LOW LEVEL NONDEFINABILITY RESULTS: DOMINATION AND RECURSIVE ENUMERATION

MINGZHONG CAI AND RICHARD A. SHORE

Abstract. We study low level nondefinability in the Turing degrees. We prove a variety of results, including for example, that being array nonrecursive is not definable by a Σ_1 or Π_1 formula in the language $(\leq, REA)$ where REA stands for the "r.e. in and above" predicate. In contrast, this property is definable by a Π_2 formula in this language. We also show that the Σ_1-theory of $(\mathcal{D}, \leq, REA)$ is decidable.

1. Introduction

A major focus of recent research on the Turing degrees, $\mathcal{D}$, has been definability, which is also a major focus of mathematical logic in general. We discuss a topic that has received almost no attention but certainly deserves some: nondefinability in the Turing degrees.

This work was motivated by the suggestion of Miller and Martin [MM] that one could prove that the classes **HI** and **HIF** of hyperimmune and hyperimmune-free degrees, respectively, are not simply definable. (A degree $\mathbf{x} \in \mathbf{HIF}$ if and only if every $f \leq_T X$ is dominated by a recursive function. The class **HI** consists of the degrees not in **HIF**.) In addition to the basic language $(\leq)$, they suggest two extensions augmenting it by either the jump operator $'$ or the relation RE for "recursively enumerable in." They also allow parameters $\bar{\mathbf{c}}$ for any specific degrees in all their languages. They prove that **HI** is not definable by a quantifier free formula in the language $(\leq, \bar{\mathbf{c}})$ (for any degrees $\bar{\mathbf{c}}$) and conjecture that this is also true for the language with jump, $(\leq,', \bar{\mathbf{c}})$. We affirm their conjecture and prove other similar theorems about these and other sets of, and relations on, degrees.

In particular, we are interested in another complementary pair of sets of degrees related to domination properties: the array nonrecursive degrees, **ANR**, that compute a function not dominated by the modulus function for $0'$, and the complementary class **AR** of array recursive degrees. In place of the relation RE, we instead study the relation REA, "recursively enumerable in and above". Note that while REA is clearly definable from RE by a quantifier free formula, we show in §6 that the reverse is not true even by a one quantifier formula (without parameters). In addition, our intuitions about relativizing the relation RE really apply to REA and all the examples we know using RE actually only use REA. Most striking among these is the two quantifier definition

---

Date: January 25, 2013.
1991 *Mathematics Subject Classification.* 03D28.
The first author was partially supported by NSF Grant DMS-0852811.
The second author was partially supported by NSF Grants DMS-0852811 and DMS-1161175.

in this language (without parameters) of **ANR** as $\{\mathbf{x}|(\forall\mathbf{y}\geq\mathbf{x})(\exists\mathbf{z}<\mathbf{y})(\mathbf{y}\text{ REA }\mathbf{z})\}$ ([CS]; [C2]).

In studying these low level nondefinability results, we can also push the quantifier bound up in searching for possible definitions of these degree classes. For example, we still do not know whether **HI** is definable in $\mathcal{D}$ (even augmented with the REA predicate and parameters), and our results show that to define **HI** we need at least a two-quantifier formula with parameters or a three-quantifier formula without parameters. One might hope then that the obstacles we meet in proving that **HI** is not definable by formulae of a certain complexity can help suggest possible definitions of **HI**.

The definability of the predicates RE and REA is also a major open question in the field. In particular, we want to see how these two predicates are related to the partial order (Turing reducibility). Our work on the REA relation (Lemma 3.3 and Theorem 4.2) suggests that the characterization of this relation as given by the modulus lemma (recalled in the proof of Lemma 3.3 below) is an important technique in studying it.

While some of our results allow parameters, most do not. Including parameters makes many more classes of degrees easily definable. Classic examples include the degrees of the productive sets: $\{\mathbf{x}|\mathbf{x}\geq\mathbf{0}'\}$; the immune sets: $\{\mathbf{x}|\mathbf{x}\geq\mathbf{0}\}$; the functions dominating every recursive function: $\{\mathbf{x}|\mathbf{x}'\geq\mathbf{0}''\}$ and the degrees of arithmetic sets (by the Exact Pair Theorem [Sp]). However, unbridled use of parameters makes everything possible definable as by Slaman and Woodin [SW, Corollary 5.6], the biinterpretability conjecture holds for $\mathcal{D}$ with parameters and so every relation on $\mathcal{D}$ definable in second order arithmetic is definable from parameters (although their proof does not produce any simple definitions). In fact, they show that there is a single degree $\mathbf{c}$ such that every relation definable in second order arithmetic is definable from $\mathbf{c}$.

The table below summarizes our current results listing classes and relations not definable by formulas in specific classes in one of the three languages indicated with or without parameters ($\bar{\mathbf{c}}$).

Class/Relation	Not defined by formula in	Language	see Section(s)
HI	Σ_0	$(\leq,', \bar{\mathbf{c}})$	7
HI	Σ_1	$(\leq, \bar{\mathbf{c}})$	8
HI	Σ_2, Π_2	$(\leq)$	9
ANR	Σ_0	$(\leq,', \bar{\mathbf{c}})$	7
ANR	Σ_1	$(\leq, \bar{\mathbf{c}})$	8
ANR	Σ_1, Π_1	$(\leq, REA)$	5, 3
ANR	Σ_2	$(\leq)$	9
HIF	Σ_1	$(\leq, REA, \bar{\mathbf{c}})$	3
AR	Σ_1	$(\leq, REA, \bar{\mathbf{c}})$	3
Jump	Σ_1	$(\leq, REA)$	5
RE	Σ_1, Π_1	$(\leq, REA)$	6
Arith	Σ_1, Π_1	$(\leq, REA)$	5
Arith	Σ_1, Π_1	$(\leq,')$	9
Arith	Σ_2, Π_2	$(\leq)$	9

An anonymous referee has pointed out that our construction in Section 3 actually leads to a stronger result that the classes **AR** and **HIF** are not Σ_1 definable in the language $(\leq,', REA, \bar{\mathbf{c}})$, and so results in Section 7 follows as corollaries. See details in the end of Section 3.

We note the particularly tight instances from this table. **ANR** is definable (as noted above) by a two quantifier formula in $(\leq, REA)$ but not by any one quantifier one. It is also not definable by a Σ_2 formula in just $(\leq)$. The relation $\mathbf{y} = \mathbf{x}'$ is definable in $(\leq, REA)$ by a Π_1 formula $(\mathbf{y}\text{ REA }\mathbf{x} \wedge \forall \mathbf{z}(\mathbf{z}\text{ REA }\mathbf{x} \rightarrow \mathbf{z} \leq \mathbf{y}))$ but not by a Σ_1 formula in this language. **Arith**, the class of degrees of arithmetic sets, is definable by a Σ_3 formula in the language with just $\leq$. Rewrite [JS, Theorem 3.3] to eliminate the join:

$$\mathbf{Arith} = \{\mathbf{x} | (\exists \mathbf{y} \geq \mathbf{x})(\forall \mathbf{z} | \mathbf{y}\ \forall \mathbf{w}(\mathbf{w} \geq \mathbf{z}, \mathbf{y} \rightarrow \exists \mathbf{u}(\mathbf{w} > \mathbf{u} > \mathbf{z}))\}.$$

On the other hand, it does not have a Σ_2 or Π_2 definition in this language.

With REA added to the language, **Arith** is Π_2 definable as

$$\{\mathbf{x} | [\forall \mathbf{a}, \mathbf{b}(\forall \mathbf{c}, \mathbf{d}((\mathbf{c} < \mathbf{a}, \mathbf{b}) \wedge (\mathbf{d}\text{ REA }\mathbf{c}) \rightarrow (\mathbf{d} < \mathbf{a}, \mathbf{b})) \rightarrow \mathbf{x} < \mathbf{a}, \mathbf{b})]\}.$$

However, it is not 1-quantifier definable even with REA in the language. In $(\leq,')$, a similar formula saying that **Arith** is the least ideal closed under jump shows that it is Π_2 definable but again it is not 1-quantifier definable.

All the analyses proceed by first finding an appropriate syntactic normal form for formulas of the class being analyzed. Typically a second step is to simplify the form by finding degrees in the class of degrees being considered with special properties such as avoiding cones determined by the parameters, being minimal or perhaps also r.e. in some other degree (or not). Then one needs to do a construction to show that whatever diagram provided by the witnesses that the special degrees are in the class can be duplicated with corresponding degrees outside it. This then shows that the formula cannot define the class being considered. The techniques for the two quantifier results with just $\leq$ depend on standard initial segment and extension of embedding results and follow similar ones for other classes in Lerman and Shore [LS] and Shore [S]. The constructions not involving REA are at the moment mostly ad hoc exploiting various forcing constructions and special facts. Most of our results involving REA depend on variations on a basic theorem providing a decision procedure for one quantifier formulas in the language $(\leq, REA)$ (see Section 4 and Theorem 4.2).

The first half of this paper focuses on the nondefinability results related to languages with the REA predicate, and the second half consists of miscellaneous results on languages without it. We finish with some remarks and open questions.

2. Basic Notions

Our notions of sets, functions, strings and trees are standard.

