FUNDAMENTA
MATHEMATICAE
195 (2007)

The effective Borel hierarchy

by

M. Vanden Boom (Notre Dame, IN)

Abstract. Let K be a subclass of $\mathrm{Mod}(\mathcal{L})$ which is closed under isomorphism. Vaught showed that K is $\mathbf{\Sigma}_\alpha$ (respectively, $\mathbf{\Pi}_\alpha$) in the Borel hierarchy iff K is axiomatized by an infinitary Σ_α (respectively, Π_α) sentence. We prove a generalization of Vaught's theorem for the effective Borel hierarchy, i.e. the Borel sets formed by union and complementation over c.e. sets. This result says that we can axiomatize an effective Σ_α or effective Π_α Borel set with a computable infinitary sentence of the same complexity. This result yields an alternative proof of Vaught's theorem via forcing. We also get a version of the pull-back theorem from Knight *et al.* which says if Φ is a Turing computable embedding of $K \subseteq \mathrm{Mod}(\mathcal{L})$ into $K' \subseteq \mathrm{Mod}(\mathcal{L}')$, then for any computable infinitary sentence φ in the language $\mathcal{L}$, we can find a computable infinitary sentence φ^* in $\mathcal{L}'$ such that for all $\mathcal{A} \in K$, $\mathcal{A} \models \varphi^*$ iff $\Phi(\mathcal{A}) \models \varphi$, where φ^* has the same complexity as φ.

1. Introduction. In descriptive set theory, we study definable sets and functions in Polish spaces, i.e. complete, separable topological spaces that admit a metric. The *Borel sets* in a Polish space frequently receive attention. These sets are constructed from the basic open sets by countable union and complementation. We form a hierarchy within the class of Borel sets, specifying the $\mathbf{\Sigma}_\alpha$ and $\mathbf{\Pi}_\alpha$ Borel sets for each ordinal α, based on the operations used to build up the set. Where a set lies in the Borel hierarchy is a reflection of the complexity of the set.

We work in the context of a particular Polish space known as $\mathrm{Mod}(\mathcal{L})$, the class of structures over some fixed countable language $\mathcal{L}$, where each structure has universe ω. López-Escobar [7] showed that a subclass K, closed under isomorphism, is Borel iff K is axiomatized by an infinitary sentence σ, where the conjunctions and disjunctions in σ are over countable sets. Vaught [8] improved on this result by showing that if K is $\mathbf{\Sigma}_\alpha$ (respec-

2000 *Mathematics Subject Classification*: Primary 03E15; Secondary 03C57, 03C75, 03D55.

Key words and phrases: effective Borel set, Vaught's theorem, computable embedding.

The author would like to thank Dr. Julia Knight at the University of Notre Dame for her feedback and support with this paper.

tively, $\mathbf{\Pi}_\alpha$) in the Borel hierarchy for some ordinal α, then σ can be taken to be a Σ_α (respectively, Π_α) sentence. That is, we can axiomatize an invariant $\mathbf{\Sigma}_\alpha$ or $\mathbf{\Pi}_\alpha$ Borel set with a sentence that is of the same complexity as the Borel set we are attempting to describe. We refer to this result as *Vaught's theorem* throughout this paper.

In this paper, we focus primarily on the *effective Borel sets*, which are built up from the basic sets by c.e. union and complementation. To make precise the c.e. unions, we assign indices to the effective Borel sets. We also specify the corresponding effective Borel hierarchy. We then prove a generalization of Vaught's theorem, which deals with the effective Borel hierarchy and a notion of "within" described below. This is our main result.

THEOREM 1.1. *Let $K \subseteq \mathrm{Mod}(\mathcal{L})$, where K is closed under isomorphism. Suppose that for some $\alpha \geq 1$, i is an index for an effective Σ_α (respectively, Π_α) set B_i such that $B_i \cap K$ is closed under isomorphism. Then we can effectively find (an index for) a computable Σ_α (respectively, Π_α) sentence σ such that $B_i = \mathrm{Mod}(\sigma)$ within K, i.e.*

$$B_i \cap K = \mathrm{Mod}(\sigma) \cap K,$$

where $\mathrm{Mod}(\sigma) = \{\mathcal{A} : \mathcal{A} \models \sigma\}$.

The effective version of Vaught's theorem follows immediately from this result.

THEOREM 1.2 (Effective version of Vaught's theorem). *Let $K \subseteq \mathrm{Mod}(\mathcal{L})$, where K is closed under isomorphism. Let $\alpha \geq 1$. If K is an effective Σ_α (respectively, Π_α) set, then we can effectively find (an index for) a computable Σ_α (respectively, Π_α) sentence σ such that K is axiomatized by σ.*

This theorem says that we can axiomatize an effective Σ_α or effective Π_α Borel set that is closed under isomorphism with a computable infinitary sentence of the same complexity. Informally, this means that the conjunctions and disjunctions in the sentence are over c.e. sets.

We also obtain some additional results. By relativizing the proof of the theorem above, we obtain an alternative proof of Vaught's theorem. This fact was noted by Knight. Unlike Vaught's category argument in [8], however, we use a (relativized) forcing argument. These types of proof are known to be related; Kechris mentions this connection and the related terminology in [5]. It remains to be seen whether there are essential or interesting differences in these proofs.

We also get the following version of a result from [6], called the pull-back theorem. For classes $K \subseteq \mathrm{Mod}(\mathcal{L})$, $K' \subseteq \mathrm{Mod}(\mathcal{L}')$ closed under isomorphism such that K Turing computably embeds into K' via a computable operator Φ, we can effectively transfer from a sentence φ in the language $\mathcal{L}'$ to a sentence φ^* in the language $\mathcal{L}$ such that for all $\mathcal{A} \in K$, $\Phi(\mathcal{A}) \models \varphi$ iff

$\mathcal{A} \models \varphi^*$. In [6], we prove a somewhat more general result for structures with universe a subset of ω, possibly finite. Here, however, we use Theorem 1.1 to obtain a restricted version of the pull-back theorem for a Turing computable embedding that takes $K \subseteq \mathrm{Mod}(\mathcal{L})$ to $K' \subseteq \mathrm{Mod}(\mathcal{L}')$. We refer the interested reader to Section 4 or [6] for definitions of these terms.

In Section 2, we give some background on ordinal notations, which we then use to index computable infinitary formulas and the effective Borel sets. In Section 3, we give a proof, via forcing, of the effective version of Vaught's theorem. We also observe that Vaught's original result can be obtained by relativizing our proof. Finally, in Section 4, we give an application of the theorem to obtain a pull-back theorem for classes of structures with universe ω.

We conclude this section with some general comments on $\mathrm{Mod}(\mathcal{L})$, since this is the context for our work.

1.1. *Background on* $\mathrm{Mod}(\mathcal{L})$. Throughout this paper, we work with a fixed predicate language $\mathcal{L}$. We assume that $\mathcal{L}$ is relational, which simplifies the topology of $\mathrm{Mod}(\mathcal{L})$, described below. However, we do not actually lose anything with this restriction, since anything we could say with an operational language could be said with a corresponding relational language. We assume further that $\mathcal{L}$ is computable, which means the set of relational symbols in $\mathcal{L}$ is computable and we can effectively determine the arity of each symbol. We consider the structures over this language with universe ω, denoted $\mathrm{Mod}(\mathcal{L})$. We let $\mathrm{Mod}(\sigma)$ be the subset of $\mathrm{Mod}(\mathcal{L})$ such that $\mathcal{A} \in \mathrm{Mod}(\sigma)$ iff $\mathcal{A} \models \sigma$.

Let $(\alpha_n)_{n\in\omega}$ be a computable list of all atomic sentences in the language $\mathcal{L} \cup \omega$, except for sentences of the form $a = b$ for $a, b \in \omega$. By omitting sentences $a = a$, which are true for all a, and sentences $a = b$, which are false for $b \neq a$, we have complete freedom over the α_n. Thus, each function $\sigma \in 2^\omega$ asserting that each α_n is true or false, corresponds with a possible structure in $\mathrm{Mod}(\mathcal{L})$. In fact, we can think of these meaningful sentences as the propositional variables of a corresponding propositional language. We define a topology on the class of structures identified by these functions, letting the basic open neighborhoods be the sets N_f corresponding to finite functions $f \in 2^{<\omega}$, where $N_f = \{\sigma \in 2^\omega : \sigma \supseteq f\}$. With this topology, $\mathrm{Mod}(\mathcal{L})$ is a Cantor space, which is an example of a Polish space [5].

Within $\mathrm{Mod}(\mathcal{L})$, we restrict our attention to subclasses $K \subseteq \mathrm{Mod}(\mathcal{L})$ which are closed under isomorphism. Equivalently, we say K is invariant, or, more precisely, invariant under the action of S_∞, the group of permutations of ω. In [8], Vaught also worked with invariant classes K, but in the more general context of a group G acting on an arbitrary Polish space X, rather than S_∞ and $\mathrm{Mod}(\mathcal{L})$ specifically.

