EFFECTIVE PRIME UNIQUENESS

PETER CHOLAK AND CHARLIE MCCOY, C.S.C.

ABSTRACT. Assuming the obvious definitions below, we show that a decidable model that is effectively prime is also effectively atomic. This implies that two effectively prime (decidable) models are computably isomorphic. This is in contrast to the theorem that there are two atomic decidable models which are not computably isomorphic. We end with a section describing the implications of this result in reverse mathematics.

1. INTRODUCTION

Our goal is to explore (decidable) prime models from the perspective of effective model theory, computability theory, and reverse mathematics. In particular, we are interested in the result that prime models are unique. Similar results have been found before. In [4], Hirschfeldt, Shore and Slaman looked at classical results involving prime and atomic models from the perspective of reverse mathematics. They left open the analysis of the prime uniqueness theorem.

We begin with a review of the relevant definitions and results from classical model theory; see [2]. For this review, fix a complete theory T of a *countable* language $\mathscr{L}$.

Definition 1. *A formula $\varphi(\vec{x})$ is complete if for every other formula $\psi(\vec{x})$, exactly one of the following holds: $T \vdash \varphi(\vec{x}) \to \psi(\vec{x})$ or $T \vdash \varphi(\vec{x}) \to \neg\psi(\vec{x})$.*

Definition 2. *A model $\mathscr{A} \models T$ is atomic if for every $\vec{a} \in \mathscr{A}$, there is a complete formula $\varphi(\vec{x})$ so that $\mathscr{A} \models \varphi(\vec{a})$.*

Definition 3. *A model $\mathscr{A} \models T$ is prime if for every other model $\mathscr{M} \models T$, there is an elementary embedding of $\mathscr{A}$ into $\mathscr{M}$.*

Theorem 1. *(Atomic Uniqueness) If $\mathscr{A}$ and $\mathscr{B}$ are countable atomic models of T, then $\mathscr{A} \cong \mathscr{B}$.*

Proof. Use a back-and-forth construction with complete formulas determining how to extend the partial isomorphism. □

Theorem 2. *(Atomic ⇒ Prime) If $\mathscr{A}$ is a countable atomic model of T, then $\mathscr{A}$ is prime.*

Proof. Use the "forth" half of the back-and-forth argument. □

Theorem 3. *(Prime ⇒ Atomic) If $\mathscr{A}$ is a prime model of T, then $\mathscr{A}$ is countable and atomic.*

Proof. By the compact theorem T has a countable model. Therefore $\mathscr{A}$ is countable. Let $\vec{a} \in \mathscr{A}$, and consider its type. For any other model $\mathscr{B}$ of T, there is an elementary embedding of $\mathscr{A}$ into $\mathscr{B}$, so that $\mathscr{B}$ also realizes this type. By the Omitting Types Theorem, the type of $\vec{a}$ includes a complete formula.

□

This work was partially supported by a grant from the Simons Foundation (#315283 to Peter Cholak).

Theorem 4. *(Prime Uniqueness) If $\mathscr{A}$ and $\mathscr{B}$ are prime models of T, then $\mathscr{A} \cong \mathscr{B}$.*

Proof. Immediate by the Atomic Uniqueness and (Prime $\Rightarrow$ Atomic). □

In [4], the authors showed that (Prime $\Rightarrow$ Atomic) holds in RCA_0, and that both (Atomic $\Rightarrow$ Prime) and Atomic Uniqueness are equivalent to ACA_0. Note that in Reverse Mathematics models are given by their complete diagram. Recall REC is the canonical model RCA_0 where the second order part is just the collection of computable sets. Every model in REC is decidable. Since the classical proof of Prime Uniqueness uses the latter two theorems, its effective or non-effective content and its place within Reverse Mathematics are not answered by these results. In order to answer these questions, we consider effective analogues of the classical definitions.

Definition 4. *Let T be a decidable theory and $\mathscr{A}$ a decidable model of T.*

(1) *The model $\mathscr{A}$ is effectively prime, if for every decidable model $\mathscr{M} \models T$, there is a computable elementary embedding $f : \mathscr{A} \to \mathscr{M}$. Note that f need not be uniformly computable in $\mathscr{A}$ and/or $\mathscr{M}$.*
(2) *The model $\mathscr{A}$ is effectively atomic if there is a computable function g that accepts as an input a tuple $\vec{a}$ from $\mathscr{A}$ (of any length) and outputs a complete formula $\varphi(\vec{x})$ so that $\mathscr{A} \models \varphi(\vec{a})$. Again g need not be uniformly computable in $\mathscr{A}$.*
(3) *The model $\mathscr{A}$ is uniformly effectively prime if there is a partial computable function Φ so that, given a decidable $\mathscr{M} \models T$, $\Phi(\mathscr{M})$ halts and outputs the code of a computable elementary embedding $f : \mathscr{A} \to \mathscr{M}$. Again Φ need not be uniformly computable in $\mathscr{A}$.*

Some observations:

(1) If two decidable models $\mathscr{A}$ and $\mathscr{B}$ of the same decidable theory T are both effectively atomic, then the classical back and forth construction produces a computable isomorphism $f : \mathscr{A} \cong \mathscr{B}$.
(2) A modification of the classical proof that atomic implies prime shows that effectively atomic implies uniformly effectively prime.

It is essential to note that the results in [4] are about decidable, atomic models, *not necessarily* effectively atomic models. To understand this distinction, we state a few easily proven results.

Proposition 5. *Let T be a theory in a computable language $\mathscr{L}$. Then the set $\{\phi \mid \phi$ is a complete formula of $T\}$ is Π^T_1.*

Proof. Check whether, for all other ψ in $\mathscr{L}$, exactly one of $T \vdash (\phi \to \psi)$ or $T \vdash (\phi \to \neg\psi)$ holds. □

Moreover, this result can actually be sharp, even in a case where the theory T is atomic, as the following result establishes.

Proposition 6. *There is a decidable, atomic theory T for which the set $\{\phi \mid \phi$ is a complete formula of $T\}$ is Π^0_1-complete.*

Proof. This follows directly from the construction in the proof of Theorem 2.3 in [4]. But we would like to present a modification which will be useful in the proof of Lemma 18. We will use the same language and the collections of Axioms 2,3,4, and 6, from [4]. We will replace the collection of Axioms 1 with the axioms that R_i are pairwise disjoint sets with exactly 2 distinct elements and collection of Axioms 5 with if $\Phi_{i,s}(s) \downarrow$ there is exactly

one $x \in R_i$ such that $R_{i,s}(x)$ and exactly one $x \in R_i$ such that $\neg R_{i,s}(x)$. Like in [4] we can show that this theory T is recursive, has quantifier elimination, is complete, decidable, and atomic. The complete formulas are $R_i(x)$ iff $\Phi_i(i)$ diverges and $R_i(x) \wedge R_{i,s}(x)$ and $R_i(x) \wedge \neg R_{i,s}(x)$ iff $\Phi_i(i)$ converges in exactly s steps. $R_i(x)$ is a complete formula of T iff $i \in \overline{K}$. □

This is in contrast with what occurs when a theory has an effectively atomic model.

Proposition 7. *Let T be a decidable theory and $\mathscr{A} \models T$ a decidable, effectively atomic model. Then $\{\phi \mid \phi$ is a complete formula of $T\}$ is computable.*

Proof. Let $\phi(\vec{x})$ be a formula with the tuple of free variables $\vec{x}$ actually occurring in ϕ. First, verify that ϕ is consistent with T. If so, since $\mathscr{A}$ is a model of T, there is a tuple of elements $\vec{a} \in \mathscr{A}$ such that $\mathscr{A} \models \phi(\vec{a})$. Since $\mathscr{A}$ is effectively atomic, we can effectively find a complete formula $\varphi(\vec{x})$ so that $\mathscr{A} \models \varphi(\vec{a})$ and hence $T \vdash \varphi(\vec{x}) \to \phi(\vec{x})$. Now, we check if $T \vdash \phi(\vec{x}) \to \varphi(\vec{x})$. If it does, then $\phi(\vec{x})$ is a complete formula, because it implies a complete formula. If it does not, then, since $T \nvdash \phi(\vec{x}) \to \neg\varphi(\vec{x})$, $\phi(\vec{x})$ is not a complete formula. □

Throughout the rest of the section and the next, a theory T will always be a complete and decidable theory of a computable language $\mathscr{L}(T)$, and all models will be decidable. Furthermore, we assume that all theories and models are presented in such a way that the associated computable language can always be recovered from the code for the theory or model. We will often re-state these facts for emphasis.

