EFFECTIVE PRIME UNIQUENESS

PETER CHOLAK AND CHARLIE MCCOY

Abstract. Assuming the obvious definitions below we show the 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.

This paper is a draft of some recent work. A polished paper will be forthcoming, but this draft announces the results, gives a detailed proof of the most difficult result, and sketches the others. Comments are very welcome!

1. Introduction

In [2], 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 analyze this open question from the perspective of effective model theory, computability theory, and reverse mathematics.

Throughout the paper, a theory $T$ will always be complete and decidable, and all models will be decidable, although we will often re-state these facts for emphasis. We give some definitions relevant to our analysis.

Definition 1. Let $T$ be a decidable theory and $A$ a decidable model of $T$.

1. The model $A$ is effectively prime, if for every decidable model $M \models T$, there is a computable elementary embedding $f : A \rightarrow M$. Note that $f$ need not be uniformly computable in $A$ and/or $M$.

2. The model $A$ is effectively atomic if there is a computable function $g$ that accepts as an input a tuple $\vec{a}$ from $A$ (of any length) and outputs a complete formula $\varphi(\vec{x})$ so that $A \models \varphi(\vec{a})$. Again $g$ need not be uniformly computable in $A$.

3. The model $A$ is uniformly effectively prime if there is a partial computable function $\Phi$ so that, given $M \models T$, $\Phi(M)$ halts and outputs the code of a computable elementary embedding $f : A \rightarrow M$. Again $\Phi$ need not be uniformly computable in $A$.

Some observations:

1. If two decidable models $A$ and $B$ of the same decidable theory $T$ are both effectively atomic, then the classical back and forth construction produces a computable isomorphism $f : A \cong B$.

2. A modification of the classic proof that atomic implies prime shows that effectively atomic implies uniformly effectively prime.

This work was partially supported by a grant from the Simons Foundation (#315283 to Peter Cholak).
The results in [2] are about decidable, atomic models not effectively atomic models. We show in the next section the following:

**Theorem 1.** Let $T$ be decidable and $\mathcal{A}, \mathcal{B} \models T$ be decidable models. Then either there is a computable isomorphism $h : \mathcal{A} \cong \mathcal{B}$; or there is a decidable $\mathcal{M} \models T$, so that either there is no computable elementary embedding of $\mathcal{A}$ into $\mathcal{M}$, or there is no computable elementary embedding of $\mathcal{B}$ into $\mathcal{M}$.

The statement of this theorem implies that two effectively prime models are computably isomorphic. Effective prime uniqueness is true. In fact, by looking carefully at the construction, we can see there is a Turing functional $\Psi$ such that if $\mathcal{A}$ and $\mathcal{B}$ are effectively prime then there is an $i,j$ such that $\Psi(\mathcal{A}, \mathcal{B}, i, j)$ is an isomorphism (or more correctly the code of an isomorphism) between $\mathcal{A}$ and $\mathcal{B}$. A rough quick analysis suggests that this result holds in RCA$_0$. We also note that the last lemma of the verification involves a complex induction argument with some heavy duty bookkeeping. In the upcoming polished draft we will improve the above theorem to the following:

**Theorem 2.** There a Turing functional $\Phi(\mathcal{A}, e)$ such that if $T$ is decidable and $\mathcal{A} \models T$ is a decidable model then either, for some $e$, $\Phi(\mathcal{A}, e)$ witnesses that $\mathcal{A}$ is effectively atomic or there is a decidable $\mathcal{M} \models T$, such that there is no computable elementary embedding of $\mathcal{A}$ into $\mathcal{M}$.

Theorem 2 implies that effectively prime implies effectively atomic. Thus, by one of the above observation, effectively prime, effectively atomic, and uniformly effectively prime are all equivalent. By one of the observations above, Theorem 2 implies Theorem 1. Therefore in the polished draft we will replace the proof of Theorem 1 with a proof of Theorem 2. The proof has the same feature as the proof of Theorem 1 but should be slightly shorter.

A code for a decidable model $\mathcal{A}$ is a Turing machine that computes the complete diagram of $\mathcal{A}$. So from a decidable model the theory is computable. So $T$ need not be an input into $\Phi$. However the next lemma shows that the input of the $e$ is necessary.

**Lemma 3.** For all $\Phi$, there is a effectively atomic $\mathcal{A}$ such that $\Phi(\mathcal{A})$ does not witness that $\mathcal{A}$ is effectively atomic.

Hence the “obvious” notion of “uniformly effectively atomic” is vacuous. The proof of lemma can be done in the language of infinitely many unary relations and depending on $\Phi$ the resulting model nothing is in any of these relations or exactly one of the relation splits the model into 2 infinite parts. Surely other similar proofs have appeared.

### 2. Proof of Theorem [1]

#### 2.1. Reference and Conventions.

This paper builds on the write-up of the Effective Completeness Theorem given in Harizanov’s survey paper in the Handbook of Recursive Mathematics, [1]. 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 $\mathcal{L}(T)$. Let $\{\sigma_e : e \in \omega\}$ be the computable enumeration of the set of all sentences in the language $\mathcal{L}(T) \cup C$. (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 $(\mathcal{L}(T) \cup C)$-theory $\Gamma \supset T$. This theory will, as usual, have Henkin witnesses, so that the desired model $\mathcal{M}$ 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 $\mathcal{M}$ to the language of $\mathcal{L}(T)$.

