

NONEXISTENCE OF MINIMAL PAIRS FOR GENERIC COMPUTABILITY

GREGORY IGUSA

ABSTRACT. A generic computation of a subset A of \mathbb{N} consists of a computation that correctly computes *most* of the bits of A , and which never incorrectly computes any bits of A , but which does not necessarily give an answer for every input. The motivation for this concept comes from group theory and complexity theory, but the purely recursion theoretic analysis proves to be interesting, and often counterintuitive. The primary result of this paper is that there are no minimal pairs for generic computability, answering a question of Jockusch and Schupp.

1. INTRODUCTION

In a recent paper, Jockusch and Schupp [1] introduce and analyze a number of notions of almost computability. Following their notation, we make the following definitions:

Definition 1.1. Let A be a subset of the natural numbers. Then A has *density 1* if the limit of the densities of its initial segments is 1, or in other words, if $\lim_{n \rightarrow \infty} \frac{|A \upharpoonright n|}{n} = 1$. In this case we will frequently say that A is *density-1*.

In this paper, a subset of the natural numbers is often referred to as a real. Note that the intersection of two reals is density-1 if and only if each of the reals is density-1. Binary branching trees and other countable sets will sometimes be referred to as reals, although the density of a real will never be mentioned unless it is explicitly described as a subset of \mathbb{N} .

Definition 1.2. A real A is *generically computable* if there exists a partial recursive function φ whose domain has density 1 such that if $\varphi(n) = 1$ then $n \in A$, and if $\varphi(n) = 0$ then $n \notin A$.

Definition 1.3. For reals A and B , A is *generically B -computable* if A is generically computable using B as an oracle. In this case, we frequently say B *generically computes* A .

Note that this notion of computation yields a binary relation on reals which is highly nontransitive, and which should not be confused by the transitive relation, *generic reducibility*, in which only an enumerably presented density-1 subset of the information in the oracle can be used. (Generic reducibility will be defined and discussed in more detail in Section 3.)

In fact, generic computation is so far from being transitive that any countable reflexive binary relation embeds in the reals under this relation. To prove this, we first make three observations:

Observation 1.4. *There exists a recursive reflexive binary relation R on \mathbb{N} such that for any reflexive binary relation R' with countable domain, R' is isomorphic to R restricted to some subset of \mathbb{N} .*

The relation R is simply the Fraisse limit of the finite reflexive binary relations. We also present a direct construction of R :

Proof. We build our relation R in stages. At the end of any stage, R will be defined on a finite initial segment of \mathbb{N} .

At stage 0, the domain of R is $\{0\}$, and $\langle 0, 0 \rangle \in R$

At stage $s + 1$, we extend R to be defined on the next 4^n many elements of \mathbb{N} , where n is the domain of R at the end of stage s . For every possible way to extend $R \upharpoonright n$ by one element, we take one of the new elements and define R on that element via the chosen extension. Every new element is related to itself (to maintain reflexivity), and no new element is related to any other new element. (Note that since the relationship is not symmetric, we need 4^n many new elements. For every old element a , we must choose whether a new element b satisfies aRb and also whether it satisfies bRa .)

This construction can clearly be carried out recursively. Any countable reflexive binary relation R' embeds in it by picking a counting $\langle a_i \rangle$ of the domain of R' , and mapping inductively each a_i to an element that was added to the domain of R at stage i . (By the construction of R , when the embedding has been defined on a_0, \dots, a_n , there exists a way to extend the embedding to map a_{n+1} to an element added at stage $n + 1$.) \square

Observation 1.5. *For any real A , there is another real $\mathcal{R}(A)$ such that A generically computes $\mathcal{R}(A)$ and A can be computed from any generic description of $\mathcal{R}(A)$.*

There are many different constructions of $\mathcal{R}(A)$, any of which will suffice for the purposes of this proposition. An example would be coding the entries of A into the columns of $\mathcal{R}(A)$, ie $n \in \mathcal{R}(A) \leftrightarrow$

$m \in A$, where 2^m is the largest power of 2 dividing n . A more thorough proof will be presented in Section 3, where the specifics of the definition of $\mathcal{R}(A)$ will become more important.

