Approximating Σ_n Classes by Σ_n Opens

Duarte Maia

November 21, 2024

1 Introduction

The context for this document is Section 6.8 of [1]. In trying to read it, more than in other sections, I found that I was not appropriately appreciating the distinction between a Σ_n class an an open generated by a Σ_n set, because the difference is not relevant in the $n = 1$ case. In this document, I will clarify (to myself) the distinction, and subsequently prove a relevant approximation lemma.

2 Classes vs. Opens

First, let us be clear about the spaces of discussion. On one hand, we have the space of finite strings $2^{<\omega}$. A subset of this space is referred to as a "set of strings", or just as a "set". On the other hand, we have Cantor space 2^{ω}. A subset of this space is referred to as a "class". We will relentlessly fail to distinguish between the following two views on what an element of Cantor space is: A subset of $\mathbb{N}$, or a sequence of zeros and ones.

The definition of Σ_n set and Σ_n space is almost the same, with some notational distinctions. We will restrict ourselves to the case $n \geq 1$.

Definition 1. A Σ_n set of strings is a set of the form

$$A = \{ \sigma \in 2^{<\omega} \mid \exists_{x_1} \forall_{x_2} \dots Q_{x_n} P(\sigma, x_1, \dots, x_n) \},$$

where P is a recursive predicate.

Definition 2. A Σ_n class is a collection of sets of numbers of the form

$$\mathcal{C} = \{ A \in 2^{\omega} \mid \exists_{x_1} \forall_{x_2} \dots Q_{x_n} P(A, x_1, \dots, x_n) \},$$

where P is a recursive predicate.

Definition 3. An *index* for a Σ_n set/class consists of an index for the recursive predicate P.

It is worth going into a little more depth about the meaning of "P is a recursive predicate" in Definition 2, notably when it comes to the meaning of A being given as input. One way to see it is one that can be computed by a Turing machine whose initial tape contains the tuple $\vec{x} = \langle x_1, \dots, x_n \rangle$ and that has access to A as an oracle tape. In other words, we may instead write $P(A, x_1, \dots, x_n)$ as $P^A(x_1, \dots, x_n)$. Note that the recursiveness requirement means that the Turing machine in question will halt regardless of input, including A.

Note that a finite computation can only check a finite amount of data from A. In other words, the value of $P(A, x_1, \dots, x_n)$ depends only on a certain long enough prefix of A, the length of which depends on the x_i and on A itself. It turns out, however, that given $\vec{x}$ we can (recursively) find a uniform bound on how long a prefix of A is necessary to determine the value of P.

Proposition 4. Let $P(X, x)$ be a recursive predicate whose first input is an element of Cantor space and whose second input is a natural number. Then, there exists a recursive function f such that the value of $P(X, x)$ depends only on $X \upharpoonright f(x)$.

Proof: Consider the following computable binary tree $T \equiv T(x)$. We say a string σ is in T if the computation of $P(\sigma, x)$ either consults elements of the input tape past the length of σ, or has not terminated after $|\sigma|$ steps. Now, since P is recursive and thus terminates on every input, there is no path through T, and thus T must be finite. A brute force search will easily find the depth of the tree $T(x)$, which provides the desired function $f(x)$. ■

Corollary 5. Let $P(X, x)$ be a recursive predicate as above. Then, for any value of $x \in \mathbb{N}$ there is a finite collection of strings $D(x)$, computable from x, such that

$$P(X, x) \text{ iff } \bigvee_{\sigma \in D(x)} \sigma \leq X,$$

where $\sigma \leq X$ means that σ is a prefix of X.

Now, let us go back to Definition 2, and let us momentarily focus on the case where $Q = \exists$. In this case, $\exists_{x_n} P(A, \vec{x})$ may be replaced by $\exists_{x_n} \exists_{i \leq N(\vec{x})} \sigma(\vec{x}, i) \leq A$, where $\sigma(\vec{x}, 0), \ldots \sigma(\vec{x}, N(\vec{x}))$ is some enumeration of $D(\vec{x})$. On the other hand, the case where $Q = \forall$ may be rearranged using quantifier laws to yield

Corollary 6. Every Σ_n class may be written in the form

$$\mathcal{C} = \{\, A \in 2^\omega \mid \exists_{x_1} \neg \exists_{x_2} \ldots \neg \exists_{x_n} (R(\vec{x}) \wedge \sigma(\vec{x}) \leq A) \,\},$$

where an index for the recursive functions R and σ may be effectively found from an index for the class.