Our main result, whose proof is in Section 2, is the following:

Theorem 8. *(Effectively Prime ⇒ Effectively Atomic) Let T be a decidable theory and $\mathscr{A} \models T$ a decidable model. Then either there is a computable function h witnessing that $\mathscr{A}$ is effectively atomic; or there is a decidable $\mathscr{M} \models T$ such that there is no computable elementary embedding of $\mathscr{A}$ into $\mathscr{M}$.*

Corollary 9. *(Effective Prime Uniqueness) Let T be decidable and $\mathscr{A}, \mathscr{B} \models T$ be decidable models. Then either there is a computable isomorphism $h : \mathscr{A} \cong \mathscr{B}$; or there is a decidable $\mathscr{M} \models T$, so that either there is no computable elementary embedding of $\mathscr{A}$ into $\mathscr{M}$, or there is no computable elementary embedding of $\mathscr{B}$ into $\mathscr{M}$.*

By the first observation after Definition 4, Theorem 8 implies Corollary 9. By the second observation, effectively prime, effectively atomic, and uniformly effectively prime are all equivalent.

Moreover, by looking carefully at the construction in Section 2, we can see that there is actually a greater degree of uniformity to Theorem 8, as stated in the next result. Note that a code for a decidable model $\mathscr{A}$ is a Turing machine that computes the complete diagram of $\mathscr{A}$. Moreover, recall that, by assumption, the presentation for $\mathscr{A}$ includes the computable language for $\mathscr{A}$. So from a decidable model it's theory can be computably recovered. Thus, T need not be an input into the functional Ψ below. However, the proposition following the result shows that the input of the e is necessary.

Corollary 10. *There a Turing functional $\Psi(\mathscr{A}, e)$ such that if $\mathscr{A}$ is a decidable model and T is its decidable theory, then either for some e, $\Psi(\mathscr{A}, e)$ is a code for a computable function witnessing that $\mathscr{A}$ is effectively atomic; or there is a decidable $\mathscr{M} \models T$, such that there is no computable elementary embedding of $\mathscr{A}$ into $\mathscr{M}$.*

Proposition 11. *For all Ψ, there in an effectively atomic $\mathscr{A}$ such that $\Psi(\mathscr{A})$ does not witness that $\mathscr{A}$ is effectively atomic.*

Proof. Fix Ψ. By the Recursion Theorem we can assume that we know the index e of the model $\mathscr{A}$ we construct. We will work in the language of infinitely unary relations, U_i, and our model has ω as its domain.

We are only concerned about the case where $\Psi(e)$ itself is the code of a computable function g that accepts tuples of $\mathscr{A}$ as inputs and outputs formulas in the language of $\mathscr{A}$; in particular, g should accept the 1-tuple 0.

If at stage s, $(\Psi_s(e))_s(0)$ does not halt, then we declare U_s to be empty. If s is the first stage at which $(\Psi_s(e))_s(0)$ halts, and it is not (the code of) a formula with one free variable, then we declare U_i to be empty for all i. If s is the first stage at which $(\Psi_s(e))_s(0)$ halts, and it is a formula with one free variable, then let l be the least number such that $l \geq s$ and $l \geq j$ for any j where U_j is mentioned in this formula. The evens go into U_{l+1} and the odds stay out. For all $i \leq l$ and $i > l+1$, we declare U_i to be empty.

The resulting $\mathscr{A}$ is the infinite model where either there is nothing in any U_i; or for $l+1$, there is nothing in any U_i for $i \neq l+1$, and U_{l+1} splits the domain into evens and odds. The complete formulas for 1-types are either $x = x$ (for everything) or the pair $U_{l+1}(x)$ (for evens) and $\neg U_{l+1}(x)$ (for odds). So $\mathscr{A}$ is effectively atomic. However, $(\Psi_s(e))_s(0)$ is certainly not a complete formula for the element 0, because on the U_j mentioned in $(\Psi_s(e))_s(0)$, the element 0 and the element 1 agree, but they disagree on U_{l+1}. □

Hence the "obvious" notion of "uniformly effectively atomic" is vacuous.

Finally, we should note that the construction given in the next section does not depend on knowing ahead of time if the model $\mathscr{A}$ is infinite or finite. But it was most likely already known that Corollary 10 and Proposition 11 hold for finite models.

For instance, for Corollary 10, let e code a "guess" at n, the size of $|\mathscr{A}| = a_0, a_1, \ldots a_n = \vec{a}$, and a "guess" at the number l of distinct automorphisms of $\mathscr{A}$ (something less than or equal to $n!$). Then enumerate the full diagram of $\mathscr{A}$ until formulas are found that reveal why the other $n!$ - l permutations of the universe are not automorphisms. Let $\Theta(\vec{a})$ be the conjunction of everything enumerated by this stage. $\Theta(\vec{x})$ is the complete formula for $\vec{a}$. The complete formula for smaller tuples can be found by quantifying out certain constants. Of course, if e codes wrong guesses about n and l – or if $\mathscr{A}$ is not, in fact, finite – then the formula $\Theta(\vec{x})$ output is not correct. But if $\mathscr{A}$ is, in fact, finite, then one of the e will encode correct guesses for n and l, and then it outputs a correct $\Theta(\vec{x})$.

To define a $\Psi(\mathscr{A}, e)$ that works uniformly for both finite and infinite $\mathscr{A}$, we need the construction given in the next section. The above paragraph is intended only to acknowledge that a much easier functional Ψ works for all finite $\mathscr{A}$.

For Proposition 11, we let our domain be $\{0,1\}$, use 0 in place of the evens and 1 in place of the odds to get a finite atomic model.

2. Proof of Theorem 8 and its Corollaries

2.1. **Reference and Conventions.** This section builds on the write-up of the Effective Completeness Theorem given in Harizanov's survey paper in the Handbook of Recursive Mathematics, [3]. However, we change some of the notations used there to fit the extra parts of our construction more naturally.

We use a Henkin Construction. Let $C = \{c_0, c_1, c_2, \ldots, c_n, \ldots\}$ be the set of new constants not in the language $\mathscr{L}(T)$. Let $\{\sigma_e : e \in \omega\}$ be a computable enumeration of the set of all sentences in the language $\mathscr{L}(T) \cup C$. (We will assume some technical things about how these sentences are enumerated, e.g., about the appearance of the constants of C; see below.)

We will effectively enumerate a complete $(\mathscr{L}(T) \cup C)$-theory $\Gamma \supset T$. This theory will, as usual, have Henkin witnesses, so that the desired model $\mathscr{M}$ has a universe consisting of equivalence classes of the constants in C, where $c_i \equiv c_j$ iff $(c_i = c_j) \in \Gamma$. Of course, technically, as a model of T, our final model is just the reduct of $\mathscr{M}$ to the language of $\mathscr{L}(T)$.

We computably enumerate Γ as $\{\delta_0, \delta_1, \ldots, \}$, where we enumerate δ_s at some point during stage s of the construction. We denote $\delta_0 \wedge \ldots \wedge \delta_s$ by $\theta_s(\vec{c}_s)$, where $\vec{c}_s$ is the tuple of all *constants of C* mentioned in the conjunction.

As we enumerate the δ_s into Γ, we have to do more than ensure that Γ is a complete diagram that contains T and has Henkin witnesses. There are two major additional components to our construction that must be incorporated. First, for each computable function Φ, we try to diagonalize against Φ being an elementary embedding of $\mathscr{A}$ into $\mathscr{M}$; if we can succeed for all Φ, then we will have proven the theorem. To this end, we fix, as is standard, a computable enumeration of all computable functions Φ. Second, for each Φ, if it looks as though we are failing at all attempts to diagonalize against this function, then we computably construct, in stagewise fashion, what we hope will be a computable h_Φ witnessing that $\mathscr{A}$ is effectively atomic. When there is no ambiguity, we will drop the Φ subscript on h.