We computably enumerate $\Gamma$ as $\{\delta_0, \delta_1, \ldots\}$, where we enumerate $\delta_s$ at some point during stage $s$ of the construction. We denote $\delta_0 \land \ldots \land \delta_s$ 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 $\delta_s$ into $\Gamma$, we have to do more than ensure that $\Gamma$ 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 $\Phi$, we try to diagonalize against $\Phi$ being an elementary embedding of $A$ into $\mathcal{M}$; and, for each computable function $\Psi$, we try to diagonalize against $\Psi$ being an elementary embedding of $B$ into $\mathcal{M}$. If we can succeed for all $\Phi$, or if we can succeed for all $\Psi$, we will have proven the theorem. To this end, we fix, as is standard, a computable enumeration of all pairs of computable functions $(\Phi, \Psi)$. Second, for each pair $(\Phi, \Psi)$, as it looks as though we are failing at all attempts to diagonalize against either function in the pair, we computably construct, in stagewise fashion, what we hope will be an isomorphism $h_{\Phi, \Psi} : A \cong B$. When there is no ambiguity, we will drop the $\Phi, \Psi$ subscript on $h$.

Just as in Harizanov’s proof of the Effective Completeness Theorem, the model $\mathcal{M}$ is really not defined until after the stagewise construction is complete, when we can define the equivalence classes according to the set $\Gamma$. (However, as in that proof, $\mathcal{M}$ 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 functions $\Phi$ and $\Psi$ as having range not in the universe of $\mathcal{M}$, precisely, 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 $\Gamma$, there is a uniformly effective way of converting in either direction between a function $\Phi : A \to C$ and a function $\Phi' : A \to \mathcal{M}$. (Given $\Phi$, and $a \in A$, we define $\Phi'(a) := [\Phi(a)]$. Given $\Phi'$, and $a \in A$, we search, using $\Gamma$, 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 $\Phi'$ as the obvious effective translation of $\Phi$. (Analogously for $\Psi$, $\Psi'$, and $B$.)

Recall that the standard enumeration of Turing computations of the form $\Phi_n(a) \downarrow = c$ is such that $a, c < s$. In our enumeration of the sentences in $\Gamma$, 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 $\sigma_e$ is such that all of the constants which appear in $\sigma_e$ are among $c_0, \ldots, c_s$. Because of how and when we decide to enumerate sentences or their negations into $\Gamma$, 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 \( \Phi_s \) or \( \Psi_s \) and which are not. For instance, \( c_1 \) may be a constant appearing in the formula \( \varphi \), a fact we denote by writing \( \varphi(c_1) \). If the variable \( x_1 \) does not appear in \( \varphi \), and we form the new formula by replacing every appearance of \( c_1 \) in \( \varphi \) with \( x_1 \), we will simply write \( \varphi(x_1) \) for this new formula. Similarly, if \( \vec{a} = \text{dom}(\Phi_s) \), and we break up the tuple \( \vec{c}_s \) into the sub-tuples \( \vec{c}_s - \Phi_s(\vec{a}), \Phi_s(\vec{a}) \), 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 \( \sigma_e \) are among \( \vec{c}_e \), and NOT to signify that all of these constants do, in fact, appear in \( \sigma_e \).

2.2. A requirement \( R_{\Phi,\Psi} \) requiring attention. For each Turing function pair \((\Phi,\Psi)\), we have the requirement \( R_{\Phi,\Psi} \):

\[
\neg(\Phi' : \mathcal{A} \prec \mathcal{M}); \text{ OR } \neg(\Psi' : \mathcal{B} \prec \mathcal{M}); \text{ OR } \text{there is a computable isomorphism } h : \mathcal{A} \cong \mathcal{B}.
\]

For each requirement \( R_{\Phi,\Psi} \), we refer to the index of the requirement as the number \( \langle \Phi,\Psi \rangle \), which is based on indices for the Turing functions and the standard pairing function. As usual, one requirement is higher priority than another if its index is lower.

**Definition 2.** A requirement of the form \( R_{\Phi,\Psi} \) is completely satisfied by stage \( s \) if AT LEAST ONE of the following three conditions holds:

1. At least one of \( \Phi_s \) or \( \Psi_s \) is not 1-1; OR
2. If \( \vec{a} = \text{dom}(\Phi_s) \), 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 \( \mathcal{A} \not\cong \exists y \theta_s(\vec{y}, \vec{a}) \); OR
3. The analogue of the previous item with \( \vec{b}, \vec{B}, \) and \( \Psi_s, \Phi_s, \vec{c}_s \) in place of \( \vec{a}, \mathcal{A} \) and \( \Phi_s \).

(Note: the substitution of \( \vec{a} \) for \( \Phi_s(\vec{a}) \) or \( \vec{b} \) for \( \Psi_s(\vec{b}) \) is unambiguous, because, if the first condition does not hold, then \( \Phi_s, \Psi_s \) are assumed to be 1-1.)

**Definition 3.** The stage \( s \) approximation to \( h_{\Phi,\Psi} \) is denoted by \( h_{\Phi,\Psi,s} \) (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 \( \emptyset \). (Even at this same stage \( s \), some of the \( h_{s-1} \) that had been initialized might be re-defined, so that some of these \( h_s \) are non-empty by the end of stage \( s \).)

**Terminology:** Consider some specific \( h \) associated with a requirement. Let \( s \) be a stage at which \( h_s \) is being properly extended. If \( h_{s-1} = \emptyset \) or if \( h_{s-1} \) has been initialized at an earlier part of this same stage \( s \), then \( s \) is a “forth” stage for this \( h \). Assume, on the other hand, that \( h_{s-1} \neq \emptyset \) and \( h_{s-1} \) is not initialized at this same stage \( s \); and \( t \) was the previous stage at which \( h_t \) was properly extended. If \( t \) was a “forth” stage, then \( s \) is a “back” stage; if \( t \) was a “back” stage, then \( t \) is a “forth” stage.