Remark 7. We only need to include R in our discussion because of the possibility that, for some tuple $\vec{x}$, the set $D(\vec{x})$ is empty. It may be avoided by having our variables x_i range over recursive/c.e. sets that may depend on the previous x_j. We will take this approach henceforth.

Let us now briefly go over an important topological facet of this corollary. For the sake of notation, we will momentarily restrict our attention to $n = 1$, 2, and 3, but similar considerations hold at all levels.

Recall that we define a topology on 2^ω, which is generated by the basic open sets

$$U_\sigma = \{\, A \in 2^\omega \mid \sigma \leq A \,\}.$$

As a consequence, Corollary 6 may be interpreted as follows:

- Any Σ_1 class may be written as $\mathcal{C} = \cup_x U_{\sigma(x)}$,
- Any Σ_2 class may be written as $\mathcal{C} = \cup_x \cap_y U^c_{\sigma(x,y)}$,
- Any Σ_3 class may be written as $\mathcal{C} = \cup_x \cap_y \cup_z U_{\sigma(x,y,z)}$,
- And so on,

where in either case σ is a recursive function and each variable ranges over a recursive set depending on the previous variables. In other words, x ranges over a recursive set C, y ranges over a uniformly recursive set $C'(x)$, and z ranges over a uniformly recursive set $C''(x, y)$. We can also assume without loss of generality that only the *last* variable ranges over a set that varies, and that all variables preceding it vary over all of $\mathbb{N}$. It's also worth noting that everything that has been done thus far is reversible, and indeed in each instance the written expression determines a Σ_n class.

Thus, we obtain that a Σ_1 class is an "effective open". In other words, while a "general" open is a union of basic open sets, a Σ_1 class is a union of basic open sets where the list of basic open sets in question can be recursively enumerated. In a similar vein, a Σ_2 class is an "effective F_σ set", and so on.

Let us discuss this idea of "effective open" a little further. It lends itself to a trivial generalization:

Definition 8. If X is an oracle, an *X-effective open* is a set of the form $U = \cup_{x\in\mathbb{N}} U_{\sigma^X(x)}$, where σ^X is an X-computable enumeration of strings.[1] Equivalently, an X-effective open is a union $U = \cup_{\sigma\in A} U_\sigma$, where A is an X-enumerable set.

By considering the case where $X = 0^{(n-1)}$, we obtain that a $0^{(n-1)}$-effective open is one which is generated by a $\Sigma_n$set of basic opens, so we make the following definition (which I believe is not standard in the literature):

Definition 9. For $n \geq 1$, a Σ_n *open* means the same as a $0^{(n-1)}$-effective open.

3 Relating Terms and Open Approximations

We now relate the following three objects of discussion, all of which are classes of subsets of Cantor space:

- Σ_n open sets.
- Open $\Sigma_n$classes,
- Σ_n classes,

We now proceed to relate them. It is worthy of note that all the theorems that will be stated henceforth, as well as their proofs, are effective, but to avoid clutter we will neglect to mention that fact.

Proposition 10. Every Σ_n open is a Σ_n class. In particular,

$$\Sigma_n \text{ open} \implies \text{Open } \Sigma_n \text{ class} \implies \Sigma_n \text{ class}.$$

Proof: A Σ_n open is of the form

$$U = \{\, A \in 2^\omega \mid \exists_\sigma[(\vec{Q}_{\vec{x}} P(\sigma, \vec{x})) \wedge \sigma \leq A]\,\},$$

where $\vec{Q}_{\vec{x}}$ abbreviates an alternation of n quantifiers starting with an $\exists$. Now, the formula at hand can be rearranged as to make it a Σ_n formula in A. To be precise, we get

$$U = \{\, A \in 2^\omega \mid \exists_\sigma \exists_{x_0} \forall_{x_1} \ldots Q_{x_n}(P(\sigma, \vec{x}) \wedge \sigma \leq A)\,\}.$$

Note that the predicate $(A, \sigma, \vec{x}) \mapsto P(\sigma, \vec{x}) \wedge \sigma \leq A$ is recursive, and so the proof is complete. ■

We now show that the three classes coincide when $n = 1$, but not for $n \geq 2$.