Just as in Harizanov's proof of the Effective Completeness Theorem, the model $\mathscr{M}$ is really not defined until after the stagewise construction is complete, when we can define the equivalence classes according to the set Γ. Nevertheless $\mathscr{M}$ will still be decidable, with either a finite universe or an infinite, computable universe, although we cannot say which ahead of time. Therefore, it will be more convenient to conceive of the Turing function Φ as having range not in the universe of $\mathscr{M}$ but in the set C of new constants $c_1, c_2, \ldots, c_n, \ldots$. This should not create any problems, because using our enumeration of Γ, there is an effective way of converting in either direction between a function $\Phi : \mathscr{A} \to C$ and a function $\Phi' : \mathscr{A} \to \mathscr{M}$. (Given Φ, and $a \in \mathscr{A}$, we define $\Phi'(a) := [\Phi(a)]$. Given Φ', and $a \in \mathscr{A}$, we search, using Γ, for the least element c in the equivalence class $\Phi'(a)$ and define $\Phi(a) := c$.) In fact, in our requirements below, we refer to Φ' as the obvious effective translation of Φ. Finally, for convenience, we assume that for all Φ, $dom(\Phi) \subseteq |\mathscr{A}|$, the computable universe of $\mathscr{A}$. (That is, we simply ignore whatever is in $dom(\Phi) - |\mathscr{A}|$.)

Recall that the standard enumeration of Turing computations of the form $\Phi_s(a) \downarrow = c$ is such that $a, c < s$. In our enumeration of the sentences in Γ, we will make sure that at least the constants $c_0, \ldots, c_s$ all appear in $\vec{c}_s$. This will ensure, simply as a matter of notational convenience, that no Turing computation produces an *output* (thought of as a member of C) that hasn't been at least technically mentioned already. (Again, this is just a matter of convenience.) Also, we assume that the enumeration of σ_e is such that all of the constants which appear in σ_e are among $c_0, \ldots, c_e$. Because of how and when we decide to enumerate sentences or their negations into Γ, these conventions will ensure that $\vec{c}_s = c_0, c_1, \ldots c_s$.

Finally, throughout much of the construction, variables are going to be substituted for constants, and vice-versa, in many formulas; and we are going to have to consider carefully which constants appearing in a formula are already in the range of a particular Φ_s and which are not. For instance, c_1 may be a constant appearing in the formula φ, a fact we denote by writing $\varphi(c_1)$. If the variable x_1 does not appear in φ, and we form the new formula by replacing every appearance of c_1 in φ with x_1, we will simply write $\varphi(x_1)$ for this new formula. Similarly, if $\vec{a} = dom(\Phi_s)$, and we break up the tuple $\vec{c}_s$ into the subtuples $\vec{c}_s - \Phi_s(\vec{a}), \Phi_s(\vec{a})$, then when we write $\theta_s(\vec{c}_s)$ as $\theta_s(\vec{c}_s - \Phi_s(\vec{a}), \Phi_s(\vec{a}))$, we DO NOT mean to suggest any deep or complex re-arrangement of the constants within the sentence.

And lastly, as is the convention with free variables, if we write something like $\sigma_e(\vec{c}_e)$, we mean to signify that all of the constants of C appearing in σ_e are among $\vec{c}_e$, and NOT to signify that all of these constants do, in fact, appear in σ_e.

2.2. **A requirement R_Φ requiring attention.** For each Turing function Φ, we have the requirement R_Φ:

$\neg(\Phi' : \mathscr{A} \prec \mathscr{M})$; OR
there is a computable function h_Φ with the following properties:

(1) the pairs in the graph of h_Φ are of the form $(\vec{a}, \varphi(\vec{x}))$, where $\vec{a} \in \mathscr{A}$, $\varphi(\vec{x})$ is a complete formula (relative to T), and $\mathscr{A} \models \varphi(\vec{a})$.
(2) for each $\vec{a} \in \mathscr{A}$, $\vec{a}$ is a sub-tuple of a tuple $\vec{a}'$ that appears in the domain of h_Φ.

For each requirement R_Φ, we refer to the *index* of the requirement and the index of Φ interchangeably. As usual, one requirement is higher priority than another if its index is lower. Recall from the previous section that $\mathscr{M}$ is a reduct from a Henkin construction built with new constants $c_0, c_1 \ldots$ and $\Phi'(a) = [\Phi(a)]$.

Note: Because of the conditions above for the function h_Φ, from h_Φ we could automatically construct a computable function g that accepts any tuple $\vec{a}$ from $\mathscr{A}$ and outputs a complete formula satisfied by $\vec{a}$. Given $\vec{a}$, by the second condition, find a tuple $\vec{a}'$ in the domain of h with $\vec{a} \subseteq \vec{a}'$, and let $h(\vec{a}')$ be $\varphi(\vec{x}')$. By the first condition, $\mathscr{A} \models \varphi(\vec{a}')$. Consider $\varphi(\vec{a}')$ as $\varphi(\vec{a}, \vec{a}' - \vec{a})$, let $\vec{x}$ be a tuple of new variables of the same length as $\vec{a}$, and let $\vec{y}$ be a tuple of new variables of the same length as $\vec{a}' - \vec{a}$. Then it is quickly verified that $\exists \vec{y} \varphi(\vec{x}, \vec{y})$ is a complete formula satisfied by $\vec{a}$.

Definition 5. *A requirement of the form R_Φ is completely satisfied by stage s if AT LEAST ONE of the following two conditions holds:*

(1) *Φ_s is not 1-1; OR*
(2) *If $\vec{a} = dom(\Phi_s)$, and we look at $\theta_s(\vec{c}_s)$ as $\theta_s(\vec{c}_s - \Phi_s(\vec{a}), \Phi_s(\vec{a}))$, and $\vec{y}$ is a tuple of new variables (not appearing among the variables in $\theta_s(\vec{c}_s)$) of the same length as $\vec{c}_s - \Phi_s(\vec{a})$, then $\mathscr{A} \not\models \exists \vec{y} \theta_s(\vec{y}, \vec{a})$. (Note: the substitution of $\vec{a}$ for $\Phi_s(\vec{a})$ is unambiguous, because, if the first condition does not hold, then Φ_s is assumed to be 1-1.)*

It is important for the reverse mathematics to note that a requirement R_Φ being completely satisfied by stage s is a computable condition. Therefore, a requirement R_Φ eventually becoming completely satisfied is a Σ_1 condition.

Definition 6. *The stage s approximation to h_Φ is denoted by $h_{\Phi,s}$ (or just h_s, if we're dropping the function subscripts). To initialize the stage $s-1$ approximation h_{s-1} at stage s simply means to re-define it to be equal to $\emptyset$.*

Since our construction informally involves substages, it might be the case $h_{\Phi,s-1}$ is initialized at a substage of stage s and at a later substage of stage s redefined to be nonempty.

Definition 7. *A requirement of the form R_Φ requires attention at stage s if*

(1) *R_Φ is not completely satisfied by stage s;*
(2) *Φ_s has converged on at least the input a_0, and one of the following is true:*
   - *$h_{\Phi,s-1} = \emptyset$ or has been initialized at this stage s, and $\Phi_s(a_0)\downarrow$; OR*
   - *Φ_s has converged on k inputs in the domain of $\mathscr{A}$, and $T \vdash \tau$, where τ expresses that there exist k distinct elements and there don't exist $k+1$ distinct elements; OR*

- *$h_{\Phi,s-1} \neq \emptyset$ and has not been initialized at this stage s; and the domain of Φ_s contains an initial segment of the universe of $\mathscr{A}$ that includes all of the tuples in $dom(h_{\Phi,s-1})$ and at least one more element.*

2.3. **Construction.** Stage 0:
$\delta_0 := (c_0 = c_0)$. All functions $h_{\Phi,0} := \emptyset$.

Stage $s = 2k+1$ for $k \in \omega$ (Henkin witness requirement):
Case 1: $\delta_k = \exists x \gamma(x) \wedge \tau$, where τ is a possibly empty conjunction of sentences of the form $(c_i = c_i)$. By convention, we know that the first element of C that does not appear in $\theta_{s-1}(\vec{c}_{s-1})$ is c_s. Define $\delta_s := \gamma(c_s) \wedge (c_s = c_s)$.
Case 2: Otherwise. Define $\delta_s := (c_s = c_s)$.

Stage $s = 2k+2$ for $k \in \omega$ (Completeness of the diagram requirement):
This portion of the construction, dedicated to the determination of δ_s at a positive even stage, employs an algorithm with a "loop" structure (that always terminates; see below). Notice that each step of the algorithm is computable.

Let e be the least e for which we have not explicitly decided whether to add σ_e or $\neg\sigma_e$ to Γ; i.e., at no previous stage t did $\delta_t := \sigma_e \wedge (c_t = c_t)$ or $\delta_t := \neg\sigma_e \wedge (c_t = c_t)$. We will work to make this determination at this stage, unless the complete satisfaction of a higher priority requirement R_Φ forces us to decide a different statement.