**Definition 4.** A requirement of the form \( R_{\Phi,\Psi} \) requires attention at stage \( s \) if

1. \( R_{\Phi,\Psi} \) is not completely satisfied by stage \( s \);
2. \( \Phi_s \) and \( \Psi_s \) both have converged on at least one input, and one of the following is true:
• \( h_{\Phi,\Psi,s-1} = \emptyset \) or has been initialized at this stage \( s \), and \( \Phi_s(a_0) \downarrow \); OR
• \( h_{\Phi,\Psi,s-1} \neq \emptyset \) and has not been initialized at this stage \( s \); and the previous stage \( t \) for which \( h_{\Phi,\Psi} \) extended \( h_{\Phi,\Psi,t-1} \) or \( \emptyset \) was a forth stage; and the domain of \( \Psi_s \) contains an initial segment of the universe of \( B \) that includes \( \text{ran}(h_{s-1}) \) and at least one more element; OR
• \( h_{\Phi,\Psi,s-1} \neq \emptyset \) and has not been initialized at this stage \( s \); and the previous stage \( t \) for which \( h_{\Phi,\Psi} \) extended \( h_{\Phi,\Psi,t-1} \) was a back stage; and the domain of \( \Phi_s \) contains an initial segment of the universe of \( A \) that includes \( \text{dom}(h_{s-1}) \) and at least one more element.

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

Stage \( s = 2k + 1 \) for \( k \in \omega \) (Henkin witness requirement):
If \( \delta_k = \exists x \gamma(x) \), then, by convention, we know that the first element of \( C \) that does not appear in \( \theta_{s-1}(c_{s-1}) \) is \( c_s \). Define \( \delta_s = \gamma(c_s) \). Otherwise, \( \delta_s := (c_s = c_s) \).

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

Let \( e \) be the least \( e \) for which we have not explicitly decided whether to add \( \sigma_e \) or \( \neg \sigma_e \) to \( \Gamma \); i.e., at no previous stage \( t \) did \( \delta_t := \sigma_t \wedge (c_t = c_t) \) or \( \delta_t := \neg \sigma_t \wedge (c_t = c_t) \). We will work to make this determination at this stage, unless the complete satisfaction of a higher priority requirement \( R_{\Phi,\Psi} \) 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 \( \bar{x} \) is a tuple of new variables (not appearing among the variables in \( \theta_{s-1}(c_{s-1}) \)) of the same length as \( c_{s-1} \), then \( T \vdash \forall \bar{x}[\theta_{s-1}(\bar{x}) \rightarrow \gamma(\bar{x})] \).
3. If it is true for either \( \gamma = \sigma^* \) or for \( \gamma = \neg \sigma^* \), then only this \( \gamma \) is consistent with \( T \) and \( \theta_{s-1}(c_{s-1}) \). Define \( \delta_s := \gamma \wedge (c_s = c_s) \) and exit the algorithm. Otherwise, then each of \( \sigma^* \) and \( \neg \sigma^* \) is consistent with \( T \) and \( \theta_{s-1}(c_{s-1}) \), so proceed to the next step.
4. Determine if there is any requirement \( R_{\Phi,\Psi} \) 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 \( \Phi \) or \( \Psi \) is not 1-1.)
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 pair \( \Phi, \Psi \) associated with a requirement that has not been completely satisfied and has index \( \leq i^* \), complete the following analysis:
   • Let \( \bar{a} = \text{dom}(\Phi_s) \) and \( \bar{b} = \text{dom}(\Psi_s) \). (Recall that, by the conventions we mentioned above, \( \text{ran}(\Phi_s) \subseteq c_{s-1} \), \( \text{ran}(\Psi_s) \subseteq c_{s-1} \), and all of the constants appearing in \( \sigma_e \) are among \( c_{s-1} \), as well.)
   • Determine if one of the following conditions hold:
     (a) For \( \gamma = \sigma^* \) or for \( \gamma = \neg \sigma^* \), if we look at \( \theta_{s-1}(c_{s-1}) \wedge \gamma \) as \( \rho(c_s - \Phi_s(\bar{a}), \Phi_s(\bar{a})) \), and if \( \bar{y} \) is a tuple of new variables (not
appearing among the variables in $\theta_{s-1}(\vec{c}_{s-1}) \land \gamma$) of the same length as $\vec{c}_{s-1} - \Phi_s(\vec{a})$, then $A \not\models \exists y \psi(\vec{y}, \vec{a})$.

(\textit{COMMENT}: Each of the sentences $\sigma^* \land \neg \sigma^*$ is consistent with $T$ and $\theta_{s-1}(\vec{c}_{s-1})$, but one of them would make it impossible for $\Phi$ to be an elementary embedding.)

(b) The previous condition does not hold, but it does hold is if we replace $\Psi_s$ for $\Phi_s$, $\vec{b}$ for $\vec{a}$, and $B$ for $A$.

(\textit{COMMENT}: Each of the sentences $\sigma^* \land \neg \sigma^*$ is consistent with $T$ and $\theta_{s-1}(\vec{c}_{s-1})$, but one of them would make it impossible for $\Psi$ to be an elementary embedding.)

(c) The previous two conditions do not hold, but for $\gamma = \sigma^*$ or for $\gamma = \neg \sigma^*$, if

- we look at $\theta_{s-1} \land \gamma$ as $\theta_{s-1}(\vec{c}_{s-1} - \Phi_s(\vec{a}), \Phi_s(\vec{a})) \land \gamma(\vec{c}_{s-1} - \Phi_s(\vec{a}));$
- $\vec{x}$ 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 $T \vdash \exists \vec{x} \exists \vec{y} \theta_{s-1}(\vec{y}, \vec{x}) \land \forall \vec{y} \theta_{s-1}(\vec{y}, \vec{x}) \rightarrow \gamma(\vec{y}, \vec{x})]$.  