Observation 1.6. *There exists a countable sequence of reals $\langle X_i \rangle$ such that for each i , X_i cannot be computed by the recursive join of the rest of the X_j .*

This is satisfied by the columns of any arithmetically Cohen generic subset of $\mathbb{N} \times \mathbb{N}$.

With these three observations, we prove:

Proposition 1.7. *For any reflexive binary relation R , there exists a subset S of \mathbb{R} such that $\langle \mathbb{N}, R \rangle$ is isomorphic to S paired with relative generic computation.*

Proof. By Observation 1.4, it suffices to prove the proposition for recursive relations.

To prove this, we first define the asymmetric join of two reals in the following way. Fix any recursive partition of \mathbb{N} into two parts, a density-0 part, S , and an infinite density-1 part, L . In particular, let S be the powers of 2 and L be the numbers that are not powers of 2. Then the asymmetric join of A and B is equal to $\mathcal{R}(A)$ on L , and has B coded in to S (2^n is in the asymmetric join of A and B iff $n \in B$).

Note then that the asymmetric join of A and B has the same Turing degree as the usual join (it computes B , and it computes a density-1 subset of $\mathcal{R}(A)$). However, the ability to generically compute this asymmetric join is equivalent to the ability to compute A . This is because a density-1 subset of the bits of the generic join must include a density-1 subset of the bits of $\mathcal{R}(A)$, and A can be computed from any density-1 subset of the bits of $\mathcal{R}(A)$. Conversely, if one could compute A , then one could compute $\mathcal{R}(A)$, and therefore one could compute the bits of the asymmetric join on all of L , yielding a generic computation of the asymmetric join of A and B .

To conclude the proof, fix a recursive relation R . Choose $\langle X_i \rangle$ as in Observation 1.6, and for each i , define Y_i to be the recursive join of the set of X_j such that $R(i, j)$. (More precisely $Y_i = \{ \langle n, j \rangle \mid n \in X_j \wedge R(i, j) \}$.) Let Z_i be the asymmetric join of X_i and Y_i .

If $R(i, j)$ then $Y_i \geq_T X_j$, so $Z_i \geq_T X_j$, so in particular we have that Z_i generically computes Z_j . Likewise, if $\neg R(i, j)$ then Z_i is computable from the join of the X_k with $k \neq j$, so in particular, $Z_i \not\geq_T X_j$, so Z_i does not generically compute Z_j .

□

2. NO MINIMAL PAIR

Despite this fact, however, generic computation admits a worthwhile analysis in terms of what can be computed from a real. Certainly, if $X \geq_T Y$, then the set of things that X generically computes contains the set of things Y generically computes, but the lack of transitivity does not allow one to draw the obvious conclusions one would like to draw. Indeed, it turns out that the reals have the recursion theoretically counterintuitive property that there are no minimal pairs for generic computation:

Theorem 2.1. *For any nonrecursive reals A and B (and thus for any non-generically-computable reals A and B), there exists a real C such that C is not generically computable, but such that C is both generically A -computable and generically B -computable.*

The method for proving this statement will be to prove that for any nonrecursive reals A and B , there are density-1 subsets of \mathbb{N} recursive in A and B such that the union of the two subsets has no density-1 r.e. subset. Then C will be the union of those two subsets. This will suffice by the following lemma:

Lemma 2.2. *Let X be a density-1 subset of \mathbb{N} . Then for any real A , A can generically compute X if and only if A can enumerate a density-1 subset of X .*

Proof. If A can enumerate a density-1 subset Y of X , then A generically computes X via the partial function $\varphi(n) = 1$ iff $n \in Y$. The domain of this function is Y , which has density 1, and all the information it gives about X is correct.

Conversely, if A generically computes X , choose φ such that φ^A is a generic computation of X . Then let $Y = \{n \mid \varphi^A(n) = 1\}$. Y is clearly enumerable from A . Also, Y is a subset of X because a generic computation is not allowed to give incorrect answers. Finally Y is density-1, because it is the intersection of two density-1 sets (X , and the domain of φ^A). □

The proof of Theorem 2.1 will be comprised of three parts:

First, we prove Proposition 2.5, that if neither A nor B is Δ_2^0 , then A and B do not form a minimal pair for generic computability. Then, with the help of a technical lemma, we generalize this proof to prove Propositions 2.7 and 2.8, the corresponding results for when one, or both of them are Δ_2^0 . Proposition 2.8 is already proved in a paper of

Downey, Jockusch, and Schupp, but the technical lemma that we use to prove Proposition 2.7 proves Proposition 2.8 as well.

