CUPPABLE DEGREES AND THE HIGH/LOW HIERARCHY

NOAM GREENBERG, KENG MENG NG, AND GUOHUA WU

Abstract. Ambos-Spies, Jockusch, Shore and Soare proved in [2] that a c.e. degree is noncappable if and only it is low-cuppable. Extending this low-cuppability, Li, Wu and Zhang proposed in [6] the notion of \textit{low}_{n}-cuppable degrees for \( n > 0 \), where a c.e. degree \( a \) is \textit{low}_{n}-cuppable if there is a low \( n \) c.e. degree \( b \) such that \( a \lor b = 0' \). The class of \textit{low}_{n}-cuppable degrees is denoted by \( LC_{n} \). This gives rise a classification of cuppable degrees. Li, Wu and Zhang proved that there is a \textit{low}_{2}-cuppable degree, but not \textit{low}_{1}-cuppable, i.e. \( LC_{1} \) is a proper subset of \( LC_{2} \). In this paper, we show the existence of an incomplete cuppable degree, which can only be cupped to \( 0' \) by high degrees. Thus, \( \bigcup_{n} \text{low}_{n} \) does not exhaust all the cuppable degrees. This refutes a claim of Li [5] that all cuppable degrees are \textit{low}_{3}-cuppable. It is still open whether \( LC_{n} = LC_{3} \), when \( n > 3 \).

1. Introduction

Two of the most influential concepts in the study of computably enumerable sets are that of lowness and prompt simplicity. Lowness is concerned with the intrinsic information content of a set (or rather, the lack thereof). Prompt simplicity was introduced by Maass [?] in connection with automorphisms of the lattice of the c.e. sets. This is a dynamic property which describes how fast elements may be enumerated into the set \( A \). A promptly simple set \( A \) is in some sense similar to \( \emptyset' \) in its dynamic properties. Ambos-Spies, Jockusch, Shore and Soare [2] proved a fundamental result which linked the dynamic property of a set with a degree theoretic property: \( A \) is promptly simple iff \( A \) can be cupped with a low set. They also explored other relationships. Recall that a c.e. degree \( a \) is \textit{cuppable} if there is an incomplete c.e. degree \( b \) such that \( a \lor b = 0' \). Dually, a c.e. degree \( a \) is \textit{cappable} if there is a noncomputable c.e. degree \( b \) such that \( a \land b = 0 \), and \( a \) is \textit{noncappable} if it is not cappable. Ambos-Spies, Jockusch, Shore and Soare also showed that a c.e. degree is noncappable if and only if it is low-cuppable.

Several recent results have drawn attention to this class, and variations on cuppability have been examined. Li, Wu and Zhang [6] defined a hierarchy of cuppable c.e. degrees \( LC_{1} \subseteq LC_{2} \subseteq LC_{3} \subseteq \cdots \), where \( LC_{n} = \{ a : \exists \text{low}_{n} \text{ c.e. degree } b \text{ such that } a \lor b = 0' \} \). They called these degrees \textit{low}_{n}-cuppable. They also showed that \( LC_{2} \neq LC_{1} \) by constructing a low_{2}-cuppable set which was not promptly simple. Therefore, \( LC_{1} \) is a proper subset of \( LC_{2} \), and hence this hierarchy of cuppable degrees is nontrivial, at least at the first two levels. It is open if the low_{n}-hierarchy collapses:

\textbf{Question 1.1.} Is each level of the low_{n}-cuppable hierarchy distinct from the next?

A. Li [5] has claimed the above to be true. In fact, he claimed that every cuppable set is in \( LC_{3} \). In Section 2 we refute the claim and show the existence of an incomplete cuppable degree, which can only be cupped to \( 0' \) by high degrees. Thus, \( \bigcup_{n} LC_{n} \) does not exhaust all of the cuppable degrees. In particular we prove:

\textbf{Theorem 2.1.} There is a cuppable degree \( a \) such that for any c.e. degree \( w \), if \( a \lor w = 0' \) then \( w \) is high.
We note that Ambos-Spies, Lachlan and Soare proved in [1] that if a c.e. degree $a$ cups a c.e. degree $b$ to $0'$, then there exists a c.e. degree $c$ strictly below $b$ such that $a$ cups $c$ to $0'$. Theorem 2.1 says that $c$ in Ambos-Spies, et al.’s paper can only be selected from the high degrees.