2.3.1. *Algorithm.*

(1) Set $\sigma^* := \sigma_e$ and $i^* := e$.
(2) Determine if the following is true: for $\gamma = \sigma^*$ or for $\gamma = \neg\sigma^*$, if $\vec{x}$ is a tuple of new variables (not appearing among the variables in $\theta_{s-1}(\vec{c}_{s-1}) \wedge \gamma(\vec{c}_{s-1})$) of the same length as $\vec{c}_{s-1}$, then $T \vdash \forall \vec{x}[(\theta_{s-1}(\vec{x}) \rightarrow \gamma(\vec{x})]$.
(3) If it is true for either $\gamma = \sigma^*$ or for $\gamma = \neg\sigma^*$, then only this γ is consistent with T and $\theta_{s-1}(\vec{c}_{s-1})$. Define $\delta_s := \gamma \wedge (c_s = c_s)$ and exit the algorithm. Otherwise, then each of σ^* and $\neg\sigma^*$ is consistent with T and $\theta_{s-1}(\vec{c}_{s-1})$, so proceed to the next step.
(4) Determine if there is any requirement R_Φ with index $\leq i^*$ that has not been completely satisfied up to this point in stage s. (Recall that a requirement can become completely satisfied at a given stage simply by the computation revealing Φ is not 1-1. Also recall that a requirement being completely satisfied by stage s is a computable condition.)
(5) If there is no such requirement, then define $\delta_s := \sigma^* \wedge (c_s = c_s)$, and exit the algorithm. Otherwise, proceed to the next step.
(6) *For each* function Φ associated with a requirement that has not been completely satisfied and has index $\leq i^*$, complete the following analysis:
   - Let $\vec{a} = dom(\Phi_s)$. (Recall that, by the conventions we mentioned above, $ran(\Phi_s) \subseteq \vec{c}_{s-1}$ and all of the constants appearing in σ_e are among $\vec{c}_{s-1}$, as well.)
   - Determine if one of the following conditions hold:
     (a) For $\gamma = \sigma^*$ or for $\gamma = \neg\sigma^*$, if we look at $\theta_{s-1}(\vec{c}_{s-1}) \wedge \gamma$ as $\rho(\vec{c}_{s-1} - \Phi_s(\vec{a}), \Phi_s(\vec{a}))$, and if $\vec{y}$ is a tuple of new variables (not appearing among the variables in $\theta_{s-1}(\vec{c}_{s-1}) \wedge \gamma$) of the same length as $\vec{c}_{s-1} - \Phi_s(\vec{a})$, then $\mathscr{A} \not\models \exists \vec{y} \rho(\vec{y}, \vec{a})$.

(**COMMENT**: Each of the sentences, σ^* and $\neg\sigma^*$, are consistent with T and $\theta_{s-1}(\vec{c}_{s-1})$, but one of them would make it impossible for Φ' to be an elementary embedding.)

(b) The previous condition does not hold, but for $\gamma = \sigma^*$ or for $\gamma = \neg\sigma^*$, if
- we look at $\theta_{s-1} \wedge \gamma$ as the formula $\theta_{s-1}(\vec{c}_{s-1} - \Phi_s(\vec{a}), \Phi_s(\vec{a})) \wedge \gamma(\vec{c}_{s-1} - \Phi_s(\vec{a}), \Phi_s(\vec{a}))$;
- $\vec{x}$ is a tuple of new variables of the same length as $\Phi_s(\vec{a})$;
- $\vec{y}$ is a tuple of new variables of the same length as $\vec{c}_{s-1} - \Phi_s(\vec{a})$,

then $T \vdash \exists\vec{x}[\exists\vec{y}(\theta_{s-1}(\vec{y},\vec{x})) \wedge \forall\vec{y}(\theta_{s-1}(\vec{y},\vec{x}) \rightarrow \gamma(\vec{y},\vec{x}))]$.
(**COMMENT**: Since condition a) doesn't hold, we know that, based on what has been declared so far in θ_{s-1}, each of σ^* and $\neg\sigma^*$ is consistent with $\vec{a} \mapsto \Phi'(\vec{a})$ as part of a potential elementary embedding. However, in this case, T guarantees that there is a tuple $\vec{x}$ of elements which satisfies the existential statements necessary to be consistent with θ_{s-1}, but which can accommodate only one of σ^* or $\neg\sigma^*$. Therefore, defining the non-trivial part of δ_s to be $\gamma \wedge \forall\vec{y}(\theta_{s-1}(\vec{y},\Phi_s(\vec{a})) \rightarrow \gamma(\vec{y},\Phi_s(\vec{a})))$. This would make it impossible for Φ' to be an elementary embedding since now the types of $\vec{a}$ in $\mathscr{A}$ and $\Phi'(\vec{a})$ in $\mathscr{M}$ are different.

(7) If all of the functions Φ that are considered don't satisfy any of the above conditions, then define $\delta_s := \sigma^* \wedge (c_s = c_s)$, and exit the algorithm. Otherwise, proceed to the next step.

(8) REDEFINE i^* to be the index of the highest priority requirement that was considered and satisfies one of conditions a) or b) under the second bullet of step (6). In the rest of the steps, Φ refers specifically to the Turing function for this requirement.

(9) If the function satisfies Step (6) condition a), then, for the appropriate γ that makes the condition satisfied (either σ^* or $\neg\sigma^*$, and there is no ambiguity which), define $\delta_s := \gamma \wedge (c_s = c_s)$; and exit the algorithm. Otherwise, proceed to the next step.

(10) If the function satisfies condition b), then it is possible that the satisfaction could be due to either $\gamma = \sigma^*$ or $\gamma = \neg\sigma^*$; if this is the case, show (arbitrary) preference for $\gamma = \sigma^*$; if not, then the γ that makes the condition satisfied is unambiguous. Now, for this γ, REDEFINE $\sigma^* := \gamma \wedge \forall\vec{y}(\theta_{s-1}(\vec{y},\Phi_s(\vec{a})) \rightarrow \gamma(\vec{y},\Phi_s(\vec{a})))$. And, with this new index i^* and this new σ^*, return to the second step of the algorithm.

(**COMMENT**: Why redefine σ^* instead of just defining δ_s to be the conjunction of this new σ^* and $(c_s = c_s)$? If δ_s were defined in this way, then the respective requirement would be satisfied; however, because this δ_s was not analyzed in the earlier steps of the algorithm, it is possible that, in adding this δ_s, as opposed to the negation of the non-trivial part, an opportunity was missed to completely satisfy a higher priority requirement. Thus, the need to redefine σ^* and restart the algorithm. Note that if no higher priority requirement meets one of the conditions of Step (6) in the next iteration, then in this next iteration the algorithm we get past Step (3) since the new σ^* is a stronger consistent clause than the old σ^*; we will exit at Step (9); δ_s will be defined as suggested, and the respective requirement will be completely satisfied.)

Notice that for each successive loop through the algorithm, the index i^* is strictly less than it was before, so the algorithm must terminate, and δ_s is well-defined.

2.3.2. *Definition/Construction of the stage s approximations to the potential isomorphisms.* If R_Φ is the highest priority requirement (with index less than or equal to e) that was not

completely completely satisfied at stage $s-1$ and is completely satisfied during this stage s, then initialize all functions h_{s-1} associated with all lower priority requirements. If there is no such requirement, then simply initialize all functions h_{s-1} associated with requirements R_Φ with index greater than or equal to e.

As the final part of the construction at positive even stages, we define $h_{\Phi,s}$ on the requirements R_Φ that still require attention at stage s (even after our work at stage s so far). We will focus on one of these and refer to it as h_s from now on. (But again, we would do this work for *every* R_Φ that still requires attention at stage s, which, by definition, is a finite number of requirements.)

Let $\vec{a} = dom(\Phi_s)$ (thought of as an ordered tuple, not just a set). By the assumption that R_Φ requires attention at this stage, if h_{s-1} had been initialized above during stage s then $\vec{a}$ contains at least a_0; if we know at this stage that $\mathscr{A}$ is finite model of size k then $\vec{a}$ is the entire universe of $\mathscr{A}$; or $\vec{a}$ contains an initial segment of the universe of $\mathscr{A}$ that includes all tuples in $dom(h_{s-1})$ and at least one more element.