(\textit{COMMENT}: Since the first condition doesn’t hold, we know that each of $\sigma^*$ 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 $\theta_{s-1}$, but which can accommodate only one of $\sigma^*$ or $\neg \sigma^*$. Therefore, defining the sentence $\delta_s$ to express this reality about $\Phi(\vec{a})$ would make it impossible for $\Phi$ to be an elementary embedding.)

(d) The previous three conditions do not hold, but the last condition is true if we replace $\Psi_s$ for $\Phi_s$, $\vec{b}$ for $\vec{a}$, and $B$ for $A$.

(\textit{COMMENT}: Since the second condition doesn’t hold, we know that each of $\sigma^*$ and $\neg \sigma^*$ is consistent with $\vec{b} \mapsto \Psi(\vec{b})$ 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 $\theta_{s-1}$, but which can accommodate only one of $\sigma^*$ or $\neg \sigma^*$. Therefore, defining the sentence $\delta_s$ to express this reality about $\Psi(\vec{b})$ would make it impossible for $\Psi$ to be an elementary embedding.)

(7) If all of the candidate pairs $\Phi, \Psi$ that are considered don’t satisfy any of the above conditions, then define $\delta_s := \sigma^* \land (c_s = c_s)$, 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 the above conditions. In the rest of the steps, $\Phi$ and $\Psi$ refer specifically to the pair of Turing functions for this requirement.

(9) If the pair satisfied the first or second condition, then, for the appropriate $\gamma$ that makes the condition satisfied (either $\sigma^*$ or $\neg \sigma^*$, and there is no ambiguity which); define $\delta_s := \gamma \land (c_s = c_s)$; and exit the algorithm. Otherwise, proceed to the next step.
(10) If the pair satisfied the third condition, 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 $\gamma$ that makes the condition satisfied is unambiguous. Now, for this $\gamma$, REDEFINE $\sigma^* := \gamma \land \forall \bar{y}(\theta_{s-1}(\bar{y}, \Phi_s(\bar{a})) \rightarrow \gamma(\bar{y}, \Phi_s(\bar{a}))$. And, with this new index $i^*$ and this new $\sigma^*$, return to the second step of the algorithm. Otherwise, proceed to the next step.

(COMMENT: Why redefine $\sigma^*$ instead of just defining $\delta_s$ to be the conjunction of this new $\sigma^*$ and $(c_s = c_s)$? If $\delta_s$ were defined in this way, then the respective requirement would be satisfied; however, because this $\delta_s$ was not analyzed in the earlier steps of the algorithm, it is possible that, in adding this $\delta_s$, 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 $\sigma^*$ and restart the algorithm.)

(11) Since we’ve reached this step, the pair satisfied the fourth condition. Again, 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 $\gamma$ that makes the condition satisfied is unambiguous. Now, for this $\gamma$, REDEFINE $\sigma^* := \gamma \land \forall \bar{y}(\theta_{s-1}(\bar{y}, \Psi_s(\bar{b})) \rightarrow \gamma(\bar{y}, \Psi_s(\bar{b}))$. And, with this new index $i^*$ and this new $\sigma^*$, return to the second step of the algorithm.

(COMMENT: Same comment as above.)

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 $\delta_s$ is well-defined.

2.3.2. Definition/Construction of the stage $s$ approximations to the potential isomorphisms. If $R_{\Phi, \Psi}$ is the highest priority requirement (with $< \Phi, \Psi >$ less than $e$) 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 $R_{\Phi, \Psi}$ with $< \Phi, \Psi >$ greater than or equal to $e$.

As the final part of the construction at positive even stages, we define $h_{\Phi, \Psi, s}$ on the requirements $R_{\Phi, \Psi}$ 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 $R_{\Phi, \Psi}$ that still requires attention at stage $s$, which, by definition, is a finite number of requirements.)

Assume we are in scenario 1 or 3 in the last part of the Definition 3 (requiring attention). We will define $h_s$ on the domain of $\Phi_s$, which – by the assumption that $R_{\Phi, \Psi}$ requires attention at this stage – contains either an initial segment of the universe of $A$ that includes $\text{dom}(h_{s-1})$ and at least one more element; or contains at least $a_0$, if $h_{s-1}$ had been initialized so far at stage $s$. Let $\bar{a} = \text{dom}(\Phi_s)$ and $\bar{a}' = \text{dom}(h_{s-1})$ or $\bar{a}' = \emptyset$, if $h_{s-1}$ had been initialized so far at this stage $s$. Again, note that by the definition of requiring attention and the assumption of which scenario we’re in, $\bar{a}' \subset \bar{a}$.

Recall that we are automatically conceiving of $\Phi_s(\bar{a})$ as being constants from $C$ and among $\bar{c}_s$. Consider the sentence $\theta_s(\bar{c}_s)$. We look at $\theta_s(\bar{c}_s)$ as $\theta_s(\bar{c}_s - \Phi_s(\bar{a}), \Phi_s(\bar{a} - \bar{a}'))$. Let $\bar{y}, \bar{x}, \bar{z}$ be three new, disjoint tuples of variables (not appearing among the variables of $\theta_s$) of the same length as $\bar{c}_s \Phi_s(\bar{a}), \Phi_s(\bar{a} - \bar{a}')$, and , respectively. Consider the formula $\phi(\bar{x}, \bar{z}) := \exists \bar{y} \theta_s(\bar{y}, \bar{x}, \bar{z})$.
Use the decidability of $B$ to determine whether there exists a tuple of distinct elements $\vec b \in B$ with the following properties:

- $\vec b \cap \text{ran}(h_{s-1}) = \emptyset$;
- $\vec b$ has the same length as $\vec a - \vec a'$;
- $B \models \phi(h_{s-1}(\vec a'), \vec b)$.

