THE BLOCK RELATION IN COMPUTABLE LINEAR ORDERS

Michael Moses
moses@gwu.edu

Department of Mathematics
The George Washington University

1

Abstract

The block relation $B(x, y)$ in a linear order is satisfied by elements that are finitely far apart; a block is an equivalence class under this relation. We show that every computable linear order with dense condensation-type (i.e., a dense collection of blocks) but no infinite, strongly η-like interval (i.e., with all blocks of size less than some fixed, finite k) has a computable copy with the non-block relation $\neg B(x, y)$ computably enumerable. This implies that every computable linear order has a computable copy with a computable non-trivial self-embedding, and that the long-standing conjecture characterizing those computable linear orders every computable copy of which has a computable non-trivial self-embedding (as precisely those that contain an infinite, strongly η-like interval) holds for all linear orders with dense condensation-type.

Introduction

Particularly attractive results of mathematical logic are those that establish an equivalence between the (syntactic) definability of a mathematical property in a particular mathematical language and the (semantic) 'algebraic' characteristics of that property. Representative of these results is the famous result of Ash and Nerode [1981] in which (under certain 'extra decidability' conditions) a relation is shown to be *intrinsically* computable in a computable mathematical structure (meaning that it is computable in every computable copy of that structure) if and only if it and its complement are definable in the structure by computable Σ_1 $\mathcal{L}_{\omega_1\omega}$ formulae. Characterizing the other side of the coin, intrinsically *non*-computable relations (for which *no* computable copy of the structure exists in which the relation is computable), has proven more difficult; the defining formulae required will be complex. Tools such as the 'separator' construction of Jockusch and Soare [1991] and other techniques used to analyze the degree spectra of relations and structures suggest approaches that were not evident before. The main result of this paper was arrived at as part of a continuing project along those lines; it implies (see Corollary 1) that there is no computable linear order all of whose non-trivial self-embeddings are intrinsically non-computable.

A *computable linear order* L is one with universe $\mathbb{N}$ on which the order relation $<_L$ is computable. By a *computable copy* of a linear order (or an order type) under discussion we will mean a computable linear order that is isomorphic (but not necessarily computably isomorphic) to the linear order in hand. The *block relation* $B_L(x, y)$ is that satisfied by elements x, y for which the

L-interval $[x, y]$ (or $[y, x]$ if $y <_L x$) is finite. In one of the earliest results about these structures, Denisov (see Goncharov and Nurtazin [1973]) and Tennenbaum (see Rosenstein [1982]) showed independently that there is a computable linear order of type $\omega + \omega^\star$ in which the block relation is not computable. In [1970] Feiner established the existence of a computable linear order no computable copy of which has the block relation computable; and in [1986] Moses characterized intrinsically computable relations on linear orders (showing that the Ash-Nerode characterization holds for linear orders without the need for 'extra decidability' conditions).

The main result of this paper establishes that every computable linear order with dense condensation-type but no infinite, strongly η-like interval has a computable copy with computably enumerable non-block relation. The *condensation-type* of a linear order is the order-type of its 'finite condensation', i.e., the linear order obtained by factoring out by the block relation, clumping each block into a single point. (See Rosenstein [1982] for further details.) A linear order (or interval in a linear order) is η-like if it is infinite but contains no infinite block; it is *strongly η-like* if, in addition, all its blocks are smaller than some fixed, finite k.

In [1940] Dushnik and Miller observed that every countably infinite linear order has a non-trivial self-embedding. The characterization of linear orders every computable copy of which has a computable non-trivial self-embedding remains open, despite repeated salvos, and despite the straightforward and long-standing conjecture (Kierstead [1987]): every computable copy of a computable linear order has a computable non-trivial self-embedding if and only if the linear order contains an infinite,

4

strongly η-like interval. The main result of this paper establishes (see Corollary 2) that this conjecture holds for linear orders whose condensation-type is dense. (This extends slightly a recent result of Downey, Kastermans, and Lempp [2009] in which the characterization is shown to hold for all η-like linear orders.) It also establishes (see Corollary 3) the corresponding characterization for the class of computable linear orders with computably-enumerable non-block relation: every copy in this class has a computable non-trivial self-embedding if and only if the linear order contains an infinite strongly η-like interval.

The Main Result

THEOREM 1:

Every computable linear order of condensation-type η with no infinite, strongly η-like interval has a computable copy with computably enumerable non-block relation.

PROOF:

Let L be a computable linear order of condensation-type η with no infinite, strongly η-like interval; we shall construct a computable linear order M with computably enumerable non-block relation and an isomorphism $f : L \to M$.

Preliminary Definitions:

Consider the universe $\mathbb{N}$ of L as being enumerated in stages, one element at a time, with L^s denoting the finite linear order enumerated by stage s, on $\{1, 2, \cdots, s\}$, ordered according to L's computable order relation $<_L$. (We take the liberty of leaving 0 out of $\mathbb{N}$ so that n is the nth element of $\mathbb{N}$, which obviates some unnecessary, and perhaps confusing, pedantry in what follows.) We will construct a computable copy M of L by enumerating its universe $\mathbb{N}$ in stages (usually several elements at each stage), with order relation $<_M$ derived from L, which order will never subsequently be changed, thus making M a computable linear order. We will also enumerate at each stage some pairs of M-elements into M's non-block relation $\neg B_M(x, y)$, which enumeration will never subsequently be withdrawn (thus making it computably enumerable). We will also define at each stage s a partial function $f^s : L^s \to M^s$, which sequence of functions will define the isomorphism $f : L \to M$ as the limit along a

particular subsequence of stages; i.e., we will define an infinite subsequence of stages $s_1, s_2, s_3, \cdots$ such that, for each a in L and b in M, $f^{s_i}(a)$ and $(f^{s_i})^{-1}(b)$ will remain unchanged for all but finitely many i, which limiting values are defined to be $f(a)$ and $f^{-1}(b)$.

