Minimal weak truth table degrees and computably enumerable Turing degrees

Rod Downey
Keng Meng Ng
Reed Solomon

May 9, 2017

Chapter 1

Introduction

Computability theorists have studied many different reducibilities between sets of natural numbers including one reducibility ($\leq_1$), many-one reducibility ($\leq_m$), truth table reducibility ($\leq_{tt}$), weak truth table reducibility ($\leq_{wtt}$) and Turing reducibility ($\leq_T$). One motivation for studying strong reducibilities (i.e. reducibilities stronger than Turing reducibility) stems from internal questions within computability theory associated with varying the access mechanism to the oracle. For example, before Post's Problem for the Turing degrees was solved, Post [29] solved it for the many-one degrees and the truth table degrees using an analysis of the connections between m-completeness and immunity and between tt-completeness and hyperimmunity. As another example, Nerode [24] characterized the connection between $\leq_{tt}$ and $\leq_T$ using partial recursive functionals which are total on all oracles.

A second motivation for studying strong reducibilities is that most natural reducibilities arising in classical mathematics tend to be stronger than Turing reducibility. Abstract algebra provides many examples of this phenomena. In combinatorial group theory, the word problem is one reducible to the conjugacy problem. In field theory, Frohlich and Shepherdson [11] proved that the root set R_F of a computable field F is Turing equivalent to the splitting set S_F of F. Miller [22] sharpened this result to show that while $S_F \leq_1 R_F$, it is possible to have $R_F \not\leq_1 S_F$. Steiner [37] strengthened Miller's negative result by constructing a computable field F for which $R_F \not\leq_{wtt} S_F$. For vector spaces, Downey and Remmel [8] proved that if V is an enumerable subspace of V_∞, then the degrees of the computably enumerable (c.e.) bases of V are precisely the weak truth table degrees below the degree of V.

Examples also abound outside of algebra. In differential geometry, wtt-reducibility proved fundamental in the work of Nabutovsky and Weinberger [23], as studied by Csima [4] and Soare [35]. In algorithmic randomness, Downey, LaForte and Terwijn [7, 9] showed that presentations of halting probabilities coincide with ideals in the c.e. wtt-degrees, and Reimann and Slaman (e.g. [31]) demonstrated that truth table degrees are precisely the correct notion for studying randomness notions for continuous measures.

A final motivation is a technical one: results about strong reducibilities and their interactions with Turing reducibility can lead to significant insight into the structure of (for example) the Turing (T-)degrees. A good example is the first paper of Ladner and Sasso [20] in which they construct locally distributive parts of the c.e. T-degrees using the *wtt*-degrees (via contiguous degrees) and their interactions with the T-degrees. Extensions of this concept resulted in the first naturally definable antichain by Cholak, Downey and Walk [1] and similar definability results from Downey, Greenberg and Weber [6]. These definability results are actively being extended via notions of *wtt*-reducibility by Downey and Greenberg [5].

For general information concerning these reducibilities, we refer the reader to the survey article by Odifreddi [26] as well as the books by Rogers [30], Odifreddi [27] and Soare [34].

Our main concern here is the interaction of *minimality* and *enumerability,* two of the most basic concepts in classical computability. Constructions of minimal degrees are typically effective forcing arguments of one kind or another and such constructions are relatively incompatible with building effective objects. For example, by the Sacks Splitting Theorem, no c.e. T-degree can be a minimal T-degree. On the other hand, it is known that there can be c.e. *sets* of minimal m-degree (for example, Lachlan [18]) and of minimal *tt*-degree (for example, Fejer and Shore [10]). Since *wtt*-reducibility is intermediate between $\leq_{tt}$ and $\leq_T$, it is natural to wonder what happens there. Again, the Sacks Splitting Theorem shows that the *wtt*-degree of a c.e. *set* cannot be a minimal *wtt*-degree, but this leaves open the intriguing possibility that a set with minimal *wtt*-degree might sit inside a c.e. T-degree. This question served as our primary motivation. Before we present our results, we discuss the history and motivation in more detail.

Whether minimal degrees exist is a basic question in any degree structure. Frequently, a positive answer to this algebraic question leads to a negative answer to the logical question of whether the first order theory (in the language of a partial order or an upper semi-lattice) is decidable. Spector [36] proved the existence of a minimal T-degree using a forcing argument with perfect trees. This type of construction eventually led to Lachlan's proof [16] that every countable distributive lattice can be embedded as an initial segment of the T-degrees and hence that the structure of the T-degrees (as an upper semi-lattice) is undecidable. Furthermore, the method of forcing with perfect closed sets is now a mainstay in set theory.

Spector's construction uses a $\mathbf{0}''$ oracle to construct a sequence of total trees which force T-minimality and hence gives a Δ^0_3 minimal T-degree. Because the trees are total, his construction also gives a minimal *wtt*-degree and a minimal *tt*-degree. Sacks [32] strengthened Spector's theorem to show that there are Δ^0_2 minimal T-degrees by using a $\mathbf{0}'$ oracle to define a sequence of partial recursive trees which force T-minimality. Because these trees are partial, his construction does not immediately give either a minimal *wtt*-degree or a minimal *tt*-degree. The use of an oracle in the construction of a minimal T-degree can be completely removed with a full approximation argument and such arguments can be used to build minimal T-degrees in a variety of contexts such as below any noncomputable c.e. T-degree (Yates [38]) or below any high Δ^0_2 T-degree (Cooper [3], later generalized by Jockusch [14] to any T-degree which is GH_1). This technique also uses partial trees and hence does not automatically produce minimal *wtt* or *tt*-degrees.

The other classical theme for the present work is that of enumerability and specifically the c.e. sets. For strong reducibilities such as $\leq_1$, $\leq_m$ and $\leq_{tt}$, the techniques for building minimal degrees and c.e. degrees can be combined. Lachlan proved that there is a c.e. minimal 1-degree ([17]) and a c.e. minimal m-degree ([18]). That is, there is a set A with minimal m-degree such that $A \equiv_m W_e$ for some c.e. set W_e. In the 1-degrees and the m-degrees, the property of being c.e. is closed downwards and therefore, to build such minimal degrees, it suffices to make them minimal within the c.e. 1-degrees or within the c.e. m-degrees. Marchenkov [21] proved that c.e. minimal tt-degrees exist, although the first direct construction of such a degree was given by Fejer and Shore [10].

As remarked earlier, for weaker reducibilities such as $\leq_T$ and $\leq_{wtt}$, the techniques for constructing minimal degrees and c.e. degrees do not mix. Sacks [33] proved that the c.e. T-degrees are dense and Ladner and Sasso [20] proved that the c.e. *wtt*-degrees are dense. So, in addition to there being no minimal T or *wtt*-degrees, there are no c.e. minimal T or *wtt*-covers. However, it is possible to get some positive results concerning the relationship between minimal T-degrees and c.e. T-degrees. For example, Yates [38] used a full approximation argument together with c.e. permitting to show that in the T-degrees, every noncomputable c.e. set bounds a minimal T-degree.

We look at Yates' Theorem from a different perspective. Instead of looking at whether noncomputable c.e. sets bound minimal degrees, we look at whether sets with minimal degree can bound noncomputable c.e. sets or can even be of c.e. degree. By the results mentioned above, if we work entirely within the T-degrees or the *wtt*-degrees, this is not possible, but it becomes nontrivial if more than one reducibility is involved. Although a minimal *wtt*-degree $\mathbf{d}$ cannot *wtt*-bound a noncomputable c.e. set, we look at what $\mathbf{d}$ bounds under Turing reducibility. Specifically, if A is a Δ^0_2 set with minimal *wtt*-degree, can there be a noncomputable c.e. set B such that $B \leq_T A$? Can we make $B \equiv_T A$? Our main theorems give a positive answer to the first question and a negative answer to the second question.

Theorem 1.1. *There is a Δ^0_2 set A and a noncomputable c.e. set B such that A has minimal wtt degree and $B \leq_T A$.*

Theorem 1.2. *No c.e. Turing degree can contain a set which is wtt-minimal.*

In addition, we show that the sets A realizing Theorem 1.1 cannot be close to $\mathbf{0}'$ in the sense that they cannot compute a promptly simple set.

Theorem 1.3. *Let V be a promptly simple c.e. set and let A be a Δ^0_2 set such that $A \geq_T V$. There exists a c.e. set B such that $0 <_T B \leq_{wtt} A$.*

In his injury-free solution to Post's Problem, Kučera [15] proved that if Y is a Δ^0_2 set of diagonally noncomputable Turing degree, then there is a promptly simple c.e. set $V \leq_T Y$. Therefore, we have the following corollary to Theorem 1.3.

Corollary 1.4. *Let A be a Δ^0_2 set such that there is a diagonally noncomputable function $f \leq_T A$. There exists a c.e. set B such that $0 <_T B \leq_{wtt} A$.*

If A has Martin-Löf Turing degree or PA Turing degree, then there is a diagonally noncomputable function $f \leq_T A$. Therefore, we obtain similar corollaries for such sets. Chapter 4 of Nies [25] has a thorough discussion of these notions including generalizations of Kučera's result for *wtt*-reductions.

Our main results take place within the Δ^0_2 sets. In the case of Theorem 1.1, this follows from the fact that full approximation arguments naturally produce Δ^0_2 sets. In the case of Corollary 1.4, we do not know if the hypothesis that A is Δ^0_2 can be weakened. It cannot be removed entirely because there are diagonally noncomputable functions of hyperimmune-free Turing degree and such degrees cannot bound noncomputable c.e. degrees.

We feel that the proof of Theorem 1.1 is of significant technical interest. The proof combines a full approximation argument to make A *wtt*-minimal with permitting to build the noncomputable c.e. set B such that $B \leq_T A$. Because of the complexity of the interactions between the *wtt*-minimality strategies and the permitting strategies, we need to use a Δ^0_3 method with linking in our tree of strategies to control the construction of the partial computable trees in the full approximation argument. The kind of inductive considerations needed for the construction of the Turing reduction somewhat resemble the methods used by Lachlan [19] in embedding nondistributive lattice in the c.e. degrees. Such techniques have hitherto never been used in a full approximation argument, which is why we will slowly work up to the full details. In Chapter 2, we give an informal sketch of the construction method for Theorem 1.1 and in Chapter 3, we present the full construction and prove it succeeds.

In Chapter 4, we prove Theorems 1.2 and 1.3 giving two different limitations on the set A in Theorem 1.1. Our proof of Theorem 1.2 is nonuniform and in Section 4.1 we prove this nonuniformity is necessary. In Section 4.2, we isolate a technical approximation condition, called an almost c.e. approximation, and we prove that if A has an almost c.e. approximation, then A is not *wtt*-minimal. In Section 4.3, we finish the proof of Theorem 1.3 by showing that if A has c.e. Turing degree but does not have an almost c.e. approximation, then A is not *wtt*-minimal. Finally, we prove Theorem 1.3 in Section 4.4.

Most of our terminology is standard and follows Soare [34]. For example, we use φ_e and W_e to denote the e-th partial computable function and the e-th computably enumerable set respectively. If Γ is a Turing reduction, we use $\Gamma^A_s(x)$ or $\Gamma^A(x)[s]$ to denote the result of running Γ for s steps with oracle A, and assume this computation only queries the oracle about numbers below s.

We use α, β, γ and δ to denote finite binary strings and λ to denote the empty string. We use $|\alpha|$ to denote the length of α, $\alpha * \beta$ to denote the concatenation of α and β, $\alpha * i$ to denote $\alpha * \langle i \rangle$, and α' to denote α with its last element removed. We write $\alpha \subseteq \beta$ to indicate that α is an initial segment of β and

$\alpha \subseteq X$ to denote that α is an initial segment of the set X viewed as an infinite binary string. $X \restriction n$ denotes the finite string $\langle X(0), \ldots, X(n) \rangle$.

The proof of Theorem 1.1 uses a full approximation argument for which Posner [28] provides an excellent introduction. The proof of Theorem 1.3 relies on basic results about promptly simple sets which can be found in Chapter XIII of Soare [34].

Finally, we use $[e]$ for the e^{th} weak truth table reduction. To be more specific, this reduction is given by a pair $e = \langle i, j \rangle$ where i is the index of a Turing functional Φ_i and j is the index of a partial computable function φ_j. We compute $[e]^A(n)$ by first calculating $\varphi_j(0), \ldots, \varphi_j(n)$. If any of these computations diverge, so does $[e]^A(n)$. If all of these computations converge, then we calculate $\Phi_i^A(n)$. If this computation converges and never queries the oracle about a number $x > \varphi_j(n)$, then we set $[e]^A = \Phi_i^A(n)$. Otherwise, $[e]^A(n)$ diverges.

Chapter 2

Informal Construction

In this section, we present an informal description of the construction used to prove Theorem 1.1. For convenience, we restate the theorem below.

Theorem 1.1. *There is a Δ_2^0 set A and a noncomputable c.e. set B such that A has minimal wtt-degree and $B \leq_T A$.*

Throughout this chapter, we will introduce various pieces of terminology in an intuitive way and the formal definitions will appear in Chapter 3. We assume familiarity with full approximation arguments as in Posner [28] and with the notation for computable trees used in minimal degree constructions as in Chapter VI of Soare [34]. In particular, a tree T is a computable function $T : 2^{<\omega} \rightarrow 2^{<\omega}$ such that $T(\alpha * 0)$ and $T(\alpha * 1)$ are incomparable extensions of $T(\alpha)$ with $T(\alpha * 1)$ to the left of $T(\alpha * 0)$. The nodes $T_s(\alpha)$ for small values of α in a tree T_s defined at stage s during the construction will do work towards meeting a minimality requirement while nodes $T_s(\alpha)$ for large values of α will be defined trivially by $T_s(\alpha * i) = T_s(\alpha) * i$.

Recall that $[e]$ denotes the e^{th} *wtt*-reduction while φ_e denotes the e^{th} Turing-reduction. We use λ to denote the empty string and α' to denote the string obtained from α by removing the last element. Whenever we define a number to be *large* or the length of a string to be *long*, we mean for it to be larger than (or longer than) any number or string used in the construction so far.

To make A have minimal *wtt*-degree, we meet

$$R_e : [e]^A \text{ total} \Rightarrow A \leq_{wtt} [e]^A \text{ or } [e]^A \text{ is computable.}$$

To make B noncomputable, we satisfy

$$P_e : B \neq \overline{W_e}.$$

We also need to meet the global requirements that B is c.e. and $B \leq_T A$ by a Turing reduction Γ which we build.

We use a full approximation argument to satisfy the R_e requirements. To meet a single R_e requirement, we build a sequence of computable trees $T_{e,s}$ on which we attempt to find $[e]$-splittings. A node $T_{e,s}(\alpha)$ is said to $[e]$-*split* if there is an $x \leq s$ such that

$$[e]_s^{T_{e,s}(\alpha*0)}(x) \downarrow \neq [e]_s^{T_{e,s}(\alpha*1)}(x) \downarrow .$$

We say that the number x is a *splitting witness* for the node $T_{e,s}(\alpha)$. A node which $[e]$-splits is said to be in the *high* $[e]$-*state* and a node which does not $[e]$-split is said to be in the *low* $[e]$-*state*.

In addition, we define the current path A_s which represents our approximation to A at the beginning of stage s. During stage s, strategies will be allowed to alter the path A_s as part of their action. Therefore, in the full construction A_s really has two subscripts $A_{\eta,s}$ where η was the last strategy to act. For simplicity of notation right now, we omit the second subscript. We also occasionally leave off the stage number subscripts, especially in our diagrams where they cause unnecessary clutter. In general, if the current path A_s goes through a node $T_{e,s}(\alpha)$, then it also goes through $T_{e,s}(\alpha * 0)$ unless some strategy has actively moved the path to go through $T_{e,s}(\alpha * 1)$.

We make two significant modifications to a typical full approximation argument. First, rather than look for $[e]$-splits for every node, we only look for $[e]$-splits along the current path. To be more specific, suppose $T_{e,s}(\alpha)$ has been defined and we are trying to define $T_{e,s}(\alpha * i)$ for $i = 0, 1$. If $T_{e,s}(\alpha) \subseteq A_s$, then we look for extensions τ_0 and τ_1 which $[e]$-split and such that either τ_0 or τ_1 is on A_s. If we find such strings, then we define $T_{e,s}(\alpha * i) = \tau_i$. Otherwise we define $T_{e,s}(\alpha * i)$ as they were defined at stage $s - 1$ (if these nodes are still available) and if not, we extend $T_{e,s}(\alpha)$ trivially (that is, we take the first available extension strings). If $T_{e,s}(\alpha)$ is not on the current path, then we define $T_{e,s}(\alpha * i)$ as they were defined on $T_{e,s-1}$ (if possible) and otherwise define them by taking the first available extensions.

The second important modification is that we will occasionally move the current path A_s for the sake of a P requirement. (See Figure 2.1.) When a requirement moves the current path, it may challenge R_e to prove that $[e]$ is total on some finite set X_e of number using oracles on the new current path. In this situation, $[e]$ has converged on all the numbers in X_e using oracles from the old current path. As long as there is a number $x \in X_e$ for which $[e]$ does not see an oracle along the new current path which makes $[e]$ converge on x, R_e remains in a *nontotal* state and we define $T_{e,s}$ trivially. (That is, we attempt to keep the nodes of $T_{e,s}$ as they were at the last stage and take the first possible extensions when this is not possible.) If R_e remains in a nontotal state forever, then $[e]^A$ is not total and R_e is satisfied.

The current path A_s settles down on larger and larger initial segments as the construction proceeds and gives us A in the limit. Furthermore, nodes $T_{e,s}(\alpha)$ which are on A reach pointwise limits and final $[e]$-states. At the end of the construction, we are in one of three situations. Either R_e is eventually in a permanent nontotal state, the nodes $T_{e,s}(\alpha)$ along A are eventually in the high state or there is a string α such that $T_{e,s}(\alpha)$ is on A and all extensions of $T_{e,s}(\alpha)$ are permanently in the low state. If R_e is permanently in the nontotal

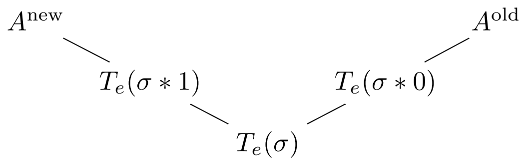

Figure 2.1: When the current path moves from $T_e(\sigma * 0)$ to $T_e(\sigma * 1)$, we challenge R_e to verify that it converges on all elements of $X_e = \{x \mid [e]^\tau_s(x)$ converges for some $\tau \supseteq T_e(\sigma * 0)\}$ using oracles along the new current path A^{new}.

state, then we win R_e because $[e]^A$ is not total. If the nodes along A are each eventually in the high state, then $A \leq_{wtt} [e]^A$. If sufficiently long nodes along A are eventually always in the low state, then $[e]^A$ is computable.

The basic idea of these computation lemmas is as in a typical full approximation argument. For the low state case, we show that once we see $[e]^{T_{e,s}(\alpha)}(x)$ converge at a stage s for some node $T_{e,s}(\alpha)$ on the current path, then this computation is equal to $[e]^A(x)$. As usual, this equality follows (for sufficiently long nodes $T_{e,s}(\alpha)$) because if not, we would later have the option of using $T_{e,s}(\alpha)$ and the node along A which gives the correct computation for $[e]^A(x)$ to make $T_{e,t}(\alpha')$ high splitting (where $t > s$ is a stage at which the correct computation appears).

For the high case, we can define A inductively using $[e]^A$ because the computations of $[e]^A$ tell us which half of each high split A eventually has to pass through. In general, this computation procedure gives a T-reduction $A \leq_T [e]^A$ and not a wtt-reduction $A \leq_{wtt} [e]^A$. To achieve a wtt-reduction, we incorporate *stretching*. (Stretching is also used by P strategies as described below.) Before describing the stretching procedure, we give the algorithm for determining the computable use for the wtt-reduction and then explain how to alter the construction so that this use function works.

To compute the use $u(m)$ of the reduction $A \leq_T [e]^A$ (and show it is a wtt-reduction) on a number m proceed as follows. Wait for a stage s and a node $T_{e,s}(\alpha) \subseteq A_s$ such that $T_{e,s}(\alpha)$ is in the high state and $|T_{e,s}(\alpha)| > m$. Define $u(m)$ to be the maximum of the splitting witnesses that R_e has seen in the construction so far.

The apparent problem with this definition is that the current path may move below $T_{e,s}(\alpha)$ at a later stage $t > s$ and along the new current path, there may not be a node of length $> m$ which is high splitting. To handle this potential problem, we redefine our trees by stretching each time we move the current path. (See Figure 2.2.) Suppose the current path moves from $T_{e,t}(\beta * 0) \subsetneq T_{e,t}(\alpha)$ to $T_{e,t}(\beta * 1)$ at stage t (for the sake of some lower priority requirement). Because $T_{e,s}(\beta) \subsetneq T_{e,s}(\alpha)$ and $T_{e,s}(\alpha)$ is high splitting, we know that $T_{e,s}(\beta)$ is high splitting (and is still high splitting at stage t). We let $\beta_{e,H}$ be the shortest node along the new current path such that $T_{e,t}(\beta_{e,H})$ is not high splitting. (In other words, $T_{e,t}(\beta'_{e,H})$ is the longest node on the new current path which is high

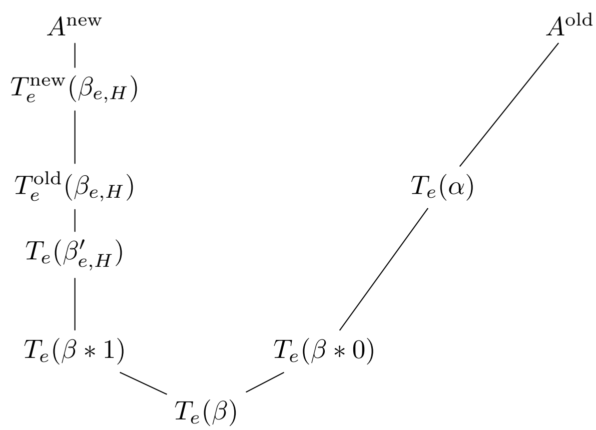

Figure 2.2: If $T_e(\alpha)$ is high splitting and the current path moves from $T_e(\beta * 0)$ to $T_e(\beta * 1)$, then we stretch $T_e^{\text{old}}(\beta_{e,H})$ to have value $T_e^{\text{new}}(\beta_{e,H})$ such that $|T_e^{\text{new}}(\beta_{e,H})| > |T_e(\alpha)| > m$.

splitting so $\beta \subseteq \beta'_{e,H} \subsetneq \beta_{e,H}$.) Because we only look for new high splits along the current path and because either $\beta'_{e,H} = \beta$ (so $T_{e,s}(\beta'_{e,H})$ is high splitting) or $\beta \subsetneq \beta'_{e,H}$ (so $T_{e,t}(\beta'_{e,H})$ is not on the current path and cannot change from low to high splitting between stages s and t), $T_{e,s}(\beta'_{e,H})$ must have been high splitting at stage s. Therefore, the splitting witness for $T_{e,t}(\beta'_{e,H})$ is less than the purported use $u(m)$.

Redefine $T_{e,t}(\beta_{e,H})$ so that it extends its old value, it has long length and is along the current path. (That is, its new length is longer than any number used so far in the construction and in particular is longer than m. For strings α such that $\beta_{e,H} \subsetneq \alpha$, extend the definition of $T_{e,t}$ trivially.) We refer to this redefinition process as stretching and say that the node $T_{e,t}(\beta_{e,H})$ is stretched. The node $T_{e,t}(\beta'_{e,H})$ is not changed by this process and it remains in the high state with the same splitting witness (which is less than $u(m)$).

Assume that the current path does not move below $T_{e,t}(\beta'_{e,H})$ after stage t. In this case, the reduction $A \leq_T [e]^A$ uses the witness for the high split at $T_{e,t}(\beta'_{e,H})$ to tell us that A passes through $T_{e,t}(\beta_{e,H})$ (which has length $> m$) since this node remains on the current path forever and hence is on A. However, this splitting witness is less than the purported use $u(m)$ for $A \leq_T [e]^A$, so $u(m)$ is correct. If the current path does move below $T_{e,t}(\beta'_{e,H})$ after stage t, then we repeat this stretching procedure at the next place where the current path moves. As long as such movement of the current path occurs only finitely often, we have the desired *wtt*-reduction.

To see that stretching does not interfere with the pointwise convergence of nodes along A, notice that a node is only stretched when the current path is moved and that node is the shortest node along the new current path which is not high splitting. Therefore, once a node becomes high splitting it is not stretched again. Since the current path will settle down on longer and longer segments, we will show that stretching only causes a finite disruption in the definition of the nodes along A. There are more subtle issues with stretching when multiple R strategies are involved and we address these below.

The basic strategy for meeting one P_e requirement (in the presence of a single R_e requirement of higher priority which is defining $T_{e,s}$) is to pick a node $T_{e,s}(\alpha)$ such that $T_{e,s}(\alpha * 0) \subset A_s$ at which to diagonalize and a large witness x with which to diagonalize. Since we have not yet put x into B, we define $\Gamma^{T_{e,s}(\alpha*0)}(x) = 0$. (Recall that Γ is the reduction we build to witness $B \leq_T A$.) We wait for x to enter W_e. If this never happens, then we never put x into B and we win P_e. If x does enter W_e at some later stage t, then we try to put x into B. (If the node $T_{e,s}(\alpha * 0)$ ever changes because of a new $[e]$-split, then we initialize this P_e strategy and start over with a new large witness x. In the full construction, we will have different P_e strategies guessing what the final state of the R_e strategy is.)

Before putting x into B, we need to get permission from A by changing A below the use of the computation $\Gamma^{T_{e,t}(\alpha*0)}(x) = 0$ which we defined at stage s. We would like to move the current path A_t from $T_{e,t}(\alpha * 0) \subseteq A_t$ to $T_{e,t}(\alpha * 1) \subseteq A_t$, declare $\Gamma^{T_{e,t}(\alpha*1)}(x) = 1$ and put x into B. However, there is a potential problem with this strategy. If the current path A_u, for some $u > t$, is ever moved so that $T_{e,t}(\alpha * 0) \subseteq A_u$ again, then we will have $\Gamma^{A_u}(x) = 0$ (by our definition that $\Gamma^{T_{e,t}(\alpha*0)}(x) = 0$) and $x \in B$. Since B must be c.e., we cannot remove x from B. Therefore, before we can put x into B, we must forbid the cone above $T_{e,t}(\alpha * 0)$ in the sense that we promise never to move the current path A_u for $u \geq t$ back to this cone again. If $T_{e,t}(\alpha)$ is in the high state, then this strategy is fine because there is no reason to look at nodes above $T_{e,t}(\alpha * 0)$ for a potential high split of $T_{e,t}(\alpha)$ since this node is already in the high state. Furthermore, we can tell from $[e]^A$ that $T_{e,t}(\alpha * 1) \subseteq A$ as opposed to $T_{e,t}(\alpha * 0) \subseteq A$.

However, there is a problem if $T_{e,t}(\alpha)$ is in the low state. If the true final state of R_e is low, then to compute $[e]^A(y)$ for any value y, we look for a node $T_{e,v}(\beta)$ on the current path in the low state such that $[e]^{T_{e,v}(\beta)}(y)$ converges and declare this to be the value of $[e]^A(y)$. This computation will be correct since otherwise we could put up another high split. However, if the node $T_{e,v}(\beta)$ happens to be in a cone like $T_{e,t}(\alpha*0)$ which is later forbidden, then it is possible that $[e]^A(y)$ has a different value and the forbidding process restricts us from putting up the new high splitting. Therefore, in this case, we do not want to rule out the possibility of using nodes above $T_{e,t}(\alpha * 0)$ to make $T_{e,t}(\alpha)$ high splitting at a later stage unless we have further evidence that $T_{e,t}(\alpha)$ should be in the low state. To accomplish this, we start a low challenge procedure to check that to the best of our knowledge, $T_{e,t}(\alpha)$ should be in the low state.

For the low challenge procedure, we let X_e be the finite set of numbers y for which we have seen $[e]$ convergence using a node above $T_{e,t}(\alpha * 0)$ as the oracle but we have not seen $[e]$ convergence using $T_{e,t}(\alpha)$ as the oracle. We move the current path A_t from $T_{e,t}(\alpha * 0)$ to $T_{e,t}(\alpha * 1)$ and declare the cone above $T_{e,t}(\alpha * 0)$ to be *frozen*. (See Figure 2.3.) This means that we no longer look at

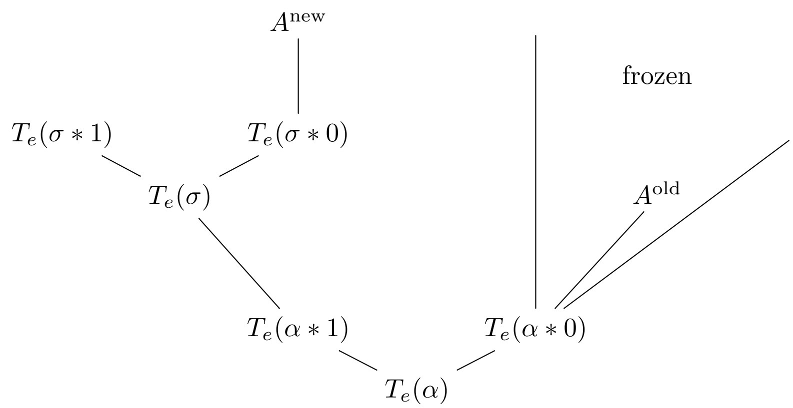

Figure 2.3: If $T_e(\alpha)$ is in the low state and we move the current path from $T_e(\alpha * 0)$ to $T_e(\alpha * 1)$ for the sake of P_e, then we freeze the cone above $T_e(\alpha * 0)$ until we have seen identical computations on all the elements of X_e using oracles along the new current path A^{new}. The auxiliary diagonalization node $T_e(\sigma)$ for P_e is chosen so that its length is greater than the use for any $[e]$ computation on an element in X_e.

computations involving nodes in this cone as oracles. P_e challenges R_e to verify that $T_{e,t}(\alpha)$ should be in the low state by providing computations along the new current path which agree with the computations from the old current path for all the numbers in X_e. We also pick a large auxiliary diagonalization spot $T_{e,t}(\sigma)$ with $T_{e,t}(\sigma * 0)$ on the (new) current path such that $T_{e,t}(\alpha * 1) \subsetneq T_{e,t}(\sigma)$. We define $\Gamma^{T_{e,t}(\sigma * 0)}(x) = 0$ since x has not yet been enumerated into B.

This auxiliary diagonalization spot is chosen to have length larger than the use of any of the computations for numbers in X_e. Since we are working with *wtt*-computations, R_e is only concerned with nodes on the current path below $T_{e,t}(\sigma)$ as oracles for the $[e]$ computations on numbers from X_e. Furthermore, while R_e is waiting for verification that $T_{e,t}(\alpha)$ really should be in the low state, it can suspend building T_e any further. That is, with the current path running through $T_{e,t}(\sigma * 0)$, R_e thinks that $[e]^A$ will not be total until it actually sees computations involving all the numbers in X_e.

If R_e sees a computation at stage $u > t$ on some element of X_e using an oracle on the current path which differs from the computation using the oracle above $T_{e,t}(\alpha * 0)$, then it unfreezes the cone above $T_{e,u}(\alpha * 0)$ (which is the same as $T_{e,t}(\alpha * 0)$ since R_e does not change T_e while it is low challenged) and it uses this computation to put $T_{e,u}(\alpha)$ in the high state. In this case, we initialize the P_e strategy and let it work with a new large witness x' at the same node $T_{e,u}(\alpha)$. (In the full construction, we will actually have a separate P_e strategy guessing that the final R_e state is high.) Since this node now has the high state, we know that we will win P_e with this new witness x' (either because x' never enters W_e or because x' does enter W_e and we can immediately diagonalize since $T_{e,u}(\alpha)$

is now in the high state).

If R_e sees computations at stage $u > t$ using oracles along the current path for all the numbers in X_e and they agree with the computations using oracles above $T_{e,t}(\alpha * 0)$, then R_e has met the low challenge and it is safe to forbid the cone above $T_{e,u}(\alpha*0)$ because we have identical computations in a nonforbidden part of the tree. That is, any future high splitting which might want to use a node above $T_{e,u}(\alpha * 0)$ can use a node above $T_{e,u}(\alpha * 1)$ instead which gives the same computation. To perform the diagonalization in this case, we use the auxiliary split $T_{e,u}(\sigma)$. We move the current path from $T_{e,u}(\sigma*0)$ to $T_{e,u}(\sigma*1)$, declare the cones above $T_{e,u}(\alpha*0)$ and $T_{e,u}(\sigma*0)$ to be forbidden, put x into B, and declare $\Gamma^{T_{e,u}(\sigma*1)}(x) = 1$. The forbidding action is allowed for $T_{e,u}(\alpha * 0)$ because we have identical computations for all numbers in X_e above $T_{e,u}(\alpha*1)$ and it is allowed for $T_{e,u}(\sigma * 0)$ because the length of this node was chosen large. That is, when we chose $T_{e,t}(\sigma)$, we had not looked at any computations above this node and because $T_{e,t}(\sigma)$ has length greater than the $[e]$ use for any number in X_e, we never need to look at computations above this node when verifying the lowness. Therefore, we are not committed to any computations above $T_{e,u}(\sigma * 0)$ at the time it is forbidden.

Finally, we might never see convergence on some number in X_e using any node above $T_{e,t}(\alpha * 1)$ (and below $T_{e,t}(\sigma)$) on the current path. In this case, R_e remains in the nontotal state forever and is won trivially because $[e]^A$ is not total. Furthermore, we can start a different version of the P_e strategy which guesses that R_e never meets the low challenge and which picks its own node above $T_{e,t}(\sigma * 0)$ at which to diagonalize and its own large witness with which to diagonalize. It gets to diagonalize immediately if it ever sees its witness enter W_e. Immediate forbidding is allowed for this strategy since the R_e strategy has not looked at any computations above $T_{e,t}(\sigma * 0)$.

This completes the informal description of the interaction between a single R strategy and a single P strategy. The interaction is significantly more complicated when multiple R strategies are involved. Before illustrating this interaction, we describe the tree of strategies used to control the full construction. An R_e strategy η has three possible outcomes: H, L, and N. We use the H (*high*) outcome whenever η finds a new high split along the current path. All strategies extending this outcome believe that the final $[e]$-state along A will be high. Each strategy μ with $\eta * H \subseteq \mu$ defines a large number p_μ and does not begin to act until the tree $T_{\eta,s}$ being built by η has the high state along the current path up to level p_μ. We use the N (*nontotal*) outcome whenever η has been challenged to verify its lowness and has not yet seen computations on all numbers in the set X_η it has been challenged to verify. All strategies extending this outcome believe that $[e]^A$ will not be total and hence they ignore the strategy R_e when making calculations about which action to take. We use the L (*low*) outcome whenever neither of the other two applies. Strategies extending this outcome think that $[e]^A$ may be total, but that the final $[e]$-state along A will be the low state. These outcomes are ordered in terms of priority with H the highest priority and N the lowest priority. (That is, $\eta * H$ is to the left of $\eta * L$ which is to the left of $\eta * N$.)

A P_e strategy η has two possible outcomes, S and W. The S outcome is used when P_e has already been satisfied by a diagonalization. Otherwise, we use the W outcome. The S outcome has higher priority than the W outcome. (That is, $\eta * S$ is to the left of $\eta * W$.) The action of a P_e strategy is finitary, while the action of an R_e strategy is infinitary.

Formally, the tree of strategies is defined by induction, with the empty string λ being the only R_0 strategy. If η is an R_e strategy, then $\eta * H$, $\eta * L$ and $\eta * N$ are P_e strategies. If η is a P_e strategy, then $\eta * W$ and $\eta * S$ are R_{e+1} strategies. To make the notation more uniform, we use $[\eta]$ and W_η to denote $[e]$ and W_e if η is an R_e or P_e strategy. We let $T_{\eta,s}$ denote the tree build at stage s by an R strategy η. Furthermore, we use the term *true path* to refer to the eventual true path through the tree of strategies. We use the term *current path* to denote the current approximation A_s to the set A.

To illustrate the remaining features of the construction, we consider four R strategies μ_i, $0 \leq i \leq 3$ and one P strategy η. Assume that the priorities are $\mu_0 < \mu_1 < \mu_2 < \mu_3 < \eta$, and that $\mu_1 = \mu_0 * L$, $\mu_2 = \mu_1 * H$, $\mu_3 = \mu_2 * L$, and $\eta = \mu_3 * H$. We consider the action of η. During this example, we assume that we never move to the left of these strategies in the tree of strategies and thus these strategies are never initialized. In particular, neither μ_0 nor μ_2 finds a new high split during our discussion.

Since η thinks the final state along A will be $\langle L, H, L, H \rangle$, there is no reason for η to pick a node at which to diagonalize that does not have this state. When η is first eligible to act, it picks a large number p_η. During each later stage at which η is eligible to act, η checks if the node $T_{\mu_3,s}(\alpha)$ along the current path with $|\alpha| = p_\eta$ has state $\langle L, H, L, H \rangle$. Until this occurs, η does not pick a node at which to diagonalize or a witness with which to diagonalize.

If η is on the true path, then eventually there will be such a node $T_{\mu_3,s}(\alpha)$. At this stage, η sets $\alpha_\eta = \alpha$ and picks a large witness x_η with which to diagonalize. η begins to wait for x_η to enter W_η (while keeping x_η out of B) and η defines $\Gamma^{T_{\mu_3,s}(\alpha_\eta * 0)}(x_\eta) = 0$. If x_η eventually enters W_η, then η begins a verification procedure to put x_η into B.

Assume x_η enters W_η at stage s. η moves the current path from $T_{\mu_3,s}(\alpha_\eta * 0)$ to $T_{\mu_3,s}(\alpha_\eta * 1)$ and freezes the cone above $T_{\mu_3,s}(\alpha_\eta * 0)$. η would like to put x_η into B, define $\Gamma^{T_{\mu_3,s}(\alpha_\eta * 1)}(x_\eta) = 1$ and forbid the cone above $T_{\mu_3,s}(\alpha_\eta * 0)$. There are two issues that need to be addressed before forbidding this cone. First, because we have moved the current path, we need to perform stretching for the sake of the strategies μ_1 and μ_3 which are in the high state in order to ensure that the set A has minimal *wtt*-degree. This issue is easy to address and does not stop us from immediately forbidding this cone. The second issue is more serious. The action of forbidding this cone is fine for μ_1 and μ_3 since $T_{\mu_3,s}(\alpha_\eta)$ is in the high μ_1 and μ_3 states. However, since $T_{\mu_3,s}(\alpha_\eta)$ is in the low μ_0 and μ_2 states, we cannot do this forbidding before finding identical computations (to the computations they have already seen) for these strategies along the new current path.

We begin with the issue of redefining the trees $T_{\mu_i,s}$ by stretching. First, we let $\beta_{\mu_0,L}$ and $\beta_{\mu_2,L}$ denote the strings such that the current path just moved

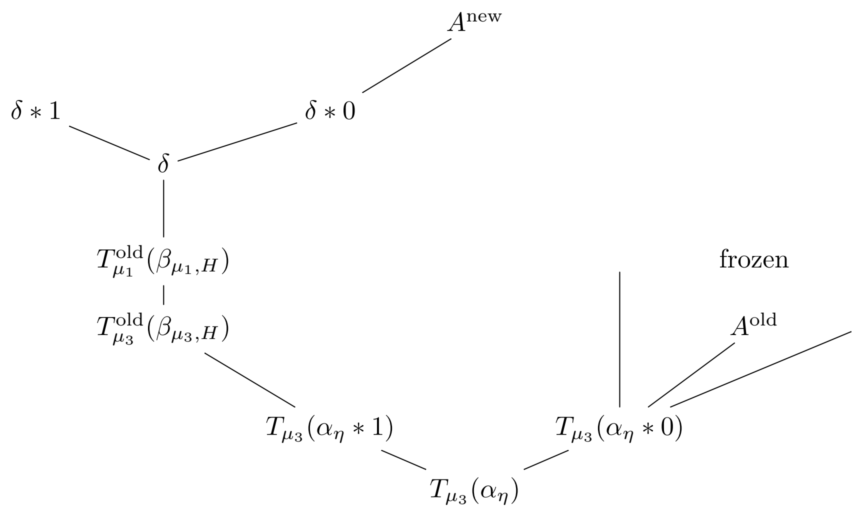

Figure 2.4: When we move the current path from $T_{\mu_3}(\alpha_\eta * 0)$ to $T_{\mu_3}(\alpha_\eta * 1)$ for the sake of the P strategy η, we freeze the cone above $T_{\mu_3}(\alpha_\eta * 0)$ and stretch the trees T_{μ_i}, $0 \leq i \leq 3$. In this figure, δ is equal to $T^{\text{new}}_{\mu_1}(\beta_{\mu_1,H})$, $T^{\text{new}}_{\mu_2}(\beta)$, $T^{\text{new}}_{\mu_3}(\beta_{\mu_3,H})$ and $T_{\mu_0}(\sigma_1)$.

from $T_{\mu_i,s}(\beta_{\mu_i,L} * 0)$ to $T_{\mu_i,s}(\beta_{\mu_i,L} * 1)$ (for $i = 0, 2$). Second, we let $\beta_{\mu_1,H}$ be the shortest string such that $T_{\mu_1,H}(\beta_{\mu_1,H})$ is on the new current path and $T_{\mu_1,s}(\beta_{\mu_1,H})$ is in the low μ_1 state. Hence, $T_{\mu_1,s}(\beta'_{\mu_1,H})$ is the longest node on the new current path which has state $\langle L, H\rangle$. Similarly, we define $\beta_{\mu_3,H}$ to be the shortest string such that $T_{\mu_3,s}(\beta_{\mu_3,H})$ is on the new current path and has state $\langle L, H, L, L\rangle$. In other words, $T_{\mu_3,s}(\beta'_{\mu_3,H})$ is the longest node on the new current path with state $\langle L, H, L, H\rangle$. Notice that $T_{\mu_3,s}(\beta_{\mu_3,H}) \subsetneq T_{\mu_1,s}(\beta_{\mu_1,H})$. Finally, let δ be a string with long length such that δ is on all of these trees and is on the new current path. Since δ has long length, our trees will have been defined trivially above δ in the sense that if $\delta \subseteq T_{\mu_i,s}(\alpha)$, then $T_{\mu_i,s}(\alpha * j) = T_{\mu_i,s}(\alpha) * j$. Therefore, in the redefinition process described below, the new versions of each tree will be subtrees of the old versions.

We redefine these trees by stretching. (See Figure 2.4. The node $T_{\mu_0}(\sigma_1)$ is introduced after the definition for stretching.) For μ_0, let $T_{\mu_0,s}$ remain the same. For μ_1, let $\hat{T}_{\mu_1} = T_{\mu_1,s}$ and we redefine $T_{\mu_1,s}$. For any node α such that $\alpha \subsetneq \beta_{\mu_1,H}$ or α is incomparable with $\beta_{\mu_1,H}$, let $T_{\mu_1,s}(\alpha) = \hat{T}(\alpha)$ (and this node retains its previous state). Redefine $T_{\mu_1,s}(\beta_{\mu_1,s}) = \delta$ and extend this definition trivially above here. That is, if $\beta_{\mu_1,H} \subseteq \alpha$ and $T_{\mu_1,s}(\alpha)$ has been defined, then set $T_{\mu_1,s}(\alpha * i) = T_{\mu_1,s}(\alpha) * i$ (and has all low states). Notice that the new definition of $T_{\mu_1,s}(\beta_{\mu_1,H})$ extends the old definition (since both the old value of $T_{\mu_1,s}(\beta_{\mu_1,H})$ and δ are on the new current path), so $T_{\mu_1,s}(\beta'_{\mu_1,s})$ is still in the high μ_1 state.

For μ_2, let β denote the string such that $T_{\mu_2,s}(\beta)$ is equal to the value of

$T_{\mu_1,s}(\beta_{\mu_1,H})$ before it was redefined by stretching. We set $\hat{T}_{\mu_2} = T_{\mu_2,s}$ and redefine $T_{\mu_2,s}$ as follows. For $\alpha \subsetneq \beta$ or α incomparable with β, set $T_{\mu_1,s}(\alpha) = \hat{T}_{\mu_2}(\alpha)$ (that is, leave these nodes unchanged). Redefine $T_{\mu_2,s}(\beta) = \delta$ and extend the definition of $T_{\mu_2,s}$ trivially above here. For μ_3, we follow essentially the same procedure as for μ_1. Set $\hat{T}_{\mu_3} = T_{\mu_3,s}$. For $\alpha \subsetneq \beta_{\mu_3,H}$ and α incomparable with $\beta_{\mu_3,H}$, define $T_{\mu_3,s}(\alpha) = \hat{T}_{\mu_3}(\alpha)$. Redefine $T_{\mu_3,s}(\beta_{\mu_3,H}) = \delta$ and extend the definition trivially above here. Notice that the new value of $T_{\mu_3,s}(\beta_{\mu_3,H})$ extends the old value of this node, so $T_{\mu_3,s}(\beta'_{\mu_3,H})$ still has state $\langle L, H, L, H\rangle$.

This completes the redefinition of these trees by stretching. The important properties to note are that each tree (except $T_{\mu_0,s}$) has a unique node along the new current path that is stretched, these nodes are all stretched to the same value (that is $T_{\mu_1,s}(\beta_{\mu_1,H}) = T_{\mu_2,s}(\beta) = T_{\mu_3,s}(\beta_{\mu_3,H}) = \delta$) and the longest nonstretched node on each tree retains its old state.

We turn to the issue of verifying lowness for μ_0 and μ_2. As with the case of a single P strategy, we must calculate the sets X_{μ_0} and X_{μ_2} on which these strategies need to verify computations. The set X_{μ_0} is calculated as before: it contains all numbers y such that μ_0 has seen $[\mu_0]$ converge on y with an oracle extending $T_{\mu_0,s}(\beta_{\mu_0,L} * 0)$ but not with $T_{\mu_0,s}(\beta_{\mu_0,L})$ as an oracle. (Recall that $\beta_{\mu_0,L}$ marks the place on $T_{\mu_0,s}$ above which the current path just moved.) The set X_{μ_2} has to be calculated slightly differently by taking into account the states of the nodes extending $T_{\mu_2,s}(\beta_{\mu_2,L} * 0)$. Let γ be the string such that $T_{\mu_2,s}(\gamma) = T_{\mu_3,s}(\alpha_\eta)$. Because μ_2 sees the state of $T_{\mu_2,s}(\gamma)$ as $\langle L, H, L\rangle$, when μ_2 looks for a high splitting for this node, it only looks at extensions of $T_{\mu_2,s}(\gamma)$ which have high μ_1 state. Therefore, we define X_{μ_2} to be all y such that μ_2 has seen a computation on y using an oracle above $T_{\mu_2,s}(\beta_{\mu_2,L} * 0)$ which has high μ_1 state and has not seen a computation on y using $T_{\mu_2,s}(\beta_{\mu_2,L})$ as the oracle. (Notice that the node $T_{\mu_2,s}(\beta_{\mu_2,s})$ and the tree above $T_{\mu_2,s}(\beta_{\mu_2,L} * 0)$ are not effected by the stretching procedure.) These are the numbers for which μ_2 has to verify its lowness.

If both $X_{\mu_0} = \emptyset$ and $X_{\mu_2} = \emptyset$, then η has permission from all of the R strategies μ_i for $i = 0, 1, 2, 3$ to immediately put x_η into B and forbid $T_{\mu_3,s}(\alpha_\eta * 0)$. (It has permission from μ_1 and μ_3 because $T_{\mu_3,s}(\alpha_\eta)$ is high μ_1 and μ_3 splitting and it has permission from μ_0 and μ_2 because there are no numbers on which these strategies need to verify their lowness.) Assume this is not the case so that some verification of lowness for either μ_0 or μ_2 (or both) is required. We split into the cases when $X_{\mu_2} = \emptyset$ and when $X_{\mu_2} \neq \emptyset$. Handling these cases requires the introduction of links into our tree of strategies.

First, assume that $X_{\mu_2} = \emptyset$ and $X_{\mu_0} \neq \emptyset$. In this case, η has permission from μ_1, μ_2 and μ_3 to forbid the cone above $T_{\mu_3,s}(\alpha_\eta * 0)$ and only has to wait for μ_0 to verify the computations on numbers in X_{μ_0}. η defines σ_1 to be the string such that $T_{\mu_0}(\sigma_1) = \delta$ (where δ is the string used in the stretching process as shown in Figure 2.4) and defines $\Gamma^{T_{\mu_0,s}(\sigma_1 * 0)}(x_\eta) = 0$. (We need this Γ computation to be defined since we have not yet placed x_η into B and we do not know ahead of time whether μ_0 will eventually verify the computations on numbers in X_{μ_0}.) η places a link from μ_0 to η, challenges μ_0 to verify its lowness and passes the set X_{μ_0} and the string $\beta_{\mu_0,L}$ to μ_0.