2. Indexing the effective Borel sets. Our goal is to describe what it means for a class of structures, a subset of $\mathrm{Mod}(\mathcal{L})$, to be effective Σ_α or effective Π_α in the effective Borel hierarchy. Recall that every Borel set is constructed by taking countable unions and complements (or, equivalently, countable intersections and complements) of some basic sets. Traditionally, we denote the class of sets at each level α in this hierarchy using bold-face notation, $\mathbf{\Sigma}_\alpha$ or $\mathbf{\Pi}_\alpha$. Effective Borel sets, however, are constructed using strictly c.e. unions and complements, and classes in the effective Borel hierarchy are written using light-face notation, Σ_α or Π_α. Not surprisingly, we begin by looking at the lowest level of the hierarchy, and then build up from there. In order to move up in the hierarchy, we need a way to index the effective Borel sets so, at say the Σ_α level, we can include all of the sets obtained by c.e. union of effective Π_β sets for $\beta < \alpha$.

There are various systems for assigning indices to the Borel sets; in the literature these indices are often called *Borel codes* [4]. In this section, we describe a method for indexing each effective Borel set, which allows us to specify the class of effective Σ_α and effective Π_α Borel sets, for computable ordinals α. The index has the same purpose as the Borel code mentioned above, but we must be more careful in picking out only the sets formed by c.e. unions and complements. A nice feature of the indexing scheme we describe here is that it parallels the method used in [1] to index computable infinitary formulas. After reviewing Kleene's system of ordinal notations, we present systems for indexing the effective Borel sets and the computable infinitary formulas.

2.1. *Ordinal notation.* Let $\mathcal{O}$ denote the set of notations in Kleene's system. Each *notation* $a \in \mathcal{O}$ represents a particular ordinal α, written $|a|_{\mathcal{O}} = \alpha$. These notations, or names, for the ordinals, specify how the ordinal is built up from below. An informal definition of the set $\mathcal{O}$, the function $|\ |_{\mathcal{O}}$, and the strict partial ordering $<_{\mathcal{O}}$ on $\mathcal{O}$ is given below (see [1] for more information).

1. Let 1 be the notation for the ordinal 0, i.e. $|1|_{\mathcal{O}} = 0$.
2. If b is a notation for the ordinal α, then 2^b is a notation for the successor ordinal $\alpha + 1$ and $b <_{\mathcal{O}} 2^b$. Moreover, for all notations c such that $c <_{\mathcal{O}} b$, $c <_{\mathcal{O}} 2^b$ as well.
3. If φ_e is a total computable function from ω to $\mathcal{O}$ such that, for all $n \in \omega$, $\varphi_e(n) <_{\mathcal{O}} \varphi_e(n+1)$, and $\lim_n |\varphi_e(n)|_{\mathcal{O}} = \alpha$, then $3 \cdot 5^e$ is a notation for the limit ordinal α and $\varphi_e(n) <_{\mathcal{O}} 3 \cdot 5^e$ for all n. Moreover, for all notations c such that $c <_{\mathcal{O}} \varphi_e(n)$, $c <_{\mathcal{O}} 3 \cdot 5^e$.

An infinite ordinal will have infinitely many different notations in Kleene's system. However, we use all possible notations because there is no reason to prefer one notation over another. The exact conventions used to distinguish

between notations for successor and limit ordinals are not important for our purposes, but we do make use of the partial ordering $<_{\mathcal{O}}$.

2.2. *Effective Borel sets.* With the notation system in place, we define the index sets for each notation $a \in \mathcal{O}$.

DEFINITION 2.1. The *index sets for the effective Borel sets*, denoted BS_a^Σ and BS_a^Π, are defined as follows for each $a \in \mathcal{O}$.

1. $BS_1^\Sigma = BS_1^\Pi = \{\ulcorner\varphi\urcorner : \varphi \text{ is a finitary quantifier-free sentence}\}$, i.e. the set of Gödel numbers for finitary quantifier-free sentences which are the result of substituting a tuple of constants for the finitely many variables in a finitary quantifier-free $\mathcal{L}$-formula.
2. Let $|a|_{\mathcal{O}} > 0$. Then
   (a) $BS_a^\Sigma = \{\langle \Sigma, a, e\rangle : e \in \omega\}$,
   (b) $BS_a^\Pi = \{\langle \Pi, a, e\rangle : e \in \omega\}$,

   where Σ and Π are coded as 0 and 1, respectively, and $\langle x, y, z\rangle$ is the code for the triple (x, y, z) under some standard coding scheme.

Next, we describe the effective Borel sets corresponding to indices in BS_a^Σ and BS_a^Π. Notice that we make use of strictly c.e. unions and intersections to build up each effective Borel set (recall that W_e denotes the eth c.e. set).

DEFINITION 2.2. Let i be an index for an effective Borel set. Then the *effective Borel set B_i with index i* is defined as follows.

1. If $i \in BS_1^\Sigma = BS_1^\Pi$, then B_i is the set of $\mathcal{B} \in \mathrm{Mod}(\mathcal{L})$ such that $\mathcal{B}$ satisfies the sentence with Gödel number i. We write this as $B_i = \mathrm{Mod}(\psi)$ where $\ulcorner\psi\urcorner = i$.
2. Assume $|a|_{\mathcal{O}} > 0$.
   (a) If $i \in BS_a^\Sigma$, then $i = \langle \Sigma, a, e\rangle$ for some $e \in \omega$ and B_i is the union of the sets B_j for $j \in W_e \cap \bigcup_{b<_{\mathcal{O}} a} BS_b^\Pi$.
   (b) If $i \in BS_a^\Pi$, then $i = \langle \Pi, a, e\rangle$ for some $e \in \omega$ and B_i is the intersection of the sets B_j for $j \in W_e \cap \bigcup_{b<_{\mathcal{O}} a} BS_b^\Sigma$.

For $|a|_{\mathcal{O}} = \alpha$ and $i \in BS_a^\Sigma$, we say B_i is *effective* Σ_α; likewise, if $i \in BS_a^\Pi$, then we say B_i is *effective* Π_α.

2.3. *Computable infinitary formulas.* We also need Kleene's system of ordinal notations in order to define *computable infinitary formulas* in a predicate language $\mathcal{L}$. We follow the definition in [1]. Informally, we start with the *basic* formulas, which are simply the atomic formulas and the negations of atomic formulas, and build up from there using conjunctions and disjunctions, and existential and universal quantifiers, as usual. However, we also allow infinite (over a c.e. set) disjunctions and conjunctions in these formulas.

We introduce this type of formula formally here since we frequently work with computable infinitary formulas in the remainder of the paper. We also mention some simple facts involving negation. Once again, we begin by defining a computable set of indices for each $a \in \mathcal{O}$.

DEFINITION 2.3. The *index sets for computable infinitary formulas*, denoted S_a^Σ and S_a^Π, are defined as follows for each $a \in \mathcal{O}$.

1. $S_1^\Sigma = S_1^\Pi = \{\ulcorner\varphi\urcorner : \varphi \text{ is a finitary quantifier-free formula}\}$.
2. Let $|a|_\mathcal{O} > 0$. Then
   (a) $S_a^\Sigma = \{\langle \Sigma, a, \overline{x}, e\rangle : \overline{x} \text{ is a tuple of variables}, e \in \omega\}$,
   (b) $S_a^\Pi = \{\langle \Pi, a, \overline{x}, e\rangle : \overline{x} \text{ is a tuple of variables}, e \in \omega\}$.

For each index i, we describe its corresponding computable infinitary formula φ_i.

DEFINITION 2.4. Let i be an index for a computable infinitary formula. Then the *computable infinitary formula* $\varphi_i(\overline{x})$ *with index* i is defined as follows.

1. If $i \in S_1^\Sigma = S_1^\Pi$, then the computable Σ_0 and computable Π_0 formula φ_i is the finitary quantifier-free formula in the language $\mathcal{L}$ with Gödel number i. We assume that these formulas are in complete disjunctive normal form, that is to say, each disjunct has the form $\pm\alpha_1 \& \cdots \& \pm\alpha_n$ where $\alpha_1, \ldots, \alpha_n$ is a complete list of atomic subformulas.
2. Assume $|a|_\mathcal{O} > 0$.
   (a) If $i \in S_a^\Sigma$, then $i = \langle \Sigma, a, \overline{x}, e\rangle$ for some tuple of variables $\overline{x}$ and $e \in \omega$, so $\varphi_i(\overline{x})$ is the disjunction of the formulas $(\exists \overline{u})\, \psi_j(\overline{x}, \overline{u})$ for $j \in W_e \cap \bigcup_{b<_\mathcal{O} a} S_b^\Pi$, where the third component of j is a tuple of variables $\overline{v}$, and $\overline{u}$ consists of the variables in $\overline{v}$ but not in $\overline{x}$.
   (b) If $i \in S_a^\Pi$, then $i = \langle \Pi, a, \overline{x}, e\rangle$ for some tuple of variables $\overline{x}$ and $e \in \omega$, so $\varphi_i(\overline{x})$ is the conjunction of the formulas $(\forall \overline{u})\, \psi_j(\overline{x}, \overline{u})$ for $j \in W_e \cap \bigcup_{b<_\mathcal{O} a} S_b^\Sigma$, where the third component of j is a tuple of variables $\overline{v}$, and $\overline{u}$ consists of the variables in $\overline{v}$ but not in $\overline{x}$.