Our strategy to make M isomorphic to L while keeping $\neg B_M(x,y)$ computably enumerable is straightforward: at each stage of our construction we will seek to define an image in M for each L-block and enumerate pairs of M-elements into $\neg B_M(x,y)$ if they lie in images of different L-blocks. We will access those L-blocks via their least-block-elements, i.e., the $\mathbb{N}$-least element of each block in L, defined by the Π_2 formula

$\forall y <_{\mathbb{N}} x \; \forall k \; \exists z >_{\mathbb{N}} k \; ((y <_L z <_L x) \vee (x <_L z <_L y)).$

We will use a computable binary relation $R(x,y)$ that underlies this Π_2 formula (in the sense that n is a least-block-element if and only if $R(n,y)$ holds for infinitely many y); an L-element n appears to be a least-block-element at stage s (we will say that n is *on* at that stage) if $R(n,s)$ holds. The true least-block-elements in L, and only the true least-block-elements, will be *on* at infinitely many stages. As observed in Jockusch [1966], we can select the $R(x,y)$ so that, for each n, there will be infinitely many stages at which the elements from among $1, 2, 3, \cdots, n$ that are *on* are precisely the true least-block-elements from among those first n L-elements. Our construction will at each stage seek, for the L-elements n that are *on*, an image in M for n's block in L, of the right size and in the right location. The image of an L-block whose least-block-element is *off* (i.e., not *on*) will need to be incorporated into another M-cluster that is enumerated subsequently (so as not to compromise $\neg B_M(x,y)$), and incorporated whole, ready to come back to life when (if) that

least-block-element next comes back *on*. The fact that every infinite interval in L contains arbitrarily large blocks is just what we need in order to accomplish this: a large enough true block will eventually appear where we need it in order to incorporate the image of an L-block whose least-block-element is *off*. Some delicacy, nothing too onerous, is required to identify the true L-block whose image in M will incorporate this fallow M-cluster.

The image in M^s of the L-block around the least-block-element n will carry an identifying label. This labeled cluster of elements will be contiguous in M^s and will remain so (as long as the cluster carries the label): no element will ever be introduced into the cluster other than at the margins. Since we can guarantee only that a true least-block-element n will appear *on* infinitely often (not that it will appear *on* cofinitely often), we will need to preserve the image of its block through those stages when it is *off*. During those stages we will seek to incorporate this M-cluster labeled for n into images of other L-blocks, causing this cluster to take on additional labels for some other seeming least-block-elements, which labels will be shaken off when n next comes back *on*. Elements that are jettisoned in this way will never subsequently return to the M-cluster, thus maintaining the integrity of M's non-block relation. So, for each true least-block-element n of L, the image in M of the L-block around n will (after a few false starts) develop as a contiguous cluster of labeled elements in M, which may contain within it another labeled cluster, denoting the fallow image in M of the block around an L-element that was once *on* but is no longer (which may contain another fallow block within it, etc.), and will, during those stages when n is *off*, become temporarily incorporated into other larger labeled clusters, which labels will subsequently be shaken off. This im-

age for n's block will grow and shrink according to the current size of n's block in L, our approximation at each stage (which we define a couple of paragraphs below) to the final L-block around n, which is, unfortunately, no better than Σ_2-definable. We will, moreover, need to manage not just the size of the image: if the L-block around n turns out to be infinite (i.e., one of ω, $\omega^\star$, or ζ), we must ensure that its image will be a block of the same order type.

We will use a modified version of the usual Π_2 guessing tree to drive our construction: the standard part imagined as branching downward, with branching at level n denoting the guess as to whether or not n is indeed a true least-block-element in L, with the left branch denoting the positive guess and the right branch the negative guess. The (finite) path that is *on* at stage s is the one from the root to the sth level that branches left at each level n if n is *on* at stage s and branches right otherwise. The (infinite) *true* path is the one that branches left at each level n precisely if n is a true least-block-element in L. We have the usual features: for each level n, the *true* path is the leftmost one that coincides infinitely often with the *on* path at the first n levels. Our construction will mesh finitely many separate constructions, one along each path of an initial segment of the tree, acting along the *on* path at each stage and using as labels on clusters in M the node σ at which those labels were created, which labeled M-cluster will denote the image of the L-block around the element that σ references. As it is, however, this tree is too simple for our purposes: as described, it will produce an image in M, of the right size and in the right location, for each L-block around a true least-block-element; it will however leave scattered throughout M the fallow images of L-blocks

centered around elements that once seemed to be least-block-elements but are no longer. In order to incorporate into these constructions the search for an appropriate preimage for each fallow M-cluster, i.e., an L-block centered around a true least-block-element into whose image we can incorporate the fallow M-cluster, we will expand our Π_2 guessing tree by adding immediately below each level n two additional sublevels (described three paragraphs below), which levels will be dynamic, in that the L-elements referenced by those nodes will (may) change as the construction progresses.

The *block at stage s* around an L-element n (a potential least-block-element) is defined to be the contiguous cluster of L^s-elements around n extending as far as possible on both sides without including any element $\mathbb{N}$-less than n, nor any element that was enumerated into L at or after the last stage (previous to s) when n was *on* (or stage 1 if n was never *on*), nor any element (other than n) that came *on* at or after the last stage when n was *on*. This is the collection of elements that, under the assumption that n is a least-block-element, stands a chance at stage s of being part of n's (true) block. There is, of course, no guarantee that they will remain contiguous as the L-enumeration continues, nor even that they are members of n's (true) block. Notice that there is a block at stage s around each L^s-element n (of size one, if not larger) and that, if m is not a true least-block-element, the block around m will eventually stop changing, and will then be a sub-block of the (true) block around m (centered around some least-block-element n). And, if n is a true least-block-element and m is truly in n's block, then, after finitely many stages, *m and all of m's block* will always be in n's block (there will come a stage after which no

new elements are enumerated between m and n and after which no elements between m and n are ever *on*). Notice also that, if m and n are in separate blocks in L, there will be infinitely many stages at which they will not be members of each other's blocks (since infinitely often there will be points between them that are *on*). Unfortunately, it could well happen that they also will be members of each other's blocks infinitely often; we must make sure that this does not inhibit the growth of the separate images in M of their separate blocks in L. Finally note that our definition allows adjacent blocks in L^s to overlap.