In the proofs below we follow the same type of argument for several Σ_1 level nondefinability results. For example, to show that **AR** is not definable by a Σ_1 formula $\exists \bar{d}\theta(x, \bar{d}, \bar{c})$ (where the c's are parameters), we assume that there is such a definition and derive a contradiction. First, we construct an AR degree $\mathbf{x}$ (with some extra properties) and, by the assumed definition, have degrees $\bar{\mathbf{d}}$ which are witness for the given Σ_1 formula. Now we produce an ANR degree $\mathbf{y}$ and some $\bar{\mathbf{d}}^*$ such that they satisfy the same atomic diagram with the parameters $\bar{\mathbf{c}}$ as do $\mathbf{x}$ and $\bar{\mathbf{d}}$. This gives us the desired contradiction. (The details are provided in the constructions below). We call the given degrees *originals* and the new ones *photocopies* of the corresponding originals. We also use the following notation: for each original degree $\mathbf{z}$, $\mathbf{z}^*$ denotes the photocopy of $\mathbf{z}$ (for example, $\mathbf{x}^* = \mathbf{y}$ in the above setting).

In our constructions involving REA, we use the following limit-modulus characterization of being REA. It is the relativization of the standard modulus lemma (see [So, Corollary III.3.4]) and is important in our analyses of the REA relation.

Lemma 2.1. *For $A \geq_T B$, the degree of A is REA the degree of B if and only if $A(x)$ has a limit approximation $g(x,s) \leq_T B$ with modulus function $m(x) \leq_T A$, i.e., $\lim_s g(x,s)$ exists and equals $A(x)$ and the approximation stops changing by $s = m(x)$.*

3. $\mathbf{AR}, \mathbf{HIF} \notin \mathbf{\Sigma}_1(\leq, REA, \bar{\mathbf{c}})$

We begin with the easier case of $\mathbf{HIF}$ not being definable by a $\mathbf{\Sigma}_1(\leq, REA, \bar{\mathbf{c}})$ formula.

Theorem 3.1. $\mathbf{HIF}$ *is not definable by a $\mathbf{\Sigma}_1(\leq, REA, \bar{\mathbf{c}})$ formula.*

Proof. Suppose, for the sake of a contradiction, that $\mathbf{HIF}$ is definable by $\exists \bar{d}\theta(x, \bar{d}, \bar{c})$, i.e., a degree $\mathbf{x}$ is HIF if and only if there are degrees $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$ such that $\theta(\mathbf{x}, \bar{\mathbf{c}}, \bar{\mathbf{d}})$ holds (where $\bar{\mathbf{c}}$ is a fixed sequence of degrees corresponding to the constant symbols $\bar{c}$). (For technical convenience we assume that one of the $\bar{\mathbf{c}}$ is $\mathbf{0}$.) Now we write θ in disjunctive normal form as $\psi_0 \vee \psi_1 \vee \cdots \vee \psi_k$. We can assume that each ψ_i is a complete diagram in all the degrees involved. For any particular HIF $\mathbf{x}$, one of these disjuncts is satisfied and we will consider it for an $\mathbf{x}$ of our choosing. Our plan is to construct a $\mathbf{y}$ and some $\mathbf{d}^*$'s that satisfy the same atomic diagram with the parameters $\mathbf{c}$'s as $\mathbf{x}$ and the $\mathbf{d}$'s do. For convenience we let X be the set of degrees consisting of $\mathbf{x}$ and the $\mathbf{d}$'s and $\mathbf{c}$'s.

We define the required photocopies by constructing a generic $\mathbf{g}$ and joining $\mathbf{g}$ with $\mathbf{x}$ to get $\mathbf{y} = \mathbf{x} \vee \mathbf{g}$ and $\mathbf{g}$ with some of the $\mathbf{d}_i$'s to get $\mathbf{d}_i^*$; other $\mathbf{d}_i$'s remain the same.

Now let us think about which of the $\mathbf{d}_i$'s need modification. Of course, we need to join the generic $\mathbf{g}$ to the $\mathbf{d}_i$'s above $\mathbf{x}$. We can choose our original $\mathbf{x}$ minimal (as there are minimal degrees in $\mathbf{HIF}$, see [Le, Chapter V]) and so we will not have to worry about degrees below $\mathbf{x}$. It is possible that some $\mathbf{d}_j$ above $\mathbf{x}$ is REA $\mathbf{d}_k$ while $\mathbf{d}_k \not\geq \mathbf{x}$. We also need to add $\mathbf{g}$ to $\mathbf{d}_k$ in this case. Now we must iterate this process. Let $Z \subset X$ be the subset of degrees generated from $\mathbf{x}$ which is closed upwards and closed under the REA relation (if $\mathbf{z} \in Z$ and $\mathbf{w} \geq \mathbf{z}$ or $\mathbf{z} REA \mathbf{w}$, then $\mathbf{w}$ is also in Z). This Z is the set of degrees we need to change.

Of course, we want to make sure that no parameter $\mathbf{c}$ is in Z (since we are not able to change them). By analyzing the definition of Z, it is easy to see that if some $\mathbf{c}$ is in Z then $\mathbf{x}$ is arithmetic in $\mathbf{c}$ (i.e., recursive in $\mathbf{c}^{(n)}$ for some $n \in \omega$). So we need to choose our original $\mathbf{x}$ not arithmetic in any of the parameters $\mathbf{c}$. Again we can do this since there are continuum many minimal HIF degrees.

So we start with a minimal HIF degree $\mathbf{x}$ (minimality here is only to guarantee that $\mathbf{x}$ is not REA in any degree below it) which is not arithmetic in any of the $\mathbf{c}$'s and so no $\mathbf{c}$ is in Z. Let $\mathbf{d}_i$ be the witnesses for the Σ_1 definition of $\mathbf{x} \in \mathbf{HIF}$. Now we build a generic G and let $\mathbf{g} = \deg(G)$. Define $\mathbf{y} = \mathbf{x} \vee \mathbf{g}$ and

$$\mathbf{d_i}^* = \begin{cases} \mathbf{d}_i \vee \mathbf{g}, & \text{if } \mathbf{d}_i \in Z; \\ \mathbf{d}_i, & \text{otherwise.} \end{cases}$$

Now we must show that $(\mathbf{y}, \bar{\mathbf{d}}^*, \bar{\mathbf{c}})$ satisfy the same atomic diagram as do $(\mathbf{x}, \bar{\mathbf{d}}, \bar{\mathbf{c}})$. It is easy to see that all positive relations are preserved by the construction. So we only need to take care of the negative relations. We need two lemmas.

Lemma 3.2. *If* $\mathbf{g}$ *is* 1*-generic in* $\mathbf{a} \vee \mathbf{b}$ *and* $\mathbf{a} \not\leq \mathbf{b}$*, then* $\mathbf{a} \not\leq \mathbf{b} \vee \mathbf{g}$ *(and so* $\mathbf{a} \vee \mathbf{g} \not\leq \mathbf{b} \vee \mathbf{g}$*.)*

Proof. This is a standard fact about 1-generics. □

Lemma 3.3. *If* $\mathbf{g}$ *is* 2*-generic in* $\mathbf{a} \vee \mathbf{b}$ *and* $\mathbf{a}$ *is not REA* $\mathbf{b}$*, then* $\mathbf{a} \vee \mathbf{g}$ *is not REA* $\mathbf{b} \vee \mathbf{g}$ *(and so* $\mathbf{a} \vee \mathbf{g}$ *is not REA* $\mathbf{b}$*).*

Proof. By Lemma 3.2, we can assume that $\mathbf{b} < \mathbf{a}$. By the same Lemma, we may also assume that $\mathbf{a} \leq \mathbf{b}'$ as being 2-generic in $\mathbf{a} \vee \mathbf{b}$ makes G 1-generic in $\mathbf{a} \vee \mathbf{b}'$ and so, if $\mathbf{a} \not\leq \mathbf{b}'$, $\mathbf{a} \not\leq \mathbf{b}' \vee \mathbf{g} =(\mathbf{b} \vee \mathbf{g})'$ as required (every 1-generic is GL_1). By the limit lemma then, we can fix a B-recursive approximation A_s to A.

By the limit-modulus characterization of being REA (see Lemma 2.1), we want to see that for any 2-generic G, there is no pair (Φ, Ψ) of reductions such that $\Phi^{B \oplus G}(x, s)$ is a limit approximation of $A \oplus G$ and $\Psi^{A \oplus G}(x)$ is a modulus function for this approximation. Consider then any (Φ, Ψ).

First, we can assume that the totality of Φ and Ψ is forced and so for any σ (a possible initial segment of G) and any input (x, s), there is always a $\tau \supset \sigma$ such that $\Phi^{B \oplus \tau}(x, s)$ and $\Psi^{A \oplus \tau}(x)$ both converge. (This is a two quantifier question and we consider only conditions σ extending the one forcing totality.)

Next, we can assume that we consider only conditions extending one forcing that $\Phi^{B \oplus G}$ gives a limit computation of $A \oplus G$ and that $\Psi^{A \oplus G}$ provides a modulus function for it. (These are again guaranteed by two quantifier statements: $\forall x, s \exists t > s(\Phi^{B \oplus G}(x, t) = (A \oplus G)(x)$ and $\forall x, v(\Psi^{A \oplus G}(x) = v \to \forall v' \geq v(\Phi^{B \oplus G}(x, v') = (A \oplus G)(x))$.) Thus if $\Phi^{B \oplus \tau'}(2x, v) \downarrow \neq A(x)$, then there is a $\tau'' \supseteq \tau'$ and a $v' > v$ such that $\Phi^{B \oplus \tau''}(2x, v') \downarrow = A(x)$ and if $\Psi^{A \oplus \tau}(x) \downarrow = v$ and $\Phi^{B \oplus \tau}(2x, v) \downarrow = a$ then $A(x) = a$.