We begin by introducing some terminology that will be used for the proofs. Let $P_i = \{n \in \mathbb{N} \mid 2^i \leq n < 2^{i+1}\}$. Note then that \mathbb{N} is the disjoint union of the P_i together with $\{0\}$. For $X \subseteq \mathbb{N}$, we say that X has a gap of size 2^{-e} at P_i if the last 2^{i-e} many elements of P_i are not elements of X . Note then the following lemma:

Lemma 2.3. *If the only elements missing from X are from gaps of the form just described, then X is density-1 if and only if for every e , X has only finitely many gaps of size 2^{-e}*

Proof. If X has a gap of size 2^{-e} at P_i then $\frac{|X \cap 2^i|}{2^i} \leq 1 - 2^{-e-1}$, so in particular, if X has infinitely many gaps of size 2^{-e} , it does not have density 1.

Conversely, if there is some i such that after P_i , all of the gaps in X have size $\leq 2^{-e}$, then for $n \geq 2^{i+1}$, the density of $X \upharpoonright n$ will always be $\geq 1 - 2^{-e+1}$. (Note that since the gaps appear at the ends of the P_i , the local minima of the density of X always occur at the end of a P_i .) If for every e , there exists such an i , then the limiting density of X will be 1. \square

Before proving any of the propositions, we prove a result that is a direct corollary of the main theorem, and that also will not be used in the proof of the theorem. The proof is short though, and the methods generalize to prove Propositions 2.5, 2.7, and 2.8.

Proposition 2.4. *For any nonrecursive real A , there is a density-1 set φ^A that is recursive in A such that φ^A has no density-1 r.e. subset.*

First, a brief overview: we will define a total Turing functional φ on 2^ω . For each e , there will be a strategy that diagonalizes against W_e being a density-1 subset of φ^X for any real X . In doing so, the e th strategy will cause at most one real X_e to have the property that φ^{X_e} is not density-1. X_e will be the leftmost path through a recursive tree T_e , so repeating the same argument with rightmost paths gives a pair of functionals φ and ψ such that for any nonrecursive X either φ^X or ψ^X is density-1.

Proof. Over the course of the construction, after stage s , $\varphi^X \upharpoonright 2^s$ will be defined for every X , using at most the first s bits of X . This will be useful in verifying that the strategies work as they are supposed to.

The e th strategy acts as follows:

Define T_e as the tree whose infinite paths consist of the reals X such that $W_e \subseteq \varphi^X$. Note that T_e is a recursive tree, since we can determine the l th level of T_e in the following way. Run the enumeration of W_e for the first l steps. For every n less than 2^l , if W_e has enumerated n , remove any σ of length l such that $\varphi^\sigma(n) = 0$. As long as l is less than the stage s of the construction, this can be accomplished recursively. Let $T_{e,s}$ be the tree consisting of all extensions of the $(s-1)$ th level of T_e .

At stage s , if $s < e$, do nothing. Also, if $T_{e,s}$ is finite, do nothing. Else, place a marker p_s on the leftmost infinite path of $T_{e,s}$, at the shortest σ on that path such that σ has no marker. Then define $\varphi^X \upharpoonright P_s$ for all X by placing a gap of size 2^{-e} into φ^X at P_s if $\sigma \prec X$, and by not placing such a gap if $\sigma \not\prec X$.

The idea here is that the marker p_s signifies the existence of a trap at P_s : W_e must either avoid enumerating any of the elements of the gap at P_s , thereby creating another instance of its density dropping below $1 - 2^{e+1}$, or it must enumerate some of those elements, thereby removing all extensions of σ from the tree T_e .

Note then that if T_e is finite, then for every X , φ^X has only finitely many gaps of size 2^{-e} . Furthermore, in this case, the strategy has guaranteed that $W_e \not\subseteq \varphi^X$ for any X . (After the stage at which the tree is seen to be finite, the e th strategy stops acting. This stage is precisely the point at which, for every X , W_e has enumerated something not in φ^X .)