At future stages, μ_0 checks whether there are computations with oracles above $T_{\mu_0,s}(\beta_{\mu_0,L} * 1)$ for all the numbers in X_{μ_0} which agree with the computations with oracles above $T_{\mu_0,s}(\beta_{\mu_0,L} * 0)$. Because $[\mu_0]$ is a *wtt* procedure and because δ was chosen to have long length, μ_0 never has to look at strings longer than $T_{\mu_0,s}(\sigma_1) = \delta$ for these computations. If μ_0 ever finds a disagreeing computation, it can put up a new high split, take outcome $\mu_0 * H$ and initialize the attempted diagonalization by η. (By our assumption for this informal description, this situation does not occur.) If μ_0 eventually finds identical computations for all the numbers in X_{μ_0}, then instead of taking outcome $\mu_0 * L$, it travels the link to η. Until such a stage arrives, μ_0 takes outcome $\mu_0 * N$ and strategies extending $\mu_0 * N$ define their trees higher up on $T_{\mu_0,s}$ so that they do not interfere with any of the nodes mentioned so far. Also, if μ_0 takes outcome N at every future stage, then $[\mu_0]^A$ is not total because it diverges on at least one number in X_{μ_0}. Therefore, assume that we eventually travel the link from μ_0 to η.

When we travel the link from μ_0 to η at stage $t > s$, η acts as follows. It moves the current path from $T_{\mu_0,t}(\sigma_1 * 0)$ to $T_{\mu_0,t}(\sigma_1 * 1)$ (these nodes are the same as they were at the end of stage s since all the action of strategies extending $\mu_0 * N$ takes place with longer nodes), it forbids the cone above $T_{\mu_0,s}(\alpha_\eta * 0)$ (since η has μ_0 permission to forbid this cone and it previously had permission from μ_i for $1 \leq i \leq 3$), it forbids the cone above $T_{\mu_0,t}(\sigma_1 * 0)$ (which is allowed by μ_0 since μ_0 did not need to look in this cone to verify its computations on numbers in X_{μ_0} and is allowed by μ_i for $1 \leq i \leq 3$ since $T_{\mu_0,s}(\sigma_1) = \delta$ was defined to have long length and only strategies extending $\mu_0 * N$ have been eligible to act between stages s and t, so none of the strategies μ_i for $0 \leq i \leq 3$ have looked at any computations in this cone) and it puts x_η into B. Because the only computations of the form $\Gamma^\gamma(x_\eta) = 0$ are $\gamma = T_{\mu_3,t}(\alpha_\eta * 0) = T_{\mu_3,s}(\alpha_\eta * 0)$ and $\gamma = T_{\mu_0,t}(\sigma_1 * 0) = T_{\mu_0,s}(\sigma_1 * 0)$, we have forbidden all strings which define a Γ computation on x_η to be equal to 0. η picks a large number k and defines $\Gamma^\gamma(x_\eta) = 1$ for all strings γ of length k which do not extend $T_{\mu_3,s}(\alpha_\eta * 0)$ or $T_{\mu_0,s}(\sigma_0 * 0)$. Therefore, $\Gamma^A(x_\eta) = 1$ and η has won its requirement.

Next, we consider the case when $X_{\mu_2} \neq \emptyset$. In this case, at stage s, η defines σ_1 to be the string such that $T_{\mu_2,s}(\sigma_1) = \delta$ (where δ is the string used in the stretching process at stage s as shown in Figure 2.4) and defines $\Gamma^{T_{\mu_2,s}(\sigma_1 * 0)}(x_\eta) = 0$. η places the link from μ_2 to η. We challenge μ_0 and μ_2 to verify their lowness (and pass them the strings $\beta_{\mu_0,L}$ and $\beta_{\mu_2,L}$ and the sets X_{μ_0} and X_{μ_2} respectively). We challenge μ_1 to verify its highness and define $x_{\mu_1} = x_\eta$. The meaning and purpose of this high challenge has not come up yet and will be explained below. Since μ_1 is an R strategy, it does not keep a value x_{μ_1} for the purposes of diagonalization. However, as we shall see, μ_1 may need to take over the Γ definition of x_η temporarily and hence it needs to retain this value as a parameter.

Consider how the construction proceeds after stage s. Until μ_0 verifies its lowness, it takes outcome $\mu_0 * N$ and the strategies extending $\mu_0 * N$ work higher on the trees and do not effect the nodes defined above. Assume that μ_0 eventually meets its low challenge at stage $s_0 > s$.

At s_0, μ_0 takes outcome $\mu_0 * L$ and μ_1 becomes eligible to act for the first time since stage s. μ_1 needs to verify that $T_{\mu_1,s_0}(\beta_{\mu_1,H})$ should be in the high $[\mu_1]$ state. (Because strategies containing $\mu_0 * N$ work higher on the trees, we have $T_{\mu_1,s_0}(\beta_{\mu_1,H}) = T_{\mu_1,s}(\beta_{\mu_1,H})$, $T_{\mu_1,s_0}(\beta_{\mu_1,H} * i) = T_{\mu_1,s}(\beta_{\mu_1,H} * i)$ for $i = 0, 1$ and the current path still goes through $T_{\mu_1,s_0}(\beta_{\mu_1,H} * 0)$. For the rest of this informal explanation, we take it for granted that strategies to the right of the μ_i or η strategies do not cause any of the named nodes defined by these strategies to change and do not cause the current path to move below any of these nodes.)

The point of verifying that $T_{\mu_1,s_0}(\beta_{\mu_1,H})$ is in the high μ_1 state is that μ_2 eventually needs to verify that it is in the low state by finding computations for each number in X_{μ_2} using oracles along the current path which are in the high μ_1 state. The length of $T_{\mu_1,s_0}(\beta_{\mu_1,H})$ was stretched at stage s, so it has length longer that the $[\mu_2]$ use of any number in X_{μ_2}. But, we need this node to be in the high μ_1 state in order to use it as a potential oracle for these $[\mu_2]$ computations on X_{μ_2}.

μ_1 begins to look for a high splitting for $T_{\mu_1,s_0}(\beta_{\mu_1,H})$. Because $T_{\mu_1,s_0}(\beta'_{\mu_0,H})$ is already high μ_1 splitting, $T_{\mu_1,s_0}(\beta_{\mu_1,H})$ is the first node on the current path which is not high μ_1 splitting. Until μ_1 finds a potential high split for this node, it takes outcome $\mu_1 * L$.

Suppose μ_1 eventually finds a pair of strings τ_0 and τ_1 which could give a high splitting for $T_{\mu_1,s_0}(\beta_{\mu_1,H})$ with either τ_0 or τ_1 on the current path. (Recall that we only look for new splittings for which half of the splitting lies on the current path. If τ_0 and τ_1 have this property, then either one or both satisfy $T_{\mu_1,s_0}(\beta_{\mu_1,H} * 0) \subseteq \tau_i$ since this node remains on the current path.) Consider the action that η eventually wants to take if this entire verification procedure stated by η comes to a conclusion. η wants to move the current path from the node $T_{\mu_2,s}(\sigma_1 * 0) = T_{\mu_1,s_0}(\beta_{\mu_1,H} * 0)$ to the node $T_{\mu_2,s}(\sigma_1 * 1) = T_{\mu_1,s_0}(\beta_{\mu_1,H} * 1)$ and forbid the cone above $T_{\mu_2,s}(\sigma_1 * 0)$ before enumerating x_η into B (because we are committed to $\Gamma^{T_{\mu_2,s}(\sigma_1 * 0)}(x_\eta) = 0$). Therefore, if we define a new high splitting for $T_{\mu_1,s_0}(\beta_{\mu_1,H})$ at stage $s_1 > s_0$, we want the values of $T_{\mu_1,s_1}(\beta_{\mu_1,H} * i)$ to satisfy the condition

$$T_{\mu_1,s_0}(\beta_{\mu_1,H} * i) \subseteq T_{\mu_1,s_1}(\beta_{\mu_1,H} * i)$$

for $i = 0, 1$. If the potential splitting pair τ_0 and τ_1 satisfies this condition, then we use them to make $T_{\mu_1,s_1}(\beta_{\mu_1,H})$ high splitting and take outcome $\mu_1 * H$. In this case, we say that μ_1 has met its high challenge.

However, it may not be the case that τ_0 and τ_1 satisfy this condition. It is possible that when we find these nodes τ_0 and τ_1 at stage $s_1 > s_0$, both nodes extend $T_{\mu_1,s_0}(\beta_{\mu_1,H} * 0)$. In this case, we want to press μ_1 to find an appropriate half for the high splitting which extends $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 1) = T_{\mu_1,s_0}(\beta_{\mu_1,H} * 1) = T_{\mu_2,s}(\sigma_1 * 1)$. Because we have two different computations using oracles extending $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0) = T_{\mu_1,s_0}(\beta_{\mu_1,H} * 0)$, this pressing amounts to forcing μ_1 to find any oracle extending $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 1)$ which gives a convergent computation with the splitting witness w_{μ_1} for the μ_1 splitting strings τ_0 and τ_1. (The splitting witness w_{μ_1} is the number on which the $[\mu_1]$ computations using

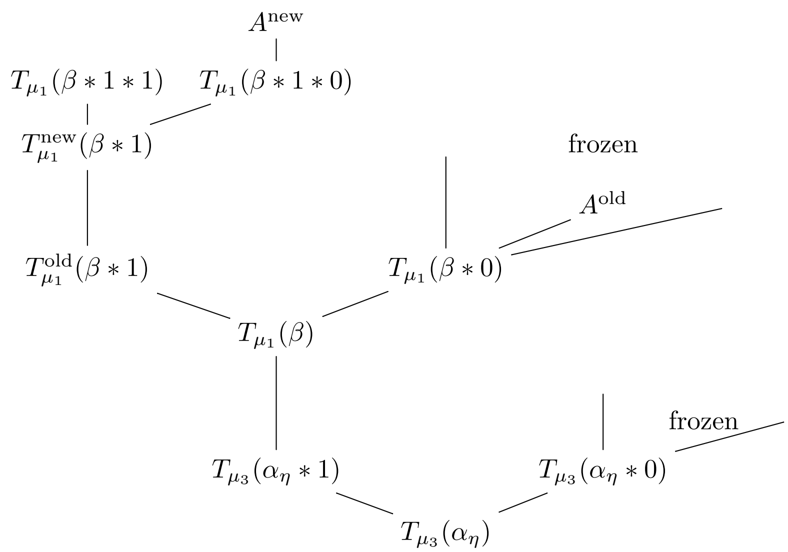

Figure 2.5: This figure represents our actions at stage s_1 when μ_1 finds a potential high split using nodes τ_0 and τ_1 extending $T_{\mu_1}(\beta_{\mu_1,H} * 0)$. For ease of notation, we have used β in place of $\beta_{\mu_1,H}$.

oracles τ_0 and τ_1 differ.) If μ_1 finds such a computation using a node extending $T_{\mu_1,s_0}(\beta_{\mu_1,H} * 1)$, then it can use this node together with one of τ_0 or τ_1 to get a high splitting for $T_{\mu_1,s_1}(\beta_{\mu_1,H})$ which has the required property above.

To accomplish this goal, μ_1 moves the current path from $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0)$ to $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 1)$ and freezes the cone above $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0)$. (See Figure 2.5.) Because μ_1 has moved the current path, it redefines the trees T_{μ_0,s_1} and T_{μ_1,s_1} by stretching. As before, we set $\beta_{\mu_0,L}$ to be the string such that the current path just moved from $T_{\mu_0,s_1}(\beta_{\mu_0,L} * 0)$ to $T_{\mu_0,s_1}(\beta_{\mu_0,L} * 1)$. Because $\mu_0 * L \subseteq \mu_1$, the tree T_{μ_0,s_1} remains the same. To redefine T_{μ_1,s_1}, set $\hat{T}_{\mu_1} = T_{\mu_1,s_1}$. For α such that $\alpha \subsetneq \beta_{\mu_1,H} * 1$ or α is incomparable with $\beta_{\mu_1,H} * 1$, define $T_{\mu_1,s_1}(\alpha) = \hat{T}_{\mu_1}(\alpha)$ (that is, leave these nodes unchanged). Redefine $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 1)$ to have long length and lie on the new current path (and hence the new definition of $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 1)$ extends the old definition). Extend the definition of T_{μ_1,s_1} trivially above this node.

Between the time μ_0 met its original low challenge at stage s_0 and the stage s_1 at which μ_1 finds the potential high split, μ_0 may have looked at computations involving oracles above $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0)$. Because we may or may not ever unfreeze the cone above this node, μ_0 needs to verify these computations along the new current path. Therefore, μ_1 issues a low challenge to μ_0 to verify the computations it has seen in this frozen cone.

μ_1 defines the set X_{μ_0} of numbers on which μ_0 has seen computations using oracles extending $T_{\mu_0,s_1}(\beta_{\mu_0,L} * 0)$ but not using $T_{\mu_0,s_1}(\beta_{\mu_0,L})$ as an oracle. It passes this set X_{μ_0} and the string $\beta_{\mu_0,L}$ to μ_0 and challenges μ_0 to verify its

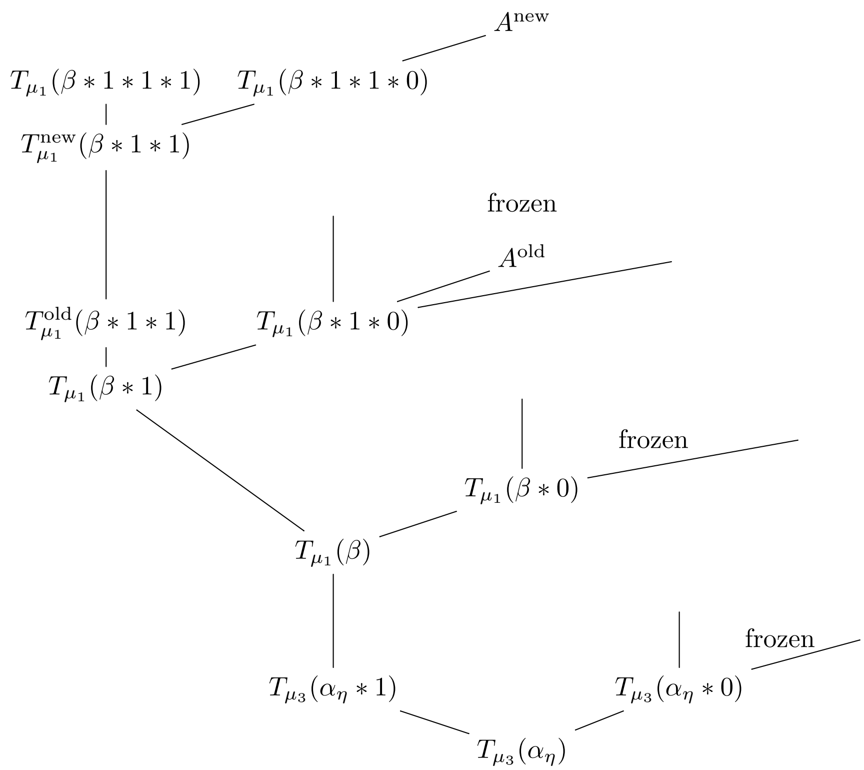

Figure 2.6: This figure represents our action at stage s_2 when μ_1 begins the process to forbid the cone above $T_{\mu_1}(\beta_{\mu_1,H} * 1 * 0)$ to eliminate the Γ definition using this node as the oracle. For ease of notation, we have used β in place of $\beta_{\mu_1,H}$.

lowness on these numbers. Furthermore, because μ_1 has moved the current path away from the node $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0) = T_{\mu_2,s}(\sigma_1 * 0)$ which was used by η in the Γ definition on x_η, μ_1 needs to take over the Γ definition of x_η. When μ_1 was challenged to verify its highness, we set $x_{\mu_1} = x_\eta$, so μ_1 defines $\Gamma^{T_{\mu_1,s_1}(\beta_{\mu_1,H}*1*0)}(x_{\mu_1}) = 0$. Once it makes this definition, μ_1 ends the stage. However, we do not want to allow μ_1 to initialize η, so μ_1 only initializes the strategies of lower priority than $\mu_1 * L$, including $\mu_1 * L$.

To see how the construction proceeds from here, assume that μ_0 eventually meets the low challenge issued by μ_1 and takes outcome $\mu_0 * L$ so that μ_1 is later eligible to act again. Because the length of $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 1)$ was stretched when μ_1 redefined the trees at stage s_1, it has length longer than the use of the *wtt* computation $[\mu_1]$ on the splitting witness w_{μ_1} for τ_0 and τ_1. Therefore, once μ_1 is eligible to act again, it checks if the $[\mu_1]$ computation on w_{μ_1} with oracle $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 1)$ converges. Until it sees this convergence, it takes outcome $\mu_1 * N$.

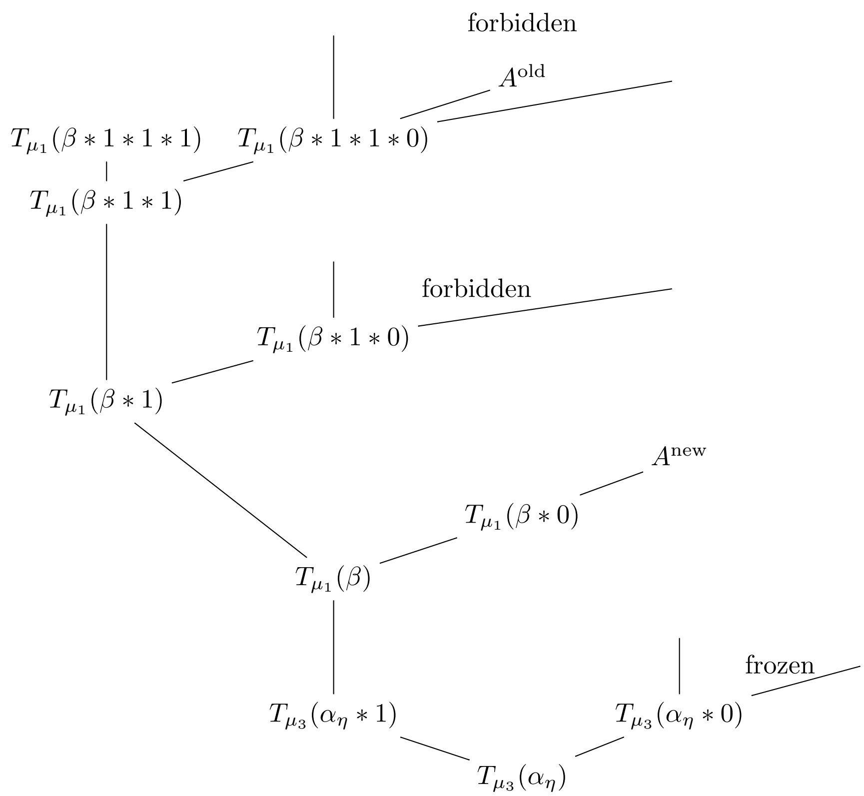

Figure 2.7: This figure represents the situation at stage s_3 when μ_1 returns the current path to $T_{\mu_1}(\beta_{\mu_1,H} * 0)$ and meets its high challenge by putting $T_{\mu_1}(\beta_{\mu_1,H})$ into the high μ_1 state. For ease of notation, we have used β in place of $\beta_{\mu_1,H}$.

If this computation never converges, then $[\mu_1]^A$ will not be total. Therefore, assume that this computation does eventually converge at stage $s_2 > s_1$. In this case, μ_1 wants to use the node $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1)$ and either τ_0 or τ_1 to make $T_{\mu_1,s_2}(\beta_{\mu_1,H})$ high μ_1 splitting. To do this, it needs to unfreeze the cone above $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0)$ that was frozen at stage s_1 and it will let the current path return to passing through $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0)$. However, when we perform this action, we don't want to leave the extra $x_{\mu_1} = x_\eta$ computation $\Gamma^{T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 0)}(x_{\mu_1}) = 0$ unforbidden because it could cause us problems if η eventually enumerates x_η into B. Therefore, before moving the current path back to $T_{\mu_1,s_1}(\beta_{\mu_1,H} * 0)$, μ_1 begins a verification procedure to forbid the cone above $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 0)$.

The R strategy μ_1 acts as though it were a P strategy with only one low R strategy of higher priority. (See Figure 2.6.) That is, it moves the current path from $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 0)$ to $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 1)$. μ_1 redefines T_{μ_0,s_2} and T_{μ_1,s_2} by stretching essentially as before: it defines $\beta_{\mu_0,L}$ and X_{μ_0}, leaves T_{μ_0,s_2} the same and stretches $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 1)$ to have long length. μ_1 calculates the set X_{μ_0}

of numbers which μ_0 has seen converge with an oracle above $T_{\mu_0,s_2}(\beta_{\mu_0,L} * 0)$ but not with $T_{\mu_0,s_2}(\beta_{\mu_0,L})$ as oracle. It defines $\Gamma^{T_{\mu_1,s_2}(\beta_{\mu_1,H}*1*1*0)}(x_{\mu_1}) = 0$ and issues a low challenge to μ_0 with $\beta_{\mu_0,L}$ and X_{μ_0}. Because $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 1)$ is redefined to have long length, μ_0 does not need to look above this node for any computations on the numbers in X_{μ_0}. Therefore, if this low challenge is met at $s_3 > s_2$, μ_1 forbids the cone above $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 0)$ (since μ_0 has verified the computations that used oracles above this node), forbids the cone above $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 1 * 0)$ (since μ_0 did not look at any computations above this cone), unfreezes the cone above $T_{\mu_1,s_3}(\beta_{\mu_1,H} * 0)$ and uses $T_{\mu_1,s_3}(\beta_{\mu_1,H} * 1)$ together with either τ_0 or τ_1 to make $T_{\mu_1,s_3}(\beta_{\mu_1,H})$ have high μ_1 state. The current path A_{s_3} also returns to passing through $T_{\mu_1,s_3}(\beta_{\mu_1,H} * 0)$ now that this node is unfrozen. (See Figure 2.7.) μ_1 has met its high challenge and takes outcome $\mu_1 * H$.

It might seem that there are too many μ_0 low challenges by μ_1. However, the first μ_0 low challenge issued by μ_1 at stage s_1 is because we cannot know whether μ_1 will ever see $[\mu_1]$ converge on w_{μ_1} with oracle $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1)$. If this computation never converges, then the cone above $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 0)$ in never unfrozen and so is essentially forbidden despite never being officially forbidden. Therefore, the first μ_0 low challenge by μ_1 at stage s_1 is to account for this possibility. The second μ_0 low challenge issued by μ_1 at s_2 is to allow the cone above $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 0)$ to be forbidden to remove the potentially damaging Γ computation on x_{μ_1} using this oracle.

Summing up the action for μ_1 which is challenged high, μ_1 meets its high challenge (in one of the two ways described above) by eventually finding a high splitting for $T_{\mu_1,s_0}(\beta_{\mu_1,H}) = T_{\mu_1,s_1}(\beta_{\mu_1,H})$ at some stage $s_3 \geq s_1$ such that $T_{\mu_1,s_0}(\beta_{\mu_1,H} * i) \subseteq T_{\mu_1,s_3}(\beta_{\mu_1,H} * i)$ for $i = 0, 1$. If it fails to find such a splitting, then it is either because μ_0 failed to meet some low challenge (in which case either we win the μ_0 requirement because $[\mu_0]^A$ is not total or else μ_0 finds a high split, takes outsome $\mu_0 * H$ and initializes μ_1) or because μ_1 failed to find an appropriate "second half" to a potential high split (in which case we win μ_1 because $[\mu_1]^A$ is not total). Furthermore, the current path at stage s_3 goes through $T_{\mu_1,s_3}(\beta_{\mu_1,H} * 0)$ and the computations $\Gamma^{T_{\mu_3,s}(\alpha_\eta * 0)}(x_\eta) = 0$ (defined by η when it originally chose x_η) and $\Gamma^{T_{\mu_1,s}(\beta_{\mu_1,H}*0)}(x_\eta) = \Gamma^{T_{\mu_2,s}(\sigma_1*0)}(x_\eta) = 0$ (defined by η at stage s when it started the verification procedure to put x_η into B) are the only Γ computations on x_η which are not forbidden at stage s_3. Finally, the node $T_{\mu_1,s_3}(\beta_{\mu_1,H}) = T_{\mu_1,s}(\beta_{\mu_1,H})$ has not changed since being stretched by η at stage s when η began its diagonalization process and is now in the high μ_1 state.

At stage s_3, μ_2 is eligible to act for the first time since stage s. μ_2 begins to verify its lowness as challenged by η at stage s. The current path still runs through $T_{\mu_3,s}(\alpha_\eta * 1)$ (where it was moved at stage s) through $T_{\mu_1,s_3}(\beta_{\mu_1,H})$ and $T_{\mu_1,s_3}(\beta_{\mu_1,H} * 0)$. (Of course, μ_3 has not been eligible to act since stage s.) We now have permission from μ_0, μ_1 and μ_3 to forbid the cone above $T_{\mu_3,s}(\alpha_\eta * 0)$ and only need to obtain μ_2 permission by verifying its computations on the numbers in X_{μ_2} along the current path using oracles in the high μ_1 state (since $T_{\mu_3,s}(\alpha_\eta)$ was already in the high μ_1 state at stage s). Because the length of

$T_{\mu_1,s}(\beta_{\mu_1,H}) = T_{\mu_1,s_3}(\beta_{\mu_1,H})$ was stretched at stage s when X_{μ_2} was defined by η and because this node is now in the high μ_1 state, μ_2 does not need to look at any computations using oracles which extend this node. Furthermore, at stage s, η defined σ_1 so that $T_{\mu_2,s}(\sigma_1) = T_{\mu_1,s}(\beta_{\mu_1,H})$. Therefore $T_{\mu_2,s_3}(\sigma_1) = T_{\mu_2,s}(\sigma_1)$ and μ_2 does not need to look at any computations using oracles above $T_{\mu_2,s_3}(\sigma_1)$.

Until μ_2 sees the correct computations on these numbers using an oracle along the current path, it takes outcome $\mu_2 * N$. If there is a number in X_{μ_2} for which μ_2 never sees a correct computation, then $[\mu_2]^A$ is not total and we win requirement μ_2. If there is a number in X_{μ_2} for which μ_2 sees a computation which does not agree with the computation along the old current path that ran through $T_{\mu_3,s}(\alpha_\eta * 0)$, then μ_2 can use this computation to define a new μ_2 high splitting, take outcome $\mu_2 * H$ and initialize η. Therefore, assume that μ_2 eventually verifies these computations at a stage $s_4 > s_3$.

In this case, μ_2 follows the link to η. (Recall that when η started the diagonalization process at stage s, it placed a link from μ_2 to η.) η now has permission from μ_i, $0 \leq i \leq 3$ to forbid the cone above $T_{\mu_3,s}(\alpha_\eta * 0)$. However, before placing x_η in B, η also needs to worry about the computation $\Gamma^{T_{\mu_2,s}(\sigma_1 * 0)}(x_\eta) = 0$ that it defined at stage s after moving the current path. Therefore, μ_2 moves the current path from $T_{\mu_2,s_4}(\sigma_1 * 0) = T_{\mu_1,s_4}(\beta_{\mu_1,H} * 0)$ to $T_{\mu_2,s_4}(\sigma_1 * 1) = T_{\mu_1,s_4}(\beta_{\mu_1,H} * 1)$, redefines T_{μ_i,s_4} for $0 \leq i \leq 2$ by stretching and freezes the cone above $T_{\mu_2,s_4}(\sigma_1 * 0)$.

Because $T_{\mu_1,s_4}(\beta_{\mu_1,H})$ is already in the high $[\mu_1]$ state, η has permission from μ_1 to forbid the cone above $T_{\mu_2,s_4}(\sigma_1 * 0)$. Because we have not considered μ_3 since stage s when η originally began its diagonalization procedure, μ_3 has not seen any computations in this cone and hence η has permission from μ_3 to forbid this cone. Because $T_{\mu_2,s_4}(\sigma_1) = T_{\mu_2,s_3}(\sigma_1) = T_{\mu_2,s}(\sigma_1)$, μ_2 did not look at any computations in the cone above $T_{\mu,s_4}(\sigma_1 * 0)$ when it verified its computations on X_{μ_2} and hence has seen no computations in this cone. Therefore, η has permission from μ_2 to forbid this cone. However, μ_0 may have seen computations using oracles in the cone above $T_{\mu_2,s_4}(\sigma_1 * 0)$ between stage s_0 when μ_0 verified its lowness and stage s_4. Therefore, η still needs μ_0 permission to forbid this cone.

To obtain this permission, η defines $\beta_{\mu_0,L}$ to be the string such that the current path moves from $T_{\mu_0,s_4}(\beta_{\mu_0,L} * 0)$ to $T_{\mu_0,s_4}(\beta_{\mu_0,L} * 1)$ and defines X_{μ_0} to be the set of all numbers y such that μ_0 has seen a computation on y using an oracle extending $T_{\mu_0,s_4}(\beta_{\mu_0,L} * 0)$ but not using oracle $T_{\mu_0,s_4}(\beta_{\mu_0,L})$. η issues a low challenge to μ_0 with X_{μ_0}. The action proceeds just as in the case when $X_{\mu_0} \neq \emptyset$ and $X_{\mu_2} = \emptyset$. That is, η sets up another Γ definition on x_η using a long string on T_{μ_0,s_4}, places a link from μ_0 to η and waits for μ_0 to verify its lowness. When this occurs, η has the last remaining permission to forbid the cone above $T_{\mu_2,s_4}(\sigma_1 * 0)$ and it has the permission to forbid the new Γ computation on x_η since μ_0 does not need to look above this large node to verify its computations and none of μ_i for $1 \leq i \leq 3$ is eligible to act and to look at any computations in this cone while μ_0 is verifying its lowness. Therefore, when μ_0 verifies its lowness, η can safely place x_η into B, forbid the remaining Γ computations on x_η (including $T_{\mu_3,s}(\alpha_\eta * 0)$), pick a large number k and define $\Gamma^\gamma(x_\eta) = 1$ for all strings γ of length k which are not forbidden. After performing this action, η has won its requirement.

Chapter 3

Formal construction

This chapter is devoted to giving the formal construction for Theorem 1.1. We begin with some notational conventions. We use the letters η, ν and μ to refer to R and P strategies and we use α, β, γ, δ, σ and τ to denote finite binary strings. λ denotes the empty string and for any nonempty string α, α' denotes the string formed by removing the last element of α. For uniformity of presentation, we regard λ' and λ'' as a special symbols distinct from λ and set $T_{\lambda'',s}$ to be an identity tree for all s.

In the tree of strategies as defined in the last chapter, λ is an R_0 strategy. In general, an R_e strategy η has successors $\eta * H$, $\eta * L$ and $\eta * N$ ordered left to right by $\eta * H <_L \eta * L <_L \eta * N$. A P_e strategy μ has successors $\mu * S$ and $\mu * W$ ordered left to right by $\mu * S <_L \mu * W$. If μ is a P_e strategy, then μ' is an R_{e-1} strategy and μ will attempt to do its diagonalization on the tree $T_{\mu',s}$ built by μ'. If η is an R_e strategy, then η'' is an R_{e-1} strategy and η will attempt to build its tree $T_{\eta,s}$ as a subtree of the tree $T_{\eta'',s}$ built by η''. Because we use the extra symbol λ'' and assume that $T_{\lambda'',s}$ is the identity tree for all s, we can treat the highest priority R strategy λ as any other strategy.

The current path $A_{\eta,s}$ at stage s is defined by induction on the sequence of strategies η which are eligible to act at stage s. When η begins its action at stage s, it uses the current path $A_{\eta',s}$ and it may move this path during its action. (The strategy λ works with the current path $A_{\lambda',s}$ defined to be the final version of the current path from stage $s-1$. As above, this convention allows us to treat λ as any other strategy.) $A_{\eta,s}$ denotes the current path at the end of η's action. (Typically, the current path is the rightmost path through $T_{\eta,s}$ which does not pass through any frozen or forbidden nodes.)

Each R_e requirement η keeps several pieces of information. $G_\eta \in \{H, L, N\}^e$ represents η's fixed guess at the final $(e-1)$ state along A in $T_{\eta,s}$. For each $i < e$ there is a unique R_i strategy $\mu \subseteq \eta$. $G_\eta(i) \in \{H, L, N\}$ is defined such that $\mu * G_\eta(i) \subseteq \eta$. Typically, if η is eligible to act at stage s, η defines a tree $T_{\eta,s}$. Each node $T_{\eta,s}(\alpha)$ is assigned an e-state $U(T_{\eta,s}(\alpha)) \in \{H, L\}^{e+1}$ (called the η state of $T_{\eta,s}(\alpha)$) which is defined by induction as in a standard full approximation argument. The η'' state of a node $T_{\eta,s}(\alpha)$ is defined to be the

$(e-1)$ state of $T_{\eta'',s}(\gamma)$ where γ is such that $T_{\eta'',s}(\gamma) = T_{\eta,s}(\alpha)$. We make some technical comments below on comparing e-states of the form $U(T_{\eta,s}(\alpha))$ (which cannot contain the letter N) and e-states of the form G_ν (which can contain the letter N).

We will abuse terminology by using the phrase "the η state of $T_{\eta,s}(\alpha)$" to refer to the η state as defined above (for example when comparing the η state to G_μ for some μ extending η) and to refer to whether or not $T_{\eta,s}(\alpha)$ is η high splitting (for example when saying that $T_{\eta,s}(\alpha)$ has the high or low η state). It will be clear from context which of these meanings is intended.

The second parameter for an R_e strategy η is $p_\eta \in \mathbb{N}$. This parameter denotes the level on the η'' tree at which we start building T_η. Before defining $T_{\eta,s}$, we wait for a string α such that $|\alpha| = p_\eta$, $U(T_{\eta'',s}(\alpha)) = G_\eta$ (ignoring for the moment the fact that G_η may contain the letter N), and $T_{\eta'',s}(\alpha)$ is on the current path. When we find such a string, we set the parameter $\alpha_\eta = \alpha$ and begin to define $T_{\eta,s}$ by setting $T_{\eta,s}(\lambda) = T_{\eta'',s}(\alpha_\eta)$.

If η is challenged low, then it is given a finite set X_η of numbers on which it is waiting for convergence and a string $\beta_{\eta,L}$ such that it is looking for convergence above either $T_{\eta,s}(\beta_{\eta,L} * 0)$ or $T_{\eta,s}(\beta_{\eta,L} * 1)$ depending on which strategy challenged η to verify its lowness.

If η is challenged high, then η is given a string $\beta_{\eta,H}$ and a number x_η. The string $\beta_{\eta,H}$ determines the node $T_{\eta,s}(\beta_{\eta,H})$ which η needs to verify is high splitting and the number x_η is the number on which η may need to define Γ computations higher on the tree if it has to move the current path while verifying its highness. In addition, η may define a number w_η on which the $[\eta]$ computations disagree for potential splitting strings τ_0 and τ_1 while it attempts to find an appropriate string τ_2 so that the two halves of the new high split will extend $T_{\eta,s}(\beta_{\eta,H} * 0)$ and $T_{\eta,s}(\beta_{\eta,H} * 1)$.

Each P_e requirement η also keeps several pieces of information. G_η is η's fixed guess at the final e-state and it is defined as in the R_e case. η defines a number p_η and a string α_η as in the R_e case and attempts to do its diagonalization at the node $T_{\eta',s}(\alpha_\eta)$. η also choses a large witness x_η with which it attempts to diagonalize.

During the construction, strategies may freeze or forbid certain nodes. We use the term *active* to refer to a node which is neither frozen nor forbidden and the term *inactive* to refer to a node that is either frozen or forbidden. We adopt the following conventions concerning inactive nodes. If α is declared frozen or forbidden, then so are all extensions of α. If $\alpha * 0$ and $\alpha * 1$ are both inactive, then so is α. We never search for splits in the part of the tree which is inactive. After the construction, we verify that the current path is always infinite.

Before giving our methods for defining trees, we make one comment on comparing e-state strings. If η is an R_e strategy, then the e-state for a node $T_{\eta,s}(\alpha)$ is denoted $U(T_{\eta,s}(\alpha))$ and is a string $\tau \in \{H, L\}^{e+1}$. If $\tau = U(T_{\eta,s}(\alpha))$ and a lower priority strategy μ is comparing τ and G_μ, then for all i such that $G_\mu(i) = N$, μ treats τ as though $\tau(i) = N$. That is, μ is guessing that the R_i strategy of higher priority is not total and hence has no interest in the i component of any e-state string. In other words, when comparing e-state strings,

μ ignores the entries for which μ is guessing nontotality. Although we continue to use the standard notations $=$, $<$, and $>$ for comparing e-state strings, they always have this addition meaning in the context of a strategy μ.

We also need to clarify the definition for a number to be large or a string to be long. During this construction, each tree $T_{\eta,s}$ which is defined at stage s is a total function from $2^{<\omega}$ to $2^{<\omega}$. When we define a number to be large, we want to say that it is larger than any number we have looked at in a meaningful way in the construction. Therefore, we define a number n to be *large* to mean that n is larger than any parameter defined so far in the construction and larger than any string used as an oracle in any computation looked at so far in the construction. We say that a string is *long* if its length is large.

We have three basic ways of defining the tree $T_{\eta,s}$ from $T_{\eta'',s}$. In all cases, η will already have defined its parameters p_η and α_η. First, we define $T_{\eta,s}$ *trivially from* $T_{\eta'',s}$ as follows. Let $T_{\eta,s}(\lambda) = T_{\eta'',s}(\alpha_\eta)$ and continue by induction. Assume that $T_{\eta,s}(\beta) = T_{\eta'',s}(\gamma)$ has been defined. If there is a most recent stage $t < s$ at which η defined $T_{\eta,t}$ and η has not been initialized since t, then we attempt to keep $T_{\eta,s}$ the same as it was at stage t. If $T_{\eta,s}(\beta) = T_{\eta,t}(\beta)$ and for $i \in \{0,1\}$, $T_{\eta,t}(\beta * i)$ is still on $T_{\eta'',s}$, then set $T_{\eta,s}(\beta * i) = T_{\eta,t}(\beta * i)$ and $U(T_{\eta,s}(\beta)) = U(T_{\eta,t}(\beta))$. If any of those conditions fails or there is not such stage t, then set $T_{\eta,s}(\beta * i) = T_{\eta'',s}(\gamma * i)$ and $U(T_{\eta,s}(\beta)) = U(T_{\eta'',s}(\gamma)) * L$.

We sometimes define a subtree of $T_{\eta,s}$ trivially by following the same algorithm above an already defined node. If $T_{\eta,s}(\beta)$ has already been defined, then *defining $T_{\eta,s}$ trivially above $T_{\eta,s}(\beta)$* means to use the above algorithm to define $T_{\eta,s}(\delta)$ for all $\beta \subset \delta$.

Second, we may define $T_{\eta,s}$ by *searching for active splittings* on $T_{\eta'',s}$. Set $T_{\eta,s}(\lambda) = T_{\eta'',s}(\alpha_\eta)$ and proceed by induction. Assume that $T_{\eta,s}(\beta) = T_{\eta'',s}(\gamma)$ has been defined.

If $T_{\eta,s}(\beta) \subseteq A_{\eta',s}$ and has η'' state G_η, then we look for an appropriate splitting extension with half of the split lying on $A_{\eta',s}$. Check for active nodes τ_0 and τ_1 on $T_{\eta'',s}$ such that

1. $|\tau_0|, |\tau_1| \leq s$ with τ_0 to the right of τ_1,
2. $T_{\eta'',s}(\gamma) \subseteq \tau_0, \tau_1$,
3. either $\tau_0 \subseteq A_{\eta',s}$ or $\tau_1 \subseteq A_{\eta',s}$,
4. $U(\tau_0) = U(\tau_1) = G_\eta$, and
5. there is an $x \leq s$ such that $[\eta]_s^{\tau_0}(x) \downarrow \neq [\eta]_s^{\tau_1}(x) \downarrow$.

If there exist such sequences, then take the first pair found, set $T_{\eta,s}(\beta * i) = \tau_i$ and set $U(T_{\eta,s}(\beta)) = G_\eta * H$. (We assume that once η has chosen such a pair, it continues to chose the same pair at future stages as long as the pair remains on $T_{\eta''}$.) In all other cases, define $T_{\eta,s}$ trivially above $T_{\eta,s}(\beta)$.

Third, a strategy η may redefine trees $T_{\mu,s}$ for R strategies $\mu \subsetneq \eta$ *by stretching*. η could be an R or a P strategy, but in either case, η will have just moved the current path. Let δ be a string of long length such that $T_{\lambda'',s}(\delta)$ is on the new current path. (Recall that $T_{\lambda'',s}$ is the identity tree, so $T_{\lambda'',s}(\delta) = \delta$.) In particular, because δ is chosen large, this node is on all of the trees $T_{\nu,s}$ for R strategies $\nu \subseteq \eta$ and this node is in the low ν state for all such ν. Furthermore, the current path goes through $T_{\lambda'',s}(\delta * 0) = \delta * 0$.

For each R strategy μ such that $\mu * L \subseteq \eta$ or $\mu * N \subseteq \eta$, let $\beta_{\mu,L}$ be the string such that η moved the current path from $T_{\mu,s}(\beta_{\mu,L} * 0)$ to $T_{\mu,s}(\beta_{\mu,L} * 1)$ or from $T_{\mu,t}(\beta_{\mu,L} * 1)$ to $T_{\mu,t}(\beta_{\mu,L} * 0)$. The procedure for redefining trees by stretching splits into two cases.

The first case is when there are no R strategies μ such that $\mu * H \subseteq \eta$. In this case, each tree $T_{\mu,s}$ remains the same and the stretching procedure has no effect. (The point in that since there are no high splitting nodes, we do not need the stretching procedure to help us define a *wtt* computation of the form $A \leq_{wtt} [\mu]^A$ for any of these strategies μ at the end of the construction. Therefore, the stretching will not be necessary in this case.)

The second case is when there is at least one R strategy μ such that $\mu * H \subseteq \eta$. Let $\mu_0 \subseteq \mu_1 \subseteq \cdots \subseteq \mu_k \subseteq \eta$ be the R strategies such that $\mu_j * H \subseteq \eta$. Let $\beta_{\mu_j,H}$ be the longest string such that $T_{\mu_j,s}(\beta_{\mu_j,H})$ is on the new current path and $U(T_{\mu_j,s}(\beta'_{\mu_j,H})) = G_{\mu_j} * H$. That is, $T_{\mu_j}(\beta_{\mu_j,H})$ is the first node on the new current path with state $G_{\mu_j} * L$. Because $U(T_{\mu_j,s}(\beta_{\mu_j,H})) = G_{\mu_j} * L$, we have

$$T_{\mu_k,s}(\beta_{\mu_k,H}) \subseteq T_{\mu_{k-1},s}(\beta_{\mu_{k-1},H}) \subseteq \cdots \subseteq T_{\mu_0,s}(\beta_{\mu_0,H}) \subseteq \delta.$$

We want to redefine the trees $T_{\nu,s}$ for R strategies $\nu \subsetneq \eta$ such that the node $T_{\mu_j,s}(\beta_{\mu_j,H})$ is stretched to have value $T_{\lambda'',s}(\delta)$. The redefinition of $T_{\nu,s}$ splits into three subcases.

First, if $\nu \subsetneq \mu_0$, then $T_{\nu,s}$ remains the same. Second, if $\nu = \mu_j$, the let $\hat{T}_{\mu_j} = T_{\mu_j,s}$ and we redefine $T_{\mu_j,s}$ as follows. For all α such that $\alpha \subsetneq \beta_{\mu_j,H}$ or α is incomparable with $\beta_{\mu_j,H}$, set $T_{\mu_j,s}(\alpha) = \hat{T}_{\mu_j}(\alpha)$ and let $U(T_{\mu_j,s}(\alpha)) = U(\hat{T}_{\mu_j}(\alpha))$. Define $T_{\mu_j,s}(\beta_{\mu_j,H}) = T_{\lambda'',s}(\delta)$ and $U(T_{\mu_j,s}(\beta_{\mu_j,H})) =$ all low states. Continue the definition of $T_{\mu_j,s}$ trivially from $\hat{T}_{\mu_j}$ above $T_{\mu_j,s}(\beta_{\mu_j}, H)$. Notice that $T_{\mu_j,s}(\beta_{\mu_j,H} * 0) = \delta * 0$ and so the current path runs through this node.

The third subcase is quite similar to the second subcase with a slight change in notation. If none of the first two subcases applies, let $j \leq k$ be the greatest number such that $\mu_j \subseteq \nu$. Set $\hat{T}_\nu = T_{\nu,s}$ and let β be the string such that $\hat{T}_\nu(\beta) =$ the value of $T_{\mu_j,s}(\beta_{\mu_j,H})$ before it was redefined by stretching. For all α such that $\alpha \subsetneq \beta$ or α is incomparable with β, set $T_{\nu,s}(\alpha) = \hat{T}_\nu(\alpha)$ and $U(T_{\nu,s}(\alpha)) = U(\hat{T}_\nu(\alpha))$. Define $T_{\nu,s}(\beta) = T_{\lambda'',s}(\delta)$ and $U(T_{\nu,s}(\beta)) =$ all low states. Continue the of $T_{\nu,s}$ trivially from $\hat{T}_\nu$ above this node. This completes the definition of redefining trees by stretching.

The construction proceeds in stages with the action at each stage s directed by the tree of strategies. At stage 0, we begin with the current path $A_0 = A_{\lambda',0} = \emptyset$ and let λ be eligible to act. At the beginning of stage $s > 0$, we define the current path A_s and $A_{\lambda',s}$ so that $A_s = A_{\lambda',s} = A_{\nu,s-1}$ where ν is the last strategy which was eligible to act at stage $s - 1$. We let λ be eligible to act to start stage s. When a strategy η acts at stage s, it may move the current path by explicitly defining $A_{\eta,s}$ from $A_{\eta',s}$. If it does not explicitly define a new current path, then $A_{\eta,s} = A_{\eta',s}$. (That is, the current path does not change.) Similarly, any parameters not explicitly redefined or canceled by initialization are assumed to retain their previous values. We proceed according to the action of the strategies until a strategy explicitly ends the stage. When a strategy η ends a stage, it will either initialize all lower priority strategies or it will initialize all strategies of lower priority than $\eta * L$ (including $\eta * L$). When a strategy is initialized, all of its parameters are canceled and become undefined. If the strategy η is eligible to act at stage s, then s is called an η stage.

We need to clarify the definition of the functional Γ. We make new definitions for Γ at the end of each stage s after we have initialized the appropriate strategies. For each $x \leq s$ such that x is not currently equal to x_η for some P strategy η and such that $x \notin B_s$, set $\Gamma^Y(x) = 0$ for all sets Y. If $x = x_\eta$ for for some P strategy η, then the construction takes care of the definition of Γ on x.

Action for a P strategy η:

Case 1. η has not acted before or has been initialized since its last action. Define p_η large, end the stage and initialize all lower priority strategies.

Case 2. p_η is defined but α_η is not defined. Let α be the unique string such that $|\alpha| = p_\eta$ and $T_{\eta',s}(\alpha) \subseteq A_{\eta',s}$. Check if $U(T_{\eta',s}(\alpha)) = G_\eta$. If not, then end the stage now and initialize the lower priority strategies. If so, define $\alpha_\eta = \alpha$, define x_η to be large and set $\Gamma^{T_{\eta',s}(\alpha_\eta * 0)}(x_\eta) = 0$. End the stage now and initialize all lower priority strategies. (After the construction we verify that $T_{\eta',s}(\alpha_\eta * 0) \subseteq A_{\eta',s} = A_{\eta,s}$ and that this node remains on the current path at future η stages unless η is initialized or η moves the current path in the verification procedure called in Case 3 below.)

Case 3. α_η and x_η are defined. Check if $x_\eta \in W_\eta$. If not, then let $\eta * W$ be eligible to act. If so, begin a verification procedure with $\sigma_0 = \alpha_\eta$. (The verification procedure is described after the description of the action for an R strategy.) At each subsequent η stage until the verification procedure concludes, the verification procedure will end the stage and initialize the lower priority strategies. (If η is on the true path, then the action of the verification procedure will be finitary.)

