On minimal wtt-degrees and computably enumerable Turing degrees

Rod Downey and Reed Solomon

February 17, 2006

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 ($\leq_T$). The motivation for studying reducibilities stronger that Turing reducibility stems from internally motivated questions about varying the access mechanism to the oracle, and the fact that most natural reducibilities arising in classical mathematics tend to be stronger than $\leq_T$. For instance consider the reduction of, say, the word problem to the conjucacy problem in combinatorial group theory. Deeper examples include Downey and Remmel's [3] proof that if V is a enumerable subspace of V_∞, then the degrees of computably enumerable (c.e.) bases of V are precisely the weak truth table (wtt-)degrees below the degree of V. Similarly, wtt-reducibility proved fundamental in the work on differential geometry Nabutovsky and Weinberger [13], as studied by Csima [2] and Soare [21]. 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. There are innumerable examples of this phenomenon and a good example is the first paper of Ladner and Sasso [11] in which they construct locally distributive parts of the c.e. T-degrees using the wtt-degrees and their interactions with the T-degrees. For general information concerning these reducibilities, we refer the reader to the survey article by Odifreddi [14] as well as the books by Rogers [17], Odifreddi [15] and Soare [20].

The concern of this paper is the interaction of minimality and enumerability, two of the basic objects of classical computability. All constructions of minimal degrees are basically effective forcing arguments of one kind or another and such constructions are relatively incompatible with the construction of effective objects. In particular, by 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 and of minimal tt-degree. Since wtt-reducibility is intermediate between $\leq_{tt}$ and $\leq_T$, it is natural to wonder what happens here. Again, Sacks Splitting Theorem shows that no wtt-degree of a c.e. *set* can have minimal wtt-degree, but this leaves open the intriguing possibility that a minimal wtt-degree might have c.e. T-degree. This question served as the primary motivation for this paper. Before we present our answers, we discuss the history and motivation in more detail.

It is surely a basic question in any degree structure whether minimal degrees exist. 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 [22] 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 [7] 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 [18] 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 or below any high T-degree. This technique also uses partial trees and hence does not automatically produce minimal wtt or tt-degrees.

The other studied theme for the present paper is that of enumerability, and hence 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 ([8]) and a c.e. minimal m-degree ([9]). (That is, there is a set A with minimal 1-degree such that $A \equiv_1 W_e$ for some c.e. set W_e. Of course, in the 1-degrees and the m-degrees, the property of being c.e. is closed downwards. Therefore, to build such minimal degrees, it suffices to make them minimal within the c.e. 1-degrees or in the c.e. m-degrees.) Marchenkov [12] proved that c.e. minimal tt-degrees exist, although the first direct construction of such a degree was given by Fejer and Shore [4].

As we 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 [19] proved that the c.e. T-degrees are dense and Ladner and Sasso [11] proved that the c.e. wtt-degrees are dense, so there are no c.e. minimal T or wtt-degrees. Thus, Turing and weak truth table reducibility differ from the stronger reducibilities with respect to the existence of c.e. minimal degrees. However, it is possible to get some positive results concerning the relationship between minimal T-degrees and c.e. T-degrees. For example, Yates [23] 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.

In this paper, we look at Yates' Theorem from a different perspective. Instead of looking at whether noncomputable c.e. degrees bound minimal degrees, we look at whether minimal degrees can bound noncomputable c.e. degrees or can even be of c.e. degree. Obviously, 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 A Turing bound a noncomputable c.e. set? Can A have c.e. T-degree? The main theorem of this paper gives a positive answer to the first question.

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

We feel that the proof of this theorem is also 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 reduction somewhat resemble the methods used by Lachlan [10] imbedding nondistributive lattice in the c.e. degrees. Such techniques have hitherto never been used in the full approximation construction, which is why we will only slowly work up to the details. The majority of this paper is concerned with the proof of Theorem 1.1: in Section 4, we give an informal sketch of the proof and in Section 5, we present the formal construction.

Before presenting the proof of Theorem 1.1, we prove two results giving limitations on possible extensions of Theorem 1.1. In particular, we consider whether a Δ^0_2 set with minimal wtt-degree can have c.e. T-degree and whether a Δ^0_2 set with minimal wtt-tree can Turing bound a noncomputable c.e. set which is "close to" $0'$ in some sense. While these limitations could be stated for wtt-minimality, the proofs yield slightly stronger results using a different notion of minimality.

Definition 1.2. A noncomputable set A is **wtt-minimal over the Turing degrees** if for any $C \leq_{wtt} A$, either C is computable or $C \equiv_T A$.

The notion of being wtt-minimal over the Turing degrees is more general than the notion of being wtt-minimal (in the sense that every wtt-minimal set is wtt-minimal over the Turing degrees) while not implying that the set is T-minimal. In Section 2, we show that we cannot extend Theorem 1.1 by making A and B have the same Turing degree.

Theorem 1.3. *If A is a noncomputable Δ^0_2 set with c.e. Turing degree, then A is not wtt-minimal over the Turing degrees (and hence is not wtt-minimal).*

In Section 3, we show that the set B in Theorem 1.1 cannot be promptly simple and hence cannot be "close" to $\mathbf{0}'$ in this sense.

Theorem 1.4. *Let V be a promptly simple c.e. set and let A be a Δ_2^0 set such that $A \geq_T V$. There exists a c.e. set B such that $0 <_T B \leq_{wtt} A$.*

Because the c.e. *wtt*-degrees are dense, the set A in the statement of Theorem 1.4 cannot have minimal *wtt*-degree. However, we can also show that A cannot be *wtt*-minimal over the Turing degrees. By Theorem 1.3, if $A \equiv_T B$ in Theorem 1.4, then A is not *wtt*-minimal over the Turing degrees. On the other hand, if $B <_T A$, then B is both *wtt* and T below A and hence A is not *wtt*-minimal over the Turing degrees. Therefore, we have the following corollary.

Corollary 1.5. *If A Turing bounds a promptly simple c.e. set, then A is not wtt-minimal over the Turing degrees.*

Most of our terminology is standard and follows Soare [20]. To distinguish between T and *wtt*-reducibilities, we use φ_e for the e^{th} Turing reduction and $[e]$ for the e^{th} weak truth table reduction. The proof of Theorem 1.1 uses a full approximation argument for which Posner [16] provides an excellent introduction. The proof of Theorem 1.4 relies on basic results about promptly simple sets which can be found in Chapter XIII of Soare [20].