In a similar fashion, we can index and describe computable infinitary formulas in a propositional language P, for P a computable set of propositional variables. We start with computable sets of indices PS_a^Σ and PS_a^Π for $a \in \mathcal{O}$. We build up from the computable Σ_0 and computable Π_0 propositional formulas (which are simply the finitary propositional formulas) using c.e. disjunctions and conjunctions as expected.

Because the negations in a given formula φ are all "inside" (appearing only in the finitary quantifier-free sentences), we must define a formula $\text{neg}(\varphi)$ of the dual form that is logically equivalent to the negation of φ.

DEFINITION 2.5. Let φ be a computable infinitary formula. Then $\mathrm{neg}(\varphi)$ is defined inductively as follows:

1. Suppose φ is finitary quantifier-free. Then $\mathrm{neg}(\varphi)$ is the formula in complete disjunctive normal form (with the same atomic subformulas) that is logically equivalent to $\neg\varphi$.
2. Suppose φ is computable Σ_α, a c.e. disjunction of formulas of the form $(\exists \overline{u})\,\psi$. Then $\mathrm{neg}(\varphi)$ is the conjunction of the formulas $(\forall \overline{u})\,\mathrm{neg}(\psi)$.
3. Suppose φ is computable Π_α, a c.e. conjunction of formulas of the form $(\forall \overline{u})\,\psi$. Then $\mathrm{neg}(\varphi)$ is the disjunction of the formulas $(\exists \overline{u})\,\mathrm{neg}(\psi)$.

We can also show the following simple fact, which is true since we have specified that the finitary quantifier-free formulas should always be written in complete disjunctive normal form.

PROPOSITION 2.6. *Let φ be a computable infinitary formula as defined above. Then* $\mathrm{neg}(\mathrm{neg}(\varphi)) = \varphi$.

Proof. The proof is a straightforward induction. ∎

3. Effective version of Vaught's theorem. A well-known result in descriptive set theory, due to López-Escobar [7] in the 1960s, says that an invariant set K is Borel iff K is the set of all models for some sentence σ. López-Escobar proved that σ is an $\mathcal{L}_{\omega_1\omega}$ sentence (an infinitary sentence in which the conjunctions and disjunctions are over countable sets and there is only finite nesting of quantifiers), but did not locate this sentence at a particular level of complexity.

By introducing special transforms with category quantifiers "for non-meagerly many" and "for co-meagerly many", Vaught [8] was able to show that K is an invariant Σ_α (respectively, Π_α) set in the Borel hierarchy iff $K = \mathrm{Mod}(\sigma)$ for an infinitary Σ_α (respectively, Π_α) sentence σ. His proof is a category argument and the transforms that he used are now known as *Vaught transforms* [5]. As mentioned in the introduction, we refer to this result as Vaught's theorem. This theorem demonstrates the strong connection between the complexity of the sentence axiomatizing an invariant Borel set, and the level of that Borel set in the hierarchy.

It is natural to ask whether a similar result holds in the effective Borel hierarchy. That is, we would like to know if

> K is closed under isomorphism and **effective** Σ_α (respectively, Π_α) in the effective Borel hierarchy iff $K = \mathrm{Mod}(\sigma)$ for σ a **computable** Σ_α (respectively, Π_α) sentence.

One direction is clear: given a subclass $K = \mathrm{Mod}(\sigma)$ for σ a computable Σ_α (respectively, Π_α) sentence, K is an effective Σ_α (respectively, Π_α) Borel set (for $\alpha = 0$, the result is by Definition 2.2, and an inductive argument shows that the result holds for all $\alpha \geq 1$).

For the converse, we actually prove a generalization involving the notion of "within". We say $\mathcal{C} = \mathcal{D}$ *within* K if $\mathcal{C} \cap K = \mathcal{D} \cap K$. We show that if K and $K \cap B_i$ are closed under isomorphism where B_i is an effective Borel set, then we can effectively find a computable infinitary sentence σ such that $B_i = \mathrm{Mod}(\sigma)$ within K. Moreover, if B_i is effective Σ_α or effective Π_α then so is σ. The effective version of Vaught's theorem follows immediately from this result by letting $K = B_i$. The remainder of this section presents the technical details of the forcing argument, culminating with the proof of this generalization and corollaries in Section 3.4.

3.1. *Forcing conditions and forcing language.* We fix an $\mathcal{L}$-structure $\mathcal{A}$ throughout this section and build a generic copy $\mathcal{B}$ of $\mathcal{A}$. Eventually, we will see that the definitions and lemmas are independent of the specific $\mathcal{A}$ used, but for now it helps to think of a particular $\mathcal{A}$ and its generic copy $\mathcal{B}$ under construction. We begin with a set $\mathcal{F}$ of forcing conditions.

Definition 3.1. The set $\mathcal{F}$ of *forcing conditions* is the set of finite partial permutations p of ω. We write p, q, r for elements of $\mathcal{F}$ and we say that q *extends* p if $q \supseteq p$.

We can think of $\mathcal{F}$ as the set of partial isomorphisms from $\mathcal{B}$, which is under construction, to $\mathcal{A}$.

We need an appropriate forcing language, suitable for talking about $\mathcal{B}$. In particular, if we have an effective Borel set B_i, then our forcing language must be able to express $\mathcal{B} \in B_i$. Because we do not need quantifiers in order to express this relationship, we choose a propositional language $P_{\mathcal{L}}$.

Definition 3.2. Let $P_{\mathcal{L}}$ be the propositional language where the propositional symbols are obtained by substituting constants for the finitely many variables in an atomic formula in the predicate language $\mathcal{L}$. The *forcing language* is the set S of computable infinitary propositional formulas in the language $P_{\mathcal{L}}$.

We can verify that this forcing language S has enough expressive power to say $\mathcal{B} \in B_i$, where i is the index for an effective Borel set.

Proposition 3.3. *Let i be an index for an effective Borel set B_i. Then we can effectively find an index j for a computable infinitary propositional formula $\psi_j \in S$ with the meaning $\mathcal{B} \in B_i$, i.e. $\mathcal{B} \models \psi_j$ iff $\mathcal{B} \in B_i$. Moreover, if B_i is effective Σ_α (respectively, Π_α), then ψ_j is computable Σ_α (respectively, Π_α) as well.*

Proof. The proof is by induction on α.

1. Let $\alpha = 0$ and consider an index i for an effective Σ_0 (equivalently, effective Π_0) Borel set. Then $i \in BS_1^\Sigma = BS_1^\Pi$ so i is the Gödel number for a finitary quantifier-free sentence formed by substituting a tuple of constants for the finitely many variables in a finitary quantifier-free $\mathcal{L}$-formula. We let j be the Gödel number for the same sentence, now thought of as a computable Σ_0 (equivalently, Π_0) formula in S built up from the propositional variables of $P_\mathcal{L}$ by finitary conjunctions and disjunctions.

2. Now consider $\alpha \geq 1$ where $|a|_\mathcal{O} = \alpha$ for some $a \in \mathcal{O}$. We assume that the result holds for all $\beta < \alpha$. Suppose i is the index for some effective Σ_α set B_i. Then i has the form $\langle \Sigma, a, e\rangle$ where $|a|_\mathcal{O} = \alpha$, and B_i is the union of the sets B_k for all indices $k \in W_e \cap \bigcup_{b<_\mathcal{O} a} BS_b^\Pi$. Since $k \in BS_b^\Pi$ with $b <_\mathcal{O} a$, B_k is effective Π_β for some $\beta < \alpha$. Thus, for each $k \in W_e \cap \bigcup_{b<_\mathcal{O} a} BS_b^\Pi$, we can apply the inductive hypothesis to effectively find a formula $\psi_{k'} \in S$ with the meaning $\mathcal{B} \in B_k$. We take ψ_j to be the disjunction of the formulas $\psi_{k'}$. This formula ψ_j has the meaning $\mathcal{B} \in B_i$ and is computable Σ_α as desired. Notice that j will be of the form $\langle \Sigma, a, \overline{x}, e'\rangle$ for some appropriate tuple of variables $\overline{x}$ and $e' \in \omega$ (the notation a, however, is the same as in i). Similarly, if i is the index for some effective Π_α set, then i has the form $\langle \Pi, a, e\rangle$ where $|a|_\mathcal{O} = \alpha$, and B_i is the intersection of the sets B_k for all indices $k \in W_e \cap \bigcup_{b<_\mathcal{O} a} BS_b^\Sigma$. We take ψ_j to be the conjunction of the formulas $\psi_{k'} \in S$, where $\psi_{k'}$ has the meaning $\mathcal{B} \in B_k$ as above. ∎

In order to be precise, we would always work with *indices* for both the effective Borel sets and computable infinitary formulas. It is the index (a number) that we can compute with, not the object itself. In the interest of comprehensibility, however, we often blur the distinction between the index and the set or formula. Thus, we will often work directly with the set or formula, knowing that we could switch to talking about the indices if we needed to be more precise.