Case 4. The verification procedure called in Case 3 ends at this stage. Forbid all cones that were η frozen by the verification procedure. Put x_η into B. Let n be a large number. For all strings γ of length n which are not η forbidden, define $\Gamma^\gamma(x_\eta) = 1$. Declare η satisfied and take outcome $\eta * S$. At future η stages, take outcome $\eta * S$.

Action for an R strategy η:

Case 1. η has not acted before or has been initialized since the last time it acted. In this case, define p_η large, end the stage and initialize all strategies of lower priority.

Case 2. η has defined p_η but not α_η. Let α be the unique string such that $|\alpha| = p_\eta$ and $T_{\eta'',s}(\alpha) \subseteq A_{\eta',s}$. If $U(T_{\eta'',s}(\alpha)) = G_\eta$ then define $\alpha_\eta = \alpha$. Otherwise, leave α_η undefined. In either case, end the stage and initialize all lower priority strategies.

Case 3. α_η is defined and η is not challenged. Define $T_{\eta,s}$ by setting $T_{\eta,s}(\lambda) = T_{\eta'',s}(\alpha_\eta)$ and searching for active splittings. If η finds a new high splitting along the current path, then let $\eta * H$ act. Else, let $\eta * L$ act.

Case 4. η was challenged high at stage $t < s$. At stage t, η was given a number x_η and a string $\beta_{\eta,H}$ such that $U(T_{\eta,t}(\beta'_{\eta,H})) = G_\eta * H$ and $T_{\eta,t}(\beta_{\eta,H})$ was stretched at the end of stage t (and hence has all low states at the end of stage t). Let γ denote the string such that at stage t we had $T_{\eta,t}(\beta_{\eta,H}) = T_{\eta'',t}(\gamma)$. After the construction, we verify the following properties. $T_{\eta'',s}(\gamma) = T_{\eta'',t}(\gamma) = T_{\eta,t}(\beta_{\eta,H})$, $U(T_{\eta'',s}(\gamma)) = G_\eta$ and $T_{\eta'',s}(\gamma * 0) \subseteq A_{\eta',s}$. At each η stage u such that $t < u < s$, $T_{\eta,u}$ was defined trivially from $T_{\eta'',u}$. If $u < v$ are η stages such that $t < u < v < s$, then $T_{\eta,t}(\beta_{\eta,H}) = T_{\eta,u}(\beta_{\eta,H}) = T_{\eta,v}(\beta_{\eta,H})$ and for $i \in \{0,1\}$, $T_{\eta,t}(\beta_{\eta,H} * i) \subseteq T_{\eta,u}(\beta_{\eta,H} * i) = T_{\eta,v}(\beta_{\eta,H} * i)$. Because η was defined trivially at any such stage u, we also have that $T_{\eta,u}(\beta_{\eta,H} * i) = T_{\eta'',u}(\gamma * i)$. Finally, when η was challenged high, the challenging strategy defined $\Gamma^{T_{\eta,t}(\beta_{\eta,H}*0)}(x_\eta) = 0$.

This case splits into the two subcases below. It is possible that η has also been challenged low at some stage after t and before the current stage. If this has occurred, then η must be in Subcase 4A.

Subcase 4A: η has not yet found a potential high splitting for $T_{\eta,t}(\beta_{\eta,H})$. Check if there are active strings τ_0 and τ_1 on $T_{\eta'',s}$ (with τ_0 to the right of τ_1) such that $T_{\eta'',s}(\gamma) = T_{\eta,t}(\beta_{\eta,H}) \subseteq \tau_0, \tau_1$, $U(\tau_0) = U(\tau_1) = G_\eta$, $\exists w_\eta([\eta]_s^{\tau_0}(w_\eta)\downarrow \neq [\eta]_s^{\tau_1}(w_\eta)\downarrow)$ and either $\tau_0 \subseteq A_{\eta',s}$ or $\tau_1 \subseteq A_{\eta',s}$. If not and η is also low challenged, proceed to Case 5 below. If not and η is not low challenged, then define $T_{\eta,s}$ trivially from $T_{\eta'',s}$ and take outcome $\eta * L$. η remains high challenged. If there are such strings τ_0 and τ_1, then fix τ_0, τ_1 and w_η, and consider the following two subcases of Subcase 4A. (Because the current path goes through $T_{\eta'',s}(\gamma*0)$ and $T_{\eta,t}(\beta_{\eta,H}*0) \subseteq T_{\eta'',s}(\gamma*0)$, we have that either $T_{\eta,t}(\beta_{\eta,H}*i) \subseteq \tau_i$ for $i = 0,1$ or $T_{\eta,t}(\beta_{\eta,H} * 0) \subseteq \tau_0, \tau_1$. Therefore, the two cases below suffice.)

Subcase 4A(i): τ_0 and τ_1 satisfy $T_{\eta,t}(\beta_{\eta,H} * i) \subseteq \tau_i$. Define $T_{\eta,s}$ from $T_{\eta'',s}$ by searching for splittings, using τ_0 and τ_1 as the successors of $T_{\eta,s}(\beta_{\eta,H})$. η is no longer challenged high and $\eta * H$ is the next strategy eligible to act. Notice that we have $T_{\eta,t}(\beta_{\eta,H} * i) \subseteq T_{\eta,s}(\beta_{\eta,H} * i)$.

Subcase 4A(ii): $T_{\eta,t}(\beta_{\eta,H} * 0) \subseteq \tau_0, \tau_1$. Define $T_{\eta,s}$ trivially from $T_{\eta'',s}$. Freeze the cone above $T_{\eta,t}(\beta_{\eta,H} * 0)$ and move the current path to be the rightmost active path through $T_{\eta,s}(\beta_{\eta,H} * 1)$.

Redefine the trees $T_{\mu,s}$ for $\mu \subsetneq \eta$ by stretching. Furthermore, stretch $T_{\eta,s}(\beta_{\eta,H} * 1)$ to have the same long length as the other stretched nodes. (That is, set $\hat{T} = T_{\eta,s}$ and redefine $T_{\eta,s}$ as follows. For all α such that $\alpha \subsetneq \beta_{\eta,H} * 1$ or α is incomparable to $\beta_{\eta,H} * 1$, set $T_{\eta,s}(\alpha) = \hat{T}(\alpha)$ and $U(T_{\eta,s}(\alpha)) = U(\hat{T}(\alpha))$. Define $T_{\eta,s}(\beta_{\eta,H} * 1) = T_{\lambda'',s}(\delta)$ (where δ is as in the stretching process just completed) and $U(T_{\eta,s}(\beta_{\eta,H} * 1)) =$ all low states. Extend the definition of $T_{\eta,s}$ trivially from $\hat{T}$ above this node.) Define $\Gamma^{T_{\eta,s}(\beta_{\eta,H}*1*0)}(x_\eta) = 0$.

For each R strategy μ such that $\mu * L \subseteq \eta$, define X_μ to be the finite set of all x for which μ has seen $[\mu]^\tau(x)$ converge for some τ on $T_{\mu,s}$ such that

$U(\tau) = G_\mu$ and $T_{\mu,s}(\beta_{\mu,L} * 0) \subseteq \tau$ but μ has not seen $[\mu]_s^{T_{\mu,s}(\beta_{\mu,L})}(x)$ converge. ($\beta_{\mu,L}$ is defined by the stretching process in the previous paragraph.) For all μ with $\mu * L \subseteq \eta$, pass X_μ and $\beta_{\mu,L}$ to μ and challenge μ low. For all μ such that $\mu * H \subseteq \eta$, challenge μ high, pass $\beta_{\mu,H}$ to μ and set $x_\mu = x_\eta$. ($\beta_{\mu,H}$ is defined by the stretching process in the previous paragraph.) End the stage and initialize all strategies of lower priority than $\eta * L$ including $\eta * L$. At the next η stage (unless η has been initialized), η will act in Subcase 4B below.

Subcase 4B. At the previous η stage, η acted in Subcase 4A(ii) or η acted in this subcase and did not call a verification procedure. Let $u < s$ denote the stage at which η acted in Subcase 4A(ii). Define $T_{\eta,s}$ trivially from $T_{\eta'',s}$. After the construction, we verify that $T_{\eta,s}(\beta_{\eta,H} * 1) = T_{\eta,u}(\beta_{\eta,H} * 1)$ and this string has state $G_\eta * L$. Furthermore, $T_{\eta,u}(\beta_{\eta,H} * 1 * i) \subseteq T_{\eta,s}(\beta_{\eta,H} * 1 * i)$ and the current path goes through $T_{\eta,s}(\beta_{\eta,H} * 1 * 0)$. Because $T_{\eta,u}(\beta_{\eta,H} * 1)$ was stretched at stage u, $T_{\eta,s}(\beta_{\eta,H} * 1)$ has length longer than the $[\eta]$ use on w_η (which is the splitting witness for τ_0 and τ_1 from Subcase A). Check if $[\eta]_s^{T_{\eta,s}(\beta_{\eta,H} * 1)}(w_\eta)$ converges. If not, let $\eta * N$ act. If so, call a verification procedure with $\sigma_0 = \beta_{\eta,H} * 1$. At subsequent η stages until the verification procedure finishes, it will end the stage and initialize strategies of lower priority than $\eta * L$ including $\eta * L$.

When the verification procedure finishes (abusing notation, at stage s), unfreeze the cone above $T_{\eta,t}(\beta_{\eta,H} * 0)$ (which was frozen in Subcase 4A(ii)). This action unfreezes the strings τ_0 and τ_1 from Subcase 4A(ii). Set $\hat{\tau}$ to be either τ_0 or τ_1, depending on which gives the computation that differs from the computation given by $T_{\eta,u}(\beta_{\eta,H} * 1)$ on w_η. Move the current path to be the rightmost active path through $\hat{\tau}$. Forbid all remaining η frozen cones. Define $T_{\eta,s}$ by searching for splitting, taking $T_{\eta,s}(\beta_{\eta,H} * 1) = T_{\eta,u}(\beta_{\eta,H} * 1)$ and $T_{\eta,s}(\beta_{\eta,H} * 0) = \hat{\tau}$ to make $T_{\eta,s}(\beta_{\eta,H})$ high splitting. When this definition is complete, redefine the trees $T_{\mu,s}$ for $\mu \subsetneq \eta * H$ by stretching. (Notice that we stretch $T_{\eta,s}$ as part of this stretching process.) Let $\eta * H$ act and η is no longer challenged high.

Case 5. η was challenged low at stage $t < s$ and passed the set X_η and a string $\beta_{\eta,L}$. If $X_\eta = \emptyset$, then take outcome $\eta * L$ and η is no longer low challenged. If $X_\eta \neq \emptyset$, then proceed as follows.

η was challenged low either by a verification procedure or by an R strategy acting in Subcase 4A(ii) of its high challenge. In either case, $\beta_{\eta,L}$ is such that the current path was moved from $T_{\eta,t}(\beta_{\mu,L} * 0)$ to $T_{\mu,t}(\beta_{\mu,L} * 1)$ and the cone above $T_{\eta,t}(\beta_{\eta,L} * 0)$ was frozen at stage t by the challenging strategy. After the construction, we verify the following properties. If γ is such that $T_{\eta'',t}(\gamma) = T_{\eta,t}(\beta_{\eta,L})$, then $T_{\eta'',s}(\gamma) = T_{\eta'',t}(\gamma)$. If u is an η stage such that $t < u < s$, then $T_{\eta,t}(\beta_{\eta,L}) = T_{\eta,u}(\beta_{\eta,L})$ and $T_{\eta,t}(\beta_{\eta,L} * i) = T_{\eta,u}(\beta_{\eta,L} * i)$ for $i \in \{0,1\}$. (To be precise, when η was challenged low at stage t, it is possible that the challenging strategy stretched the node $T_{\eta,t}(\beta_{\eta,L} * 1)$. Therefore, the reference to this node is to the stretched version, if such stretching took place.) Finally, the current path continues to run through $T_{\eta,u}(\beta_{\eta,L} * 1)$.

By the definition of X_η, for each $x \in X_\eta$, there is a corresponding string γ_x on $T_{\eta,t}$ such that $T_{\eta,t}(\beta_{\eta,L} * 0) \subseteq \gamma_x$ and $[\eta]_t^{\gamma_x}(x)$ converges. Consider all nodes

δ such that $T_{\eta'',s}(\delta)$ is on the current path, $T_{\eta,t}(\beta_{\eta,L} * 1) \subseteq T_{\eta',s}(\delta)$, $|T_{\eta'',s}(\delta)|$ is greater than any of the $[\eta]$ uses for $x \in X_\eta$ and $U(T_{\eta'',s}(\delta)) = G_\eta$. If there is no such δ, then define $T_{\eta,s}$ trivially from $T_{\eta'',s}$ and take outcome $\eta * N$. Otherwise, let δ_η denote the shortest length such δ.

Consider each $x \in X_\eta$ in sequential order and check whether $[\eta]_s^{T_{\eta'',s}(\delta_\eta)}(x)$ converges. If not, then define $T_{\eta,s}$ trivially from $T_{\eta'',s}$ and take outcome $\eta * N$. If this computation does converge, then check whether it equals $[\eta]^{\gamma_x}(x)$. If so, then consider the next value in X_η. If not, then unfreeze all cones frozen by the challenging strategy, so in particular γ_x is unfrozen. Define $T_{\eta,s}$ from $T_{\eta'',s}$ by searching for splittings. γ_x and $T_{\eta'',s}(\delta_\eta)$ will give a new high split on $T_{\eta,s}$ so take outcome $\eta * H$. (In this case, since the strategy which challenged η extends $\eta * L$, it will be initialized at the end of the stage.) If all of the elements of X_η have convergent computations which agree with their γ_x computations, then define $T_{\eta,s}$ trivially from $T_{\eta'',s}$, declare the low challenge met and take outcome $\eta * L$ unless the challenging strategy established a link from η in which case follow the link.

Verification Procedure.

A verification procedure can be called either by a P strategy η or by an R strategy η acting in Subcase 4B of the high challenge. In either case, when η first calls the verification procedure, it has just defined a string σ_0 and it has a witness x_η. (The string σ_0 should contain a subscript indicating that it is part of a verification procedure called by η, but we omit this extra piece of notation.)

The verification procedure acts in cycles, beginning with the 0^{th} cycle. When the n^{th} cycles starts, we will have defined the string σ_n. If $n \geq 1$, then we will have followed a link from the strategy μ_{n-1} to η such that $\mu_{n-1} * L \subseteq \eta$ and μ_{n-1} is the lowest priority strategy challenged low by η at the $(n-1)^{\text{st}}$ cycle. (When the verification procedure is first called, we begin with σ_0 and have not followed any link. To make the notation uniform, we set $\mu_{-1} = \eta$ and treat the 0^{th} cycle like any other cycle.) The following is the action for the n^{th} cycle of this verification procedure.

At the start of the n^{th} cycle, the current path goes through $T_{\mu_{n-1},s}(\sigma_n * 0)$ and the node $T_{\mu_{n-1},s}(\sigma_n * 1)$ is active. (If $n = 0$ and the verification procedure was called by a P strategy μ_{-1}, then we need to replace $T_{\mu_{-1},s}$ by $T_{\mu'_{-1},s}$. Similar comments apply throughout the rest of this procedure. If $n \geq 1$, then μ_{n-1} is an R strategy, so no such replacement is necessary.) Furthermore, if $n \geq 1$ and $t < s$ is the stage at which the $(n-1)^{\text{st}}$ cycle started, then $T_{\mu_{n-1},s}(\sigma_n) = T_{\mu_{n-1},t}(\sigma_n)$ and $T_{\mu_{n-1},t}(\sigma_n * i) \subseteq T_{\mu_{n-1},s}(\sigma_n * i)$ for $i = 0, 1$. During the $(n-1)^{\text{st}}$ cycle, we defined $\Gamma^{T_{\mu_{n-1},t}(\sigma_n * 0)}(x_\eta) = 0$. If $n = 0$, then we have already defined $\Gamma^{T_{\mu_{-1},s}(\sigma_0 * 0)}(x_\eta) = 0$. (We verify all of these properties after the construction.)

Move the current path from $T_{\mu_{n-1},s}(\sigma_n * 0)$ to be the rightmost active path through $T_{\mu_{n-1},s}(\sigma_n * 1)$. If $n = 0$, then declare $T_{\mu_{-1},s}(\sigma_0 * 0)$ to be η frozen and if $n \geq 1$, then declare $T_{\mu_{n-1},t}(\sigma_n * 0)$ to be η frozen. (That is, we freeze the string that was used in the Γ definition on x_η.) For strategies $\mu \subsetneq \mu_{n-1}$, redefine the trees by stretching. For each R strategy μ such that $\mu * L \subseteq \mu_{n-1}$, define

X_μ to be the finite set of numbers x such that μ has seen $[\mu]^\gamma(x)$ converge for some γ on $T_{\mu,s}$ such that $T_{\mu,s}(\beta_{\mu,L} * 0) \subseteq \gamma$, $U(\gamma) = G_\mu * L$ and μ has not seen $[\mu]^{T_{\mu,s}(\beta_{\mu,L})}(x)$ converge. ($\beta_{\mu,L}$ is defined by the stretching process.) If all the X_μ sets are empty, then the verification procedure is complete and we return to the action of the strategy that called the verification procedure.

If some $X_\mu \neq \emptyset$, then set μ_n to be the lowest priority strategy such that $X_\mu \neq \emptyset$. (After the construction, we verify that $\mu_n \subsetneq \mu_{n-1}$.) Let σ_{n+1} denote the node such that $T_{\mu_n,s}(\sigma_{n+1})$ was redefined to be equal to $T_{\lambda'',s}(\delta)$ by the stretching procedure in the previous paragraph. (That is, $T_{\mu_n,s}(\sigma_{n+1})$ is the least node along the new current path in $T_{\mu_n,s}$ which was stretched.) Because of the stretching, the length of $T_{\eta,s}(\sigma_{n+1})$ is large, the current path goes through $T_{\mu_n,s}(\sigma_{n+1} * 0)$ and $T_{\mu_n,s}(\sigma_{n+1} * 1)$ is active. Define $\Gamma^{T_{\mu_n,s}(\sigma_{n+1}*0)}(x_\eta) = 0$.

Place a link from μ_n to η. For all ν such that $\nu * L \subseteq \mu_n * L$, challenge ν low and pass $\beta_{\nu,L}$ and X_ν to ν. For all ν such that $\nu * H \subseteq \mu_n$, challenge ν high, pass $\beta_{\nu,H}$ to ν and set the witness $x_\nu = x_\eta$. ($\beta_{\nu,H}$ was defined by the stretching process above.) If η is an R strategy, initialize all strategies of lower priority than $\eta * L$ including $\eta * L$. If η is a P strategy, then initialize all lower priority strategies. End the stage. When η is next eligible to act, we begin the $(n+1)^{\text{st}}$ cycle of the verification procedure and check if the verification procedure is now complete or if we need to go through the whole $(n+1)^{\text{st}}$ cycle.

This completes the description of the construction. Before we begin the sequence of lemmas to prove the construction succeeds, we point out several features of the construction which the reader can check by observation. First, the places where we may find new high splittings are Case 3, Subcases 4A(i) and 4B, and Case 5 of an R strategy. In Cases 3, 4A(i) and 5, one half of the new high split is already on the current path. In Subcase 4B, we explicitly move the current path so that one half of the new high split (namely $\hat{\tau}$) lies on the new current path. Therefore, the only time the current path moves is when we explicitly move it. (That is, we are not in the typical situation of a full approximation argument in which the current approximation to the set being constructed is defined to be the rightmost path through the tree. In that setting, the current approximation is implicitly changed by the addition of new high splits.)

Second, the movement of the current path is only caused by a verification procedure or by a high challenged R strategy acting in Subcase 4A(ii) or 4B. Whenever we explicitly move the current path in one of these cases, we also stretch nodes along the new current path. Furthermore, these are the only times when we stretch nodes.

Third, if a node becomes frozen at a stage s, then some strategy must have moved the current path below this node. This property follows because the only time nodes are frozen is in Subcase 4A(ii) of a high challenge and in a verification procedure.

Fourth, links are only established by a verification procedure and these procedures are only called by P strategies acting in Case 3 of the P action and by high challenged R strategies acting in Subcase 4B of a high challenge.

Finally, the only time new challenges are issued is by a verification procedure or by a high challenged R strategy acting in Subcase 4A(ii). In either of these cases, the strategy issuing the new challenges ends the current stage. This fact implies that at any given stage, at most one strategy can issue new challenges.

We say that the current path *moves below a node* $T_{\eta,s}(\alpha)$ if there is a string $\beta \subseteq \alpha$ such that either $T_{\eta,s}(\beta) \subseteq A_{\eta,s}$ but $T_{\eta,s}(\beta) \not\subseteq A_{\mu,t}$, or $T_{\eta,s}(\beta) \not\subseteq A_{\eta,s}$ but $T_{\eta,s}(\beta) \subseteq A_{\mu,t}$ for some strategy μ and stage $t \geq s$ (with $\eta \subseteq \mu$ if $t = s$). We say that the current path *moves below level* l *of* $T_{\eta,s}$ if the current path moves below $T_{\eta,s}(\alpha)$ for some string α of length l.

We present the series of lemmas to prove that our construction succeeds. We begin with some terminology and properties of the links. If there is a link between strategies ν and $\hat{\nu}$ such that $\nu \subsetneq \mu \subsetneq \hat{\nu}$, we say that the link *jumps over* μ. If $\mu * L \subseteq \hat{\nu}$, then we say the link *lands above* $\mu * L$. If $\mu * H \subseteq \hat{\nu}$, then we say the link *lands above* $\mu * H$. The idea is that a link which jumps over μ and lands above $\mu * L$ (or $\mu * H$) gives a way for a strategy extending $\mu * L$ (or $\mu * H$) to be eligible to act without μ acting. The following lemma says that if μ is low challenged, then there cannot be a link jumping over μ and landing above $\mu * L$.

Lemma 3.1. *The following situation cannot occur at any stage: μ has been challenged low by $\hat{\mu}$ and there is a link from ν to $\hat{\nu}$ such that $\nu \subsetneq \mu$ and $\mu * L \subseteq \hat{\nu}$.*

Proof. Because μ is challenged low by $\hat{\mu}$, we have $\mu * L \subseteq \hat{\mu}$. Because the link between ν and $\hat{\nu}$ can only be established when $\hat{\nu}$ challenges ν low, we have $\nu * L \subseteq \hat{\nu}$. Furthermore, $\nu \subsetneq \mu \subseteq \hat{\nu}$ and $\nu * L \subsetneq \hat{\nu}$ together imply that $\nu * L \subseteq \mu$ and hence $\nu * L \subseteq \hat{\mu}$.

For a contradiction, assume that $\hat{\mu}$ challenges μ low at stage s and before this low challenge is removed (either by being met or by $\hat{\mu}$ being initialized) there is a link between ν and $\hat{\nu}$ (which may already be present at stage s). Furthermore, we can assume without loss of generality that μ is such that no strategy $\eta \subsetneq \mu$ is ever in the situation of being challenged low with a link jumping over η and landing above $\eta * L$. (If there were such an η, we consider it instead of μ.) In particular, there is never a situation in which ν is challenged low with a link jumping over ν and landing above $\nu * L$. We will refer to this assumption as our wlog assumption about ν. (This assumption is really about μ but we will only apply it in this special case concerning $\nu \subsetneq \mu$.)

First, we show that this situation cannot occur if $\hat{\nu} \neq \hat{\mu}$. Consider when the link from ν to $\hat{\nu}$ is established. It cannot have been established at stage s since at any given stage, at most one strategy issues new low challenges. Since we assume $\hat{\mu}$ challenges μ at stage s and $\hat{\nu} \neq \hat{\mu}$, we cannot also have $\hat{\nu}$ issuing low challenges and establishing a link at stage s.

Assume that the link from ν to $\hat{\nu}$ is established at $u < s$ and hence ν is challenged low by $\hat{\nu}$ at stage $u < s$. In this case, consider how $\hat{\mu}$ comes to be eligible to act at stage s. If s is a ν stage, then the only possible outcomes for ν are $\nu * H$ and $\nu * N$ since ν cannot meet its low challenge at s without following (and hence removing) the link. Because $\nu * L \subseteq \hat{\mu}$, there must be a link jumping over ν and landing above $\nu * L$ at stage s while ν remains low challenged. However, this contradicts our wlog assumption about ν.

Assume that the link from ν to $\hat{\nu}$ is established at $u > s$ and that u is the first stage at which a link jumping over μ and landing above $\mu * L$ is established. Because u is a $\hat{\nu}$ stage and there is no link already jumping over μ and landing above $\mu * L$, u must also be a μ stage. However, this is impossible since the only possible outcomes for μ are $\mu * H$ and $\mu * N$ unless μ meets the low challenge issued by $\hat{\mu}$ to μ at stage s. This completes the proof that we cannot have $\hat{\nu} \neq \hat{\mu}$.

Second, we show that we cannot have $\hat{\mu} = \hat{\nu}$. Assume $\hat{\mu} = \hat{\nu}$. Then $\hat{\mu}$ must issue the low challenges to both ν and μ. Consider when $\hat{\mu}$ issues the low challenge to ν and establishes the link from ν to $\hat{\nu} = \hat{\mu}$.

Assume the link from ν to $\hat{\mu}$ is established before stage s. In this case, by our wlog assumption about ν, there cannot be a link jumping over ν and landing above $\nu * L$ at stage s. Therefore, since s is a $\hat{\mu}$ stage and $\nu * L \subseteq \hat{\mu}$, s must also be a ν stage. At stage s, ν either takes outcome $\nu * H$ or $\nu * N$ (in which case $\hat{\mu}$ cannot act at stage s) or ν follows the link to $\hat{\mu}$ (in which case the link is removed before $\hat{\mu}$ challenges μ low). All cases lead to a contradiction.

Assume the link from ν to $\hat{\mu}$ is established at stage s. Then ν must be the lowest priority strategy such that $\hat{\mu}$ calculates $X_\nu \neq \emptyset$. Then $\hat{\mu}$ only challenges a strategy γ low at stage s if $\gamma * L \subseteq \hat{\mu}$ and $\gamma \subseteq \nu$. This contradicts the fact that $\hat{\mu}$ challenges μ low at stage s since $\nu \subsetneq \mu$.

Assume the link from ν to $\hat{\mu}$ is established at stage $t > s$ and t is the first stage after s at which such a link is established. t must be a $\hat{\mu}$ stage. If t is a μ stage, then either we take outcome $\mu * H$ or $\mu * N$ (which contradicts the fact that t is a $\hat{\mu}$ stage) or we follow the link from μ to $\hat{\mu}$ and remove the low challenge to μ (which contradicts the fact that μ is still low challenged when the link from ν to $\hat{\nu}$ is established). Therefore, t cannot be a μ stage and so there must be a link jumping over μ and landing above $\mu * L$ established before stage t by some strategy other than $\hat{\mu}$. In the first case, we showed that this situation is impossible. □

A case analysis similar to the one for Lemma 3.1 proves the following lemma.

Lemma 3.2. *If μ is challenged high, then there cannot be a link jumping over μ and landing above $\mu * H$.*

Lemma 3.3. *If η is challenged low, then no strategy μ with $\eta * L \subseteq \mu$ is eligible to act until the low challenge has been met or is cancelled by initialization.*

Proof. Assume that η is challenged low by $\hat{\eta}$ at stage s (and hence $\eta * L \subseteq \hat{\eta}$). At every η stage until the low challenge is met, η takes either outcome $\eta * H$ (which causes $\hat{\eta}$ to be initialized and the low challenge to be removed) or outcome $\eta * N$. Therefore, the only way for a strategy μ with $\eta * L \subseteq \mu$ to be eligible to act while η remains low challenged is to have a link jumping over η and landing above $\eta * L$. Such a link contradicts Lemma 3.1. □

Lemma 3.4. *A strategy μ can be challenged low by at most one strategy at a time.*

Proof. Assume that μ is challenged low by $\hat{\mu}$ at stage s. The only strategies $\hat{\nu}$ which can challenge μ low satisfy $\mu * L \subseteq \hat{\nu}$. By Lemma 3.3, no such strategy is eligible to act after stage s and before the low challenge issued by $\hat{\mu}$ is met or cancelled by initialization. □

Essentially the same proofs as for Lemmas 3.3 and 3.4 establish the following two lemmas.

Lemma 3.5. *If η is challenged high by $\hat{\eta}$, then no strategy μ with $\eta * H \subseteq \mu$ is eligible to act until the high challenge has been met or is cancelled by initialization.*

Lemma 3.6. *A strategy μ can be challenged high by at most one strategy at a time.*

It is possible for a strategy η to be challenged both high and low at the same time. However, if η is challenged high at stage s_0 by $\hat{\eta}$, then $\eta * H \subseteq \hat{\eta}$ so any low challenges to η issued before stage s_0 are removed by initialization at stage s_0. (Also, there is no link jumping over η and landing above $\eta * L$ at the end of stage s_0.) As long as η acts in Subcase 4A of the high challenge and fails to find a potential split, it takes outcome $\eta * L$. A strategy μ with $\eta * L \subseteq \mu$ could challenge η low. Suppose this happens at stage $s_1 > s_0$. At s_1, η must still be acting in Subcase 4A of the high challenge and not finding a potential high split. If η ever finds such a potential high split, then it acts either in Subcase 4A(i) or 4A(ii). In either of these cases, μ (which issued the low challenge to η) will be initialized. Furthermore, if η continues to act in Subcase 4B of the high challenge, then it does not take outcome $\eta * L$ and hence cannot be challenged low again until it is either initialized or meets its high challenge. The conclusion of this observation is that η can only be both high and low challenged if the high challenge comes first and the low challenge comes while η is still acting in Subcase 4A of the high challenge and has not yet found a potential high split. Therefore, in our construction, we gave all the necessary instructions for handling a strategy which is both high and low challenged.

Lemma 3.7. *If η calls a verification procedure, no strategy μ with $\eta \subsetneq \mu$ is eligible to act until the verification procedure is met or is cancelled by initialization.*

Proof. Assume that η calls a verification procedure at stage s. η will end every stage after s at which it is eligible to act until it is either initialized or the verification procedure is met. Therefore, it suffices to show that there are no links jumping over η at the end of stage s. If η is a P strategy, then η initializes all lower priority requirements at stage s and hence there are no links jumping over η at the end of stage s.

If η is an R strategy, then η must be acting in Subcase 4B of a high challenge and the verification procedure called by η initializes all strategies below $\eta * L$ at s. Therefore it suffices to show that there is no link at stage s between strategies ν and $\hat{\nu}$ where $\nu * L \subseteq \eta$ and $\eta * H \subseteq \hat{\nu}$. Suppose there is such a link. Since η ends stage s and does not take outcome $\eta * H$ until after the verification procedure for the high challenge is met, the link must have been established before stage s. This means that ν is low challenged by $\hat{\nu}$ before stage s. Consider how η is eligible to act at stage s. There cannot be a link jumping over ν and landing above $\nu * L$ at stage s by Lemma 3.1, so s must be a ν stage. ν either takes outcome $\nu * H$ or $\nu * N$ (contradicting the fact that s is an η stage) or η meets the low challenge and follows the link which jumps over η (again contradiction the fact that s is an η stage). □

Lemma 3.8. *If η is challenged high, then this high challenge is part of a series of high challenges started by some P strategy $\hat{\eta}$. Furthermore, if η moves the current path from $T_{\eta,s}(\gamma * 0)$ to $T_{\eta,s}(\gamma * 1)$ or from $T_{\eta,s}(\gamma * 1)$ to $T_{\eta,s}(\gamma * 0)$ during this series of challenges as part of either Subcase 4A(ii) or Subcase 4B (including any verification procedures called by this subcase) of the high challenge, then $|\gamma| > p_{\hat{\eta}}$.*

Proof. Suppose that η is challenged high by η_0 at s_0, so $\eta * H \subseteq \eta_0$. If η_0 is a P strategy, then $\hat{\eta} = \eta_0$. Otherwise, η_0 is an R strategy which is challenging η high as part of its own high challenge. Therefore, η_0 must have been high challenged by some η_1 at $s_1 < s_0$, so $\eta_0 * H \subseteq \eta_1$ and hence $\eta * H \subseteq \eta_1$. If η_1 is a P strategy, then $\hat{\eta} = \eta_1$. Otherwise, we repeat the argument just given. It is clear that tracing this sequence of high challenges back in time must yield a P strategy $\hat{\eta} = \eta_n$ such that $\eta * H \subseteq \hat{\eta}$ and $\hat{\eta}$ issued its original challenges at stage s_n.

When $\hat{\eta}$ issues its challenges at stage s_n, it moves the current path from $T_{\hat{\eta}',s_n}(\alpha_{\hat{\eta}} * 0)$ to $T_{\hat{\eta}',s_n}(\alpha_{\hat{\eta}} * 1)$. The string $\alpha_{\hat{\eta}}$ has length $p_{\hat{\eta}}$. Therefore, for any R strategy $\mu \subseteq \hat{\eta}$, if γ_μ is such that $T_{\mu,s_n}(\gamma_\mu) = T_{\hat{\eta}',s_n}(\alpha_{\hat{\eta}})$, then $|\gamma_\eta| > p_{\hat{\eta}}$. Also, if μ (with $\mu * H \subseteq \hat{\eta}$) is high challenged during the sequence of high challenges initiated by the action of $\hat{\eta}$ and μ moves the current path at stage $s > s_n$ due to its action in Subcase 4A(ii) or Subcase 4B of the high challenge, then this movement occurs above the place where $\hat{\eta}$ originally moved the path. The statement of the lemma follows. □

Lemma 3.9. *Let η be a strategy such that η defines p_η at stage t. Unless η is initialized, the current path cannot move below level $p_\eta + 1$ of the tree defined by η' (if η is a P strategy) or by η'' (if η is an R strategy) before η defines α_η.*

Proof. The analysis is the same regardless of whether η is a P or R strategy, with only a change in notation between whether η works on the tree built by η' or η''. Rather than repeating the argument twice, we give the proof in the case when η is a P strategy.

Assume that no strategy initializes η after stage t and before η defines α_η. Since no strategy to the left of η in the tree of strategies can act without initializing η, we can assume no such strategy moves the current path before η defines α_η. At stage t, η initializes all strategies of lower priority, hence these strategies work at or above level $p_\eta + 1$ in the tree defined by η' and cannot move the current path below level $p_\eta + 1$ of the tree defined by η'. Furthermore, by Lemma 3.8, no R strategy $\nu \subseteq \eta$ can move the path below this level because of a series of challenges started by a P strategy of lower priority than η. We are left to consider the other possible actions of strategies ν such that $\nu \subseteq \eta$ at the stages before η defines α_η.

We split the proof into two cases based on the ways that the current path can be moved after t and before η defines α_η. First, the current path could be moved by a P strategy $\nu \subseteq \eta$ which calls a verification procedure in Case 3 of the P action. In this case, ν initializes all lower priority strategies including η contrary to our assumption.

Second, the current path could be moved by a high challenged R strategy $\nu \subseteq \eta$ acting in Subcase 4A(ii) or 4B of the high challenge (including the verification procedure called by Subcase 4B). Let $\hat{\nu}$ denote the P strategy which called the verification procedure starting the sequence of high challenges that led to this high challenge to ν. As mentioned above, $\hat{\nu}$ must have higher priority than η, so either $\hat{\nu} \subseteq \eta$ or $\hat{\nu} <_L \eta$. If $\hat{\nu}$ starts this sequence of challenges at a stage $\geq t$, then η is initialized when $\hat{\nu}$ acts contrary to our assumption.

If $\hat{\nu}$ starts the sequence of challenges at a stage $< t$, the since $\hat{\nu}$ has not completed its verification procedure, we must have $\hat{\nu} <_L \eta$ by Lemma 3.7. Because a high challenged strategy in this sequence of high challenges only moves the current path when it issues new high challenges in Subcase 4A(ii) or 4B of the high challenge, we can assume that ν is already high challenged at stage t. (Otherwise, tracing backwards in time from the stage at which ν is high challenged after t, we can find an R strategy which is high challenged at stage t in this sequence of high challenges and which later moves the current path to issue new high challenges to continue this sequence leading to the high challenge of ν. We work with this strategy instead.) We must have either $\nu * H \subseteq \eta$ or $\nu * H <_L \eta$. If $\nu * H \subseteq \eta$, then by Lemma 3.5, η is not eligible to act until the high challenge is met or removed by initialization, so η is not eligible to act at stage t contrary to our assumption. If $\nu * H <_L \eta$, then η has lower priority than $\nu * L$ and hence is initialized when ν moves the current path by acting in Subcase 4A(ii) or 4B of the high challenge contrary to our assumption. □

Lemma 3.10. *Assume a P strategy η defines α_η at stage s. Then $T_{\eta',s}(\alpha_\eta)$, $T_{\eta',s}(\alpha_\eta * 0)$ and $T_{\eta',s}(\alpha_\eta * 1)$ are all active at stage s and the current path runs through $T_{\eta',s}(\alpha_\eta * 0)$. If η is an R strategy that defines α_η at stage s, then the same statement is true when η'' is substituted for η'.*

Proof. As in the proof of Lemma 3.9, we give the proof in the case when η is a P strategy. Let $t < s$ be the stage such that η defined p_η at t and η is not initialized between defining p_η at t and defining α_η at s. Let α be the string such that $|\alpha| = p_\eta$ and $T_{\eta,t}(\alpha) \subseteq A_{\eta',t}$. Because p_η is defined large and $T_{\eta',t}(\alpha)$ is active (as it is on the current path), $T_{\eta,t}(\alpha * 0) \subseteq A_{\eta',t}$ and both $T_{\eta,t}(\alpha_\eta * 0)$ and $T_{\eta,t}(\alpha_\eta * 1)$ are active. By Lemma 3.9, the current path does not change below level $p_\eta + 1$ in the tree defined by η' between stages t and s. Therefore, when η defines α_η, we still have $T_{\eta,s}(\alpha) \subseteq A_{\eta',s}$ and hence $\alpha_\eta = \alpha$. Furthermore, $T_{\eta',s}(\alpha * 0) = T_{\eta',s}(\alpha_\eta * 0)$ is still on the current path (and hence is still active) and $T_{\eta',s}(\alpha * 1) = T_{\eta',s}(\alpha_\eta * 1)$ is still active (because nodes can only become inactive when the current path moves below them). □

The analysis given in Lemma 3.9 can be applied in a more general context. We say that a node $T_{\eta,s}(\alpha)$ *effects initialization* if any number defined to be large after $T_{\eta,s}(\alpha)$ is defined has to be larger than the length of $T_{\eta,s}(\alpha)$. That is, either $T_{\eta,s}(\alpha)$ (or any longer node) has been used as an oracle for a computation viewed in the construction or some parameter has been defined which is larger than $T_{\eta,s}(\alpha)$. We will only apply Lemmas 3.11 and 3.12 in situations in which α is equal to some parameter in the construction such as α_η or $\beta_{\eta,H}$.

Lemma 3.11. *Let η be an R strategy, s be an η stage and α be a string such that $T_{\eta,s}(\alpha)$ is defined and effects initialization. For each ν such that $\nu * H \subseteq \eta$, let γ_ν be such that $T_{\nu,s}(\gamma_\nu) = T_{\eta,s}(\alpha)$. Assume that for all $\gamma \subsetneq \gamma_\nu$, $T_{\nu,s}(\gamma)$ is high ν splitting. Then, for all η stages $u \geq s$, $T_{\eta,u}(\alpha) = T_{\eta,s}(\alpha)$ unless η is initialized, η finds a new high split below $T_{\eta,s}(\alpha)$ or some strategy μ such that $\eta \subseteq \mu$ moves the current path below $T_{\eta,s}(\alpha)$ at a stage t such that $s \leq t < u$. Furthermore, if $T_{\eta,s}(\alpha) \subseteq A_{\eta,s}$, then $T_{\eta,s}(\alpha)$ remains on the current path unless η is initialized or some strategy μ such that $\eta \subseteq \mu$ moves the current path below $T_{\eta,s}(\alpha)$ at a stage t such that $s \leq t$.*

Proof. Unless η is initialized, the value of $T_{\eta,s}(\alpha)$ can only change if some R strategy $\mu \subseteq \eta$ finds a new high split below $T_{\eta,s}(\alpha)$ at a future stage or if $T_{\eta,s}(\alpha)$ changes values due to stretching. By the hypotheses, no strategy $\nu \subsetneq \eta$ can find a new high split below this node without moving the path in the tree of strategies to the left of η and initializing η. Therefore, only η can change the value of this node by finding a new high split. The value of the node can only be changed by stretching if the current path moves below this node. Hence, we can finish the proof by giving an analysis of which strategies μ can move the current path below this node without initializing η. This analysis is similar to the one given in the proof of Lemma 3.9.

First, if $\mu <_L \eta$, then μ cannot act without initializing η, so we can assume no such strategy moves the current path below $T_{\eta,s}(\alpha)$. Second, if $\eta <_L \mu$, then μ is initialized at stage s, so it works higher on the trees than $T_{\eta,s}(\alpha)$ at future stages. Therefore, no such strategy can cause the path to move below $T_{\eta,s}(\alpha)$ and by Lemma 3.8, no R strategy $\nu \subsetneq \eta$ can cause the current path to move below $T_{\eta,s}(\alpha)$ because of a series of high challenges initiated by μ such that $\eta <_L \mu$.

Third, suppose $\mu \subsetneq \eta$ moves the current path below $T_{\eta,s}(\alpha)$ at a stage $t > s$. Let $\hat{\mu}$ denote the P strategy which initiates the series of challenges leading to μ moving the current path. (As noted at the end of the previous paragraph, we know that $\hat{\mu}$ is not to the right of η in the tree of strategies.) If $\hat{\mu} \subseteq \eta$, then because s is an η stage, Lemma 3.7 implies that $\hat{\mu}$ must initiate this series of challenges after stage s. However, in this case, $\hat{\mu}$ initializes η when it calls its verification procedure to initiate the series of challenges. If $\hat{\mu} <_L \eta$, then $\hat{\mu}$ must initiate its series of challenges before stage s and as in the proof of Lemma 3.9, we can assume that μ is challenged high at stage s. We split into the cases when

$\mu * H \subseteq \eta$ and when $\mu * H <_L \eta$. In the first case, Lemma 3.5 contradicts the fact that s is an η stage. In the second case, η has lower priority than $\mu * L$ and hence is initialized when μ moves the current path in either Subcase 4A(ii) or 4B of the high challenge.

We now know that we cannot have $\eta <_L \hat{\mu}$, $\hat{\mu} \subseteq \eta$ or $\hat{\mu} <_L \eta$. It remains to consider the case when $\eta \subsetneq \hat{\mu}$. If $\hat{\mu}$ issues its challenges after stage s, then $\hat{\mu}$ moves the current path after stage s when it issues these challenges (and before μ moves the current path). Therefore, we have met the conditions of the lemma in this case. Otherwise, $\hat{\mu}$ calls its verification procedure and issues its first challenges before stage s. In this case, since μ is high challenged in the series of challenges started by $\hat{\mu}$, we have $\mu * H \subsetneq \hat{\mu}$. Together with the case assumption that $\mu \subsetneq \eta \subseteq \hat{\mu}$, we have $\mu * H \subseteq \eta$. Since s is an η stage, μ cannot be high challenged at stage s by Lemma 3.5. We can assume that μ is the first strategy such that $\mu \subsetneq \eta$ to move the current path below $T_{\eta,s}(\alpha_\eta)$ after stage s. There must be a ν such that ν is high challenged at s (in the series started by $\hat{\mu}$) and such that ν issues high challenges after stage s which lead to the high challenge of μ. By the comments above, we know that $\eta \subseteq \nu$. Therefore, when ν issues its high challenges after stage s (and before μ moves the current path), ν moves the current path below $T_{\eta,s}(\alpha_\eta)$. Therefore, the conditions of the lemma are true in this case as well. □

Lemma 3.12. *Let η be an R strategy, s be an η stage and α be a string such that $T_{\eta,s}(\alpha)$ is defined, effects initialization, has η'' state G_η and may or may not be η high splitting. For all η stages $u \geq s$, $T_{\eta,u}(\alpha) = T_{\eta,s}(\alpha)$ unless η is initialized, η finds a new high split below $T_{\eta,s}(\alpha)$ or some strategy μ such that $\eta \subseteq \mu$ moves the current path below $T_{\eta,s}(\alpha)$ at a stage t such that $s \leq t < u$. Furthermore, if $T_{\eta,s}(\alpha) \subseteq A_{\eta,s}$, then $T_{\eta,s}(\alpha)$ remains on the current path unless η is initialized or some strategy μ such that $\eta \subseteq \mu$ moves the current path below $T_{\eta,s}(\alpha)$ at a stage t such that $s \leq t$.*

Proof. This lemma follows immediately from Lemma 3.11. □

Lemma 3.13. *Assume that an R strategy η defines α_η at stage t. Unless η is initialized, $T_{\eta'',u}(\alpha_\eta) = T_{\eta'',t}(\alpha_\eta) \subseteq A_{\eta'',u}$ for all η stages $u > t$.*

Proof. When η defines α_η at stage t, we have $U(T_{\eta'',t}(\alpha_\eta)) = G_\eta$. We apply Lemma 3.12 to this node to show that it cannot change after stage t unless η is initialized. By Lemma 3.12, the only R strategy which could change the value of this node by finding a new high splitting is η''. However, if $\eta'' * H \subseteq \eta$, then this node is already η'' high splitting as are the nodes below it on $T_{\eta'',t}$. If $\eta'' * H <_L \eta$, then η is initialized when η'' finds a new high split below this node. Therefore, unless η is initialized, the value of $T_{\eta'',t}(\alpha_\eta)$ does not change due to finding a new high splitting.

Next, we consider how $T_{\eta'',t}(\alpha_\eta)$ could change values after t because of stretching. If this nodes changes values because of stretching, then the current path must move below it. Therefore, we can finish the proof by showing that the current path cannot be moved below $T_{\eta'',t}(\alpha_\eta)$ without initializing η.

By Lemma 3.12, unless η'' (and hence η) is initialized or a strategy μ with $\eta'' \subseteq \mu$ moves the current path below $T_{\eta'',t}(\alpha_\eta)$, $T_{\eta'',t}(\alpha_\eta)$ remains on the current path. At stage t, η initializes all lower priority strategies, so each strategy μ such that $\eta \subsetneq \mu$ works with strings which are too long to move the current path below $T_{\eta'',t}(\alpha_\eta)$. If η moves the current path, then it does so above $T_{\eta'',t}(\alpha_\eta)$ (since η defines $T_{\eta,t}(\lambda) = T_{\eta'',s}(\alpha_\eta)$ and η only moves the current path on its own tree) and not below $T_{\eta'',s}(\alpha_\eta)$. If η' moves the current path, then because η' is a P strategy, it initializes η.

It remains to consider the case when η'' moves the current path below $T_{\eta'',t}(\alpha_\eta)$ after stage t. Suppose η'' moves the current path after stage t because it is high challenged in a series of challenges started by some P strategy $\hat{\mu}$ with $\eta'' * H \subseteq \hat{\mu}$. If the high challenge issued to η'' occurs before stage t, then $\eta'' * H <_L \eta$ by Lemma 3.5 and the fact that t is an η stage. Therefore, η is initialized when η'' moves the current path as part of its high challenge. If the high challenge is issued after stage t, then we break into cases depending on whether $\eta \subsetneq \hat{\mu}$ or $\hat{\mu} = \eta'$. (Since $\hat{\mu}$ is a P strategy and $\eta'' \subseteq \hat{\mu}$, these are the only possibilities.) In the former case, the path is moved above $T_{\eta'',t}(\alpha_\eta)$ and in the later case, η is initialized when $\hat{\mu}$ initiates the series of challenges by calling a verification procedure. □

Lemma 3.14. *Assume that a P strategy η defines α_η at stage t.*

1. *Unless η is initialized, $T_{\eta',u}(\alpha_\eta) = T_{\eta',t}(\alpha_\eta) \subseteq A_{\eta,u}$ for all η stages $u \geq t$.*
2. *Unless η is initialized or calls a verification procedure, $T_{\eta',u}(\alpha_\eta * i) = T_{\eta',t}(\alpha_\eta * i)$ for $i = 0, 1$ and these nodes remain active at all η' stages $u \geq t$ and $T_{\eta,u}(\alpha_\eta * 0) \subseteq A_{\eta,u}$.*

Proof. We first establish Property 1. Because $U(T_{\eta',t}(\alpha_\eta)) = G_\eta$, we can apply Lemma 3.12 to $T_{\eta',t}(\alpha_\eta)$. The value of this node can only change if η' is initialized, if η' finds a new high split below this node, or if some strategy μ such that $\eta' \subseteq \mu$ moves the current path below this node. We consider each of these cases separately.