Now we argue that in this situation we can produce a B-recursive limit computation of A with an A-recursive modulus function for the desired contradiction. Given x and s consider all $v, \tau \subseteq \tau', t$ (all bounded by s) such that $\Psi_s^{A_t \oplus \tau}(2x) = v$ and $\Phi_s^{B \oplus \tau'}(2x, v) = a$. If there are no such $v, \tau \subseteq \tau', t$ then our approximation says $A(x) = 0$. If every such quadruple $v, \tau \subseteq \tau', t$ gives the same value a then we say $A(x) = a$. If different $v, \tau \subseteq \tau', t$ give different values for a, then for each of them we look for a $\tau'' \supseteq \tau'$ and $v' \geq v$ such that $\Phi^{B \oplus \tau''}(2x, v') \neq \Phi^{B \oplus \tau'}(2x, v)$. By our assumptions, we must find such τ'' and v' for all $v, \tau \subseteq \tau', t$ with $\Phi^{B \oplus \tau'}(2x, v) \neq A(x)$. Thus we are left with only one seemingly viable possible value for $A(x)$ and we say that it is the value of our approximation at s.

We want to show that this procedure gives a correct limit computation of $A(x)$ for each x. Consider any $v, \tau \subseteq \tau'$ such that $\Psi^{A \oplus \tau}(2x) \downarrow = v$ and $\Phi^{B \oplus \tau'}(2x, v) \downarrow = a$ and any t such that A_t is correct on the use of the first computation and also larger than the number of steps needed for both computations to converge. This tuple $v, \tau \subseteq \tau'$

and t will be one we find at every $s \geq t$. Moreover, the computations of $\Psi_s^{A_t \oplus \tau}(2x) = v$ and $\Phi_s^{B \oplus \tau'}(2x, v) = a$ at s are correct computations from A and B. If there were any $\tau'' \supseteq \tau$ and $v' > v$ such that $\Phi^{B \oplus \tau''}(2x, v') \neq \Phi^{B \oplus \tau'}(2x, v)$ we would contradict our having forced $\Phi^{B \oplus G}$ to be a limit computation of $A \oplus G$ with modulus function $\Psi^{A \oplus G}$. Thus, our procedure can never eliminate the value a and so that is the limit value of our approximation. Again, by our assumptions, it is also the correct value of $A(x)$. Finally, it is clear that A (which, remember, computes B and so the approximation A_s as well) can find such a stage s and so a modulus function for our approximation. $\square$

Using Lemmas 3.2 and 3.3, it is now not difficult to argue case by case that negative relations are also preserved. Consider first two degrees $\mathbf{a} \not\leq \mathbf{b}$ in X. Their photocopies are either unchanged or gotten by joining with $\mathbf{g}$. It is obvious that $\mathbf{a} \vee \mathbf{g} \not\leq \mathbf{b}$. The other two cases follow from Lemma 3.2.

Now take two degrees $\mathbf{a}$ not REA $\mathbf{b}$. If $\mathbf{a}$ is not Turing above $\mathbf{b}$, then $\mathbf{a}^*$ is not Turing above $\mathbf{b}^*$ by the above argument, and so not REA $\mathbf{b}^*$. So we can assume that $\mathbf{a} > \mathbf{b}$. In our procedure, if we join $\mathbf{g}$ to $\mathbf{b}$ to get $\mathbf{b}^*$, then we also join $\mathbf{g}$ to $\mathbf{a}$ to get $\mathbf{a}^*$. That is, we don't need to consider the case $\mathbf{a}$ not REA $\mathbf{b} \vee \mathbf{g}$. The remaining two cases follow from Lemma 3.3.

By genericity $\mathbf{g}$ and so $\mathbf{y} = \mathbf{x} \vee \mathbf{g}$ is HI ([J2]) for the desired contradiction. $\square$

We can now note that this argument gives the analogous result for **AR**.

Theorem 3.4. **AR** *is not definable by a* $\mathbf{\Sigma}_1(\leq, REA, \bar{\mathbf{c}})$ *formula.*

Proof. We again begin with an AR minimal degree $\mathbf{x}$ which is not arithmetic in any of the parameters (using standard minimal degree construction with cone avoiding) and build the required photocopies as above. Now note that any 2-generic is ANR (the set of conditions making the principal function of the generic be larger (at n many inputs) than the modulus function for $0'$ are dense and Π_1^0). As **ANR** is, like **HI**, closed upward, our $\mathbf{y} = \mathbf{x} \vee \mathbf{g}$ is also ANR as required. $\square$

Remarks. It is natural to ask whether this last result holds with REA replaced by RE. In fact, it is actually not difficult to prove, in analogy with Lemma 3.3, that if $\mathbf{a}$ is not r.e. in $\mathbf{b}$ (not necessarily above), and $\mathbf{g}$ is 1-generic in $\mathbf{a} \vee \mathbf{b}$, then $\mathbf{a}$ is not r.e. in $\mathbf{b} \vee \mathbf{g}$ either. Indeed if A is not r.e. in B then $A \oplus G$ is not r.e. in $B \oplus G$. However, we do not know whether the degree $\mathbf{a} \vee \mathbf{g}$ is r.e. in $\mathbf{b} \vee \mathbf{g}$ or not.

In addition, as pointed out by an anonymous referee, based on the proofs above, we can actually show that **HIF** (or **AR**) is not definable by Σ_1 formula in the language of order, jump and REA, i.e., we can add the jump to the language.

First of all, we need to split the formulas regarding double (or higher) jumps into single ones. For example, $x'' = b$ is rewritten as $(x' = y) \wedge (y' = b)$ (with y bounded by an existential quantifier), so for our purpose we can regard the jump as a binary relation symbol.

The construction is essentially the same as in the proof above. We pick a sufficiently generic degree $\mathbf{g}$ and join it with the degrees in Z. Now we only need to show that both positive and negative "jump relations" are preserved.

Consider $\mathbf{a} = \mathbf{b}'$ as originals. If we join $\mathbf{g}$ to $\mathbf{a}$ in the construction, then we have to join $\mathbf{g}$ to $\mathbf{b}$ as well (since $\mathbf{a}$ is REA $\mathbf{b}$). Note that 1-generic degrees are $\mathbf{GL}_1$, and it easily relativizes to $(\mathbf{b} \vee \mathbf{g})' = \mathbf{b}' \vee \mathbf{g} = \mathbf{a} \vee \mathbf{g}$. Therefore the positive jump relations are preserved.

For the negative ones, by Lemmas 3.2 and 3.3, we only need to consider the case when $\mathbf{a}$ is REA $\mathbf{b}$ but $\mathbf{a}$ is not $\mathbf{b}'$. Now again by Lemma 3.2, $\mathbf{a} \vee \mathbf{g}$ is not above $\mathbf{b}'$ for any $\mathbf{g}$ which is 1-generic in $\mathbf{a} \vee \mathbf{b}'$, therefore $\mathbf{a} \vee \mathbf{g} \neq (\mathbf{b} \vee \mathbf{g})'$.

Our results in Section 7 now follow as a corollary. We nonetheless provide a separate proof there because it illustrates a different approach and directly answers Miller and Martin's original question.

4. Decidability of the Σ_1 theory of $(\mathcal{D}, \leq, REA)$

The construction in the previous section heavily uses the given partial order of the originals to preserve the positive relations. This idea cannot be used to solve similar problems for **HI** and **ANR** simply because both classes are upward closed and so, for example, there is no way to build a degree $\mathbf{y}$ above a given ANR $\mathbf{x}$ as its photocopy and have $\mathbf{y}$ be AR. So what we need to do is to build some photocopy $\mathbf{y}$ with some $\mathbf{d}^*$'s by some different construction and still make them look like the originals.

This is in general not easy: In addition to building such a $\mathbf{y}$, we also need to build some $\mathbf{d}^*$'s satisfying some given order and REA relations. Especially in the **ANR** case, we probably need to build some $\mathbf{d}^*$ below $\mathbf{y}$ such that $\mathbf{y}$ is REA $\mathbf{d}^*$. (Recall that every ANR degree is RRE, i.e. it is r.e. in some degree strictly below it [CS]). So as the first step towards the solution of our problem we try to investigate the embedding problem, i.e., which (finite) partial orders with some extra REA relations and their negations can be embedded into the Turing degrees $(\mathcal{D}, \leq, REA)$.

We begin by listing some obvious properties that such a partial order must satisfy. To simplify our analysis we take the REA relation to mean r.e. in and strictly above. This certainly entails no loss of generality as each version of the relation is definable from the other by a quantifier free formula.

Definition 4.1. An *REA partial order* (REA-PO) is a partial order $\leq$ with an extra binary relation REA satisfying the following two axioms:

Strict-Order: a REA $b \Rightarrow a > b$.
Transitivity: a REA $b \wedge a > c > b \Rightarrow a$ REA c.

It is easy to see that, if one takes a finite collection of Turing degrees and considers the REA relations among them, they form an REA-PO with the natural definitions of the order and REA relations. Not too surprisingly, the converse also holds (although the proof is not as easy as might be expected).

Theorem 4.2. *Every finite REA-PO can be embedded in $(\mathcal{D}, \leq, REA)$.*

The proof of this theorem has several pieces.