3.2. *Definition of forcing and basic lemmas.* We are now ready to define forcing and prove the usual lemmas.

Definition 3.4 (Forcing). Let $p \in \mathcal{F}$ and let $\varphi \in S$ be a formula of the forcing language. We define p *forces* φ, written $p \Vdash \varphi$, inductively below. We say p *decides* φ, written $p \parallel \varphi$, if $p \Vdash \varphi$ or $p \Vdash \mathrm{neg}(\varphi)$.

1. Suppose φ is computable Σ_0 and Π_0. Then $p \Vdash \varphi$ if all of the constants in φ are in $\mathrm{dom}(p)$ and p makes φ true in $\mathcal{A}$.
2. Suppose φ is computable Σ_α ($\alpha > 0$) of the form $\bigvee_i \psi_i$ where each ψ_i is computable Π_{β_i} for $\beta_i < \alpha$. Then $p \Vdash \varphi$ if $p \Vdash \psi_i$ for some i.

3. Suppose φ is computable Π_α ($\alpha > 0$) of the form $\bigwedge_i \psi_i$ where each ψ_i is computable Σ_{β_i} for some $\beta_i < \alpha$. Then $p \Vdash \varphi$ if for each i and for each $q \supseteq p$, it is not the case that $q \Vdash \mathrm{neg}(\psi_i)$.

Given a forcing condition $p \in \mathcal{F}$ and a formula $\varphi \in S$ in the forcing language, we have the following basic lemmas, proved using induction on the complexity of the formula φ.

LEMMA 3.5 (Extension). *If $p \Vdash \varphi$ and $q \supseteq p$, then $q \Vdash \varphi$.*

Proof. We proceed by induction on the complexity of φ.

1. First, suppose φ is computable Σ_0 and Π_0; that is, φ is finitary. Then φ has only finitely many propositional symbols and hence only finitely many constants $\overline{b}$ from $|\mathcal{B}|$. We write $\varphi = \varphi(\overline{b})$ to emphasize that φ is an $\mathcal{L}$-sentence with constants from $|\mathcal{B}|$. If $p \Vdash \varphi(\overline{b})$ then $\overline{b} \subseteq \mathrm{dom}(p)$ and p makes $\varphi(\overline{b})$ true in $\mathcal{A}$ by mapping $\overline{b}$ to some tuple $\overline{a}$ of constants from $|\mathcal{A}|$. If $q \supseteq p$ then $\overline{b}$ is also in $\mathrm{dom}(q)$, so $q(b) = p(b)$ for all $b \in \overline{b}$. Thus, q maps $\overline{b}$ to the same $\overline{a}$, thereby making $\varphi(\overline{b})$ true in $\mathcal{A}$ as well. Hence $q \Vdash \varphi$.

2. Suppose φ is computable Σ_α ($\alpha > 0$) of the form $\bigvee_i \psi_i$, where each ψ_i is computable Π_{β_i} for $\beta_i < \alpha$. If $p \Vdash \varphi$, then $p \Vdash \psi_i$ for some i. But if $q \supseteq p$, then $q \Vdash \psi_i$ by the inductive hypothesis, so $q \Vdash \varphi$.

3. Suppose φ is computable Π_α ($\alpha > 0$) of the form $\bigwedge_i \psi_i$, where each ψ_i is computable Σ_{β_i} for $\beta_i < \alpha$. If $p \Vdash \varphi$, then for all i and for all $r \supseteq p$, we do not have $r \Vdash \mathrm{neg}(\psi_i)$. In particular, if $q \supseteq p$, then for all i and for all $r \supseteq q \supseteq p$, it is not the case that $r \Vdash \mathrm{neg}(\psi_i)$. Thus, $q \Vdash \varphi$. ∎

LEMMA 3.6 (Consistency). *It is not the case that $p \Vdash \varphi$ and $p \Vdash \mathrm{neg}(\varphi)$.*

Proof. We proceed by induction on φ.

1. Suppose φ is computable Σ_0 and Π_0. If $p \Vdash \varphi$, then all of the constants in φ are in $\mathrm{dom}(p)$ and p makes φ true in $\mathcal{A}$. Since p cannot also make the negation true in $\mathcal{A}$, it does not force $\mathrm{neg}(\varphi)$. Likewise, if $p \Vdash \mathrm{neg}(\varphi)$, it does not force φ.

2. Suppose φ is computable Σ_α ($\alpha > 0$) of the form $\bigvee_i \psi_i$, where each ψ_i is computable Π_{β_i} for $\beta_i < \alpha$. Then $\mathrm{neg}(\varphi) = \bigwedge_i \mathrm{neg}(\psi_i)$ where the $\mathrm{neg}(\psi_i)$ are computable Σ_{β_i}. Assume by contradiction that $p \Vdash \varphi$ and $p \Vdash \mathrm{neg}(\varphi)$. Since $p \Vdash \varphi$, there is some j such that $p \Vdash \psi_j$. But $p \Vdash \mathrm{neg}(\varphi)$ implies that for all i and for all $q \supseteq p$, q does not force $\mathrm{neg}(\mathrm{neg}(\psi_i)) = \psi_i$ (recall that this equality holds by Proposition 2.6). In particular, it is not the case that $p \Vdash \psi_j$. By the inductive hypothesis, p cannot force both ψ_j and $\mathrm{neg}(\psi_j)$, so we have a contradiction.

3. The dual case is even simpler. Suppose φ is computable Π_α ($\alpha > 0$) of the form $\bigwedge_i \psi_i$, where each ψ_i is computable Σ_{β_i} for $\beta_i < \alpha$. Suppose by

contradiction $p \Vdash \varphi$ and $p \Vdash \mathrm{neg}(\varphi)$. Then $p \Vdash \mathrm{neg}(\psi_i)$ for some i, while for all i and for all $q \supseteq p$, $q \nVdash \mathrm{neg}(\psi_i)$, a contradiction. ∎

LEMMA 3.7 (Density). *There exists $q \supseteq p$ such that $q \parallel \varphi$.*

Proof. Once again, we proceed by induction on φ.

1. Suppose $\varphi = \varphi(\overline{b})$ is computable Σ_0 and Π_0. Take any $q \supseteq p$ such that $\overline{b} \subseteq \mathrm{dom}(q)$. Then q will make either $\varphi(\overline{b})$ or $\mathrm{neg}(\varphi(\overline{b}))$ true in $\mathcal{A}$, so $q \parallel \varphi$.

2. Suppose φ is computable Σ_α of the form $\bigvee_i \psi_i$, where each ψ_i is computable Π_{β_i} for some $\beta_i < \alpha$. Assume that for all $q \supseteq p$, $q \nVdash \varphi$. It suffices to show there is some $q \supseteq p$ such that $q \Vdash \mathrm{neg}(\varphi) = \bigwedge_i \mathrm{neg}(\psi_i)$. By our assumption, for all $q \supseteq p$ and for all i, $q \nVdash \psi_i$. By Proposition 2.6, $\psi_i = \mathrm{neg}(\mathrm{neg}(\psi_i))$. Thus, for all $q \supseteq p$ and for all i, $q \nVdash \mathrm{neg}(\mathrm{neg}(\psi_i))$. But this is what it means for p to force $\mathrm{neg}(\varphi)$, so we have found our desired forcing condition deciding φ.

3. Now suppose φ is computable Π_α of the form $\bigwedge_i \psi_i$, where each ψ_i is computable Σ_{β_i} for some $\beta_i < \alpha$. As in the previous case, assuming that $p \nVdash \varphi$, we must show that there is some $q \supseteq p$ such that $q \Vdash \mathrm{neg}(\varphi)$, where $\mathrm{neg}(\varphi) = \bigvee_i \mathrm{neg}(\psi_i)$. Since $p \nVdash \varphi$, for some i there is an extension $q \supseteq p$ such that $q \Vdash \mathrm{neg}(\psi_i)$. Hence $q \Vdash \mathrm{neg}(\varphi)$. ∎

We are now in a position to build $\mathcal{B}$ using a special sequence of forcing conditions.

DEFINITION 3.8. A *complete forcing sequence*, or *c.f.s.*, is a sequence $(p_n)_{n\in\omega}$ of forcing conditions in $\mathcal{F}$ such that:

1. $p_{n+1} \supseteq p_n$,
2. for all $a \in |\mathcal{A}|$, there is an n such that $a \in \mathrm{ran}(p_n)$,
3. for all $\varphi \in S$, there exists n such that $p_n \parallel \varphi$.

By the density and extension lemmas, it is clear that a c.f.s. exists. We fix such a c.f.s. $(p_n)_{n\in\omega}$ and let $f = \bigcup_n p_n$. Then f is a 1-1 function since it is the union of 1-1 partial permutations. Condition 2 implies $\mathrm{ran}(f) = \omega$. Condition 3 says that all formulas in the forcing language must be decided by some forcing condition in the sequence, including sentences of the form $b = b$, which are in the forcing language for all constants $b \in \omega$. In order to decide a formula, however, all constants (namely, b) mentioned in the formula must be in the domain of the forcing condition. Hence, $\mathrm{dom}(f) = \omega$. Thus, f is a permutation of ω, mapping $|\mathcal{B}|$ 1-1 and onto $|\mathcal{A}|$, and we obtain our generic copy $\mathcal{B}$ of $\mathcal{A}$ from f as planned.