On the other hand, if T_e is infinite, then the leftmost path, X_e , of T_e has infinitely many markers on it, so φ^{X_e} has infinitely many gaps of size 2^{-e} so in particular, W_e is not density-1. ($W_e \subset \varphi^X$ for every infinite path X through T_e . This is because of how T_e is defined.) Furthermore, if $X \neq X_e$ then X has only finitely many markers on it, so φ^X has only finitely many gaps of size 2^{-e} (If $X \neq X_e$ then there is some stage s and some $\sigma \prec X$ such that after stage s , σ never looks like it might be an initial segment of the leftmost path of T_e , so after that stage s , X can only get at most $|\sigma|$ many markers placed on it)

Finally, note that the strategies have no need to interact: they ignore each other's markers and trees, and at state s , at most $s+1$ many strategies are eligible to act, and φ^X gets defined on P_s for every X by just defining $\varphi(X)$ to be the intersection of the sets that each strategy wants φ^X to be. By Lemma 2.3, a real X will have the property that φ^X is not density-1 if and only if some specific strategy causes $\varphi(X)$ to not be density-1. Thus, the only reals X such that φ^X is not density-1 are the X_e .

As mentioned in the overview, repeating the same construction again with rightmost paths will finish the proof, since if X is the leftmost path of one recursive tree and the rightmost path of another, then X is recursive. If this is not the case, then for one of the two constructions, the set computed from X is density-1 and has no density-1 r.e. subset. \square

Next, we proceed to prove Proposition 2.5. The proof is effectively the same as the proof of Proposition 2.4, with the only major modification being that we define two functionals simultaneously, and replace T_e with a 4-ary branching tree whose paths correspond to pairs of reals $\langle X, Y \rangle$ such that $\varphi^X \cup \psi^Y \subset W_e$.

Proposition 2.5. *If A and B form a minimal pair for generic computability, then either A or B is Δ_2^0 .*

Proof. Again, we describe how the e th strategy acts at stage s :

If $s < e$, do nothing. Otherwise define $T_{e,s}$ as the set of pairs $\langle \sigma, \tau \rangle$ such that $|\sigma| = |\tau|$ and such that $W_{e,s} \upharpoonright 2^{s-1} \subset \varphi_{s-1}^X \cup \psi_{s-1}^Y$ for some $X \succ \sigma$, and some $Y \succ \tau$. Note that this will be recursive, since we define φ and ψ applied to P_s by the end of stage s .

Then, if $T_{e,s}$ is finite, do nothing. Otherwise put a marker p_s on the leftmost infinite path of $T_{e,s}$, at the shortest pair $\langle \sigma, \tau \rangle$ on that path such that $\langle \sigma, \tau \rangle$ has no marker. Then define $\varphi_s^X \upharpoonright P_s$ and $\psi_s^Y \upharpoonright P_s$ for all X and Y by placing a gap of size 2^{-e} into φ_s^X at P_s if $\sigma \prec X$, and by not placing such a gap if $\sigma \not\prec X$, and likewise placing a gap of size 2^{-e} into ψ_s^Y at P_s if and only if $\tau \prec Y$.

Note, as before, that only one path T_e gets infinitely many markers on it, so in particular only finitely many markers are placed on nodes that are not on that path. That path corresponds to a pair of reals $\langle X_e, Y_e \rangle$, and if $X \neq X_e$ then φ^X will have only finitely many gaps of size 2^{-e} . Note also that the leftmost path of T_e computes both X_e and Y_e , so in particular, both are Δ_2^0 , but it is not true that either is necessarily the leftmost path of a recursive tree, so we are unable to use the previous trick to get them to be recursive.

Again, the strategies do not interfere with each other, and so if A is different from all of the X_e and B is different from all of the Y_e then φ^A is density-1, ψ^B is density-1, and $\varphi^A \cup \psi^B$ has no density-1 r.e. subset. By the comment after the statement of Theorem 2.1, this suffices. \square

Now we prove our technical lemma, which states that the “leftmost path” in the above construction can be replaced by any uniformly chosen Δ_2^0 infinite path through T_e .