Suppose we are given a finite REA-PO $A = \{a_1, a_2, \ldots, a_n\}$. For each a_i, we want to construct a set G_i and some pairs l_i^j's and m_i^j's of functions. The G_i are generic objects similar to the ones used for the standard embedding of finite partial orders in

$\mathcal{D}$. Each pair $l_i^j(x,s)$ and $m_i^j(x)$ is devoted to the REA requirement $a_i\ REA\ a_j$. We try to make $l_i^j(x,s)$ a limit computation of the set A_i we want to build of degree $\mathbf{a}_i$ with $m_i^j(x)$ a modulus function for this approximation (see Lemma 2.1).

The degrees we are going to construct are defined inductively from the bottom of our REA-PO up. We let:

$$(\dagger):\ A_i = G_i \oplus \bigoplus_{a_i > a_j} A_j \oplus \bigoplus_{a_i REA a_j} (l_i^j \oplus m_i^j) \oplus \bigoplus_{a_k REA a_i} l_k^i.$$

Each A_i is simply a join of G's, l's and m's. We let $\mathbf{a}_i = \deg(A_i)$. (Note that each A_i is a function.) We say that the sets and functions (G's, l's and m's and other A's) mentioned in this formula *appear* in (the definition of) A_i.

4.1. **Forcing notion.** We now define a notion of forcing $\mathcal{P}$. A condition in $\mathcal{P}$ is a collection p of finite initial segments of the G, l, m's as above such that each m_i^j appears to be a modulus function for l_i^j which limit computes A_i as far as m_i^j can tell. This means that when $m_i^j(x)$ has been defined at some x, then the corresponding limit computation $l_i^j(x,s)$ cannot change beyond $s = m_i^j(x)$, and the final value $l_i^j(x, m_i^j(x))$ equals $A_i(x)$, which we require to be defined if $m_i^j(x)$ has been defined.

We say such pair $l_i^j(x,s), m_i^j(x)$ (or any one of them) is *associated with* the value $A_i(x)$ (and vice versa). Note that in A_i, such $m_i^j(x)$ is always coded in a position after $A_i(x)$ by the formula ($\dagger$) above (i.e., $m_i^j(x) = A_i(x')$ where $x' > x$). This fact will be useful later.

A forcing condition q *extends* p if each finite initial segment extends the corresponding one. (This automatically implies that q does not violate the l-m rules p imposes.) It is easy to see that both the set of forcing conditions and the extension relation are recursive.

4.2. **Positive order and REA requirements.** To see that positive order requirements ($A_j \leq_T A_i$) and positive REA requirements ($A_i\ REA\ A_j$) are satisfied, we only need to show that, given a forcing condition p, one can find an extension q of p where the domains of A_j and A_i are extended, or equivalently, it suffices to know how to extend individual sets or functions G, l, m's.

Note that in a forcing condition, it is possible that $A_i(x)$ is defined but some associated limit computation $l - m$ has not yet settled down, i.e., $m_i^j(x)$ is still undefined.

To extend G, we simply extend it (indeed, arbitrarily) without changing anything else. It is easy to see that the extension q we get satisfies the definition of a condition.

To extend l, we only need to follow the rules m set up. That is, if some $m_i^j(x)$ has been defined, then $l_i^j(x,s)$ for $s \geq m_i^j(x)$ has to be defined to be the same as $A_i(x)$. If the corresponding m has not yet been defined, then we can extend l freely.

To extend m, we first recall that, in the formula defining A_i (($\dagger$) as above), any modulus value $m_i^j(x)$ is coded at a position x' after the value $A_i(x)$ with which it is associated. Therefore here we can assume that this $A_i(x)$ value has already been defined. Then we find the initial segment where the corresponding l_i^j function has been defined. We may need to extend l_i^j for one more change to make it agree with the value of A_i it limit computes. Then we can define m_i^j to be the last number at which l_i^j changes. For example, suppose we want to define $m_i^j(x)$. At the moment $l_i^j(x,s)$ has been defined up to some s_0 and the last value is t, where $A_i(x) = t'$ which might be different from t, and so we need to extend $l_i^j(x,s)$ further with value t'. Then we define $m_i^j(x)$ to be the last s where $l_i^j(x,s)$ changes (for example, in this case $m_i^j(x) = s_0+1$).

Therefore when we construct a generic filter and let $\mathbf{a}_i = \deg(A_i)$ all the positive order and REA requirements are automatically satisfied.

Now we must show that the negative order and REA relations in our REA-PO are also preserved. We first consider the nonorder requirements and argue that sufficient genericity suffices to satisfy them.

4.3. **Negative order requirements.** Say we have a fixed requirement $A_i \neq \Phi(A_j)$ and a forcing condition p, and we want to find a forcing condition q extending p which forces $A_i \neq \Phi(A_j)$.

Our construction here is not the easiest possible, but we want to introduce a module of "modification" that we will use later for the negative REA requirements.

We pick the first undefined number $G_i(x') = A_i(x)$ (this implies that all associated modulus $m_i^k(x)$'s are undefined in p). We ask whether there is a forcing extension q of p which forces $\Phi(A_j;x)\downarrow$. If not, then p already forces $A_i \neq \Phi(A_j)$. If so, we pick such q and check if its $A_i(x)$ equals $\Phi(A_j;x)$ or not. Note that if $A_i(x)$ is still undefined, then as argued in §4.2, we can extend q and define $A_i(x)$ to be any value we want, and in particular we can make it different from $\Phi(A_j;x)$. If at p, $A_i(x)$ is different from $\Phi(A_j;x)$, then p already forces $A_i \neq \Phi(A_j)$. So we only need to handle the case when $G_i(x') = A_i(x) = \Phi(A_j;x)$ at p.

Now the plan is to change q to q' by modifying $A_i(x)$ together with some associated l-m pairs. We call this a *modification process*. In q, we first change $G_i(x') = A_i(x)$ to be a different value. This value is associated with some limit computation l-m pairs ($l_i^k(x,s)$-$m_i^k(x)$) for A_k limit computing A_i. In order to make a new forcing condition q', we may need to change some of the values of some of these l-m pairs. In this case, for each l-m pair whose limit value needs modification, we make $m(x)$ large and extend l to change the limit value without changing the current values. For example, if the current $m_i^k(x)$ has been defined and $l_i^k(x,s)$ has been defined up to $s = s_0$, then we let the new value of $m_i^k(x)$ be s_0+1, and extend $l_i^k(x,s)$ to change the limit value at s_0+1 which agrees with the new $A_i(x)$.

Note that in the forcing notion, each l_i^k limit computation corresponds to a unique modulus m_i^k, and so here there is no worry that the extended limit computation function l_i^k conflicts with other coding requirements.

For each m value modified, we need to inductively change its own associated limit computation (as a value of $A(x)$) in the same way. We will see that we eventually finish this modification process and get a new forcing condition q' where $A_i(x)$ is different.

Briefly, we have a queue of l-m pairs we need to modify. Each time we pick the first one from the queue, change its m value and extend l, and then add all existing l'-m' pairs for limit computing the m value we just modified into the end of the queue. As we noted earlier, for such l'-m' pairs, the m' value is coded after the m value it is associated with, so this modification process will eventually stop.

Now we want to show that in q', the computation of $\Phi(A_j;x)$ remains the same as that in q, i.e., the use of A_j is unchanged. Each m_i^k (the modulus of some limit computation for A_i) only appears in A_i and all A_l above A_i, therefore it is not difficult to prove by induction that in the above modification process, the only modified values are in A_i or in some A_l which are above A_i. Since $a_j \not\geq a_i$, A_j will not be changed up to the use of $\Phi(A_j;x)$ at q. So the new forcing condition q' forces $A_i \neq \Phi(A_j)$. Here is the lemma (as proved above) which we will need later for the REA case.

Lemma 4.3. *In this modification process, the only changed values (other than the initial one at $G_i(x') = A_i(x)$ are in m's that correspond to limit computations of A_i or some A_l above A_i.*

In addition, note that we chose $A_i(x)$ to be undefined at p, and if some modulus function m is defined at p, then the associated A value is also defined at p (see definition of the notion of forcing in §4.1). Then it is easy to prove by induction that none of the modified values from q to q' are already defined at p. As a result, q' still extends p. This shows that we can always find an extension of a given p which forces $A_i \neq \Phi(A_j)$.

4.4. **Negative REA requirement.** Now we want to deal with negative REA requirements: $\neg(A_i\ REA\ A_j)$. First of all we only need to handle this requirement for pairs $A_i > A_j$. In addition, by the Transitivity Axiom we know that there is no $a_k < a_j$ such that $a_i\ REA\ a_k$. This will turn out to be the crucial property we need to show that the A_j part of a condition is not changed in some modification processes. For example, for all l-m pairs in A_i which aim to limit compute $A_i(G_i)$, neither l nor m appears in A_j. This fact will be used in the proof that A_j is unchanged after some modification process (see Lemmas 4.4 and 4.7).

Recall that we need to make sure that the degree of A_i is not REA the degree of A_j, so in order to satisfy the requirement, we need to make sure that none of the pairs $\lambda(x,s) = \Lambda(A_j;x,s)$ and $\mu(x) = M(A_i;x)$ form a limit computation-modulus function pair for computing A_i. So suppose we are given a forcing condition p and such a pair of functionals (Λ, M) and we want to find a q which forces that (Λ, M) is not a limit computation pair for A_i. We use the notation $M_p(\alpha_i;x)$ to mean $M(\alpha_i;x)$ where α_i is the finite initial segment of A_i in the forcing condition p.