2 Proof of Theorem 1.3

In this section, we give the proof of Theorem 1.3. For convenience, we restate it here.

Theorem 2.1. *If A is a noncomputable Δ_2^0 set with c.e. Turing degree, then A is not wtt-minimal over the Turing degrees.*

Proof. Let A be a noncomputable Δ_2^0 set with c.e. Turing degree. Fix a c.e. set B such that $A \equiv_T B$ and fix Turing reductions Φ and Δ such that $\Phi^A = B$ and $\Delta^B = A$. For any stage s and any number y, if $\Delta_s^{B_s}(y)\downarrow$, then let $\delta(y,s)$ denote the use of this computation. (Similarly, let $\phi(x,s)$ denote the use of $\Phi_s^{A_s}(x)$.) Because Φ and Δ are total, $\delta(x,s)$ and $\phi(x,s)$ reach limits for each x as $s \to \infty$.

To prove that A is not *wtt*-minimal over the Turing degrees, we construct a noncomputable Δ_2^0 set E such that $E \leq_{wtt} A$ and $A \not\leq_T E$. We have to meet the following requirements. First, to make E noncomputable, we satisfy

$$P_e : \varphi_e \neq E$$

for each $e \in \omega$. (By this, we mean that φ_e does not compute the characteristic function of E.) Second, we make $E \leq_{wtt} A$ using permitting with a fixed computable bound. More specifically, for all s and k, we guarantee that

$$A_s|k = A|k \Rightarrow E_s|k = E|k.$$

Third, to make $A \not\leq_T E$, we satisfy

$$N_\Gamma : \Gamma^E \neq A$$

where Γ ranges over all Turing reductions.

We use the following length of agreement function between the fixed Turing reductions Φ and Δ.

$$l(s) = \max\{x \mid \forall y \leq x\,(\Delta_s^{B_s}(y) \downarrow = A_s(y) \wedge \forall q \leq \delta(y,s)\,[\Phi_s^{A_s}(q) \downarrow = B_s(q)\,]\,)\}$$

Since $\Phi^A = B$ and $\Delta^B = A$, this length of agreement function will approach ∞ in the limit. A stage s is called *expansionary* if $l(s) > l(t)$ for all $t < s$. By speeding up the approximations to A and B, we can assume that every stage of our construction is expansionary in this sense. (We will have different length of agreement functions $l(\Gamma, s)$ for the N_Γ requirements and the stages of the construction will not all be expansionary with respect to the $l(\Gamma, s)$ functions.)

Our strategies to meet P_e and N_Γ are both finitary so the argument is a finite priority requirement. As usual for these types of requirements, the P_e requirements attempt to put a single diagonalizing element into E while the N_Γ strategies attempt to restrain E from changing on certain uses. These strategies interact in a standard way. Each time a P_e strategy places a number into E, it injures all lower priority N_Γ requirements, causing them to lose their current restraint and begin again. Each time an N_Γ strategy establishes a restraint, it injures all lower priority P_e requirements, causing them to work with numbers above this restraint. Because the strategies are all finitary, each strategy eventually becomes the highest priority strategy not yet satisfied and hence acts as though it is the highest priority strategy. Therefore, we describe the action of each strategy, make clear why it is finitary and eventually succeeds in isolation, and leave the formal details of putting these strategies together in a finite injury argument to the reader.

We begin with the strategy for P_e. The strategy proceeds in cycles and each cycle potentially computes an initial segment of A. The main point is that if we go through infinitely many cycles, then we have a computation procedure for A, contrary to our assumption that A is noncomputable. Therefore, the strategy for P_e will go through only finitely many cycles and will be finitary. When a P_e strategy is initialized, it cancels all of its witnesses and followers, ceases all action for its current cycles and begins again with its first cycle as though it had never acted.

Step 1 for cycle n. Pick a large witness z_n and wait for $l(s) > z_n$. This step is finitary since $l(s) \to \infty$ in the limit.

Step 2. Assume that $l(s) > z_n$. Pick a large follower x_n^s for z_n and begin to wait for $\varphi_e(x_n^s)$ to converge to 0. (We continue with the construction while waiting for this convergence. If we see $\varphi_e(x_n^s)$ converge to 0 at a later stage, we say the follower x_n^s is *realized*.) By our definition of $l(s)$, we have that $\Delta_s^{B_s}$ computes the initial segment of A_s up to z_n and that $\Phi_s^{A_s}$ computes the initial segment of B_s needed for the use of these $\Delta_s^{B_s}$ computations. Therefore, the composition of use functions $\phi(\delta(z_n, s), s)$ is defined at stage s. The important feature of the follower x_n^s is that it is chosen large, so $x_n^s > \phi(\delta(z_n, s), s)$. If there is a change in A or B at stage $t > s$ (while still waiting for x_n^s to be realized) that causes $\phi(\delta(z_n, t), t) \neq \phi(\delta(z_n, s), s)$, we cancel the follower x_n^s, rechoose x_n^t to be large so that $x_n^t > \phi(\delta(z_n, t), t)$, and switch to waiting for x_n^t to be realized (that is, for $\varphi_e(x_n^t)$ to converge to 0).

Because $\Phi^A = B$ and $\Delta^B = A$, we must either eventually see some follower x_n^s that is realized or settle on a final follower x_n^s which is never realized. In the former case, we proceed to Step 3. In the latter case, we have $\varphi_e(x_n^s) \neq 0$ (and it might not even converge) and we never put x_n^s into E, so we win P_e. Therefore, if we get stuck forever at Step 2 of a cycle for P_e, we win P_e. Hence, assume that we reach a stage t such that x_n^t is realized. (We do not assume that $\phi(\delta(z_n, t), t)$ has reached its limit; it may continue to change while we are in Step 3.)

Step 3. Assume that x_n^t is realized and let $c_n = \phi(\delta(z_n, t), t)$. We perform the following three actions. First, declare that our attempt to compute A says $A|z_n = A_t|z_n$. Second, start the $(n+1)^{\text{st}}$ cycle for P_e. Third, wait for a stage $u > t$ such that $A_u|c_n \neq A_t|c_n$.

Notice two things about this step. We cannot have $u > t$ with $\phi(\delta(z_n, u), u) \neq \phi(\delta(z_n, t), t)$ without $A_u|c_n \neq A_t|c_n$. Also, if we never see such a stage u, then our guess that $A|z_n = A_t|z_n$ is correct. Therefore, if we get stuck forever in Step 3 of cycle n, then we have correctly computed A up to z_n.