First, if η' is initialized, then so is η. Second, assume that η' finds a new high split below $T_{\eta',t}(\alpha_\eta)$ after stage t. $T_{\eta',t}(\alpha_\eta)$ must not be η' high splitting at stage t, so because $U(T_{\eta',t}(\alpha_\eta)) = G_\eta$, we must have $\eta' * L \subseteq \eta$ or $\eta' * N \subseteq \eta$. Therefore, η is initialized when η' finds the new high split. Third, assume that some μ with $\eta' \subseteq \mu$ moves the current path below $T_{\eta',t}(\alpha_\eta)$. Because η initializes all lower priority strategies at stage t, μ must be equal to either η or η'. (If μ is to the left of η, then η would be initialized when μ acts to move the current path.) Suppose $\mu = \eta$. In this case, μ only moves the current path above $T_{\eta',t}(\alpha_\eta)$. Suppose $\mu = \eta'$. In this case, since η' is an R strategy, it only moves the current path during a high challenge. Suppose $\hat{\eta}$ issues the high challenge to η', so $\eta' * H \subseteq \hat{\eta}$. If $\eta' * H$ is to the left of η, then η is initialized when η' moves the current path. If $\eta' * H = \eta$, then η initialized $\hat{\eta}$ at stage t and hence any movement in the current path caused by a series of challenges initialized by $\hat{\eta}$ is above $T_{\eta',t}(\alpha_\eta)$. This completes the proof of Property 1.

To establish Property 2, we cannot necessarily apply Lemma 3.12 since we don't know what the states of $T_{\eta',t}(\alpha_\eta * i)$ are. However, we claim that we can use Lemma 3.11. To see this fact, we split into two cases. If there is no strategy ν such that $\nu * H \subseteq \eta$, then we can apply Lemma 3.12 (since G_η contains all low states) and the argument is just as before. Otherwise, fix ν to be the lowest priority strategy such that $\nu * H \subseteq \eta$ and let γ_ν be such that $T_{\nu,t}(\gamma_\nu) = T_{\eta',t}(\alpha_\eta)$. Since $T_{\nu,t}(\gamma_\nu)$ is high ν splitting and none of the strategies between ν and η are in the high state, we have $T_{\eta',t}(\alpha_\eta * i) = T_{\nu,t}(\gamma_\nu * i)$. Since the ν state of $T_{\nu,t}(\gamma_\nu)$ is $G_\nu * H$, we have the hypotheses for Lemma 3.11. The rest of the proof of Property 2 is a similar case analysis to the analysis in the proof of Property 1, except we use Lemma 3.11 in place of Lemma 3.12. □

We now consider the action of strategies which are high challenged or which call a verification procedure. Let η be a strategy and s be a stage such that η is either challenged high at s or η begins a verification procedure at stage s. Assume that η is not initialized before the challenge or verification is met (if it is ever met) and that every strategy $\nu * L \subseteq \eta$ (or $\nu * H \subseteq \eta$) which is low (respectively high) challenged eventually meets its challenge. Furthermore, assume that η is eligible to act infinitely often after stage s (or at least until the challenge is met or the verification is complete). We prove the following two lemmas simultaneously by induction on the length of η under these conditions.

Lemma 3.15. *Let η be a strategy that calls a verification procedure at stage s under these conditions. Let t_0 be the stage at which η calls its verification procedure with σ_0 and let t_n denote the stage at which we return to the verification procedure for the n^{th} time (and start the n^{th} cycle). In the following two properties, we work with the notation σ_n and μ_n as in the description of a verification procedure, we set $\mu_{-1} = \eta$ and we work with the notation as though η is an R strategy. (If η is a P strategy, we need to replace $T_{\mu_{-1}}$ by $T_{\mu'_{-1}}$ and $G_{\mu_{-1}} * L$ by $G_{\mu_{-1}}$.)*

1. *When the verification procedure is called at stage t_0, we have $T_{\mu_{-1},t_0}(\sigma_0 * 0) \subseteq A_{\mu_{-1},t_0}$, $T_{\mu_{-1},t_0}(\sigma_0 * 1)$ is active, $\Gamma^{T_{\mu_{-1},t_0}(\sigma_0 * 0)}(x_\eta) = 0$ and $U(T_{\mu_{-1},t_0}(\sigma_0)) = G_{\mu_{-1}} * L$.*

2. *For $n \geq 1$, when we follow the link from μ_{n-1} to η at stage t_n and begin the n^{th} cycle, we have the following properties: $T_{\mu_{n-1},t_{n-1}}(\sigma_n) = T_{\mu_{n-1},t_n}(\sigma_n)$, $U(T_{\mu_{n-1},t_n}(\sigma_n)) = G_{\mu_{n-1}} * L$, $T_{\mu_{n-1},t_{n-1}}(\sigma_n * i) \subseteq T_{\mu_{n-1},t_n}(\sigma_n * i)$ for $i = 0, 1$, $T_{\mu_{n-1},t_n}(\sigma_n * 0) \subseteq A_{\mu_{n-1},t_n}$ and $T_{\mu_{n-1},t_n}(\sigma_n * 1)$ is active.*

Furthermore, there are only finitely many cycles before the verification procedure is complete. When the verification procedure is complete, all the strings γ such that the verification procedure defined $\Gamma^\gamma(x_\eta) = 0$ are currently η frozen.

Lemma 3.16. *Assume that η is high challenged at stage s under the conditions given above.*

1. *Unless η is initialized or meets its challenge, $T_{\eta,s}(\beta_{\eta,H})$ remains the same and on the current path at future η stages.*

2. *At the first η stage $s_0 > s$, $U(T_{\eta,s_0}(\beta_{\eta,H})) = G_\eta * L$ and $T_{\eta,s}(\beta_{\eta,H} * i) \subseteq T_{\eta,s_0}(\beta_{\eta,H} * i)$ for $i = 0, 1$. The nodes remain the same and active with $T_{\eta,s_0}(\beta_{\eta,H} * 0)$ on the current path at future η stages unless η acts to change them.*

3. *One of the following must occur.*

   (a) *At all future η stages, η acts in Subcase 4A without finding a potential high splitting. In this case, at every future η stage, η either takes outcome $\eta * L$ or acts as in a low challenged case if it is later challenged low.*

   (b) *η eventually acts in Subcase 4A(i) and wins the high challenge.*

   (c) *There is an η stage $s_1 > s_0$ at which η acts in Subcase 4A(ii). At the next η stage $s_2 > s_1$, $U(T_{\eta,s_2}(\beta_{\eta,H} * 1)) = G_\eta * L$ and this node remains unchanged and on the current path at future η stages unless η acts to change this. Furthermore, $T_{\eta,s_1}(\beta_{\eta,H} * 1 * i) \subseteq T_{\eta,s_2}(\beta_{\eta,H} * 1 * i)$ for $i = 0, 1$ and both of these nodes are active. These nodes also remain the same with $T_{\eta,s_2}(\beta_{\eta,H} * 1 * 0)$ on the current path at future η stages unless η acts to change this. Either η takes outcome $\eta * N$ at all future η stages or η eventually meets its high challenge.*

4. *If η meets the high challenge at $s_3 > s$, then $T_{\eta,s}(\beta_{\eta,H}) = T_{\eta,s_3}(\beta_{\eta,H})$, $U(T_{\eta,s_3}(\beta_{\eta,H})) = G_\eta * H$ and $T_{\eta,s}(\beta_{\eta,H} * i) \subseteq T_{\eta,s_3}(\beta_{\eta,H} * i)$ for $i = 0, 1$. Furthermore, all strings γ such that η defined $\Gamma^\gamma(x_\eta) = 0$ in Subcase 4A(ii) or in a verification procedure called in Subcase 4B are forbidden.*

We prove Lemmas 3.15 and 3.16 simultaneously by induction on the length of η. We begin with Lemma 3.16. Let $\hat{\eta}$ be the strategy which challenges η high at stage s. When $\hat{\eta}$ issues the challenge, it moves the current path and stretches $T_{\eta,s}(\beta_{\eta,H})$ to have large length and to have all low states. Furthermore, $T_{\eta,s}(\beta_{\eta,H})$ and $T_{\eta,s}(\beta_{\eta,H} * 0)$ are on the current path and $T_{\eta,s}(\beta_{\eta,H} * 1)$ is active. $\hat{\eta}$ also challenges each strategy ν such that $\nu * H \subseteq \eta$ high (and by induction Lemma 3.16 applies to these strategies). For each such strategy ν, $T_{\nu,s}(\beta_{\nu,H})$ is stretched and is equal to $T_{\eta,s}(\beta_{\eta,H})$.

Consider Property 1 in Lemma 3.16 and consider the value of $T_{\eta,s}(\beta_{\eta,H})$ after it is stretched. For each ν such that $\nu * H \subseteq \eta$, $T_{\nu,s}(\beta_{\nu,H}) = T_{\eta,s}(\beta_{\eta,H})$. Furthermore, $T_{\nu,s}(\beta'_{\nu,H})$ is high ν splitting. Therefore, we can apply Lemma 3.11 to $T_{\eta,s}(\beta_{\eta,H})$. $T_{\eta,s}(\beta_{\eta,H})$ can only change if η is initialized, η finds a new high split below $T_{\eta,s}(\beta_{\eta,H})$ or some μ with $\eta \subseteq \mu$ moves the current path below $T_{\eta,s}(\beta_{\eta,H})$. Because $T_{\eta,s}(\beta'_{\eta,H})$ is already high η splitting, η does not find new high splits below $T_{\eta,s}(\beta_\eta)$. Because all strategies to the right of $\eta * H$ are initialized at stage s when η is high challenged, the only $\mu \neq \eta$ with $\eta \subseteq \mu$ which can move the current path below $T_{\eta,s}(\beta_{\eta,H})$ satisfy $\eta * H \subseteq \mu$. However, none of these strategies are eligible to act until η meets the high challenge or is initialized. Finally, η only moves the current path above $T_{\eta,s}(\beta_{\eta,H})$ during the high challenge. Therefore, we have established Property 1.

Consider Property 2 in Lemma 3.16. By the next η stage $s_0 > s$ each strategy ν with $\nu * H \subseteq \eta$ has met its high challenge. By Property 4 of Lemma 3.16, we have $T_{\nu,s}(\beta_{\nu,H} * i) \subseteq T_{\nu,s_0}(\beta_{\nu,H} * i)$ and $U(T_{\nu,s_0}(\beta_{\nu,H})) = G_\nu * H$. Also, if ν is such that $\nu * L \subseteq \eta$ or $\nu * N \subseteq \eta$, then ν cannot have found a new high split along the current path without initializing η, so ν does not change the values of nodes along the current path. Therefore, $U(T_{\eta,s_0}(\beta_{\eta,H})) = G_\eta * L$ and $T_{\eta,s}(\beta_{\eta,H} * i) \subseteq T_{\eta,s_0}(\beta_{\eta,H} * i)$.

We also have the hypotheses for Lemma 3.11 for $T_{\eta,s_0}(\beta_{\eta,H} * i)$ since for any $\nu * H \subseteq \eta$ we have $T_{\nu,s_0}(\beta_{\nu,H})$ is high ν splitting. Therefore, no strategy $\nu \subsetneq \eta$ can change the values of $T_{\eta,s_0}(\beta_{\eta,H} * i)$ for $i = 0, 1$ or move the current path from $T_{\eta,s_0}(\beta_{\eta,H} * 0)$ at any η stage after s_0 without initializing η. Furthermore, until η meets its high challenge, it takes either outcome $\eta * L$ or $\eta * N$. Since all of the strategies of lower priority than $\eta * L$ (including $\eta * L$) were initialized at stage s, they all work higher on the trees than these nodes and hence cannot move the current path below any of these nodes. Therefore, unless η moves the current path, both $T_{\eta,s_0}(\beta_{\eta,H} * 0)$ and $T_{\eta,s_0}(\beta_{\eta,H} * 1)$ remain active with $T_{\eta,s_0}(\beta_{\eta,H} * 0)$ on the current path at future η stages. Hence, we have established Property 2.

Once we begin Subcase 4A of the high challenge, one of three things must happen. Either we never find a potential high split or we eventually find a potential high split and act in either Subcase 4A(i) or4 A(ii). If we never find a potential high split, then at every future η stage, we either take outcome $\eta * L$ (if η is not also low challenged) or we act as in the low challenge case (if η is also low challenged). This establishes Property 3(a). If we ever act in Subcase 4A(i), then the high challenge is met and we clearly meet the conditions of Property 4 of Lemma 3.16. This establishes Property 3(b).

Consider what happens if η acts in Subcase 4A(ii) at some stage $s_1 > s_0$. In this case, η moves the current path from $T_{\eta,s_1}(\beta_{\eta,H} * 0)$ to $T_{\eta,s_1}(\beta_{\eta,H} * 1)$ and stretches $T_{\eta,s_1}(\beta_{\eta,H} * 1)$. η defines $\Gamma^{T_{\eta,s_1}(\beta_{\eta,H}*1*0)}(x_\eta) = 0$ and performs the various calculations to issue its challenges. We can apply the same arguments used to establish Properties 1 and 2 in Lemma 3.16 to $T_{\eta,s_1}(\beta_{\eta,H} * 1)$ to get the following properties: $T_{\eta,s_1}(\beta_{\eta,H} * 1)$ doesn't change after this stage; at the next η stage $s_2 > s_1$, $U(T_{\eta,s_2}(\beta_{\eta,H} * 1)) = G_\eta * L$, $T_{\eta,s_1}(\beta_{\eta,H} * 1 * i) \subseteq T_{\eta,s_2}(\beta_{\eta,H} * 1 * i)$, these nodes remain active and these nodes will not change unless η later changes them in Subcase 4B. Also, the current path runs through $T_{\eta,s_2}(\beta_{\eta,H} * 1 * 0)$ and it will continue to run through this node unless η changes this in Subcase 4B.

η acts in Subcase 4B at the next η stage s_2 and begins to wait for $[\eta]^{T_{\eta,s_2}(\beta_{\eta,H}*1)}(w_\eta)$ to converge. (Because $T_{\eta,s_1}(\beta_{\eta,H} * 1)$ was stretched, the length of $T_{\eta,s_2}(\beta_{\eta,H} * 1)$ is longer than the use of $[\eta]$ on w_η.) If this computation never converges, then at all future η stages, η takes outcome $\eta * N$. If this does eventually converge at stage $t_0 \geq s_2$, then η calls a verification procedure with $\sigma_0 = \beta_{\eta,H} * 1$. Notice that we have $\Gamma^{T_{\eta,t_0}(\sigma_0 * 0)}(x_\eta) = 0$, the current path runs through $T_{\eta,t_0}(\sigma_0 * 0)$, $T_{\eta,t_0}(\sigma_0 * 1)$ is active and $U(T_{\eta,t_0}(\sigma_0)) = G_\eta * L$ when the verification procedure is called. (These facts verify Property 1 in Lemma 3.15 in the case when η is a high challenged R strategy calling a verification procedure.) Technically, in our induction, we now need to show that Lemma 3.15 holds. We do this below without assuming anything except the properties just listed. Given that Lemma 3.15 holds for η, we know that it terminates after finitely many stages. When it terminates at stage s_3, η declares the high challenge won and takes outcome $\eta * H$.

We need to see that the conditions in Property 4 hold in this case. The cone above $T_{\eta,s_1}(\beta_{\eta,H} * 0)$ (which has remained frozen since stage s_1) is unfrozen and η uses $T_{\eta,s_3}(\beta_{\eta,H} * 1) = T_{\eta,s_2}(\beta_{\eta,H} * 1)$ and either τ_1 or τ_0 (in the notation from the construction case for a high challenged strategy) to make $T_{\eta,s_3}(\beta_{\eta,H})$ high splitting. By Property 1, $T_{\eta,s}(\beta_{\eta,H}) = T_{\eta,s_3}(\beta_{\eta,H})$. By Property 2 and the fact that η just found a high split for $T_{\eta,s_3}(\beta_{\eta,H})$, we have $U(T_{\eta,s_3}(\beta_{\eta,H})) = G_\eta * H$. Since $T_{\eta,s}(\beta_{\eta,H} * 1) \subseteq T_{\eta,s_1}(\beta_{\eta,H} * 1) = T_{\eta,s_3}(\beta_{\eta,H} * 1)$ and $T_{\eta,s_2}(\beta_\eta * 0) \subseteq \tau_0, \tau_1$ (and the cone above $T_{\eta,s_2}(\beta_\eta * 0)$ has not changed since it was frozen at stage s_2), $T_{\eta,s}(\beta_{\eta,H} * i) \subseteq T_{\eta,s_3}(\beta_{\eta,H} * i)$ for $i = 0, 1$.

Finally, all definitions of the form $\Gamma^\gamma(x_\eta) = 0$ made by η are either made by the verification procedure (in which case they are currently η frozen by Lemma 3.15) or made by the action of η in Subcase 4A(ii). The only definition made in Subcase 4A(ii) is for $\gamma = T_{\eta,s_1}(\beta_\eta * 1 * 0)$. Since this node was frozen when the verification procedure was called with $\sigma_0 = \beta_\eta * 1$, the oracle string used in each Γ definition made for x_η by η in meeting its high challenge is frozen when the verification procedure ends. Therefore, all of these oracle strings are forbidden by η in Subcase 4B when the verification procedure ends. The conditions of Property 4 are met and we have completed the proof of Lemma 3.16.

Consider Lemma 3.15. To see that Property 1 holds at stage t_0, we need to consider separately the cases when the verification procedure is called by an R strategy in Subcase 4B of a high challenge and when the verification procedure is called by a P strategy. If η is an R strategy acting in Subcase 4B, then we have verified these properties above. If η is a P strategy acting in Case 3, then $\sigma_0 = \alpha_\eta$ and $\mu'_{-1} = \eta'$. By Lemma 3.14, $T_{\eta',t_0}(\alpha_\eta * 0) = T_{\eta',t_0}(\sigma_0 * 0)$ is on the current path and $T_{\eta',t_0}(\alpha_\eta * 1) = T_{\eta',t_0}(\sigma_0 * 1)$ is active when the verification procedure is called. When α_η was chosen at $u < t_0$, $U(T_{\eta',u}(\alpha_\eta)) = G_\eta$. If any higher priority strategy found a new high split to raise the state of some string below this node after u, then η would have been initialized and α_η would have been redefined. Therefore, $U(T_{\eta',t_0}(\alpha_\eta)) = G_\eta$. Finally, when α_η was defined at stage $u < t_0$, η picked x_η and defined $\Gamma^{T_{\eta',u}(\alpha_\eta * 0)}(x_\eta) = 0$. Because $T_{\eta',u}(\alpha_\eta * 0) = T_{\eta',t_0}(\alpha_\eta * 0)$, we have all the required properties of $\sigma_0 = \alpha_\eta$ at stage t_0. This establishes Property 1.

At stage t_0, the verification procedure moves the current path from $T_{\mu_{-1},t_0}(\sigma_0 * 0)$ to $T_{\mu_{-1},t_0}(\sigma_0 * 1)$ and freezes the cone above $T_{\mu_{-1},t_0}(\sigma_0 * 0)$. It redefines T_{ν,t_0} for $\nu \subseteq \mu_{-1}$ by stretching and defines X_ν for $\nu * L \subseteq \mu_{-1}$. Assume that not all of the X_ν are empty. (That is, the verification procedure does not end at this stage.) We define μ_0 to be the least priority strategy such that $X_{\mu_0} \neq \emptyset$ and define σ_1 so that $T_{\mu_0,t_0}(\sigma_1)$ is the least node along the current path on T_{μ_0,t_0} which was stretched. Because the length of $T_{\mu_0,t_0}(\sigma_1)$ is long and $T_{\mu_0,t_0}(\sigma_1)$ is active, the current path runs through $T_{\mu_0,t_0}(\sigma_1 * 0)$ and $T_{\mu_0,t_0}(\sigma_1 * 1)$ is active. We place a link from μ_0 to η, define $\Gamma^{T_{\mu_0,t_0}(\sigma_1 * 0)}(x_\eta) = 0$ and issue the appropriate challenges. The stage ends and either all lower priority strategies are initialized (if η is a P strategy) or all strategies of lower priority than $\eta * L$

are initialized (if η is an R strategy).

Consider the action of the R strategies $\nu \subseteq \mu_0$ between stages t_0 and t_1. If $\nu * H \subseteq \mu_0$, then ν is challenged high at stage t_0 and $\beta_{\nu,H}$ is such that $T_{\nu,t_0}(\beta_{\nu,H}) = T_{\mu_0,t_0}(\sigma_1)$ (since σ_1 is the stretched node of T_{μ_0,t_0}). By our assumption, ν meets its high challenge at some stage $u > t_0$. By Lemma 3.16, $U(T_{\nu,u}(\beta_{\nu,H})) = G_\nu * H$ and $T_{\nu,t_0}(\beta_{\nu,H} * i) \subseteq T_{\nu,u}(\beta_{\nu,H} * i)$.

If $\nu * L \subseteq \eta$ and $\nu \subseteq \mu_0$, then by our assumption, ν eventually meets its low challenge. At each ν stage u at which ν is still low challenged, it defines $T_{\nu,u}$ trivially from $T_{\nu'',u}$. Furthermore, at stages u after ν has met its low challenge, it defines $T_{\nu,u}$ by searching for high splittings and failing to find them. Therefore, it does not change any values on $T_{\nu,u}$.

If $\nu * N \subseteq \eta$, then ν must have been high or low challenged before stage t_0 by a strategy to the left of η in the tree of strategies. ν cannot meet this challenge without initializing η, and therefore ν must take outcome $\nu * N$ at every ν stage between t_0 and t_1. Hence, it defines $T_{\nu,u}$ trivially from $T_{\nu'',u}$ at each ν stage u between t_0 and t_1.

When μ_0 meets its low challenge and follows the link back to η, we have the following properties. $T_{\mu_0,t_1}(\sigma_1) = T_{\mu_0,t_0}(\sigma_1)$ since the current path has not moved below here and no R strategy has found a high split below here. Each ν such that $\nu * H \subseteq \mu_0$ has found a ν high split for $T_{\nu,t_0}(\beta_\nu) = T_{\mu_0,t_0}(\sigma_1)$ and no ν such that $\nu * L \subseteq \mu_0$ or $\nu * N \subseteq \mu_0$ has found a new high split below this node or changed the values of its nodes below here. Hence, $U(T_{\mu_0,t_1}(\sigma_1)) = G_{\mu_0} * L$. Furthermore, since the high splits found by strategies such that $\nu * H \subseteq \mu_0$ have the property that $T_{\nu,t_0}(\beta_{\nu,H} * i) \subseteq T_{\nu,u}(\beta_{\nu,H} * i)$ when they are found at stage u and since the current path does not move below these nodes before stage t_1 (by a case analysis as in the proof of Lemma 3.11), we have that $T_{\mu_0,t_0}(\sigma_1 * i) \subseteq T_{\mu_0,t_1}(\sigma_1 * i)$, that these nodes are still active and that $T_{\mu_0,t_1}(\sigma_1 * 0)$ is still on the current path. Therefore, we have established Property 2 of Lemma 3.15 in the case when $n = 1$. Applying this reasoning inductively gives the full version of Property 2.

It remains to see that the verification procedure only acts finitely often before ending. For $n \geq 1$, consider the definition of μ_n at stage t_n. Because we follow a link from μ_{n-1} to η at stage t_n and because this link is established at stage t_{n-1}, none of the strategies ν such that $\mu_{n-1} \subsetneq \nu$ and $\nu * L \subseteq \eta$ is eligible to act between stages t_{n-1} and t_n. Therefore, none of these strategies has seen any new computations and $X_\nu = \emptyset$ for all of these strategies.

Furthermore, we claim that $X_{\mu_{n-1}} = \emptyset$ at stage t_n. To see this fact, we need to distinguish $X_{\mu_{n-1}}$ as defined during the $(n-1)^{\text{st}}$ cycle, which we denote $X'_{\mu_{n-1}}$, and $X_{\mu_{n-1}}$ as defined during this n^{th} cycle, which we denote $X_{\mu_{n-1}}$. $T_{\mu_{n-1},t_{n-1}}(\sigma_n)$ was stretched at stage t_{n-1} so it has length longer than the $[\mu_{n-1}]$ use of any number $x \in X'_{\mu_{n-1}}$. Therefore, μ_{n-1} never looks above this node for computations on elements of $X'_{\mu_{n-1}}$ between stages t_{n-1} and t_n. $\beta_{\mu_{n-1},L}$ is defined at stage t_n to be such that when the verification procedure moves the current path from $T_{\mu_{n-1},t_n}(\sigma_n * 0)$ to $T_{\mu_{n-1},t_n}(\sigma_n * 1)$, it moves from $T_{\mu_{n-1},t_n}(\beta_{\mu_{n-1},L} * 0)$ to $T_{\mu_{n-1},t_n}(\beta_{\mu_{n-1},L} * 1)$. Therefore, $\beta_{\mu_{n-1},L}$ is defined at stage t_n to be equal to σ_n. Because $T_{\mu_{n-1},t_{n-1}}(\sigma_n) = T_{\mu_{n-1},t_n}(\sigma_n) =$

$T_{\mu_{n-1},t_n}(\beta_{\mu_{n-1},L})$, μ_{n-1} has never looked at computations using oracles above $T_{\mu_{n-1},t_n}(\beta_{\mu_{n-1},L})$. It follows that $X_{\mu_{n-1}}$ is defined to be $\emptyset$ at stage t_n and hence $\mu_n \subsetneq \mu_{n-1}$. Therefore, we can only return to the verification procedure finitely often before it discovers that all $X_\mu = \emptyset$ and ends.

Finally, we need to check that all Γ definitions made by the verification procedure are frozen when the procedure terminates. In the n^{th} cycle, η defines $\Gamma^{T_{\mu_n,t_n}(\sigma_{n+1}*0)}(x_\eta) = 0$. In the $(n+1)^{\text{st}}$ cycle, η moves the current path from $T_{\mu_n,t_{n+1}}(\sigma_{n+1} * 0)$ to $T_{\mu_n,t_{n+1}}(\sigma_{n+1} * 1)$. Since $T_{\mu_n,t_{n+1}}(\sigma_{n+1}) = T_{\mu_n,t_n}(\sigma_{n+1})$ and $T_{\mu_n,t_n}(\sigma_{n+1} * i) \subseteq T_{\mu_n,t_{n+1}}(\sigma_{n+1} * i)$ for $i = 0, 1$, the node $T_{\mu_n,t_n}(\sigma_{n+1} * 0)$ is frozen by η. Therefore, at the start of the $(n+1)^{\text{st}}$ cycle, the Γ definition made by the verification procedure in the n^{th} cycle is frozen. This completes the proof of Lemma 3.15.

Having gained some understanding of strategies which are challenged high, we turn to strategies η which are challenged low. Assume η is challenged low by $\hat{\eta}$. This could happen either because $\hat{\eta}$ calls a verification procedure or because $\hat{\eta}$ is challenged high and acting in Subcase 4A(ii). We begin with the case when $\hat{\eta}$ calls a verification procedure. Assume that η is challenged low by $\hat{\eta}$ at stage s as part of the n^{th} cycle of a verification procedure. By setting $\mu_{-1} = \hat{\eta}$ and imagining a "trivial link" from μ_{-1} to $\hat{\eta}$, we can treat the 0^{th} cycle with the same notation as the n^{th} cycle. In this situation, we have just followed a link from μ_{n-1} to $\hat{\eta}$ and $\hat{\eta}$ moves the current path from $T_{\mu_{n-1},s}(\sigma_n * 0)$ to $T_{\mu_{n-1},s}(\sigma_n * 1)$. By the proof of Lemma 3.15, we know $U(T_{\mu_{n-1},s}(\sigma_n)) = G_{\mu_{n-1}} * L$. (Technically, if $\hat{\eta}$ is a P strategy and $n = 0$, then we have $U(T_{\mu'_{-1},s}(\sigma_0)) = G_{\mu_{-1}}$ instead. This minor change in notation is the only difference between $\hat{\eta}$ being a P or R strategy and it does not effect the argument below.) Because $\hat{\eta}$ challenges η low during this cycle, we know $\eta \subseteq \mu_n$ and $\eta * L \subseteq \hat{\eta}$. $\beta_{\eta,L}$ is defined such that the current path just moved from $T_{\eta,s}(\beta_{\eta,L} * 0)$ to $T_{\eta,s}(\beta_{\eta,L} * 1)$. $\hat{\eta}$ also redefines the tree $T_{\eta,s}$ by stretching. In the argument below, we consider the trees before they are stretched by $\hat{\eta}$ and we make comments at the end of the proof to take into account the effect of stretching.

Lemma 3.17. *Under these circumstances, $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$, even after $\hat{\eta}$ performs its stretching.*

Proof. We split into two cases: when there is an R strategy ν such that $\nu * H \subseteq \mu_{n-1}$ and when there is no such strategy. If there is no R strategy ν with $\nu * H \subseteq \mu_{n-1}$, then G_η contains only low states, so $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$.

Assume there is a strategy ν such that $\nu * H \subseteq \mu_{n-1}$. In this case, we first need a better understanding of where exactly the current path moves. Let ν be the lowest priority R strategy such that $\nu * H \subseteq \mu_{n-1}$. Consider an R strategy $\hat{\nu}$ such that $\nu * H \subseteq \hat{\nu} \subseteq \mu_{n-1}$ and how $\hat{\nu}$ defines its trees at $\hat{\nu}$ stages before μ_{n-1} follows its link at stage s. Because ν is the lowest priority strategy with $\nu * H \subseteq \mu_{n-1}$, we know that either $\hat{\nu} * N \subseteq \mu_{n-1}$ or $\hat{\nu} * L \subseteq \mu_{n-1}$. If $\hat{\nu} * N \subseteq \hat{\eta}$, then $T_{\hat{\nu},s}$ is defined trivially from $T_{\hat{\nu}'',s}$ because trees are always defined trivially when a strategy takes the N outcome. If $\hat{\nu} * L \subseteq \hat{\eta}$, then $\hat{\nu}$ cannot have found a new high splitting along the current path, so $\hat{\nu}$ searches for new high splits and defines $T_{\hat{\nu},s}$ trivially when it doesn't find any. Therefore, all trees $T_{\hat{\nu},s}$ for $\nu * H \subseteq \hat{\nu} \subseteq \mu_{n-1}$ are defined trivially.

Let γ be such that $T_{\nu,s}(\gamma) = T_{\mu_{n-1},s}(\sigma_n)$. Because all the trees between $\nu * H$ and μ_{n-1} are defined trivially, $T_{\mu_{n-1},s}(\sigma_n * i) = T_{\nu,s}(\gamma * i)$. Because $U(T_{\mu_{n-1},s}(\sigma_n)) = G_{\mu_{n-1}} * L$ and $\nu * H \subseteq \mu_{n-1}$, we know that $U(T_{\nu,s}(\gamma)) = G_\nu * H$. Let $t \leq s$ be the ν stage at which $T_{\nu,t}(\gamma)$ becomes ν high splitting. Because we chose high splitting extensions for $T_{\nu,t}(\gamma)$ at stage t, the ν'' state of each $T_{\nu,t}(\gamma * i)$ is G_ν. A case analysis using Lemma 3.11 shows that the values of $T_{\nu,t}(\gamma)$, $T_{\nu,t}(\gamma * 0)$ and $T_{\nu,t}(\gamma * 1)$ do not change and the current path does not move below these nodes after ν's action at stage t and before we follow the link from μ_{n-1} to $\hat{\eta}$ at stage s. Therefore, when we follow the link from μ_{n-1} to $\hat{\eta}$ at stage s, we have that the ν'' state of each $T_{\nu,s}(\gamma * i)$ is G_ν (and they may or may not be ν high splitting).

At stage s, $\hat{\eta}$ moves the current path from $T_{\mu_{n-1},s}(\sigma_n * 0)$ to $T_{\mu_{n-1},s}(\sigma_n * 1)$ and hence from $T_{\nu,s}(\gamma * 0)$ to $T_{\nu,s}(\gamma * 1)$. $\beta_{\eta,L}$ is defined such that the current path just moved from $T_{\eta,s}(\beta_{\eta,L} * 0)$ to $T_{\eta,s}(\beta_{\eta,L} * 1)$.

We break into cases depending on whether $\nu * H \subseteq \eta$ or $\eta \subsetneq \nu$. (Notice that $\eta \neq \nu$ since $\nu * H \subseteq \hat{\eta}$ and $\eta * L \subseteq \hat{\eta}$.) If $\nu * H \subseteq \eta$, then since all the trees between $\nu * H$ and μ_{n-1} are defined trivially at stage s, $\beta_{\eta,L}$ is such that $T_{\nu,s}(\gamma) = T_{\eta,s}(\beta_{\eta,L})$ and $T_{\nu,s}(\gamma * i) = T_{\eta,s}(\beta_{\eta,L} * i)$. Because there are no high states between ν and η (since ν was lowest priority strategy with $\nu * H \subseteq \mu_{n-1}$), $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$ as required.

If $\eta \subsetneq \nu$, then we may have $T_{\nu,s}(\gamma) \subsetneq T_{\eta,s}(\beta_{\eta,L})$ because $T_{\nu,s}(\gamma)$ is ν high splitting. However, we do have that $T_{\eta,s}(\beta_{\eta,L} * i) \subseteq T_{\nu,s}(\gamma * i)$ since γ and $\beta_{\eta,L}$ are such that the current path just moved from $T_{\nu,s}(\gamma * 0)$ to $T_{\nu,s}(\gamma * 1)$ and from $T_{\eta,s}(\beta_{\eta,L} * 0)$ to $T_{\eta,s}(\beta_{\eta,L} * 1)$. Because $U(T_{\nu,s}(\gamma)) = G_\nu * H$, the ν'' states of $T_{\nu,s}(\gamma * i)$ are G_ν and $\eta \subsetneq \nu$, it follows that $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$ as required.

Finally, when $\hat{\eta}$ redefines the trees by stretching in the verification procedure, it may be that $T_{\eta,s}(\beta_{\eta,L} * 1)$ is stretched. However, if it is stretched, then it is the least node on $T_{\eta,s}$ which is stretched, so the stretched value of this node extends the prestretched value. Hence the state of $T_{\eta,s}(\beta_{\eta,L})$ remains the same. (It is important that we considered the state of $T_{\nu,s}(\gamma * 1)$ before it is potentially stretched. $T_{\nu,s}(\gamma * 1)$ may be the least node of $T_{\nu,s}$ which is changed by stretching, in which case, $U(T_{\nu,s}(\gamma * 1))$ has all low states after it is redefined.) □

A similar argument proves the same statement in the case when η is challenged low by a strategy $\hat{\eta}$ which is acting in Subcase 4A(ii) of a high challenge.

Lemma 3.18. *Assume η is challenged low at stage s by a strategy $\hat{\eta}$ which is acting in Subcase 4A(ii) of a high challenge. Then $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$.*

Lemma 3.19. *Assume that η is low challenged by $\hat{\eta}$ at stage s. Unless η is initialized, we have the following properties.*

1. *At least until η meets its low challenge, $T_{\eta,s}(\beta_{\eta,L})$ remains unchanged at future η stages. $T_{\eta,s}(\beta_{\eta,L} * 1)$ may be stretched at stage s, but then remains unchanged and on the current path at future η stages.*

2. *Either η takes $\eta * N$ at every future η stage or η eventually meets the low challenge or η finds a new high split using a number from X_η.*

Proof. Property 2 follows immediately by inspecting the action of a low challenged strategy. We show Property 1. By Lemmas 3.17 and 3.18, $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$. By the definition of $\beta_{\eta,L}$, the current path just moved to $T_{\eta,s}(\beta_{\eta,L} * 1)$ and this node may have been stretched. Consider which strategies could change $T_{\eta,s}(\beta_{\eta,L} * 1)$ or move the current path below this node without initializing η. Obviously nothing to the left of η can cause these changes and because all strategies to the right of η are initialized by $\hat{\eta}$ when η is challenged, they work higher on the trees. The only strategies ν with $\eta \subsetneq \nu$ which are eligible to act before η meets its challenge satisfy $\eta * N \subseteq \nu$. Since $\eta * L \subseteq \hat{\eta}$, these strategies are initialized by $\hat{\eta}$ at stage s and work higher on the trees.

Consider a strategy $\nu \subsetneq \eta$. If ν is a P strategy, then it initializes all lower priority strategies including η when it moves the current path. If ν is an R strategy and $\nu * L \subseteq \eta$ or $\nu * N \subseteq \eta$, then ν cannot find high splits below $T_{\eta,s}(\beta_{\eta,L})$ or move the current path without initializing η. If $\nu * H \subseteq \eta$, then $T_{\eta,s}(\beta_{\eta,L})$ is already ν high splitting since $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$. Therefore, any new high splits would be above this node. Furthermore, ν is challenged high by $\hat{\eta}$ at stage s so if it moves the current path, it does so from $T_{\nu,s}(\beta_{\nu,H} * 0)$ to $T_{\nu,s}(\beta_{\nu,H} * 1)$. Because $\nu * H \subseteq \hat{\eta}$, $T_{\nu,s}(\beta_{\nu,H})$ was stretched at stage s and so $T_{\eta,s}(\beta_{\eta,L} * 1) \subseteq T_{\nu,s}(\beta_{\nu,H})$. Therefore, any movement of the path caused by ν will not effect $T_{\eta,s}(\beta_{\eta,L} * 1)$. This establishes Property 1. □

We define the true path in the tree of strategies as usual: an R_e or P_e strategy η is on the true path if and only if η is the leftmost strategy acting for R_e or P_e which is eligible to act infinitely often. We next show that various properties hold of strategies on the true path and that the true path is infinite.

Lemma 3.20. *Assume that η is on the true path.*

1. *η is initialized only finitely often.*

2. *If η is never initialized after stage t, then for all $\mu * L \subseteq \eta$, μ meets all low challenges issued after t and for all $\mu * H \subseteq \eta$, μ meets all high challenges issued after t.*

3. *p_η and α_η are eventually permanently defined. Furthermore, if they are permanently defined at stage s, then $T_{\eta'',s}(\alpha_\eta)$ (if η is an R strategy) or $T_{\eta',s}(\alpha_\eta)$ (if η is a P strategy) has reached a limit and is on the current path at all future stages. Therefore, $T_{\eta,s}(\lambda)$ reaches its limit at stage s.*

4. *η has a successor on the true path.*

Proof. We proceed by induction on the length of η. Let s be an η stage such that no strategy $\mu \subsetneq \eta$ is initialized after s, both p_μ and α_μ are permanently defined before stage s and no strategy to the left of η in the tree of strategies is eligible to act after s.

To prove Property 1, we examine how strategies $\nu \subsetneq \eta$ could end a stage after s and initialize η. If $\nu \subsetneq \eta$ is a P strategy, then ν only ends a stage and initializes lower priority strategies when it acts in Case 1 or Case 2 or calls a verification procedure in Case 3. Since p_ν and α_ν are permanently defined by stage s, ν does not act in either Case 1 or 2 after stage s. Since s is an η stage, ν cannot be in the middle of a verification procedure at stage s (by Lemma 3.7). Suppose η calls a verification procedure after stage s. This means ν has not yet reached Case 4 of the P action at stage s, so $\nu * W \subseteq \eta$. Applying Property 2 of Lemma 3.20 inductively to ν and using the fact that ν is not initialized after stage s, we conclude from Lemma 3.15 that this verification procedure eventually ends and ν acts in Case 4 of the P action. After this stage, ν takes outcome $\nu * S$ contradicting the fact that no strategy to the left of η acts after stage s. Therefore, ν does not initialize η after stage s.

If $\nu \subsetneq \eta$ is an R strategy, then ν only ends a stage and initializes lower priority strategies when it acts in Case 1 or Case 2 or Subcases 4A(ii) or 4B of the high challenge R action. As above, ν does not act in Case 1 or Case 2 after stage s. When ν acts in Subcase 4A(ii) (and later in Subcase 4B) of a high challenge, it initializes all strategies of lower priority than $\nu * L$ (including $\nu * L$). Therefore, if $\nu * H \subseteq \eta$, then η is not initialized by ν after stage s. Otherwise, suppose $\nu * L \subseteq \eta$ or $\nu * N \subseteq \eta$ and consider what happens when ν acts in one of these subcases. Suppose ν acts in Subcase 4A(ii) after stage s. ν initializes η and ends the stage. Applying Property 2 of Lemma 3.20 inductively to ν and using the fact that ν is not initialized after s, we conclude from Lemma 3.16 that ν either takes outcome $\nu * N$ at all future stages (and hence does not initialize η again) or ν eventually calls a (finitary) verification procedure in Subcase 4B and wins the high challenge. However, in the latter case, ν takes outcome $\nu * H$ which moves the path in the tree of strategies to the left of η after stage s contrary to our assumption. Therefore, after stage s, ν initializes η at most once. This completes the proof of Property 1.

We show Property 2 by induction on μ. Assume that $\mu * L \subseteq \eta$. We inductively apply Property 2 in Lemma 3.20 together with Property 2 in Lemma 3.19 to μ. If μ is challenged low after stage s, then either μ eventually meets this challenge or at all future μ stages μ takes outcome $\mu * N$. Because there cannot be a link jumping over $\mu * L$ while μ is low challenged, the latter situation contradicts the fact that η is on the true path.

Assume that $\mu * H \subseteq \eta$ and μ is challenged high after stage s. We inductively apply Property 2 of Lemma 3.20 together with Lemma 3.16 to μ. If μ fails to meet the high challenge, then either μ never finds a potential high split in Subcase 4A or it eventually acts in Subcase 4A(ii). If μ eventually acts in Subcase 4A(ii) but does not meet the high challenge, then μ remains high challenged forever and takes outcome $\mu * N$ at every future μ stage. Since there are no links jumping over $\mu * H$ while μ is high challenged, this contradicts the fact that η is on the true path. If μ never finds a potential high split in Subcase 4A, then at every future μ stage either μ takes outcome $\mu * L$ (if μ is not also low challenged) or μ acts as in the low challenge case. If μ acts in the low challenge case, it cannot find a new high split (since otherwise it would have found it when it looked in Subcase 4A in the high challenge action) so it either takes outcome $\mu * L$ or $\mu * N$. Since it is impossible for μ to take outcome $\mu * H$ in this situation and since there are no links jumping over $\mu * H$ when μ is high challenged, this contradicts the fact that η is on the true path. This completes the proof of Property 2.

To see Property 3, notice that p_η is permanently defined at the first η stage after which η is never initialized again. η now begins to look for a node α of length p_η such that $T_{\eta'',s}(\alpha)$ (if η is an R strategy) or $T_{\eta',s}(\alpha)$ (if η is a P strategy) is on the current path and has state G_η. Because p_η is defined to be large, this node starts out with all low states. If G_η contains all low states, we pick α_η at the next η stage. Otherwise, G_η has at least one high state, so η ends the stage and tries again at each subsequent η stage. Each strategy ν such that $\nu * H \subseteq \eta$ finds a new high split along the current path each time it takes outcome $\nu * H$. Therefore, each time η is eligible to act, the state of some node on the current path has increased. Since η is eligible to act infinitely often and p_η does not change, η must eventually see a suitable node on the current path with state G_η and define α_η. The rest of Property 3 follows by Lemmas 3.13 and 3.14. This completes the proof of Property 3.

Finally, we verify Property 4. Assume s is an η stage such that η has permanently defined p_η and α_η by stage s. If η is a P strategy, then η defines x_η permanently at the same stage as it defines α_η. Either x_η eventually enters W_η after stage s or it does not. If x_η never enters W_η, then η takes outcome $\eta * W$ at every future η stage, so $\eta * W$ is on the true path. If x_η eventually enters W_η, then η calls a verification procedure at the next η stage. By Lemma 3.15 and Property 2 of Lemma 3.20, this verification procedure is finite. When it ends, η acts in Case 4 of the P strategy and takes outcome $\eta * S$. At every future η stage, η takes outcome $\eta * S$, so $\eta * S$ is on the true path.

Assume that η is an R strategy. After stage s, η never acts in Cases 1 or 2 for an R strategy. Therefore, the only times that η ends a stage after s is when η acts in Subcase 4A(ii) or in a verification procedure called by Subcase 4B of a high challenge. We split into three cases depending on whether η is challenged infinitely often or finitely often and whether it meets the last high challenge (if it is challenged high only finitely often).

First, suppose that there is a stage $t > s$ after which η is never challenged high and that η has met its last high challenge by stage t. Because the only times that η can end the stage are during a high challenge, η will take one of its three outcomes at every η stage after t. Because η is eligible to act infinitely often, at least one of its successors must be eligible to act infinitely often. The leftmost such outcome is on the true path.

Second, suppose that η is challenged high infinitely often. Let $t_1 < t_2 < \cdots$ denote the stages after s at which some strategy issues a high challenge to η. Because η can be high challenged by at most one strategy at a time, η must either meet the high challenge issued at t_i before t_{i+1} or the challenge issued at t_i must be removed by initialization before stage t_{i+1}. Let $\hat{\eta}$ be the strategy that issues the high challenge at stage t_i. We know $\eta * H \subseteq \hat{\eta}$ and no strategy ν with $\eta * H \subseteq \nu$ is eligible to act until η meets the challenge or it is removed by initialization. Because of these facts and because $\eta * H$ is the left most outcome of η, the only strategies that could remove the challenge by initialization are those of higher priority than η.

Suppose ν has higher priority than η and ν initializes $\hat{\eta}$. If ν is to the left of η or $\nu \subsetneq \eta$ is a P strategy, then ν also initializes η contrary to assumption. If $\nu \subseteq \eta$ is an R strategy, then (since ν doesn't act in Cases 1 or 2 after stage s), ν acts in either Subcase 4A(ii) or 4B of a high challenge and initializes all strategies of lower priority than $\nu * L$. Therefore, $\hat{\eta}$ has lower priority than $\nu * L$. Because $\nu \subseteq \eta \subseteq \hat{\eta}$, we must have either $\nu * L \subseteq \hat{\eta}$ or $\nu * N \subseteq \hat{\eta}$. Putting together the facts that $\nu \subseteq \eta$, $\eta * H \subseteq \hat{\eta}$ and either $\nu * L \subseteq \hat{\eta}$ or $\nu * N \subseteq \hat{\eta}$ implies that either $\nu * L \subseteq \eta$ or $\nu * N \subseteq \eta$. Therefore, when ν initializes $\hat{\eta}$, it also initializes η contrary to our assumption. Hence, the challenge issued by $\hat{\eta}$ cannot be removed by initialization after stage s, so η must meet each of these high challenges. When η meets a high challenge, it takes outcome $\eta * H$. Therefore, $\eta * H$ is eligible to act infinitely often. Since $\eta * H$ is the leftmost outcome of η, it must be on the true path.

Third, suppose that η is only challenged high finitely often after s but it fails to meet the last high challenge. Let $t > s$ be the stage at which this last high challenge is issued. We split into cases depending on how η acts while trying (and failing) to meet this high challenge. η either acts in Subcase 4A at every future η stage (and fails to find a potential high split) or η eventually acts in Subcase 4A(ii). (η cannot act in Subcase 4A(i) since it would win the high challenge in that subcase.) If η ever acts in Subcase 4A(ii), then by Lemma 3.16, η must either win the high challenge or take outcome $\eta * N$ at every future η stage. Since η does not win the challenge, $\eta * N$ is on the true path.

Suppose η never finds a potential high split in Subcase 4A of the high challenge. At every η stage after t, η either takes outcome $\eta * L$ or acts as a low challenged strategy (if η is also low challenged). The only possible outcomes for a low challenged strategy are L and N. Therefore, at every future η stage, η either takes outcome $\eta * L$ or $\eta * N$, so one of these must be on the true path. □

Lemma 3.21. *$A = \lim_s A_s$ is a Δ^0_2 set.*

Proof. Let $\eta_0 \subseteq \eta_1 \subseteq \eta_2 \subseteq \cdots$ be the sequence of R strategies on the true path and let $s_0 < s_1 < s_2 < \cdots$ be a sequence of stages such that for all k, s_k is an η_k stage by which α_{η_k} has been permanently defined. By Lemma 3.20, $T_{\eta_k, s_k}(\lambda) = T_{\eta''_k, s_k}(\alpha_{\eta_k})$ has reached its limit and is contained in the current path at all future stages. Therefore, A is determined up to the length of this node at stage s_k. □

We know that for an R strategy η on the true path, $T_{\eta,s}(\lambda)$ reaches a limit. We need to show that various other nodes also approach limits.