First we try to force totality of both functionals. We ask whether there is a forcing extension $q \leq p$ and a pair $\langle x,s\rangle$ such that no $r < q$ makes $\Lambda_r(\alpha_j;x,s)$ converge. If so, q already forces the requirement. Similarly we can force the totality of $M(A_i)$.

Now we ask if there is a forcing extension q of p and an x which sees that $M_q(\alpha_i;x)$ is not the modulus of $\Lambda_q(\alpha_j;x,s)$, i.e., $M_q(\alpha_i;x)\downarrow= t$ but $\Lambda_q(\alpha_j;x,s)$ changes after t. If so, such q also forces the requirement.

Similarly we can also ask if it is the case that the final value does not equal $A_i(x)$. If so, we are also done. So if none of these questions have a positive answer, then we have a forcing condition p which forces totality of both Λ and M (in a weak sense as above), and forces that M is a modulus for Λ limit computing A_i in the sense that for every x and for every extension $q \leq p$, if $M_q(\alpha_i;x)$ converges, then $\Lambda_q(\alpha_j;x,s)$ stops changing within modulus $M_q(\alpha_i;x)$, and the final value equals $\alpha_i(x)$ (at q).

Now we want to derive a contradiction. Similar to the negative order construction above, we find the first x such that $A_i(x) = G_i(x')$ has not yet been defined at p. The plan is to use what we have forced above to construct an extension of p which forces that (Λ, M) is not a limit computation pair witnessing A_i REA A_j. In other words, we want to derive a contradiction and so the above scenario cannot happen in the construction.

By our assumption, we can find an extension $q \le p$ such that $A_i(x)$, $M_q(\alpha_i; x) = s_0$, $\Lambda_q(\alpha_j; x, s_0) = A_i(x)$ have all been defined.

In q, we want to change $A_i(x) = G_i(x')$ (say $= 0$) to a different value (1). In order to make a new forcing condition, we may have to change various l-m functions as well.

The plan here at this step is the same as that in the negative order construction (§4.3): For each l-m pair we need to modify, we change the value of m to a large number and extend the l part. So the values that need to be changed are all in m's. It follows by induction that these values are in some m's that only appear in A_i or in A's above this A_i. In particular, the A_j part of q is preserved in this process (Lemma 4.3). Let us call the new forcing condition q'. Also note that by the same argument as in the last paragraph of §4.3, this q' extends our original forcing condition p.

Now again by our assumption, we can extend q' to r' where $M_{r'}(\alpha_i; x) \downarrow = s_1$ and $\Lambda_{r'}(\alpha_j; x, s_1) = 1$, i.e., the new value of $A_i(x)$. It is obvious that $s_1 > s_0$ since q' does not change the A_j part of q and so $\Lambda_{q'}(\alpha_j; x, s_0) = \Lambda_q(\alpha_j; x, s_0) = 0$.

Then we want to change r' "back" to r such that the A_i part is changed back to what it was in q. That is, change $A_i(x)$ back to 0 and do another chain reaction modification.

The modification process here is a bit trickier than what we used above to go from q to q'. The new rule is as follows: Step by step we modify values and add the new l-m pairs we need to modify to the end of a queue. Each time we take the first pair from the queue. If the place at which the m in the l-m pair needs modification is z and $m(z) = A_i(z')$ is already defined at q, then we change $m(z)$ back to its old value at q and change all l values after the old modulus at q back to their old limit values. Then, we add all existing l'-m' pairs for computing m and the l values we just modified into the queue. Otherwise (even if they are in the current A_i but the m value is not defined at q) we redefine m to be large and extend the l part, and add the associated l'-m' pairs for computing m to the queue. (The reason why this process will eventually stop requires some extra work, see Lemma 4.5 below.)

In the first option, we say both l and m are modified, and in the second option, we say m is modified but l is extended.

Now we need a few lemmas to show that this new forcing condition r gives a desired contradiction.

Lemma 4.4. *The values that are modified in this process are either in l's appearing in some A_k that enumerates A_i (and hence not below A_j) or in m's appearing in sets above any of these A_k's or A_i.*

Proof. When we changed q to q', the modified values are in m's which are modulus functions which correspond to limit computations for A_i or for some A_l above A_i (Lemma 4.3). So here when we apply the first option modification, it must be the case that the m's we changed back correspond to limit computations that compute A_i but do not appear in any sets strictly below A_i. Therefore according to the construction, the only case in which some l values get modified is when they are in some A_k that enumerates A_i. The second option modification only changes m without changing l (i.e., the chain reaction for such modifications only result in modifying functions appearing in higher A's), and so the lemma follows. □

Lemma 4.5. *The second modification process eventually stops.*

Proof. By the proof of the previous lemma, the only modified l values are those which are used for some A_k enumerating A_i and which were extended in the first modification process (from q to q'). There are only finitely many l positions to modify, and remaining modifications happen only in modulus values which are coded after the values they are associated with. Therefore the modification eventually stops, as the forcing condition is finite. □

Lemma 4.6. *After this modification, r is a valid forcing condition.*

Proof. This is straightforward. In both cases, when we change some value $A(z)$ which has limit computation l-m, we always guarantee that we modify (or extend) this pair as well so that the limit computation is correct. □

Lemma 4.7. *After this modification, the A_j part of r' is compatible with that of r.*

Proof. By Lemma 4.4, no values in A_j will be modified in this construction (though it may get extended). □

Lemma 4.8. *After this modification, the A_i part of r is compatible with that at q.*

Proof. In particular, this is because the values that need second option modification in sets below A_i are all undefined at q. When we changed q to q', we modified m and extended l, and so when we change m back, all l's that are modified are undefined at q. Now it is not difficult to prove that step by step following the same procedure as in the modification from q to q', the first option changes A_i back to its old version at q. □

It is also easy to prove by induction that $r < p$ since none of these modifications affect p where $A_i(x)$ is undefined. Now we get a contradiction. By Lemma 4.8 $M_r(\alpha_i; x) = M_q(\alpha_i; x) = s_0$ but by Lemma 4.7 $\Lambda_r(\alpha_j; x, s_1) = \Lambda_{r'}(\alpha_j; x, s_1) = 1 \neq A_i(x)$. This finishes the proof of Theorem.4.2.

Corollary 4.9. *The Σ_1 theory of $(\mathcal{D}, \leq, REA)$ is decidable.*

Remark. It is possible to prove Theorem 4.2 for countable REA-PO's by modifying the notion of forcing to allow the sets and functions to be drawn from the natural countable set but restricting any condition to mention only finitely many of them.

5. Some nondefinability corollaries

We first try to analyze the construction of the previous section and see how low down in the degrees we can embed our REA-PO.

It seems that we asked $0''$ questions in satisfying the negative REA requirements. However, the proof can be rephrased as a $0'$ construction.

We start with p and pick $A_i(x)$ to be undefined. We first try to force $M(\alpha_i; x)$ to converge (one question to $0'$) and if this is possible we then try to force convergence of $\Lambda(\alpha_j; x, M(\alpha_i; x))$ (one question to $0'$). So we get a forcing extension $q \leq p$.

We change q to q' as in the proof by changing the value of $A_i(x)$ (recursive construction) and extend q' to r' (again two questions to $0'$). Then we change r' back to r by changing $A_i(x)$ back (recursive construction) and the proof ensures that r extends p and forces that A_i is not limit computed from $\Lambda(A_j)$ via modulus $M(A_i)$.

In summary, we only need to ask at most four questions to $0'$ to get the forcing extension we need. Moreover, it is easy to see that forcing negative order requirements only requires one question to $0'$, and one can add in requirements to force the jump of each A_i, so by standard arguments one can embed the REA-PO in the superlow degrees (a degree $\mathbf{a}$ is superlow if $\mathbf{a}' \leq_{wtt} \mathbf{0}'$). Every superlow degree is array recursive (see [Sc, Prop 6.3]), and so every finite REA-PO can be embedded into the array recursive degrees. By a similar argument as before, we get the following:

Theorem 5.1. **ANR** *is not definable in* $(\mathcal{D}, \leq, REA)$ *by a* Σ_1 *formula.*

In addition, we get the exact definability bound of the Turing jump in $(\mathcal{D}, \leq, REA)$:

Theorem 5.2. *The Turing jump is definable in* $(\mathcal{D}, \leq, REA)$ *by a* Π_1 *formula but not by a* Σ_1 *formula.*

Proof. For the claim about definability, note that $y = x'$ is the maximum degree which is REA x, i.e., $y = x'$ if an only if $(y\ REA\ x) \wedge \forall z(z\ REA\ x \Rightarrow z \leq y)$.

For the nondefinability claim, assume that the jump is defined (as a relation) by $\exists \bar{d}\theta(x, y, \bar{d})$. Consider any $\mathbf{y} = \mathbf{x}'$ and the corresponding witnesses $\bar{\mathbf{d}}$. One can embed the REA-PO given by $\mathbf{x}, \mathbf{y}, \bar{\mathbf{d}}$ into the superlow degrees and there the photocopy $\mathbf{y}^*$ cannot be the jump of $\mathbf{x}^*$, which gives the desired contradiction. □

It is not difficult to see that we can embed a finite REA-PO into **Arith**, the degrees of arithmetic sets, and also into non-**Arith**, the complementary class of degrees of nonarithmetic sets. Therefore the same argument as above yields the following.