If so, then define $h_s$ by $h_s(\vec a') := h_{s-1}(\vec a')$ and $h_s(\vec a - \vec a') := \vec b$, where $\vec b$ is the first such. If not, then just let $h_s = h_{s-1}$. (The key thing that the Verification subsection below must establish is that either $h_s$ will always properly extend $h_{s-1}$, or at some stage the requirement – remember this $h$ is attached to a specific $R_{\Phi, \Psi}$ – will stop requiring attention.)

If we are in scenario 2, the definition is analogous, with $B, \Psi_s, \text{ran}(h_{s-1}), \text{dom}(h_{s-1}), \vec b = \text{dom}(\Psi_s), \vec b = \text{ran}(h_{s-1})$, and $h_{s-1}^{-1}$ replacing $A, \Phi_s, \text{dom}(h_{s-1}), \text{ran}(h_{s-1}), \vec a = \text{dom}(\Phi_s), \vec a' = \text{dom}(h_{s-1})$, and $h_{s-1}$, respectively.

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

This concludes the construction.

2.4. Verification.

**Lemma 4.** $M$ is decidable and $M \models T$.

**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 $\Gamma$ in the expanded language by eventually adding $\sigma_e$ or $\neg \sigma_e$ (with a trivial conjunct of the form $(c_s = c_s)$ appended) to $\Gamma$. It is true that, even if $\sigma_e$ is the original sentence considered at a particular even stage $s$, the above algorithm, because of conditions c) and d), might redefine $\delta_s$ to be a sentence that implies neither $\sigma_e$ nor $\neg \sigma_e$. Now, without any such delays, the sentence $\sigma_e$ would be decided by stage $2(e + 2)$ 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 $\sigma_e$ or $\neg \sigma_e$ (with a trivial conjunct appended) is included in $\Gamma$.

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 $h_{\Phi', \Psi', s}$ is an additional component of our construction, but we note two important things. First, because of the decidability of $A$ and $B$, there is no infinite search in the construction of the $h_{\Phi, \Psi, s}$. Second, this part of the construction does not affect choices in how we build $M$ and the complete theory $\Gamma$.

**Lemma 5.** If every requirement $R_{\Phi, \Psi}$ requires attention only finitely often, then either $A \not\models M$ or $B \not\models M$.

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

- the corresponding $\Phi' : A \rightarrow M$ or the corresponding $\Psi' : A \rightarrow M$ is not 1-1; OR
- for some tuple $\vec{a} \in A$ and some formula $\varphi(\vec{x})$, $A \models \varphi(\vec{a})$ and $M \models \neg \varphi(\Phi'(\vec{a}))$; or
- for some tuple $\vec{b} \in B$ and some formula $\varphi(\vec{x})$, $B \models \varphi(\vec{b})$ and $M \models \neg \varphi(\Psi'(\vec{b}))$.

(See the above section on conventions on the connection between $\Phi, \Psi$ and $\Phi', \Psi'$.)

Second, because either $\Phi$ or $\Psi$ is not total, and hence either $\Phi'$ or $\Psi'$ is not total.

Now, as the section on conventions explained, every computable function $f : A \rightarrow M$ is equal to $\Phi'$ for some $\Phi : A \rightarrow C$; and every computable function $g : B \rightarrow M$ is equal to $\Psi'$ for some $\Psi : B \rightarrow C$. Therefore, if every requirement $R_{\Phi, \Psi}$ stops requiring attention by some stage $s$, then either every computable function from $A$ to $M$ fails to be an elementary embedding; or every every computable function from $B$ to $M$ fails to be an elementary embedding.

\[ \square \]

Therefore, for the rest of this verification, we assume that there is a requirement $R_{\Phi, \Psi}$ and stages $s^* \leq s$ with the following three properties:

- $R_{\Phi, \Psi}$ requires attention infinitely often.
- $s^*$ is the least stage $t$ with the following property: for each stage $u > t$, it is NOT the case that a requirement $R_{\Phi', \Psi'}$ of priority higher than that of $R_{\Phi, \Psi}$ first becomes completely satisfied at $u$.
- $s$ is the first stage $\geq s^*$ so that $R_{\Phi, \Psi}$ requires attention at $s$.

With this requirement $R_{\Phi, \Psi}$ and these stages $s^*$ and $s$ fixed, we must prove that $h_{\Phi, \Psi}$ is an isomorphism from $A$ to $B$. We will simply refer to this function as $h$ from now on, and its stage $t$ approximation as $h_t$. The following long lemma will essentially complete this proof. Recall the terminology about “back” and “forth” stages from subsection 1.2. Recall the notation from subsection 1.1 that $\theta_t$ is the conjunction of all sentences of $\Gamma$ enumerated by the end of stage $t$.

**Lemma 6.** For each stage $t \geq s$ for which $R_{\Phi, \Psi}$ requires attention, the following is true:

1. If $t = s$, then $h_t$ properly extends $\emptyset$; if $t > s$, then $h_{t-1}$ is not empty, and $h_t$ properly extends $h_{t-1}$. The approximation $h_t$ is 1-1.
2. If $t$ is a forth stage, then $\text{dom}(h_t) = \text{dom}(\Phi_t)$; if $t$ is a back stage, then $\text{ran}(h_t) = \text{dom}(\Psi_t)$.
3. Let $t$ be a forth stage, and $\vec{a} = \text{dom}(\Phi_t)$, and $h_t(\vec{a}) = \vec{b}$. Then for all $u \geq t$, if $\theta_u(c_u) = \theta_u(c_u - \Phi_t(\vec{a}), \Phi_t(\vec{a}))$, then $A \models \exists \vec{y}_u \theta_u(\vec{y}_u, \vec{a})$ and $B \models \exists \vec{y}_u \theta_u(\vec{y}_u, \vec{b})$. In particular, for any proposition $\rho(\vec{x})$ in the original language, with $\vec{x}$ of the same length as $\vec{b}$ and $\vec{a}$, $A \models \rho(\vec{a})$ iff $B \models \rho(\vec{b})$.
4. Let $t$ be a back stage, and $\vec{b} = \text{dom}(\Psi_t)$, and $h_t(\vec{a}) = \vec{b}$. Then for all $u \geq t$, if $\theta_u(c_u) = \theta_u(c_u - \Psi_t(\vec{b}), \Psi_t(\vec{b}))$, then $A \models \exists \vec{y}_u \theta_u(\vec{y}_u, \vec{a})$. In particular, for any proposition $\rho(\vec{x})$ in the original language, with $\vec{x}$ of the same length as $\vec{b}$ and $\vec{a}$, $A \models \rho(\vec{a})$ iff $B \models \rho(\vec{b})$.