Recall that we are automatically conceiving of $\Phi_s(\vec{a})$ as being constants from C and among $\vec{c}_s$. Consider the sentence $\theta_s(\vec{c}_s)$. We look at $\theta_s(\vec{c}_s)$ as $\theta_s(\vec{c}_s - \Phi_s(\vec{a}), \Phi_s(\vec{a}))$. Let $\vec{y}$, $\vec{x}$ be two new, disjoint tuples of variables (not appearing among the variables of θ_s) of the same length as $\vec{c}_s - \Phi_s(\vec{a})$, $\Phi_s(\vec{a})$, respectively. Define $h_s(\vec{a}) := \phi(\vec{x}) = \exists \vec{y} \theta_s(\vec{y}, \vec{x})$. (Clearly, $\mathscr{A} \models \phi(\vec{a})$, because R_Φ has not been completely satisfied). The majority of the Verification subsection below is devoted to proving that – for any requirement R_Φ to receive attention infinitely often, and after finitely much initialization due to higher priority requirements has stopped – the formulas $\phi(\vec{x})$ are complete.)

Finally, for all other functions $h_{\hat{\Phi}}$ associated with other requirements that have not already been initialized at this stage s, let $h_{\hat{\Phi},s} := h_{\hat{\Phi},s-1}$.

This concludes the construction.

2.4. **Verification.**

Lemma 12. *$\mathscr{M}$ is decidable and $\mathscr{M} \models T$.*

Proof. The construction is an expansion on the standard Henkin construction. All of the components that guarantee the claim of the lemma are included. First, the construction constructs a complete theory Γ in the expanded language by eventually adding σ_e or $\neg\sigma_e$ (with a trivial conjunct of the form $(c_s = c_s)$ appended) to Γ. It is true that, even if σ_e is the original sentence considered at a particular even stage s, the above algorithm, because of Step (6) condition b), might redefine δ_s to be a sentence that implies neither σ_e nor $\neg\sigma_e$. Now, without any such delays, the sentence σ_e would be decided by stage $2e+2$ at the latest. However, the decision can be delayed only by R requirements with index $< e$. Therefore, stage $s = 4e+4$ provides an upper bound on the stage by which σ_e or $\neg\sigma_e$ (with a trivial conjunct appended) is included in Γ.

Second, the algorithm employed at even stages, which is not part of the standard Henkin construction, always terminates, and it preserves consistency with T throughout. Third, the odd stages simply guarantee the existence of Henkin witnesses. Fourth, as in the standard Henkin construction, elements of the model are equivalence classes of constant symbols.

Finally, the definitions of the parts of functions $h_{\Phi,s}$ is an additional component of our construction, but this part of the construction does not affect choices in how we build $\mathscr{M}$ and the complete theory Γ.

□

Lemma 13. *If every requirement R_Φ requires attention only finitely often, then $\mathscr{A}$ is not embeddable by a computable embedding into $\mathscr{M}$.*

Proof. Assume every requirement requires attention only finitely often. By definition, there are only two reasons that a requirement R_Φ stops requiring attention by stage s. First, because R_Φ becomes completely satisfied by s, so one of the following is true:

- the corresponding $\Phi' : \mathscr{A} \to \mathscr{M}$ is not 1-1; OR
- for some tuple $\vec{a} \in \mathscr{A}$ and some forumla $\varphi(\vec{x})$, $\mathscr{A} \models \varphi(\vec{a})$ and $\mathscr{M} \models \neg\varphi(\Phi'(\vec{a}))$.

(See the above section on conventions and facts regarding the connection between Φ and Φ'.)

Second, because $dom(\Phi)$ does not include the universe of $\mathscr{A}$, and hence Φ' does not include the universe of $\mathscr{A}$.

Now, as the section on conventions explained, every computable function $f : \mathscr{A} \to \mathscr{M}$ is equal to Φ' for some $\Phi : \mathscr{A} \to C$. Therefore, if *every requirement* R_Φ stops requiring attention by some stage s, then every computable function from $\mathscr{A}$ to $\mathscr{M}$ fails to be an elementary embedding.

□

Remark 14. *Therefore, for the rest of this verification, we assume that there is a requirement R_Φ and stages $s^* \leq s$ with the following three properties:*

- *R_Φ requires attention infinitely often.*
- *s^* is the least stage t with the following three properties:*
  - *$t >$ the index of Φ*
  - *for each stage $u \geq t$, it is NOT the case that a requirement $R_{\hat{\Phi}}$ of priority higher than that of R_Φ first becomes completely satisfied at u;*
  - *for each e less than or equal to the index of Φ, the algorithm in sub-subsection 2.3.1 has explicitly added σ_e or $\neg\sigma_e$ to Γ before stage t.*
- *s is the first stage $\geq s^*$ so that R_Φ requires attention at s.*

This requirement and these stages will be of particular importance as we state and prove the uniform version of this theorem below.

With this requirement R_Φ and these stages s^* and s *fixed*, we must prove that $h_\Phi = \bigcup_{t \geq s} h_{\Phi,t}$ has the properties stated near the beginning of Subsection 2.2. We will refer to this function simply as h from now on, and its stage t approximation as h_t. The following long lemma will essentially complete this proof. Recall the notation from subsection 2.1 that θ_w is the conjunction of all sentences of Γ enumerated by the end of stage w.

Lemma 15. *For each stage $t \geq s$ for which R_Φ requires attention, we recall or consider the following notational conventions:*

(1) *$\vec{a}_t = dom(\Phi_t)$;*
(2) *$\vec{x}_t$ is a tuple of new variables (i.e., not appearing in θ_t) of the same length as $\vec{a}_t$ (which is the same length as $\Phi(\vec{a}_t)$ since Φ is 1-1);*
(3) *for each $u \geq t$, $\vec{y}_u$ is a tuple of new variables (i.e., not appearing in θ_u) of the same length as $\vec{c}_u - \Phi_t(\vec{a}_t)$;*
(4) *$h_t(\vec{a}_t) = \phi(\vec{x}_t) = \exists\vec{y}_t\theta_t(\vec{y}_t, \vec{x}_t)$;*
(5) *for each $u \geq t$, we consider $\theta_u(\vec{c}_u) = \theta_u(\vec{c}_u - \Phi_t(\vec{a}), \Phi_t(\vec{a}))$, and we assume (making trivial changes, if necessary) that θ_u does not use any of the variables in the tuple $\vec{x}_t$.*

Then for all $u \geq t$, $\mathscr{A} \models \exists\vec{y}_u\theta_u(\vec{y}_u, \vec{a}_t)$ and $T \vdash \phi(\vec{x}_t) \to \exists\vec{y}_u\theta_u(\vec{y}_u, \vec{x}_t)$.
(Note: in (3), (4), and the conclusion of the lemma, the different subscripts u and t are intentional.)

Proof. Let $t \geq s$ be a stage where R_Φ requires attention.

For all $u \geq t$, the first part of the statement must be true. Assume otherwise. Then, since $u \geq t$, $dom(\Phi_t) \subseteq dom(\Phi_u)$; and so, it would certainly be the case that if $\vec{a} = dom(\Phi_u)$, and we look at $\theta_u(\vec{c}_u)$ as $\theta_u(\vec{c}_u - \Phi_u(\vec{a}), \Phi_u(\vec{a}))$, and $\vec{y}$ is a tuple of new variables (not appearing among the variables in $\theta_u(\vec{c}_{u+1})$) of the same length as $\vec{c}_u - \Phi_u(\vec{a})$, then $\mathscr{A} \not\models \exists \vec{y} \theta_u(\vec{y}, \vec{a})$. Therefore, R_Φ would be completely satisfied, and would no longer receive attention. Therefore, for all $u \geq t$, $\mathscr{A} \models \exists \vec{y}_u \theta_u(\vec{y}_u, \vec{a}_t)$.

We prove the second part of the statement by induction on $u \geq t$. For $u = t$, of course, $\phi(\vec{x}_t)$ and $\exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$ are exactly the same formula, so the statement is obviously true. Assume that for all u' with $t \leq u' \leq u$, $T \vdash \phi(\vec{x}_t) \rightarrow \exists \vec{y}_{u'} \theta_{u'}(\vec{y}_{u'}, \vec{x}_t)$. We must show that $T \vdash \phi(\vec{x}_t) \rightarrow \exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)$.

Recall that the statement $\theta_{u+1}(\vec{c}_{u+1})$ is just the statement $\theta_u(\vec{c}_u) \wedge \delta_{u+1}$, where δ_{u+1} is the sentence added at stage $u+1$ of the construction given in subsection 2.3. The form of this sentence δ_{u+1} depends on the number $u+1$. We consider the cases.