As mentioned previously, our main concern will be to incorporate every fallow M-cluster into the image of a true L-block. The fallow M-cluster around an M-element n is the cluster defined by all the labels that n carries of nodes on paths to the left of the node σ at which we are acting, i.e., the cluster of elements around n that also carry (any one of) those labels. Any preimage that we find for n while acting at σ must incorporate wholly, into a single L-block, all of this fallow M-cluster around n (in anticipation of it coming back *on* at some later stage). If n carries no such labels (of nodes left of σ), then there is no fallow M-cluster for us to worry about; however, rather than set up separate machinery to handle this simpler case, we will consider the singleton n, by itself, to be the fallow M-cluster around n.

The search for a preimage for the fallow M-cluster around n will begin rather simply: we will identify the size of the block that we need (the current size of the fallow M-cluster) and the L-interval (between the L-blocks around higher priority elements) in which we need it. We will maintain an ordered list of contiguous sequences of L-elements of the required size within this interval, adding new contiguous sequences to the right end of

the list as they are enumerated into L and removing from the list those sequences that are no longer contiguous (within which a new L-element has been introduced). The leftmost on this list will be the selected preimage at that stage. Notice that, since every infinite interval in L contains arbitrarily large blocks, the selected preimage will, after a finite number of stages, settle on a truly contiguous sequence of L-elements suitable for use as the preimage for the fallow M-cluster around n.

Since our construction is centered around least-block-elements, we will need to identify the least-block-element in L within whose block this selected preimage lies. We will maintain an ordered list of possible least-block-elements consisting of all i that are $\mathbb{N}$-less than the $\mathbb{N}$-least element of the selected preimage (no other i could possibly be the least-block-element whose block contains that preimage). We will order the i on the list according to the last time that a new element was introduced between i and the selected preimage, moving i to the right end of the list each time a new element is introduced between i and the selected preimage (ordering them as in $\mathbb{N}$ if there is more than one such i being moved). Notice that, for a particular selected preimage, this procedure will eventually identify, as the leftmost element in this list that appears *on* infinitely often, the true least-block-element i in L whose block truly contains that selected preimage. This i will eventually cease being moved to the right end of the list and any elements to its left on the list will be in i's L-block, i.e., the same block as the selected preimage, but will not be true least-block-elements and, consequently, will eventually cease appearing *on*. We manage this procedure by splitting our Π_2 guessing tree according to this list of all the i that could possibly be the least-block-element whose L-block

contains the selected preimage, which splitting will be dynamic in that the i that these nodes will reference will (may) change from stage to stage. As noted, the i's referenced by the leftmost of these nodes will eventually cease changing, and one of them, the true least-block-element whose block contains the selected preimage, will appear *on* infinitely often.

An obvious problem will occur if the true least-block-element i so identified (whose block contains the selected preimage) turns out to be an n of higher priority. The L-block around that n will already have an image in M and M's non-block relation $\neg B_M(x, y)$ will not allow us to incorporate the fallow M-cluster into that existing image. So we need to make sure that the selected preimage for the fallow M-cluster is not contained within the blocks around the higher priority n. We do this by identifying a pair p, q of true least-block-elements within the appropriate interval in L (strictly between the higher priority n) and conducting our search for a preimage within the L-interval $[p, q]$. In order to settle on one such pair p, q we will maintain an ordered list of pairs p, q of L-elements (ordered according to when the pair was introduced into L) that lie within the appropriate interval between higher priority n, working at each stage within the interval defined by the leftmost pair on this list both of whose elements are *on*. Notice that, for a particular set of higher priority n, this procedure will identify, as the leftmost pair on this list that appears *on* infinitely often, a pair p, q of true least-block-elements (within which infinite interval $[p, q]$ we are guaranteed to find a preimage that is not part of the block around any higher priority n). All the pairs to the left of p, q will eventually cease appearing *on* and the preimage searches conducted between those pairs will be abandoned; all preimage

searches conducted between pairs to the right of p, q will be rescinded the next time p, q comes *on*. We manage this procedure with another branching of our Π_2 guessing tree, with the nodes referencing in the order that they were enumerated into L, from left to right, the p, q in the above-mentioned list, and conduct a search as described in the previous paragraph within each interval $[p, q]$ for a preimage for the fallow M-cluster and for the true least-block-element i whose block contains the selected preimage. The pairs p, q that these nodes reference will not change but new nodes will be added to the right end as new elements are enumerated into L.

This then is our Π_2 guessing tree, with three sublevels for each n: branching at sublevel-one, if necessary, to manage the search for an image in M for the L-block around the least-block-element n. Each node at the previous level will have a binary split at sublevel-one for n denoting the guess as to whether n is a true least-block-element in L (left branch) or not (right), i.e. whether or not n is *on*. Branching at sublevels two and three, if necessary, to manage the search for a preimage in L for the fallow M-cluster around n: each of the nodes from sublevel-one will split at sublevel-two into finitely (but unboundedly) many branches denoting the ordered list of all pairs p, q that lie in L^s between the higher priority elements between whose images the M-element n lies, and within which interval $[p, q]$ we will seek to identify a preimage for the fallow M-cluster around n. Each of these sublevel-two nodes p, q will split at sublevel-three into finitely (but unboundedly) many branches denoting the ordered list of possibilities for the least-block-element i within $[p, q]$ whose block contains the selected preimage. Note that these second and third sublevels are dynamic in that the num-

ber of branches at sublevel-two will increase as more pairs p, q become enumerated between the higher priority elements (although the p, q that each node references will not change), and the L-elements i referenced by the nodes at sublevel-three will change as elements are introduced into L between i and the selected preimage (although the number of branches will not change). Branching at these three sublevels for n occurs only if necessary: if n in L is already referenced by an earlier sublevel-three node on the branch (as a least-block-element i whose block contains the preimage for some M-element $m <_{\mathbb{N}} n$), we do not need the sublevel-one binary split for n. Similarly, branching at sublevels two and three for n occurs only if the M-element n does not already carry the label of some earlier node on the branch. The *on* path through this tree at stage s is defined in the obvious way: the left or right branch of the sublevel-one binary split for each $n \leq_{\mathbb{N}} s$ depending on whether or not n is *on* at stage s, followed by the leftmost sublevel-two pair p, q that are both *on* (if any), followed by the leftmost sublevel-three i that is *on* (if any). Note the 'if any': the *on* branch at stage s will not extend beyond sublevel-one for an n if no p, q for that M-element n is *on*, and not extend beyond sublevel-two if no suitable i within that $[p, q]$ is *on*. Note also that, since the fallow M-cluster around n is determined by (the labels from nodes on branches to the left of) the node p, q at which we're acting, we can give a precise definition of the *on* path at each stage only later, as part of the description of our construction. For the same reason, the existence of the *true* path, the leftmost infinite path whose initial segments will coincide with the *on* path infinitely often, can be established only after the description of our construction.