Lemma 3.22. *Let η be an R strategy with $\eta * H$ on the true path. Let t be a stage such that α_η is defined permanently by stage t (and hence η is not initialized after t). For any α and any $s > t$, if $U(T_{\eta,s}(\alpha)) = G_\eta * H$ and $T_{\eta,s}(\alpha)$ becomes high splitting at stage s, then $T_{\eta,s}(\alpha)$ has reached a limit.*

Proof. By Lemma 3.12, $T_{\eta,s}(\alpha)$ can only change if it is stretched because the current path is moved below $T_{\eta,s}(\alpha)$ by a strategy μ such that $\eta \subseteq \mu$. However, if any such strategy moves the current path below $T_{\eta,s}(\alpha)$ at stage $u \geq s$ and redefines $T_{\eta,u}$ by stretching, then the least stretched node on $T_{\eta,u}$ has state $G_\eta * L$. Since $T_{\eta,s}(\alpha)$ already has state $G_\eta * H$, it cannot be changed by stretching. □

Lemma 3.23. *Let η be an R strategy on the true path. There is a sequence of strings α_j and η stages t_j indexed by $j \in \omega$ such that $\alpha_0 = \lambda$, α_{j+1} is either $\alpha_j * 0$ or $\alpha_j * 1$, $T_{\eta,t_j}(\alpha_j)$ has reached its limit denoted by $T_\eta(\alpha_j)$, $U(T_{\eta,t_j}(\alpha_j))$ is either $G_\eta * L$ or $G_\eta * H$, $T_{\eta,t_j}(\alpha_j) \subseteq A_{\eta,t_j}$ and the current path never moves below $T_{\eta,t_j}(\alpha_j)$ after stage t_j. (Hence $T_{\eta,t_j}(\alpha_j) = T_\eta(\alpha_j) \subseteq A$.) In addition, the following properties hold.*

1. *$U(T_{\eta,s}(\alpha_j))$ may change at a later stage $s > t_j$, but it reaches a limit denoted by $U(T_\eta(\alpha_j))$ which is either $G_\eta * L$ or $G_\eta * H$. Furthermore both successor nodes $T_{\eta,s}(\alpha_j * i)$ eventually reach limits.*
2. *If $\eta * H$ is on the true path, then $U(T_\eta(\alpha_j)) = G_\eta * H$.*
3. *If $\eta * L$ is on the true path, then there is an n such that $U(T_\eta(\alpha_j)) = G_\eta * L$ for all $j \geq n$.*
4. *If $\eta * N$ is on the true path, then there is a stage t such that $T_{\eta,s}$ is defined trivially from $T_{\eta'',s}$ at all η stages $s > t$.*

Proof. The proof proceeds by induction on η and for each fixed η by induction on j. Let t_0 be a stage such that α_η is permanently defined by stage t_0 and such that if $\eta * L$ (or $\eta * N$) is on the true path, then $\eta * H$ (respectively $\eta * H$ and $\eta * L$) is never eligible to act after stage t_0. By Lemma 3.20, $T_{\eta,t_0}(\lambda) = T_{\eta'',t_0}(\alpha_\eta) \subseteq A_{\eta,t_0}$ has reached its limit, $U(T_{\eta,t}(\lambda)) = G_\eta$ (and may or may not be high $[\eta]$ splitting), and the current path never moves below this node after stage t_0. Therefore, the statement in the main body of the lemma is true when $j = 0$. Assume by induction that $T_{\eta,t_j}(\alpha_j)$ satisfies the conditions in the main body of the lemma. We need to show that Properties 1–4 hold as well.

Before proving these properties, consider what changes can take place in T_{η,t_j} after stage t_j. No R strategy of higher priority can find a new high splitting at or below $T_{\eta,t_j}(\alpha_j)$. Therefore, these strategies do not cause a change in $T_{\eta,t_j}(\alpha_j * i)$ after stage t_j. Consider how the current path could move below $T_{\eta,t_j}(\alpha_j * i)$ after stage t_j (which must occur if these nodes change value because of stretching). Let $\hat{\eta}$ be a P strategy which initiates a series of challenges (via a verification procedure) that cause the current path to move below $T_{\eta,t_j}(\alpha_j * i)$ after stage t_j. We split into cases depending on whether $\hat{\eta}$ calls its verification procedure at a stage $< t_j$ or $\geq t_j$.

Assume $\hat{\eta}$ calls its verification procedure before stage t_j. We further split into cases depending on the relative positions of η and $\hat{\eta}$ in the tree of strategies. If $\eta <_L \hat{\eta}$, then since t_j is an η stage, $\hat{\eta}$ is initialized at the end of stage t_j and its series of challenges is removed by initialization. If $\hat{\eta} \subsetneq \eta$, then η is not eligible to act until the verification procedure is complete. In this case, since t_j is an η stage, the verification procedure must be complete by stage t_j and hence there are no challenges left to move the path. If $\eta \subseteq \hat{\eta}$, then all the challenges issued to strategies $\nu \subsetneq \eta$ in the series initiated by $\hat{\eta}$ before t_j have been met (again since t_j is an η stage). Therefore, we only need to consider the action of strategies ν such that $\eta \subseteq \nu \subseteq \hat{\eta}$ after stage t_j (which we handle in a separate case below).

Finally, assume that $\hat{\eta} <_L \eta$. In this case, let ν be the highest priority strategy currently challenged in the series of challenges initiated by $\hat{\eta}$. In ν is challenged low, then $\nu * L \subseteq \hat{\eta}$. Since t_j is an η stage, we cannot have $\nu * L \subseteq \eta$. Therefore, η is to the right of $\nu * L$ in the tree of strategies. If ν ever meets its low challenge or finds a new high split using a number from X_ν, then ν will move the path in the tree of strategies to the left of η after stage t_j, contrary to our assumption. Therefore, this low challenge is never met or removed by initialization, so the series of challenges issued by $\hat{\eta}$ never moves the current path after t_j. If ν is challenged high, then $\nu * H \subseteq \hat{\eta}$. Again, because t_j is an η stage, η must have lower priority than $\nu * L$. Therefore, if ν ever moves the path in either Subcase 4A(ii) or 4B of the high challenge, it initializes η after t_j contrary to assumption.

We now have established that if $\hat{\eta}$ starts a series of challenges before t_j that has not terminated by t_j and this series of challenges causes the current path to move below $T_{\eta,t_j}(\alpha_j * i)$ after stage t_j, then some strategy ν such that $\eta \subseteq \nu$ must move the current path. On the other hand, if $\hat{\eta}$ does not start its series of challenges until after t_j and this series of challenges moves the current path below $T_{\eta,t_j}(\alpha_j * i)$ after stage t_j, then $\hat{\eta}$ itself moves the current path below $T_{\eta,t_j}(\alpha_j * i)$ after t_j. The key point is that in either case, if the current path is moved below $T_{\eta,t_j}(\alpha_j * i)$ at a future stage $t \geq t_j$, then the movement is caused by a strategy ν such that $\eta \subseteq \nu$ and hence the current path is moved on the tree $T_{\eta,t}$ at this future stage t. Because the current path runs through $T_{\eta,t_j}(\alpha_j)$ permanently after stage t_j, the only places where this movement can take place are from $T_{\eta,t}(\alpha_j * 0)$ to $T_{\eta,t}(\alpha_j * 1)$ or from $T_{\eta,t}(\alpha_j * 1)$ to $T_{\eta,t}(\alpha_j * 0)$. Because the value of $T_{\eta,t_j}(\alpha_j)$ does not change after stage t_j, the least nodes which could be stretched in either of these cases are $T_{\eta,t}(\alpha_j * 1)$ (in the first case) and $T_{\eta,t}(\alpha_j * 0)$ (in the second case). However, in either of these cases, the stretched value of $T_{\eta,t}(\alpha_j * i)$ extends the prestretched value. Therefore, the state of $T_{\eta,t_j}(\alpha_j)$ cannot be lowered because of stretching.

Consider Property 1. By the comments in the previous paragraph, the state of $T_{\eta,t_j}(\alpha_j)$ cannot be lowered because of stretching. Therefore, if η eventually finds a high split for $T_{\eta,t_j}(\alpha_j)$, then the final state of this node is $G_\eta * H$ and otherwise the final state is $G_\eta * L$. Furthermore, the current path can only move between $T_{\eta,t}(\alpha_j * 0)$ and $T_{\eta,t}(\alpha_j * 1)$ finitely many times after t_j. (Roughly, it can move back and forth between these nodes at most once for each strategy ν which is high challenged at $t \geq t_j$ and has $\beta_{\nu,H}$ defined so that $T_{\eta,t}(\alpha_j) = T_{\nu,t}(\beta_{\nu,H})$.) Therefore, each of the nodes $T_{\eta,t_j}(\alpha_j * i)$ can be changed at most finitely often because of stretching and at most once by η finding a new high splitting after stage t_j. Hence, there is a stage $s_0 > t_j$ at which these nodes have reached their limits and the current path does not move again below them. Set $\alpha_{j+1} = \alpha_j * 0$ or $\alpha_j * 1$ depending on which one the current path goes through permanently. Since Lemma 3.23 applies inductively to the R strategies $\subsetneq \eta$, the state of $T_{\eta,s}(\alpha_{j+1})$ must eventually reach $G_\eta * L$ at some later stage and we set t_{j+1} equal to this stage. Notice that the hypotheses for the main body of Lemma 3.23 are now satisfied for $j+1$.

Consider the case when $\eta * H$ is on the true path. Because $\eta * H$ is eligible to act infinitely often and each time $\eta * H$ is eligible to act η finds a new high splitting along the current path, η must eventually find a high splitting for $T_{\eta,t_j}(\alpha_j)$. This establishes Property 2.

Consider the case when $\eta * L$ is on the true path. By our assumption, η never takes outcome $\eta * H$ after stage t_0. Therefore, η never finds a new high split along the current path after this stage. Therefore, the only high splits which occur in the trees $T_{\eta,s}$ for $s \geq t_0$ are the ones that are already present at stage t_0. This fact implies Property 3.

Consider the case when $\eta * N$ is on the true path. By our assumption on stage t_0 in the first paragraph of this proof, η never takes outcome $\eta * L$ or $\eta * H$ after t_0. Therefore, Property 4 follows from the fact that whenever η takes outcome $\eta * N$, it defines $T_{\eta,s}$ trivially from $T_{\eta'',s}$. □

We turn to checking that Γ^A is defined correctly so that $\Gamma^A = B$. First, we verify that $\Gamma^A(x) = 1$ if and only if $x \in B$, and (after an additional technical lemma), we check that if $x \notin B$, then $\Gamma^A(x) = 0$. Note that x is enumerated into B if and only if $x = x_\eta$ for a P strategy η which acts in Case 4.

Lemma 3.24. *For all x, $\Gamma^A(x) = 1$ if and only if $x = x_\eta$ for some P strategy x which reaches Case 4 of its action and hence $x \in B$.*

Proof. Case 4 of a P strategy is the only place where computations of the form $\Gamma^\gamma(x) = 1$ are defined. Therefore, if $\Gamma^A(x) = 1$, then $x = x_\eta$ for some P strategy η which acts in Case 4.

For the other direction, assume that η is a P strategy which acts in Case 4 with x_η at stage s. To get to Case 4, η must have called a verification procedure at some stage $t < s$ which finished at stage s. When the verification procedure is called, the only Γ definition for x_η is $\Gamma^{T_{\eta,t}(\alpha_\eta * 0)}(x_\eta) = 0$. η sets $\sigma_0 = \alpha_\eta$ when it calls the verification procedure, so this procedure freezes $T_{\eta,t}(\alpha_\eta * 0)$. Because the verification procedure eventually finishes, all of the challenges issued by this procedure must be met (and all the challenges they issue must be met, etc.) so Lemma 3.15 applies. Therefore, at stage s, all strings γ such that $\Gamma^\gamma(x_\eta) = 0$ are frozen by the verification procedure. η forbids all of these frozen strings, so the current path will never again pass through any of these strings. Furthermore, it picks a large value n and defines $\Gamma^\gamma(x_\eta) = 1$ for all strings γ of length n which have not been forbidden by η. Whatever A turns out to be, it must contain one of these strings and therefore $\Gamma^A(x_\eta) = 1$ as required. □

Lemma 3.25. *Let η be a P strategy which initiates a series of challenges by calling a verification procedure. If ν is an R strategy which is challenged high in* *this series of challenges at stage s and ν is passed x_ν and $\beta_{\nu,H}$, then $x_\nu = x_\eta$ and $\Gamma^{T_{\nu,s}(\beta_{\nu,H}*0)}(x_\nu) = 0$.*

Proof. We proceed by induction on the depth in the series of challenges. That is, a strategy challenged high by η is challenged at depth 1. If $\hat{\nu}$ is challenged high at depth n by η and ν is challenged high by $\hat{\nu}$, then ν is challenged at depth $n+1$.

The base case is when ν is challenged high by the n^{th} cycle in the verification procedure called by η. In this case, (following the notation of the verification procedure) η defines $\Gamma^{T_{\mu_n,t_n}(\sigma_{n+1}*0)}(x_\eta) = 0$ and passes $x_\nu = x_\eta$ and $\beta_{\nu,H}$ to ν. Because $\beta_{\nu,H}$ is the least node which is stretched on T_{ν,t_n} in this cycle, we have $T_{\nu,t_n}(\beta_{\nu,H}*0) = T_{\mu_n,t_n}(\sigma_{n+1}*0)$. Hence the result holds for this high challenge.

For the induction case, assume that $\hat{\nu}$ has been high challenged in the series of challenges (say at stage u) and $\hat{\nu}$ challenges ν high. By induction, $x_{\hat{\nu}} = x_\eta$ and $\Gamma^{T_{\hat{\nu},u}(\beta_{\hat{\nu},H}*0)}(x_{\hat{\nu}}) = 0$. Let s_0 be the next $\hat{\nu}$ stage after it is challenged high. By Lemma 3.16, $T_{\hat{\nu},u}(\beta_{\hat{\nu},H}*0) \subseteq T_{\hat{\nu},s_0}(\beta_{\hat{\nu},H}*0)$, so $\Gamma^{T_{\hat{\nu},s_0}(\beta_{\hat{\nu},H}*0)}(x_{\hat{\nu}}) = 0$. In order to challenge ν high, $\hat{\nu}$ must act in Subcase 4A(ii) at a stage $s_1 > s_0$. When $\hat{\nu}$ challenges ν high, it moves the current path to $T_{\hat{\nu},s_1}(\beta_{\hat{\nu},H}*1)$, stretches the trees and defines $\Gamma^{T_{\hat{\nu},s_1}(\beta_{\hat{\nu},H}*1*0)}(x_{\hat{\nu}}) = 0$. It sets $x_\nu = x_{\hat{\nu}} = x_\eta$ and passes $\beta_{\nu,H}$ to ν. Because $\beta_{\nu,H}$ is the least node on T_{ν,s_2} which is stretched, we have $T_{\nu,s_2}(\beta_{\nu,H}*0) = T_{\hat{\nu},s_2}(\beta_{\hat{\nu},H}*1*0)$. Hence the result holds for this high challenge.

If all the challenges issued by $\hat{\nu}$ at s_2 are met, then $\hat{\nu}$ begins to act in Subcase 4B of the high challenge. Suppose $\hat{\nu}$ calls a verification procedure at stage s_3. A similar argument shows that the high challenges issued by each of the cycles of the verification procedure have the required properties. Because a high challenged strategy $\hat{\nu}$ only issues more high challenges through Subcase 4A(ii) and 4B, this step completes the proof. □

Lemma 3.26. *For all x, if $x \notin B$, then $\Gamma^A(x) = 0$.*

Proof. As noted before Lemma 3.24, $x \in B$ if and only if $x = x_\eta$ for a P strategy η which reaches Case 4 of the P action. Therefore, if $x \notin B$, either x is never equal to x_η for a P strategy η or x is equal to x_η for some P strategy η but η is initialized before reaching Case 4 or x is permanently equal to x_η for a P strategy η but η never reaches Case 4.

First, suppose that x is never equal to x_η. At the end of stage x, we define $\Gamma^Y(x) = 0$ for all Y. Second, suppose $x = x_\eta$ but η is initialized at stage s after $x_\eta = x$ is defined. Without loss of generality, assume $s \geq x$. At the end of stage s, η is initialized so x is not longer of the form x_η. Therefore, we define $\Gamma^Y(x) = 0$ for all Y. It is clear that in either of these cases, $\Gamma^A(x) = 0$.

Third, suppose that x_η is defined to be x at stage s, η is never initialized after stage s and η never reaches Case 4. In this case, α_η is permanently defined at stage s and we set $\Gamma^{T_{\eta',s}(\alpha_\eta*0)}(x) = 0$. By Lemma 3.10, $T_{\eta',s}(\alpha_\eta * 0)$ is on the current path. We split into two subcases. For the first subcase, suppose η never calls a verification procedure. By Lemma 3.14, $T_{\eta',s}(\alpha_\eta * 0)$ remains on the current path forever, so $\Gamma^A(x) = 0$.

For the other subcase, suppose that η does call a verification procedure with $\sigma_0 = \alpha_\eta$ in Case 3 of the P action. Because η does not reach Case 4, this verification procedure does not finish but also does not end because of initialization. Therefore, some challenge in the series of challenges initiated by η is never met. We need to examine which strategies can move the current path below $T_{\eta',s}(\alpha_\eta * 0)$ and check that each time the current path is moved by a strategy challenged in this series of challenges, the strategy moving the current path makes new Γ definition for $x_\eta = x$ which remains on the current path unless another strategy which is also challenged in the series of challenges initiated by η moves the current path later. The last such strategy to move the current path will put up a Γ definition for $x_\eta = x$ using an oracle string which remains on the current path forever and hence is an initial segment of A.

When η calls the verification procedure in Step 3 of a P action at stage t_0 (to follow the notation of the verification procedure) with the witness x_η, no strategy to the left of η is ever eligible to act again since we assume this verification procedure is not removed by initialization. By Lemma 3.7, no strategy μ such that $\eta \subsetneq \mu$ is eligible to act after t_0 since we assume this procedure is never completed. Also, η initializes all strategies of lower priority, so they work higher on the trees.

If $\mu \subseteq \eta$ is a P strategy, then μ cannot move the current path without initializing η contrary to our assumption. An R strategy μ with μ with $\mu * L \subseteq \eta$ or $\mu * N \subseteq \eta$ does not move the current path, so we are left to consider R strategies μ with $\mu * H \subseteq \eta$.

If $\mu * H \subseteq \eta$, then μ could move the current path in Subcase 4A(ii) or 4B of a high challenge issued in the series of challenges initiated by η. In this case, when μ moves the current path, it initializes all strategies of lower priority than $\mu * L$ (including $\mu * L$). Therefore, these strategies are again forced to work higher on the tree than the new Γ definitions set up by μ (which we will examine below) and so they cannot move the path below the oracle string used by μ in its new Γ definition. Finally, notice that by Lemma 3.25, $x_\mu = x_\eta$ so the Γ definitions made by μ are for x_η.

We split the remainder of the proof into two cases which correspond to the two ways the current path can be moved below a string used as a Γ definition on x_η. Because one of the cycles in the verification procedure called by η does not end, we assume it is the n^{th} cycle. (We follow the notation of the verification procedure and the notation used in Lemma 3.15. In particular, we assume this n^{th} cycle starts at stage t_n by following a link from μ_{n-1} and that it defines μ_n and continues the verification procedure.) The first case is when η moves the current path in the n^{th} cycle but none of the strategies it challenges high move the current path after stage t_n. The second case is when at least one of the high challenged strategies such that $\nu * H \subseteq \mu_n$ does move the current path in Subcase 4A(ii) or 4B of the high challenge.

First, suppose that in the n^{th} cycle of the verification procedure called by η, none of the R strategies challenged high move the current path. For the n^{th} cycle, η defines $\Gamma^{T_{\mu_n,t_n}(\sigma_{n+1}*0)}(x_\eta) = 0$ and initializes all lower priority strategies. We claim that the current path continues to go through $T_{\mu_n,t_n}(\sigma_{n+1} *$

0) at all future stages (and hence $\Gamma^A(x_\eta) = 0$). The strategies to the left of η are never able to act after stage t_n (since they would initialize η), the strategies ν such that $\nu \subseteq \mu_n$ do not move the current path by assumption and the strategies ν such that $\mu_n * N \subseteq \nu$ or ν is to the right of μ_n in the tree of strategies are initialized at stage t_n by η and hence work higher on the trees than $T_{\mu_n,t_n}(\sigma_{n+1} * 0)$. Furthermore, because the n^{th} cycle for η never ends, one of the strategies $\nu \subseteq \mu_n$ never meets its low or high challenge. Therefore, the only strategies eligible to act after stage t_n are to the right of μ_n, satisfy $\nu \subseteq \mu_n$ or satisfy $\mu_n * N \subseteq \nu$ (since if μ_n ever took outcome $\mu_n * L$, it would follow the link back to η ending the n^{th} cycle). None of these strategies move the current path below $T_{\mu_n,t_n}(\sigma_{n+1} * 0)$, so it remains on the current path forever.

Second, suppose that some strategy ν which is high challenged in the series of challenges initiated by η does move the current path. By Lemma 3.25, when ν is challenged high at stage $t \geq t_n$, then $\Gamma^{T_{\nu,t}(\beta_{\nu,H}*0)}(x_\nu) = 0$ and $x_\nu = x_\eta$. (Remember that ν is challenged high in the series of challenges initiated by η, so it may not have been directly challenged high by η.) Whenever ν acts to move the current path, it puts up a new Γ definition for x_ν.

In particular, if ν acts in Subcase 4A(ii) at stage $s_1 > t$, it defines $\Gamma^{T_{\nu,s_1}(\beta_{\nu,H}*1*0)}(x_\eta) = 0$ and issues high challenges to μ such that $\mu * H \subseteq \nu$. If one of these high challenged strategies μ moves the current path, it takes over the Γ definition on $x_\mu = x_\nu = x_\eta$. If we return to ν at stage $s_2 > s_1$, then by Lemma 3.16, $T_{\nu,s_1}(\beta_{\nu,H} * 1 * 0) \subseteq T_{\nu,s_2}(\beta_{\nu,s_2} * 1 * 0)$, $T_{\nu,s_2}(\beta_{\nu,H} * 1 * 0)$ is on the current path and it remains on the current path unless ν calls a verification procedure in Subcase 4B of the high challenge. Therefore, if ν never calls this verification procedure, the computation $\Gamma^{T_{\nu,s_2}(\beta_{\nu,H}*1*0)}(x_\nu) = 0$ implies that $\Gamma^A(x_\eta) = 0$ as required.

Suppose ν does call a verification procedure in Subcase 4B of its high challenge. This verification procedure takes over the Γ definitions on x_ν. Either some cycle of this verification procedure doesn't finish or the verification procedure does finish. In the former case, suppose the n^{th} cycle is started but not finished. If none of the strategies challenged high by this cycle move the current path, then the argument given above in the similar case for η tells us that the Γ definition made by ν for x_ν in the n^{th} cycle implies $\Gamma^A(x_\nu) = \Gamma^A(x_\eta) = 0$ as required. If one of the strategies challenged high by the n^{th} cycle in ν's verification procedure does move the current path, then it takes over the Γ definition on x_ν (and we repeat this argument for that strategy).

Finally, consider the latter case in the previous paragraph: the verification procedure called by ν ends and ν meets its high challenge at stage $s_3 > s_2$. In this case, the current path is moved to pass through $T_{\nu,s_3}(\beta_{\nu,H} * 0)$. By Lemma 3.16, $T_{\nu,t}(\beta_{\nu,H} * 0) \subseteq T_{\nu,s_3}(\beta_{\nu,H} * 0)$ (recall that t was the stage at which ν was challenged high), so we have $\Gamma^{T_{\nu,s_3}(\beta_{\nu,H}*0)}(x_\nu) = 0$. The string $T_{\nu,s_3}(\beta_{\nu,H} * 0)$ remains on the current path unless another strategy moves the current path below this node. However, ν takes outcome $\nu * H$ at stage s_3, so it initializes all strategies to the right of $\nu * H$ and none of these strategies can move the current path below this node. If ν is the last strategy which is high challenged in the series of challenges initiated by η and which moves the current path, then

$T_{\nu,s_3}(\beta_{\nu,H} * 0)$ remains on the current path forever and we have $\Gamma^A(x_\nu) = 0$ as required. Otherwise, the next strategy which is in this series and which moves the current path takes over the Γ definition on x_η. The last such strategy to move the current path leaves a Γ definition on x_η for which the oracle string remains on the current path forever. □

We get the following result as an immediate consequence of Lemmas 3.24 and 3.26.

Lemma 3.27. *$\Gamma^A = B$, so $B \leq_T A$.*

Lemma 3.28. *All P requirements are met, so B is a noncomputable c.e. set.*

Proof. Fix a P requirement and let η be the strategy on the true path for this requirement. Let x_η be the final witness for η and assume it is defined by stage s. If $x_\eta \notin W_\eta$, then η takes outcome $\eta * W$ at every η stage after s and η never acts in Step 4 of the P action. Therefore, $x_\eta \notin B$ and P is won.

If $x_\eta \in W_\eta$, then there is an η stage after s at which η calls the verification procedure in Step 3. By Lemma 3.15, this procedure ends after finitely many η stages so η eventually reaches Step 4 and enumerates x_η into B winning P. □

To complete our proof, we give the computation lemmas showing that A has minimal *wtt* degree.

Lemma 3.29. *If $\eta * N$ is on the true path, then $[\eta]^A$ is not total.*

Proof. Fix an η stage s such that η takes outcome $\eta * N$ at every η stage after s. Because η takes outcome $\eta * N$ at stage s, either η is acting in Subcase 4B of a high challenge or η is low challenged. We consider each of these possibilities separately.

Assume that η has been high challenged by $\hat{\eta}$ before stage s and that η acts in Subcase 4B of the high challenge for the first time at stage s. At the previous η stage $t < s$, η must have acted in Subcase 4A(ii) of the high challenge and defined the parameter w_η. As in the proof of Lemma 3.16, $T_{\eta,s}(\beta_{\eta,H} * 1 * 0) \subseteq A_{\eta,s}$ and the length of this node is longer than the use of $[\eta]$ on w_η. The current path is not moved below $T_{\eta,s}(\beta_{\eta,H} * 1 * 0)$ unless η moves it because it sees $[\eta]^{T_{\eta,s}(\beta_{\eta,H} * 1 * 0)}(w_\eta)$ converge. However, if η sees this computation converge, it moves the current path and takes outcome $\eta * H$, contrary to our assumption. Therefore, η never sees this computation converge and the current path never moves below $T_{\eta,s}(\beta_{\eta,H} * 1 * 0)$. Because the use of $[\eta]$ on w_η is less than the length of $T_{\eta,s}(\beta_{\eta,H} * 1 * 0)$ and this node remains forever on the current path, we have that $[\eta]^A(w_\eta)$ diverges and hence $[\eta]^A$ is not total.

Assume that η is low challenged by $\hat{\eta}$ at stage $t < s$ and s is the first η stage after t. By Lemma 3.19 (and because η never meets this low challenge), $T_{\eta,s}(\beta_{\eta,L} * 1)$ remains on the current path forever. By Lemma 3.23, there is a stage $u > s$ and a string γ such that $\beta_{\eta,L} * 1 \subseteq \gamma$, $T_{\eta,u}(\gamma)$ has reached its limit, $U(T_{\eta,u}(\gamma)) = G_\eta * L$, $T_{\eta,u}(\gamma) \subseteq A$ and the length of $T_{\eta,u}(\gamma)$ is longer than the $[\eta]$ use of any number in X_η. If $[\eta]^{T_{\eta,u}(\gamma)}(x)$ converges for each $x \in X_\eta$, then eventually η sees these computations and either meets its low challenge (taking outcome $\eta * L$) or finds a new high split (taking outcome $\eta * H$). Either option violates our assumptions and hence there must be at least one number $x \in X_\eta$ for which $[\eta]^{T_{\eta,u}(\gamma)}(x)$ diverges. Because $T_{\eta,u}(\gamma) \subseteq A$ and the length of $T_{\eta,u}(\gamma)$ is longer than the $[\eta]$ use of each $x \in X_\eta$, there must be at least one number $x \in X_\eta$ for which $[\eta]^A(x)$ diverges. Therefore, $[\eta]^A$ is not total. $\square$

Lemma 3.30. *Let η be an R strategy such that $\eta * L$ is on the true path. If $[\eta]^A$ is total, then $[\eta]^A$ is computable.*

Proof. Let s be a stage such that α_η is permanently defined by s and η never takes outcome $\eta * H$ after s. By Lemma 3.20 (since $\eta * L$ is never initialized after s), η meets all low challenges issued after stage s. Furthermore, if $\mu * L \subseteq \eta$, then μ meets all low challenges after stage s and if $\mu * H \subseteq \eta$, then μ meets all high challenges after s.

To calculate $[\eta]^A(x)$, let $t_0 > s$ be an η stage and let γ_0 be a string such that η takes outcome $\eta * L$ at t_0, $T_{\eta,t_0}(\gamma_0) \subseteq A_{\eta,t_0}$, $U(T_{\eta,t_0}(\gamma_0)) = G_\eta * L$ and $[\eta]_{t_0}^{T_{\eta,t_0}(\gamma_0)}(x)$ converges. (Such t_0 and η_0 must exist by Lemma 3.23 since $[\eta]^A$ is total.) We claim that $[\eta]^A(x) = [\eta]_{t_0}^{T_{\eta,t_0}(\gamma_0)}(x)$.

To prove the claim, we need to examine how the current path could be moved below $T_{\eta,t_0}(\gamma_0)$. Suppose μ moves the current path below this node after stage t_0. We cannot have $\mu <_L \eta$ (since these do not act after stage s), $\eta <_L \mu$ or $\eta * N \subseteq \mu$ (since these strategies are initialized at t_0). Suppose $\mu \subsetneq \eta$. μ cannot be a P strategy, since it would initialize η when it moved the path. If μ is an R strategy, then it can only move the current path when it is high challenged. If $\mu * L \subseteq \eta$ or $\mu * N \subseteq \eta$, then μ would initialize η when it moved the current path. Therefore, assume $\mu * H \subseteq \eta$. By Lemma 3.2, μ is not high challenged when η acts at stage t_0. Therefore, it must become high challenged later before moving the current path. However, if γ_μ is such that $T_{\mu,t_0}(\gamma_\mu) = T_{\eta,t_0}(\gamma_0)$, then $T_{\mu,t_0}(\gamma_\mu)$ is already μ high splitting. Therefore, any movement of the current path by μ in a high challenge would be above this node. It follows that no strategy $\mu \subsetneq \eta$ moves the current path below this node after stage t_0.

We also cannot have $\mu = \eta$ since η can only be high challenged by strategies extending $\eta * H$ and no such strategy is eligible to act after stage s. Therefore, the only strategies μ which could move the current path below $T_{\eta,t_0}(\gamma_0)$ after stage t_0 satisfy $\eta * L \subseteq \mu$.

Let μ be the first strategy which causes such a movement in the current path below $T_{\eta,t_0}(\gamma_0)$ after stage t_0 and let $u_1 > t_0$ be the stage at which it moves the current path. To be specific with our notation, we assume that μ is a P strategy which is just calling a verification procedure. However, similar arguments handle the cases when μ is an R strategy acting in Subcase 4A(ii) or 4B of a high challenge and when μ is either a P or R strategy which is returning to a previously called verification procedure.

In this situation, μ moves the current path from $T_{\mu',u_1}(\alpha_\mu * 0)$ to $T_{\mu',u_1}(\alpha_\mu * 1)$ and defines $\beta_{\eta,L}$ to be the string such that the current path moved from $T_{\eta,u_1}(\beta_{\eta,L} * 0)$ to $T_{\eta,u_1}(\beta_{\eta,L} * 1)$. Because this movement is below $T_{\eta,t_0}(\gamma_0)$, we have $T_{\eta,u_1}(\beta_{\eta,L} * 0) \subseteq T_{\eta,t_0}(\gamma_0)$. If $[\eta]^{T_{\eta,u_1}(\beta_{\eta,L})}(x)$ converges, then we must have $[\eta]^{T_{\eta,u_1}(\beta_{\eta,L})}(x) = [\eta]^{T_{\eta,t_0}(\gamma_0)}(x)$ and hence this movement of the current path does not effect our computation procedure. Therefore, assume that $[\eta]^{T_{\eta,u_1}(\beta_{\eta,L})}(x)$ diverges. In this case, $x \in X_\eta$, so μ challenges η low and any link which is placed by μ is from a strategy ν such that $\eta \subseteq \nu$.

By the comments in the first paragraph of this proof, the challenges issued by μ to higher priority strategies than η are eventually met and η eventually meets the low challenge. Let $t_1 > u_1$ be the stage at which η meets this low challenge. At this stage, η has found a string γ_1 such that $T_{\eta,t_1}(\gamma_1) \subseteq A_{\eta,t_1}$, $U(T_{\eta,t_1}(\gamma_1)) = G_\eta * L$ and $[\eta]_{t_1}^{T_{\eta,t_1}(\gamma_1)}(x)$ converges and is equal to $[\eta]_{t_0}^{T_{\eta,t_0}(\gamma_0)}(x)$. We can now repeat this argument. Let μ_2 be the first strategy which moves the current path below $T_{\eta,t_1}(\gamma_1)$ at some stage $u_2 \geq t_1$. μ_2 must satisfy $\eta * L \subseteq \mu_2$. Just as above, there would be a stage $t_2 > t_1$ and a string γ_2 such that $T_{\eta,t_2}(\gamma_2)$ is on the new current path A_{η,t_2}, $U(T_{\eta,t_2}(\gamma_2)) = G_\eta * L$ and $[\eta]_{t_2}^{T_{\eta,t_2}(\gamma_2)}(x)$ converges and is equal to $[\eta]_{t_1}^{T_{\eta,t_1}(\gamma_1)}(x) = [\eta]_{t_0}^{T_{\eta,t_0}(\gamma_0)}(x)$. Because $[\eta]$ is a *wtt* procedure and because the current path settles down on longer and longer initial segments, these path movements below the use of $[\eta]$ on x can only happen finitely often. Therefore, by induction we get that $[\eta]_{t_0}^{T_{\eta,t_0}(\gamma_0)}(x) = [\eta]^A(x)$. □

Lemma 3.31. *Let η be an R strategy such that $\eta * H$ is on the true path. If $[\eta]^A$ is total, then $A \leq_{wtt} [\eta]^A$.*

Proof. Fix η such that $\eta * H$ is on the true path and $[\eta]^A$ is total. Let s_λ be a stage such that $T_{\eta,s_\lambda}(\lambda)$ has reached its final value (and hence η is never initialized after s_λ) and $U(T_{\eta,s_\lambda}(\lambda)) = G_\eta * H$. We have $T_{\eta,s_\lambda}(\lambda) \subseteq A_{\eta,s_\lambda}$. By Lemma 3.20, $T_{\eta,s_\lambda}(\lambda)$ has reached its final value and $T_{\eta,s_\lambda}(\lambda) = T_\eta(\lambda) \subseteq A$. We define a Turing procedure Δ_η^X for any oracle X, show that if $X = [\eta]^A$, then $\Delta_\eta^X = A$, and finally show that Δ_η has computably bounded use for any oracle and hence is a *wtt* procedure.

Fix any oracle set X. We define Δ_η^X by defining a (possibly finite) sequence of strings $\lambda = \sigma_0 \subseteq \sigma_1 \subseteq \cdots$ and stages $s_\lambda = t_0 < t_1 < \cdots$ using oracle questions answered by X. At each stage t_i we will have the following properties: $T_{\eta,t_i}(\sigma_i) \subseteq A_{\eta,t_i}$ and $U(T_{\eta,t_i}(\sigma_i)) = G_\eta * H$ (and hence $T_{\eta,t_i}(\sigma_i)$ has reached its final value by Lemma 3.22). The comments in the first paragraph explain why these properties hold for σ_0 and t_0. Once σ_i and t_i are calculated, let $l_i =$ the length of $T_{\eta,t_i}(\sigma_i)$ and set $\Delta_\eta^X \upharpoonright l_i = T_{\eta,t_i}(\sigma_i)$.

Assume we have used X to calculate σ_i and t_i. Because $U(T_{\eta,t_i}(\sigma_i)) = G_\eta * H$, there is a splitting witness x_i such that $[\eta]_{t_i}^{T_{\eta,t_i}(\sigma_i*0)}(x_i)$ and $[\eta]_{t_i}^{T_{\eta,t_i}(\sigma_i*1)}(x_i)$ converge and are unequal. Check which computation agrees with $X(x_i)$ and set $\sigma_{i+1} = \sigma_i * 0$ or $\sigma_i * 1$ so that $[\eta]_{t_i}^{T_{\eta,t_i}(\sigma_{i+1})}(x_i) = X(x_i)$. Wait for a stage t_{i+1} such that $T_{\eta,t_{i+1}}(\sigma_{i+1}) \subseteq A_{\eta,t_{i+1}}$ and $U(T_{\eta,t_{i+1}}(\sigma_{i+1})) = G_\eta * H$. If we never see such a stage, then Δ_η^X diverges on all inputs $\geq l_i$. If we do see such a stage, then let $l_{i+1} =$ the length of $T_{\eta,t_{i+1}}(\sigma_{i+1})$ and set $\Delta_\eta^X \upharpoonright l_{i+1} = T_{\eta,t_{i+1}}(\sigma_{i+1})$. This completes the description of Δ_η.

Next, we check that if $X = [\eta]^A$, then $\Delta_\eta^X = A$. To prove this fact, we show by induction on i that σ_i exists and $T_{\eta,t_i}(\sigma_i) \subseteq A$. When $i = 0$, this is clear. Assume that σ_i is defined and $T_{\eta,t_i}(\sigma_i) \subseteq A$. Let x_i be a number such that $[\eta]^{T_{\eta,t_i}(\sigma_i * 0)}(x_i)$ and $[\eta]_{t_i}^{T_{\eta,t_i}(\sigma_i * 1)}(x_i)$ converge and are unequal. By Lemma 3.22 and the proof of Lemma 3.23, we know that $T_{\eta,t_i}(\sigma_i)$ has reached its final value. Furthermore, we know that the values of $T_{\eta,t_i}(\sigma_i * 0)$ and $T_{\eta,t_i}(\sigma_i * 1)$ can change at most finitely often after stage t_i, that these changes are due to stretching, and that the stretched values of these nodes always extended their prestretched values. Therefore, one of the strings $T_{\eta,t_i}(\sigma_i * 0)$ or $T_{\eta,t_i}(\sigma_i * 1)$ has to be an initial segment of A and because $X = [\eta]^A$, σ_{i+1} must be defined such that $T_{\eta,t_i}(\sigma_{i+1}) \subseteq A$. Eventually, the current path has to run through $T_{\eta,t_i}(\sigma_{i+1})$ (although this node may have been stretched by the time it does) and because $\eta * H$ is on the true path, there must be a stage $t_{i+1} > t_i$ such that $T_{\eta,t_i}(\sigma_{i+1}) \subseteq T_{\eta,t_{i+1}}(\sigma_{i+1}) \subseteq A_{\eta,t_{i+1}}$ and $U(T_{\eta,t_{i+1}}(\sigma_{i+1})) = G_\eta * H$. Therefore, we eventually define t_{i+1} and have $T_{\eta,t_{i+1}}(\sigma_{i+1}) \subseteq A$ as required.

Finally, we show that the use of Δ_η is computably bounded for all oracles and hence it is a *wtt* procedure. To bound the use of this procedure on input m, calculate as follows. Wait for a stage $t \geq s_\lambda$ such that $t > m$ and there is a string σ such that $T_{\eta,t}(\sigma) \subseteq A_{\eta,t}$, $U(T_{\eta,t}(\sigma)) = G_\eta * H$, $T_{\eta,t}(\sigma)$ becomes high splitting at t and the length of $T_{\eta,t}(\sigma)$ is greater than m. (Because $[\eta]^A$ is total and $\eta * H$ is on the true path such a pair σ and t must exist.) Let k be the maximum of all $[\eta]$ high splitting witnesses seen by η during the course of the construction up to stage t. We claim that the use of Δ_η on input m for any oracle X is bounded by k.

To prove our claim, let X be any oracle and let σ_i and t_i be the last pair defined by the procedure Δ_η^X by the stage t indicated above for use calculation on m. (Because σ_0 and t_0 are defined at stage s_λ and $t \geq s_\lambda$, $i \geq 0$ is defined.) Let x_i be the splitting witness for this pair of strings, let σ_{i+1} be either $\sigma_i * 0$ or $\sigma_i * 1$ depending on which gives the computation that agrees with $X(x_i)$ and let l_i denote the length of $T_{\eta,t_i}(\sigma_i)$. Because the string σ_i is defined by stage t, we know $k \geq x_i$. Furthermore, all the splitting witnesses which have been used to determine σ_i are $\leq k$. If $m < l_i$, then Δ_η^X has already converged on m and has use $\leq k$ since the splitting witnesses (which are the only values of X which we consult) are all $\leq k$.

Assume $m \geq l_i$. First, we claim that at stage t, $U(T_{\eta,t}(\sigma_{i+1})) = G_\eta * L$. This follows because we only look for high splits along the current path. Therefore, if $U(T_{\eta,t}(\sigma_{i+1})) = G_\eta * H$, then at some stage u between t_i and t, we had $T_{\eta,u}(\sigma_{i+1}) \subseteq A_{\eta,u}$ and it became high splitting. However, in this case, $t_{i+1} = u \leq t$, contradicting the fact that t_{i+1} is not yet defined at stage t.

Second, we claim that at stage t, $T_{\eta,t}(\sigma_{i+1})$ is not on the current path. This follows because at stage t, we just found that a new node $T_{\eta,t}(\sigma)$ on the current path which is high splitting. Furthermore, $T_{\eta,t}(\sigma)$ has length $> m$. Hence $T_{\eta,t}(\sigma)$ is not equal to $T_{\eta,t}(\sigma_i)$ (which has length $\leq m$), so $t > t_i$. Thus, if $T_{\eta,t}(\sigma_{i+1})$ were along the current path as well, then it would be high splitting and we would have defined t_{i+1} by stage t.

Therefore, we know that at stage t, $T_{\eta,t}(\sigma_{i+1})$ is not on the current path and it has state $G_\eta * L$. There are now two possibilities. First, it is possible that there is never a stage t_{i+1}. In this case, Δ^X_η never consults the oracle again (and so has use bounded by k) and diverges on m. Second, it is possible that there is a stage $t_{i+1} > t$. In this case, some P or R strategy must move the current path so that it passes through $T_{\eta,t}(\sigma_{i+1})$ at a stage $u > t$. Because t is an η stage at which η takes outcome $\eta * H$, all strategies to the right of $\eta * H$ in the tree of strategies are initialized at t and work higher on the trees. By Lemma 3.2, if $\nu * H \subseteq \eta$, then ν is not high challenged at stage t. Therefore, the first strategy to move the current path so that it passes through $T_{\eta,t}(\sigma_{i+1})$ must satisfy $\eta * H \subseteq \mu$. Let $u > t$ be the stage when μ moves the current path. Because $\eta * H \subseteq \mu$, $U(T_{\eta,u}(\sigma_i)) = G_\eta * H$ and $T_{\eta,u}(\sigma_{i+1}) = G_\eta * L$ (before it is stretched), $T_{\eta,u}(\sigma_{i+1})$ is stretched to have long length when μ moves the current path. In particular, $T_{\eta,u}(\sigma_{i+1})$ has length longer than m. Therefore, when $T_{\eta,u}(\sigma_{i+1})$ later reaches state $G_\eta * H$ and t_{i+1} is defined, we set $l_{i+1} =$ the length of $T_{\eta,t_{i+1}}(\sigma_{i+1})$, so $l_{i+1} > m$ and $\Delta^X_\eta \upharpoonright l_{i+1} = T_{\eta,t_{i+1}}(\sigma_{i+1})$. Furthermore, we know that $T_{\eta,t_{i+1}}(\sigma_i * 0)$ extends $T_{\eta,t_i}(\sigma_i * 0)$ and $T_{\eta,t_{i+1}}(\sigma_i * 1)$ extends $T_{\eta,t_i}(\sigma_i * 1)$. Therefore, $x_i \leq k$ is still a splitting witness for these two nodes. Hence, we do not need any more of the oracle X to calculate $\Delta^X_\eta \upharpoonright l_{i+1}$. This completes the proof that the use is bounded by k. □

This concludes the proof of the main theorem, Theorem 1.1.

Chapter 4

Limiting results

In this chapter, we prove Theorems 1.2 and 1.3 giving limitations on possible extensions of Theorem 1.1. For convenience, we restate these theorems here.

Theorem 1.2. *No c.e. Turing degree can contain a set of which is wtt-minimal.*

Theorem 1.3. *Let V be a promptly simple c.e. set and let A be a Δ^0_2 set such that $A \geq_T V$. There exists a c.e. set B such that $0 <_T B \leq_{wtt} A$.*

To prove Theorem 1.2, we need to show that for any set A of c.e. degree, there is a set B such that $\emptyset <_T B <_{wtt} A$. In Section 4.1, we prove that such a set B cannot be obtained uniformly from A. In Section 4.2, we prove Theorem 1.2 under the assumption that A has an almost c.e. approximation (which is defined in that section) and we develop a closely related method for approximating general sets of c.e. Turing degree. We complete the proof of Theorem 1.2 in Section 4.3 and we prove Theorem 1.3 in Section 4.4.

4.1 Uniformity issues

Consider how we might try to alter the proof of Theorem 1.1 to make the set A have c.e. Turing degree. As before we build A via a Δ^0_2 approximation A_s and our R requirements (to make A have minimal *wtt* degree) remain the same.

To ensure that A has c.e. Turing degree, we build a modulus function for A. Recall that a total function f is a modulus function for a Δ^0_2 approximation A_s to A if the following condition holds for every x.

$$\forall s \geq f(x)\, \forall y \leq x\, (y \in A_s \Leftrightarrow y \in A)$$

In other words, the Δ^0_2 approximation has settled to its limiting values on all numbers up to x by stage $f(x)$. By the Modulus Lemma, A has c.e. Turing degree if and only if there is a Δ^0_2 approximation A_s to A such that A can compute a modulus for this approximation. Therefore, rather than directly constructing a c.e. set B as in the proof of Theorem 1.1, we can build a Turing functional Φ such that Φ^A is a modulus function for our approximation A_s. To ensure that A is not computable, we need to satisfy diagonalization requirements P_e for each index e (described below).

We begin with a proposition that says we can carry out such a construction as long as we consider only a single R requirement. The proof of this proposition is similar to (but considerably simpler than) the proof of Theorem 1.1, so we merely sketch the argument. To simplify the technical details in this sketch, we will be somewhat informal about the diagonalization requirements P_e. We view P_e as requiring that we respond to some Σ^0_1 event dictated by W_e (namely a designated witness entering W_e) by moving the approximation A_s at a predetermined place. More formally, we would define an auxiliary c.e. set B and a Turing functional Γ such that $\Gamma^A = B$ and our requirement P_e would be $B \neq \overline{W_e}$. To avoid complicated our sketch with standard details for constructing Γ and B, we limit our P_e strategies to moving the current path and forbidding cones.

Proposition 4.1. *For any wtt-functional* $[e]$*, we can build a non-computable set* A *of c.e. Turing degree such that if* $[e]^A$ *is total, then either* $[e]^A$ *is computable or* $[e]^A \geq_{wtt} A$*.*

Proof. We build a computable approximation A_s to A and a Turing functional Φ such that Φ^A is a modulus function for this approximation. Because we are only concerned with the R requirement given by $[e]$, we build a single sequence of computable trees $T_{e,s}$ and hence we drop the index e on these trees. To build T_s, we attempt to find $[e]$-splits along the current path A_s and we will use stretching when we need to verify computations through low challenges. As usual, we obtain $A \leq_{wtt} [e]^A$ if the nodes of T_s along A_s are all eventually in the high state, while $[e]^A$ will be computable if sufficiently long nodes remain in the low state permanently.

Later, we will want to use the fact that this construction is uniform in the index e. To ensure this uniformity, we need to allow parts of the trees T_s to be in a non-total state while we wait for low challenges to be met.