Case 1a) $u+1 = 2k+1$ for some $k \in \omega$, and $\delta_k = \exists x \gamma(x) \wedge \tau$, where τ is a conjunction of sentences of the form $(c_i = c_i)$. Then $\delta_{u+1} = \gamma(c_{u+1}) \wedge (c_{u+1} = c_{u+1})$. Since $u > k$, the sentence δ_k is already included as one of the conjuncts of $\theta_u(\vec{c}_u)$. Therefore, $\exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$ has the form $\exists \vec{y}_u [\ldots \wedge \exists x \gamma(x) \wedge \ldots]$, where whatever elements of $\vec{c}_k (\subseteq \vec{c}_u)$ appearing in $\gamma(x)$ have been replaced by the corresponding elements of $\vec{y}_u$ or $\vec{x}_t$, according to our normal substitution conventions. In particular, we assume that the variable x in $\gamma(x)$ is not one of the variables in the tuple $\vec{x}_t$.

Similarly, since $\theta_{u+1}(\vec{c}_{u+1}) = \theta_u(\vec{c}_u) \wedge \delta_{u+1}$, and $\delta_{u+1} = \gamma(c_{u+1}) \wedge (c_{u+1} = c_{u+1})$, $\exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)$ has the form $\exists \vec{y}_u \exists y_{u+1} [\ldots \wedge \exists x \gamma(x) \wedge \ldots \wedge \gamma(y_{u+1}) \wedge (y_{u+1} = y_{u+1})]$, where all other substitutions of the variables of $\vec{y}_u$ and $\vec{x}_t$ in the two appearances of γ are exactly the same. Furthermore, by our conventions, neither $\gamma(x)$ nor any of the other conjuncts in θ_u makes any mention of c_{u+1}. Therefore, the formula $\exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)$ and the formula $\exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$ are logically equivalent. Since $T \vdash \phi(\vec{x}_t) \rightarrow \exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$, $T \vdash \phi(\vec{x}_t) \rightarrow \exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)$.

Case 1b): $u+1 = 2k+1$ for some $k \in \omega$, but δ_k does not have the above form of an existential sentence (with a trivial τ attached). In this case δ_{u+1} is just the trivial sentence $(c_{u+1} = c_{u+1})$, so again, trivially, the formula $\exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)$ and the formula $\exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$ are logically equivalent. Therefore, $T \vdash \phi(\vec{x}_t) \rightarrow \exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)$.

Case 2: $u+1 = 2k+2$. Therefore, δ_{u+1} is determined by the algorithm in sub-subsection 2.3.1. That is, $\delta_{u+1} = \pm \sigma^* \wedge (c_{u+1} = c_{u+1})$ for σ^* relative to the last iteration of the algorithm at stage $u+1$. For the rest of this proof, we refer to the non-trivial part of δ_{u+1} as γ; i.e., $\gamma = \sigma^*$ or $\gamma = \neg \sigma^*$. Note that c_{u+1} does not appear in γ.

If the algorithm at stage $u+1$ at this last iteration exits at Step 3, then it is the case that $T \vdash \forall \vec{z} [\theta_u(\vec{z}) \rightarrow \gamma(\vec{z})]$. Therefore, since by induction hypothesis, $T \vdash \phi(\vec{x}_t) \rightarrow \exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$, and $\theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t) = \theta_u(\vec{y}_u, \vec{x}_t) \wedge \gamma(\vec{y}_u, \vec{x}_t) \wedge (y_{u+1} = y_{u+1})$, $T \vdash \phi(\vec{x}_t) \rightarrow \exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)$.

It cannot be the case that the algorithm exits at Step 9 for the sake of Φ, for then Φ would be completely satisfied and would stop receiving attention.

Finally, for the rest of this case, we assume, in order to obtain a contradiction, that $T \nvdash [\phi(\vec{x}_t) \rightarrow \exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)]$. That is, we assume that

$$T \vdash \exists \vec{x}_t [\phi(\vec{x}_t) \wedge \forall \vec{y}_{u+1} (\neg \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t))].$$

Again, since $\theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t) = \theta_u(\vec{y}_u, \vec{x}_t) \wedge \gamma(\vec{y}_u, \vec{x}_t) \wedge (y_{u+1} = y_{u+1})$, $\forall \vec{y}_{u+1} (\neg \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t))$ is logically equivalent to $\forall \vec{y}_u (\neg \theta_u(\vec{y}_u, \vec{x}_t) \vee \neg \gamma(\vec{y}_u, \vec{x}_t))$, which is logically equivalent to $\forall \vec{y}_u (\theta_u(\vec{y}_u, \vec{x}_t) \rightarrow \neg \gamma(\vec{y}_u, \vec{x}_t))$. Therefore, $T \vdash \exists \vec{x}_t [\phi(\vec{x}_t) \wedge \forall \vec{y}_u (\theta_u(\vec{y}_u, \vec{x}_t) \rightarrow \neg \gamma(\vec{y}_u, \vec{x}_t))]$.

Moreover, by induction hypothesis, $T \vdash \phi(\vec{x}_t) \to \exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$. And so,

$$T \vdash \exists \vec{x}_t [\exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t) \wedge \forall \vec{y}_u (\theta_u(\vec{y}_u, \vec{x}_t) \to \neg\gamma(\vec{y}_u, \vec{x}_t))].$$

Now, except for the use of $\neg\gamma$ instead of γ, this last statement is almost exactly what appears at stage $u+1$ in Condition b) under the second bullet point of Step (6) of the algorithm, which is $T \vdash \exists \vec{x}[\exists \vec{y}(\theta_u(\vec{y}, \vec{x})) \wedge \forall \vec{y}(\theta_u(\vec{y}, \vec{x}) \to \gamma(\vec{y}, \vec{x}))]$. However, we have to be careful, because the length of the tuples is not correct; i.e., at stage $u+1$, the length of $\vec{x}$ mentioned in the algorithm is the same as the length of the range of Φ_{u+1}, and the length of $\vec{y}$ mentioned in the algorithm is the same as the length of ($\vec{c}_u -$ the range of Φ_{u+1}). Notice, because $u \geq t$, that the length of $\vec{x}_t$ is less than or equal to that of $\vec{x}$ in the algorithm, so the length of $\vec{y}_u$ is greater than or equal to that of $\vec{y}$ in the algorithm. Nevertheless, the following paragraph establishes that, indeed, $T \vdash \exists \vec{x}[\exists \vec{y}(\theta_u(\vec{y}, \vec{x})) \wedge \forall \vec{y}(\theta_u(\vec{y}, \vec{x}) \to \neg\gamma(\vec{y}, \vec{x}))]$.

Rather than working purely syntactically, it is easier to consider an arbitrary model $\mathscr{D}$ of the theory T. Since $T \vdash \exists \vec{x}_t [\exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t) \wedge \forall \vec{y}_u (\theta_u(\vec{y}_u, \vec{x}_t) \to \neg\gamma(\vec{y}_u, \vec{x}_t))]$, there is a $\vec{d}_t \in \mathscr{D}$ of the same length as $\vec{x}_t$ and a $\vec{d}'$ of the same length as $\vec{y}_u$ so that $\mathscr{D} \models \theta_u(\vec{d}', \vec{d}_t)$ and $\mathscr{D} \models \forall \vec{y}_u (\theta_u(\vec{y}_u, \vec{d}_t) \to \neg\gamma(\vec{y}_u, \vec{d}_t))$. (It is possible that there is repetition of elements within or between these two tuples of $\mathscr{D}$; for instance, the formula θ may not say that all of the elements in $\vec{x}_t$ are unequal.) Next, simply "regroup" the elements of $\vec{d}_t$ and $\vec{d}'$ to get new tuples $\vec{b}$ and $\vec{b}'$ in $\mathscr{D}$ of the length of $\vec{x}$ and $\vec{y}$, respectively, in the algorithm. (We are not talking about any deep re-arrangement here; we're just looking at what elements of $\mathscr{D}$ are substituted for what variables in θ_u and γ. Again, repetition of elements within and/or between the tuples $\vec{b}$ and $\vec{b}'$ may occur.) Notice, since $\vec{x}$ is at least as long as $\vec{x}_t$, that $\vec{b}$ contains all of $\vec{d}_t$, and possibly more. Clearly, since $\mathscr{D} \models \theta_u(\vec{d}', \vec{d}_t)$, $\mathscr{D} \models \exists \vec{y} \theta_u(\vec{y}, \vec{b})$. Now assume that there is $\vec{b}'' \in \mathscr{D}$ of the same length as $\vec{y}$ such that $\mathscr{D} \models (\theta_u(\vec{b}'', \vec{b}) \wedge \gamma(\vec{b}'', \vec{b}))$. But since $\vec{b}$ contains all of $\vec{d}_t$, if we simply make the "reverse" regrouping of $\vec{b}'', \vec{b}$ to get $\vec{d}^*, \vec{d}_t$, then we'd have $\mathscr{D} \models (\theta(\vec{d}^*, \vec{d}_t) \wedge \gamma(\vec{d}^*, \vec{d}_t))$, which contradicts the fact that $\mathscr{D} \models \forall \vec{y}_u [\theta_u(\vec{y}_u, \vec{d}_t) \to \neg\gamma(\vec{y}_u, \vec{d}_t)]$. Hence, the assumption of the existence of $\vec{b}''$ is false. That is, $\mathscr{D} \models \exists \vec{y} \theta_u(\vec{y}, \vec{b})$ and $\mathscr{D} \models \forall \vec{y} [\theta_u(\vec{y}, \vec{b}) \to \neg\gamma(\vec{y}, \vec{b})]$. And so, $\mathscr{D} \models \exists \vec{x}[\exists \vec{y}(\theta_u(\vec{y}, \vec{x})) \wedge \forall \vec{y}(\theta_u(\vec{y}, \vec{x}) \to \neg\gamma(\vec{y}, \vec{x}))]$. Since $\mathscr{D}$ was an arbitrary model of T, we can conclude that $T \vdash \exists \vec{x}[\exists \vec{y}(\theta_u(\vec{y}, \vec{x})) \wedge \forall \vec{y}(\theta_u(\vec{y}, \vec{x}) \to \neg\gamma(\vec{y}, \vec{x}))]$.