Step 4. When we see $u > t$ such that $A_u|c_n \neq A_t|c_n$, we put x_n^t into E. Because $c_n = \phi(\delta(z_n, t), t)$ and $\phi(\delta(z_n, t), t) < x_n^t$, we have permission (at least temporarily) to put x_n^t into E. There are two cases to consider here.

Step 4, Case 1. There is a number $k < \delta(z_n, t)$ enumerated into B at stage u. (Notice that if $A_t|z_n \neq A_u|z_n$, then we must be in this case.) In this case, we know that there must eventually be a permanent change in A below c_n (so also below x_n^t). Hence, although A may make many future changes below c_n, it can never return to $A_t|c_n$ because B is computably enumerable and so the change in B below $\delta(z_n, t)$ is permanent. Therefore, we can put x_n^t permanently into E and win P_e. We quit all other work on P_e and declare it satisfied.

Step 4, Case 2. There is no change in B up to $\delta(z_n, t)$ (and hence no change in A up to z_n). In this case, we still put x_n^t into E (temporarily winning P_e), but we may have to take x_n^t back out of E if at some future stage A reverts back to A_t up to c_n. (That is, if there is no change in B up to c_n, then the opponent is free to make $A_v|c_n = A_t|c_n$ at a later stage $v > u$.) Should the opponent do this, we lose our permission to keep x_n^t in E and must take it out. In this case, we return to Step 3 and wait for the next stage u at which $A_u|c_n \neq A_t|c_n$.

To emphasize the finitary nature of P_e, we summarize what can happen. Every cycle started for P_e proceeds from Step 1 to Step 2. If any cycle waits forever at Step 2, we win because the final follower x_n^s is not realized and is never put into E. Furthermore, no new cycles are started while one is waiting at Step 2. Therefore, assume that all cycles started for P_e move from Step 2 to Step 3. Each time a cycle moves to Step 3, we start a new cycle with a large value for z_n. If any cycle reaches Case 1 of Step 4, then we win P_e as described in that case and cease our action for P_e. Therefore, assume that no cycle ever reaches Case 1 of Step 4. In this case, we never stop the action of P_e so we start cycle n for each $n \in \omega$. Each of these cycles reaches Step 3 and may go on to Case 2 of Step 4. Furthermore, the n^{th} cycle may bounce back and forth between Step 3 and Case 2 of Step 4 many times as A changes below c_n. However, since the approximation to A must settle down up to c_n, the n^{th} cycle eventually settles into either waiting forever at Step 3 or waiting forever at Case 2 of

Step 4. In either case, we have that $A|z_n = A_t|z_n$ (where t is the stage at which the n^{th} cycle saw x_n^t realized). Hence our guesses at the initial segments of A are all correct and we have a computation procedure for the noncomputable set A. Since this cannot happen, we must have that either some cycle gets stuck forever at Step 2 (and we do not begin any further cycles and take no further action, other than waiting, for P_e) or some cycle eventually is in Case 1 of Step 4 (in which case we stop the action of P_e because we know x_n^t is permanently allowed in E). In either case, the action of P_e is finitary.

Next, we describe the action of N_Γ, where Γ is a Turing reduction. If $\Gamma_s^{E_s}(x)$ converges, we let $\gamma(x, s)$ denote to use of this computation. We work with the following length of agreement function and maximum use function.

$$l(\Gamma, s) = \max\{x \mid \forall y \leq x\,(\Gamma_s^{E_s}(y) \downarrow = A_s(y))\}$$
$$u(\Gamma, s) = \max\{\gamma(y, s) \mid y \leq l(\Gamma, s)\}$$

We say that a stage s is Γ-expansionary if $l(\Gamma, s) > l(\Gamma, t)$ for all $t < s$. Notice that $\Gamma^E = A$ if and only if $l(\Gamma, s)$ goes to infinity in the limit.

The action of N_Γ also proceeds in cycles for each $n \in \omega$. Each cycle will potentially compute an initial segment of B such that if infinitely many cycles establish a restraint, we have a computation procedure for the noncomputable set B. Because of this contradiction, the restraint imposed by N_Γ will be finitary.

Step 1 for cycle n. Wait for a Γ-expansionary stage s such that $l(\Gamma, s) > \phi(\delta(n, s), s)$. Because $\phi(\delta(n, s), s)$ eventually reaches a limit, if we wait at Step 1 forever then $l(\Gamma, s)$ is bounded and we win N_Γ. Therefore, assume that we eventually see such a Γ-expansionary stage. (We do not assume that $\phi(\delta(n, s), s)$ has reached its limit at this stage. These uses can still change in Step 2 below.)

Step 2. When we see $l(\Gamma, s) > \phi(\delta(n, s), s)$, we act as follows. Fix the stage s_n at which we first see this inequality. Declare our attempted computation of B to say $B|\delta(n, s_n) = B_{s_n}|\delta(n, s_n)$. Start the $(n+1)^{\text{st}}$ cycle for N_Γ. Restrain E from changing below $u(\Gamma, s_n)$.

Consider what happens if our guess that $B|\delta(n, s_n) = B_{s_n}|\delta(n, s_n)$ is incorrect. In this case, some number $q < \delta(n, s_n)$ must enter B after stage s_n. This entry means that there is eventually a permanent change in A below $\phi(\delta(n, s_n), s_n)$. Because $l(\Gamma, s_n) > \phi(\delta(n, s_n), s_n)$ and we restrained E from changing below $u(\Gamma, s_n)$, we have that Γ^E computes $A_{s_n}|\phi(\delta(n, s_n), s_n)$ which is not the same as $A|\phi(\delta(n, s_n), s_n)$. Therefore, if our guess at B is incorrect, we win N_Γ. Furthermore, we can recognize this win since it involves a number entering B and this number must enter permanently since B is computably enumerable. Therefore, if we ever see a number $q < \delta(n, s_n)$ enter B, we stop N_Γ from imposing more restraint and we declare N_Γ satisfied.

On the other hand, if all of our cycles move past Step 1 and all of our guesses at initial segments of B are correct, then we have a computation procedure for the noncomputable set B. Because this cannot happen, we must either have some cycle which waits forever at Step 1 (in which case we start no further cycles and impose a finite restraint) or we eventually see a number enter B below $\delta(n, s)$ for one of our cycles in Step 2 (in which case we stop N_Γ

from imposing further restraint and declare N_Γ satisfied). In either case, we win N_Γ while imposing only finite restraint on E. □

3 Proof of Theorem 1.4

In this section, we prove Theorem 1.4. For convenience, we restate it here. (We refer the reader to Soare [20] for information on promptly simple sets and degrees. Below, we state the property of promptly simple sets which we will use in the construction.)