Proposition 11. Every Σ_1 class is a Σ_1 open.

Proof: A Σ_1 class $\mathcal{C}$ is the set of $A \in 2^\omega$ satisfying $\exists_\sigma \exists_x (P(\sigma, x) \wedge \sigma \leq A)$. In other words, $\mathcal{C}$ is the union of $U_{\pi_1(p)}$, where p ranges over the set of pairs (σ, x) that satisfy $P(\sigma, x)$, which is evidently enumerable. ■

Corollary 12. A Σ_n open is the same as a $\Sigma_1^{0^{(n-1)}}$ class, where the superscript denotes relativization in the obvious way.

Proof: Inspection of the proof of Proposition 11 will show, by relativization, that to define a $\Sigma_1^{0^{(n-1)}}$ class is the same as to define a $0^{(n-1)}$-enumerable collection of basic open sets, i.e. a Σ_n collection of basic open sets. ■

Proposition 13. For $n \geq 2$ there is a Σ_n class that is not open, and an open Σ_n class that is not a Σ_n open set.

[1] Technically, an edge case must be made in this definition for the case where U is the empty set. This can be fixed in several ways, such as having the union range over an arbitrary computable/c.e. set instead of $\mathbb{N}$, or allowing σ^X to be nontotal. That said, the definition that follows is better in this respect.

Proof: For the first part, it suffices to find a Σ_2 class that is not open, and here it is: The singleton $\{0\}$.

For the second part, we also reduce to the case $n = 2$, by showing that there is an open Σ_2 class that is not a $\Sigma_1^{0'}$ class. The general case follows by relativization to $0^{(n-1)}$, to get an open $\Sigma_2^{0^{(n-1)}}$ class – which is easily seen to be a Σ_n class – which is not a $\Sigma_1^{(0^{(n-1)})'}$ class, i.e. a $\Sigma_1^{0^{(n)}}$ class, i.e. a Σ_n open.

The following argument is adapted from page 76 of [1]. The idea is to build an open set U that is of the form $U = 2^\omega \setminus \{A\}$, with A built in such a way as not to be $0'$-computable. We shall ensure that A is "simple" enough that there is a Σ_2 formula that determines membership in U, making U an open Σ_2 class. On the other hand, the fact that A is not $0'$-computable will guarantee that U is not a Σ_2 open, because in this case A would be $0'$-computable as follows: To determine $A(n)$, enumerate basic open sets of U until you've either entirely covered $\{B \mid B(n) = 0\}$ or $\{B \mid B(n) = 1\}$. This happens in finite time because each of these subsets of Cantor space is compact, and one of them (the one which does not contain A) is contained in U.

The definition of A requires the following lemma, whose proof we relegate to below:

Lemma 14. There is a computable binary tree T that admits a unique path A that does not terminate in infinitely many zeros. Moreover, A is not $0'$-computable.

With this lemma: The class $\{A\}$ is easily seen to be a Π_2 class:

$$\{A\} = \{\, B \in 2^\omega \mid \forall_x (A\restriction x \text{ is in } T) \wedge \forall_x \exists_y (y > x \wedge A(x) = 1) \,\}.$$

Thus, the complement $2^\omega \setminus \{A\}$ is a Σ_2 class. ■

Proof (Lemma 14): We shall diagonalize against every limit computable function. To this effect, let λ_{en} be a partial computable universal collection of sequences of $0,1$-valued functions, with $\lambda_e(x) := \lim_n \lambda_{en}(x)$ being an enumeration of limit-computable $0,1$-valued functions. We will outline the way in which we diagonalize against λ_0, then explain how the process proceeds recursively.

The top of the tree looks as follows, with the left starting out as a path containing only zeros, and the right branch starting with 1, followed by a tree that we call T_0 whose purpose will be to diagonalize against λ_1.

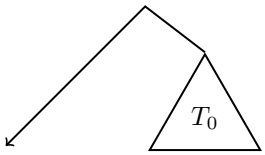

Figure 1: The start of the construction of T.

Now, in parallel with making the tree grow, we evaluate $\lambda_{0n}(0)$ for larger and larger values of n. More precisely, we will in parallel evaluate $\lambda_{0n}(0)$ for *every* $n \in \mathbb{N}$ in parallel, until this expression halts for some value of $n = n_0$.