So we imagine our Π_2 guessing tree as a subtree of the infinite tree branching downward from the root with three sublevels for each $n = 1, 2, 3, \cdots$: a binary split at sublevel-one for n (from each of the sublevel-three nodes for $n-1$), followed by an ω split from each of these nodes at sublevel-two, followed by another ω split from each of these nodes at sublevel-three. Our Π_2 guessing tree will employ at each stage a finite, left-most collection of these nodes at each level. If branching at sublevel-one (or sublevels two and three) for an n is unnecessary, we imagine our tree as taking the leftmost branch, with that sublevel-one node referencing no L-element n (or the sublevel-two node referencing no p, q and the sublevel-three node no i), merely providing a path through to the next level. Otherwise, our guessing tree will include the nodes at sublevels one, two, and three as described in the previous paragraph. We will drive the constructions described in the previous paragraphs along each path of this dynamic Π_2 guessing tree, acting along the *on* path at each stage. The infinite *true* path will be the leftmost one whose initial segments coincide with the *on* path infinitely often. Construction along paths to the left of the *true* path will eventually cease; construction along paths to the right will always subsequently be rescinded; only the construction along the *true* path will drive inexorably forward. The only nodes σ at level n that will be used as labels are sublevel-one nodes (to label the image in M of the L-block around the least-block-element n referenced by σ) and sublevel-three nodes (to label the image in M of the L-block around the least-block-element i that σ references, which image will include the fallow M-cluster around n). At the end of the construction at each stage the L-block around the least-block-element referenced by a node σ along the *on* path will be

in one-to-one correspondence with the M-cluster labeled σ.

The union of these bijective correspondences over all the nodes σ along the *on* path at stage s, between the L-block around the element that σ references and the M-cluster labeled with that σ, will be used to define the partial function $f^s : L^s \to M^s$. We have to be a little careful here since, as mentioned before, L-elements may belong to more than one L-block (and so have more than one image in M). L-elements referenced by nodes along the *on* path will correspond to just one M-element at stage s. L-elements a that are *off* at this stage may belong to as many as two of the L-blocks around elements referenced by nodes along the *on* path and, consequently, correspond to more than one M-element at that stage. If a has several such images, we define $f^s(a)$ to be the one defined by the highest priority node σ (closest to the root). This will make each $f^s : L^s \to M^s$ a partial bijective function, which functions will allow us to define our $f : L \to M$ isomorphism: we will establish the existence of a subsequence $s_1, s_2, s_3, \cdots$ of stages on which the construction will resemble a 'finite injury' construction of the isomorphism f in that the f^{s_i} will eventually include each L-element and each M-element, and will remain fixed on each of these elements for all but finitely many i. (The existence of this subsequence is predicated on the fact, guaranteed by Jockusch [1966], that the computable $R(x, y)$ underlying our Π_2 least-block-element formula was chosen so that, for each n, there will be infinitely many stages at which the elements from among $1, 2, 3, \cdots, n$ that are *on* are precisely the true least-block-elements from among those first n L-elements.) As mentioned before, M-elements will be ordered when introduced, which order will be preserved, thus making M a computable lin-

ear order. Pairs of M^s-elements not tagged with a common label will never assume a common label at a later stage and will end up in separate blocks in M, which ensures that the enumeration at each stage of all pairs of elements in M^s that do not have a common label will provide a computable enumeration of M's non-block relation $\neg B_M(x, y)$.

A delicacy in our setup, crucial to its success, bears highlighting: the fallow M-cluster around n and its selected preimage within an interval $[p, q]$ are determined at the sublevel-two node referencing that p, q (in step 5 of the construction), not at the sublevel-three nodes below, at which we will act (in step 8) to incorporate n and its fallow M-cluster into the image of an L-block around an *on* element i within $[p, q]$. That fallow M-cluster around n is defined by the labels that n carries from nodes on paths to the left of this node p, q, whereas, when acting at the sublevel-three node i below this node p, q, we must in fact incorporate into the image of the L-block around i the larger fallow M-cluster around n defined by *all* labels that n carries from nodes on paths to the left of the node i, which is one sublevel below the node p, q. The additional labels on n that make this fallow M-cluster larger would come (only) from the sublevel-three nodes j immediately below this node p, q and to the left of the node i at which we're acting, labels that were placed to mark the images of the L-blocks around those j. We are able to skirt this obstacle by dint of the fact that, if i is indeed the true least-block-element whose block contains the selected preimage of the fallow M-cluster around n as calculated at the node p, q, then, eventually, those j *and their L-blocks* will all be contained within the L-block around i.

The Construction:

For each $s = 1, 2, 3, \cdots$ in turn begin the construction at stage s by starting at the root of our Π_2 guessing tree and working downward, defining the *on* path through the three sublevels for each $n \leq_{\mathbb{N}} s$, and acting as described below at each node along that path. Our action will define for some of those nodes an L-element that the node references and a cluster of M-elements labeled with that node, which labeled cluster will be in bijective correspondence with the block around the referenced L-element. The order $<_M$ is derived from $<_L$ and defined as elements are introduced into M. At the end of each stage, all pairs of M-elements that do not have a common label are enumerated into M's non-block relation $\neg B_M$, and the bijective correspondences between the L-elements referenced by nodes along the *on* path and the M-clusters labeled with those nodes are used as already described to define the partial bijection $f^s : L^s \to M^s$ (each $f^s(a)$ is defined to be the image in the bijective correspondence defined by the highest priority node along the *on* path whose L-block includes a). One piece of the algorithm is executed repeatedly through the construction at each stage and is presented here as a blanket instruction: as each node along the *on* path is defined, delete the effects of all earlier action at nodes on all paths to the right (i.e., consider those nodes as no longer referencing any L-elements and remove all labels in M tagged with those nodes).