The result of Ambos-Spies, Jockusch, Shore and Soare demonstrated a certain robustness in the class of low-cuppable sets. Variations of this class have been studied, with the most notable ones being the superlow-cupparable degrees ($SLC$), and the c.e. degrees which can be cupped with an array computable c.e. degree ($AC$-cuppable). Nies asked if $SLC$ and $LC$ were equal, and this was answered in the negative by Diamondstone [8]. Recently Downey, Greenberg, Miller and Weber [3] have investigated the class of $AC$-cuppable sets. In particular, they prove that if $A$ is promptly simple, then there is an array computable set $C$ which cups with $A$. They also showed that there was an $AC$-cuppable set which was not promptly simple. Since their proof uses a priority tree, it was not immediately obvious that the constructed cupping partner was low as well. They asked if every promptly simple set had a low and array computable cupping partner. In Section 3 we answer their question and show that this is the case. In fact we will show that every cupping partner of a promptly simple c.e. set $A$ will also wtt-compute a low cupping partner for $A$:

Theorem 3.1. Suppose $A$ is promptly simple, and $C$ is such that $\emptyset' \leq_T A \oplus C$. Then, there is a low set $B \leq_{wtt} C$ such that $\emptyset' \leq_T A \oplus B$.

Finally in Section 4 we examine the class $AC$-cuppable. We show that $AC$-cuppable is exactly the same as $LC_2$; as a corollary we get the result of Downey, Greenberg, Miller and Weber:

Theorem 4.1. Suppose $\emptyset' \leq_T A \oplus C$ where $C$ is low_2. Then, there is an array computable set $B$ such that $\emptyset' \leq_T A \oplus B$.

A superlow set $A$ can be viewed as a low set with a computable bound (on the number of mind changes witnessing the lowness of $A$). This analogy can be extended to compare an array computable set with a low_2 set in the same way; indeed array noncomputable sets share many of the properties of a nonlow_2 set with respect to lattice embedding. Our result in Theorem 4.1 says that the expected analogue of Diamondstone’s result does not hold. This supports the intuition that with enough arithmetical complexity, the classes $LC_n$ cannot be separated from each other. Finally we remark that the cupping classes low_2 + $AC$-cuppable and $AC$-cuppable coincide with $LC$ and $LC_2$ respectively and we in fact get nothing new:

$$SLC \Rightarrow LC_1 = \text{low} + AC\text{-cuppable} \Rightarrow LC_2 = AC\text{-cuppable}.$$  

Our terminology and notation are standard and follows Soare’s book [9]. A number $p$ is defined as fresh at stage $s$ means that $p > s$ and $p$ is not mentioned so far.

2. Cupping with only high degrees

Theorem 2.1. There is a cuppable degree $a$ such that for any c.e. degree $w$, if $a \lor w = 0'$ then $w$ is high.

We will construct four c.e. sets $A, C, E, P$ and a p.c. functional $\Gamma$ satisfying the following requirements:

- $G: \emptyset' = \Gamma^{A \oplus C}$;
- $P_e: E \neq \Phi_C^e$;
2.1. Description of strategies. In the following, we describe strategies satisfying these requirements.

2.1.1. The $G$-strategy. The $G$-strategy codes the halting problem $\emptyset'$ into $A \oplus C$ in a standard way as follows:

1. **Rectification:**
   - If there is an $x$ such that $\Gamma^{A \oplus C}(x) \neq \emptyset'(x)$, then let $k$ be the least such $x$, and enumerate $\gamma(k)$ into $C$. We also require that if $x < y$ then $\gamma(x)$ is always less than $\gamma(y)$, if both are defined at a stage. So if $\gamma(x)$ is enumerated into $C$, then all $\Gamma^{A \oplus C}(y)$, $y \geq x$, will also become undefined.

2. **Extension:**
   - Let $k$ be the least number $x$ such that $\Gamma^{A \oplus C}(x) \uparrow$, define $\Gamma^{A \oplus C}(k) = \emptyset'(k)$ with the use $\gamma(k)$ fresh.

Note that according to the description above, the $G$-strategy never enumerates any number into $A$. It will not be true generally in the construction, as otherwise, $C$ would be complete, contradicting the $P$-requirements. Therefore, the $G$-strategy described above will threaten some of the $P$-requirements. We will see soon how to solve this problem in the description of the $P$-strategies.