If $\lambda_{0n_0}(0)\!\downarrow = 0$, we continue as we were, but we shall now only evaluate $\lambda_{0n}(0)$ for $n > n_0$. On the other hand, if $\lambda_{0n_0}(0)\!\downarrow = 1$, we *suspend* the construction of T_0, in a sense to be made precise soon. At this point, on the left hand side, we branch off and do something similar to what was done at the origin: On the left, we keep appending just 0, and on the right we construct a new subtree T_1 whose purpose is to diagonalize against λ_1.

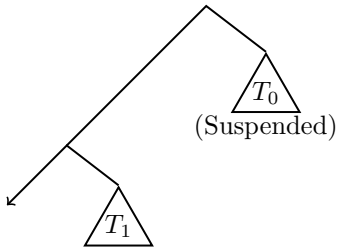

Figure 2: After we've guessed $\lambda_0(0) = 1$.

After this point, we continue evaluating $\lambda_{0n}(0)$ in parallel for $n > n_0$. If we find $\lambda_{0n}(0)\downarrow = 1$, we continue the process as we were. On the other hand, if we find $\lambda_{0n}(0)\downarrow = 0$, for $n = n_1$ say, we will switch back to T_0. More concretely, we *permanently* suspend the construction of T_1, and *un*suspend the construction of T_0. Again, the meaning of suspension will be defined shortly.

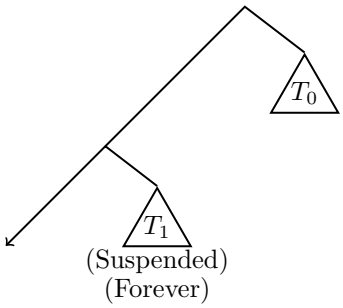

Figure 3: After we've guessed $\lambda_0(0) = 1$, followed by $\lambda_0(0) = 0$.

The process continues, evaluating $\lambda_{0n}(0)$ for $n > n_1$. If we find a result of 0, we continue the process as we were. Otherwise, we again suspend T_0 and begin the construction of a *new* tree T_2on the left branch.

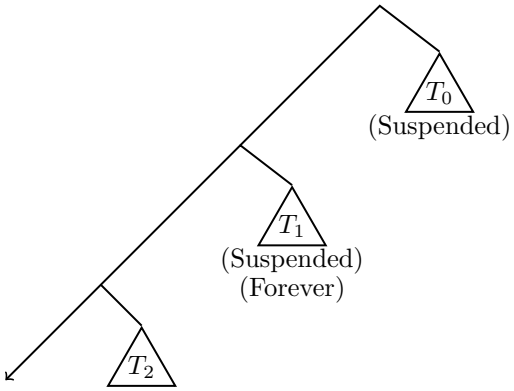

Figure 4: After we've guessed $\lambda_0(0) = 1$, then 0, then 1 again.

This process continues indefinitely. We will now explain the construction of the subtrees T_i. They are constructed the exact same way, only with the following differences:

- The trees we have been referring to so far should be built as to diagonalize against λ_1 instead of λ_0. In turn, they will have their own subtrees, which should be built as to diagonalize against λ_2, and so on.

- In particular, the diagonalization must take the tree's position into account. In other words, T_0 will be keeping track of $\lambda_1(1)$, but if T_1 has its root node at σ, it needs to keep track of $\lambda_1(|\sigma|)$, and so on.
- Finally, there is the matter of suspension, which we now explain.

As we said, the trees T_i should be constructed recursively in the same way as the big tree we've been describing, except that these subtrees may be told to suspend sometime during the construction. This means that all branching and function evaluation is put on pause, and while this happens, every currently-hanging leaf is to continue growing to the left. This means that a tree that is left suspended forever will have a finite fixed number of paths, all of which terminate in an infinite string of zeros. On the other hand, a tree (in the above case, only T_0, but this will apply to other trees that are built from the recursion) may eventually be told to "unsuspend". This simply means that it should resume diagonalizing and branching from where it left off.

Now that the construction of the tree – let's call it T – is complete, let us show that it satisfies the required properties. It is clearly computable, so we show that it admits exactly one path A that does not terminate in all zeros, and that this path A is not $0'$-computable.