DEFINITION 3.9. We say $\mathcal{B}$ is a *generic copy* of $\mathcal{A}$ if $\mathcal{B} \cong \mathcal{A}$ via the function $f = \bigcup_n p_n$ given by a c.f.s. $(p_n)_{n\in\omega}$.

In this context, however, $\mathcal{B}$ is a predicate structure; that is, we write $\mathcal{B} \models \varphi$ thinking of φ as a predicate formula that is satisfied with elements of $|\mathcal{B}|$ substituted for the free variables in φ. Technically, however, we are still working with a propositional language and propositional formulas and hence need a propositional structure $\mathcal{B}^*$ appropriate for the forcing language. We can define a propositional structure as a map $g : P_{\mathcal{L}} \to \{0,1\}$ in 2^ω such that $g(\varphi) = 1$ if φ is a positive sentence from the atomic diagram of $\mathcal{B}$ and $g(\varphi) = 0$ otherwise (see [3] for a similar definition). In particular, we can associate a propositional structure $\mathcal{B}^*$ with the function $f = \bigcup p_n$ generating $\mathcal{B}$. Thus, the propositional symbols of $\mathcal{B}^*$ are the atomic sentences of $D(\mathcal{B})$. Since the propositional structure $\mathcal{B}^* \models \varphi$ if and only if the predicate structure $\mathcal{B} \models \varphi$, we shall call this structure $\mathcal{B}$ as well.

We conclude this subsection with an important lemma showing the strong connection between the formulas forced by some p_n in the c.f.s. and the truth of these formulas in $\mathcal{B}$.

LEMMA 3.10 (Truth and forcing). *For all $\varphi \in S$, $\mathcal{B} \models \varphi$ iff there exists n such that $p_n \Vdash \varphi$.*

Proof. We proceed by induction on φ (once again considering $\mathcal{B}$ as a propositional structure and φ as a propositional formula).

1. Suppose $\varphi = \varphi(\overline{b})$ is computable Σ_0 and Π_0. If $\mathcal{B} \models \varphi$ then $f = \bigcup_n p_n$ maps $\overline{b}$ to some tuple $\overline{a}$, making $\varphi(\overline{a})$ true in $\mathcal{A}$. Thus, there is some n such that $\mathrm{dom}(p_n)$ includes $\overline{b}$ and p_n makes $\varphi(\overline{b})$ true in $\mathcal{A}$, so $p_n \Vdash \varphi$. Conversely, if there is some n such that $p_n \Vdash \varphi$, then p_n makes φ true in $\mathcal{A}$. Since $\mathcal{B} \cong_f \mathcal{A}$, $\mathcal{B} \models \varphi$.

2. Suppose φ is computable Σ_α of the form $\bigvee_i \psi_i$, and assume that the statement holds for each ψ_i, computable Π_{β_i} for some $\beta_i < \alpha$. If $\mathcal{B} \models \varphi$, then there exists i such that $\mathcal{B} \models \psi_i$, and so by the inductive hypothesis, there is an n such that $p_n \Vdash \psi_i$. Hence, $p_n \Vdash \varphi$. Likewise, if there is an n such that $p_n \Vdash \varphi$, then $p_n \Vdash \psi_i$ for some i. By the inductive hypothesis, $\mathcal{B} \models \psi_i$, and so $\mathcal{B} \models \varphi$ as desired.

3. Now suppose φ is computable Π_α of the form $\bigwedge_i \psi_i$, and assume that the statement holds for each ψ_i, computable Σ_{β_i} for some $\beta_i < \alpha$. First, assume that $\mathcal{B} \models \varphi$. Then for all i, $\mathcal{B} \models \psi_i$ and so by the inductive hypothesis, there is some $n(i)$ such that $p_{n(i)} \Vdash \psi_i$. There is also some m such that $p_m \,\|\, \varphi$. Assume by contradiction that $p_m \Vdash \mathrm{neg}(\varphi)$, so there is some j such that $p_m \Vdash \mathrm{neg}(\psi_j)$. Let $M = \max\{m, n(j)\}$. By the extension lemma, however, $p_M \Vdash \psi_j$ and $p_M \Vdash \mathrm{neg}(\psi_j)$, which is a contradiction by consistency. Therefore, we must have $p_m \Vdash \varphi$. Next, assume that for some m, $p_m \Vdash \varphi$. Then for all i and for all $n \geq m$, it is not the case that $p_n \Vdash \mathrm{neg}(\varphi)$. That is, there is no i such that $p_n \Vdash \mathrm{neg}(\psi_i)$. Suppose by contradiction that

$\mathcal{B} \not\models \varphi$. Then there is some j such that $\mathcal{B} \not\models \psi_j$, and so by the inductive hypothesis, there is some $n(j)$ such that $p_{n(j)} \Vdash \mathrm{neg}(\psi_j)$. Again, we define $M = \max\{m, n(j)\}$. Then by extension, $p_M \Vdash \mathrm{neg}(\psi_j)$, a contradiction. Hence, $\mathcal{B} \models \varphi$. ∎

3.3. *Definability of forcing.* We now show that forcing is definable. The observant reader will notice that these formulas are independent of the particular $\mathcal{A}$ that we fixed at the beginning of the section, which will be crucial in proving our main result.

LEMMA 3.11 (Definability of forcing conditions). *Let $\overline{b}$ be a tuple of distinct constants from ω, with a corresponding tuple $\overline{x}$ of variables. Then we can find a finitary quantifier-free formula* $\mathrm{force}_{\overline{b}}(\overline{x})$ *in the predicate language $\mathcal{L}$ such that $\mathcal{A} \models \mathrm{force}_{\overline{b}}(\overline{a})$ iff the relation p (with domain $\overline{b}$) which maps $\overline{b}$ to $\overline{a}$ is a* 1-1 *function (i.e. iff p is a forcing condition).*

Proof. Let $\mathrm{force}_{\overline{b}}(\overline{x})$ be the conjunction of formulas $\neg(x_i = x_j)$ for each $b_i \neq b_j$ in $\overline{b}$. ∎

LEMMA 3.12 (Definability of forcing). *Let $\varphi \in S$ and let $\overline{b}$ be a tuple of distinct constants. Then we can find a computable infinitary predicate formula* $\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$ *in the predicate language $\mathcal{L}$ such that $\mathcal{A} \models \mathrm{Force}_{\overline{b},\varphi}(\overline{a})$ iff the relation p (with domain $\overline{b}$) taking $\overline{b}$ to $\overline{a}$ is a forcing condition and $p \Vdash \varphi$. Moreover, if φ is computable Σ_α, or computable Π_α, then so is* $\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$.

Proof. Once again, the proof is by induction on φ.

1. Suppose φ is computable Σ_0 and Π_0. If $\overline{b}$ does not include all of the constants from $|\mathcal{B}|$ appearing in φ, then let $\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$ be $\bot$.

Otherwise, we can write $\varphi = \varphi(\overline{b})$, again emphasizing that φ is a finite conjunction of basic propositional symbols with constants in $\overline{b}$. Recall that the propositional symbols in the forcing language are simply atomic formulas in the predicate language $\mathcal{L}$ with a finite tuple of constants $\overline{b}$ for the variables. Replacing the constants $\overline{b}$ with variables $\overline{x}$ again, we have a finitary quantifier-free formula $\varphi(\overline{x})$ in the predicate language $\mathcal{L}$, and we let $\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$ be

$$\mathrm{force}_{\overline{b}}(\overline{x}) \,\&\, \varphi(\overline{x}).$$

Notice that $\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$ is also finitary, computable Σ_0 and Π_0.

2. Suppose φ is computable Σ_α of the form $\bigvee_i \psi_i$, where each ψ_i is computable Π_{β_i} for some $\beta_i < \alpha$. Now $p \Vdash \varphi$ iff $p \Vdash \psi_i$ for some i. By the inductive hypothesis, for each i, we can find an appropriate Π_{β_i} formula $\mathrm{Force}_{\overline{b},\psi_i}(\overline{x})$ such that $p \Vdash \psi_i$ iff $\mathcal{A} \models \mathrm{Force}_{\overline{b},\psi_i}(\overline{a})$. Thus, we may let

$\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$ be

$$\mathrm{force}_{\overline{b}}(\overline{x}) \,\&\, \bigvee\!\!\bigvee_i \mathrm{Force}_{\overline{b},\psi_i}(\overline{x}),$$

a computable Σ_α predicate formula.