After the construction at stage s for $1, 2, 3, \cdots, n-1$, continue the construction for n ($\leq_{\mathbb{N}} s$) by performing in order the following steps:

At sublevel-one for n:

1: If n is *off* at stage s, the current *on* path takes the right

branch at sublevel-one and no further action is performed
at this sublevel; go to step 3. If n is *on* at stage s, the
current *on* path takes the left branch. If n is *on* but is
already referenced by an earlier node along the current *on*
path, then no further action is performed at this sublevel;
go to step 3. Otherwise, n is referenced by this node at
sublevel-one for n.

2: Arrange that the M-cluster labeled with this node n is in
bijective correspondence with the L-block around n, with
the correspondence mapping n to the $\mathbb{N}$-least element in
the M-cluster. Begin with the existing M-cluster around
n labeled with this node; build the cluster from scratch if
none exists. Remove labels from some of the outermost
elements in the cluster, or add new elements labeled with
this node to the outer ends of the cluster (using the $\mathbb{N}$-least
elements that have not yet been enumerated into M), as
necessary, to make the M-cluster the same size as the L-
block around n at this stage. Do this carefully: remove
labels from elements or add new elements to the left end of
the M-cluster so that there are exactly as many elements
in the cluster to the left of its $\mathbb{N}$-least element as there are
to the left of n in the L-block around n; then handle the
right side similarly. If this is a completely new M-cluster,
situate it correctly in M, with order derived from L, with
respect to all M-clusters that are labeled with earlier nodes
along the *on* path (and as far left as possible with respect
to all other M-elements).

At sublevel-two for n:

3: If the M-element n is labeled with an earlier node along

the current *on* path, then the current *on* path goes through the leftmost nodes at sublevels two and three for n and no further action is performed at these sublevels. Go to step 1 of the construction for $n + 1$.

4: Otherwise, consider the location in M of this element n with respect to all M-clusters labeled with earlier nodes along the current *on* path. We must find a preimage for n that lies in the same interval in L with respect to the preimages of these higher priority labeled M-clusters. The sublevel-two nodes for n are taken to reference, from left to right, an ordering (according to when they were enumerated into L) of all pairs p, q that lie within this interval in L^s. If none of these pairs p, q is *on* (i.e., both p and q *on* at stage s), the current *on* path does not extend to sublevel-two for n and no further action is performed at these sublevels; go to step 1 of the construction for $n + 1$. If at least one of these pairs p, q is *on*, then the next node along the current *on* path is the leftmost of these p, q that is *on* at this stage.

5: Consider the selected preimage within $[p, q]$ (the leftmost in the list of contiguous sequences of the right size) for the fallow M-cluster around n (defined by the labels on n from all nodes on paths to the left of p, q), both of which are described in the preliminary definitions section. If too few elements have been enumerated into $[p, q]$ for a preimage to exist, then the current *on* path does not extend beyond sublevel-two for n and no further action is performed at these sublevels; go to step 1 of the construction for $n + 1$. If this fallow M-cluster around n or its selected preimage within this $[p, q]$ has changed since the last time the node

p, q was along the current *on* path, delete the effect of all earlier action at nodes on all paths below this node.

At sublevel-three for n:

6: Consider the leftmost sublevel-three nodes immediately below the node p, q as referencing a list of all i in $[p, q]$ that are $\leq_{\mathbb{N}}$ the $\mathbb{N}$-least element of the selected preimage (of step 5), ordered from left to right according to the last time that an element was enumerated into L between i and the preimage (as described in the preliminary definitions section). If the i referenced by one of these nodes has changed since the last stage, delete the effect of all earlier action at that node and at all nodes on paths below it. For the remaining sublevel-three nodes (that do reference an i), cut back, if necessary, the M-cluster labeled with that node so that it is no larger than the L-block around the i that the node references (i.e., remove that node's label from all M-elements in the cluster that correspond to L-elements that were previously in the L-block around i but are no longer).

7: If there is no i on this list that is *on* at this stage and whose current block in L wholly contains the selected preimage of the fallow M-cluster around n *and* the block around the L-element referenced by each sublevel-three node below this p, q and to the left of the node i, then the current *on* path does not extend to sublevel-three for n and no further action is performed at this sublevel; go to step 1 of the construction for $n + 1$. Otherwise the next node along the current *on* path is the leftmost node i with these properties.

8: Arrange that the M-cluster labeled with this node i (including the *whole* labeled block around n) is in bijective corre-

spondence with the L-block around i, which correspondence maps the selected preimage to the fallow M-cluster around n. Begin with the M-cluster around n that includes not only the fallow M-cluster around n as calculated at the node p, q (see step 5) but also all other labels on n (from all sublevel-three nodes below p, q and on or to the left of the current *on* path). Add, if necessary, new elements to the outer ends (using the $\mathbb{N}$-least elements that have not yet been enumerated into M) to make this M-cluster around n (labeled with the node i) the same size as the L-block around i. Note that i was selected (in step 7) to make this possible. Do this carefully: begin with the M-cluster around n consisting of all elements that share a label with n and add, if necessary, new elements to the left end of the cluster so that there are exactly as many elements in the cluster to the left of the fallow M-cluster around n as there are to the left of its selected preimage in the L-block around i; then handle the right side similarly. This ends the construction for n; go to step 1 of the construction for $n + 1$.

Final Verification:

Preliminary observations:

- M is a computable linear order: elements are ordered as they are introduced into M, which order is never changed.