Now, then, we must ask why δ_{u+1} was defined to be $\gamma \wedge (c_{u+1} = c_{u+1})$. It cannot be that the algorithm stopped and exited at Step (3), for then, as noted above, the statement we're trying to prove would be true. Moreover, by the assumptions about stage s, the index of Φ is small enough that Φ will be considered in the *first iteration* of the algorithm at Step (6), since $u+1 > s$. Therefore, Φ would be considered at Step (6) of *all iterations* of the algorithm at stage $u+1$ unless the algorithm re-defines i^* and exits the algorithm in order to completely satisfy a higher priority requirement. But by the assumption about stage s, all higher priority requirements that will ever be completely satisfied already have been completely satisfied. Therefore, no higher priority requirement at stage $u+1$ (or any later stage) can be not completely satisfied and meet one of the conditions in Step (6). And again, as noted above, it cannot be that the algorithm exits at Step (9) for the sake of Φ. Consequently, $\mathscr{A} \models \exists \vec{y}[\theta_u(\vec{y}, dom(\Phi_{u+1})) \wedge \gamma(\vec{y}, dom(\Phi_{u+1}))]$ and $\mathscr{A} \models \exists \vec{y}[\theta_u(\vec{y}, dom(\Phi_{u+1})) \wedge \neg\gamma(\vec{y}, dom(\Phi_{u+1}))]$. Moreover, in the above paragraph, we saw that $T \vdash \exists \vec{x}[\exists \vec{y}(\theta_u(\vec{y}, \vec{x})) \wedge \forall \vec{y}(\theta_u(\vec{y}, \vec{x}) \to \neg\gamma(\vec{y}, \vec{x}))]$. Therefore, at the iteration of the algorithm with this particular σ^*, Φ *does* satisfy condition (b) under the second bullet point of Step (6). And since no higher priority requirements become completely satisfied at stage $u+1$, this means that δ_{u+1} should NOT have been defined to be $\gamma \wedge (c_{u+1} = c_{u+1})$. Instead, δ_{u+1} should have been defined to be $\neg\gamma \wedge \forall \vec{y}(\theta_u(\vec{y}, ran(\Phi_{u+1})) \to \neg\gamma(\vec{y}, ran(\Phi_{u+1}))) \wedge (c_{u+1} =$

c_{u+1}). But then R_Φ would become completely satisfied at stage $u+1$ and hence would never again require attention. This is a contradiction. Therefore, the additional assumption must be false. That is, $T \vdash [\phi(\vec{x}_t) \to \exists \vec{y}_{u+1} \theta_{u+1}(\vec{y}_{u+1}, \vec{x}_t)]$. □

If we continue all of the notation from the previous lemma, then almost instantly we obtain the following as a corollary:

Corollary 16. *For all $t \geq s$, and for all $\rho(\vec{x}_t)$ in the original language,*

(1) $T \vdash \phi(\vec{x}_t) \to \rho(\vec{x}_t)$ *if* $\mathscr{A} \models \rho(\vec{a}_t)$ *and*
(2) $T \vdash \phi(\vec{x}_t) \to \neg\rho(\vec{x}_t)$ *if* $\mathscr{A} \models \neg\rho(\vec{a}_t)$

Therefore, for each $\vec{a}_t$, $h_t(\vec{a}_t) := \phi(\vec{x}_t)$ is a complete formula.

Proof. Fix $t \geq s$ and $\rho(\vec{x}_t)$ in the original language. Note that the σ_e enumerate all sentences in the expanded language, and for each e, there is a u so that $\pm\sigma_e$ is one of the conjuncts of θ_u. Therefore, there is some $u \geq t$ such that $\exists \vec{y}_u \theta_u(\vec{y}_u, \vec{x}_t)$ looks like $\exists \vec{y}_u[\ldots \wedge \rho(\vec{x}_t) \wedge \ldots]$ or like $\exists \vec{y}_u[\ldots \wedge \neg\rho(\vec{x}_t) \wedge \ldots]$. Now apply the conclusion of the previous lemma. □

Finally, note that h is not initialized at any stage $t \geq s$, and, by assumption, R_Φ requires attention infinitely often. Therefore, by definition of requiring attention, if $|\mathscr{A}|$ is finite, then $|\mathscr{A}| \subseteq dom(\Phi_t)$ for some $t \geq s$. If, instead, $|\mathscr{A}|$ is infinite, then, by definition, for each stage $t \geq s$ where R_Φ requires attention, $dom(\Phi_t)$ includes an *initial segment* of the universe of $\mathscr{A}$ that includes all tuples in the domain of h_{t-1} and at least one more element. And by construction, at a stage $t \geq s$ where R_Φ requires attention, h_t is defined on $dom(\Phi_t) = \vec{a}_t$ (thought of as a tuple of elements). Therefore, whether $|\mathscr{A}|$ is finite or infinite, for every tuple $\vec{a}$ in $\mathscr{A}$, there is a $t \geq s$ so that $\vec{a} \subseteq \vec{a}_t$. This fact and the previous corollary combine to demonstrate that $\mathscr{A}$ is effectively atomic. This concludes the proof of Theorem 8. □

Proof of Corollary 10. Note that once we have fixed a requirement R_Φ, a stage s^* and a stage s as in Remark 14, the above construction produces the needed h such that $h(\vec{a})$ is the complete formula for $\vec{a}$. The h is constructed uniformly in our model $\mathscr{A}$, a requirement R_Φ, a stage s^* and a stage s. We can think of latter three items as coded by e. Hence the construction defines a computable Ψ such that $\Psi(\mathscr{A}, e)$ is (the code for) the corresponding h. So either there is a requirement R_Φ, a stage s^* and a stage s as in Remark 14, which are then coded by e, and $\Psi(\mathscr{A}, e)$ is the computable function witnessing that $\mathscr{A}$ is effectively atomic; or there is a decidable $\mathscr{M} \models T$, such that there is no computable elementary embedding of $\mathscr{A}$ into $\mathscr{M}$. □

3. Implications in Reverse Mathematics

The main theorem of this paper, Theorem 8, is that Effectively Prime ⇒ Effectively Atomic. In the context of Reverse Mathematics, or, more precisely, in some model of second order arithmetic, to say a model $\mathscr{A}$ of a theory T is "effectively prime" is really just to say that it is prime inside the model of second order arithmetic; that is, the necessary embeddings establishing that $\mathscr{A}$ is prime must be among the functions of the model of second order arithmetic.

However, as we have stressed above, to say that $\mathscr{A}$ is effectively atomic is not the same as saying that it is atomic, because the definition of "atomic" does not include the existence of a single function that "picks out" a complete formula for each tuple. By "effectively atomic" in a model of second order arithmetic we mean that the function picking out the complete formulas exists inside this model of second order arithmetic.