3. Suppose φ is computable Π_α of the form $\bigwedge\!\!\bigwedge_i \psi_i$, where each ψ_i is computable Σ_{β_i} for some $\beta_i < \alpha$. Recall that $p \Vdash \varphi$ iff for all i and for all $q \supseteq p$, $q \nVdash \psi_i$. As in the Σ_α case, we use the inductive hypothesis to find appropriate formulas defining forcing for each ψ_i. In this case, however, we are interested in formulas $\mathrm{Force}_{\overline{b},\overline{b}',\mathrm{neg}(\psi_i)}(\overline{x},\overline{x}')$ for each i and for all tuples of distinct constants $\overline{b}$ and $\overline{b}'$. If $\overline{b}$ is in $\mathrm{dom}(p)$ for some forcing condition p, we can think of $\overline{b}'$ as representing the additional constants mapped to $\overline{x}'$ by some extension $q \supseteq p$. Thus, we let $\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$ be

$$\mathrm{force}_{\overline{b}}(\overline{x}) \,\&\, \bigwedge\!\!\bigwedge_{i,\overline{b}'} (\forall \overline{x}')\, \mathrm{neg}(\mathrm{Force}_{\overline{b},\overline{b}',\psi_i}(\overline{x},\overline{x}')).$$

Since $\mathrm{neg}(\psi_i)$ is computable Π_{β_i}, the formulas $\mathrm{Force}_{\overline{b},\overline{b}',\mathrm{neg}(\psi_i)}(\overline{x},\overline{x}')$ are also computable Π_{β_i}. Hence the formulas $\mathrm{neg}(\mathrm{Force}_{\overline{b},\overline{b}',\mathrm{neg}(\psi_i)}(\overline{x},\overline{x}'))$ are computable Σ_{β_i}, making $\mathrm{Force}_{\overline{b},\varphi}(\overline{x})$ computable Π_α as desired. ∎

3.4. *Proof of main result.* We use the forcing machinery we have developed in this section to prove our main result.

THEOREM 1.1. *Let $K \subseteq \mathrm{Mod}(\mathcal{L})$, where K is closed under isomorphism. Suppose that for some $\alpha \geq 1$, i is an index for an effective Σ_α (respectively, Π_α) set B_i such that $B_i \cap K$ is closed under isomorphism. Then we can effectively find (an index for) a computable Σ_α (respectively, Π_α) sentence σ such that $B_i = \mathrm{Mod}(\sigma)$ within K.*

Proof. We first define σ when i is an index for an effective Σ_α set, and verify that for a fixed $\mathcal{A} \in K$, $\mathcal{A} \in B_i$ iff $\mathcal{A} \in \mathrm{Mod}(\sigma)$. Next, we show that the result holds for all $\mathcal{A} \in K$, so $B_i = \mathrm{Mod}(\sigma)$ within K. Finally, we consider the case when i is an index for an effective Π_α set.

Definition of σ in the effective Σ_α case. Let i be an index for an effective Σ_α set. By Proposition 3.3, we can effectively find a computable Σ_α formula ψ_j in the language $P_{\mathcal{L}}$ such that for all $\mathcal{B} \in K$, $\mathcal{B} \in B_i$ iff $\mathcal{B} \models \psi_j$. Now ψ_j may involve infinitely many constants. We want to replace ψ_j by a predicate $\mathcal{L}$-sentence which satisfies the same property. We do this using the forcing mechanism we have developed.

Let $\mathcal{A} \in K$. We think of the forcing construction for building a generic copy $\mathcal{B}$ of $\mathcal{A}$, given by a c.f.s. $(p_n)_{n\in\omega}$. By the truth and forcing lemma, $\mathcal{B} \models \psi_j$ iff there is some $n \in \omega$ such that $p_n \Vdash \psi_j$. Moreover, by the definability of forcing lemma, for all tuples of distinct constants $\overline{b}$, there is a

computable Σ_α formula $\mathrm{Force}_{\overline{b},\psi_j}(\overline{x})$ in the predicate language $\mathcal{L}$ such that $\mathcal{A} \models \mathrm{Force}_{\overline{b},\psi_j}(\overline{a})$ iff the function p mapping $\overline{b}$ to $\overline{a}$ is a forcing condition and $p \Vdash \psi_j$. Thus, we take σ to be the computable Σ_α sentence

$$\bigvee_{\overline{b},\overline{x}} (\exists \overline{x})\, \mathrm{Force}_{\overline{b},\psi_j}(\overline{x}),$$

a c.e. disjunction over tuples of constants $\overline{b}$ and corresponding tuples of variables $\overline{x}$, which says that there is some forcing condition forcing ψ_j.

Verification. We must check that this is the desired $\mathcal{L}$-formula corresponding to the effective Σ_α set B_i. Recall that we started with a fixed but arbitrary $\mathcal{A} \in K$. Assume further that $\mathcal{A} \in B_i$ and consider the generic copy $\mathcal{B}$ of $\mathcal{A}$ ($\mathcal{B} \cong \mathcal{A}$). Since $B_i \cap K$ is closed under isomorphism, we have $\mathcal{B} \in B_i \cap K$ and thus $\mathcal{B} \models \psi_j$. This implies that there is an $n \in \omega$ such that the forcing condition p_n, mapping some $\overline{b}$ to $\overline{a}$, forces ψ_j. Thus, $\mathcal{A} \models \mathrm{Force}_{\overline{b},\psi_j}(\overline{a})$, so $\mathcal{A} \in \mathrm{Mod}(\sigma)$ as desired. Likewise, if we assume that $\mathcal{A} \in \mathrm{Mod}(\sigma)$, then there is some forcing condition p forcing ψ_j, and consequently we can build a generic copy $\mathcal{B}$ of $\mathcal{A}$ such that $\mathcal{B} \models \psi_j$. But the fact that K is closed under isomorphism implies $\mathcal{B} \in K$. Thus, $\mathcal{B} \in B_i$ as well. Moreover, since the intersection $B_i \cap K$ is closed under isomorphism and $\mathcal{B} \cong \mathcal{A}$, $\mathcal{A} \in B_i$.

Generalization to all $\mathcal{A} \in K$. Recall that we have been working with a fixed $\mathcal{A} \in K$ in mind. We have shown that for this fixed $\mathcal{A}$, $\mathcal{A} \in B_i \cap K$ iff $\mathcal{A} \in \mathrm{Mod}(\sigma) \cap K$. However, we seek to prove this result for all $\mathcal{A} \in K$. It is clear that there is some dependence on $\mathcal{A}$. Given a forcing condition p, the particular formulas in S forced by p depend on the specific $\mathcal{A}$ that we selected. Likewise, the generic copy $\mathcal{B}$ depends on $\mathcal{A}$ and the specific c.f.s. used to obtain this copy. Nevertheless, the *method* we used to define forcing and choose a c.f.s. never depended on $\mathcal{A}$. Similarly, the proofs of the lemmas in this section did not rely on any special knowledge of $\mathcal{A}$. In particular, we defined the formulas $\mathrm{Force}_{\overline{b},\psi}(\overline{x})$ based only on $\overline{b}$ and ψ, independently of $\mathcal{A}$. Thus, our forcing mechanism works uniformly for all $\mathcal{A}$, and the result is true *for all* $\mathcal{A} \in K$. This means that for all $\mathcal{A} \in K$, $\mathcal{A} \in B_i$ iff $\mathcal{A} \in \mathrm{Mod}(\sigma)$, so $B_i = \mathrm{Mod}(\sigma)$ within K.

Definition in the effective Π_α case. Let i be an index for an effective Π_α set. This implies that $\overline{B}_i$ is effective Σ_α, so we can use the argument above to find a computable Σ_α sentence σ such that $\overline{B}_i = \mathrm{Mod}(\sigma)$ within K. But this means that $B_i = \mathrm{Mod}(\mathrm{neg}(\sigma))$ within K, where $\mathrm{neg}(\sigma)$ is computable Π_α. ∎

We use this theorem in the remainder of the paper to prove some additional results. The first (and easiest) application is to get the effective version of Vaught's theorem.

THEOREM 1.2 (Effective version of Vaught's theorem). *Let $K \subseteq \mathrm{Mod}(\mathcal{L})$, where K is closed under isomorphism. Let $\alpha \geq 1$. If K is an effective Σ_α (respectively, Π_α) set with index i, then we can effectively find (an index for) a computable Σ_α (respectively, Π_α) sentence σ such that $K = \mathrm{Mod}(\sigma)$.*

Proof. This is a special case of Theorem 1.1 above: take $K = B_i$. ∎

Knight observed that we can relativize Theorem 1.2 in order to obtain an alternative proof of Vaught's theorem.