Lemma 2.6. *Let \mathcal{F} be any function from reals to reals such that for a 4-ary branching tree T , if T is infinite, then $\mathcal{F}(T)$ is an infinite path through T , and such that $\mathcal{F}(T)$ is uniformly recursive in T' . Then for any reals A and B , one of the following three things holds.*

- 1: *A and B do not form a minimal pair for generic computability.*
- 2: *There exists a recursive tree T such that $\mathcal{F}(T) \geq_T A$.*
- 3: *There exists a recursive tree T such that $\mathcal{F}(T) \geq_T B$.*

So, for example, letting \mathcal{F} be the function corresponding to the construction in the proof of the low basis theorem, we could prove that if A and B form a minimal pair for generic computability, then either A or B must be low.

Proof. We modify the proof of Proposition 2.5 by, for each e , choosing a Δ_2^0 index for $\mathcal{F}(T_e)$. This can be done uniformly by the fixed point theorem, since we can uniformly compute the trees T_e from the graphs of φ and ψ , and since \mathcal{F} is assumed to be uniform. (By the fixed point theorem, we may assume we have a fixed index for the graphs of the Turing functionals φ, ψ that we build. The graphs are recursive, as at stage s , $\varphi^X(n), \psi^Y(m)$ are defined for all X and Y and for all $n, m \leq 2^s$.)

Having chosen a Δ_2^0 index for each $\mathcal{F}(T_e)$, at each stage, instead of placing a marker on the shortest unmarked node of the leftmost infinite path of T_e , the e th strategy places a marker on the shortest unmarked node of the current approximation to $\mathcal{F}(T_e)$. We then proceed to place the corresponding gaps in φ and ψ in the usual way. The e th strategy does nothing if $T_{e,s}$ is finite.

Then, as before, if $T_{e,s}$ is infinite, then the only path that gets infinitely many markers is $\mathcal{F}(T_e)$. This is because for any n , after the first n bits of $\mathcal{F}(T_e)$ have stabilized to their final configuration, all future markers will be on extensions of $\mathcal{F}(T_e) \upharpoonright n$. In this case, W_e is not density-1, so the e th strategy succeeds. If $T_{e,s}$ is finite, then for any X and for any Y , $\varphi^X \cup \psi^Y \not\subseteq W_e$ as before.

Then, for each e , define X_e and Y_e as before, note that $\mathcal{F}(T_e)$ computes either of them, and note that φ^X and ψ^Y are both generic computations of the same non-generically computable real, as long as for every e , $X \neq X_e$ and $Y \neq Y_e$.

□

We now can prove Proposition 2.7 as a direct corollary of this lemma

Proposition 2.7. *If A is Δ_2^0 and B is not Δ_2^0 then A and B do not form a minimal pair for generic computability.*

Proof. For any infinite recursive tree T , and any nonrecursive Δ_2^0 set A , $0'$ can uniformly (in indices for T and A) compute an infinite path $Z \in [T]$ such that $Z \not\leq_T A$.

Apply Lemma 2.6 using the \mathcal{F} representing this computation, and note then that for any T , $\mathcal{F}(T) \not\leq A$ by construction. Also, $\mathcal{F}(T) \not\leq B$ because B is not Δ_2^0 , and $\mathcal{F}(T)$ is. Therefore, A and B do not form a minimal pair for generic computability. \square

Note that this same proof can be modified to prove Proposition 2.8, by simultaneously avoiding the cones above both A and B .

Proposition 2.8. (Downey, Jockusch, and Schupp) *If A and B are both Δ_2^0 then A and B do not form a minimal pair for generic computability.*

3. GENERIC REDUCIBILITY

Given the wildly nontransitive nature of generic computation, it seems natural to attempt to generalize it to some transitive notion of reducibility with the same basic properties. The most important properties that one might expect of it would be to preserve the definition of the generically computable sets, and in particular a generic reduction using a generically computable oracle should be the same as a generic computation. In this section, we introduce four notions of generic reducibility, prove that two of them are equivalent, and prove that at least two of them are distinct. We then analyze the implications of Theorem 2.1 towards these notions of generic reducibility.