The basic strategy for P_e is to choose a node α such that $T_s(\alpha)$ and $T(\alpha * 0)$ are on the current path and a large diagonalizing witness x. If x later enters W_e, P_e would like to move the current path from $T_s(\alpha * 0)$ to $T_s(\alpha * 1)$ and forbid the cone above $T_s(\alpha * 0)$ so that this movement is permanent. If $T_s(\alpha)$ is in the high $[e]$-state, then there is no problem with immediately forbidding $T_s(\alpha * 0)$ as there is only one R requirement. However, if $T_s(\alpha)$ is in the low $[e]$-state, then we would like to stretch $T_s(\alpha * 1)$ to have length longer than any oracle $T_s(\beta)$ with $\alpha * 0 \subseteq \beta$ used in a computation $[e]^{T_s(\beta)}(y)$ we have seen so far and challenge P_e to verify these computations using the new value of $T_s(\alpha * 1)$ as the oracle. (Below, we refer to this process simply as stretching $T_s(\alpha * 1)$.) There are three possible outcomes: we verify all of the previous computations allowing the construction to continue in the low state using $T_s(\alpha * 1)$ in place of $T_s(\alpha * 0)$ giving us permission to forbid $T_s(\alpha * 0)$, we find a computation allowing us to put up a new high split and make progress towards making the sequence of trees high splitting (and hence abandon this attempt at satisfying

P_e), or we have some computation which is never verified ensuring that $[e]^A$ is not total as long as $T_s(\alpha * 1)$ remains on the current path.

The basic strategy for defining Φ^A is to choose strings δ such that $T_s(\delta)$ is on the current path A_s and define $\Phi^{T_s(\delta)}(|T_s(\delta')|) = s$ at stage s. This definition makes a promise that if $T_s(\delta)$ is an initial segment of A, then the approximation to A never changes below $|T_s(\delta')|$ after stage s. In other words, if we ever move the current path away from $T_s(\delta')$ at a future stage, then $T_s(\delta)$ must be immediately forbidden. For this strategy to succeed, we need to eventually choose strings δ of arbitrarily long length for which we make such definitions and $T_s(\delta)$ is an initial segment of A.

There is a significant conflict between the strategies for P_e and for defining Φ^A. Suppose P_e has chosen a node α with $T_s(\alpha)$ in the low state and would like to diagonalize at $T_s(\alpha)$ if x later enters W_e. While waiting for x to enter W_e, we need to make definitions for Φ^A involving strings δ extending $\alpha * 0$. For example, we may define $\Phi^{T_{s_0}(\alpha*0*0)}(|T_{s_0}(\alpha*0)|) = s_0$ at some stage $s_0 > s$. If x enters W_e at stage $s_1 > s_0$ (with $T_{s_1}(\alpha)$ still in the low state), then P_e wants to move the current path from $T_{s_1}(\alpha * 0)$ to $T_{s_1}(\alpha * 1)$ and freeze (but not forbid) $T_{s_1}(\alpha * 0)$. Until the low challenge is met, we cannot forbid $T_{s_1}(\alpha * 0)$ because we may need to use an extension of $T_{s_1}(\alpha * 0)$ as half of a new high split if we get a different computation using (the stretched) $T_{s_1}(\alpha * 1)$ as oracle. However, as soon as we move the current path away from $T_{s_1}(\alpha * 0) = T_{s_0}(\alpha * 0)$, the promise accompanying the definition of $\Phi^{T_{s_0}(\alpha*0*0)}(|T_{s_0}(\alpha * 0)|) = s_0$ requires us to immediately forbid $T_{s_0}(\alpha * 0 * 0) = T_{s_1}(\alpha * 0 * 0)$. But, we may well have seen computations using oracles extending $T_{s_0}(\alpha * 0 * 0)$ so we are prohibited from forbidding this node until the computations are verified.

To solve this conflict, we modify the P_e strategy to issue a sequence of low challenges allowing it to move the current path at a decreasing sequence of nodes eventually culminating in moving the current path at the diagonalizing node. Let s be a stage at which P_e sees its witness x_e enter W_e and wants to move the current path from $T_s(\alpha_e * 0)$ to $T_s(\alpha_e * 1)$ where $T_s(\alpha_e)$ is in the low state. For $k \leq s$, let γ_k be the string of length k such that $T_s(\gamma_k)$ is on the current path A_s. For simplicity of notation, we assume $\gamma_k = 0^k$ and we assume that we have not looked at any computations using an oracle extending $T_s(\gamma_s)$. Some of these strings γ_k may have been used to make Φ definitions of the form $\Phi^{T_s(\gamma_{k+1})}(|T_s(\gamma_k)|) = \Phi^{T_s(\gamma_k*0)}(|T_s(\gamma_k)|) \leq s$. Again, to simplify the notation, assume that strings of the form $\gamma_{2\ell}$ have been used in the Φ definitions and that the stage s is even. Throughout the description below, we assume no new high splits are found below $T_s(\alpha_e)$ and so all the nodes mentioned retain their values unless they are stretched.

P_e begins by stretching $T_s(\gamma_{s-2} * 1)$ and moving the current path from $T_s(\gamma_{s-2} * 0) = T_s(\gamma_{s-1})$ to (the stretched) $T_s(\gamma_{s-2} * 1)$. Since s is even, we have defined $\Phi^{T_s(\gamma_s)}(|T_s(\gamma_{s-1})|) = s$, and hence must forbid $T_s(\gamma_s)$. However, this action is fine because we have not seen any computations using oracles extending $T_s(\gamma_s)$. Notice that $T_s(\gamma_{s-2} * 0) = T_s(\gamma_{s-1})$ is not forbidden because $T_s(\gamma_{s-2} * 0 * 1) = T_s(\gamma_{s-1} * 1)$ remains a viable extension of this node.

P_e challenges $[e]^{T_s(\gamma_{s-2}*1)}$ to verify all of the computations which used oracles extending $T_s(\gamma_{s-2} * 0) = T_s(\gamma_{s-1})$. Because $T_s(\gamma_{s-2} * 1)$ was stretched, we do not need to look at any oracles extending $T_s(\gamma_{s-2} * 1)$ during this verification process. Furthermore, we set $\Phi^{T_s(\gamma_{s-2}*1*0)}(|T_s(\gamma_{s-2} * 1)|) = s$ to make progress on the definition of Φ^A. While waiting for these computations to converge, we launch versions of each P requirement to work in the cone above $T_s(\gamma_{s-2}*1*0)$. Because these versions of the P requirements can assume $[e]^A$ will be partial (as we haven't verified the low challenge yet), they can immediately forbid nodes when they need to diagonalize. Therefore, if the low challenge is not met, $[e]^A$ will be partial and we will still guarantee that A is not computable and Φ^A is a modulus function (as we also continue to make definitions for Φ^A above $T_s(\gamma_{s-2} * 1)$).

Assume that the low challenge is eventually met at stage $s_1 > s$. At this point, all of the computations which used oracles extending $T_s(\gamma_{s-2} * 0)$ are now held by $T_{s_1}(\gamma_{s-2} * 1)$ and therefore, we have permission to forbid $T_s(\gamma_{s-2} * 0) = T_{s_1}(\gamma_{s-2} * 0) = T_{s_1}(\gamma_{s-1})$. P_e now moves the current path for the second time as follows. We stretch $T_{s_1}(\gamma_{s-3} * 1) = T_s(\gamma_{s-3} * 1)$ to have long length and move the current path from $T_s(\gamma_{s-3} * 0) = T_s(\gamma_{s-2})$ to $T_s(\gamma_{s-3} * 1)$. (These nodes have retained their values at s_1 except for the stretching.) Because $\Phi^{T_s(\gamma_{s-2}*1*0)}(|T_s(\gamma_{s-2} * 1)|) = s$ and we moved the path below $T_s(\gamma_{s-2} * 1)$, this action requires us to forbid the cone above $T_s(\gamma_{s-2} * 1 * 0)$ which is allowed because we did not look at any computations in this cone during the low challenge. However, the node $T_s(\gamma_{s-2} * 1)$ remains viable and since it holds the computations originally obtained above $T_s(\gamma_{s-2} * 0) = T_s(\gamma_{s-1})$, we can forbid the cone above $T_s(\gamma_{s-1})$ as well.

We now issue the second low challenge for $[e]^{T_{s_1}(\gamma_{s-3}*1)}$ to verify the computations which have been obtained using oracles extending $T_{s_1}(\gamma_{s-3} * 0) = T_s(\gamma_{s-2})$. The argument repeats exactly as above. Because $T_{s_1}(\gamma_{s-3} * 1)$ was stretched, we do not need to look at computations involving nodes extending $T_{s_1}(\gamma_{s-3} * 1)$ during the verification. We define $\Phi^{T_{s_1}(\gamma_{s-3}*1*0)}(|T_{s_1}(\gamma_{s-3} * 1)|) = s_1$ to extend the definition of Φ^A. Each P strategy will start a version working in the cone above $T_{s_1}(\gamma_{s-3} * 1 * 0)$ assuming that the low challenge is never verified. If we never verify the low computations, then we win because $[e]^A$ is partial and we still ensure A is not computable and Φ^A is a modulus function. If the low challenge is met at $s_2 > s_1$, then we have permission to forbid $T_s(\gamma_{s-2}) = T_{s_2}(\gamma_{s-2})$ as the computations are now held by $T_{s_1}(\gamma_{s-3} * 1)$.

The pattern now repeats, but there is one final comment to make about this process. We stretch $T_{s_2}(\gamma_{s-4} * 1) = T_s(\gamma_{s-4} * 1)$ and move the current path from $T_{s_2}(\gamma_{s-4} * 0) = T_s(\gamma_{s-3})$ to (the stretched) $T_{s_2}(\gamma_{s-4} * 1)$. Because s was an even stage, $s - 2$ is even and hence at stage s, we had already defined $\Phi^{T_s(\gamma_{s-2})}(|T_s(\gamma_{s-3})|) \leq s$. Therefore, moving the path away from $T_s(\gamma_{s-3})$ requires us to immediately forbid $T_s(\gamma_{s-2})$. However, we have just obtained permission to forbid $T_s(\gamma_{s-2})$. In general, our method of working down the current path in this inductive manner is set up to give us permission to forbid the strings required by the definitions of Φ.

Continuing in this manner and using the fact that α_e is one of the γ_k nodes (and assuming the low challenges are all met), we eventually arrive at a stage u

such that (our stretched) $T_u(\alpha_e * 0 * 1)$ holds all of the computations originally seen with oracles extending $T_s(\alpha_e * 0 * 0)$. At this point, we have solved our original conflict as we have permission to stretch $T_u(\alpha_e * 1) = T_s(\alpha_e * 1)$, move the current path from $T_u(\alpha_e * 0) = T_s(\alpha_e * 0)$ to $T_u(\alpha_e * 1)$ and immediately forbid $T_u(\alpha_e * 0 * 0) = T_s(\alpha_e * 0 * 0)$. We issue one last low challenge for $[e]^{T_u(\alpha_e * 1)}$ to verify the computations using oracles extending $T_u(\alpha_e * 0) = T_s(\alpha_e * 0)$. If this low challenge is never met, our construction succeeds because of the versions of P strategies working above $T_u(\alpha_e * 1)$ under the assumption that $[e]^A$ is partial, and if the low challenge is met, we win P_e by forbidding $T_s(\alpha_e * 0)$.

This completes our informal description of a P_e strategy which guesses T_s is eventually permanently in the low state. As there are no additional conflicts, it is straightforward to turn this description into a formal argument. □

Corollary 4.2. *There is no wtt-functional* $[e]$ *such that for every noncomputable set* A *of c.e. Turing degree,* $[e]^A$ *is total and* $\emptyset <_T [e]^A <_{wtt} A$.

By Corollary 4.2, we cannot use a single *wtt*-procedure to uniformly produce witnesses to Theorem 1.2. However, we could ask about other forms of uniformity. Is there is a method of indexing sets of c.e. Turing degree and a partial computable function f such that for a noncomputable set A with index e (in our indexing method), we are guaranteed that $f(e)$ is defined and $\emptyset <_T [f(e)]^A <_{wtt} A$? We end this section by showing that this is not possible for two natural methods of indexing sets of c.e. Turing degree.

Let Z_e denote the e-th Σ^0_2 set with the approximation $Z_{e,s}$ given by the e-th Σ^0_2 predicate. We say $\langle e, i\rangle$ is an c.e. degree index for a Δ^0_2 set A of c.e. degree if $A = Z_e$ and Φ^A_i is a modulus function for $A_s = Z_{e,s}$. The proof of Proposition 4.1 is uniform relative to this indexing method in the sense that the proof produces a computable function $g(r)$ such that $g(r) = \langle e, i\rangle$ where $\langle e, i\rangle$ is a c.e. degree index for a noncomputable set A of c.e. Turing degree such that if $[r]^A$ is total, then either $[r]^A$ is computable or $A \leq_{wtt} [r]^A$.

Of course, we can give other types of indices for a set A of c.e. degree. For example, we could index A by $\langle e, k, i, j\rangle$ where $A = Z_e$, $A = \Phi^{W_k}_i$ and $W_k = \Phi^A_j$. By the proof of the Modulus Lemma, we can uniformly translate between indices of these two different forms. Therefore, the results below apply to this type of indexing as well.

To get our strong non-uniformity result, we will use the relativized version of the Recursion Theorem with Parameters which says that for any computable function $f(x, y)$, there is a computable function $n(y)$ such that for all oracles A and for all y, $\Phi^A_{n(y)} = \Phi^A_{f(n(y),y)}$ as partial functions. (See Soare [34] Chapters II and III.) Moreover, by the proof of this theorem, these functions have identical use functions. We will use this property to give a version of the recursion theorem for *wtt*-indices.

Because we will shift between different types of indices, recall that an index for a *wtt*-functional is a pair $\langle e, i\rangle$ where e is an index for a Turing functional Φ_e and i is an index for a partial computable function φ_i. We compute $[\langle e, i\rangle]^A(n)$ by first calculating $\varphi_i(0), \ldots, \varphi_i(n)$. If any of these computations fail to converge, then $[\langle e,i\rangle]^A(n)$ diverges without asking an oracle question. Otherwise, we calculate $\Phi_e^A(n)$. We set $[\langle e,i\rangle]^A(n) = \Phi_e^A(n)$ if the computation converges and never quarries the oracle about a number $x > \varphi_i(n)$, and the computation $[\langle e,i\rangle]^A(n)$ diverges otherwise.

In general, for a partial computable function φ_i and a Turing functional Φ_e, we say φ_i bounds the use of Φ_e if for all oracles A and all inputs n such that $\Phi_e^A(n)$ converges, we have that $\varphi_i(0), \dots, \varphi_i(n)$ also converge and the computation $\Phi_e^A(n)$ never quarries the oracle about a number $x > \varphi_i(n)$.

To move from *wtt*-indices to Turing indices for functionals, we fix a computable function $T(e,i)$ which gives the Turing index for the *wtt*-functional $[\langle e,i\rangle]$. Note that if φ_i is a total computable function, then for every A and n, $\varphi_i(n)$ bounds the use of $\Phi^A_{T(e,i)} = [\langle e,i\rangle]^A$ in the usual sense. Furthermore, if φ_i is partial, then φ_i bounds the use of $\Phi_{T(e,i)}$ in the sense of the previous paragraph and for every oracle A, $\Phi^A_{T(e,i)}$ is partial. More importantly for the proof below, if Φ_e is a Turing functional such that φ_i (whether partial or total) bounds the use of Φ_e^A for every A, then $\Phi_e^A = [\langle e,i\rangle]^A$ for every A. That is, Φ_e and $[\langle e,i\rangle]$ are equal as functionals.

The next proposition gives a version of the recursion theorem for *wtt*-indices. In the statement of this proposition, we think of the computable function f as a mapping between *wtt*-indices.

Proposition 4.3. *Let $f(x,y) : \mathbb{N}^2 \to \mathbb{N}^2$ be a computable function. There is a wtt-index $\langle e,i\rangle$ such that for all A, $[\langle e,i\rangle]^A = [f(e,i)]^A$.*

Proof. Let $h(x,y) = T(f(x,y))$. Since $h(x,y)$ is a computable function from $\mathbb{N}^2$ to $\mathbb{N}$, the relativized version of the Recursion Theorem with Parameters gives us a computable function $n(y)$ such that for all A and y, we have $\Phi^A_{h(n(y),y)} = \Phi^A_{n(y)}$ as partial functions and the uses of these computations are identical. By the definitions of the functions h and T, $\Phi^A_{h(n(y),y)} = [f(n(y),y)]^A$ for all A. Therefore, the use of $\Phi^A_{h(n(y),y)}$, and hence also the use of $\Phi^A_{n(y)}$, is bounded by $\varphi_{\pi_2(f(n(y),y))}$ where $\pi_2(\langle u,v\rangle) = v$ is the second projection function on pairs.

Let $k(y)$ be the computable function defined by $k(y) = \pi_2(f(n(y),y))$. By the Recursion Theorem, there is an index a such that $\varphi_{k(a)} = \varphi_a$ as partial computable functions. By the definition of k, we have

$$\varphi_{\pi_2(f(n(a),a))} = \varphi_a$$

and so the use of $\Phi^A_{n(a)}$ is bounded by φ_a for every A. Therefore, we have $\Phi^A_{n(a)} = [\langle n(a),a\rangle]^A$ and

$$[f(n(a),a)]^A = \Phi^A_{h(n(a),a)} = \Phi^A_{n(a)} = [\langle n(a),a\rangle]^A$$

as required to prove the proposition with $e = n(a)$ and $i = a$. □

We end this section with the stronger non-uniformity result.

Proposition 4.4. *There is no partial computable function $f(x,y)$ such that for every pair $\langle x,y\rangle$ which is a c.e. degree index for a noncomputable Δ^0_2 set (that is, $A = Z_x$ is not computable and Φ^A_y is a modulus function for A), $f(x,y)$ converges and $\emptyset <_T [f(x,y)]^A <_{wtt} A$.*

Proof. Suppose there is such a partial computable function $f(x,y)$. Let $g(e,i)$ be the function witnessing the uniformity in the proof of Proposition 4.1. That is, for all *wtt*-indices $\langle e,i\rangle$, $g(e,i) = \langle x,y\rangle$ where $\langle x,y\rangle$ is the c.e. degree index for a noncomputable set A such that if $[\langle e,i\rangle]^A$ is computable, then either $[\langle e,i\rangle]^A$ is computable or $A \leq_{wtt} [\langle e.i\rangle]^A$. Note that the composition $f \circ g : \mathbb{N}^2 \to \mathbb{N}^2$ is a total computable function.

Applying Proposition 4.3 to $f \circ g$ we get a pair $\langle e,i\rangle$ such that $[\langle e,i\rangle]^X = [f(g(e,i))]^X$ for all sets X. Let A be the noncomputable set with c.e. degree index $g(e,i)$. The properties of f tell us that $\emptyset <_T [f(g(e,i))]^A <_{wtt} A$, so in particular, $[f(g(e,i))]^A$ is total. Therefore, $[\langle e,i\rangle]^A$ is also total. Since $g(e,i)$ is the c.e. degree index of A, the properties of g tell us that either $[\langle e,i\rangle]^A = [f(g(e,i))]^A$ is computable or $A \leq_{wtt} [\langle e,i\rangle]^A = [f(g(e,i))]^A$, both of which give a contradiction. □

4.2 Almost c.e. approximations

Over the next two sections, we present the proof of Theorem 1.2. In this section, we identify a specific type of approximation, called an almost c.e. approximation, such that if A has an almost c.e. approximation then it is straightforward to verify that there is a c.e. set $B \leq_{wtt} A$ of the same Turing degree as A. Thus, if such a set A is not computable, it cannot have minimal *wtt*-degree. After completing this case, we show that any set with c.e. Turing degree has an approximation which possesses most of the properties of an almost c.e. approximation. We will use this approximation to complete our proof in the next section.

Definition 4.5. A set A has an *almost c.e. approximation* if there exists a computable sequence of finite strings $\{\sigma_i[s] \mid i < s, s \in \omega\}$ satisfying the following properties for every $i < s$.

(P1) $\sigma_i[s] \subseteq \sigma_{i+1}[s]$.

(P2) $\sigma_i[s]$ and $\sigma_i[s+1]$ are either equal or incomparable.

(P3) If $\sigma_i[s]$ and $\sigma_i[s+1]$ are incomparable, then $\sigma_i[t]$ and $\sigma_i[s]$ are incomparable for every $t \geq s+1$.

(P4) For each i, $\lim_s \sigma_i[s]$ exists and $A = \bigcup_i \lim_s \sigma_i[s]$.

An almost c.e. approximation of A is a sequence of "marked" initial segments $\sigma_0[s] \subseteq \sigma_1[s] \subseteq \cdots \subseteq \sigma_{s-1}[s] \subseteq A_s$ at each stage s such that each time we move away from a marked segment (i.e. $\sigma_i[s] \not\subseteq A_{s+1}$), we cannot return to extend this marked segment at any future stage $t > s+1$ (i.e. $\sigma_i[s] \not\subseteq A_t$). For example, every c.e. set A has an almost c.e. approximation by setting $\sigma_i[s] = A_s \upharpoonright i$.

If A has an almost c.e. approximation, then A is clearly Δ^0_2. However, an almost c.e. approximation might not be either a left or right c.e. approximation since we might restore part (but not all) of $\sigma_i[s]$ at a future stage t after $\sigma_i[s] \not\subseteq A_{s+1}$. We say that these approximations are almost c.e. because they act as c.e. approximations to A modulo the marked segments. That is, $\sigma_i[s]$ is a correct initial segment of A as long as $\forall t \geq s\,(\sigma_i[s] \subseteq A_t)$, but as soon as this Π^0_1 statement fails, we know that $\sigma_i[s]$ is not a correct initial segment.

Proposition 4.6. *If A has an almost c.e. approximation then there is a c.e. set B such that $A \leq_T B \leq_{wtt} A$.*

Proof. Fix an almost c.e. approximation $\{\sigma_i[s] \mid i < s, s \in \omega\}$ of A. For each $i < s$ let $q_i^s = \max\{|\sigma_i[t]| \mid t \leq s\}$ and note that $\lim_s q_i^s$ exists since $\lim_s \sigma_i[s]$ exists. Let B be the set of all triples $\langle \sigma, q, i\rangle$ such that for some s, $\sigma = \sigma_i[s]$, $q = q_i^s$ and $\sigma_i[s] \neq \sigma_i[s+1]$.

From its definition, B is a c.e. set. In particular, a triple $\langle \sigma, q, i\rangle$ is only eligible to be enumerated into B if it has the form $\langle \sigma_i[t], q_i^t, i\rangle$ with $\sigma_i[t]$ and q_i^t calculated at some stage t of our almost c.e. approximation. Given such a triple $\langle \sigma_i[t], q_i^t, i\rangle$, we eventually enumerate this triple into B if and only if $\sigma_i[t] \not\subseteq A$. For one direction, if $\sigma_i[t] \not\subseteq A$, then we eventually see a stage $s \geq t$ such that $\sigma_i[t] = \sigma_i[s] \not\subseteq A_{s+1}$ and hence $\sigma_i[s] \neq \sigma_i[s+1]$. For the other direction, if $\sigma_i[t] \subseteq A$, then by Property (P3) of Definition 4.5, $\sigma_i[t] = \sigma_i[s]$ for all $s \geq t$.

To see that $A \leq_T B$, for each i, we search for the least stage s such that $\langle \sigma_i[s], q_i^s, i\rangle \notin B$. By the previous paragraph, such s exists and is the least stage such that $\sigma_i[s] \subseteq A$ (or equivalently, the least stage such that $\sigma_i[s] = \lim_t \sigma_i[t]$). Since $A = \bigcup_i \lim_t \sigma_i[t]$, this process suffices to compute A.

To see that $B \leq_{wtt} A$, fix a triple $\langle \sigma, q, i\rangle$. We search for the first stage s such that either $q_i^s > q$ or $\sigma_i[s] \subseteq A$. Because there are only finitely many possible values for strings $\sigma_i[t]$ of length less than q and because the values of q_i^t are non-decreasing in t, the existence of this stage s follows from Property (P3) of Definition 4.5. Furthermore, to compute s, we only need access to the first q many bits of A.

Suppose $q_i^s > q$. Because the values q_i^t are non-decreasing in t, we will not enumerate $\langle \sigma, q, i\rangle$ into B after stage s. Therefore, $\langle \sigma, q, i\rangle \in B$ if and only if there is a stage $t < s$ such that $\sigma = \sigma_i[t]$, $q = q_i^t$ and $\sigma_i[t] \neq \sigma_i[t+1]$.

On the other hand if $\sigma_i[s] \subseteq A$ then for every $t \geq s$ we have $\sigma_i[t] = \sigma_i[s]$ and $q_i^t = q_i^s$, which again means that $\langle \sigma, q, i\rangle \in B$ if and only if there is some $t < s$ so that $\sigma = \sigma_i[t]$, $q = q_i^t$ and $\sigma_i[t] \neq \sigma_i[t+1]$. □

Corollary 4.7. *If A has an almost c.e. approximation, then A is not wtt-minimal.*

Proof. Corollary 4.7 follows immediately from Proposition 4.6 because the non-computable c.e. *wtt*-degrees are dense. □

Corollary 4.7 completes the proof of Theorem 1.2 in the case when A has an almost c.e. approximation. Our next goal is to show that if A has c.e. Turning degree, then A can be approximated using strings which have Properties (P1), (P2) and (P4) from Definition 4.5.

Fix a set A of non-computable c.e. Turing degree. As noted in the previous section, by the Modulus Lemma, there is a Δ^0_2 approximation A_s to A such that A can compute a modulus for this approximation. We fix such an approximation A_s and a Turing functional Ψ such that Ψ^A is a modulus for the approximation A_s. Without loss of generality, we assume that if $\Psi^{A_s}(x)[s]$ converges, then $\Psi^{A_s}(y)[s]$ also converges for all $y < x$.

We use the fixed Δ^0_2 approximation A_s and functional Ψ to define a finite set of strings at each stage s which will eventually give us a approximation to A similar to an almost c.e. approximation. At each stage s, we compute a finite sequence $\alpha_0[s], \dots, \alpha_{k_s}[s]$ of initial segments of A_s as follows. Set $\alpha_0[s] = A_s \upharpoonright 0 = \langle A_s(0) \rangle$. If $\alpha_i[s]$ is defined, then we define $\alpha_{i+1}[s]$ to be the first string found satisfying

(C1) $\alpha_i[s] \subseteq \alpha_{i+1}[s] \subseteq A_s$ and

(C2) $\Psi^{\alpha_{i+1}[s]}(|\alpha_i[s]|)[s]$ converges.

If no such string $\alpha_{i+1}[s]$ is found, then our sequence of approximating strings ends with $\alpha_i[s]$ and we set $k_s = i$. To be more precise about the search procedure to define $\alpha_{i+1}[s]$, we first check whether $\Psi^{A_s}(|\alpha_i[s]|)[s]$ converges. If so, we take $\alpha_{i+1}[s]$ to be the shortest initial segment of A_s such that this computation does not query any bits greater than $|\alpha_{i+1}[s]|$ (so it satisfies (C2)) and such that it is also long enough to satisfy (C1). Note that the sequence $\alpha_0[s], \dots, \alpha_{k_s}[s]$ is uniformly computable in s.

It is straightforward to check by induction on i that $\alpha_i[s]$ is defined for cofinitely many stages s and that $\lim_s \alpha_i[s] = \alpha_i$ exists and is an initial segment of A. We want to make the set of these approximating sequences look more like an almost c.e. approximation by speeding up the computation procedure Ψ^A to ensure that at stage s, we define $\alpha_i[s]$ for all $i < s$. That is, we want to think of $\alpha_i[s]$ performing the same approximating task as $\sigma_i[s]$.

Definition 4.8. We say that s is an *n-modulus* stage if for all $x \leq n$, there is a $t \leq s$ such that $\Psi^{A_s}(x)[s] = t$ and for all stages u such that $t \leq u \leq s$, $A_u \upharpoonright x = A_t \upharpoonright x$.

Intuitively, s is an n-modulus stage if $\Psi^{A_s}[s]$ converges on all inputs up to n and the output stages are consistent (as far as we can tell at stage s) with Ψ^A being a modulus function for A. Since Ψ^A is a modulus function for A, it follows that for each n, there will be cofinitely many n-modulus stages.

Definition 4.9. We say that s is an *ℓ-approximation* stage if $\alpha_0[s], \dots, \alpha_{\ell-1}[s]$ are defined and s is an $|\alpha_{\ell-1}[s]|$-modulus stage.

That is, s is an ℓ-approximation stage if $\alpha_i[s]$ is defined for all $i < \ell$ and for every $x \leq |\alpha_{\ell-1}[s]|$, the computations $\Psi^{A_s}(x)[s]$ are consistent (as far as we can tell at stage s) with Ψ^A being a modulus function. Again, because Ψ^A is a modulus function, there are cofinitely many ℓ-approximation stages for each ℓ.

Let $0 = t_0 < t_1 < t_2 < \cdots$ be a sequence of stages such that for $s > 0$, t_s is an s-approximation stage. We speed up our computations to run along these chosen stages so we can treat stage s as an s-approximation stage. That is, we assume that at stage s, the strings $\alpha_i[s]$ are defined for $i < s$ and that for all $x \leq |\alpha_{s-1}[s]|$, the computation $\Psi^{A_s}(x)[s] = t$ converges with $t \leq s$ and for all u such that $t \leq u \leq s$, $A_u \upharpoonright x = A_t \upharpoonright x$.

In particular, we now have an approximation to A by finite strings in stages given by $\{\alpha_i[s] \mid i < s, s \in \omega\}$. In the remained of this section, we verify properties of these approximating sequence and show that they closely resemble an almost c.e. approximation. First, we show that they satisfies Properties (P1) and (P4) of an almost c.e. approximation.

Lemma 4.10. *Our sequences $\{\alpha_i[s] \mid i < s, s \in \omega\}$ satisfy $\alpha_i[s] \subseteq \alpha_{i+1}[s]$. Furthermore, $\lim_s \alpha_i[s]$ exists and is an initial segment of A.*

Proof. The first statement is just the the condition (C1). The proof of the second statement is a straightforward induction on i. The base case is clear since $\alpha_0[s] = \langle A_s(0)\rangle$ and A_s is a Δ^0_2 approximation for A. For the induction case, let s_i denote a stage such that $\alpha_i[s_i]$ has reached its limiting value. The value of $\alpha_{i+1}[s]$ will stabilize by stage $s > s_i$ such that $\Phi^{A_s}(|\alpha_i[s_i]|)[s]$ converges and A_s is correct up to the maximum of $|\alpha_i[s_i]|$ and the use of the computation. □

Lemma 4.11. *Let $i < s < t$ be such that $\alpha_i[s] \subseteq A_t$. For all $j \leq i$, we have $\alpha_j[t] = \alpha_j[s]$, and for all $j < i$ and all u satisfying $s \leq u \leq t$, we have $\alpha_j[u] = \alpha_j[s]$.*

Proof. We proceed by induction of i. Since $\alpha_0[s] = \langle A_s(0)\rangle$, the statement is clear for $i = 0$. Assume the lemma holds for i and we prove it for $i+1$. Fix $t > s$ such that $\alpha_{i+1}[s] \subseteq A_t$. Since $\alpha_i[s] \subseteq \alpha_{i+1}[s] \subseteq A_t$, the induction hypothesis gives that $\alpha_j[t] = \alpha_j[s]$ for all $j \leq i$ and that $\alpha_j[u] = \alpha_j[s]$ for all $j < i$ and $s \leq u \leq t$. Because $\alpha_i[t] = \alpha_i[s]$ and $\alpha_{i+1}[s] \subseteq A_t$, we have that $\Phi^{A_t}(|\alpha_i[t]|)[t]$ converges by the same computation as $\Phi^{A_s}(|\alpha_i[s]|)[s]$, and therefore, that $\alpha_{i+1}[t]$ is chosen to be the same initial segment as $\alpha_{i+1}[s]$.

To complete the proof, assume for a contradiction that there is a stage u with $s < u < t$ such that $\alpha_i[u] \neq \alpha_i[s]$. Fix such a stage u. Since $\alpha_{i+1}[s]$ is defined, we know that

$$\Psi^{\alpha_{i+1}[s]}(|\alpha_i[s]|)[s] = s' \leq s.$$

As above, since $s < t$, $\alpha_i[s] = \alpha_i[t]$ and $\alpha_{i+1}[s] \subseteq A_t$, we have

$$\Psi^{A_t}(|\alpha_i[t]|)[t] = s'.$$

Because t is an $|\alpha_i[t]|$-modulus stage, we have

$$A_v \upharpoonright |\alpha_i[t]| = A_{s'} \upharpoonright |\alpha_i[t]|$$

for all stages v such that $s' \leq v \leq t$. However, $s' \leq s < u < t$ and $\alpha_i[s] = \alpha_i[t]$, so we conclude that

$$A_u \upharpoonright |\alpha_i[s]| = A_s \upharpoonright |\alpha_i[s]| = \alpha_i[s]$$

and hence $\alpha_i[s] \subseteq A_u$. By the induction hypothesis, $\alpha_i[u] = \alpha_i[s]$ contradicting our assumption that $\alpha_i[u] \neq \alpha_i[s]$ and concluding the proof. □

The next lemma shows that these approximating strings also satisfy Property (P2) of an almost c.e. approximation.

Lemma 4.12. *For all s and all $i < s$, $\alpha_i[s]$ and $\alpha_i[s+1]$ are either equal or incomparable.*

Proof. The proof proceeds by induction on i with the case $i = 0$ holding trivially by definition. For the induction case, assume $i > 0$ with $\alpha_i[s]$ and $\alpha_i[s+1]$ comparable. It follows that $\alpha_{i-1}[s]$ and $\alpha_{i-1}[s+1]$ are comparable and hence equal. In particular, $|\alpha_{i-1}[s]| = |\alpha_{i-1}[s+1]|$. If $\alpha_i[s+1] \subsetneq \alpha_i[s] \subseteq A_s$, then the string $\alpha_i[s+1]$ was available as a potential value to be chosen for $\alpha_i[s]$ (i.e. it is an initial segment of A_s extending $\alpha_{i-1}[s]$ and is long enough to use as an oracle for the convergent computation on $|\alpha_{i-1}[s]|$) and so we would have chosen $\alpha_i[s]$ to be the shorter string $\alpha_i[s+1]$. Therefore, we cannot have $\alpha_i[s+1] \subsetneq \alpha_i[s] \subseteq A_s$. So, we must have $\alpha_i[s] \subseteq \alpha_i[s+1] \subseteq A_{s+1}$ and hence by Lemma 4.11, $\alpha_i[s] = \alpha_i[s+1]$. □

In this proof, we use the fact that $s + 1 > s$ but we never use the fact that $s + 1$ is the stage immediately after s. Therefore, this proof really shows that comparable strings $\alpha_i[s]$ and $\alpha_i[t]$ at stages $s < t$ must be equal. We will use this property repeatedly in the verification of the construction in the next section.

Lemma 4.13. *For all stages $s < t$ and indices $i < s$, if $\alpha_i[s]$ and $\alpha_i[t]$ are comparable, then $\alpha_i[s] = \alpha_i[t]$.*

Our next fact shows that although the strings $\alpha_i[s]$ may not satisfy Property (P3) of an almost c.e. approximation, they do satisfy a similar property. We can have stages $s < u < t$ such that $\alpha_i[s] \neq \alpha_i[u]$ but $\alpha_i[s] = \alpha_i[t]$. However, when this happens, we can guarantee that the value of $\alpha_{i+1}[t]$ is not equal to any value of this string prior to stage s.

Lemma 4.14. *Suppose $\alpha_i[s]$ is defined and $\alpha_i[s] \not\subseteq A_u$ for some $u > s$. At every future stage $t > u$, if $\alpha_i[s] \subseteq A_t$, then $\alpha_{i+1}[t] \neq \alpha_{i+1}[s']$ for all $s' \leq s$.*

Proof. Suppose that $\alpha_i[s] \not\subseteq A_u$ for some $u > s$ and fix a stage $t > u$ such that $\alpha_i[s] \subseteq A_t$. Assume for a contradiction that $\alpha_{i+1}[t] = \alpha_{i+1}[s']$ for a fixed $s' \leq s$.

By Lemma 4.11, $\alpha_i[s] \subseteq A_t$ implies $\alpha_i[s] = \alpha_i[t]$. Similarly, $\alpha_i[s'] \subseteq \alpha_{i+1}[s'] = \alpha_{i+1}[t] \subseteq A_t$ implies $\alpha_i[s'] = \alpha_i[t]$ and hence $\alpha_i[s'] = \alpha_i[s]$. However, since $\alpha_i[s] \not\subseteq A_u$, we know that $\alpha_i[s] \neq \alpha_i[u]$ and so $\alpha_i[s'] \neq \alpha_i[u]$. Putting these facts together, we have stages $s' < u < t$ with $\alpha_i[u] \neq \alpha_i[s']$ and $\alpha_{i+1}[s'] \subseteq A_t$ contradicting Lemma 4.11. □

We next give a slight strengthening of this lemma. We say that a string $\alpha_i[s]$ is *new* if $\alpha_i[s] \neq \alpha_i[s']$ for all $s' < s$. Similarly, we say $\alpha_i[s]$ was *new at stage* t (or *first appeared at stage* t) if $t \leq s$, $\alpha_i[s] = \alpha_i[t]$ and $\alpha_i[t]$ was new.

Lemma 4.15. *If $s_0 < s_1 < s_2$ are stages such that $\alpha_i[s_0] \neq \alpha_i[s_1]$ but $\alpha_i[s_0] = \alpha_i[s_2]$, then $\alpha_{i+1}[s_2] \neq \alpha_{i+1}[s']$ for all $s' \leq s_1$. In particular, if $\alpha_{i+1}[s_2]$ was new at stage t, then $s_1 < t$.*

Proof. The second statement in the lemma follows immediately from the first statement. To prove the first statement, fix stages $s_0 < s_1 < s_2$ and an index i as described. Suppose for a contradiction that $\alpha_{i+1}[s_2] = \alpha_{i+1}[s']$ for some $s' \leq s_1$, and hence that $\alpha_{i+1}[s'] \subseteq A_{s_2}$. By Lemma 4.11, $\alpha_i[s'] = \alpha_i[s_2]$ (and hence $\alpha_i[s'] = \alpha_i[s_0]$) and for all stages u such that $s' \leq u \leq s_2$, $\alpha_i[u] = \alpha_i[s']$. In particular, since $s' \leq s_1 \leq s_2$, we have $\alpha_i[s_1] = \alpha_i[s']$ and therefore $\alpha_i[s_1] = \alpha_i[s_0]$ for the desired contradiction. □

Before finishing this section, we want to slightly alter our definition of the sequence of strings $\alpha_i[s]$ by adding two stretching conditions in the case when $i > 0$. First, by Lemma 4.15, we know that if $s_0 < s_1$ with $\alpha_i[s_0] \neq \alpha_i[s_1]$, then the values of $\alpha_{i+1}[t]$ for $t > s_1$ are all initially chosen after stage s_1. In such a situation, we want to choose the strings $\alpha_{i+1}[t]$ to have length longer than s_1. Second, when a value $\alpha_{i+1}[s]$ is new (i.e. $\alpha_{i+1}[s] \neq \alpha_{i+1}[s']$ for all $s' < s$), we want to choose $\alpha_{i+1}[s]$ so that its length is at least as long as the lengths of the values of $\alpha_{i+1}[s']$ for $s' < s$.

Formally, we define $\alpha_0[s] = A_s \restriction 0 = \langle A_s(0) \rangle$ and we choose $\alpha_{i+1}[s]$ to satisfy

(C1) $\alpha_i[s] \subseteq \alpha_{i+1}[s] \subseteq A_s$,

(C2) $\Psi^{\alpha_{i+1}[s]}(|\alpha_i[s]|)[s]$ converges,

(C3) if there are stages $s_0 < s_1 < s$ with $\alpha_i[s_0] = \alpha_i[s]$ and $\alpha_i[s_0] \neq \alpha_i[s_1]$, then $|\alpha_{i+1}[s_1]| > s_1$, and

(C4) if $\alpha_{i+1}[s] \neq \alpha_{i+1}[s']$ for all $s' < s$, then $|\alpha_{i+1}[s]| \geq \max\{|\alpha_{i+1}[s']| \mid s' < s\}$.

To incorporate these stretching conditions, when defining $\alpha_{i+1}[s]$, we first check whether there is a stage $u < s$ such that $\alpha_i[s] = \alpha_i[u]$ and $\alpha_{i+1}[u] \subseteq A_s$. If so, we set $\alpha_{i+1}[s] = \alpha_{i+1}[u]$ for the least such state u. Otherwise, we choose $\alpha_{i+1}[s]$ to be the least initial segment of A_s satisfying (C1)-(C4). By speeding up our computations as before, we assume that at stage s, the strings $\alpha_0[s], \ldots, \alpha_{s-1}[s]$ are defined. With minor changes, the arguments for the properties given in Lemmas 4.10 through 4.15 go through so we maintain these properties.

4.3 Proof of Theorem 1.2

We turn to the proof of Theorem 1.2. Fix a set A of noncomputable c.e. degree and by Corollary 4.7 assume that A does not have an almost c.e. approximation. We use this assumption in an essential way during the construction.

We need to construct a noncomputable set C such that $C \leq_{wtt} A$ and $A \not\leq_{wtt} C$. In fact, C will have the stronger property that $A \not\leq_T C$. We meet the following requirements:

$$\mathcal{P}_e : C \neq \Delta_e$$
$$\mathcal{R}_e : \Phi_e^C \neq A$$

where Δ_e is the e^{th} partial computable function and Φ_e is the e^{th} Turing functional. The construction is finite injury and the requirements are given priority $\mathcal{P}_0 < \mathcal{R}_0 < \mathcal{P}_1 < \cdots$.

4.3.1 Definition of $C \leq_{wtt} A$

The reduction $C \leq_{wtt} A$ will have identity bounded use. We indirectly build C using the notion of *marks*. At each stage of the construction we may declare an unmarked number marked (marking a number), or declare an already marked number unmarked (removing a mark). Since competing $\mathcal{R}$ requirements may have different views about wanting to have a number marked or unmarked, we will allow a number to be conditionally unmarked with respect to a neighborhood.

A neighborhood $N(i,s)$ is specified by an index i and a stage s. The neighborhood $N(i,s)$ is the set of all X such that $\alpha_i[s] \subseteq X$ and $\alpha_{i+1}[t] \not\subseteq X$ for any $t \leq s$. Therefore a neighborhood $N(i,s)$ contains the possible values for A if it is the case that the approximation for A moves away from $\alpha_i[s]$ but later returns to $\alpha_i[s]$. The neighborhood $N(i,s)$ is said to *apply at stage* $u > s$ (or *be applicable at* u) if $A_u \in N(i,s)$.

Each number may be declared marked at most once. The marking of a number is global and applies to all neighborhoods. A number can be declared unmarked with respect to some neighborhood only if it is already marked. We will ensure during the construction that a mark on m can only be placed after stage m.

Intuitively, if a number x has been marked but has not been unmarked with respect to an applicable neighborhood at a stage s, we will have (as long as it is consistent to do so) $C_s(x) = 1$, and $C_s(x) = 0$ otherwise. Formally, we have the following definition.

Definition 4.16. We define the stage s approximation C_s of C as follows. For each $x < s$ if $A_t \upharpoonright x \not\subseteq A_s$ for every $x < t < s$, then $C_s(x) = 1$ if and only if there is a mark on x which has not yet been removed with respect to a neighborhood that currently applies. Otherwise, $C_s(x) = C_t(x)$ for the least stage t such that $x < t < s$ such that $A_t \upharpoonright x \subseteq A_s$.

The following lemma shows this definition ensures $C \leq_{wtt} A$ as required.

Lemma 4.17. *For every x and $s > t > x$, if $A_t \upharpoonright x \subseteq A_s$ then $C_s(x) = C_t(x)$. Hence $C = \lim_s C_s$ exists and C is computable from A with identity bounded use.*

Proof. The first statement follows by a straightforward induction on s. To see that $C = \lim_s C_s$ exists, fix x and let $s' > x$ be such that $A_s \upharpoonright x = A \upharpoonright x$ for all $s \geq s'$. Since $C_s(x) = C_{s'}(x)$ for all $s \geq s'$, $\lim_s C_s(x)$ exists.

To compute $C(x)$ from A, let $s > x$ be the least stage such that $A_s \upharpoonright x = A \upharpoonright x$. Since s is chosen least, $A_t \upharpoonright x \not\subset A_s$ for all t such that $x < t < s$. By definition, $C_s(x) = 1$ if and only if there is a mark on x at stage s which has not been removed with respect to a neighborhood containing A_s. By the first statement in the lemma, $C(x) = C_s(x)$ and hence we can determine the value of $C(x)$ using only $A \upharpoonright x$. □

4.3.2 Informal description of the $\mathcal{P}_e$ strategy

We describe the basic strategy to meet a single $\mathcal{P}_e$ requirement. $\mathcal{P}_e$ defines a sequence of followers $p_e(0) < p_e(1) < \cdots$ at stages $s_e(0) < s_e(1) < \cdots$ and attempts to use $\alpha_{p_e(i)}[s_e(i+1)]$ to compute A. In addition, $\mathcal{P}_e$ defines a sequence of marked numbers $m_e(0) < m_e(1) < \cdots$ with $m_e(i)$ marked at stage $s_e(i+1)$ and tries to ensure that for some i, we have $C(m_e(i)) = 1 \neq \Delta_e(m_e(i))$. Because A is not computable, one of these diagonalization attempts will succeed.

At stage $s_e(i+1)$, $\mathcal{P}_e$ declares that it has computed A up to $|\alpha_{p_e(i)}[s_e(i+1)]|$ to be equal to $\alpha_{p_e(i)}[s_e(i+1)]$ and it marks a number $m_e(i)$ for which it has seen $\Delta_e(m_e(i)) = 0$. If for all $t \geq s_e(i+1)$, $\alpha_{p_e(i)}[s_e(i+1)] = \alpha_{p_e(i)}[t]$, then $\mathcal{P}_e$'s declared computation is correct. Since A is not computable, there must be a follower $p_e(i)$ and stage $t > s_e(i+1)$ such that $\alpha_{p_e(i)}[t] \neq \alpha_{p_e(i)}[s_e(i+1)]$. Under the right circumstances, the movement of A from $A_{s_e(i+1)}$ to A_t will cause the mark on $m_e(i)$ to change the definition of $C_{s_e(i+1)}(m_e(i)) = 0$ to $C_t(m_e(i)) = 1$ permanently.

More formally, $\mathcal{P}_e$ acts as follows.

1. Choose $p_e(0)$ large at stage $s_e(0)$. Assume that $p_e(i)$ is the largest defined follower and $p_e(i)$ was chosen at stage $s_e(i)$.

2. Wait for a stage $s > s_e(i)$ at which there is a fresh number m (unmarked and unused by any other requirement) such that

   (a) $m_e(i-1) < m < s$ (where $m_e(-1) = 0$),

   (b) $C_s \upharpoonright m = \Delta_{e,s} \upharpoonright m$ (so $C_s(m) = \Delta_{e,s}(m) = 0$ because m is unmarked),

   (c) $\alpha_{p_e(i)+1}[u] = \alpha_{p_e(i)+1}[m]$ for all stages u such that $m \leq u \leq s$, and

   (d) $|\alpha_{p_e(i)+1}[u]| < m$ for all stages $u \leq s$.

3. When $\mathcal{P}_e$ sees such a stage s and witness m, it

   (a) sets $m_e(i) = m$ and marks $m_e(i)$,

   (b) sets $s_e(i+1) = s$ and chooses $p_e(i+1)$ large,

   (c) declares it has computed $A \upharpoonright |\alpha_{p_e(i)}[s_e(i+1)]| = \alpha_{p_e(i)}[s_e(i+1)]$, and

   (d) returns to Step 2 with i incremented to $i+1$.

To see why this strategy should succeed, recall that $\alpha_{p_e(i)}[s]$ takes only finitely many values as s increases because it has a limit. Therefore, there will be cofinitely many stages at which there is a (fixed) witness m satisfying the conditions in 2(a), 2(c) and 2(d). If 2(b) is never satisfied at any of these stages, then $C \neq \Delta_e$ and $\mathcal{P}_e$ is won. However, as noted above, if $\mathcal{P}_e$ produces infinitely many followers and for every $t > s_e(i+1)$, $\alpha_{p_e(i)}[s_e(i+1)] = \alpha_{p_e(i)}[t]$, then A would be computable. Therefore, we consider the least index i and least stage $t > s_e(i+1)$ at which $\alpha_{p_e(i)}[t] \neq \alpha_{p_e(i)}[s_e(i+1)]$. Since $\Delta_e(m_e(i)) = 0$, we need to explain why $C(m_e(i)) = 1$. In fact, we show that $C_{t'}(m_e(i)) = 1$ for all $t' \geq t$.

Fix a stage $t' \geq t$. To determine the value of $C_{t'}(m_e(i))$, let $s \leq t'$ be the least stage such that $m_e(i) < s$ and $A_{t'} \upharpoonright m_e(i) = A_s \upharpoonright m_e(i)$. If $s_e(i+1) < s$, then since $m_e(i)$ is marked at stage $s_e(i+1)$, it follows from the definition of C that $C_s(m_e(i)) = 1$ and hence by Lemma 4.17, $C_{t'}(m_e(i)) = C_s(m_e(i)) = 1$. Therefore, it suffices to show that we cannot have $s \leq s_e(i+1)$.