Theorem 5.3. **Arith** *is not definable in* $(\mathcal{D}, \leq, REA)$ *by a* Σ_1 *or* Π_1 *formula.*

6. R.E.,NON-R.E. $\notin \mathbf{\Sigma}_1(\leq, REA)$

In this section we use the same methods to show that "r.e. in" (y is r.e. in x) is not definable in $(\mathcal{D}, \leq, REA)$ by a Σ_1 or a Π_1 formula.

First of all, to show that "not r.e. in" is not definable by Σ_1 formula, we start with incomparable originals $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{y}$ is not r.e. in $\mathbf{x}$ (simply take two minimal degrees that are "far away" from each other). Now we have a list of witnesses $\bar{\mathbf{d}}$ and an REA-PO $(x, y, \bar{d})$.

To this REA-PO we add a new element m and make it below both x and y, and so m is automatically below the elements which are above either x and y. In addition, we make it incomparable with all the other elements (elements that are above neither x nor y). For the REA relation, we make y REA m but no other new REA relation hold. It is not difficult to check that the new partial order $(x, y, m, \bar{d})$ is an REA-PO, and so by Theorem 4.2 we can embed this REA-PO into $(\mathbf{D}, \leq, REA)$. After removing the degree which corresponds to m, we get photocopies of our original degrees with the desired contradiction that $\mathbf{y}^*$ is now r.e. in $\mathbf{x}^*$ since it is REA in $\mathbf{m}$ which is below $\mathbf{x}^*$.

To show that "r.e. in" is not definable by Σ_1 formula, we need a bit more work. We start with incomparable originals $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{y}$ is r.e. in $\mathbf{x}$ and $\mathbf{y}$ is minimal. (We can find such a pair by, for example, the construction of a minimal degree which is Σ_2 but not Δ_2 from [S2].)

Now by our argument we have an original REA-PO $(x, y, \bar{d})$ with the extra property that there is no d below both x and y such that y bounds some z which is REA d (by our extra condition on $\mathbf{x}$ and $\mathbf{y}$ above).

We follow the same forcing construction as in Theorem 4.2 to make photocopies with an extra requirement that $\mathbf{y}^*$ is not r.e. in $\mathbf{x}^*$.

So it suffices to deal with this extra requirement here. For convenience we use a_j and a_i to respectively denote x and y (so we want to make $\deg(A_i)$ not r.e. in A_j).

Here we want to diagonalize against all triples of functionals (Φ, Ψ, W) where Φ and Ψ give the reductions between A_i and some set Y in the same degree ($\Psi(\Phi(A_i)) = \Psi(Y) = A_i$), and $W(A_j)$ gives an A_j-r.e. set which equals Y.

Of course we can start to diagonalize against a given triple (Φ, Ψ, W) by asking whether we can force Φ or Ψ to be partial, or if their composition gives some set which is different from A_i, or whether we can force $Y = \Phi(A_i) \neq W(A_j)$. If all these attempts fail, we show that along the generic filter $Y \leq_T A_j$ and so get a contradiction (we can make A_i and A_j Turing incomparable).

We only need to show that Y is co-r.e. in A_j. To tell if some $y \notin Y$, we try to find a forcing extension q of p such that the A_j part of q is the true one (agrees with the generic filter in the end) and $\Phi_q(\alpha_i; y) = 0$. Whenever we find such q we claim that $y \notin Y$.

It is easy to see that if y is not in Y, then along the generic filter we can always find such a forcing condition q. The hard part is to show that if y is in Y then we cannot find such q. We assume for a contradiction that there is a $q \leq p$ such that its A_j part is correct (compatible with the generic filter) and $\Phi_q(\alpha_i; y) = 0$ but $W(A_j; y) = 1$. Now we want to extend this q to get a forcing condition which forces that $\Phi(A_i) \neq W(A_j)$. The idea is, of course, that we can extend the α_j part of q until we see $W(A_j; y) = 1$. The problem is that, q might have promised some modulus requirements which may not be compatible with the real A_j up to the use which enumerates y into $W(A_j)$.

This is handled in a modification process similar to what we used in Section 4. Starting from q, we extend the α_j at q following the real A_j up to the use of $W(A_j; y) = 1$. If the added values do not cause a contradiction (i.e., they still obey the forcing condition requirements), then we do nothing. Once we detect that there is a contradiction, we want to modify some values. The contradiction comes from either G, l or m which appears in A_j. There will be no problem for m, because the corresponding limit computation l also appears in A_j (we know A_j itself, which comes from the generic filter, is consistent); there will be no problem for G either, since at q, if we have decided some modulus function, then the corresponding G is also defined. The only trouble is to extend l where the m function does not appear in A_j. For example, some m function (appearing in some A_k REA A_j) may have decided a small modulus, but A_j may have changes after that small modulus.

Once we see such a situation, we know that this pair l-m needs modification and add them to the queue. The modification process goes exactly as in §4.3, i.e., we change the modulus to be large and extend the limit computation, then add in the corresponding l'-m' we need modify into the end of the queue.

Lemma 6.1. *The values that are modified in this process are m values which appear in some A_k which is REA some A_l appearing in A_j (including A_j itself), or a set above any of such A_k's. In addition, such A_k do not appear in A_j.*

Proof. By our argument above, the first modification happens when A_j contains some limit computation l but not the modulus, which corresponds to limit computing some A_k REA A_l in the lemma. Then the chain reaction only happen at m's which are above such A_k's. Since A_j does not contain the modulus which needs modification, such A_k are not below A_j. □

By our extra condition (recall that $\mathbf{x}$ is minimal and so not above any degree REA any degree below $\mathbf{y}$) and by the above lemma, one can see that A_i will remain unmodified in the process. Also by the lemma, we know that A_j will remain unmodified (but could be extended). So we can get a new forcing condition $q' < p$ (it is easy to see that the p part of the forcing condition is unmodified in the process) such that its A_i part is the same as that in q ($\Phi_{q'}(\alpha_i; y) = \Phi_q(\alpha_i; y) = 0$) but $W_{q'}(A_j; y) = 1$. This gives the desired contradiction.

7. ANR, HI $\notin \mathbf{\Sigma_0}(\leq,', \bar{\mathbf{c}})$

With the jump added to our language, things get more complicated. Suppose we need to build two degrees $\mathbf{y} > \mathbf{x}$. When we build $\mathbf{x}$, there is a lot of freedom to choose $\mathbf{y}$. Even if we require $\mathbf{y}$ REA $\mathbf{x}$, we can still choose from a countable class of degrees. However if we have $\mathbf{y} = \mathbf{x}'$, then we have no choice at all when we fix $\mathbf{x}$. In particular, the following question is still open:

Question 7.1. *Which degrees above $\mathbf{0}'$ are jumps of HIF degrees?*

We don't even know whether this class (jumps of HIF degrees) is closed downwards (for degrees above $\mathbf{0}'$, of course).

This is actually a big obstacle for nondefinability results even in the quantifier-free case. For example, it is hard and sometimes impossible to construct an HIF degree whose jump is above a fixed $\mathbf{c} > \mathbf{0}'$ (If the jump of a degree is above $\mathbf{0}''$, then the degree is automatically hyperimmune, see for example [J1]). Miller and Martin conjectured that **HI** is not definable by quantifier-free formulae in $(\leq,', \bar{\mathbf{c}})$, and here we confirm this conjecture by proving the equivalent proposition that **HIF** is not quantifier-free definable.

Theorem 7.1. **HIF** *is not definable by a quantifier free formula in $(\leq,', \bar{\mathbf{c}})$.*

Proof. Using the same type of argument as in §3, we have an original $\mathbf{x}$ which is HIF and a quantifier free definition of **HIF** with constants $\bar{\mathbf{c}}$. As before, we will choose a particular $\mathbf{x}$. For example, we can take our $\mathbf{x}$ to be incomparable with all nonrecursive constants appearing in the formula. We regard all the jumps of constants as constants, i.e., for each constant c, if c' appears in the formula we will add a new constant $\tilde{c}$ and add $c' = \tilde{c}$ as a conjunct (for convenience we still write c' but regard it as a constant). Once $\mathbf{x}$ is chosen, we can assume that our formula is a conjunct of atomic formulas or their negations which specify a complete diagram in $\leq$ for the (finitely many) $\mathbf{x}, \mathbf{x}', \ldots$ and $\bar{\mathbf{c}}$ appearing in it.

We now think of the conjuncts (atomic formulae or negations of atomic formulae) appearing in our formula as a list of requirements. We can remove the requirements that involve a degree and its own jump (or iterated jumps) such as $x \leq x'$ or $x' \leq x'''$ which, by basic results about the jump, must be satisfied by all degrees. Similarly we omit any conjunct of the form $\mathbf{0} \leq \mathbf{z}$. We can also assume that no obviously false conjuncts are included, i.e. the diagram is consistent with the axioms for a jump partial order ([HS]).

So the remaining requirements look like the following:

$$\psi: \; P(x') \wedge \bigwedge_{i \in I_1} (x \not\leq a_i) \wedge \bigwedge_{i \in I_2} (b_i \not\leq x),$$

where $P(x')$ is the collection of all conjuncts about x' (or higher jumps such as x'' and x''') but not about x.

Our plan is to build a photocopy hyperimmune degree $\mathbf{y}$ whose jump is $\mathbf{x}'$ (so $\mathbf{y}$ automatically satisfy $P(x')$) and such that $\mathbf{y}$ satisfies all remaining conjuncts in ψ. Then we get our contradiction because $\mathbf{y}$ satisfies ψ, but is hyperimmune.