**Proof.** Our proof is by induction. We begin at stage $s$. By the assumptions about $s$ and $s^*$, and by the details of initialization, either $s = 0$, or $h_{s-1} = \emptyset$ or $h_{s-1}$ is
initialized at a point of stage $s$. Therefore, since $R_{\Phi, \Psi}$ requires attention at $s$, the construction of $h_s$ simply begins with $\emptyset$. We use the notation of our construction of $h_s$ (see the previous section), so that $\tilde{a} = \text{dom}(\Phi_s)$ and, in this case, $\tilde{a}'$ must be $\emptyset$. The sentence $\theta_s(c_s) = \theta_s(c_s - \Phi_s(\tilde{a}_s), \Phi_s(\tilde{a}))$. We then consider the formulas $\theta_s(\tilde{y}, \tilde{x})$ and $\phi(\tilde{x}) := \exists \tilde{y} \theta_s(\tilde{y}, \tilde{x})$. Now, it must be the case that $A \models \exists \tilde{y} \theta_s(\tilde{y}, \tilde{a})$; otherwise we would have completely satisfied $R_{\Phi, \Psi}$ by this stage, and then $R_{\Phi, \Psi}$ would not receive attention infinitely often. Consequently, $T \models \exists \tilde{y} \theta_s(\tilde{y}, \tilde{x})$. Moreover, since the elements of $\tilde{a}$ are all distinct, if we add a clause $\tau(\tilde{x})$ simply stating that the elements of $\tilde{x}$ are distinct, then $T \models \exists \tilde{y} \tau(\tilde{y}, \tilde{x})$. And so, there is $\tilde{b}$ of distinct elements in $B$ of the same length as $\tilde{a}$ so that $B \models \phi(\tilde{b})$. By construction, we define $h_s(\tilde{a}) = \tilde{b}$. Thus, 1. and 2. are true for $s$.

For each $u \geq s$, if we look at $\theta_u$ as $\theta_u(c_u - \Phi_u(\tilde{a}))$, and consider the formula $\theta_u(\tilde{y}_u, \tilde{x})$ obtained as usual, then $A \models \exists \tilde{y}_u \theta_u(\tilde{y}_u, \tilde{a})$; otherwise, as noted above, we would have completely satisfied $R_{\Phi, \Psi}$ by stage $u$, and then $R_{\Phi, \Psi}$ would not receive attention infinitely often. Statement 3. of the lemma, in regard to this stage $s$, simply makes the analogous claim about $B$ and $\tilde{b}$.

The proof of Statement 3. is by induction on $u \geq s$. The claim is automatically true for $s$ by the choice of $\tilde{b}$ above. We assume that the statement is true for $u$; i.e., $B \models \exists \tilde{y}_u \theta_u(\tilde{y}_u, \tilde{b})$. We must show that it’s true for $u+1$. In order to make this argument, we have to recall how the construction forms $\theta_{u+1}$ from $\theta_u$ by adding $\delta_{u+1}$ as a conjunct.

If $u+1 = 2k+1$, and $\delta_k$ is an existential sentence, then the sentence $\delta_{u+1}$ added simply ensures there exists a “Henkin witness.” Since the Henkin witness chosen, $c$, is larger than any constant appearing so far, $c$ cannot actually be any of the constants among $c_u$, nor can there be any clauses in $\theta_{u+1}$ that explicitly express it to be equal or unequal to any of the constants among $c_u$. Moreover, the existential statement $\delta_k$ already is part of the the sentence $\theta_u$ and $B \models \exists \tilde{y}_u \theta_u(\tilde{y}_u, \tilde{b})$, by induction hypothesis. Consequently, after the addition of $\delta_{u+1}$ to $\theta_u$ to form $\theta_{u+1}$, the formation of $\exists \tilde{y}_{u+1} \theta_{u+1}(\tilde{y}_{u+1}, \tilde{b})$ essentially just transforms the clause $\delta_{u+1}$ – except for some unimportant additions and/or changes of variables – back into an existential statement that was already part of $\exists \tilde{y}_u \theta_u(\tilde{y}_u, \tilde{b})$, which, again, $B$ satisfies by induction hypothesis. And so, $B \models \exists \tilde{y}_{u+1} \theta_{u+1}(\tilde{y}_{u+1}, \tilde{b})$.

If $u+1 = 2k+2$, then the construction adds $\delta_{u+1}$ beginning with a consideration of deciding some $\sigma$, and ultimately $\delta_{u+1}$ is equal to $\pm \sigma^* \land (c_{u+1} = c_{u+1})$ for $\sigma^*$ relative to the last iteration of the algorithm run at stage $u$. For the rest of this proof of Statement 3. for stage $s$, we will refer to the non-trivial part of $\delta_{u+1}$ as $\gamma$ (i.e., $\gamma = \sigma^*$ or $\gamma = -\sigma^*$).

Since $u \geq s$, $\Phi_s(\tilde{a})$ must be among $c_u$, all the constants appearing among those of $\sigma$, must be among $c_u$, and again, by induction hypothesis, $B \models \exists \tilde{y}_u \theta_u(\tilde{y}_u, \tilde{b})$. Let $\tilde{b}'$ be a tuple of elements that witnesses $B \models \theta_u(\tilde{b}', \tilde{b})$.