Definition 3.1. A *generic description* of a real A is a set S of ordered pairs $\langle n, x \rangle$, with $n \in \mathbb{N}, x \in \{0, 1\}$, such that:

- if $\langle n, 0 \rangle \in S$ then $n \notin A$,
- if $\langle n, 1 \rangle \in S$ then $n \in A$,
- and $\{n \mid \exists x \langle n, x \rangle \in S\}$ is density-1.

It should be mentioned that this notation conflicts slightly with the notation of Jockusch and Schupp, in that they define a generic description as a partial function, and this would be the graph of a generic description, by their definition. For the purposes of generic reduction though, it is more useful to have “generic description” be defined as a set, since the output of a generic computation is a generic description, and so the input of a generic reduction should be a generic description. It is interesting to notice, though, that the output of a generic computation of A is more than just a generic description of A , in that it is an *enumeration* of a generic description of A . For this reason, we define:

Definition 3.2. A *time-dependent generic description* of a real A is a set S of ordered triples $\langle n, x, l \rangle$, with $n, l \in \mathbb{N}, x \in \{0, 1\}$, such that $\{\langle n, x \rangle \mid \exists l \langle n, x, l \rangle \in S\}$ is a generic description of A .

Then, B generically computes A if and only if B can enumerate a generic description of A , which is true if and only if B can compute a time-dependent generic description of A .

Jockusch and Schupp define generic reducibility via enumeration operators. The relation is basically the one that one might intuitively construct, using only densely much information from the oracle, A , to deduce a generic computation of B . For those familiar with the notation, the definition is as follows:

Definition 3.3. A *generic reduction* of B from A is an enumeration operator which, given any generic description of A as input, outputs a generic description of B . B is *generically reducible to A* if there exists a generic reduction of B from A . In this case, we write $A \geq_G B$.

For those unfamiliar with the notation,

Definition 3.4. An *enumeration operator* is an r.e. set W of codes for pairs $\langle n, D \rangle$ where $n \in \mathbb{N}$ and D is a code for a finite subset of \mathbb{N} . It is thought of as a function, sending a real A to the set $\{n \mid \exists D [D \subseteq A \wedge \langle n, D \rangle \in W]\}$.

It should be noted that generic reduction has the following features: The computation of B from A must be uniform in the generic description of A , and the computation is only allowed to reference which sets are contained in the graph of the input set when computing a generic description for the output set (so in particular, giving less information about A never results in more information about B , and information is not allowed to be deduced from the rate/order of enumeration of the graph of A).

With these comments in mind, we make the following definitions:

Definition 3.5. A *time-dependent generic reduction* of B from A is a Turing functional which, given any time-dependent generic description of A as input, outputs a time-dependent generic description of B . B is *time-dependently generically reducible to A* if there exists a time-dependent generic reduction of B from A .

Definition 3.6. B is *non-uniformly generically reducible to A* if for every generic description of A , there is an enumeration operator which outputs a generic description of B using the given generic description of A as input.

Definition 3.7. B is *non-uniformly time-dependently generically reducible to A* if every time-dependent generic description of A , can compute a time-dependent generic description of B .

Again, we mention that the ability to compute a time-dependent generic description of a set is equivalent to the ability to enumerate a generic description of the set, and so the difference between the time-dependent and non-time-dependent reductions is entirely a difference in terms of what the input of the reduction is. The outputs are phrased differently just to make transitivity obvious, and also to make it easier to work with any given form of reduction on its own. Also for the remainder of this paper, whenever a non-time-dependent generic description is used as an oracle, only positive information about elements in the description will be used for the computation.

From the definitions, we may immediately conclude the following implications.

Observation 3.8. *The existence of a uniform reduction of either type implies the existence of a non-uniform reduction of the corresponding type.*

Observation 3.9. *The existence of a non-time-dependent reduction of either type implies the existence of a time-dependent reduction of either type.*

Observation 3.8 is trivially true. Observation 3.9 is true since from any time-dependent generic description, one can enumerate a generic description, and then simply ignore the order in which the elements were enumerated. The rules of enumeration operators do not allow for an obvious converse to this, although the first proposition that we prove is that the converse of does hold for the uniform generic reductions, and so in particular, the two uniform reductions are equivalent.