- M's non-block relation $\neg B_M(x, y)$ is computably enumerable: elements in M that have a common label will never increase their distance from each other so long as they retain that label (since no elements are ever introduced internally into a labeled block), and elements that do not have a

common label will never assume one at a later stage (only new elements are used to grow M-clusters).

- All M-clusters with labels from nodes along the *on* path will be ordered with respect to each other exactly as their preimages are ordered in L, and no M-element will ever belong to more than one of these M-clusters (labeled with nodes along the *on* path).

- A labeled M-cluster will grow only at stages when the labeling node is along the *on* path and will, at the end of those stages, be the same size as the block at that stage around the L-element that the node references.

- An M-cluster labeled with a node will disappear completely (all labels tagged with that node removed) only if the node is on a path to the right of the *on* path, or below a sublevel-two node p, q at which the fallow M-cluster around n or its selected preimage have been redefined (step 5), or at or below a sublevel-three node whose referent has changed (step 6).

We show now that there is an infinite *true* path that, for every level n, is the leftmost path down to that level that coincides with the *on* path infinitely often; that every node on this path will change the L-element that it references only finitely often, with its final referents (if any) being true least-block-elements; and that every M-cluster labeled with a node on this path will disappear (completely) only finitely many times. We will establish these facts simultaneously, by induction on n, through the three sublevels for each n.

Assume that we have established the above-mentioned facts for $1, 2, \cdots n - 1$. Let s be a stage after which the *on* branch is never to the left of the *true* branch all the way down to sublevel-three for $n - 1$ (such stages, as we have noted before, are guaranteed by Jockusch [1966]), by which stage all these nodes have taken on their final referents; and after which stage no M-cluster labeled with one of these nodes ever (completely) disappears. We will now establish the facts for the three sublevels for n. All stages mentioned below are intended to be subsequent to s.

- At sublevel-one for n: if n is not a true least-block-element, consider a stage after which n never again appears *on*. At the next stage when the *on* path coincides with the *true* path down to sublevel-three for $n - 1$, the *on* path will take the right branch at sublevel-one for n as will the *true* path: the *on* path will never again be left of this node; this node will never again take on a referent, nor will it ever again label an M-cluster. If n is a true least-block-element, consider the next stage when n is *on* and the *on* path coincides with the *true* path down to sublevel-three for $n - 1$; the *on* path will at this stage take the left branch at sublevel-one for n as will the *true* path, and the *on* path will never again be left of this node. If n is already referenced by some (sublevel-three) node earlier on the *true* path, this node will never again take on a referent, nor will it ever again label an M-cluster. If n is not referenced by any earlier node on the *true* path, then it will subsequently always be the referent of this node, which will label an M-cluster (corresponding to the L-block around n) that will never (completely) disappear.

- At sublevel-two for n: consider a subsequent stage by which

25

the M-element n has assumed its final position (in or out) with respect to all M-clusters labeled with nodes down to sublevel-one for n on the *true* path (M-elements that do not share a label will never subsequently assume a common one), and when the *on* path coincides with the *true* path down to sublevel-one for n. If n belongs to (precisely) one of the M-clusters labeled with an earlier node on the *true* path then, at this stage, the *on* path (and the *true* path) will take the leftmost branch at sublevels two and three for n; the *on* path will never again be left of these nodes; neither of these nodes will ever again take on a referent; nor will they ever again label an M-cluster. If n belongs to no M-cluster labeled with an earlier node on the *true* path, consider the interval in L between the elements referenced by these earlier nodes on the *true* path between whose images n lies. Since this interval is infinite (the L-elements referenced by these nodes are all true least-block-elements), there will be a least pair p, q (least according to when the pair was enumerated into L) of elements within this L-interval both of which are true least-block-elements. Consider a subsequent stage when p, q is *on* (i.e., both p and q are *on*) and after which no pair enumerated earlier into this L-interval ever appears *on*. At this stage the fallow M-cluster around n as calculated at p, q will have taken on its final form (no further labels from nodes on paths to the left of p, q will be added). After this stage, every time the *on* path coincides with the *true* path down to sublevel-one for n, it will take the branch through this node p, q at sublevel-two for n. Consider a further stage when the selected preimage within this interval $[p, q]$ for the fallow M-block around n

(the leftmost on the list of candidates) is indeed a true set of contiguous L-elements.

- At sublevel-three for n: let i be the true least-block-element within this L-interval $[p, q]$ whose (true) block contains the above-mentioned selected preimage for the fallow M-block around n. Consider a subsequent stage when the (finite) contiguous part of the L-block around the selected preimage that contains all elements that are $\leq_{\mathbb{N}}$ than the $\mathbb{N}$-least element of the selected preimage have been enumerated into L and after which none of these elements (other than i) are ever again on. After this stage the leftmost sublevel-three nodes below node p, q will refer to these elements (that are $\mathbb{N}$-less than the $\mathbb{N}$-least element of the selected preimage), one of which will be i, and there will be no redefining of referents at these nodes. Consider a subsequent stage when i is on and when the on path coincides with the $true$ path down to sublevel-two for n; the on path will at this stage take the branch through node i at sublevel-three for n as will the $true$ path; the on path will never again be left of this node; this node will always refer to i; and there will be an M-cluster labeled with this node i that will always contain the fallow M-cluster around n.

This ends the proof of the above-mentioned three facts.

Consider the subsequence $s_1, s_2, s_3, \cdots$ of stages in which each s_n is the first stage after s_{n-1} at which the on path coincides with the $true$ path down to sublevel-three for n, after which the on path is never to the left of these nodes, after which the L-elements that these nodes reference are never changed, and after which the M-clusters labeled with these nodes are never

(completely) deleted. If, at this stage s_n, the node σ on the *true* path at sublevel-one for n references n, then, since the bijection defined at σ between the L-block around n and its image always maps n to the $\mathbb{N}$-least element of the image, we have that $f^{s_n}(n) = \lim_{i \to \infty} f^{s_i}(n)$. Similarly, if, at this stage s_n, the node σ on the *true* path at sublevel-three for n references an L-element i, then, since the bijection defined at that σ is centered around a map of the selected preimage onto the fallow M-cluster, it follows that the f^{s_i} have by stage s_n taken on their final limiting value on the M-element n. Moreover, since no more elements will be enumerated into L between i and the selected preimage after this stage, these functions have by this stage also taken on their final limiting value on the least-block-element i. So the partial bijections f^{s_i} eventually take on final limiting values on every M-element and on every least-block-element in L.