If, in the final iteration of the main algorithm, the exit takes place at Step 2, then, since $B \models \theta_u(\tilde{b}', \tilde{b})$, it must be the case that $B \models \gamma(\tilde{b}', \tilde{b})$. And so, $B \models \exists \tilde{y}_{u+1} \theta_{u+1}(\tilde{y}_{u+1}, \tilde{b})$. Therefore, for the rest of this proof of Statement 3. for stage $s$, we assume that for the final iteration of the algorithm, the exit does not take place at Step 2.

Since $u \geq s$, all requirements of priority higher than that of $R_{\Phi, \Psi}$ that will ever be completely satisfied already have done so. Therefore, since $R_{\Phi, \Psi}$ requires
attention infinitely often, the pair \( \Phi, \Psi \) couldn’t have met condition a) or condition b) with either \( \gamma \) or \( \neg \gamma \) in Step 6.

We now will focus on condition c); this will be the key to completing the proof of Statement 3. for stage \( s \). First, we must understand how the tuples \( \vec{b}^*, \vec{b} \in B \) are “split up” in a way relevant to the condition c) in Step 6. Note that it is possible that \( \vec{b}^* \) is not a tuple of distinct elements, and it is possible that \( \vec{b} \) and \( \vec{b}^* \) are not disjoint tuples. Let \( \vec{a}'' = \text{dom}(\Phi_{u+1}) \). (Recall, by our conventions, that we’re guaranteed \( \Phi_{u+1}(\vec{a}'') \subseteq \vec{c}_u \).) In \( \theta_u(\vec{b}^*, \vec{b}) \), the tuple \( \vec{b} \) replaces some of the elements of \( \Phi_{u+1}(\vec{a}'') \), namely, those corresponding to \( \Phi_{u+1}(\vec{a}) \). Part of the tuple \( \vec{b}^* \) replaces \( \Phi_{u+1}(\vec{a}'') - \Phi_{u+1}(\vec{a}) \); call this \( \vec{b}_i^* \). Form the tuple, possibly with repetition of elements, \( \vec{b}, \vec{b}_1^* \), arranged in the proper order so that it corresponds to \( \Phi_{u+1}(\vec{a}'') \), again with \( \vec{b} \) in the same role as \( \Phi_{u+1}(\vec{a}) \); call this tuple \( \vec{b}_y \), since it (with possible repetition) is the same length as the tuple of variables \( \vec{x} \) constructed in Step 6 for the sake of the analysis of condition c). Part of the tuple \( \vec{b}^* \) replaces \( \vec{c}_u - \Phi_{u+1}(\vec{a}'') \); call this tuple \( \vec{b}_y \). Again, it may be the case that \( \vec{b}_y \) is not disjoint from \( \vec{b}_x \).

During the final iteration of the algorithm, condition c) was checked with \( \sigma^* \) and \( \neg \sigma^* \). Of course, since \( B \models \theta_u(\vec{b}^*, \vec{b}) \), it simply requires “re-grouping” (but NOT a re-arranging in any significant way) of variables to see that \( B \models \theta_u(\vec{b}_y, \vec{b}_x) \).

Now assume, for a contradiction, that \( B \not\models \exists \vec{y}\theta_u(\vec{y}, \vec{b}_x) \land \gamma(\vec{y}, \vec{b}_x) \). Therefore, \( B \models \exists \vec{y}\theta_u(\vec{y}, \vec{b}_x) \land \forall \vec{y}\theta_u(\vec{y}, \vec{b}_x) \rightarrow \neg \gamma(\vec{y}, \vec{b}_x) \). Therefore, \( R_{\Phi, \Psi} \) would have satisfied condition c) with \( \neg \gamma \). Now, what could have prevented us from re-defining \( \sigma^* \) accordingly, redefining \( i^* \) to be the index of \( R_{\Phi, \Psi} \), and running another iteration of this algorithm? Only if a requirement of higher priority satisfies one of the conditions a) - d). But if that is the case, then, eventually, a requirement of higher priority must become completely satisfied at this stage, but that cannot be, since \( u \geq s \). And so, after one more iteration of the algorithm is run, \( \delta_{u+1} \) would be defined to be the conjunction of this new \( \sigma^* \) and \( (c_u = c_u) \), unless this iteration revealed that a requirement of higher priority would become completely satisfied; but again, that cannot be, since \( u \geq s \). However, this definition of \( \delta_{u+1} \) would mean the the requirement \( R_{\Phi, \Psi} \) itself becomes completely satisfied at stage \( u \); however, then \( R_{\Phi, \Psi} \) would not require attention infinitely often, a contradiction.

Thus, \( B \models \exists \vec{y}\theta_u(\vec{y}, \vec{b}_x) \land \gamma(\vec{y}, \vec{b}_x) \). Therefore, since \( \theta_{u+1} = \theta_u \land \delta_{u+1} \), and since \( \delta_{u+1} = \gamma \land (c_{u+1} = c_u) \), \( B \models \exists \vec{y}\exists z(\theta_{u+1}(\vec{y}, \vec{z}, \vec{b}_x)) \). Finally, let \( \vec{b}_y^* \) in \( B \) witness that \( B \models \theta_{u+1}(\vec{b}_y^*, \vec{b}_x) \). (Again, there could be non-trivial intersection between \( \vec{b}_y^* \) and \( \vec{b}_y \).) We re-separate \( \vec{b}_x \) back into \( \vec{b} \) and \( \vec{b}_1^* \); if some element \( b \) appears in both \( \vec{b} \) and \( \vec{b}_1^* \) or \( \vec{b}_y^* \), we use the constants of \( \delta \) appearing \( \theta_{u+1} \) and the mapping \( \Phi(\vec{a}) \) among those constants to determine whether to regard it as an element of \( \vec{b} \) or \( \vec{b}_1^* \) or \( \vec{b}_y^* \); and we replace all of the elements regarded as part of \( \vec{b}_1^* \) or \( \vec{b}_y^* \) by variables in such a way as to match \( \theta_{u+1}(\vec{b}_{u+1}, \vec{b}) \). Therefore, \( B \models \exists \vec{y}\theta_{u+1}(\vec{y}_{u+1}, \vec{b}) \).