Theorem 3.1. *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

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

satisfies $l(s+1) > l(s)$ for all s. (That is, we assume that every stage of our construction is expansionary.) 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 [20] Chapter XIII, Theorem 1.7):

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

$W_{e \,\mathrm{at}\, s}$ means that $x \in W_{e,s}$ and $x \notin W_{e,s-1}$. For $x \leq l(s)$, we use $\gamma(x,s)$ to denote the use of $\Gamma^{A_s}_s(x)$.

To make B noncomputable, we meet the requirement R_e that $B \neq \overline{W_e}$ 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 guarantee that

$$A_s|x = A|x \Rightarrow B_s|x = B|x.$$

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 extra 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 $y^{s_1}_n$ such that $y^{s_1}_n > \gamma(z_n, s_1)$. Notice that if there is a change in $V_{s_1}|z_n$, then there must be a corresponding change in $A_{s_1}|\gamma(z_n, s_1)$, which we would like to use as a permission to put $y^{s_1}_n$ into B.

We say $y^{s_1}_n$ is *realized* at $t \geq s_1$ if $y^{s_1}_n \in W_{e,t}$. We say that $y^{s_1}_n$ is *canceled* at stage $t > s_1$ if $\gamma(z_n, t) \neq \gamma(z_n, s_1)$ and $y^{s_1}_n$ has not yet been realized. If $y^{s_1}_n$ is canceled at stage t, then we pick a new follower $y_n^t > \gamma(z_n, t)$. Notice that 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 $\gamma(z_n, s_2)$ (which is $< y_n^{s_2}$ and so would give us permission to put $y_n^{s_2}$ into B). We want to force a number below z_n into V in order to cause a change in A below $y_n^{s_2}$.

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. 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

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

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 $V_{s_2}|z_n \neq V|z_n$ (recall that V is c.e.), we know that $A_{s_2}|\gamma(z_n, s_2) \neq A|\gamma(z_n, s_2)$ (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 $y^{s_1}_{e,n} > \gamma(z_{e,n}, s_1)$. 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 $V_{s_2}|z_{e,n} = V_{\hat{t}}|z_{e,n}$. If there is a $\hat{t}$ such that $V_{s_2}|z_{e,n} \neq V_{\hat{t}}|z_{e,n}$, 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 3.2. *$B \leq_{wtt} A$.*

Proof. 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 $V_s|z_{e,n} \neq V_{\hat{t}}|z_{e,n}$. Because V is c.e., this inequality implies that $V_s|z_{e,n} \neq V|z_{e,n}$.

We claim that $A_s|y^s_{e,n} \neq A|y^s_{e,n}$ and hence enumerating $y^s_{e,n}$ into B is allowed by our permitting. For a contradiction, suppose that $A_s|y^s_{e,n} = A|y^s_{e,n}$. Since $\gamma(z_{e,n}, s) < y^s_{e,n}$, we have $A_s|\gamma(z_{e,n}, s) = A|\gamma(z_{e,n}, s)$. Because $l(s) > z_{e,n}$, $\Gamma^{A_s}_s|z_{e,n} = \Gamma^A|z_{e,n}$ and hence $V_s|z_{e,n} = V|z_{e,n}$. This contradicts the condition on V in the last sentence of the previous paragraph. □

Lemma 3.3. *Each R_e requirement is won.*

Proof. This proof proceeds as a finite injury argument. Assume that at stage s, requirement R_e has priority. That is, assume that R_e is never initialized by any R_i with $i < e$ after stage s. For a contradiction, assume that $B = \overline{W_e}$.

Claim. R_e has infinitely many realized followers.

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, $y_{e,n}^{s_2}$ 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 $y_{e,n}^{s_2} \notin W_e$, then $B \neq \overline{W_e}$ because we never put $y_{e,n}^{s_2}$ into B. Hence, $y_{e,n}^{s^2} \in W_e$, 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.

Since each $z_{e,m}$ for $m \geq n$ eventually gets a realized follower, we have that $z_{e,m} \in W_{f(e)}$ and so $W_{f(e)}$ is infinite. Also, since we did not put any of the followers into B, there is a sequence of stages $s_n, s_{n+1}, \ldots, s_m, \ldots$ such that

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

However, since $W_{f(e)} \subseteq \{z_{e,n} | n \in \omega\}$, there can be at most finitely many x for which the prompt permitting function works. This violates the fact that V is promptly simple. $\square$

4 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. 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. For uniformity of presentation (that is, to be able to treat λ like any other string), we regard λ' and λ'' are distinct symbols. 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. (We will be more precise about this definition in the formal construction.)

Theorem 4.1. *There is a Δ_2^0 set A and a noncomputable c.e. set B such that A is wtt-minimal and $B \leq_T A$.*

To make A be *wtt*-minimal, 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. (We assume the reader is familiar with full approximation arguments. Posner [16] is an excellent introduction

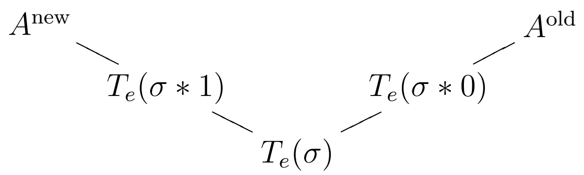

Figure 1: When the current path moves from $T_e(\sigma * 0)$ to $T_e(\sigma * 1)$, we challenge R_e to verify that it converges on all elements of $X_e = \{x \mid [e]_s^\tau(x) \text{ converges for some } \tau \supseteq T_e(\sigma * 0)\}$ using oracles along the new current path A^{new}.

to these arguments.) 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* state and a node which does not $[e]$-split is said to be in the *low* state. In addition, we define a current path A_s which represents our stage s approximation to A. (Technically, we define A_s at the beginning of stage s and then allow strategies which act during stage s to change this path. 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.)