Suppose for a contradiction that $A_{t'} \upharpoonright m_e(i) = A_s \upharpoonright m_e(i)$ with $s \leq s_e(i+1)$. By Condition 2(d) when $m_e(i)$ is defined at stage $s_e(i+1)$, $|\alpha_{p_e(i)+1}[s]| < m_e(i)$ and hence $\alpha_{p_e(i)+1}[s] \subseteq A_{t'}$. By Lemma 4.11, for all stages u such that $s \leq u \leq t'$, $\alpha_{p_e(i)}[u] = \alpha_{p_e(i)}[s]$. However, we have $s \leq s_e(i+1) < t \leq t'$, so it follows that $\alpha_{p_e(i)}[s_e(i+1)] = \alpha_{p_e(i)}[t]$ for the desired contradiction.

This completes the initial description of the $\mathcal{P}_e$ strategy. Based on our explanation for why the strategy will succeed, it might appear that the set C will be computably enumerable. To see why this appearance is deceptive, notice that Conditions 2(b) and 2(c) have the potential to make $m_e(i)$ much larger than $|\alpha_{p_e(i)+1}[s_e(i+1)]|$. Let $m = m_e(i)$ and $s = s_e(i+1)$. Consider the case when there is an index $j > p_e(i)+1$ such that $|\alpha_j[s]| < m < |\alpha_{j+1}[s]|$. (See Figure 4.1.) By Condition 2(b), we know $C_s(m) = 0$. We could have a stage $t_0 > s$ such that $\alpha_j[t_0] \neq \alpha_j[s]$ and hence the approximation A_{t_0} differs from A_s below m. If $A_{t_0} \upharpoonright m$ appears as an initial segment of the approximation to A for the first time at stage t_0, then by definition, $C_{t_0}(m) = 1$. However, at a later stage $t_1 > t_0$, we could have $\alpha_j[t_1] = \alpha_j[s]$. Since $\alpha_j[t_1]$ is returning to the previous value $\alpha_j[s]$ after changing at t_0, we know $\alpha_{j+1}[t_1]$ differs from $\alpha_{j+1}[u]$ for all $u \leq t_0$. However, since $m < |\alpha_{j+1}[s]|$, we could have $\alpha_{j+1}[t_1] \upharpoonright m = \alpha_{j+1}[s] \upharpoonright m$, in which case $A_{t_1} \upharpoonright m = A_s \upharpoonright m$ and so by definition, $C_{t_1}(m) = 0$. Thus, our approximation to C need not be a c.e. approximation.

This example also illustrates the general problem we need to confront with the $\mathcal{R}_e$ strategies. To make the problem for an $\mathcal{R}_e$ strategy easier to illustrate (see Figure 4.2 for a picture), suppose that in the example above, $|\alpha_{j+1}[t_0]| > m$ (as shown in Figure 4.1). At stage $t_2 > t_1$, the opponent is free to move the approximation A_{t_2} so that $A_{t_2} \upharpoonright m = A_{t_0} \upharpoonright m$ by making $\alpha_j[t_2] = \alpha_j[t_0]$. This change at stage t_2 causes $C_{t_2}(m) = 1$. Later, the opponent can give us a stage $t_3 > t_2$ at which $\alpha_j[t_3] = \alpha_j[t_1] = \alpha_j[s]$. While this change in the approximation to A back to extending $\alpha_j[s]$ causes $\alpha_{j+1}[t_3]$ to differ from $\alpha_{j+1}[u]$ for $u \leq t_2$, there is nothing to stop $\alpha_{j+1}[t_3] \upharpoonright m = \alpha_{j+1}[s] \upharpoonright m$ and hence causing the value of $C_{t_3}(m)$ to change back to $C_{t_3}(m) = 0$.

By following this strategy, the opponent has created a split using the values

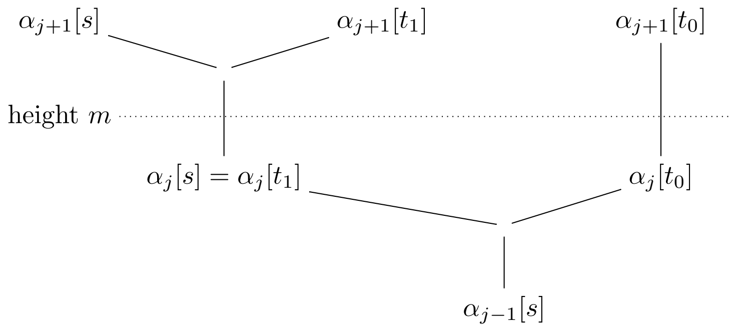

Figure 4.1: The set C need not be computable enumerable.

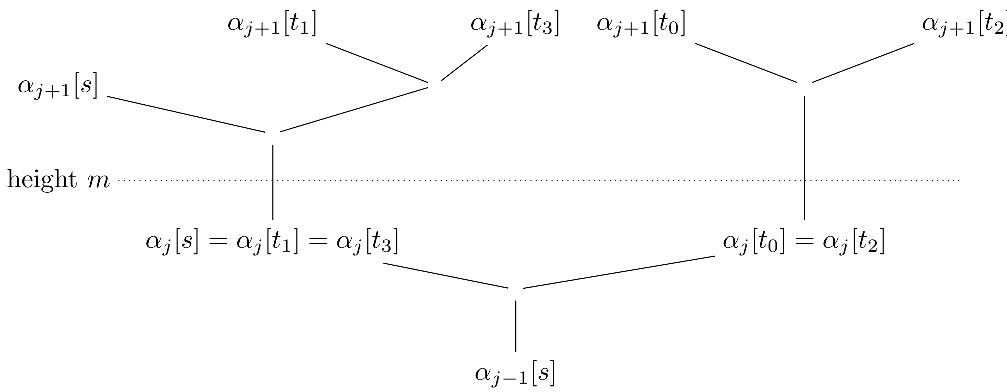

Figure 4.2: The opponent can define a split allowing C to compute A if we are not careful.

of $\alpha_j[s]$ and $\alpha_j[t_0]$ and has threatened to start building a splitting tree which could be used to compute A from C. That is, if left unchecked, the opponent can guarantee that if $C(m) = 0$, then $\alpha_j[s] \subseteq A$ and if $C(m) = 1$, then $\alpha_j[t_0] \subseteq A$. If we are not careful, the opponent can use infinitely many $\mathcal{P}$ strategies together to construct a splitting tree allowing C to compute A. Preventing the opponent from constructing such a tree will be the main goal of the $\mathcal{R}$ requirements.

4.3.3 Informal description of an $\mathcal{R}_e$ strategy

The main goal of the $\mathcal{R}_e$ strategy is to prevent the opponent from building an e-splitting tree that enables C to compute A. $\mathcal{R}_e$ defines a sequence of indices $r_e(0) < r_e(1) < \cdots$ and attempts to use the strings $\alpha_{r_e(i)}$ at certain specific stages to define an almost c.e. approximation to A. The fact that A does not have an almost c.e. approximation will prevent $\mathcal{R}_e$ from acting infinitely often and hence there will be a finite stage at which $\mathcal{R}_e$ becomes satisfied. Because the $\mathcal{R}$ requirements do not mark numbers, and hence do not cause any numbers to enter C, we describe the action of a single $\mathcal{R}_e$ in the presence of $\mathcal{P}$ requirements of lower priority which are marking numbers.

When $\mathcal{R}_e$ is first eligible to act, it defines the index $r_e(0)$ large and begins to wait for a stage s_0 at which $\alpha_{r_e(0)+1}[s_0] \subseteq \Phi_e^C[s_0]$ with some use $u_0 < s_0$. When such a stage s_0 appears, $\mathcal{R}_e$ defines $r_e(1)$ to be a large index and sets $\sigma^0_{e,0} = \alpha_{r_e(0)}[s_0]$. In general, the string $\sigma^i_{e,n}$ with be the n-th string defined in the i-th level of a potential almost c.e. approximation to A. (In the end, if $\mathcal{R}_e$ acts infinitely often, we will thin out this collection of strings to get the actual almost c.e. approximation of A.) Furthermore, for all marked numbers $m \geq |\alpha_{r_e(0)+1}[s_0]|$ such that $C_{s_0}(m) = 0$, $\mathcal{R}_e$ removes the mark on m with respect to the neighborhood $N(r_e(0), s_0)$. (The reason for this removal will be explained below.)

Consider what can happen to $\alpha_{r_e(0)}[s]$ for $s > s_0$. If $\alpha_{r_e(0)}[s] = \alpha_{r_e(0)}[s_0]$ for all $s > s_0$, then we have made progress towards computing A. Since A is not computable, this behavior cannot continue indefinitely for larger indices $r_e(i)$. Therefore, the interesting case is when there is a stage $s_1 > s_0$ at which $\alpha_{r_e(0)}[s_1] \neq \alpha_{r_e(0)}[s_0]$. If we were lucky enough to have $C_{s_1} \upharpoonright u_0 = C_{s_0} \upharpoonright u_0$, then we would have $\alpha_{r_e(0)}[s_1] \not\subseteq \Phi_e^C[s_1]$ because the use of the computation from stage s_0 showing $\alpha_{r_e(0)+1}[s_0] \subseteq \Phi_e^C[s_0]$ has been preserved. In this case, we would have (at least temporarily) satisfied $\mathcal{R}_e$. However, the use u_0 could be large and so, as we saw in the examples from the informal $\mathcal{P}$ strategies, the opponent can arrange things so that $C_{s_1} \upharpoonright u_0 \neq C_{s_0} \upharpoonright u_0$.

However, we have gained control over what would happen if there were a stage $s_2 > s_1$ at which $\alpha_{r_e(0)}[s_2]$ reverted back to $\alpha_{r_e(0)}[s_0]$. Suppose that $s_2 > s_1$ is the least stage at which $\alpha_{r_e(0)}[s_2] = \alpha_{r_e(0)}[s_0]$. By Lemma 4.15, $\alpha_{r_e(0)+1}[s_2] \neq \alpha_{r_e(0)+1}[t]$ for all $t \leq s_1$. Furthermore, by the stretching condition (C3), $|\alpha_{r_e(0)+1}[s_2]| > s_1$. Since $u_0 < s_0$, we have $|\alpha_{r_e(0)+1}[s_2]| > u_0$. We claim that $C[s_2] \upharpoonright u_0 = C[s_0] \upharpoonright u_0$. To prove this claim it suffices to show that $C_{s_0}(m) = C_{s_2}(m)$ for all numbers $m < u_0$ which have been marked at some stage before s_0 because $\mathcal{R}_e$ initializes the $\mathcal{P}$ strategies at stage s_0 so any number marked after s_0 will be chosen large and hence will be greater than u_0. Fix $m < u_0$ which is marked before stage s_0 and we break into cases to show $C_{s_0}(m) = C_{s_2}(m)$.

First, suppose $m \leq |\alpha_{r_e(0)}[s_0]|$. Since $\alpha_{r_e(0)}[s_2] = \alpha_{r_e(0)}[s_0]$, we have $A_{s_2} \upharpoonright m = A_{s_0} \upharpoonright m$ and hence $C_{s_2}(m) = C_{s_0}(m)$ by Lemma 4.17.

Second, suppose $m \geq |\alpha_{r_e(0)+1}[s_0]|$. In this case, we claim that $A_{s_2} \upharpoonright m$ first appears as an approximation to A at stage s_2 and hence (by definition) $C_{s_2}(m) = 1$ if and only if there is a mark on m which applies at stage s_2. Before proving the claim, we show that $C_{s_2}(m) = C_{s_0}(m)$ follows from the claim. Recall that since $m \geq |\alpha_{r_e(0)+1}[s_0]|$, when $\mathcal{R}_e$ acts at stage s, it removes the mark on m with respect to the neighborhood $N(r_e(0), s_0)$ if $C_{s_0}(m) = 0$. If $C_{s_0}(m) = 1$, then the mark on m was not removed at stage s_0 and hence $C_{s_2}(m) = 1$. On the other hand, if $C_{s_0}(m) = 0$, then the mark on m was removed at stage s_0 with respect to the neighborhood $N(r_e(0), s_0)$. This neighborhood applies at s_2 because $\alpha_{r_e(0)}[s_2]$ has reverted back to $\alpha_{r_e(0)}[s_0]$ after changing values at s_1. Therefore, the mark on m does not apply at s_2 and hence $C_{s_2}(m) = 0$.

We now prove that claim. If $A_{s_2} \upharpoonright m = A_t \upharpoonright m$ for some $t \leq s_1$, then we would have $\alpha_{r_e(0)+1}[s_2] = \alpha_{r_e(0)+1}[t]$ for a contradiction. Therefore, $A_{s_2} \upharpoonright m$ first appears after stage s_1. Since $|\alpha_{r_e(0)}[s_0]| < m$ and s_2 is the first stage after s_1 at which $\alpha_{r_e(0)}[s_2] = \alpha_{r_e(0)}[s_0]$, $A_{s_2} \upharpoonright m \neq A_t \upharpoonright m$ for all t such that $s_1 \leq t < s_2$. This completes the proof of the claim and finishes the second case.

Finally, suppose $|\alpha_{r_e(0)}[s_0]| < m < |\alpha_{r_e(0)+1}[s_0]|$. Because $m < |\alpha_{r_e(0)+1}[s_0]|$, the mark on m is not removed when $\mathcal{R}_e$ acts at s_0. Assume $m = m_j(k)$ is marked by $\mathcal{P}_j$ at stage $t_0 < s_0$ with associated string $\alpha_{p_j(k)}[t_0]$. By Conditions 2(c) and 2(d) in the action of $\mathcal{P}_j$, we know that $\alpha_{p_j(k)}$ is constant on the interval $[m, t_0]$ of stages and that $|\alpha_{p_j(k)+1}[t]| < m$ for all $t \leq t_0$. Since $\mathcal{P}_j$ has lower priority than $\mathcal{R}_e$, we have $r_e(0) < p_j(k)$ and hence $|\alpha_{r_e(0)+1}[t]| < m$ for all $t \leq t_0$. Because m is marked at stage t_0 and the mark is not removed at stage s_0, we know that $C_s(m) = 1$ for any stage $s > t_0$ at which $A_s \upharpoonright m \neq A_{t_0} \upharpoonright m$.

We claim that $C_{s_0}(m) = C_{s_2}(m) = 1$. To see that $C_{s_0}(m) = 1$, note that if $A_{s_0} \upharpoonright m = A_{t_0} \upharpoonright m$, then we would have $\alpha_{r_e(0)+1}[s_0] = \alpha_{r_e(0)+1}[t_0]$ because $|\alpha_{r_e(0)+1}[t_0]| < m$. Therefore, we would have $|\alpha_{r_e(0)+1}[s_0]| < m$ contradicting our case assumption on the size of m. Similarly, to see that $C_{s_2}(m) = 1$, note that if $A_{s_2} \upharpoonright m = A_{t_0} \upharpoonright m$, then we would have $\alpha_{r_e(0)+1}[s_2] = \alpha_{r_e(0)+1}[t_0]$. However, $\alpha_{r_e(0)+1}[s_2] \neq \alpha_{r_e(0)+1}[t]$ for all $t \leq s_1$ giving the desired contradiction.

This completes the proof that $C_{s_2} \upharpoonright u_0 = C_{s_0} \upharpoonright u_0$. What does this fact tell us about the construction? If A_s ever moves away from $\sigma^0_{e,0}$ at stage s_1 and later returns to $\sigma^0_{e,0}$ at stage s_2, then the computation $\alpha_{r_e(0)+1}[s_0] \subseteq \Phi^C_e[s_2]$ holds because the use of the computation $\alpha_{r_e(0)+1}[s_0] \subseteq \Phi^C_e[s_0]$ was preserved. However, the strings $\alpha_{r_e(0)+1}[s_2]$ and $\alpha_{r_e(0)+1}[s_2]$ are incomparable and hence $\alpha_{r_e(0)+1}[s_2] \not\subseteq \Phi^C_e[s_2]$. Therefore, $\mathcal{R}_e$ looks (at least temporarily) satisfied at s_2.

The general strategy for $\mathcal{R}_e$ uses this procedure to define the sequence $r_e(0) < r_e(1) < \cdots$ of witness indices and the strings $\sigma^i_{e,n}$ of potential members of an almost c.e. approximating family. At stage s, we fix the largest index i (if any) such that $\sigma^i_{e,n} \subseteq A_s$ for some n (and let $i = -1$ if there is no such index). If $\alpha_{r_e(i+1)+1}[s] \subseteq \Phi^C_e[s]$, then we set $\sigma^{i+1}_{e,k} = \alpha_{r_e(i+1)+1}[s]$ and define $r_e(i+2)$ large (if it is not yet defined). As a technical point, in the full construction, it will be convenient to keep the indices $r_e(i)$ for different values of e and i spread out. Therefore, in addition to choosing $r_e(i+1)$ large, we will also make sure it is even.

The key property of each of these $\sigma^i_{e,n}$ strings is similar to that shown for $\sigma^0_{e,0}$. Suppose $\sigma^i_{e,n} = \alpha_{r_e(i)}[s_0]$ is defined at stage s_0. If the approximation A_s ever moves away from $\sigma^i_{e,n}$ after s_0 and later returns to $\sigma^i_{e,n}$ at stage $s' > s_0$, then $\alpha_{r_e(i)}[s'] \not\subseteq \Phi^C_e[s']$ and we have (at least temporarily) satisfied $\mathcal{R}_e$. In the end, either we settle permanently on such a string $\sigma^i_{e,n}$ (and win $\mathcal{R}_e$ permanently) or we stop seeing correct computations $\Phi^C_e[s]$ for initial segments of A (and hence win $\mathcal{R}_e$) or $\mathcal{R}_e$ acts infinitely often. If $\mathcal{R}_e$ acts infinitely often, then we can restrict our attention to stages at which we define strings $\sigma^i_{e,n}$. By the argument given for $\sigma^0_{e,0}$, we know that whenever we return to a previously defined $\sigma^i_{e,n}$ we do not have the appropriate computations to define a new σ string. Therefore, by restricting to these stages, once we move away from a string $\sigma^i_{e,n}$, we can never return to this string. This property is exactly the property of an almost c.e. approximation that is missing from the set of $\alpha_k[s]$ strings. In this way, we will extract an almost c.e. approximation for A in the case when $\mathcal{R}_e$ acts infinitely often. Since A does not have an almost c.e. approximation, the action of $\mathcal{R}_e$ must be finitely and we eventually win $\mathcal{R}_e$ permanently.

4.3.4 Formal construction

Each $\mathcal{R}_e$ requirement defines an increasing sequence of parameters $r_e(0) < r_e(1) < \cdots$ and uses the associated strings $\alpha_{r_e(i)}[s]$ to build a c.e. set of strings $\{\sigma^i_{e,u} \mid i, u \in \omega\}$ which threatens to generate an almost c.e. approximation to A. Each $\mathcal{P}_e$ requirement defines an increasing sequence of parameters $p_e(0) < p_e(1) < \cdots$ with associated marks $m_e(i)$. It uses $\alpha_{p_e(i)}$ to attempt to compute A and uses $m_e(i)$ to attempt to diagonalize making C noncomputable. During the construction the $\mathcal{P}$ requirements will place marks while the $\mathcal{R}$ requirements will remove them with respect to certain neighborhoods.

At stage 0, we initialize every requirement. This means we make every parameter associated with a requirement undefined. As usual we assume that the value of the parameters $r_e(i), p_e(i)$ and $m_e(i)$ are always larger than the last stage where the requirement is initialized. When $\mathcal{P}_e$ is initialized, the parameters $m_e(i)$ become undefined but we do not remove the marks previously set by $\mathcal{P}_e$. Once set, a mark can only be removed by an $\mathcal{R}$ requirement.

At stage $s > 0$, we define what it means for a requirement to require and to get attention.

For $\mathcal{P}_e$, let i_0 be the largest number (if any) such that $p_e(i_0)$ is currently defined. $\mathcal{P}_e$ requires attention if one of the following holds.

($\mathcal{P}_e$.1) The number i_0 is undefined, i.e. $p_e(0)$ is not currently defined.

($\mathcal{P}_e$.2) There is a number $m < s$ never used by any requirement such that

- $\alpha_{p_e(i_0)}[t] = \alpha_{p_e(i_0)}[m]$ for all stages t such that $m \leq t \leq s$,
- $m > |\alpha_{p_e(i_0)+1}[u]|$ for $u \leq s$,
- $m > m_e(i_0 - 1)$ and
- $\Delta_e \restriction m = C_s \restriction m$.

To give $\mathcal{P}_e$ attention in ($\mathcal{P}_e$.1), we set $p_e(0)$ to be a large number. In ($\mathcal{P}_e$.2), we set $m_e(i_0) = m$, set $p_e(i_0 + 1)$ to be a large number and mark the number $m_e(i_0)$.

For $\mathcal{R}_e$, let i be the largest such that $\sigma^i_{e,u} \subseteq A_s$ for some u, and $i = -1$ if no such i is found. $\mathcal{R}_e$ requires attention if one of the following holds.

($\mathcal{R}_e$.1) $r_e(0)$ is undefined.

($\mathcal{R}_e$.2) $\alpha_{r_e(i+1)+1}[s] \subseteq \Phi^C_e[s]$.

To give $\mathcal{R}_e$ attention in ($\mathcal{R}_e$.1), we set $r(0)$ to be a large even number. In ($\mathcal{R}_e$.2), we declare $\sigma_{e,v}^{i+1} = \alpha_{r_e(i+1)}[s]$ for the least v such that $\sigma_{e,v}^{i+1}$ has not yet received a value. If $r_e(i+2)$ is undefined, we pick a large even value for it and otherwise we leave the value as previously defined. For every number $n > |\alpha_{r_e(i+1)+1}[s]|$ such that $C_s(n) = 0$, we remove the mark on n with respect to the neighborhood $N(r_e(i+1), s)$.

At stage s the construction, we pick the highest priority requirement requiring attention from amongst the first s many requirements, give it attention according the description above, initialize all lower priority requirements and go to the next stage. This ends the description of the construction.

4.3.5 Verification

The verification of the construction is given by the following series of lemmas.

Lemma 4.18. *Fix a number m marked by a $\mathcal{P}$ requirement at stage s_1. If the mark on m is removed with respect to a neighborhood $N(r_1, t_1)$ which applies at A_s, then $s_1 < t_1 < s$.*

Proof. Since marks are not removed before they are set, we have $s_1 < t_1$. For the neighborhood $N(r_1, t_1)$ to apply to A_s, we must have $t_1 \leq s$ and furthermore $\alpha_{r_1+1}[u] \not\subseteq A_s$ for all $u \leq t_1$, which implies that $t_1 < s$. □

Lemma 4.19. *For each m, k and stages $t < s$, if there are requests to remove the mark on m with respect to both $N(k, t)$ and $N(k, s)$, then $\alpha_k[s] \neq \alpha_k[t]$.*

Proof. Suppose the mark on m is removed with respect to $N(k, t)$ by $\mathcal{R}_e$. At stage t, we must have $k = r_e(i+1)$ where i is the largest number such that $\sigma_{e,u}^i \subseteq A_t$ for some u. Furthermore, $m > |\alpha_{r_e(i+1)+1}[t]| = |\alpha_{k+1}[t]|$ and we set $\sigma_{e,v}^{i+1} = \alpha_{r_e(i+1)}[t] = \alpha_k[t]$ for some v.

Assume the mark on m is removed with respect to $N(k, s)$ at a stage $s > t$. Because $k = r_e(i+1)$ at stage t and because we always choose witnesses for $\mathcal{R}$ requirements fresh, the removal of the mark on m with respect to $N(k, s)$ must be done by the requirement $\mathcal{R}_e$ and this requirement cannot have been initialized between stages t and s. Therefore, $\sigma_{e,v}^{i+1} = \alpha_k[t]$ has retained its value at stage s.

Assume for a contradiction that $\alpha_k[s] = \alpha_k[t]$. Then $\sigma_{e,v}^{i+1} = \alpha_k[s] \subseteq A_s$ at stage s. By construction, if $\mathcal{R}_e$ removes a mark at stage s, it must be with respect to a neighborhood of the form $N(r_e(j+1), s)$ for $j \geq i+1$. In particular, the first coordinate of this neighborhood cannot be equal to k, giving the desired contradiction. □

Lemma 4.20. *Let $s_1 < s_2$ be stages and m be a number such that there is a mark on m which applies at s_2. Assume the mark on m was set before s_1 and that $A_{s_1} \upharpoonright m \not\subset A_v$ for all v such that $m < v < s_1$. If $A_{s_1} \upharpoonright m = A_{s_2} \upharpoonright m$, then the mark on m also applies at the earlier stage s_1.*

Proof. Assume for a contradiction that the mark on m does not apply at stage s_1. Since the mark was set before s_1, the mark must have been removed with respect to some neighborhood $N(k, s_0)$ at a stage $s_0 < s_1$ such that $N(k, s_0)$ applies at stage s_1. The mark on m must be set before it is removed at s_0, so $m < s_0$. The neighborhood $N(k, s_0)$ applies at s_1, so we have $\alpha_k[s_0] \subseteq A_{s_1}$ but $\alpha_{k+1}[u] \not\subseteq A_{s_1}$ for all $u \leq s_0$. Since the mark on m is removed at s_0, we have $m > |\alpha_{k+1}[s_0]|$. Because $|\alpha_k[s_0]| < |\alpha_{k+1}[s_0]| < m$ and $\alpha_k[s_0] \subseteq A_{s_1}$, we have $\alpha_k[s_0] \subseteq A_{s_1} \upharpoonright m$.

Claim 4.21. $A_{s_1} \upharpoonright m$ is incomparable with each $\alpha_{k+1}[u]$ for all $u \leq s_0$.

Before proving this claim, we show how to use it to finish the proof of Lemma 4.20. To do so, we show that the neighborhood $N(k, s_0)$ applies at s_2 and hence the mark on m doesn't apply at s_2 giving the desired contradiction. We need to check the two conditions for $N(k, s_0)$ to apply at s_2. First, $\alpha_k[s_0] \subseteq A_{s_2}$ because $\alpha_k[s_0] \subseteq A_{s_1} \upharpoonright m = A_{s_2} \upharpoonright m$. Second, for each $u \leq s_0$, $\alpha_{k+1}[u] \not\subseteq A_{s_2}$ because $\alpha_{k+1}[u]$ is incomparable with $A_{s_1} \upharpoonright m = A_{s_2} \upharpoonright m$ by the claim. Therefore, if we can verify the claim, the proof will be complete.

To prove the claim, fix $u \leq s_0$ and we split into several cases. If $|\alpha_{k+1}[u]| \leq m$, then since $\alpha_{k+1}[u] \not\subseteq A_{s_1}$, it follows that $\alpha_{k+1}[u]$ is incomparable with $A_{s_1} \upharpoonright m$. Therefore, we can assume that $|\alpha_{k+1}[u]| > m$. If $m < u$, then we have $m < u \leq s_0 < s_1$ and so $m < u < s_1$. It follows that $A_{s_1} \upharpoonright m \not\subseteq A_u$ by the condition on s_1 in the statement of the lemma (setting $v = u$). Since $\alpha_{k+1}[u] \subseteq A_u$, we have $A_{s_1} \upharpoonright m \not\subseteq \alpha_{k+1}[u]$ and so these strings are incomparable as required.

The final case to consider to prove the claim is when $|\alpha_{k+1}[u]| > m$ and $u \leq m$. We show that this case cannot occur given the assumptions of the lemma. Since $\alpha_{k+1}[u]$ is defined, we know $k + 1 \leq u$ and hence $k < u \leq m < s_0 < s_1$. Let $\mathcal{R}_i$ be the requirement that removes the mark on m, so $k = r_i(n)$ for some n. Let $\mathcal{P}_j$ be the requirement which sets the mark on $m = m_j(\ell)$ for some ℓ, which must occur before the mark is removed at stage s_0. $\mathcal{R}_i$ cannot be initialized between the stage at which is defines $k = r_i(n)$ and stage s_0 when it removes the mark on m with respect to $N(k, s_0)$. If $\mathcal{R}_i$ defines $r_i(n) = k$ after the mark on m is set by $\mathcal{P}_j$, then $\mathcal{R}_i$ would define $r_i(n) = k > m$. Since $k < m$, this implies that $\mathcal{R}_i$ must define $r_i(n) = k$ before the mark is set on m by $\mathcal{P}_j$. Therefore, $\mathcal{P}_j$ must have lower priority than $\mathcal{R}_i$ because otherwise it would initialize $\mathcal{R}_i$ (and cancel $r_i(n) = k$) when it sets the mark on m.

By the previous paragraph, we know that the order of events is as follows. $\mathcal{R}_i$ defines $r_i(n) = k$ initializing the lower priority $\mathcal{P}_j$. Later, $\mathcal{P}_j$ defines $p_j(\ell) > k$ and eventually sets a mark on $m = m_j(\ell)$, necessarily at a stage after m. When $\mathcal{P}_j$ sets the mark on m, it chooses m greater than the maximum of all values of $|\alpha_{p_j(\ell)+1}[w]|$ for all $w \leq$ the stage at which the mark is set. Since the mark is set after stage m, $u \leq m$ and $k < p_j(\ell)$, it follows that $|\alpha_{k+1}[u]| < m$. However, our case assumption was that $|\alpha_{k+1}[u]| > m$ so we have obtained the desired contradiction. $\square$

Lemma 4.22. *Suppose that requirement $\mathcal{P}$ marks a number $m = m(i_0)$ at stage s_0. If $s > s_0$ is such that $\alpha_{p(i_0)}[s_0] \not\subseteq A_s$, then at every future stage $s' \geq s$, as long as m is still marked (i.e. the mark has not be removed with respect the a neighborhood applicable at s'), we have $C_{s'}(m) = 1$.*

Proof. Let $p = p(i_0)$. First, we show that if $s_1 > s_0$ is the least stage such that $\alpha_p[s_0] \not\subseteq A_{s_1}$, then assuming m is still marked at s_1, $C_{s_1}(m) = 1$. Since the mark on m has not been removed with respect to a neighborhood which applies at s_1, by the definition of $C_{s_1}(m)$, it suffices to show that $A_v \upharpoonright m \not\subseteq A_{s_1}$ for all stages v such that $m < v < s_1$.

When m is marked at stage s_0, Condition ($\mathcal{P}$.2) must hold, so for all v such that $m \leq v \leq s_0$, we have $\alpha_p[v] = \alpha_p[s_0]$. Furthermore, by the choice of s_1, we have $\alpha_p[v] = \alpha_p[s_0]$ for all v such that $s_0 \leq v < s_1$, and therefore, $\alpha_p[v] = \alpha_p[s_0]$ for all v such that $m \leq v < s_1$.

In addition, when m is marked at stage s_0, we have $m > |\alpha_{p+1}[s_0]| > |\alpha_p[s_0]|$. By the previous paragraph, this inequality implies $m > |\alpha_p[v]|$, and hence $\alpha_p[v] \subseteq A_v \upharpoonright m$, for all $m \leq v < s_1$. Since $\alpha_p[v] = \alpha_p[s_0] \not\subseteq A_{s_1}$, we have shown that $A_v \upharpoonright m \not\subset A_{s_1}$ for all v such that $m \leq v < s_1$ as required to prove the lemma for s_1.

To complete the proof, we consider stages $s' > s_1$ such that the mark has not been removed from m with respect to a neighborhood applicable at s'. Assume for a contradiction that $s' > s$ is the least stage such that m is marked at s' but $C_{s'}(m) = 0$. We verify three claims and then break our proof into cases.

Our first claim is that there is a stage v such that $m < v < s'$ with $A_v \upharpoonright m \subset A_{s'}$. If there were no such stage v, then since there is a mark on m at s', we would define $C_{s'}(m) = 1$. Therefore, fix the least v such that $m < v < s'$ and $A_v \upharpoonright m \subset A_{s'}$.

Our second claim is that $C_v(m) = 0$. By Lemma 4.17, $A_v \upharpoonright m \subset A_{s'}$ implies $C_{s'}(m) = C_v(m)$ and hence $C_v(m) = 0$.

Finally, our third claim is that the mark on m does not apply at v. By the minimality of v, $A_w \upharpoonright m \not\subset A_v$ for all $m < w < v$. Therefore, if a mark on m applied at v, we would define $C_v(m) = 1$ contrary to our second claim. We now split into cases depending on whether $v > s_0$ or $v \leq s_0$.

For the first case, assume that $s_0 < v$. By the minimality of v, the fact that $s_0 < v < s'$ with the mark on m set at s_0 and the assumption that the mark on m applies at s', it follows from Lemma 4.20 that the mark on m applies at stage v (setting the values $s_1 = v$ and $s_2 = s'$ in Lemma 4.20). This conclusion contradicts the third claim above.

For the second case, assume that $v \leq s_0$. In this case, we have $m < v \leq s_0 < s_1$. By the second paragraph of this proof, these inequalities imply that $\alpha_p[v] = \alpha_p[s_0]$. Since $\mathcal{P}$ marks m at s_0, we have $m > |\alpha_{p+1}[v]| > |\alpha_p[v]|$. Since $A_v \upharpoonright m \subset A_{s'}$, it follows that $\alpha_p[s_0] \subseteq A_{s'}$ and $\alpha_{p+1}[v] \subseteq A_{s'}$, and hence $\alpha_p[s_0] = \alpha_p[s']$ and $\alpha_{p+1}[v] = \alpha_{p+1}[s']$. We now have that $s_0 < s_1 < s'$ with $\alpha_p[s_0] = \alpha_p[s']$ but $\alpha_p[s_0] \neq \alpha_p[s_1]$. Therefore, $\alpha_p[s']$ is reverting to a previously defined value and so by Lemma 4.15, $\alpha_{p+1}[s'] \neq \alpha_{p+1}[t]$ for all $t \leq s_1$. In particular, $\alpha_{p+1}[s'] \neq \alpha_{p+1}[v]$ giving the desired contradiction. □

Lemma 4.23. *Let s be a stage and let m be a number which is marked at stage $s_1 < s$ such that the mark on m has been removed with respect to a neighborhood $N(r_1, t_1)$ which applies at A_s. If $\alpha_{r+1}[s] \neq \alpha_{r+1}[u]$ for all stages u such that $s_1 < u < s$, then $r_1 \leq r$.*

Proof. Since the mark is removed with respect to $N(r_1, t_1)$ which applies to A_s, we have $s_1 < t_1 < s$ (by Lemma 4.18) and $\alpha_{r_1}[t_1] \subseteq A_s$. Since $\alpha_{r_1}[t_1] \subseteq A_s$, we have $\alpha_{r_1}[t_1] = \alpha_{r_1}[s]$. Suppose for a contradiction that $r < r_1$. Because $\alpha_{r_1}[t_1] = \alpha_{r_1}[s]$ and $r + 1 \leq r_1$, we have $\alpha_{r+1}[s] = \alpha_{r+1}[t_1]$ contradicting the assumption on the values of α_{r+1}. □

We say that $A_t \upharpoonright m$ is *new at t* if $m < t$ and $A_t \upharpoonright m \neq A_u \upharpoonright m$ for all u such that $m < u < t$. Note that if $A_t \upharpoonright m$ is new, then $C_t(m) = 1$ if and only if there is a mark on m which has not been removed with respect to a neighborhood which applies to A_t. Similarly, if $v \leq t$ is such that $A_v \upharpoonright m$ is new at v and $A_t \upharpoonright m = A_v \upharpoonright m$, then $C_t(m) = 1$ if and only if there is a mark on m at stage v which has not been removed with respect to a neighborhood which applies to A_v.

Lemma 4.24. *Fix $\mathcal{R}_e$ and assume it is never initialized again. Let s_0 be a stage at which $\mathcal{R}_e$ defines $r_e(i) = r$, let $s_2 > s_0$ be such that $\mathcal{R}_e$ defines $\sigma^i_{e,n} = \alpha_r[s_2]$ and let $s_3 > s_2$ be the least stage such that $\alpha_r[s_3] \neq \alpha_r[s_2]$. For all $t > s_3$, if $\alpha_r[t] = \alpha_r[s_2]$, then $C_t \upharpoonright s_2 = C_{s_2} \upharpoonright s_2$.*

Lemma 4.24 is the heart of our verification. To see why, notice that the at stage s_2, we have $\alpha_{r+1}[s_2] \subseteq \Phi^C_e[s_2]$ because $\mathcal{R}_e$ defines $\sigma^i_{e,n} = \alpha_r[s_2]$ and the use of this computation is bounded by s_2. Lemma 4.24 implies that C_t and C_{s_2} agree up to this use and therefore $\alpha_{r+1}[s_2] \subseteq \Phi^C_e[t]$. Since $\alpha_r[s_2] = \alpha_r[t]$ but $\alpha_r[s_2] \neq \alpha_r[s_3]$, we know $\alpha_{r+1}[t]$ is incomparable with $\alpha_{r+1}[s_2]$ and hence $\alpha_{r+1}[t] \not\subseteq \Phi^C_e[t]$.

Proof. Fix $s_0 < s_2 < s_3$ as in the statement of the lemma. By the stretching condition (C3), we have $|\alpha_{r+1}[t]| \geq s_2$. Since C is computed from A with identity bounded use, the value of $C_t \upharpoonright s_2$ is determined by the string $\alpha_{r+1}[t]$.

Consider which numbers $m \leq s_2$ could potentially lead to a difference between $C_t(m)$ and $C_{s_2}(m)$. If $m \leq |\alpha_r[s_2]|$, then $C_t(m) = C_{s_2}(m)$ because $\alpha_r[t] = \alpha_r[s_2]$ and the computation of C from A has identity bounded use. If m is never marked, then $C_t(m) = C_{s_2}(m) = 0$. Therefore, we may assume that $m > |\alpha_r[s_2]|$ and that m is marked at some stage.

If m is marked by a $\mathcal{P}$ strategy of higher priority than $\mathcal{R}_e$, then $\mathcal{P}$ initializes $\mathcal{R}_e$ when m is marked. This marking must come before $r_e(i) = r$ is defined at stage s_0 as $\mathcal{R}_e$ is never initialized after s_0 by assumption. In this case, $\mathcal{R}_e$ would define $r_e(i) = r > m$, so $m < r < |\alpha_r[s_2]|$ contrary to our assumption (from the previous paragraph) that $m > |\alpha_r[s_2]|$. Therefore, we may assume m is marked by a $\mathcal{P}$ strategy of lower priority than $\mathcal{R}_e$. If the lower priority $\mathcal{P}$ strategy marked m before stage s_0, then we would also define $r_e(i) = r > m$, so we may assume that $\mathcal{P}$ marks m after stage s_0.

When $\mathcal{R}_e$ defines $\sigma^i_{e,n} = \alpha_r[s_2]$ at stage s_2, it initializes all lower priority $\mathcal{P}$ strategies. Before a lower priority strategy $\mathcal{P}_j$ can mark another number $m = m_j(k)$, it must first define $p_j(k)$ and m must be a stage number after $\alpha_{p_j(k)}$ has been defined. So, if m is marked after $\mathcal{P}_j$ is initialized at s_2, then $s_2 < m$. Therefore, we can assume that m is marked before stage s_2.

Summing up this discussion, it suffices to prove that $C_t(m) = C_{s_2}(m)$ for all numbers $m > |\alpha_r[s_2]|$ which are marked by a lower priority strategy $\mathcal{P}$ between stages s_0 and s_2. For the remainder of the proof, assume that $\mathcal{P} = \mathcal{P}_j$ is a strategy of lower priority than $\mathcal{R}_e$ which marks $m = m_j(k)$ at stage s_1 with $s_0 < m < s_1 < s_2$. Let $p = p_j(k)$ be the associated index value at stage s_1 and let $\alpha_p[s_1]$ be the associated string. By the conditions in ($\mathcal{P}_j$.2), the string α_p is constant on the interval $[m, s_1]$ of stages. By the initialization at s_0, we have $r < p$ and hence the strings α_r and α_{r+1} are also constant on the interval $[m, s_1]$ of stages. When $\mathcal{P}$ marks m at s_1, it satisfies $|\alpha_{p+1}[u]| < m$ for all $u \leq s_1$ and hence $|\alpha_{r+2}[u]| < m$ for all $u \leq s_1$. To complete the proof, we need to show that $C_t(m) = C_{s_2}(m)$ where $t > s_3$ is an arbitrary stage at which $\alpha_r[t] = \alpha_r[s_2]$. We summarize this information for later reference.

(A1) The events at stages $s_0 < m < s_1 < s_2 < s_3 < t$ are as follows.

- At s_0, $\mathcal{R}_e$ defines $r_e(i) = r$.
- At s_1, the lower priority $\mathcal{P}$ marks m with associated string $\alpha_p[s_1]$.
- The strings α_p, α_r and α_{r+1} are constant on the stages in $[m, s_1]$.
- At s_2, $\mathcal{R}_e$ defines $\sigma^i_{e,n} = \alpha_r[s_2]$.
- The stage $s_3 > s_2$ is the least such that $\alpha_r[s_3] \neq \alpha_r[s_2]$.
- At $t > s_3$, we have $\alpha_r[t] = \alpha_r[s_2]$.

(A2) $|\alpha_r[s_2]| = |\alpha_r[t]| < m < s_2 \leq |\alpha_{r+1}[t]|$.

(A3) $r < p$ and $|\alpha_{r+2}[u]| < m$ for all $u \leq s_1$.

For the remainder of this proof, fix v to be the least stage such that $m < v \leq t$ and $A_v \upharpoonright m = A_t \upharpoonright m$. Thus, $A_v \upharpoonright m$ is new at stage v and $C_t(m) = C_v(m)$. Similarly, fix v_2 to be the least stage such that $m < v_2 \leq s_2$ and $A_{v_2} \upharpoonright m = A_{s_2} \upharpoonright m$. Again, $A_{v_2} \upharpoonright m$ is new at stage v_2 and $C_{s_2}(m) = C_{v_2}(m)$. Because $\alpha_r[s_2] = \alpha_r[t]$ has length less than m, we have

$$\alpha_r[v_2] = \alpha_r[s_2] = \alpha_r[t] = \alpha_r[v]. \tag{4.1}$$

If $A_t \upharpoonright m = A_{s_2} \upharpoonright m$, then we immediately get $C_t(m) = C_{s_2}(m)$. Therefore, we can assume that $v \neq v_2$ and $v \neq s_2$.

Claim 4.25. $|\alpha_{r+1}[v]| > m$ and $\alpha_{r+1}[v] \neq \alpha_{r+1}[u]$ for all $u < v$.

Proof. Suppose $|\alpha_{r+1}[v]| \leq m$. Since $A_v \upharpoonright m = A_t \upharpoonright m$ and $|\alpha_{r+1}[v]| \leq m$, we have $\alpha_{r+1}[t] = \alpha_{r+1}[v]$ by Lemma 4.11 and hence $|\alpha_{r+1}[t]| \leq m$ contradicting (A2). For the second statement, we have $\alpha_{r+1}[v] \neq \alpha_{r+1}[u]$ for all $u \leq s_1$ since

$|\alpha_{r+1}[u]| < m$ for $u \leq s_1$ by (A3). Because $A_v \upharpoonright m$ is new at v, we know $A_v \upharpoonright m \neq A_u \upharpoonright m$ for all u such that $m < u < v$. Since $A_v \upharpoonright m \subseteq \alpha_{r+1}[v]$, it follows that $\alpha_{r+1}[v] \neq \alpha_{r+1}[u]$ for all u such that $m < u < v$. Since $m < s_1$, these two cases cover all $u < v$. □

Claim 4.26. $s_1 < v$.

Proof. Suppose $v \leq s_1$. By (A3), $|\alpha_{r+1}[v]| < m$ which contradicts Claim 4.25. □

The importance of Claim 4.26 is that we know m has been marked by $\mathcal{P}$ before stage v. Therefore, $C_v(m) = 1$ (and hence $C_t(m) = 1$) if and only if the mark on m has not been removed with respect to a neighborhood which applies to A_v.

Claim 4.27. If an $\mathcal{R}$ requirement removes the mark on m with respect to a neighborhood $N(r_1, t_1)$ which applies to A_v, then $r_1 \leq r$.

Proof. Since the mark on m is set at $s_1 < v$ and $\alpha_{r+1}[v] \neq \alpha_{r+1}[u]$ for $u < v$, this claim follows from Lemma 4.23. □

The final claim is stated in a general form because we will later apply it in cases with $k = r + 1$ and $k = r + 2$.

Claim 4.28. For any index k, if $|\alpha_k[s_2]| \geq m$, then $|\alpha_k[v_2]| \geq m$ and $\alpha_k[v_2] \neq \alpha_k[u]$ for all u such that $m < u < v_2$ and hence for all u such that $s_1 < u < v_2$.

Proof. For a contradiction, assume that $|\alpha_k[v_2]| < m$. Because $A_{v_2} \upharpoonright m = A_{s_2} \upharpoonright m$, we have $\alpha_k[s_2] = \alpha_k[v_2]$ and hence $|\alpha_k[s_2]| < m$ contradicting the hypothesis of this claim. For the second part, assume $m < u < v_2$. If $|\alpha_k[u]| < m$ then $\alpha_k[u] \neq \alpha_k[v_2]$ because $|\alpha_k[v_2]| \geq m$. If $|\alpha_k[u]| \geq m$, then $\alpha_k[u] \neq \alpha_k[v_2]$ since $A_u \upharpoonright m \neq A_{v_2} \upharpoonright m$ (because $A_{v_2} \upharpoonright m$ is new at v_2) and $A_{v_2} \upharpoonright m \subseteq \alpha_k[v_2]$. □

We now proceed to the main part of the proof of Lemma 4.24 by breaking into three cases.

Case 1. Assume that $\mathbf{C_{s_2}(m) = 1}$. Our goal is to show that $C_v(m) = 1$. As noted after Claim 4.26, it suffices to show that the mark on m has not been removed with respect to a neighborhood $N(r_1, t_1)$ which applies to A_v. For a contradiction, assume that the mark has been removed by some $\mathcal{R}$ requirement with respect to such a neighborhood $N(r_1, t_1)$. By Claim 4.27, $r_1 \leq r$. Since m is marked at s_1 and $N(r_1, t_1)$ applies to A_v, we have $s_1 < t_1 < v$ by Lemma 4.18. Furthermore, because $N(r_1, t_1)$ applies to A_v, $\alpha_{r_1}[t_1] \subseteq A_v$ and so

$$\alpha_{r_1}[t_1] = \alpha_{r_1}[v] \tag{4.2}$$

but $\alpha_{r_1+1}[u] \not\subseteq A_v$ for all $u \leq t_1$. We break into cases depending on whether $r_1 = r$ or $r_1 < r$.

First, suppose that $r_1 = r$. Since $r = r_e(i)$ is an $\mathcal{R}_e$ parameter, the removal of the mark with respect to the neighborhood $N(r_1, t_1)$ is done by $\mathcal{R}_e$ at stage

t_1 in conjunction with defining $\sigma^i_{e,n'} = \alpha_{r_1}[t_1]$ for some n'. Because $r_1 = r$ and $\alpha_{r_1}[t_1] = \alpha_{r_1}[v]$ by Equation (4.2), we have by Equation (4.1) that

$$\alpha_{r_1}[t_1] = \alpha_{r_1}[v] = \alpha_r[v] = \alpha_r[s_2] = \sigma^i_{e,n}.$$

Therefore, the removal of the mark on m is done by $\mathcal{R}_e$ when it defines $\sigma^i_{e,n} = \alpha_r[s_2]$ and so $t_1 = s_2$. However, $C_{s_2}(m) = 1$, so $\mathcal{R}_e$ does not remove the mark on m at $t_1 = s_2$ giving the desired contradiction.

Second, suppose that $r_1 < r$. We claim that the neighborhood $N(r_1, t_1)$ applies to A_{v_2}. This claim gives the desired contradiction because $A_{v_2} \upharpoonright m$ is new at v_2 so the value of $C_{v_2}(m)$ is determined by whether there is a mark on m which applies at stage v_2. Since the mark on m has been removed with respect to $N(r_1, t_1)$ which applies to A_{v_2}, we conclude that $C_{v_2}(m) = 0$ contradicting the case assumption that $C_{s_2}(m) = C_{v_2}(m) = 1$.