The theorem's more technical statement is that, given any $\emptyset$-decidable $\mathscr{A}$, i.e., a structure whose complete diagram is computable ($\leq_T \emptyset$), there is a $\emptyset$-decidable $\mathscr{M}$ such that either, for all $\Phi' \leq_T \emptyset$, Φ' does not witness that $\mathscr{A} \prec \mathscr{M}$, or there is a $h \leq_T \emptyset$ witnessing that $\mathscr{A}$ is effectively atomic. In the construction, we used that φ_e is a listing, computable (in $\emptyset$), of all functions that are partial computable (in $\emptyset$). This basic fact follows immediately from the Enumeration Theorem. In fact, every possible Φ' appears as infinitely many φ_e, and so, we could use this listing to try to diagonalize against all Φ'. If we were able to diagonalize against all Φ', then we would have that $\mathscr{A}$ is not effectively prime. Otherwise, if we were not, then $\mathscr{A}$ would be effectively atomic. So, the construction is a "failed" priority argument.

Corollary 17. *Effectively Prime $\Rightarrow$ Effectively Atomic holds in all topped models of RCA_0, i.e., all models containing a set X in which all other sets are computable. Hence, Prime Uniqueness holds in all topped models of RCA_0.*

Proof. First we will consider only standard models. Relativizations of the first statement in the above paragraph and Enumeration Theorem replace the $\emptyset$ with the set X and both relativizations remain true. Therefore, we immediately conclude that Effectively Prime $\Rightarrow$ Effectively Atomic holds in all standard, topped models of RCA_0.

Since the relativized Enumeration Theorem holds in RCA_0, a careful analysis of the proof and its induction arguments is needed for nonstandard topped models. The key is that Σ_1 bounding and bounded Σ_1 comprehension holds in RCA_0. The fact that Σ_1 bounding holds in RCA_0 is well known. Recall that bounded Σ_1 comprehension is for all Σ_1 formulas, $\varphi(x)$, and all k, there is a an Z such that $i \in Z$ iff $i < k$ and $\varphi(i)$. For details of why bounded Σ_1 comprehension holds in RCA_0 see Theorem II.3.9 of [5]. There are a few places where these concepts are used.

The first is to show δ_s exists and our algorithm at each stage terminates. For $l \leq s$, it is Σ_1 in RCA_0 to determine if during stage s there is a substage (a loop though the algorithm) where $i^* = l$. This Σ_1 formula in RCA_0 says that there is a series of formulas (in our fixed language) and substages such that this series witness that $i^* = l$. This Σ_1 formula needs to be coded carefully using some type of course of values recursion. By bounded Σ_1 comprehension the finite set X of such l exists. Hence is possible to find the least l where $l = i^*$ and therefore δ_s exists.

The second place where Σ_1 bounding and bounded Σ_1 comprehension is used is in Remark 14 to show a requirement R_Φ, a stage s^* and a stage s as in Remark 14 exist. Assume that there is a computable elementary embedding of $\mathscr{A}$ into $\mathscr{M}$. Let Φ be any (but not necessarily the least) witness of this embedding. So R_Φ will require attention infinitely often. A requirement being completely satisfied is Σ_1. By bounded Σ_1 comprehension and Σ_1 bounding, there is a stage s' where every requirement with higher priority than R_Φ which is going to be satisfied will be satisfied by stage s'. Now it is straightforward to find $s^* \geq s'$ and s as in the Remark.

We also need Σ^0_1 induction to ensure that if R_Φ requires attention infinitely often then $dom(\Phi)$ is $|\mathscr{A}|$, see the paragraph after the proof of Corollary 16. Consider the set of l such that there is stage s where the length of the largest initial segment included in $dom(\Phi_t)$ is greater than l. This is a Σ^0_1 definable cut and hence $\mathbb{N}$.

Therefore, Effectively Prime $\Rightarrow$ Effectively Atomic holds in all non-standard, topped models of RCA_0, as well. $\square$

Theorem 8 does not necessarily hold in a non-topped model of RCA_0. The use of the top X was essential in the above proof. We are grateful to Richard Shore and Leo Harrington for this observation and for pointing it out to us.

In fact, the following example shows that Effectively Prime $\Rightarrow$ Effectively Atomic does not always hold. We thank David Belanger for this observation which is connected to his paper [1].

Lemma 18. *Let $\mathscr{S}$ be a Scott Set such that for some $X \in \mathscr{S}$, $X' \notin \mathscr{S}$, then "Effectively Prime $\Rightarrow$ Effectively Atomic" does not hold in $\mathscr{S}$ (when $\mathscr{S}$ is viewed as the second order part of a standard model of second order arithmetic).*

Proof. Let T be the theory from Proposition 6 relativized to the above X. Let $\mathscr{M}$ be a countable model of T in $\mathscr{S}$. $\mathscr{M}$'s isomorphism class is determined by the number of elements which realize the non principal type $p(x) = \{\neg R_i(x) | i \in \omega\}$. The prime model $\mathscr{A}$ has no elements realizing this type. $\mathscr{A} \in \mathscr{S}$ since $\mathscr{A}$ can be computably built from X. Computably in $\mathscr{M}$ we can find two distinct elements, $x^{\mathscr{M}}_{i,1}, x^{\mathscr{M}}_{i,2}$ realizing $R_i(x)$ in $\mathscr{M}$. A function computing the complete formulas for $x^{\mathscr{A}}_{i,1}$ is not in $\mathscr{S}$ since such a function computes X'.

Let $Tr \subseteq 2^{<\omega}$ be the set of σ such that for all $i, s \leq |\sigma|$, $R^{\mathscr{A}}_{i,s}(x^{\mathscr{A}}_{i,1})$ iff $R^{\mathscr{M}}_{i,s}(x^{\mathscr{M}}_{i,\sigma(i)})$. Tr is computable in $\mathscr{A} \oplus \mathscr{M} \oplus X$ and has at least one node at each level s. Therefore in $\mathscr{S}$ there is an $f \in [Tr]$. For such an f, the types of $x^{\mathscr{A}}_{i,1}$ and $x^{\mathscr{M}}_{i,f(i)}$ are the same for each i in $\mathbb{N}$ and hence the map sending $x^{\mathscr{A}}_{i,1}$ to $x^{\mathscr{M}}_{i,f(i)}$ can be computably (in f and $\mathscr{M}$) extended into an embedding. This embedding is also in $\mathscr{S}$.

So $\mathscr{A}$ is effectively prime in $\mathscr{S}$ but not effectively atomic in $\mathscr{S}$. Note that if $\mathscr{B} \in \mathscr{S}$ is also prime then a similar argument shows that there is an isomorphism between $\mathscr{A}$ and $\mathscr{B}$ in $\mathscr{S}$. So $\mathscr{A}$ is not part of a counterexample to effectively $\square$

Corollary 19. *$WKL_0 \wedge \neg\, ACA_0$ implies the negation of "Effectively Prime $\Rightarrow$ Effectively Atomic". So "Effectively Prime $\Rightarrow$ Effectively Atomic" implies $ACA_0 \vee \neg WKL_0$.*

Question 20. *What is the reverse mathematics strength of "Effectively Prime $\Rightarrow$ Effectively Atomic"?*

We know that Prime Uniqueness holds in topped models of RCA_0 by Corollary 17. When the 1-types determine all types, the construction from Myhill's Isomorphism Theorem produces an isomorphism between $\mathscr{A}$ and $\mathscr{B}$ from the two embeddings. However we do not even know whether Prime Uniqueness fails in some Scott Set for more complicated theories.

Question 21. *Does Prime Uniqueness hold in RCA_0? in WKL_0? What is the reverse mathematics strength of Prime Uniqueness?*

References

[1] David R. Belanger. Reverse mathematics of first-order theories with finitely many models. *J. Symb. Log.*, 79(3):955–984, 2014.

[2] C. C. Chang and H. J. Keisler. *Model theory*, volume 73 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, third edition, 1990.

[3] Valentina S. Harizanov. Pure computable model theory. In *Handbook of recursive mathematics, Vol. 1*, volume 138 of *Stud. Logic Found. Math.*, pages 3–114. North-Holland, Amsterdam, 1998.

[4] Denis R. Hirschfeldt, Richard A. Shore, and Theodore A. Slaman. The atomic model theorem and type omitting. *Trans. Amer. Math. Soc.*, 361(11):5805–5837, 2009.

[5] Stephen G. Simpson. *Subsystems of second order arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1999.

Department of Mathematics, University of Notre Dame
E-mail address: cholak@nd.edu

Department of Mathematics, University of Portland
E-mail address: mccoy@up.edu