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 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 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.) 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 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

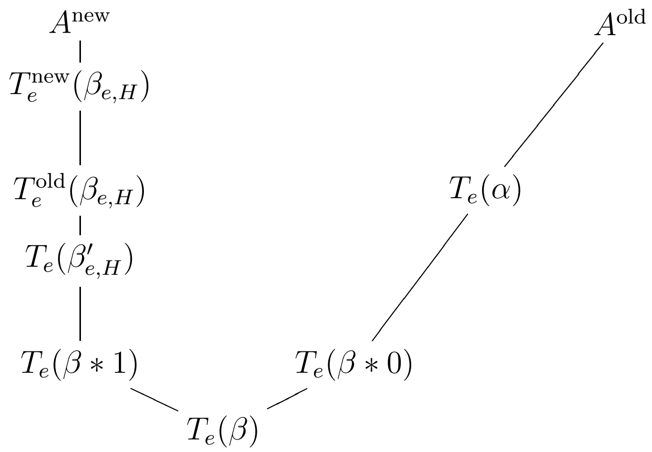

Figure 2: If $T_e(\alpha)$ is high splitting and the current path moves from $T_e(\beta * 0)$ to $T_e(\beta * 1)$, then we stretch $T_e^{\text{old}}(\beta_{e,H})$ to have value $T_e^{\text{new}}(\beta_{e,H})$ such that $|T_e^{\text{new}}(\beta_{e,H})| > |T_e(\alpha)| > m$.

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 3.) This means that we no longer look at 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

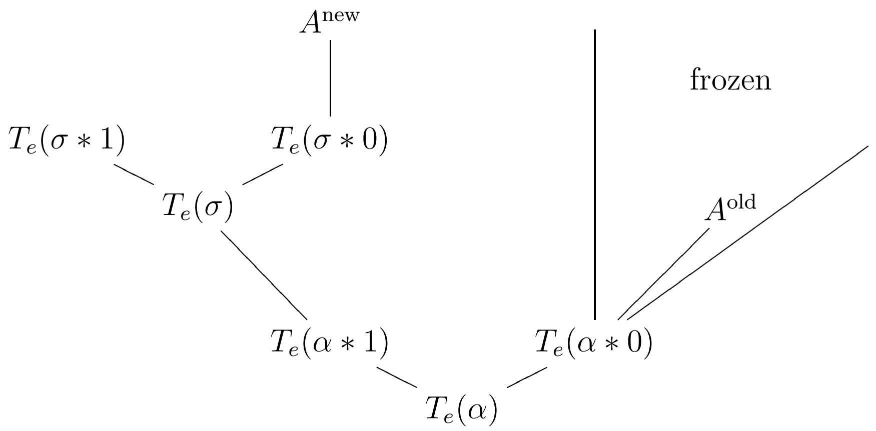

Figure 3: If $T_e(\alpha)$ is in the low state and we move the current path from $T_e(\alpha * 0)$ to $T_e(\alpha * 1)$ for the sake of P_e, then we freeze the cone above $T_e(\alpha * 0)$ until we have seen identical computations on all the elements of X_e using oracles along the new current path A^{new}. 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.

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 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 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 (that is, longer length than any number or string considered in the construction so far) such that δ is on all of these trees and is on the new current path.

We redefine these trees by stretching. (See Figure 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 $T_{\mu_1,s}(\alpha) = \hat{T}_{\mu_2}$ (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

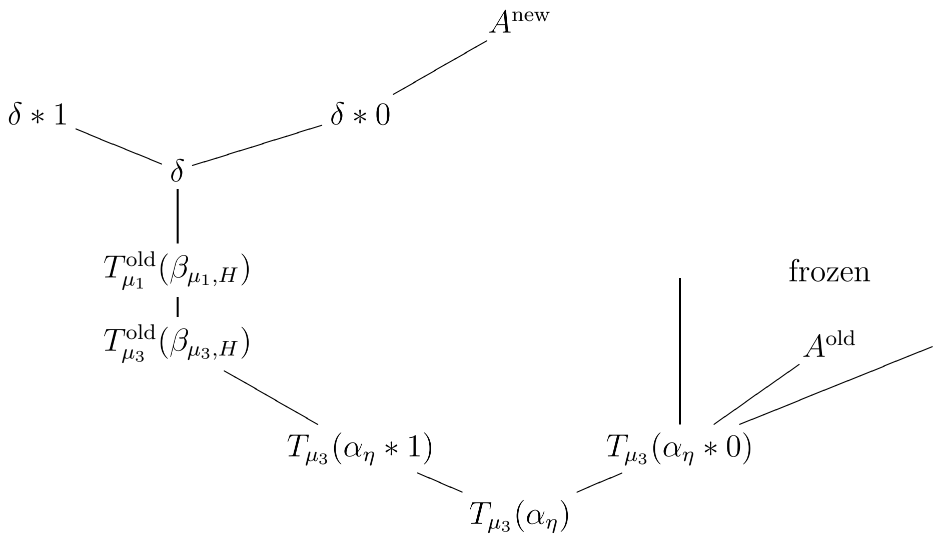

Figure 4: When we move the current path from $T_{\mu_3}(\alpha_\eta * 0)$ to $T_{\mu_3}(\alpha_\eta * 1)$ for the sake of the P strategy η, we freeze the cone above $T_{\mu_3}(\alpha_\eta * 0)$ and stretch the trees T_{μ_i}, $0 \leq i \leq 3$. In this figure, δ is equal to $T^{\text{new}}_{\mu_1}(\beta_{\mu_1,H})$, $T^{\text{new}}_{\mu_2}(\beta)$, $T^{\text{new}}_{\mu_3}(\beta_{\mu_3,H})$ and $T_{\mu_0}(\sigma_1)$.

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 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 of the numbers 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 $= 0$. η picks a large number k (larger than any number or length of string used in the construction so far) 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 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 is 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