It remains to show that the neighborhood $N(r_1, t_1)$ applies to A_{v_2}. Because $r_1 + 1 \leq r$, Equation (4.1) implies

$$\alpha_{r_1+1}[v_2] = \alpha_{r_1+1}[s_2] = \alpha_{r_1+1}[t] = \alpha_{r_1+1}[v]. \tag{4.3}$$

First, we check that $t_1 < v_2$. Suppose for a contradiction that $v_2 \leq t_1$. By Equation (4.3), $\alpha_{r_1+1}[v_2] = \alpha_{r_1+1}[v]$. However, since $N(r_1, t_1)$ applies to A_v, $\alpha_{r_1+1}[v] \neq \alpha_{r_1+1}[u]$ for all $u \leq t_1$, and so in particular, $\alpha_{r_1+1}[v] \neq \alpha_{r_1+1}[v_2]$ for the desired contradiction.

Second, we check that $\alpha_{r_1}[t_1] \subseteq A_{v_2}$. Since $\alpha_{r_1}[t_1] = \alpha_{r_1}[v]$ by Equation (4.2) and $\alpha_{r_1}[v] = \alpha_{r_1}[v_2]$ by Equation (4.3), we conclude that $\alpha_{r_1}[t_1] = \alpha_{r_1}[v_2] \subseteq A_{v_2}$.

Finally, we check that $\alpha_{r_1+1}[u] \not\subseteq A_{v_2}$ for all $u \leq t_1$. Suppose for a contradiction that $\alpha_{r_1+1}[u] \subseteq A_{v_2}$ and $u \leq t_1$. Because $\alpha_{r_1+1}[u] \subseteq A_{v_2}$, we have $\alpha_{r_1+1}[u] = \alpha_{r_1+1}[v_2]$ and since $\alpha_{r_1+1}[v_2] = \alpha_{r_1+1}[v]$ by Equation (4.3), we conclude that $\alpha_{r_1+1}[u] = \alpha_{r_1+1}[v]$ and thus $\alpha_{r_1+1}[u] \subseteq A_v$. However, since $u \leq t_1$ and the neighborhood $N(r_1, t_1)$ applies to A_v, we know $\alpha_{r_1+1}[u] \not\subseteq A_v$ giving the desired contradiction. This completes the proof that the neighborhood $N(r_1, t_1)$ applies to A_{v_2} and so completes the proof of Case 1.

Case 2. Assume that $\mathbf{v > s_2}$ and $\mathbf{C_{s_2}(m) = 0}$. Our goal is to show that $C_v(m) = 0$. We split into cases depending on whether $m > |\alpha_{r+1}[s_2]|$ or $|\alpha_{r+1}[s_2]| \leq m$.

For the first case, suppose that $m > |\alpha_{r+1}[s_2]|$. Since $C_{s_2}(m) = 0$, $\mathcal{R}_e$ removes the mark on m with respect to the neighborhood $N(r, s_2)$ at stage s_2 when it defines $\sigma^i_{e,u} = \alpha_r[s_2]$. We will show that the neighborhood $N(r, s_2)$ applies to A_v completing this case because $A_v \upharpoonright m$ is new at v and hence $C_v(m) = 0$ because the mark on m does not apply at stage v.

To see that $N(r, s_2)$ applies to A_v, note that $s_2 < v$ by our Case 2 assumption and $\alpha_r[s_2] = \alpha_r[v] \subseteq A_v$ by Equation (4.1). Finally, $\alpha_{r+1}[u] \not\subseteq A_v$ for all $u \leq s_2$ by Claim 4.25 and our Case 2 assumption that $s_2 < v$. Therefore, the neighborhood $N(r, s_2)$ applies to A_v completing the first case.

For the second case, assume $m \leq |\alpha_{r+1}[s_2]|$. We claim that $m \leq |\alpha_{r+1}[v_2]|$. To see why, suppose $|\alpha_{r+1}[v_2]| < m$. Since $A_{v_2} \upharpoonright m = A_{s_2} \upharpoonright m$, it follows that $\alpha_{r+1}[v_2] = \alpha_{r+1}[s_2]$ and hence $|\alpha_{r+1}[s_2]| < m$, contradicting our case assumption that $m \leq |\alpha_{r+1}[s_2]|$. Therefore, $m \leq |\alpha_{r+1}[v_2]|$.

Next, we claim that $s_1 < v_2$, so $\mathcal{P}$ has marked m before stage v_2. To see why, suppose that $v_2 \leq s_1$. By (A3), $|\alpha_{r+1}[v_2]| < m$ contradicting the fact that $m \leq |\alpha_{r+1}[v_2]|$.

Because $A_{v_2} \upharpoonright m$ is new at v_2 and $A_{v_2} \upharpoonright m \subseteq \alpha_{r+1}[v_2]$, we have $\alpha_{r+1}[v_2] \neq \alpha_{r+1}[u]$ for all u such that $m < u < v_2$. Since $m < s_1 < v_2$, we know

$$\alpha_{r+1}[v_2] \neq \alpha_{r+1}[u] \text{ for all } s_1 \leq u < v_2 \tag{4.4}$$

At this point, we know $A_{v_2} \upharpoonright m$ is new at v_2, $C_{v_2}(m) = C_{s_2}(m) = 0$ and $\mathcal{P}$ marks m before stage v_2. Therefore, the mark on m must have been removed by some $\mathcal{R}$ requirement with respect to a neighborhood $N(r_1, t_1)$ which applies to A_{v_2} and so

$$s_1 < t_1 < v_2. \tag{4.5}$$

By Lemma 4.23 and Equation (4.4), we have $r_1 \leq r$. Because $N(r_1, t_1)$ applies to A_{v_2}, we know that $\alpha_{r_1}[t_1] \subseteq A_{v_2}$ and $\alpha_{r_1+1}[u] \not\subseteq A_{v_2}$ for all $u \leq t_1$. We will show that the neighborhood $N(r_1, t_1)$ applies to A_v which implies that $C_v(m) = 0$ as required.

We check the three conditions for $N(r_1, t_1)$ to apply to A_v. First, by Equation (4.5) and our Case 2 assumption that $s_2 < v$, we have $t_1 < v_2 \leq s_2 < v$ and so $t_1 < v$.

Second, we claim that $\alpha_{r_1}[t_1] \subseteq A_v$. Since $\alpha_{r_1}[t_1] \subseteq A_{v_2}$, we have $\alpha_{r_1}[t_1] = \alpha_{r_1}[v_2]$. Because $r_1 \leq r$ and $\alpha_r[v_2] = \alpha_r[v]$ by Equation (4.1), we conclude that $\alpha_{r_1}[t_1] = \alpha_{r_1}[v]$ and hence $\alpha_{r_1}[t_1] \subseteq A_v$.

Finally, we claim that $\alpha_{r_1+1}[u] \not\subseteq A_v$ for all $u \leq t_1$. If $r_1 = r$, this fact follows from Claim 4.25 because $t_1 < v$. If $r_1 < r$, then $r_1 + 1 \leq r$ and so $\alpha_{r_1+1}[v] = \alpha_{r_1+1}[v_2]$ by Equation (4.1). Since $\alpha_{r_1+1}[v_2] \neq \alpha_{r_1+1}[u]$ for all $u \leq t_1$, it follows that $\alpha_{r_1+1}[v] \neq \alpha_{r_1+1}[u]$ for all $u \leq t_1$ and hence $\alpha_{r_1+1}[u] \not\subseteq A_v$ for all $u \leq t_1$. This completes the proof that the neighborhood $N(r_1, t_1)$ applies to A_v and so completes the proof of Case 2.

Case 3. Assume that $\mathbf{v < s_2}$ and $\mathbf{C_{s_2}(m) = 0}$. Our goal is to show that $C_v(m) = 0$. We split into cases for $|\alpha_{r+1}[s_2]| \leq m$ and $|\alpha_{r+1}[s_2]| > m$.

For the first case, suppose that $|\alpha_{r+1}[s_2]| \leq m$. We claim that $\alpha_{r+1}[s_2] = \alpha_{r+1}[u]$ for some $u < v$. To see why, note that $|\alpha_{r+1}[s_2]| \leq m < |\alpha_{r+1}[v]|$ by Claim 4.25. If $\alpha_{r+1}[s_2]$ first appeared after stage v, then by the stretching convention (C4), we would have $|\alpha_{r+1}[s_2]| \geq |\alpha_{r+1}[v]|$. Therefore, $\alpha_{r+1}[s_2]$ must be returning to a previous value from before stage v.

Let $u_2 < v$ be the least stage such that $\alpha_{r+1}[u_2] = \alpha_{r+1}[s_2]$, so $\alpha_{r+1}[u_2] \neq \alpha_{r+1}[x]$ for all $x < u_2$. Since $|\alpha_{r+1}[s_2]| \leq m$ and $A_{v_2} \upharpoonright m = A_{s_2} \upharpoonright m$, we have $\alpha_{r+1}[s_2] = \alpha_{r+1}[v_2]$. Therefore, $\alpha_{r+1}[u_2] = \alpha_{r+1}[v_2] = \alpha_{r+1}[s_2]$ but $\alpha_{r+1}[v]$ differs from these strings. Summarizing, we have

$$u_2 < v < s_2 \text{ with } \alpha_{r+1}[u_2] = \alpha_{r+1}[v_2] = \alpha_{r+1}[s_2] \text{ but } \alpha_{r+1}[u_2] \neq \alpha_{r+1}[v]. \tag{4.6}$$

We claim that $s_1 < v_2$. Suppose for a contradiction that $v_2 \leq s_1$. By (A3), $|\alpha_{r+2}[v_2]| < m$, and so because $A_{v_2} \upharpoonright m = A_{s_2} \upharpoonright m$, we have $\alpha_{r+2}[v_2] = \alpha_{r+2}[s_2]$. Our goal is to show this equality is impossible under the current assumptions. By Claim 4.26, our Case 3 assumption that $v < s_2$ and our local assumption that $v_2 \leq s_1$, these stages are ordered as $v_2 \leq s_1 < v < s_2$. By Equation (4.6), $\alpha_{r+1}[v_2] = \alpha_{r+1}[s_2]$ but $\alpha_{r+1}[v_2] \neq \alpha_{r+1}[v]$. Therefore, $\alpha_{r+2}[s_2]$ cannot return to the value of $\alpha_{r+2}[v_2]$ giving the desired contradiction.

Since $s_1 < v_2$, $\mathcal{P}$ has marked m before stage v_2. Since $C_{v_2}(m) = C_{s_2}(m) = 0$ and $A_{v_2} \upharpoonright m$ is new at v_2, the mark on m must have been removed by an $\mathcal{R}$ requirement with respect to a neighborhood $N(r_1, t_1)$ which applies to A_{v_2}. Therefore, we have

$$s_1 < t_1 < v_2 \text{ and } \alpha_{r_1}[t_1] = \alpha_{r_1}[v_2] \text{ but } \alpha_{r_1+1}[u] \not\subseteq A_{v_2} \text{ for all } u \leq t_1. \tag{4.7}$$

We claim that $r_1 \leq r$. Because r_1 and r are parameters chosen by $\mathcal{R}$ requirements, they are both even. Therefore, it suffices to show that $r_1 \leq r+1$. By Equation (4.6), we have $u_2 < v < s_2$, $\alpha_{r+1}[u_2] = \alpha_{r+1}[s_2]$ and $\alpha_{r+1}[v] \neq \alpha_{r+1}[u_2]$, and hence we conclude that $\alpha_{r+2}[s_2] \neq \alpha_{r+2}[u]$ for all $u \leq v$. In particular, by our stretching convention (C4), $|\alpha_{r+2}[s_2]| \geq |\alpha_{r+2}[u]|$ for all $u \leq v$. Because $|\alpha_{r+1}[v]| \geq m$, we have $|\alpha_{r+2}[s_2]| \geq m$. By Claim 4.28 with $k = r+2$, $\alpha_{r+2}[v_2] \neq \alpha_{r+2}[u]$ for all u such that $s_1 < u < v_2$. It follows by Lemma 4.23 that $r_1 \leq r+1$ and hence $r_1 \leq r$.

To show $C_v(m) = 0$ and complete the first case, it suffices to show that this neighborhood $N(r_1, t_1)$ applies to A_v.

First, we show that $t_1 < v$. Assume for a contradiction that $v \leq t_1$. Since $u_2 < v$ by Equation (4.6) and $t_1 < v_2$ by Equation (4.7), we have $u_2 < v \leq t_1 < v_2$. Because $\alpha_{r+1}[u_2] = \alpha_{r+1}[v_2]$ by Equation (4.6) and $r_1 \leq r$, we have $\alpha_{r_1+1}[u_2] = \alpha_{r_1+1}[v_2]$. However, $u_2 < t_1$ and $\alpha_{r_1+1}[u_2] = \alpha_{r_1+1}[v_2]$ contradict the fact from Equation (4.7) that $\alpha_{r_1+1}[u] \not\subseteq A_{v_2}$ for all $u \leq t_1$. Thus, we have shown $t_1 < v$.

Second, we show that $\alpha_{r_1}[t_1] \subseteq A_v$. Since $r_1 \leq r$ and, by Equation (4.1), $\alpha_r[v] = \alpha_r[v_2]$, we have $\alpha_{r_1}[v] = \alpha_{r_1}[v_2]$. But, by Equation (4.7), $\alpha_{r_1}[t_1] = \alpha_{r_1}[v_2]$, and so $\alpha_{r_1}[t_1] = \alpha_{r_1}[v] \subseteq A_v$ as required.

Third, we show that $\alpha_{r_1+1}[u] \not\subseteq A_v$ for all $u \leq t_1$. If $r_1 = r$, then this follows by Claim 4.25 and the fact that $t_1 < v$. If $r_1 < r$, then $r_1 + 1 \leq r$ and so $\alpha_{r_1+1}[v] = \alpha_{r_1+1}[v_2]$ by Equation (4.1). Since, by Equation (4.7), $\alpha_{r_1+1}[v_2] \neq \alpha_{r_1+1}[u]$ for all $u \leq t_1$, it follows that $\alpha_{r_1+1}[v] \neq \alpha_{r_1+1}[u]$ for all $u \leq t_1$ and hence $\alpha_{r_1+1}[u] \not\subseteq A_v$ for all $u \leq t_1$. This completes the first case in Case 3.

The remaining case in Case 3 is when $m < |\alpha_{r+1}[s_2]|$. By Claim 4.28 with $k = r+1$, $m \leq |\alpha_{r+1}[v_2]|$ and $\alpha_{r+1}[v_2] \neq \alpha_{r+1}[u]$ for all u such that $s_1 < u < v_2$. Note that we do have $s_1 < v_2$ (i.e. this interval of stages for u is not empty) since if $v_2 \leq s_1$, then by (A3), we would have $|\alpha_{r+1}[v_2]| < m$ for the desired contradiction.

Since $\mathcal{P}$ marked m at stage $s_1 < v_2$ and $C_{v_2}(m) = 0$ by our Case 3 assumption, the mark on m must have been removed by an $\mathcal{R}$ requirement with respect to a neighborhood $N(r_1, t_1)$ which applies to A_{v_2}. Thus, we have

$$s_1 < t_1 < v_2 \text{ and } \alpha_{r_1}[t_1] = \alpha_{r_1}[v_2] \text{ but } \alpha_{r_1+1}[u] \not\subseteq A_{v_2} \text{ for all } u \leq t_1. \quad (4.8)$$

Because $s_1 < v_2$ and $\alpha_{r+1}[v_2] \neq \alpha_{r+1}[u]$ for all u such that $s_1 < u < v_2$, it follows by Lemma 4.23 that $r_1 \leq r$. To prove $C_v(m) = 0$ and complete our final case, it suffices to show that the neighborhood $N(r_1, t_1)$ applies to A_v.

First, we show that $t_1 < v$. For a contradiction, assume that $v \leq t_1$. If $r_1 < r$, then $\alpha_{r_1+1}[v] = \alpha_{r_1+1}[v_2]$ by Equation (4.1) and so $\alpha_{r_1+1}[v] \subseteq A_{v_2}$. Since $v \leq t_1$, this contradicts Equation (4.8). Therefore, assume that $r_1 = r$. In this case, the neighborhood is $N(r, t_1)$ so the removal is done by $\mathcal{R}_e$. When $\mathcal{R}_e$ acts at t_1 to remove the mark on m with respect to $N(r, t_1)$, it defines $\sigma^i_{e,u'} = \alpha_r[t_1]$ for some u'. Since $t_1 < s_2$ and $\mathcal{R}_e$ defines $\sigma^i_{e,n} = \alpha_r[s_2]$ at s_2, this implies $\alpha_r[t_1] \neq \alpha_r[s_2]$. However, since $r_1 = r$ and $\alpha_{r_1}[t_1] = \alpha_{r_1}[v_2]$ by Equation (4.8), we have $\alpha_r[t_1] = \alpha_r[v_2]$. But, $\alpha_r[v_2] = \alpha_r[s_2]$ by Equation (4.1) and hence $\alpha_r[t_1] = \alpha_r[s_2]$ for the desired contradiction.

Second, $\alpha_{r_1}[t_1] \subseteq A_v$ because $\alpha_{r_1}[t_1] = \alpha_{r_1}[v_2]$ (by Equation (4.8)) and $\alpha_{r_1}[v_2] = \alpha_{r_1}[v]$ (by Equation (4.1) and $r_1 \leq r$), so $\alpha_{r_1}[t_1] = \alpha_{r_1}[v]$.

Finally, we show that $\alpha_{r_1+1}[u] \not\subseteq A_v$ for all $u \leq t_1$. If $r_1 = r$, then this follows from Claim 4.25 since $t_1 < v$. Therefore, suppose $r_1 < r$ and $u \leq t_1$ with $\alpha_{r_1+1}[u] \subseteq A_v$, so $\alpha_{r_1+1}[u] = \alpha_{r_1+1}[v]$. Since $\alpha_r[v] = \alpha_r[v_2]$, we have $\alpha_{r_1+1}[v] = \alpha_{r_1+1}[v_2]$ and hence $\alpha_{r_1+1}[u] = \alpha_{r_1+1}[v_2]$ contradicting Equation (4.8). This completes the proof of Case 3 and finishes the proof of our lemma. □

Lemma 4.29. *Each requirement is initialized finitely often.*

Proof. We proceed by induction on the ordering of requirements. Assume the requirement $\mathcal{P}_e$ is initialized finitely often. We argue that $\mathcal{P}_e$ receives attention finitely often. For a contradiction, suppose $\mathcal{P}_e$ receives attention infinitely often.

Consider the final version of $\mathcal{P}_e$ (i.e. assume we are past the last stage at which $\mathcal{P}_e$ is initialized). Marks set by the final version of $\mathcal{P}_e$ cannot be removed by a higher priority $\mathcal{R}$ strategy since the removal would initialize $\mathcal{P}_e$. Also, when a mark is set by the final version of $\mathcal{P}_e$, $\mathcal{P}_e$ initializes all lower priority $\mathcal{R}$ strategies. Any parameters chosen by these lower priority strategies in the future will be too large to remove the mark set by $\mathcal{P}_e$. Therefore, no number marked by the final version of $\mathcal{P}_e$ is removed with respect to any neighborhood. Because $\mathcal{P}_e$ acts infinitely often, p_e and m_e are defined on all inputs. Let $u_0 < u_1 < \cdots$ be the stages such that $p_e(i+1)$ and $m_e(i)$ are defined at stage u_i. At stage u_i, condition ($\mathcal{P}_e$.2) holds and since $m_e(i)$ is first marked at stage u_i, we must have $C_{u_i}(m_e(i)) = 0$ and hence $\Delta_e(m_e(i)) = 0$.

Claim 4.30. For all i and all $s > u_i$, $\alpha_{p_e(i)}[u_i] \subseteq A_s$.

Proof. For a contradiction, fix $s > u_i$ such that $\alpha_{p_e(i)}[u_i] \not\subseteq A_s$. Since the mark on $m_e(i)$ is never removed with respect to any neighborhood, we have $C_{s'}(m_e(i)) = 1$ for all $s' \geq s$ by Lemma 4.22. Fix $j > i$ such that $s < u_j$. Condition ($\mathcal{P}_e$.2) holds at u_j with $t = m_e(j)$, so $\Delta_e(m_e(i)) = C_{u_j}(m_e(i))$ because

$m_e(i) < m_e(j)$. However, $\Delta_e(m_e(i)) = 0$ and $C_{u_j}(m_e(i)) = 1$ for the desired contradiction. □

By Claim 4.30, $\alpha_{p_e(i)}[u_i] \subseteq A$ for every i and hence A is computable giving the contradiction which establishes that $\mathcal{P}_e$ receives attention only finitely often.

We turn to $\mathcal{R}_e$. Assume that $\mathcal{R}_e$ is initialized finitely often and we work after the last stage at which $\mathcal{R}_e$ is initialized so that any parameter $r_e(i)$ or string $\sigma^i_{e,u}$ which is defined retains its value through the remainder of the construction. Suppose for a contradiction that $\mathcal{R}_e$ receives attention infinitely often. In this case, ($\mathcal{R}_e$.2) must apply infinitely often. For each defined string $\sigma^i_{e,u}$, we have $\sigma^i_{e,u} = \alpha_{r_e(i)}[s]$ where s is the stage at which $\sigma^i_{e,u}$ is defined. For any fixed i, there are only finitely many versions of $\alpha_{r_e(i)}[s]$ (as a function of s) and hence there are only finitely many strings $\sigma^i_{e,u}$ defined for any fixed i. Therefore, as ($\mathcal{R}_e$.2) applies infinitely often, we must eventually define $r_e(i)$ for each i.

To obtain a contradiction, it suffices to define an almost c.e. approximation $\widehat{\sigma}_i[s]$ for A. Fix a sequence of stages $t_0 < t_1 < \cdots$ such that for all s and all $i < s$, there is a u such that $\sigma^i_{e,u} \subseteq A_{t_s}$ and such that ($\mathcal{R}_e$.2) applies at stage t_s. Note that the value of u depends on both i and s and that at stage t_s, $\sigma^i_{e,u} = \alpha_{r_e(i)}[t_s]$. Setting $i = s-1$ shows there is a defined string $\sigma^{s-1}_{e,u} \subseteq A_{t_s}$ so that $\sigma^s_{e,u} = \alpha_{r_e(s)}[t_s]$ is defined (for some u) at stage t_s if it is not already defined.

For all s and all $i < s$, we define

$$\widehat{\sigma}_i[s] = \sigma^i_{e,u} = \alpha_{r_e(i)}[t_s].$$

To complete the proof of this lemma, it suffices to show that these strings form an almost c.e. approximation to A. We check Conditions (P1)-(P4) of Definition 4.5.

For Condition (P1), fix s and $i < s-1$. We have $\widehat{\sigma}_i[s] \subseteq \widehat{\sigma}_{i+1}[s]$ because $\widehat{\sigma}_i[s] = \alpha_{r_e(i)}[t_s]$, $\widehat{\sigma}_{i+1}[s] = \alpha_{r_e(i+1)}[t_s]$ and $\alpha_{r_e(i)}[t_s] \subseteq \alpha_{r_e(i+1)}[t_s]$. For Condition (P2), we need to show that if $\widehat{\sigma}_i[s]$ and $\widehat{\sigma}_i[s+1]$ are comparable, then $\widehat{\sigma}_i[s] = \widehat{\sigma}_i[s+1]$. However, $\widehat{\sigma}_i[s] = \alpha_{r_e(i)}[t_s]$ and $\widehat{\sigma}_i[s+1] = \alpha_{r_e(i)}[t_{s+1}]$. By Lemma 4.13, if $\alpha_{r_e(i)}[t_s]$ and $\alpha_{r_e(i)}[t_{s+1}]$ are comparable, then they are equal. For Condition (P4),

$$\lim_s \widehat{\sigma}_i[s] = \lim_s \alpha_{r_e(i)}[t_s]$$

exists and is an initial segment of A.

It remains to verify Condition (P3). Fix s and $i < s$ such that $\widehat{\sigma}_i[s]$ and $\widehat{\sigma}_i[s+1]$ are incomparable and fix $k > s+1$. We show that $\widehat{\sigma}_i[s]$ is incomparable with $\widehat{\sigma}_i[k]$. Since $\widehat{\sigma}_i[s] = \alpha_{r_e(i)}[t_s]$ and $\widehat{\sigma}_i[k] = \alpha_{r_e(i)}[t_k] \subseteq A_{t_k}$, it suffices to show that if $t > t_{s+1}$ and $\alpha_{r_e(i)}[t_s] \subseteq A_t$ then ($\mathcal{R}_e$.2) does not hold at stage t. Therefore, fix $t > t_{s+1}$ and assume that $\alpha_{r_e(i)}[t_s] \subseteq A_t$ and hence $\alpha_{r_e(i)}[t_s] = \alpha_{r_e(i)}[t]$. Fix u such that $\widehat{\sigma}_i[s] = \sigma^i_{e,u}$ so that we have

$$\widehat{\sigma}_i[s] = \sigma^i_{e,u} = \alpha_{r_e(i)}[t_s] \subseteq A_t. \tag{4.9}$$

We show that ($\mathcal{R}_e$.2) does not apply at t.

Let $s_2 \leq t_s$ be the stage at which $\sigma^i_{e,u} = \alpha_{r_e(i)}[s_2]$ is defined. Since $\sigma^i_{e,u} = \alpha_{r_e(i)}[s_2] = \alpha_{r_e(i)}[t_s]$ and $\alpha_{r_e(i)}[t_s] = \alpha_{r_e(i)}[t]$, it follows that $\alpha_{r_e(i)}[s_2] = \alpha_{r_e(i)}[t]$. However, $\widehat{\sigma}_i[s] = \alpha_{r_e(i)}[t_s]$ and $\widehat{\sigma}_i[s+1] = \alpha_{r_e(i)}[t_{s+1}]$ are incomparable, so $\alpha_{r_e(i)}[t_s] \neq \alpha_{r_e(i)}[t_{s+1}]$ and hence $\alpha_{r_e(i)}[s_2] \neq \alpha_{r_e(i)}[t_{s+1}]$. Altogether, we have $s_2 < t_{s+1} < t$ with $\sigma^i_{e,u} = \alpha_{r_e(i)}[s_2]$ defined at stage s_2, $\alpha_{r_e(i)}[s_2] = \alpha_{r_e(i)}[t]$ and $\alpha_{r_e(i)}[s_2] \neq \alpha_{r_e(i)}[t_{s+1}]$. Therefore, by Lemma 4.24, $C_t \upharpoonright s_2 = C_{s_2} \upharpoonright s_2$.

Condition ($\mathcal{R}_e$.2) applies at stage s_2 when $\sigma^i_{e,u} = \alpha_{r_e(i)}[s_2]$ is defined so $\alpha_{r_e(i)+1}[s_2] \subseteq \Phi^C_e[s_2]$. Let $U < s_2$ denote the use of this computation. Since $\alpha_{r_e(i)}[s_2] = \alpha_{r_e(i)}[t] \neq \alpha_{r_e(i)}[t_{s+1}]$ with $s_2 < t_{s+1} < t$, $\alpha_{r_e(i)}[t]$ is returning to a previous value after changing at stage t_{s+1}. Therefore, $\alpha_{r_e(i)+1}[t] \neq \alpha_{r_e(i)+1}[s_2]$ and hence these strings are incomparable.

We are now in a position to show that ($\mathcal{R}_e$.2) does not apply at t. Because $\sigma^i_{e,u} \subseteq A_t$ by Equation (4.9), we need (at least) $\alpha_{r_e(i+1)+1}[t] \subseteq \Phi^C_e[t]$ for ($\mathcal{R}_e$.2) to apply at t. However, because $C_t \upharpoonright s_2 = C_{s_2} \upharpoonright s_2$, we have $\alpha_{r_e(i)+1}[s_2] \subseteq \Phi^C_e[t]$ and hence $\alpha_{r_e(i)+1}[t] \not\subseteq \Phi^C_e[t]$ because $\alpha_{r_e(i)+1}[t]$ and $\alpha_{r_e(i)+1}[s_2]$ are incomparable. Therefore, $\alpha_{r_e(i+1)+1}[t] \not\subseteq \Phi^C_e[t]$ as well and hence ($\mathcal{R}_e$.2) cannot apply at t as required. □

Lemma 4.31. *Each requirement is satisfied.*

Proof. By Lemma 4.29, each requirement receives attention finitely often. Obviously for $\mathcal{P}_e$ we cannot have $\Delta_e = C$, and for $\mathcal{R}_e$ we cannot have $\Phi^C_e = A$, otherwise the requirement would act infinitely often. □

This ends the proof of Theorem 1.2.

4.4 Proof of Theorem 1.3

In this section, we prove Theorem 1.3. For convenience, we restate it here. We refer the reader to Soare [34] for information on promptly simple sets and degrees, although below we state the property of promptly simple sets which we will use in the construction.

Theorem 1.3. *Let V be a promptly simple c.e. set and let A be a Δ^0_2 set such that $A \geq_T V$. There exists a c.e. set B such that $0 <_T B \leq_{wtt} A$.*

Before presenting the formal construction, we fix notation and give an intuitive sketch of how to meet one requirement. Let V and A be as in the statement of the theorem and fix a Turing reduction $\Gamma^A = V$. We speed up the Δ^0_2 approximation to A, the enumeration of V and the reduction Γ so that the length of agreement function

$$l(s) = \max\{x \mid \forall y \leq x\, (\Gamma^{A_s}_s(x) \downarrow = V_s(x))\}.$$

satisfies $l(s+1) > l(s)$ for all s. That is, we assume that every stage of our construction is expansionary. For $x \leq l(s)$, we use $\gamma(x, s)$ to denote the use of $\Gamma^{A_s}_s(x)$.

Because V is promptly simple, there is a fixed computable function $p(s)$ for which we have the following property for all e (see Soare [34] Chapter XIII, Theorem 1.7):

$$W_e \text{ infinite } \Rightarrow \exists^\infty x \exists s (x \in W_{e \,\mathrm{at}\, s} \wedge V_s \upharpoonright x \neq V_{p(s)} \upharpoonright x).$$

The notation $W_{e \,\mathrm{at}\, s}$ means that $x \in W_{e,s}$ and $x \notin W_{e,s-1}$.

To make B noncomputable, we meet the requirement

$$R_e : \ B \neq \overline{W_e}$$

for every e. R_e is met by choosing a witness which we attempt to put into B if it ever enters W_e. To make $B \leq_{wtt} A$, we use permitting to guarantee that

$$A_s \upharpoonright x = A \upharpoonright x \ \Rightarrow \ B_s \upharpoonright x = B \upharpoonright x$$

for every x, so the computation of B from A has identity bounded use.

Consider a single R_e requirement in the presence of our permitting. We attempt to meet R_e in cycles (which may be initialized by higher priority requirements, but only finitely often). The prompt simplicity of V will insure that only finitely many cycles are needed for R_e.

Assume that the n^{th} cycle for R_e starts at stage s. Pick a large prefollower z_n. (In the formal construction, we will denote such a witness by $z_{e,n}$ to indicate it is the n^{th} prefollower for R_e. For now, we leave off the subscript e since we are only considering one requirement.) Wait for a stage $s_1 > s$ such that $l(s_1) > z_n$. At stage s_1, pick a large follower $y_n^{s_1}$ such that $y_n^{s_1} > \gamma(z_n, s_1)$ and $y_n^{s_1} \notin W_{e,s_1}$. Notice that if there is a change in $V_{s_1} \upharpoonright z_n$, then there must be a corresponding change in $A_{s_1} \upharpoonright \gamma(z_n, s_1)$, which we would like to use as a permission to put $y_n^{s_1}$ into B.

We say $y_n^{s_1}$ is *realized* at $t > s_1$ if $y_n^{s_1} \in W_{e,t}$. We say that $y_n^{s_1}$ is *canceled* at stage $t > s_1$ if $\gamma(z_n, t) \neq \gamma(z_n, s_1)$ and $y_n^{s_1}$ has not yet been realized. If $y_n^{s_1}$ is canceled at stage t, then we pick a new large follower $y_n^t > \gamma(z_n, t)$ such that $y_n^t \notin W_{e,t}$. Since $t > s_1$, we have $l(t) > l(s_1) > z_n$ and so the computation $\Gamma_t^{A_t}(z_n)$ does converge and $\gamma(z_n, t)$ is defined. In general, we use the notation y_n^t for the follower of z_n at stage t, if there is one. Because there is a final use $\gamma(z_n)$ for $\Gamma^A(z_n)$, the sequence of followers for any given prefollower z_n is finite and must eventually settle down on a single follower.

Assume that at some stage $s_2 > s_1$, the current follower $y_n^{s_2}$ becomes realized (that is, it enters W_e at s_2). We want to use the prompt simplicity of V to get permission to put $y_n^{s_2}$ into B. Two technical problems arise at this point. Prompt simplicity tells us that if W_e is infinite, then there are infinitely many numbers $x \in W_e$ for which if x enters W_e at stage t, then a number below x must enter V between stage t and stage $p(t)$. The first technical problem is that $y_n^{s_2}$ may not be one of these infinitely many elements of W_e for which the condition of prompt simplicity holds. The second technical problem is that even if $y_n^{s_2}$ is one of the numbers for which the condition of prompt simplicity holds, it only causes a number below $y_n^{s_2}$ (and not necessarily below z_n) to enter V.

Numbers below $y_n^{s_2}$ are potentially too large to force the desired change in A below $\gamma(z_n, s_2)$ when they enter V. Recall that we want a number below z_n to enter V in order to force a permanent change in A below $\gamma(z_n, s_2)$, which we can use (since $\gamma(z_n, s_2) < y_n^{s_2}$) as permission to put $y_n^{s_2}$ into B.

We solve these problems with a computable function f which for any e gives an index for a Turing procedure $\varphi_{f(e)}$ which does the following on input x. (The existence of such a function f follows from the Recursion Theorem.) First, it runs our construction until it finds out if $x = z_n$ for some n in a cycle of R_e. If it never finds such a z_n, then $\varphi_{f(e)}(x) \uparrow$. Once it finds $x = z_n$, it watches the construction until it sees a realized follower y_n^s. Again, if it never sees one, then $\varphi_{f(e)}(x) \uparrow$. Once it sees a realized follower, $\varphi_{f(e)}(x)$ converges and outputs 0. (The output is irrelevant; only the fact that it converges matters.) The point of this procedure is that it halts on exactly the prefollowers of R_e which have realized followers. Notice also that if y_n^t enters W_e at stage t, then $\varphi_{f(e)}$ takes at least t steps to halt.

Returning to the scenario of our construction, recall that z_n is our follower and that $y_n^{s_2}$ has just entered W_e at stage s_2. This scenario implies that $\varphi_{f(e)}(z_n)$ halts after at least s_2 many steps. Calculate the stage $t \geq s_2$ such that z_n enters $W_{f(e)}$ at t. Look at each stage $\hat{t}$ between s_2 and $p(t)$ to see if

$$V_{s_2} \upharpoonright z_n \neq V_{\hat{t}} \upharpoonright z_n.$$

If we find such a stage, then we know

$$A_{s_2} | \gamma(z_n, s_2) \neq A_{\hat{t}} | \gamma(z_n, s_2).$$

Furthermore, since $V_{s_2} \upharpoonright z_n \neq V \upharpoonright z_n$ (since V is c.e.), we know that $A_{s_2} \upharpoonright \gamma(z_n, s_2) \neq A \upharpoonright \gamma(z_n, s_2)$ (even though A is Δ_2^0). Therefore, we have permission to put $y_n^{s_2}$ into B and win R_e. If we do not find such a stage $\hat{t}$, then we start the $(n+1)^{st}$ cycle of R_e and initialize everything of lower priority.

The prompt simplicity of V guarantees that $W_{f(e)}$ cannot be infinite, for if so, there would have been a chance to put one of the followers into B. This would imply there were no new prefollowers for R_e, which in turn makes $W_{f(e)}$ finite.

We now present the formal construction and lemmas verifying that the construction succeeds. The priority on our requirements is $R_0 < R_1 < \cdots$ and the construction is finite injury. As above, we assume that $\Gamma^A = V$ and that for every s, $l(s+1) > l(s)$. Let p denote the prompt permitting function for V under this enumeration. At stage 0, set $B_0 = \emptyset$.

At stage s+1, run the current cycle (as described below) for each R_e with $e \leq s$ (in order of their priority) which is not already satisfied. If some R_e ends a cycle and initializes all R_i with $i > e$, then end the stage early. (We initialize R_i by canceling any current prefollowers and followers and setting it at the start of its next cycle.)

Cycle n for R_e: Assume that the cycle starts at stage s. Pick a large prefollower $z_{e,n}$. The cycle takes no more action until the first stage s_1 at which $l(s_1) > z_{e,n}$. At stage s_1 pick a large follower $y_{e,n}^{s_1} > \gamma(z_{e,n}, s_1)$ such that

$y^{s_1}_{e,n} \notin W_{e,s_1}$. As noted above, we use the notation $y^t_{e,n}$ for the current follower of $z_{e,n}$ at stage t.

We say that $y^t_{e,n}$ is *realized* at $t > s_1$ if $y^t_{e,n} \in W_{e,t}$. The current follower $y^{s_1}_{e,n}$ is *canceled* and a new large follower is chosen at t if $\gamma(z_{e,n}, s_1) \neq \gamma(z_{e,n}, t)$ and $y^{s_1}_{e,n}$ has not yet been realized. The cycle takes no more action, except to cancel and pick new followers as necessary, until a stage s_2 when the current follower $y^{s_2}_{e,n}$ is realized.

Suppose $y^{s_2}_{e,n}$ is realized at stage s_2. Find the number $t \geq s_2$ such that $z_{e,n}$ enters $W_{f(e)}$ at t. Calculate $V_{\hat{t}}$ for each $\hat{t}$ such that $s_2 < \hat{t} < p(t)$ and for each such value of $\hat{t}$ check if $V_{s_2} \upharpoonright z_{e,n} = V_{\hat{t}} \upharpoonright z_{e,n}$. If there is a $\hat{t}$ such that $V_{s_2} \upharpoonright z_{e,n} \neq V_{\hat{t}} \upharpoonright z_{e,n}$, then put $y^{s_2}_{e,n}$ into B and declare R_e satisfied. If there is no such $\hat{t}$, then end this stage and initialize all requirements of lower priority. (At the next stage, R_e will begin its $(n+1)^{st}$ cycle.) This ends the description of cycle n for R_e and the description of the formal construction.

Lemma 4.32. *$B \leq_{wtt} A$.*

Proof. By construction, each element in B is a realized follower $y^s_{e,n}$. Suppose $y^s_{e,n}$ is realized at stage s and we enumerate it into B. There must be a number $\hat{t}$ with $s < \hat{t} < p(t)$ (where t is the stage at which $z_{e,n}$ entered $W_{f(e)}$) such that $V_s \upharpoonright z_{e,n} \neq V_{\hat{t}} \upharpoonright z_{e,n}$. Because V is c.e., this inequality implies that $V_s \upharpoonright z_{e,n} \neq V \upharpoonright z_{e,n}$.

We claim that $A_s \upharpoonright y^s_{e,n} \neq A \upharpoonright y^s_{e,n}$ and hence enumerating $y^s_{e,n}$ into B is allowed by our permitting. For a contradiction, suppose that $A_s \upharpoonright y^s_{e,n} = A \upharpoonright y^s_{e,n}$. Since $\gamma(z_{e,n}, s) < y^s_{e,n}$, we have $A_s \upharpoonright \gamma(z_{e,n}, s) = A \upharpoonright \gamma(z_{e,n}, s)$. Because $l(s) > z_{e,n}$, $\Gamma^{A_s}_s \upharpoonright z_{e,n} = \Gamma^A \upharpoonright z_{e,n}$ and hence $V_s \upharpoonright z_{e,n} = V \upharpoonright z_{e,n}$ giving the desired contradiction. □

Lemma 4.33. *Each R_e requirement is won.*

Proof. This proof proceeds as a finite injury argument. Assume that R_e is never initialized by any R_i with $i < e$ after stage s. We need to show that R_e is met (that is, $B \neq \overline{W_e}$) and that R_e only initializes lower priority requirements finitely often.

The requirement R_e only initializes lower priority requirements when it ends a cycle because it found a realized follower with no corresponding change in V. Therefore, if R_e initializes the lower priority requirements infinitely often, then it must have infinitely many realized followers. We make a similar claim if $B = \overline{W_e}$.

Claim 4.34. If $B = \overline{W_e}$, then R_e has infinitely many realized followers.

To prove the claim, assume $B = \overline{W_e}$ and suppose R_e is in cycle n. We have chosen $z_{e,n}$ and when $l(s_1) > z_{e,n}$ we chose a follower $y^{s_1}_{e,n}$. This follower may be canceled, but eventually we get to a stage s_2 with a true use $\gamma(z_{e,n}, s_2)$. After this stage, $y^{s_2}_{e,n}$ will never be canceled. We do not need to worry about $z_{e,n}$ being initialized since nothing of higher priority initializes it and R_e only initiates a new cycle after a realized follower is found.

If $y^{s_2}_{e,n} \notin W_e$, then $B \neq \overline{W_e}$ because we never put $y^{s_2}_{e,n}$ into B. Hence, $y^{s^2}_{e,n} \in W_e$, but since we never get to put this element into B, we know that we eventually move on to the next cycle. The same scenario happens in the $(n+1)^{st}$ cycle: $z_{e,n+1}$ eventually gets a realized follower, but doesn't put it into B and so moves on to the next cycle. In this way it is clear that for every $m > n$, there is a prefollower $z_{e,m}$ which eventually get a realized follower. This completes the proof of the claim.

To finish the proof of the lemma, it suffices to show that R_e cannot have infinitely many realized followers. Assume that each $z_{e,m}$ for $m \geq n$ eventually gets a realized follower. Since each $z_{e,m} \in W_{f(e)}$, $W_{f(e)}$ is infinite. Also, we do not put any of the realized followers into B since doing so would satisfy R_e and cause it to stop initiating new cycles, thereby not having infinitely many realized followers. It follows that there is a sequence of stages $s_n, s_{n+1}, \ldots, s_m, \ldots$ such that

$$z_{e,m} \in W_{f(e)\,\mathrm{at}\,s_m} \text{ but } V_{s_m} \upharpoonright z_{e,m} = V_{p(s_m)} \upharpoonright z_{e,m}$$

for every $m \geq n$. However, since $W_{f(e)} \subseteq \{z_{e,n} | n \in \omega\}$, this condition implies there can be at most finitely many x for which the prompt permitting function works, contradicting the fact that V is promptly simple. □

Bibliography

[1] P. Cholak, R. Downey and R. Walk, Maximal contiguous degrees, *The Journal of Symbolic Logic*, **67**(1), 2002, 409-437.

[2] C.T. Chong and R.G. Downey, Degrees bounding minimal degrees, *Mathematical Proceedings of the Cambridge Philosophical Society*, **105**(2), 1989, 211-222.

[3] S.B. Cooper, Minimal degrees and the jump operator, *The Journal of Symbolic Logic* **38**(2), 1973, 249-271.

[4] B. F. Csima, *Applications of Computability Theory to Prime Models and Differential Geometry*, Ph.D. Dissertation, The University of Chicago, 2003.

[5] R. Downey and N. Greenberg, A transfinite hierarchy of computably enumerable degrees, unifying classes and natural definability, in preparation.

[6] R. Downey, N. Greenberg and R. Weber, Totally $< \omega$ computably enumerable degrees and bounding critical triples, *Journal of Mathematical Logic* **7**(2), 2007, 145-171.

[7] R. Downey and G. LaForte, Presentations of computably enumerable reals, *Theoretical Computer Science* **284**(2), 2002, 539-555.

[8] R. G. Downey and J. B. Remmel, Classification of degree classes associated with r.e. subspaces, *Annals of Pure and Applied Logic* **42**(2), 1989, 105-125.

[9] R. Downey and S. Terwijn, Computably enumerable reals and uniformly presentable ideals, *Dagstuhl Seminar on Computability and Complexity in Analysis 2001*, *Math Logic Quarterly* **48**(1), 2002, 29-40.

[10] P.A. Fejer and R.A. Shore, A direct construction of a minimal recursively enumerable truth table degree, in K. Ambos-Spies, G.H. Muller, and G.E. Sacks eds., *Recursion Theory Week, Proceedings Oberwolfach 1989*, Lecture Note in Mathematics **1432**, Springer Verlag, Berlin, 1990, 187-204.

[11] A. Frohlich and J.C. Shepherdson, Effective procedures in field theory, *Philosophical Transactions of the Royal Society London, Series A* **248**, 1956, 407-432.

[12] C.A. Haught and R.A. Shore, Undecidability and initial segments of the (r.e.) tt-degrees, *The Journal of Symbolic Logic* **55**(3), 1990, 987-1006.

[13] C.A. Haught and R.A. Shore, Undecidability and initial segments of the wtt-degrees $\leq 0'$, in K. Ambos-Spies, G.H. Muller, and G.E. Sacks eds., *Recursion Theory Week, Proceedings Oberwolfach 1989*, Lecture Notes in Mathematics **1432**, Springer Verlag, Berlin, 1990, 223-244.

[14] C. Jockusch Jr., Simple proofs of some theorems on high degrees, *Canadian Journal of Mathematics* **29**(5), 1977, 1072-1080.

[15] A. Kučera, An alternative, priority-free, solution to Post's Problem, in *Mathematical Foundation of Computer Science (Bratislava 1986)*, Lecture Notes in Computer Science **233**, Springer, Berlin, 1986, 493-500.

[16] A. Lachlan, Distributive initial segments of the degrees of unsolvability, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* **14**, 1968, 457-472.

[17] A. Lachlan, Initial segments of the one-one degrees, *Pacific Journal of Mathematics* **29**, 1969, 351-366.

[18] A. Lachlan, Recursively enumerable many-one degrees, *Algebra and Logic* **11**, 1972, 186-202.

[19] A. Lachlan, Embedding nondistributive lattices in the recursively enumerable degrees, in W. Hodges ed., *Conference in Mathematical Logic, London 1970*, Lecture notes in mathematics **255**, Springer-Verlag, New York, 1972, 149-177.

[20] R.E. Ladner and L.P. Sasso, Jr., The weak truth table degrees of recursively enumerable sets, *Annals of Mathematical Logic* **8**(4), 1975, 429-448.

[21] S.S. Marchenkov, The existence of recursively enumerable minimal truth-table degrees, *Algebra and Logic* **14**(4), 1975, 257-261.

[22] R. Miller, Is it harder to factor a polynomial or to find a root?, *Transactions of the American Mathematical Society* **362**(10), 2010, 5261-5281.

[23] A. Nabutovsky and S. Weinberger, The fractal nature of Riem/Diff I, *Geometriae Dedicata* **101**, 2003, 1-54.

[24] A. Nerode, General topology and partial recursive functionals, *Talks from Cornell Summer Institute of Symbolic Logic*, Cornell, 1957, 247-251.

[25] A. Nies, *Computability and Randomness*, Oxford Logic Guides 51, Oxford University Press, Oxford, 2009.

[26] P. Odifreddi, Strong reducibilities, *Bulletin (New Series) of the American Mathematical Society* **4**(1), 1981, 37-86.

[27] P. Odifreddi, *Classical recursion theory: the theory of functions and sets of natural numbers*, North Holland, Amsterdam, 1989.

[28] D.B. Posner, A survey of non-r.e. degrees $\leq 0'$, in F.R. Drake and S.S. Wainer eds., *Recursion Theory: Its Generalisation and Applications (Proceedings Logic Colloquium, University of Leeds, Leeds 1979)*, London Mathematical Society Lecture Note Series **45**, Cambridge University Press, Cambridge, 1980, 52-109.

[29] E.L. Post, Recursively enumerable sets of positive integers and their decision problems, *Bulletin of the American Mathematical Society* **50**, 1944, 284-316.

[30] H. Rogers, *Theory of recursive functions and effective computability*, McGraw-Hill, New York, 1967.

[31] J. Reimann and T. Slaman, Measures and their random reals, *Transactions of the American Mathematical Society* **367**(7), 2015, 5081-5097.

[32] G.E. Sacks, A minimal degree less than $0'$, *Bulletin of the American Mathematical Society* **67**, 1961, 416-419.

[33] G.E. Sacks, The recursively enumerable degrees are dense, *Annals of Mathematics (2)* **80**, 1964, 300-312.

[34] R.I. Soare, *Recursively enumerable sets and degrees*, Springer–Verlag, Heidelberg, 1987.

[35] R. I. Soare, Computability theory and differential geometry, *Bulletin of Symbolic Logic* **10**(4), 2004, 457–486.

[36] C. Spector, On degrees of recursive unsolvability, *Annals of Mathematics (2)*, **64**, 1956, 581-592.

[37] R.M. Steiner, Computable fields and the bounded Turing reduction, *Annals of Pure and Applied Logic* **163**(6), 2012, 730-742.

[38] C.E.M. Yates, Initial segments of the degrees of unsolvability, part II, *The Journal of Symbolic Logic* **35**(2), 1970, 243-266.