To describe the path A, we begin with the following observation. Either the tree changes its mind about $\lambda_0(0)$ infinitely often, or it does not. In the former case, we see that T_0 (as above) is the only subtree that is left unsuspended for an infinite amount of time (though not consecutively). In the latter case, it will eventually stop alternating suspensions, and thus there is a subtree T_i that is left unsuspended for an infinite amount of time. In either case, there is a unique subtree T_i that is left unsuspended for infinite time, and our path A starts by going to the root of that subtree. Note that any path that does not do the same is forced by construction to terminate in an infinite string of zeros!

In turn, a similar reasoning shows that T_i admits exactly one sub-subtree that is left unsuspended for an infinite amount of time, and so on; at each step, the path A follows the subtree that is left unsuspended for an infinite amount of time.

Now that we've shown that A is the only path that does not terminate in an infinite string of zeros,[2] we turn to proving that A is not $0'$-computable.

Suppose that A was $0'$-computable, or equivalently limit-computable. Suppose that $A = \lambda_n$. Let σ be the root of the (sub)^{n}tree that A passes through. Then, $q = \lambda_n(|\sigma|)$ is well-defined, and this subtree is given infinitely much time to approximate it, whence the tree's approximation of $\lambda_n(|\sigma|)$ will eventually converge. Thus, every subtree below σq is suspended, and will stay suspended forever. Thus, A must follow $\sigma(1-q)$, which contradicts the assumption that $A = \lambda_n$. The proof is thus complete. ■

Regardless of the above result, it turns out that any Σ_n class can be (effectively) approximated arbitrarily closely from the outside by Σ_n opens. This is precisely what is necessary to ensure that (cf. §6.8 of [1]) the notions of Martin-Löf test corresponding to each of these collections of classes provide the same notion of randomness.

Instead of directly proving the full result, we start with an attempt at providing intuition by manually doing the cases $n = 1$, 2, 3, and 4.

Proposition 15. Given a Σ_1 class $\mathcal{C}$ and a rational $\varepsilon > 0$, there is a $\Sigma_1$open $U \supseteq \mathcal{C}$ so that $\mu(U) < \mu(\mathcal{C}) + \varepsilon$.

Proof: By Proposition 11, just set $U = \mathcal{C}$. ■

Proposition 16. Given a Σ_2 class $\mathcal{C}$ and a rational $\varepsilon > 0$, there is a Σ_2 open $U \supseteq \mathcal{C}$ so that $\mu(U) < \mu(\mathcal{C}) + \varepsilon$.

Proof: By Remark 7, we can write $\mathcal{C}$ in the form

$$\mathcal{C} = \bigcup_x \bigcap_y U^c_{\sigma(x,y)},$$

[2]Note that at each subtree we descend to we gain a 1, which is why A does not end with only zeros. Another way to prove this is to note that if ended with a string of zeros, it would be computable and therefore $0'$-computable, contrary to what we are about to show.

where x ranges over the natural numbers and y ranges over some computable set that depends uniformly on x. Our approach will be as follows: We know that each $U_{\sigma(x,y)}$ is actually clopen, with $U^c_{\sigma(x,y)}$ being the union of U_τ over all τ of same size as $\sigma(x,y)$ that disagree with it. As such, for each fixed x, $F_x := \cap_y U^c_{\sigma(x,y)}$ can be approximated from the outside by open sets by considering the partial (finite) intersections. We will show that this can be done uniformly with good control of the measure:

Lemma 17. Given a Π_1 class F and a rational $\varepsilon > 0$, there is a $0'$-effective open $U \supseteq F$ so that $\mu(U) < \mu(F) + \varepsilon$.

Once we have established Lemma 17, the proof of Proposition 16 follows as such: Approximate each F_x by a $0'$-effective open set $U_x = \cup_y U_{\nu(x,y)}$ within $2^{-x-1}\varepsilon$ measure, and elementary measure estimates yield that

$$U = \bigcup_{x,y} U_{\nu(x,y)} \tag{1}$$

is the desired open approximation of $\mathcal{C}$. ■

It remains only to establish Lemma 17.