For the other elements a of L observe that, once all the elements between a and the least-block-element n of a's block have been enumerated into L and have ceased coming *on*, the image of a in the bijection between the L-block around n and its image will remain fixed. The only complication is that a may appear to be in the L-blocks around more than one of the elements referenced by nodes on the *true* path and $f^{s_i}(a)$ may be defined via the L-block around the element referenced by some node earlier than that which references n (an erroneous definition in that a is not truly a member of that earlier L-block). But, once the *on* path coincides at these stages s_i with the *true* path far enough down to include a pair of true least-block-elements around n that separates it from the elements referenced by all earlier nodes on the *true* path, then, at all subsequent stages, the $f^{s_i}(a)$ will be defined correctly, via the correspondence between the L-block

around n and its image.

Thus $f(x) = \lim_{i \to \infty} f^{s_i}(x)$ over these stages s_i defines an isomorphism from the given computable linear order L to the constructed M, which is a computable linear order with computably enumerable non-block relation $\neg B_M(x, y)$.

$\square$

Extending the theorem: our construction requires that the condensation-type of L is not just dense but is dense without endpoints (i.e., η rather than $1 + \eta$, $\eta + 1$, or $1 + \eta + 1$); the result carries over nonetheless ...

THEOREM 1 EXTENDED:

Every computable linear order with dense condensation-type and with no infinite, strongly η-like interval has a computable copy with computably enumerable non-block relation.

PROOF:

Consider any computable linear order L with condensation-type $1 + \eta$ and with no infinite, strongly η-like interval. Form the computable linear order L' by adding to the left of L a decidable copy of $\omega \times \eta$ to produce a computable linear order L' of condensation-type η with no infinite, strongly η-like interval. Apply Theorem 1 to L' to produce a computable copy M' with computably enumerable non-block relation. Produce from this M' an $M \cong L$ by deleting everything in M' to the left of the element that corresponds to the leftmost point of L. If L has no leftmost point, i.e., if its leftmost block is of type $\omega^\star$ or ζ, then delete everything in M' to the left of an element that corresponds to some point in this leftmost block of L, and add to the left of the resulting linear order a decidable copy of $\omega^\star$. In either case, this will produce a computable copy M of L with

29

computably enumerable non-block relation. Computable L with condensation-types $\eta + 1$ and $1 + \eta + 1$ are handled similarly.

$\square$

Situations in which the theorem does not apply: crucial use is made in our construction (in step 4) of the existence of the pair of least-block-elements p, q in L between least-block-elements of higher priority. Their existence could not be guaranteed if L's condensation-type were not η; nor does the result carry over to such linear orders: *there is a computable linear order with no infinite, strongly η-like interval that has no computable copy with computably enumerable non-block relation.* (A *discrete* example of such a linear order, i.e., with every block of type ζ, is presented in Moses [1988].)

Similarly, crucial use is made (in step 5) of the existence of arbitrarily large blocks within every infinite L-interval, which could not be guaranteed if L had an infinite strongly η-like interval; nor does the result carry over to such linear orders: *there is a computable linear order with dense condensation-type that has no computable copy with computably enumerable non-block relation.* (In [1997] Coles, Downey, and Khoussainov present an example of such a linear order, with condensation-type $\eta + 1$, which in fact has no computable copy in which the initial segment with condensation-type η, i.e. everything but the final $\omega^\star$-block, is computably enumerable.)

Note also that, while our $\neg B_M(x, y)$ is computably enumerable, we cannot guarantee the same of its complement $B_M(x, y)$: every time its node comes *on*, a labeled M-block shakes off those 'outer' elements labeled also with nodes on branches to the right. Nor can the result be strengthened to make $B_M(x, y)$ com-

putable: *there is a computable linear order with dense condensation-type and no infinite, strongly η-like interval no computable copy of which has its block relation $B(x, y)$ computably enumerable.* To construct such a linear order, begin with a Π_2 suborder of the standard computable linear order of type η that has no Π_1 copy. (A Π_n linear order is that defined as a suborder of the standard computable linear of type η, or indeed of any computable linear order, by a Π_n subset of the universe $\mathbb{N}$. That there is, for each n, a Π_{n+1} linear order that has no Π_n copy is well known, see Rosenstein [1982].) Produce a computable linear order L from this computable copy of η by replacing each element i with a block of size p_i, the ith prime, if i is an element of the Π_2 suborder, and a block of type ζ if it is not. (That such a computable linear order can be produced from any Π_2 linear order is established in Fellner [1976].) L has condensation-type η and, since no two finite blocks are of the same size, has no infinite, strongly η-like interval. And it has no computable copy M with computable block relation since, if it did, the set $\{x : \forall y >_M x \, (\neg B_M(x, y))\}$ defines a Π_1 copy of the Π_2 linear order that we began with, which does not exist.

Non-Trivial Self-Embeddings

In [1940] Dushnik and Miller observed that every countably infinite linear order has a non-trivial self-embedding. The linear orders constructed independently by Denisov (see Goncharov and Nurtazin [1973]) and Tennenbaum (see Rosenstein [1982]) show that this does not carry over to computable linear orders: not every infinite computable linear order has a computable non-trivial self-embedding. A long-standing open conjecture (see [1987] Kierstead) characterizes the infinite computable linear orders every computable copy of which has a computable non-trivial self-embedding as precisely those with an infinite, strongly η-like interval. Theorem 1 allows us to establish the conjecture for all linear orders with dense condensation-type (see Corollary 2 below). This is a slightly stronger result than the main theorem of Downey, Kastermans, and Lempp [2009], which establishes this result for all η-like linear orders (which have dense condensation-type but no infinite blocks).

Theorem 1 also allows us to characterize, and this time completely, the other side of the coin:

COROLLARY 1:

Every computable linear order has a computable copy with a computable non-trivial self-embedding.