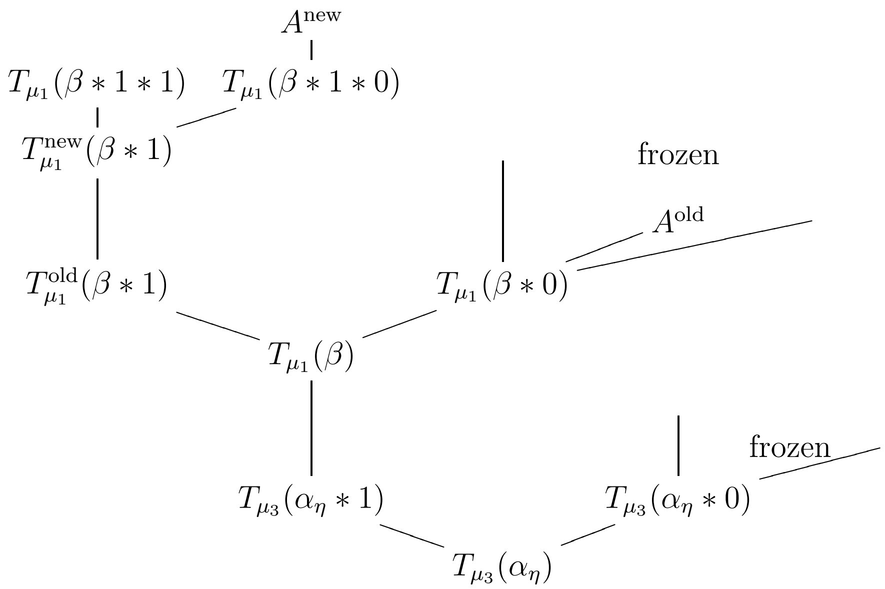

Figure 5: This figure represents our actions at stage s_1 when μ_1 finds a potential high split using nodes τ_0 and τ_1 extending $T_{\mu_1}(\beta_{\mu_1,H} * 0)$. For ease of notation, we have used β in place of $\beta_{\mu_1,H}$.

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 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 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 $\hat{T}_{\mu_1} = T_{\mu_1,s}$. 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 $T_{\mu_1,s}(\beta_{\mu_1,H} * 1)$ 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.

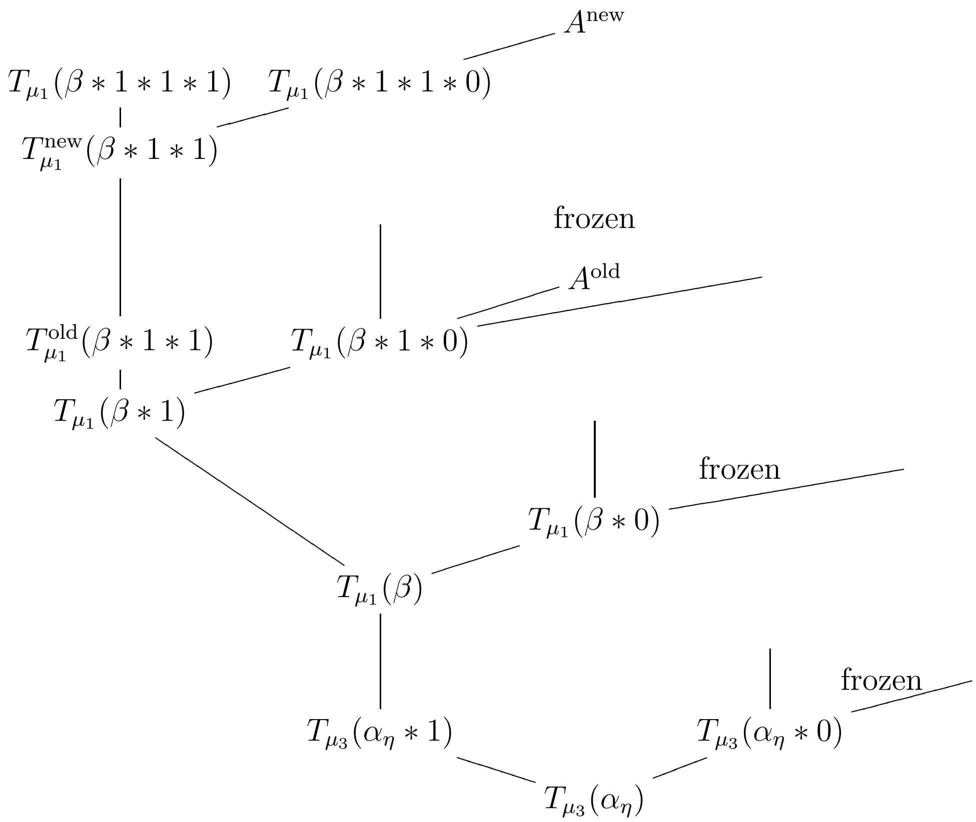

Figure 6: This figure represents our action at stage s_2 when μ_1 begins the process to forbid the cone above $T_{\mu_1}(\beta_{\mu_1,H} * 1 * 0)$ to eliminate the Γ definition using this node as the oracle. For ease of notation, we have used β in place of $\beta_{\mu_1,H}$.

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 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 $\mu_1 * L$.

Consider 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.

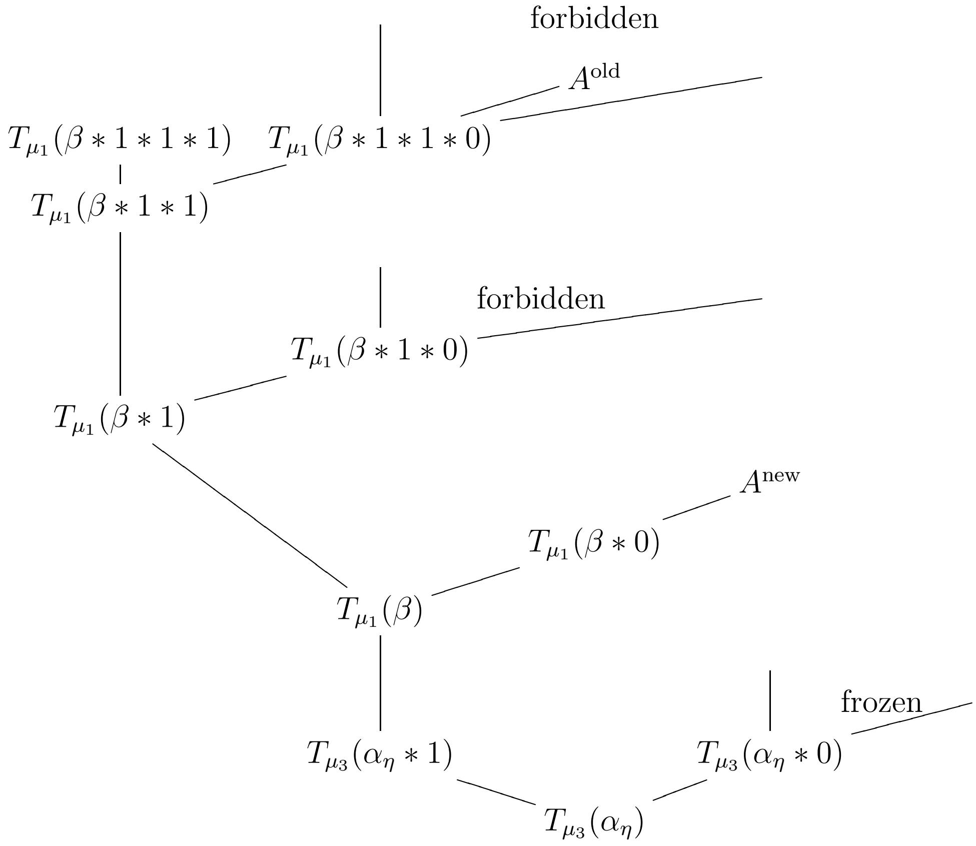

Figure 7: This figure represents the situation at stage s_3 when μ_1 returns the current path to $T_{\mu_1}(\beta_{\mu_1,H} * 0)$ and meets its high challenge by putting $T_{\mu_1}(\beta_{\mu_1,H})$ into the high μ_1 state. For ease of notation, we have used β in place of $\beta_{\mu_1,H}$.