Proof (Lemma 17): Let $F = \cap_y U^c_{\sigma(y)}$, and suppose $\varepsilon > 0$ has been given. Define $F^k = \cap_{y<k} U^c_{\sigma(y)}$ as the k-th approximation of F. The function $k \mapsto \mu(F^k)$ is easily seen to be computable,[3] as well as decreasing, so it furnishes an approximation of $\mu(F)$ from above. Moreover, using $0'$, we can find approximations from below. More precisely, given a rational q we ask "will $\mu(F^k)$ ever be below q?", and if the answer is "no" we now know that $\mu(F) \geq q$. As such, $0'$ is able to find some k such that $\mu(F) \geq \mu(F^k) - \varepsilon$. This F^k is an effective open, and so furnishes the desired approximation. ■

Remark 18. This proof of Lemma 17 actually proves the following stronger statement: Given $F \in \Pi_1$ and ε, there is a *computable* (cl)open approximating F from the outside within ε measure. However, the process is only $0'$-effective. In other words, in Equation (1), for each fixed value of x the set of valid indices $\langle x, y\rangle$ and the function $\nu(x,y)$ on these indices are both computable, but not uniformly in x; the entire aggregation is only $0'$-computable.

Proposition 19. Given a Σ_3 class $\mathcal{C}$ and a rational $\varepsilon > 0$, there is a Σ_3 open $U \supseteq \mathcal{C}$ so that $\mu(U) < \mu(\mathcal{C}) + \varepsilon$.

Proof: By Remark 7, write

$$\mathcal{C} = \bigcup_x \bigcap_y \bigcup_z U_{\sigma(x,y,z)}.$$

As with Proposition 16, the main step is to approximate each $F_x = \cap_y \cup_z U_{\sigma(x,y,z)}$ from the outside by an open set. In a similar fashion to the proof thereof, we will want to look at the partial approximations $F^k_x = \cap_{y<k} \cup_z U_{\sigma(x,y,z)}$, each of which is open, and find a ($0''$-effective) value of k for which F^k_x has at most $2^{-x-1}\varepsilon$ excess measure.

For a fixed value of k, we can write F^k_x as a computable union of opens, say $\cup_w U_{\sigma'(x,k,w)}$. Unlike in the proof of Lemma 17, we cannot compute $\mu(F^k_x)$, and indeed it will generally not be a rational. However, there is a computable sequence $\{m_i\}_{i\in\mathbb{N}}$ that approximates $\mu(F^k_x)$ from below. This means that there is a $0'$-computable way to approximate $\mu(F^k_x)$ from above, namely, for a given $q \in \mathbb{Q}$, ask the Halting Problem if m_i ever exceeds q.

By enumerating the values of $q \in \mathbb{Q}$ that are above $\mu(F^k_x)$ for *some* $k \in \mathbb{N}$, that is, the set

$$\{\, q \in \mathbb{Q} \mid \exists_k q > \mu(F^k_x) \,\},$$

we can provide a sequence $\{q_i\}_{i\in\mathbb{N}}$ that approximates $\mu(F_x) = \mu(\cap_{k\in\mathbb{N}} F^k_x)$ from above. Finally, $0''$ can use the "does this sequence ever go below $p \in \mathbb{Q}$" trick to find a sequence $\{p_i\}_{i\in\mathbb{N}}$ that approximates $\mu(F_x)$ from

[3] This requires writing F^k as a disjoint (finite) union $\cup_z U_{\sigma'(z)}$ and adding up $\sum_z 2^{-|\sigma'(z)|}$.

below. Thus, $0''$ can compute arbitrary approximations to $\mu(F_x)$, and moreover it can find a value of k for which $\mu(F_x^k) < \mu(F_x) + \delta$ for a given δ: Simply find an approximation M of $\mu(F_x)$ that is sure to be within $\delta/2$ of the real thing, followed by seeking out a value of k for which $M + \delta/2$ is an approximation of $\mu(F_x^k)$ by excess.

To conclude the proof: Let $k(x)$ be a $0''$-computable function so that $\mu(F_x^{k(x)}) < \mu(F_x) + 2^{-x-1}\varepsilon$. Then, $U = \cup_x F_x^{k(x)}$ is a $0''$-effective open set that approximates $\mathcal{C}$ from above with at most ε excess measure, as was desired. ■

Proposition 20. Given a Σ_4 class $\mathcal{C}$ and a rational $\varepsilon > 0$, there is a Σ_4 open $U \supseteq \mathcal{C}$, so that $\mu(U) < \mu(\mathcal{C}) + \varepsilon$.