COROLLARY 3.13 (Relativization of Theorem 1.2). *Let X be an arbitrary set. Let $K \subseteq \mathrm{Mod}(\mathcal{L})$, where K is closed under isomorphism. Let $\alpha \geq 1$. If K is an X-effective Σ_α (respectively, X-effective Π_α) set with index i, then we can effectively find (an index for) an X-computable Σ_α (respectively, X-computable Π_α) sentence σ such that $K = \mathrm{Mod}(\sigma)$.*

Proof. We relativize the results in Sections 2 and 3 to an arbitrary set X. First, consider ordinals computable in X, with a corresponding set of computable notations (a notation for an X-computable limit ordinal α is $3 \cdot 5^e$, where e is an index for an X-computable function that yields an increasing sequence with limit α). Next, with this new set of notations, we define the X-effective Borel hierarchy and X-computable infinitary formulas. The forcing proof then follows as is, substituting X-effective Borel set for effective Borel set and X-computable sentence for computable sentence. ∎

THEOREM 3.14 (Vaught). *Let $K \subseteq \mathrm{Mod}(\mathcal{L})$, where K is closed under isomorphism. Let $\alpha \geq 1$. If K is $\mathbf{\Sigma}_\alpha$ (respectively, $\mathbf{\Pi}_\alpha$) in the Borel hierarchy, then there is a Σ_α (respectively, Π_α) sentence σ such that $K = \mathrm{Mod}(\sigma)$.*

Proof. Recall that effective Borel sets are constructed by taking c.e. unions and complements of basic sets. Arbitrary Borel sets in $\mathrm{Mod}(\mathcal{L})$, on the other hand, are constructed using countable union and complementation, not necessarily over c.e. sets. These operations, however, are computable relative to some X, namely the (Borel) code X for K itself (the details of how we code K in X are not important, see [4] for an example). If K is $\mathbf{\Sigma}_\alpha$, for example, then K is X-effective Σ_α, so we can apply Theorem 3.13 to get an X-computable Σ_α sentence σ such that $K = \mathrm{Mod}(\sigma)$. Hence σ is the desired Σ_α sentence. Likewise for K that are $\mathbf{\Pi}_\alpha$ in the Borel hierarchy. ∎

4. Application of the main result. In [2], two notions of effective embedding are introduced: computable embedding and Turing computable embedding. These embeddings provide a way to compare classes of countable structures, while allowing finer distinctions than the Borel embeddings which served as inspiration for them. Computable embeddings were originally chosen for investigation in [2], but various results on Turing computable embeddings, defined as follows, are presented in [6].

DEFINITION 4.1. A *Turing computable embedding* of a class K into K' is an operator $\Phi = \varphi_e$ such that:

1. for all $\mathcal{A} \in K$, $\varphi_e^{D(\mathcal{A})} = \chi_{D(\mathcal{B})}$ for some $\mathcal{B} \in K'$,
2. if $\mathcal{A}, \mathcal{A}' \in K$ have corresponding Φ-image structures $\mathcal{B}, \mathcal{B}' \in K'$, respectively, then $\mathcal{A} \cong \mathcal{A}'$ iff $\mathcal{B} \cong \mathcal{B}'$.

We write $K \leq_{\mathrm{tc}} K'$ if there is such an embedding.

In the context of [2] and [6], K and K' are classes of countable structures with universe a *subset* of ω, possibly finite. The classes are closed under isomorphism, modulo this restriction on the universe. Given a Turing computable embedding Φ, we have a uniform effective procedure to pass from the diagram of an input structure $\mathcal{A}$ in K to some output structure $\mathcal{B} = \Phi(\mathcal{A})$ in K', and we do so in a way that respects isomorphism. In order to compute the characteristic function of the output structure $\mathcal{B}$, we can use information about what elements are in the universe of $\mathcal{A}$ as well as information about what elements are *not* in the universe of $\mathcal{A}$.

The primary result in [6] is a pull-back theorem which says that given a Turing computable embedding Φ such that $K \leq_{\mathrm{tc}} K'$ via Φ, and a computable infinitary sentence φ in the language of K', we can find a computable infinitary sentence φ^* of the same complexity in the language of K such that for all $\mathcal{A} \in K$, $\Phi(\mathcal{A}) \models \varphi$ iff $\mathcal{A} \models \varphi^*$. We call φ^* a *pull-back* of φ. This is an important tool in proving non-embeddability results and in giving characterizations of the classes that Turing computably embed into a given class K. We refer the interested reader to [6].

In the present paper, however, we are working with $K \subseteq \mathrm{Mod}(\mathcal{L})$ and $K' \subseteq \mathrm{Mod}(\mathcal{L}')$, so K and K' are classes of structures with universe *all* of ω. This context is of classical interest, so we would like to look at Turing computable embeddings involving classes K, K' like this. In this context, we do not need to worry about elements that are not in the universe of $\mathcal{A} \in K$ since the universe of all input and output structures is ω. Indeed, for each halting computation of $\Phi = \varphi_e$, the information we care about is $\gamma(\overline{a})$, a finite conjunction of basic sentences in the language $\mathcal{L}$ corresponding to the questions (with positive answers) asked of the oracle $D(\mathcal{A})$ during the course of the computation. We say that the computation *uses* $\gamma(\overline{a})$.

When working with classes of structures like this, we can use the general pull-back theorem from [6]. Nevertheless, it is interesting to see that our main result also yields something in this direction. In this section, we use Theorem 1.1 to get a proof of the pull-back theorem for Turing computable embeddings that take $K \subseteq \mathrm{Mod}(\mathcal{L})$ to $K' \subseteq \mathrm{Mod}(\mathcal{L}')$.

4.1. *A pull-back theorem.* We start with a computable infinitary sentence φ in the predicate language $\mathcal{L}'$. Although we cannot immediately find

a pull-back of φ in the predicate language $\mathcal{L}$, we can easily find a pull-back $\widehat{\varphi}$ in the propositional language $P_{\mathcal{L}}$ using an inductive argument (recall that the propositional symbols of $P_{\mathcal{L}}$ are atomic $\mathcal{L}$-formulas with constants substituted for the finitely many variables). We then form a corresponding effective Borel set and apply the main result, Theorem 1.1, to get the pull-back φ^* as desired. Thus, the outline of the proof is as follows:

$$\begin{array}{ll}
\varphi & \text{computable infinitary sentence in predicate language } \mathcal{L}' \text{ (given)} \\
\downarrow & \\
\widehat{\varphi} & \text{pull-back in propositional language } P_{\mathcal{L}} \text{ (Lemma 4.2)} \\
\downarrow & \\
\mathrm{Mod}(\widehat{\varphi}) & \text{effective Borel set (Lemma 4.3)} \\
\downarrow & \\
\varphi^* & \text{pull-back in predicate language } \mathcal{L} \text{ (Theorem 1.1).}
\end{array}$$

We begin with a lemma that allows us to find a pull-back in $P_{\mathcal{L}}$. Unlike the pull-back φ^* that we are attempting to find in the predicate language $\mathcal{L}$, the pull-back in $P_{\mathcal{L}}$ may involve infinitely many constants.

LEMMA 4.2. *Assume $K \leq_{\mathrm{tc}} K'$ via $\Phi = \varphi_e$ and let φ be a computable infinitary sentence which is the result of substituting a finite tuple of constants for the free variables in a computable infinitary formula in the predicate language $\mathcal{L}'$. Then we can effectively find a computable infinitary sentence $\widehat{\varphi}$ in the propositional language $P_{\mathcal{L}}$ such that for all $\mathcal{A} \in K$, $\Phi(\mathcal{A}) \models \varphi$ iff $\mathcal{A} \models \widehat{\varphi}$. Moreover, if φ is computable Σ_α, or computable Π_α, for $\alpha \geq 1$, then so is $\widehat{\varphi}$.*

Proof. The proof is by induction on the complexity of φ.

1. Suppose φ is computable Σ_0 and Π_0. We may assume that $\varphi = \varphi(\overline{b})$ is a finite conjunction of basic sentences ψ_i, where $\overline{b}$ is a finite tuple of constants. Consider one such ψ_i and let H be the c.e. set of halting computations such that Φ halts with output 1 on input ψ_i. Each computation $h \in H$ uses some $\gamma_h(\overline{a})$, so we can let $\widehat{\psi_i}$ be $\bigvee_{h \in H} \gamma_h(\overline{a})$. This is a computable Σ_1 sentence in the propositional language $P_{\mathcal{L}}$, and for $\mathcal{A} \in K$, $\Phi(\mathcal{A}) \models \psi_i$ iff $\mathcal{A} \models \widehat{\psi_i}$. Thus, we can take $\widehat{\varphi}$ to be the finite conjunction of the $\widehat{\psi_i}$ as defined above. Then $\widehat{\varphi}$ is also computable Σ_1 and $\Phi(\mathcal{A}) \models \varphi$ iff $\mathcal{A} \models \widehat{\varphi}$.

2. Suppose φ is computable Σ_α ($\alpha \geq 1$) of the form $\bigvee_i (\exists \overline{u}_i)\, \psi_i(\overline{u}_i)$, where each ψ_i is computable Π_{β_i} for some $\beta_i < \alpha$. Then let $\widehat{\varphi}$ be

$$\bigvee_{i,\overline{d}_i} \widehat{\psi_i(\overline{d}_i)},$$

which, by the inductive hypothesis, is computable Σ_α and satisfies the desired property.