Proposition 3.10. $A \geq_G B$ if and only if the following holds:

There is a Turing functional φ such that for any time-dependent generic description X of A , φ^X is a generic computation of B .

By Observation 3.9, we need only prove that the second implies the first.

Proof. Assume that from every time-dependent generic description X of A , φ^X is a generic computation of B . Then in particular, there are no time-dependent generic descriptions X such that φ^X gives false information about B . So to generically reduce B to A , one needs only to enumerate everything that φ would enumerate, given any ordering of the generic description.

In other words, let W be the set of all $\langle a, D \rangle$ such that a codes an ordered pair $\langle x, y \rangle$, D is a finite set of ordered pairs $\langle n_i, m_i \rangle, i \leq c$, and such that there exists a sequence $\langle l_i \mid i \leq c \rangle$ where φ^X gives output y on input x for some X extending $\{\langle \langle n_i, m_i \rangle, l_i \rangle \mid i \leq c \}$.

Then, for any generic description of A , the output of W will be the union of all outputs of φ^X for any time-dependent versions of that generic description, and in particular, will be a generic description of B . \square

Next, we show that neither of the nonuniform reductions is equivalent to the uniform reduction. To prove this, we introduce some notation. Recall from Section 1, the definition of $\mathcal{R}(X)$:

Definition 3.11. For any real X , $n \in \mathcal{R}(X) \leftrightarrow m \in X$, where 2^m is the largest power of 2 dividing n .

Note now the following strengthening of Observation 1.5, proved by Jockusch and Schupp [1], but with a proof included for completeness:

Observation 3.12. *For any real X , X computes $\mathcal{R}(X)$ uniformly and X can be computed uniformly from any generic description of $\mathcal{R}(X)$. Therefore, the map sending $X \mapsto \mathcal{R}(X)$ induces an embedding from the Turing degrees to the generic degrees.*

Proof. X computes $\mathcal{R}(X)$ uniformly, and so generically computes $\mathcal{R}(X)$ uniformly.

Conversely, to compute the m th bit of X from a generic description of $\mathcal{R}(X)$, search for any n such that 2^m is the largest power of 2 dividing n and such that the generic description of $\mathcal{R}(X)$ has a value for the n th bit of $\mathcal{R}(X)$. Use that as the value for the m th bit of X . There must be such an n because the set of numbers divisible by 2^m and not by 2^{m+1} has positive density (in fact, has density $\frac{1}{2^{m+1}}$), and the generic description has values for density-1 many bits of $\mathcal{R}(X)$.

The proof of the embedding follows directly: If X computes Y then any generic description of $\mathcal{R}(X)$ can be used uniformly to compute X and therefore Y and therefore $\mathcal{R}(Y)$. Likewise, if $\mathcal{R}(X) \geq_G \mathcal{R}(Y)$, then X can compute $\mathcal{R}(X)$, which can generically compute $\mathcal{R}(Y)$. Y can then be recovered from this generic description. \square

Note that this observation and proof hold for any of the forms of generic reduction.

We now introduce an alternate definition, which would have sufficed for the purposes of Observation 1.5, but with the property that one cannot uniformly recover X from a generic description of $\tilde{\mathcal{R}}(X)$.

Definition 3.13. For any real X , $n \in \tilde{\mathcal{R}}(X) \leftrightarrow m \in X$, where 2^m is the largest power of 2 less than n .

The key distinction here is that $\mathcal{R}(X)$ codes each of the entries of X into a positive density set, and so any generic description of $\mathcal{R}(X)$ must be able to recover all the entries of X , while $n \in \tilde{\mathcal{R}}(X) \leftrightarrow m \in X$ codes the entries of X into progressively larger sets, each finite, and so any generic description of $n \in \tilde{\mathcal{R}}(X) \leftrightarrow m \in X$ must be able to recover all but finitely many of the entries of X .