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 $\mu_1 * N$.

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 $T_{\mu_1,s_2}(\beta_{\mu_1,H} * 1 * 0)$.

μ_1 acts as though it were a P strategy with only one low R strategy of higher priority. (See

Figure 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 7.) μ_1 has met its high challenge and takes outcome μ_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 μ_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. 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.

5 Formal construction for Theorem 1.1

Before giving the formal construction, we list 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'' as a special symbol distinct from λ and set $T_{\lambda'',s}$ to be an identity tree for all s.

In the tree of strategies, 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. $A_{\eta,s}$ 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.

$p_\eta \in \mathbb{N}$ is the level on the η'' tree at which we start building T_η. That is, 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 $\alpha_\eta = \alpha$ 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 is at stage s is a total function from $2^{<\omega}$ to $2^{<\omega}$. Therefore, in some sense we use all the elements of ω at each stage s! However, 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. One way to do this is say to limit our trees $T_{\eta,s}$ to being finite functions from strings of length $\leq s$ to $2^{<\omega}$. However, it seems more natural to view the trees as total functions. 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 such 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 $\Gamma^{\emptyset}(x) = 0$. 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 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 occured, then η must be in Subcase A.

Subcase A: 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 $T_{\eta,s}(\gamma) = T_{\eta,t}(\beta_{\eta,H}) \subseteq \tau_0, \tau_1$, $U(\tau_0) = U(\tau_1) = G_\eta$, $\exists w_\eta([\eta]^{\tau_0}_s(w_\eta) \downarrow \neq [\eta]^{\tau_1}_s(w_\eta) \downarrow)$ 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 A. (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 A(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 A(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 B below.

Subcase B. At the previous η stage, η acted in Subcase A(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 A(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 A(ii)). This action unfreezes the strings τ_0 and τ_1 from Subcase A(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 A(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 B 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, Cases 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 Case 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 A(ii) or B. 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 A(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 B 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 A(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 5.1. *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 $\hat{\nu} \neq \hat{\eta}$.

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. $\square$

A case analysis similar to the one for Lemma 5.1 proves the following lemma.

Lemma 5.2. *If μ is challenged high, then there cannot be a link jumping over μ and landing above $\mu * H$.*

Lemma 5.3. *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 5.1. $\square$

Lemma 5.4. *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 5.3, 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. Therefore μ can only be challenged low by one strategy at a time. $\square$

Essentially the same proofs as for Lemmas 5.3 and 5.4 establish the following two lemmas.

Lemma 5.5. *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 5.6. *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 A 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 A 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 A(i) or A(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 B 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 A of the high challenge and has not yet found a potential high split. Therefore, in our construction, we give all the necessary instructions for handling a strategy which is both high and low challenged.

Lemma 5.7. *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 jumping links over η at the end of stage s.

If η is an R strategy, then η must be acting in Subcase B 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 5.1, 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 5.8. *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 A(ii) or Subcase B (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 A(ii) or Subcase B 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 5.9. *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 5.8, 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 A(ii) or B of the high challenge (including the verification procedure called by Subcase B). 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 5.7. Because a high challenged strategy in this sequence of high challenges only moves the current path when it issues new high challenges in Subcase A(ii) or B 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 5.5, 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 A(ii) or B of the high challenge contrary to our assumption. ☐

Lemma 5.10. *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 5.9, 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 5.9, 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 5.9 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 any computation viewed in the construction or some parameter has been defined which is larger than $T_{\eta,s}(\alpha)$. We will only apply Lemmas 5.11 and 5.12 in situations in which α is equal to some parameter in the construction such as α_η or $\beta_{\eta,H}$.

Lemma 5.11. *Let η be an R strategy, s be an η stage and α as 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. Because 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 5.9.

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 5.8, 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 5.7 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 5.9, 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 5.5 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 A(ii) or B 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 5.5. 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 5.12. *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 5.11. □

Lemma 5.13. *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 5.12 to this node to show that it cannot change after stage t unless η is initialized. By Lemma 5.12, 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 5.12, 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 5.5 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. $\square$

Lemma 5.14. *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 5.12 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 5.12 since we don't know what the states of $T_{\eta',t}(\alpha_\eta * i)$ are. However, we claim that we can use Lemma 5.11. 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 5.11 (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 5.11. 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 5.11 in place of Lemma 5.12. $\square$

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 5.15. *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 5.16. *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 A 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 A(i) and wins the high challenge.*

(c) *There is an η stage $s_1 > s_0$ at which η acts in Subcase A(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 A(ii) or in a verification procedure called in Subcase B are forbidden.*

We prove Lemmas 5.15 and 5.16 simultaneously by induction on the length of η. We begin with Lemma 5.16. 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 5.16 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 5.16 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 5.11 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 5.16. 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 5.16, 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 5.11 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 A 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 A(i) or 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 A(i), then the high challenge is met and we clearly meet the conditions of Property 4 of Lemma 5.16. This establishes Property 3(b).

Consider what happens if η acts in Subcase A(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 5.16 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 B. 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 B.

η acts in Subcase B 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 5.15 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 5.15 holds. We do this below without assuming anything except the properties just listed. Given that Lemma 5.15 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 5.15) or made by the action of η in Subcase A(ii). The only definition made in Subcase A(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 B when the verification procedure ends. The conditions of Property 4 are met and we have completed the proof of Lemma 5.16.

Consider Lemma 5.15. 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 B of a high challenge and when the verification procedure is called by a P strategy. If η is an R strategy acting in Subcase B, 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 5.14, $T_{\eta',t_0}(\alpha_\eta * 0) = T_{\eta',t_0}(\sigma_0 * 0)$ 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.