3. Suppose φ is computable Π_α ($\alpha \geq 1$) of the form $\bigwedge\!\!\!\bigwedge_i (\forall \overline{u}_i)\, \psi_i(\overline{u}_i)$, where each ψ_i is computable Σ_{β_i} for some $\beta_i < \alpha$. If $\alpha = 1$, then $\mathrm{neg}(\psi_i(\overline{u}_i))$ is a computable Π_0 formula for each i, so for all possible tuples of constants $\overline{d}_i$ which we can substitute for the variables $\overline{u}_i$, we can use the first case to form $\widehat{\mathrm{neg}(\psi_i(\overline{d}_i))}$, a computable Σ_1 sentence in $P_{\mathcal{L}}$. We let $\widehat{\varphi}$ be

$$\mathrm{neg}\Big(\bigvee_{i,\overline{d}_i} \widehat{\mathrm{neg}(\psi_i(\overline{d}_i))}\Big),$$

so $\widehat{\varphi}$ is computable Π_1 as desired. Otherwise, if $\alpha > 1$, let $\widehat{\varphi}$ be $\bigwedge\!\!\!\bigwedge_{i,\overline{d}_i} \widehat{\psi_i(\overline{d}_i)}$, which by the inductive hypothesis is computable Π_α and satisfies the required property.

(Note that, for the Π_1 case, we must write $\widehat{\varphi}$ in terms of the dual. This ensures that the complexity of φ and $\widehat{\varphi}$ matches at this level. We could, of course, write it in this form for all Π_α cases, but it is only necessary when $\alpha = 1$.) ∎

As we did before in the forcing argument, when we write $\mathcal{A} \models \widehat{\varphi}$ we are thinking of $\mathcal{A}$ as a propositional structure. Likewise, later in this section when we write $\mathcal{A} \models \varphi^*$ we are thinking of $\mathcal{A}$ as a predicate structure, even though we do not explicitly make this distinction.

Next we pass from a sentence $\widehat{\varphi}$ in the propositional language $P_{\mathcal{L}}$ to an effective Borel set of the same complexity.

LEMMA 4.3. *Let $\widehat{\varphi}$ be a computable Σ_α (respectively, Π_α) sentence in the propositional language $P_{\mathcal{L}}$. Then* $\mathrm{Mod}(\widehat{\varphi}) = B_j$ *where j is an index for an effective Σ_α (respectively, Π_α) set.*

Proof. Once again, the proof is by induction on the complexity of $\widehat{\varphi}$.

1. Suppose $\widehat{\varphi}$ is computable Σ_0 and Π_0. Then $\widehat{\varphi}$ is a finitary conjunction of basic propositional symbols. However, we can also think of $\widehat{\varphi}$ as a finitary quantifier-free formula in the predicate language $\mathcal{L}$ with a tuple of constants substituted for the finitely many variables in the formula. By Definition 2.2, $\mathrm{Mod}(\widehat{\varphi}) = \{\mathcal{B} : \mathcal{B} \models \widehat{\varphi}\} = B_j$ where $j \in BS_1^\Sigma = BS_1^\Pi$ is the Gödel number of the sentence $\widehat{\varphi}$.

2. Suppose $\widehat{\varphi}$ is computable Σ_α ($\alpha \geq 1$) of the form $\bigvee_i \widehat{\psi}_i$, where each $\widehat{\psi}_i$ is computable Π_{β_i} for some $\beta_i < \alpha$. Then $\mathrm{Mod}(\widehat{\varphi})$ is the c.e. union of the sets $\mathrm{Mod}(\widehat{\psi}_i)$, which are effective Π_{β_i} by the inductive hypothesis. Thus, $\mathrm{Mod}(\widehat{\varphi})$ is an effective Σ_α set B_j, where j is of the form $\langle \Sigma, a, e\rangle$ and $|a|_{\mathcal{O}} = \alpha$.

3. Likewise, suppose $\widehat{\varphi}$ is computable Π_α ($\alpha \geq 1$) of the form $\bigwedge_i \widehat{\psi}_i$, where each $\widehat{\psi}_i$ is computable Σ_{β_i} for some $\beta_i < \alpha$. By the inductive hypothesis, $\mathrm{Mod}(\widehat{\varphi})$ is the c.e. intersection of the effective Σ_{β_i} sets $\mathrm{Mod}(\widehat{\psi}_i)$. Again,

this is enough to conclude that $\mathrm{Mod}(\widehat{\varphi})$ is an effective Π_α set B_j, with an index j of the form $\langle \Pi, a, e\rangle$ for some $|a|_{\mathcal{O}} = \alpha$. ∎

We are now ready to prove the pull-back theorem. It may be helpful to review the outline of the proof given at the beginning of this section.

THEOREM 4.4 (Pull-back theorem for $\leq_{\mathrm{tc}}$ with $K \subseteq \mathrm{Mod}(\mathcal{L})$ and $K' \subseteq \mathrm{Mod}(\mathcal{L}')$). *Let $K \subseteq \mathrm{Mod}(\mathcal{L})$ and $K' \subseteq \mathrm{Mod}(\mathcal{L}')$ where K, K' are closed under isomorphism. If $K \leq_{\mathrm{tc}} K'$ via $\Phi = \varphi_e$, then for any computable infinitary sentence φ in the language $\mathcal{L}'$, we can find a computable infinitary sentence φ^* in the language $\mathcal{L}$ such that for all $\mathcal{A} \in K$, $\Phi(\mathcal{A}) \models \varphi$ iff $\mathcal{A} \models \varphi^*$. Moreover, if φ is computable Σ_α, or computable Π_α, for $\alpha \geq 1$, then so is φ^*.*

Proof. We give the proof for φ computable Σ_α. The proof is similar when starting with a computable Π_α sentence.

Suppose φ is a computable Σ_α sentence in the language $\mathcal{L}'$. By Lemma 4.2, there is a computable Σ_α sentence $\widehat{\varphi}$ in the propositional language $P_{\mathcal{L}}$ such that, for all $\mathcal{A} \in K$, $\Phi(\mathcal{A}) \models \varphi$ iff $\mathcal{A} \models \widehat{\varphi}$. However, this is not yet the required sentence: we want to find a sentence φ^* in the predicate language $\mathcal{L}$ which satisfies the same property. We continue by taking $\widehat{\varphi}$ and forming $\mathrm{Mod}(\widehat{\varphi}) \subseteq \mathrm{Mod}(\mathcal{L})$, which is an effective Σ_α set by Lemma 4.3.

We do not know how the operator Φ behaves on structures outside of K. However, for $\mathcal{A}, \mathcal{A}' \in K$, if $\mathcal{A}' \cong \mathcal{A}$, then $\Phi(\mathcal{A}') \cong \Phi(\mathcal{A})$ since the Turing computable embedding Φ respects isomorphism (see Definition 4.1). Thus,

$$\mathcal{A} \models \widehat{\varphi} \;\Rightarrow\; \Phi(\mathcal{A}) \models \varphi \;\Rightarrow\; \Phi(\mathcal{A}') \models \varphi \;\Rightarrow\; \mathcal{A}' \models \widehat{\varphi},$$

so $\mathrm{Mod}(\widehat{\varphi}) \cap K$ is closed under isomorphism and we have satisfied the hypotheses of Theorem 1.1. Applying this theorem, we see that $\mathrm{Mod}(\widehat{\varphi})$ is axiomatized (within K) by a computable Σ_α sentence σ, which is in the language $\mathcal{L}$. We define φ^* to be the sentence σ given by Theorem 1.1. Thus, for $\mathcal{A} \in K$,

$$\Phi(\mathcal{A}) \models \varphi \;\Leftrightarrow\; \mathcal{A} \models \widehat{\varphi} \;\Leftrightarrow\; \mathcal{A} \in \mathrm{Mod}(\widehat{\varphi}) = B_i \;\Leftrightarrow\; \mathcal{A} \models \varphi^*$$

as desired. Moreover, since Lemmas 4.2 and 4.3 and Theorem 1.1 are effective, we have effectively found the pull-back of φ in the predicate language $\mathcal{L}$ using this procedure. ∎

References

[1] C. J. Ash and J. F. Knight, *Computable Structures and the Hyperarithmetical Hierarchy*, Elsevier, 2000.

[2] W. Calvert, D. Cummins, J. F. Knight and S. Miller, *Comparing classes of finite structures*, Algebra Logika 34 (2004), 666–701 (in Russian); English transl.: Algebra Logic 43 (2004), 374–392.

[3] H. B. Enderton, *A Mathematical Introduction to Logic*, Academic Press, 1972.
[4] T. Jech, *Set Theory*, 2nd ed., Springer, 1997.
[5] A. S. Kechris, *Classical Descriptive Set Theory*, Springer, 1995.
[6] J. F. Knight, S. Miller, and M. Vanden Boom, *Turing computable embeddings*, J. Symbolic Logic, to appear.
[7] E. G. K. López-Escobar, *An interpolation theorem for denumerably long formulas*, Fund. Math. 57 (1965), 253–272.
[8] R. Vaught, *Invariant sets in topology and logic*, ibid. 82 (1974), 269–294.

Department of Mathematics
University of Notre Dame
Notre Dame, IN 46556, U.S.A.
E-mail: mvandenb@nd.edu

Received 9 September 2006;
in revised form 7 May 2007