PROOF:

If the linear order has a pair of adjacent blocks then it has an interval of type $\omega + 1$ or $1 + \omega^{\star}$ or $\omega + \omega^{\star}$. Create a computable copy of the linear order by replacing this closed interval with a decidable copy of the same type (and leaving everything else the

same). This computable copy has a computable non-trivial self-embedding (the identity everywhere outside the interval and, inside it, mapping each element to its immediate successor in the ω-block and its immediate predecessor in the $\omega^\star$-block, both of which are computably identifiable). If the linear order has an infinite, strongly η-like interval then it clearly has a computable non-trivial self-embedding (mapping elements within that interval into separate blocks). By our Extended Theorem 1, the remaining computable linear orders all have computable copies with computably enumerable non-block relation, which, as in the strongly η-like interval case, can be used to define a computable non-trivial self-embedding of that copy by mapping elements into separate blocks.

$\square$

COROLLARY 2:

The self-embedding conjecture holds for linear orders with dense condensation-type: *every computable copy of a computable linear order with dense condensation-type has a computable non-trivial self-embedding if and only if the linear order contains an infinite, strongly η-like interval.*

PROOF SKETCH:

Consider the straightforward technique for constructing, from a computable linear order L, a computable copy M with no computable non-trivial self-embedding: a 'finite-injury' construction, meshing the enumeration of a computable copy with the diagonalization across all partial computable functions, the candidates for a possible non-trivial computable self-embedding. The requirements for n, with priority decreasing as n increases, would arrange for an image in M for $n \in L$, for a preimage in

L for $n \in M$, and that the nth partial computable function φ_n is not a self-embedding on M, with, as usual, action to meet the requirements for n being conducted without redefining the $L \to M$ isomorphism on the points that witness our meeting the requirements with priority higher than n. An obvious strategy for ensuring that φ_n is not a self-embedding on M, used for instance to similar purpose in Moses [1988], is to wait until images $\varphi_n(a)$ and $\varphi_n^2(a)$ are defined for some $a \in M$ with all three within the same interval between points of higher priority (on which the $L \to M$ isomorphism cannot be redefined) and with either $a <_M \varphi_n(a) <_M \varphi_n^2(a)$ or $a >_M \varphi_n(a) >_M \varphi_n^2(a)$. Then, every time a new point is enumerated into this interval (between higher priority points) in L, redefine the $L \to M$ isomorphism within this higher-priority interval to 'feed' the M-interval $[a, \varphi_n(a)]$ until it is larger than $[\varphi_n(a), \varphi_n^2(a)]$. Once this has been achieved, all that remains is to ensure that the M-interval $[\varphi_n(a), \varphi_n^2(a)]$ stops growing, which can be arranged by identifying a contiguous set of L-elements of the right size within this higher-priority interval and making it the preimage of the M-interval $[\varphi_n(a), \varphi_n^2(a)]$. Since every infinite L-interval contains arbitrarily large blocks, such a contiguous set of L-elements must exist. We can identify one such contiguous set by maintaining a list of all contiguous sets of the right size (ordered according to when they were enumerated into L) and deleting those that cease being contiguous (as we did in selecting a preimage for the fallow M-block in the proof of Theorem 1). This will eventually leave a true contiguous set as the leftmost in the list. The problem is that we may not be able to redefine the isomorphism from L to M sufficiently so as to make this selected contiguous sequence the preimage of $[\varphi_n(a), \varphi_n^2(a)]$, which would

(could) happen if the selected preimage is in fact in the same block as one of the higher-priority elements. This was resolved in our proof of Theorem 1 by using the points p, q, which ensured that the selected preimage was in a separate block from those of the higher-priority elements. If L has its non-block relation computably enumerable, we can use it to select a preimage for $[\varphi_n(a), \varphi_n^2(a)]$ that is in a separate block from all higher-priority elements, and so carry through the construction.

$\square$

Note that this in fact establishes a result that is stronger than Corollary 2:

COROLLARY 3:

Every computable linear order that has a computable copy with computably enumerable non-block relation has a computable non-trivial self-embedding if and only if it contains an infinite, strongly η-like interval.

References

[1981] C. Ash and A. Nerode, *Intrinsically recursive relations*, 'Aspects of effective algebra', edited by J.N. Crossley (Upside Down A Book Company, Steel's Creek, Australia), 26-41

[1997] R. Coles, R. Downey, and B. Khoussainov, *On initial segments of computable linear orders*, Order **14**, 107-124

[2009] R. Downey, B. Kastermans, and S. Lempp, *On computable self-embeddings of computable linear orders*, J. Symbolic Logic **74**, 1352-1366

[1940] B. Dushnik and and E. Miller, *Concerning similarity transformations of linearly ordered sets*, Bull. Amer. Math. Soc. **46**, 322-326

[1970] L. Feiner, *Hierarchies of Boolean algebras*, J. Symbolic Logic **35**, 365-373

[1976] S. Fellner, *Recursiveness and finite axiomatizability of linear orderings*, Thesis Rutgers University, N.J.

[1973] S. S. Goncharov and A. T. Nurtazin, *Constructive models of complete decidable theories*, Algebra i Logika **12**, 125-142

[1966] C. Jockusch, Jr., *Reducibilities in recursive function theory*, Thesis MIT, Cambridge, Mass.

[1991] C. Jockusch, Jr. and R. Soare, *Degrees of orderings not isomorphic to recursive linear orderings*, 'International Symposium on Mathematical Logic and its Applications (Nagoya, 1988)', Ann. Pure Appl. Logic **52**, 39-64

[1987] H. Kierstead, *On Π_1 automorphisms of recursive linear orders*, J. Symbolic Logic **52**, 681-688

[1986] M. Moses, *Relations intrinsically recursive in linear orders*, Zeitschr. f. math. Logik und Grundlagen d. Math. **32**, 467-472

[1988] M. Moses, *Decidable discrete linear orders*, J. Symbolic Logic **53**, 531-539

[1982] J. Rosenstein, *Linear orderings*, Academic Press, New York/London