Proposition 3.14. *Let A be a real such that one cannot uniformly compute A from an arbitrary cofinite subset of the entries of A . Then $\mathcal{R}(A)$ is non-uniformly generically equivalent to $\tilde{\mathcal{R}}(A)$ (and therefore non-uniformly time-dependently generically equivalent), but $\tilde{\mathcal{R}}(A) \not\leq_G \mathcal{R}(A)$. Therefore, neither of the non-uniform notions of generic reducibility is equivalent to the uniform notion.*

Note that there exists reals satisfying the hypothesis of the proposition, for example any 1-random real, or any arithmetically Cohen generic real.

Proof. First of all, any generic description of $\mathcal{R}(A)$ can be used uniformly to recover A by Observation 3.12. It can therefore compute $\tilde{\mathcal{R}}(A)$. Likewise, from any generic description of $\tilde{\mathcal{R}}(A)$, one can uniformly compute all but finitely many bits of A . From this, one can non-uniformly compute A , and therefore compute $\mathcal{R}(A)$. However, this cannot be done uniformly, as follows:

Any cofinite subset of the entries of A can be used to uniformly compute a generic description of $\tilde{\mathcal{R}}(A)$. Any generic description of $\mathcal{R}(A)$ can be used to uniformly compute A . Thus, since there is no uniform way to compute A from a cofinite subset of its entries, there is no uniform way to go from a generic description of $\tilde{\mathcal{R}}(A)$ to a generic description of $\mathcal{R}(A)$. \square

As yet, however, there do not appear to be any theorems that have been proved for one form of generic reduction that have not also been proved for every other form. Indeed, for the remainder of this paper generic reduction will refer to any of the definitions.

Theorem 2.1 sheds very little light on the question of the existence of a minimal pair in the generic degrees. It does, however, strengthen a previous result of Jockusch and Schupp. Borrowing a term from the study of enumeration reducibility, we make the following definition:

Definition 3.15. A generic degree \mathbf{a} is *quasi-minimal* if \mathbf{a} is nonzero, and if for every nonrecursive real X , the generic degree of $\mathcal{R}(X)$ is not below \mathbf{a} .

Then, while proving that the embedding from the Turing degrees to the generic degrees is not surjective, Jockusch and Schupp [1] actually prove the stronger result that there exists a quasi-minimal generic degree. Theorem 2.1 allows us to strengthen this result:

Proposition 3.16. *For every nonzero generic degree \mathbf{a} , there is a quasi-minimal generic degree \mathbf{b} such that $\mathbf{a} \geq_G \mathbf{b}$.*

Proof. Let \mathbf{a} be a nonzero generic degree. If \mathbf{a} is quasi-minimal, then we are done. Else, choose A nonrecursive, so that the generic degree of $\mathcal{R}(A)$ is below \mathbf{a} . In the Turing degrees, every nonzero degree is half of a minimal pair, so choose B such that A and B form a minimal pair for Turing reducibility. By Theorem 2.1, choose some C , not generically computable, such that A and B can both generically compute C .

Then C generically reduces to both $\mathcal{R}(A)$, and $\mathcal{R}(B)$, and the generic degree of C must be quasi-minimal, since if there were some D such that $C \geq_G \mathcal{R}(D)$, then we would have $\mathcal{R}(A) \geq_G \mathcal{R}(D)$ and also $\mathcal{R}(B) \geq_G \mathcal{R}(D)$, and therefore $A \geq_T D$ and also $B \geq_T D$, and so D would have to be recursive, since A and B form a minimal pair. Let \mathbf{b} be the generic degree of C . \square

Unfortunately, it seems very difficult to generalize the methods of this paper to prove the lack of a minimal pair for generic reducibility, since if there is such a pair, then at least one of the two degrees in the pair would have to be quasi-minimal, and the construction in the proof of Theorem 2.1 involves each of A and B actually computing something which is a generic description of C , but the quasi-minimal generic degrees are unable to use a generic reduction to completely compute any nonrecursive real. However, Proposition 3.16 gives some hope of a different construction, since it ensures that if there exists a minimal pair for generic reducibility, then there exists one in which both halves of the minimal pair are quasi-minimal.

Working in the other direction, it would also be interesting to know whether two quasi-minimal generic degrees can join to a generic degree that is not quasi-minimal.

REFERENCES

- [1] Carl Jockusch and Paul Schupp, *Generic computability, Turing degrees, and asymptotic density*, to appear in *Journal of the London Mathematical Society*.