To get such a degree we build $\mathbf{y}$ low above $\mathbf{x}$. This guarantees that its jump is the same as $\mathbf{x}'$ and it is hyperimmune since it is relatively Δ_2 ([MM, Theorem 1.2]). (Note that using $\mathbf{x}'$ as an oracle, it is easy to make sure that $\mathbf{y} \leq \mathbf{x}'$.) We use finite forcing to build initial segments σ_i recursively in $\mathbf{x}'$ and let $\cup_i \sigma_i = Y$. We then take $\mathbf{y} = \deg(X \oplus Y)$ where $X \in \mathbf{x}$ and so guarantee that $\mathbf{y} \geq \mathbf{x}$. In the construction we force the jump of $X \oplus Y$ as we go along. Therefore $(X \oplus Y)' \leq_T X'$, i.e., $\mathbf{y}' = \mathbf{x}'$. To finish the proof we discuss how to satisfy all the requirements in the conjuncts listed in ψ.

Avoiding lower cones ($y \not\leq a_i$) is automatic: each $\mathbf{a}_i$ is not above $\mathbf{x}$, so it cannot be above $\mathbf{y}$ as $\mathbf{y} \geq \mathbf{x}$.

For upper cones, note that we only need to consider those $\mathbf{b_i}$'s that are below $\mathbf{x}'$: otherwise $\mathbf{y}$ cannot be above such $\mathbf{b_i}$. Now given σ we can ask whether there are two extensions τ_0 and τ_1 of σ such that $(X \oplus \tau_0)|_e(X \oplus \tau_1)$, i.e., there is an n such that $\varphi_e^{X \oplus \tau_0}(n) \downarrow \neq \varphi_e^{X \oplus \tau_1}(n)$. If so, we take the one which differs with a fixed $B_i \in \mathbf{b_i}$. If not, we can easily argue that the function $\varphi_e^{X \oplus Y}$ is recursive in X if it is total, and so it is not equal to B_i as $\mathbf{b_i} \not\leq \mathbf{x}$. In either case we can satisfy the requirement $\mathbf{b_i} \not\leq \mathbf{y}$. $\square$

Corollary 7.2. AR *is not definable by a quantifier free formula in* $(\leq, ', \bar{\mathbf{c}})$.

Proof. As HIF implies AR, we can begin with a quantifier free definition of **AR** in $(\leq, ', \bar{\mathbf{c}})$ and analyze it as in the proof of the Theorem for the same $\mathbf{x}$. The same construction of $\mathbf{y}$ can easily be augmented to make it ANR simply by making its principal function infinitely often be larger than the modulus function for $0'$ while staying recursive in $\mathbf{x}'$. This produces an ANR degree $\mathbf{y}$ which, as above, satisfies the assumed quantifier free definition of **AR** for the desired contradiction. $\square$

8. HI, ANR $\notin \mathbf{\Sigma_1}(\leq, \bar{\mathbf{c}})$

Theorem 8.1. **HI** *and* **ANR** *are not definable by* $\mathbf{\Sigma_1}(\leq, \bar{\mathbf{c}})$ *formulas.*

Proof. The proof is essentially the same for both classes. We give the analysis for **HI** and simply note any comments needed for **ANR** in brackets []. We follow the same general strategy and use the same notations as in the previous sections. That is, we have a hyperimmune [array nonrecursive] degree $\mathbf{x}$ and a sequence of witnesses $\mathbf{d_0}, \ldots, \mathbf{d_n}$ which satisfy a Σ_1 formula $\exists \bar{d} \varphi(x, \bar{d}, \bar{c})$ where c's are parameters. We write φ in disjunctive normal form and by a suitable choice of $\mathbf{x}$ can pick one disjunct ψ (which is a conjunction of atomic formulae or their negations) which says that $\mathbf{x}$ is not in the lower cone or the upper cone of any $\mathbf{c_i}$. Our plan is to find some new $\mathbf{d_0^*}, \ldots, \mathbf{d_n^*}$ and a new hyperimmune-free [and so array recursive] $\mathbf{y}$ which satisfy ψ.

Recall our notation which, given an original degree $\mathbf{z}$, has $\mathbf{z}^*$ denote the corresponding photocopy. In this construction, if $\mathbf{z} = \mathbf{x}$, then $\mathbf{z}^* = \mathbf{y}$; if $\mathbf{z} = \mathbf{d_i}$, then $\mathbf{z}^* = \mathbf{d_i^*}$; and if $\mathbf{z} = \mathbf{c_i}$, then $\mathbf{z}^* = \mathbf{c_i}$. In the same fashion, if we have a set S of degrees, we let $S^* = \{\mathbf{z}^* : \mathbf{z} \in S\}$. Now we fix S as the set of degrees $\mathbf{x}$ and all $\mathbf{d}_i$ which are below $\mathbf{x}$.

We will need to impose some extra conditions on ψ before we begin our construction but initially our plan is as follows: First, we find $\mathbf{y}$ and some degrees $\mathbf{d_i^*}$ below it such that the these degrees provide the same partial order on S^* as that given on S by $\mathbf{x}$ and the $\mathbf{d_i}$'s below $\mathbf{x}$. We then use a construction similar to the one used in Section 3 to build the other $\mathbf{d}^*$'s above members of S^*.

For convenience we let $\mathbf{d_0}, \ldots, \mathbf{d_l}$ be the $\mathbf{d}$'s below $\mathbf{x}$. The problem is that, for example, if $\mathbf{d_0}$ is below some $\mathbf{c_i}$, then it is difficult to build our $\mathbf{d_0^*}$. In particular, it might be very hard to make it hyperimmune-free (as we need to make $\mathbf{y}$ hyperimmune-free [and so array recursive]).

Thus we also assume that in ψ, there is no formula of the form $d_i \leq c$ for $0 \leq i \leq l$ and any constant c. The reason we can make this assumption is that we can take $\mathbf{x}$ to be a hyperimmune degree which avoids all upper cones above any nonrecursive $\mathbf{d}$ which is below any constant $\mathbf{c}$. This is a countable list of requirements and we can use standard constructions to get a 1-generic hyperimmune [and so array nonrecursive] degree which avoids these cones.

Now we follow our plan: first we build $\mathbf{y}$ and $\mathbf{d_0^*}, \ldots, \mathbf{d_l^*}$ which provide the same partial order on S^* as $\mathbf{x}$ and $\mathbf{d_0}, \ldots, \mathbf{d_l}$ do on S and make sure that $\mathbf{y}$ is hyperimmune-free [and so array recursive] and the following holds:

$(*)$ For any subset of degrees T^* of S^*, any finite set R of constants and any fixed degree $\mathbf{z}$ among the $\mathbf{x}$, $\mathbf{c_i}$'s and $\mathbf{d_i}$'s, if the join of all degrees in $T \cup R$ is not above $\mathbf{z}$, then the join of all degrees in $T^* \cup R$ is not above $\mathbf{z}^*$.

This property seems ad hoc, but in fact it is natural and essential in the construction. For example, it implies that $\mathbf{y}$ is not above or below any of the $\mathbf{c_i}$'s. Later this property will also guarantee that when we construct other $\mathbf{d}^*$'s, we do not get a degree which is too high by joining it with some degrees we already have.

To carry out the construction of S^* with property $(*)$, we use the uniform tree constructions using usl tables which are used to build initial segments of $\mathcal{D}$ and with which we assume familiarity (see [Le, Chapter VI]). We need to satisfy the usual conditions preserving the usl relations as given by those on S as well as those that force totality to guarantee that the degrees constructed (and in particular the top one $\mathbf{y}$) are hyperimmune-free. We do not need to satisfy the most complicated conditions that would make S^* an initial segment but do need some extra requirements to satisfy $(*)$.

Consider a tree and a requirement $Z^* \neq \varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}$ (where Y may be viewed as included among the D_i^*). If Z is some constant C, then we simply ask whether we can find e-splittings on the tree for $\varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}$. If so, we can choose one to diagonalize against Z. If not, then it is easy to see that, if $\varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}$ is total, $Z \leq_T \bigoplus C_i$ which contradicts our assumption. If Z is X or some D_j, then we pick two nodes τ_0, τ_1 on the tree which are congruent modulo $\bigoplus D_i^*$ but disagree on Z^* (say, at x). This is possible since $Z \not\leq_T \bigoplus D_i$ and by our usl representation we are able to make $Z^* \not\leq_T \bigoplus D_i^*$. Then, we try to extend τ_0 to τ' where we have a convergent computation of $\varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}(x)$. If there is no such τ', then we take the full subtree above τ_0 to satisfy our requirement. If there is such a τ', then we extend τ_1 in the same way to τ'' such that τ' and τ'' are congruent modulo $\bigoplus D_i^*$ (by uniformity). Therefore at τ' and τ'', the computations $\varphi_e^{\bigoplus D_i^* \oplus \bigoplus C_i}(x)$ give the same value. We can pick one whose $Z^*(x)$ is different from this value and take the full subtree above it.

Now we need to show that these $\mathbf{d_i^*}$'s satisfy the same relations with the constants as their originals: For each original $\mathbf{d_i}$ ($i = 1, 2, \ldots, l$ or $\mathbf{x}$), $\mathbf{d_i}$ is not in any of the upper or lower cone of any constant $\mathbf{c}$. So we only need to show that $\mathbf{d_i^*}$ is incomparable with each of the constants. It is not difficult to see that these relations are guaranteed by property $(*)$.