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 5.16, $U(T_{\nu,u}(\beta_{\nu,H})) = G_\nu * H$ 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 high 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 5.11), 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 5.15 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 5.15.

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 A(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 5.15, 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 5.17. *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 5.11 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.) $\square$

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 A(ii) of a high challenge.

Lemma 5.18. *Assume η is challenged low at stage s by a strategy $\hat{\eta}$ which is acting in Subcase A(ii) of a high challenge. Then $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$.*

Lemma 5.19. *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 5.17 and 5.18, $U(T_{\eta,s}(\beta_{\eta,L})) = G_\eta * L$. 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. $\square$

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 5.20. *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 5.7). 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 5.20 inductively to ν and using the fact that ν is not initialized after stage s, we conclude from Lemma 5.15 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 η is on the true path. 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 A(ii) or B 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 A(ii) (and later in Subcase B) 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 A(ii) after stage s. ν initializes η and ends the stage. Applying Property 2 of Lemma 5.20 inductively to ν and using the fact that ν is not initialized after s, we conclude from Lemma 5.16 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 B 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 5.20 together with Property 2 in Lemma 5.19 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 5.20 together with Lemma 5.16 to μ. If μ fails to meet the high challenge, then either μ never finds a potential high split in Subcase A or it eventually acts in Subcase A(ii). If μ eventually acts in Subcase A(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 A, 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 A 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 5.13 and 5.14. 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 5.15 and Property 2 of Lemma 5.20, 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 A(ii) or in a verification procedure called by Subcase B 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 A(ii) or B 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 A at every future η stage (and fails to find a potential high split) or η eventually acts in Subcase A(ii). (η cannot act in Subcase A(i) since it would win the high challenge in that subcase.) If η ever acts in Subcase A(ii), then by Lemma 5.16, 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 A 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. $\square$

Lemma 5.21. *$A = \lim_s A_s$ 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 5.20, $T_{\eta_k,s_k}(\lambda) = T_{\eta''_k,s_k}(\alpha_{\eta_k})$ 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. $\square$

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 5.22. *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 5.12, $T_{\eta,s}(\alpha)$ 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. $\square$

Lemma 5.23. *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 5.20, $T_{\eta,t_0}(\lambda) = T_{\eta'',t_0}(\alpha_\eta) \subseteq A_{\eta,t_0}$ 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 A(ii) or B 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 5.23 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 5.23 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. Because $\eta * N$ is the rightmost outcome of η, we are never to the left of $\eta * N$ in the tree of strategies after stage t_0. Therefore, η must take outcome $\eta * N$ at every future η stage. Property 4 follows from the fact that whenever η takes outcome $\eta * N$, it defines $T_{\eta,s}$ trivially from $T_{\eta'',s}$. □

Lemma 5.24. *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. The only place where computations of the form $\Gamma^\gamma(x) = 1$ are defined is in Case 4 of the action of a P strategy. 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 5.15 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 5.25. *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 5.16, $T_{\hat{\nu},u}(\beta_{\hat{\nu},H}*0) \subseteq T_{\hat{\nu},s_0}(\beta_{\hat{\nu},H}*0)$, 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 A(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 B 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 A(ii) and B, this step completes the proof. □

Lemma 5.26. *For all x, if $x \notin B$, then $\Gamma^A(x) = 0$.*

Proof. Because Case 4 of the P action is the only place that elements are enumerated into B, we have that $x \in B$ 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 $\Gamma^\emptyset(x) = 0$. 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 $\Gamma^{\emptyset}(x) = 0$. 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 5.10, $T_{\eta',s}(\alpha_\eta * 0)$ is on the current path. We split into two subcases. For the first subcase, suppose η never calls a verification procedure. By Lemma 5.14, $T_{\eta',s}(\alpha_\eta * 0)$ 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 5.7, 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 A(ii) or B 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 5.25, $x_\mu = x_\eta$ 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 5.15. 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 A(ii) or B 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 5.25, 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 A(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 5.16, $T_{\nu,s_1}(\beta_{\nu,H} * 1 * 0) \subseteq T_{\nu,s_2}(\beta_{\nu,s_2} * 1 * 0)$, $T_{\nu,s_2}(\beta_{\nu,H} * 1 * 0)$ is on the current path and it remains on the current path unless ν calls a verification procedure in Subcase B 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 B 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 5.16, $T_{\nu,t}(\beta_{\nu,H} * 0) \subseteq T_{\nu,s_3}(\beta_{\nu,H} * 0)$ (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. $\square$

We get the following result as an immediate consequence of Lemmas 5.24 and 5.26.

Lemma 5.27. *$\Gamma^A = B$, so $B \leq_T A$.*

Lemma 5.28. *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. This procedure ends after finitely many η stages so η eventually reaches Step 4 and enumerates x_η into B winning P. $\square$

Lemma 5.29. *If $\eta * N$ is on the true path, then Γ^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 B 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 B of the high challenge for the first time at stage s. At the previous η stage $t < s$, η must have acted in Subcase A(ii) of the high challenge and defined the parameter w_η. As in the proof of Lemma 5.16, $T_{\eta,s}(\beta_{\eta,H} * 1 * 0) \subseteq A_{\eta,s}$ 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 5.19 (and because η never meets this low challenge), $T_{\eta,s}(\beta_{\eta,L} * 1)$ remains on the current path forever. By Lemma 5.23, 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 5.30. *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 5.20 (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 5.23 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 5.2, 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 A(ii) or B 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 5.31. *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 $T_{\eta,s_\lambda}(\lambda) \subseteq A_{\eta,s_\lambda}$. 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 5.22). 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 5.22 and the proof of Lemma 5.23, 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 hence 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 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_η^X 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 5.2, 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 $\Delta^X_\eta \restriction l_{i+1} = T_{\eta,t_{i+1}}(\sigma_{i+1})$. 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 $\Delta^X_\eta \restriction l_{i+1}$. This completes the proof that the use is bounded by k. □

References

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

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

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

[4] 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.

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

[6] 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.

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

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

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

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

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

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

[13] A. Nabutovsky and S. Weinberger, *The fractal nature of Riem/Diff I*, Geom. Dedicata 101 (2003), 1–54.

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

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

[16] D.B. Posner, A survey of non-r.e. degrees $\leq 0'$, in *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.

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

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

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

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

[21] R. I. Soare, *Computability theory and differential geometry*, Bull. Symbolic Logic 10 (2004), 457–486.

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

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