Proof: By Remark 7, write

$$\mathcal{C} = \bigcup_x \bigcap_y \bigcup_z \bigcap_w U^c_{\sigma(x,y,z,w)}.$$

This high in the hierarchy, an induction starts to reveal itself. In order to simplify the problem down to something managable, we would like to approximate $Q_{xy} = \cup_z \cap_w U^c_{\sigma(x,y,z,w)}$ by an open superset V_{xy}. This can be done $0'$-effectively by Proposition 16, and the sets V_{xy} are Σ_2 opens. It should sound plausible – and we will do the details shortly – that if we're careful in how we go about it, we can set things up so that

$$\mu\left(\bigcap_y V_{xy}\right) < \mu\left(\bigcap_y Q_{xy}\right) + 2^{-x-1}\varepsilon = \mu\left(\bigcap_y \bigcup_z \bigcap_w U^c_{\sigma(x,y,z,w)}\right) + 2^{-x-1}\varepsilon, \tag{2}$$

and therefore so that

$$\mu\left(\bigcup_x \bigcap_y V_{xy}\right) < \mu\left(\bigcup_x \bigcap_y \bigcup_z \bigcap_w U^c_{\sigma(x,y,z,w)}\right) + \varepsilon = \mu(\mathcal{C}) + \varepsilon.$$

Now the result follows by relativization of Proposition 16. More precisely, we have in our hands a $\Sigma_2^{0''}$ class, namely $\mathcal{C}' = \cup_x \cap_y V_{xy}$, and Proposition 16 gives us a $\Sigma_2^{0''}$ open that approximates $\mathcal{C}'$ from above with at most ε excess measure, and therefore approximates $\mathcal{C}$ from above with at most 2ε measure. The result therefore follows by noticing that by Definition 9 a $\Sigma_2^{0''}$ open is the same as a $(0'')'$-computable open, i.e. a $0^{(3)}$-computable open, i.e. a Σ_4 open.

The only thing remaining to finish the proof is to be a little more careful about the definition of V_{xy}. The following argument is a little clumsy, but it's straightforward. Set V_{xy} to approximate Q_{xy} by at most $\delta_y = 2^{-x-y-2}\varepsilon$. In other words, we have $V_{xy} = Q_{xy} \cup E_{xy}$, with $\mu(E_{xy}) < \delta_y$. Then, we have the set-theoretic inclusion

$$\bigcap_y V_{xy} \subseteq \bigcap_y Q_{xy} \cup \bigcup_y E_{xy}.$$

As a consequence, we have the inequality

$$\mu\left(\bigcap_y V_{xy}\right) \leq \mu\left(\bigcap_y Q_{xy}\right) + \sum_y \mu(E_{xy}) = \mu\left(\bigcap_y Q_{xy}\right) + \sum_y \delta_y = \mu\left(\bigcap_y Q_{xy}\right) + 2^{-x-1}\varepsilon.$$

This is sufficient to establish Equation 2, and the proof is complete. ■

We are finally in a position to establish the general result.

Theorem 21. Given a Σ_n class $\mathcal{C}$ and a rational $\varepsilon > 0$, there is a Σ_n open $U \supseteq \mathcal{C}$ so that $\mu(U) < \mu(\mathcal{C}) + \varepsilon$. This procedure is $0^{(n-1)}$-effective.

Proof: The proof follows by induction, with step size equal to 2. The cases $n = 1$ and $n = 2$ are Propositions 15 and 16. For the induction step, assume that the case for n has been proven, and we shall now prove the case $n + 2$.

To proceed, write the Σ_{n+2} class $\mathcal{C}$ as

$$\mathcal{C} = \bigcup_x \bigcap_y Q_{xy},$$

where $\{Q_{xy}\}_{x,y\in\mathbb{N}}$ is an effective family of Σ_n classes. As in the proof of Proposition 20, approximate each of these $0^{(n-1)}$-effectively by a Σ_n open V_{xy}, with excess measure less than $2^{-x-y-2}\varepsilon$. Then, following a relativized version of the proof of Proposition 20, we obtain a $0^{(n+1)}$-effective approximation of $\mathcal{C}$ with excess measure less than ε, which proves the desired result. ■

References

[1] Rodney G. Downey and Denis R. Hirschfeldt. *Algorithmic randomness and complexity.* Theory and Applications of Computability. Springer, New York, 2010.