Next let $\mathbf{d_{l+1}}, \ldots, \mathbf{d_k}$ be the $\mathbf{d}$'s that are above any degree in S. We need to build new $\mathbf{d_{l+1}^*}, \ldots, \mathbf{d_k^*}$ which satisfy the same diagram (we regard the remaining $\mathbf{d}$'s as constants). This construction follows the same idea as in Section 3. We find $\mathbf{g_{l+1}}, \ldots, \mathbf{g_k}$ which are mutually 1-generic over $S^* \cup R$ where R is the set of all constants. We let the $\mathbf{d}^*$'s be the following joins:

$$\mathbf{d_i^*} = \bigvee_{\mathbf{d_j} \leq \mathbf{d_i}} \mathbf{g_j} \vee \bigvee_{\mathbf{z} \leq \mathbf{d_i}, \mathbf{z} \in S \cup R} \mathbf{z}^*.$$

We need to show that each $\mathbf{d_i^*}$ (among $\mathbf{d_{l+1}^*}, \ldots, \mathbf{d_k^*}$) satisfies the same relations with different $\mathbf{z}^*$ as its original $\mathbf{d_i}$ does (with $\mathbf{z}$). We discuss this by cases:

If $\mathbf{z}$ is below $\mathbf{d_i}$, then it is automatic by the definition of $\mathbf{d_i^*}$ that $\mathbf{z}^* \leq \mathbf{d_i^*}$. If $\mathbf{z}$ is above $\mathbf{d_i}$, then it can only be the case that $\mathbf{z}$ is some $\mathbf{d_j}$ and we are back in the first case. So we only need to take care of the negative order relations that say $\mathbf{z}$ is incomparable with $\mathbf{d_i}$. Here we further divide into subcases.

If $\mathbf{z}$ is $\mathbf{x}$ (or a constant $\mathbf{c}$, or a degree $\mathbf{d}$ in S, as the arguments for them are essentially the same), we need to show that $\mathbf{d_i^*}$ is incomparable with $\mathbf{y}$. First, $\mathbf{y}$ is not above $\mathbf{d_i^*}$ because $\mathbf{g_i}$ is not below $\mathbf{y}$; $\bigvee_{\mathbf{z} \leq \mathbf{d_i}, \mathbf{z} \in S \cup R} \mathbf{z}^*$ is not above $\mathbf{y}$ by property $(*)$ (since their originals do not join to some degree above $\mathbf{x}$), and so by Lemma 3.2 joining it with some mutually 1-generic degrees does not make $\mathbf{d_i^*}$ above $\mathbf{y}$.

If $\mathbf{z}$ is some other $\mathbf{d_j}$ among $\mathbf{d_{l+1}}, \dots, \mathbf{d_k}$, then since $\mathbf{d_i}$ and $\mathbf{d_j}$ are incomparable, $\mathbf{g_i}$ and $\mathbf{g_j}$ only appear in one join but not the other. By 1-genericity $\mathbf{d_i^*}$ and $\mathbf{d_j^*}$ are incomparable.

This gives the desired contradiction that $\mathbf{y}$ with witnesses $\mathbf{d}^*$'s satisfy the same formula $\exists d_0 \exists d_1 \dots \exists d_n \varphi(x, d_0, d_1, \dots, d_n)$ but $\mathbf{y}$ is hyperimmune-free [and so array recursive]. □

9. Remarks

9.1. **Nondefinability in the language** ($\leq$)**.** Shore [S, Proposition 7.6], following related results in [LS], showed that neither the $\mathbf{Low}_2$ nor non-$\mathbf{Low}_2$ degrees are definable by a Π_2 formula in the language of order alone (without parameters). The only property needed of these classes for the proofs there is that one can embed every finite lattice as an initial segment in them. The rest of the proofs rely only on general algebraic facts about $\mathcal{D}$ and Kleene-Post type results about extension of embeddings. Thus we can use the same argument to get the same nondefinability result for any class of degrees that contains copies of all finite lattices as initial segments of $\mathcal{D}$.

This argument works for **HI, HIF, AR, Arith** and non-**Arith** (by classical results on embedding initial segments, see [Le]). We do not know whether **ANR** is definable by a Π_2 formula in the language with just order (with REA added in, it is definable as mentioned in the Introduction).

9.2. **Nondefinability in the language** ($\leq,'$)**.** Following the outline of the arguments in Sections 5, one can get similar Σ_1-nondefinability results for the language with jump from embedding theorems about jump partial orders. For example, it is known that every finite jump partial order ([HS]), or even finite jump upper-semi lattice ([M]), is embeddable in the Turing degrees. By modifying their constructions one can easily make one of the elements HI or ANR, and so we know that **HIF** and **AR** are not Σ_1 definable in ($\leq,'$). In addition, one can also make sure that the degrees constructed are either all arithmetic or all nonarithmetic. This shows that **Arith** is not definable by Σ_1 or Π_1 formulae in ($\leq,'$) (while there is a Π_2 definition again as mentioned in the Introduction).

9.3. **Open Questions.** First of all, at the Σ_1 level, there is one obvious remaining open question: Is **HI** definable by Σ_1 formula in $(\leq, REA)$? We believe arguments similar to those in Section 5 may work (in particular. we would want to build a REA-PO and make a certain element HIF), but our first attempts seem have meet some obstacles.

Most of our analysis applies to the language with REA but not RE. It is an interesting question whether similar results can be obtained for RE. If so, they seem likely to require priority type arguments rather than (or in addition to) forcing arguments. In another direction, we can also try to find possible definitions of RE in the language augmented by REA, which might be easier than the language with order alone.

Of course, many other degree classes can be analyzed for definability and nondefinability results.

References

[CS] M. Cai and R. A. Shore, *Domination, forcing, array nonrecursiveness and relative recursive enumerability*, Journal of Symbolic Logic, Volume 77, 2012, Pages 33–48.

[C1] M. Cai, *A hyperimmune minimal degree and an ANR 2-minimal degree*, Notre Dame Journal of Formal Logic, Vol. 51, Number 4, 2010, Pages 443–455.

[C2] M. Cai, *Array nonrecursiveness and relative recursive enumerability*, Journal of Symbolic Logic, Volume 77, 2012, Pages 21–32.

[DJS] R. Downey, C. Jockusch and M. Stob, *Array nonrecursive degrees and genericity*, London Mathematical Society Lecture Notes Series 224, 1996, University Press, Pages 93–105.

[F] R. M. Friedberg, *A criterion for completeness of degrees of unsolvability*, Journal of Symbolic Logic, Vol. 22, 1957, pages 159–160.

[HS] P. G. Hinman and T.A. Slaman, *Jump embeddings in the Turing degrees*, Journal of Symbolic Logic, vol. 56 (1991), pages 563–591.

[J1] C. Jockusch, *The Degrees of Hyperhyperimmune Sets*, The Journal of Symbolic Logic, Volume 34, Issue 3 (1969), pages 489–493.

[J2] C. G. Jockusch, *Degrees of generic sets*, in Recursion Theory: its Generalisations and Applications, F. R. Drake and S. S. Wainer (editors), Cambridge University Press, 1980, pages 110–139.

[JS] C. G. Jockusch, Jr. and R. A. Shore, *Pseudo-jump operators II: transfinite iterations, hierarchies and minimal covers*, Journal of Symbolic Logic, Vol. 49, 1984, pages 1205–1236.

[Le] M. Lerman, *Degrees of Unsolvability: Local and Global Theory*, Perspectives in Mathematical Logic, Volume 11, Springer-Verlag, 1983.

[LS] M. Lerman and R. A. Shore, *Decidability and invariant classes for degree structures*, Transactions of the American Mathematical Society, 310 (1988), pages 669–692.

[L] A. E. M. Lewis, Π^0_1 *classes, strong minimal covers and hyperimmune-free degrees*, Bulletin of the London Mathematical Society, volume 39, number 6, 2007, pages 892–910.

[MM] W. Miller and D. A. Martin, *The Degrees of Hyperimmune Sets*, Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik, vol. 14 (1968), pages 159–166.

[M] A. Montalbán *Embedding Jump Upper Semilattices into the Turing Degrees*, Journal of Symbolic Logic , Vol. 68, No. 3 (2003), pages 989–1014

[Sc] B. Schaeffer, *Dynamic notions of genericity and array noncomputability*, Annals of Pure and Applied Logic, Volume 95, 1998, pages 37–69.

[So] R. I. Soare, *Recursively enumerable sets and degrees*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1987.

[S] R. A. Shore, *Direct and local definitions of the Turing jump*, Journal of Mathematical Logic, Vol. 7, 2007, pages 229–262.

[S2] R. A. Shore, *Minimal degrees which are* Σ^0_2 *but not* Δ^0_2, Proc. American Mathematical Society, 132 (2004), pages 563–565.

[SS] R. A. Shore and T. A. Slaman, *Defining the Turing jump*, Mathematical Research Letters, Vol. 6, 1999, pages 711–722.

[SW] T. A. Slaman, *Global properties of the Turing degrees and the Turing jump*, in *Computational prospects of infinity, Part I, Tutorials*, Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singapore, 2008, World Sci. Publ., Hackensack, NJ, pages 83–101.

[Sp] C. Spector, *On Degrees of Recursive Unsolvability*, Annals of Mathematics, Vol 64, No. 3, 1956, pages 581–592.

Department of Mathematics, University of Wisconsin, Madison WI 53705
E-mail address: `mcai@math.wisc.edu`

Department of Mathematics, Cornell University, Ithaca NY 14853
E-mail address: `shore@math.cornell.edu`