Finally, since the construction includes all sentences in the expanded language among the \( \sigma_c \), we can conclude that for any proposition \( \rho(\vec{x}) \) in the original language \( \mathcal{L} \), with \( \vec{x} \) of the same length as \( \vec{b} \) and \( \vec{d} \), \( A \models \rho(\vec{a}) \) iff \( B \models \rho(\vec{b}) \). This concludes the proof of Statements 1. - 3. for the base stage, stage \( s \).

Assume \( t > s \), \( R_{\Phi, \Psi} \) requires attention at stage \( t \), and Statements 1.-4. are true of all stages \( w \) with \( s \leq w < t \) at which \( R_{\Phi, \Psi} \) requires attention. We must
prove Statements 1.-4. about t. First, we consider the case where t is a “back stage,” so that the previous stage at which $R_{\Phi, \Psi}$ required attention, call it $t'$, was a “forth stage.” By induction hypothesis, $h_{t'}$ is not empty. Furthermore, by construction, $h_w$ is never initialized at any $w \geq s$, and $h_{t-1} = h_{t'}$, since $R_{\Phi, \Psi}$ does not require attention between $t'$ and $t$. Also, by definition of requiring attention, $\text{dom}(\Psi_t) = b$ contains an initial segment of the universe of $B$ that includes the range of $h_{t'}$ and at least one more element; let $\text{ran}(h_{t'}) = \vec{b}'$ and $h_{t'}(\vec{a}') = \vec{b}'$. Consider $\theta(c_t - \Psi_t(b), \Psi_t(\vec{b}'), \Psi_t(\vec{b} - \vec{b}'))$. We know that $B \models \exists \vec{y} \exists \vec{z}[\theta(c_t, \vec{y}, \vec{z}) \land \tau(\vec{y}, \vec{z})]$; otherwise, $R_{\Phi, \Psi}$ would be completely satisfied by this stage, and $R_{\Phi, \Psi}$ would not receive attention infinitely often. Moreover, if we add the clause $\tau(\vec{b}', \vec{z})$ that says the elements of $\vec{b}'$ are distinct, the elements of $\vec{z}$ are distinct, and the elements of $\vec{b}'$ and $\vec{z}$ are disjoint, then $B \models \exists \vec{y} \exists \vec{z} \theta(c_t, \vec{y}, \vec{z}) \land \tau(\vec{y}, \vec{z})$. Now, by induction hypothesis, Statement 4. is true about $t'$. Therefore, if we let $\rho(\vec{x})$ be the formula $\exists \vec{y} \exists \vec{z} \theta(c_t, \vec{y}, \vec{z}) \land \tau(\vec{y}, \vec{z})$, then $A \models \rho(\vec{a})$ iff $B \models \rho(\vec{b})$, so $A \models \rho(\vec{a'})$.

Identify the first tuple $\vec{a''}$ in $A$ that witnesses $A \models \exists \vec{y} \exists \vec{z} \theta(c_t, \vec{a'}, \vec{a''}) \land \tau(\vec{a'}, \vec{a''})$. Because the clause $\tau(\vec{x}, \vec{z})$ is included, and because $\Psi_t$ is 1-1 (otherwise, $R_{\Phi, \Psi}$ would stop receiving attention), we know that the tuple $\vec{a'}, \vec{a''}$ has no repetitions and has the same length as $\vec{b}$. And so, the construction defines $h_t(\vec{a'}, \vec{a''}) = \vec{b}', b - \vec{b}'$. Let $\vec{a} := \vec{a'}, \vec{a''}$. This establishes Statement 1.-2. about stage $t$.

We must prove Statement 4., since $t$ is a “back stage.” This proof, however, should proceed in an exactly analogous fashion to the base case. For this part of the argument, there is no extra wrinkle to the argument because of the fact that $t > s$ and $h_t$ is an extension of $h_{t-1}$; that is, as far as the rest of the argument goes, $t$ might as well be the base case, except for the fact that the roles of $A, \Phi$ are replaced by $B, \Psi$.

Similarly, only analogous changes would have to be made to prove Statements 1.-4. in the case where $t$ is a “forth stage.”

Recall that, by definition, the requirement $R_{\Phi, \Psi}$ receiving attention at a stage $t$ implies the growth of $\text{dom}(\Psi_t)$ (resp. $\text{dom}(\Psi_t)$) to include initial segments of the universe of $A$ (resp. $B$) that include the domain (resp. range) of $h_{t-1}$ and at least one more element. Therefore, since, by Statements 2. and 3. of the above lemma, the domain (resp. range) of $h_t$ equals the $\text{dom}(\Psi_t)$ (resp. $\text{dom}(\Psi_t)$), $h := \bigcup_{t \geq s} h_t$ is a total, onto function from $A$ onto $B$. By Statement 1., $h_t$ is 1-1 at every stage $t \geq s$, so $h$ is 1-1. Finally, by Statements 3. and 4., for any $\vec{a} \in A$, and any formula $\rho(\vec{x})$ in the original language, $A \models \rho(\vec{a})$ iff $B \models \rho(h(\vec{a}))$. Therefore, $h : A \cong B$.

This completes the proof of the main result.

References

[1] 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.

[2] 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.
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