < \max \vec{x}$ and we certainly also have $\min \vec{y} < \min \vec{x}$. When evaluating condition (vi) at stage s_1, the first $g(d_0) \subseteq \vec{y}$ to be defined must lie to the right of the last $g(d_1) \subseteq \vec{y}$ to be defined (the same argument as above). So condition (vi) is not a problem for $\vec{y}$. This shows that $R_{\vec{x}}$ must receive attention at stage s_1 in Phase 2. □

We are now ready to complete the proof of Theorem 3.1. Given a linear order $\mathcal{A}$, denote by $C(\mathcal{A})$ the condensation of $\mathcal{A}$, and $[x]$ to be all the elements of the same block as $x \in \mathcal{A}$. (Recall that the condensation of a linear order $\mathcal{A}$ is the set $\{[x] \mid x \in \mathcal{A}\}$ ordered in the obvious way). The following fact is immediate:

Fact 3.12. *Let $\mathcal{A}$ and $\mathcal{B}$ be linear orders. Then $\mathcal{A} \cong \mathcal{B}$ if and only if there exists an isomorphism $\varphi : C(\mathcal{A}) \mapsto C(\mathcal{B})$ such that for every $x \in \mathcal{A}$, $[x]$ has the same order type as $\varphi([x])$.*

Now let us see why the statement of Theorem 3.1 is true. Suppose $\mathcal{M}$ is a computable copy of $2 \cdot \mathcal{L}'$. The entire proof up to now constructs a computable $\mathcal{L}$ and an order preserving $g : \mathcal{L} \mapsto \mathcal{M}^{<\omega}$. Let $g^* : C(\mathcal{L}) \mapsto C(\mathcal{M})$ be the map induced by g, i.e., $g^*([a]) = [\min g(a)]$. Let us now verify that g^* is an isomorphism. If $[a] = [b]$ but $g(a)$ and $g(b)$ are in different blocks, then by Lemma 3.10 and 3.11 and the fact that g is order preserving, we can assume that if x lies between $g(a)$ and $g(b)$ then $[x]$ is finite. By Lemma 3.8, $g^{-1}([x])$ is nonempty, and contains elements between a and b. This is impossible as there are infinitely many such x, but only finitely many elements between a and b. Hence g^* is well-defined.

Now suppose that a and b are in different blocks but $g(a)$ and $g(b)$ are in the same block. Assume $a < b$. If there exists some $a < c < b$ such that $g(c) \supsetneq g(a)$ and $g(c) \supsetneq g(b)$, then by Lemma 3.4(iii) we have $c > b$ or $c < a$, which is impossible. So we assume that this is not the case. Since $a < b$ we have that $g(a)$ lies to the left of $g(b)$, or $g(a)$ and $g(b)$ are comparable. First assume that $g(a)$ lies to the left of $g(b)$. Now $g(c)$ has to be comparable with either $g(a)$ or $g(b)$, or it lies in between, but only finitely many c can have $g(c)$ in between, so we can apply Lemma 3.9 to get a contradiction. Now assume, without loss of generality, that $g(a) \subseteq g(b)$. But now $g(c)$ must be comparable with $g(a)$, and as we assumed $g(c) \not\supseteq g(b)$, we get a contradiction to the fact that there are infinitely many such c. Hence g^* is injective.

That g^* is surjective obviously follows from Lemmas 3.8, 3.10 and 3.11. As g is order preserving and g^* is injective, hence g^* is order preserving. Hence g^* is an isomorphism. Since $C(2 \cdot \mathcal{L}) \cong C(\mathcal{L})$ and from Lemmas 3.8, 3.10 and 3.11, and the fact that g^* is injective and well-defined, we see that $[x]_{2\cdot\mathcal{L}}$ and $g^*([x]_{\mathcal{L}})$ have the same order type. Hence $2 \cdot \mathcal{L} \cong \mathcal{M} \cong 2 \cdot \mathcal{L}'$. Since $C(2 \cdot \mathcal{L}) \cong C(\mathcal{L})$ and $C(2 \cdot \mathcal{L}') \cong C(\mathcal{L}')$, it follows from Fact 3.12 that $\mathcal{L} \cong \mathcal{L}'$. Hence, $\mathcal{L}'$ is computable, completing the proof of Theorem 3.1. □

4. The case $n = 1$

Let τ be a nonempty computable strongly η-like linear order. It means that τ does not contain infinite blocks. Let k be the size of a largest block of τ. Suppose that τ contains only finitely many k-blocks. Thus, if $\tau \cdot L$ has a computable copy for some L, then it is easy to see that $k \cdot L$ has a $\mathbf{0}'$-computable copy. Now, by Theorem 2.1 L has also a $\mathbf{0}'$-computable copy. We guess that such type τ gives the answer for Problem 1.1 in the case $n = 1$. Namely, our conjecture is

Conjecture 4.1. *Let τ be computable. The following are equivalent:*

(1) τ is strongly η-like with finitely many largest blocks and nonempty.

(2) For any countable linear order $\mathcal{L}$, $\mathcal{L}$ has a computable copy iff $\tau \cdot \mathcal{L}$ has a computable copy.

(3) For any countably infinite linear order $\mathcal{L}$, $\mathcal{L}$ has a computable copy iff $\tau \cdot \mathcal{L}$ has a computable copy.

References

[ATF09] Alaev, Pavel E.; Thurber, John J.; Frolov, Andrey N., *Computability on linear orderings enriched with predicates*, Algebra and Logic **48** no. 5 (2009), 313–320.

[AK00] Ash, Christopher J.; Knight Julia F., *Computable structures and the hyperarithmetical hierarchy*, Studies in Logic and the Foundations of Mathematics, 144, North-Holland, Amsterdam, 2000.

[Do98] Downey, Rodney G., *Computability theory and linear orders*, In: "Handbook of Recursive Mathematics, Vol. 2", Ershov, Yuri L.; Goncharov, Sergey S.; Nerode, Anil; Remmel, Jeffrey B., (eds.), Studies in Logic and the Foundations of Mathematics, Elsevier, Amsterdam, 1998, pp. 823–976.

[DK92] Downey, Rodney G.; Knight, Julia F., *Orderings with α-th jump degree $\mathbf{0}^{(\alpha)}$* Proc. Amer. Math. Soc. **114** (1992) 545–552.

[Fe76] Fellner, Stephen M., *Recursive and finite axiomatizability of linear orderings*, Ph. D. Thesis (Rutgers, New Brundswick, NJ, 1976).

[Fr06] Frolov, Andrey N., *Δ^0_2-copies of linear orderings*, Algebra and Logic **45** no. 3 (2006), 201–209.

[Fr12] Frolov, Andrey N., *Linear orderings. Coding theorems*, Uchenye Zapiski Kazanskogo Univ. 154 mo. 2 (2012), 142–151. (English translation forthcoming in Lobachevskii J. Math., as *The coding theorems of linear orderings*)

(Frolov) Higher Institute of Information Technology and Intelligent Systems, Kazan Federal University, Kazan 420008, RUSSIA

Email address: a.frolov.kpfu@gmail.com

(Lempp) Department of Mathematics, University of Wisconsin, Madison, WI 53706-1388, USA

Email address: lempp@math.wisc.edu

URL: http://www.math.wisc.edu/~lempp/

(Ng, Wu) School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371, Republic of Singapore

Email address: kmng@ntu.edu.sg

URL: http://www3.ntu.edu.sg/home/kmng/

Email address: guohua@ntu.edu.sg

URL: http://www3.ntu.edu.sg/home/guohua/