Returning to the building of $\Gamma$, we will ensure that the use function $\gamma$ of $\Gamma$ has the following basic properties:

1. Whenever we define $\gamma(x)$, we define it as a fresh number;
2. For any $k, s$, if $\Gamma^{A \oplus C}(k)[s] \downarrow$, then $\gamma(k)[s]$ is not in $A_{s} \cup C_{s}$;
3. For any $x, y$, if $x < y$, and $\gamma(y)[s] \downarrow$ then $\gamma(x)[s] \downarrow$ with $\gamma(x)[s] < \gamma(y)[s]$;
4. $\Gamma^{A \oplus C}(x)$ is undefined at a stage $s$ iff some number $z \leq \gamma(x)$ is enumerated into $A$ or $C$;
5. If $\Gamma^{A \oplus C}(x)$ is defined at a stage $s$, and $x$ enters $\emptyset'$ at stage $s + 1$, then a number $z \leq \gamma(x)$ is enumerated into $A$ or $C$.

Rules (1-5) above ensure that if $\Gamma^{A \oplus C}$ is totally defined then $\Gamma^{A \oplus C}$ computes $\emptyset'$ correctly, and hence $G$ is satisfied. So the crucial key of the $G$-strategy is to make $\Gamma^{A \oplus C}$ totally defined. These rules are called the $\gamma$-rules.

2.1.2. A $P$-strategy. A $P$-strategy is a variant of the Friedberg-Muchnik strategy. That is, we choose $x$ as a big number first, and wait for $\Phi_{e}^{C}(x)$ to converge to 0. If it never occurs, then $P$ is satisfied. Otherwise, suppose that $\Phi_{e}^{C}(x)$ converges to 0 at stage $s$, then we want to preserve this computation and also enumerate $x$ into $E$ to make $E(x) = 1 \neq 0 = \Phi_{e}^{C}(x)$, satisfying $P$. However, as coding $\emptyset'$ into $A \oplus C$ has the highest priority, when we want to preserve this computation $\Phi_{e}^{C}(x)$, we need to make sure that this computation is clear of the coding markers. That is, we want to ensure that no smaller $\gamma$-uses can later be enumerated into $C$ to change this computation.
For this purpose, we set a number \( k \) as fixed, and whenever we want to preserve a computation \( \Phi^C_e(x) \), we enumerate \( \gamma(k) \) into \( A \) to undefine \( \gamma(y) \) for \( y \geq k \). This action ensures that if \( \gamma(y) \), \( y \geq k \), is defined later, then it will be defined as a number bigger than \( \varphi_e(x) \), and hence this computation \( \Phi^C_e(x) \) will not be changed even if the newly defined \( \gamma(y) \) is enumerated into \( C \) later.

Now consider the possible enumerations of \( \gamma(y) \) into \( C \) for \( y < k \). These numbers can be small, and enumerating these numbers into \( C \) can definitely change computation \( \Phi^C_e(x) \). We cannot prevent such enumerations from happening, but fortunately, such enumerations can happen only when \( \emptyset' \) changes below \( k \), which can happen at most \( k \) many times. Hence if \( k \) is fixed, then after \( \emptyset' \not\models k \) settles down, no further such enumerations can happen again. If we now want to preserve a computation \( \Phi^C_e(x) \), then enumerating \( \gamma(k) \) into \( A \) will ensure that this computation \( \Phi^C_e(x) \) is clear of the \( \gamma \)-uses.

We will call \( k \) the **threshold** of this \( \mathcal{P} \)-strategy, and whenever \( \emptyset' \) change below \( k \), we reset this strategy by undefining all the associated parameters, except for \( k \) itself. This strategy can be reset in this way at most \( k \) times.

2.1.3. **An \( \mathcal{R} \)-strategy.** We now consider how to satisfy an \( \mathcal{R}_e \)-requirement. We will show that if \( P = \Phi^{A \oplus W}_e \), then we can construct a p.c. functional \( \Delta_e \) such that for any \( i \), \( \text{Tot}(i) = \lim_x \Delta^W_e(i, x) \).

Here \( \text{Tot} \) is the index set \( \{ i : \varphi_i \text{ is total}\} \), a \( \Pi^0_2 \) complete set. For this purpose, we need to ensure that if \( P = \Phi^{A \oplus W}_e \), then the following subrequirements are satisfied:

\[
\mathcal{S}_{e,i} : \lim_x \Delta^W_e(i, x) = \text{Tot}(i).
\]

Our construction will proceed on a priority tree, and we let \( \beta \) be an \( \mathcal{R} \)-strategy. \( \beta \) is a mother strategy, and has two outcomes, \( f \) (for finitary, which means that \( P \) and \( \Phi^{A \oplus W}_e \) agree only on a finite initial segment) and \( \infty \) (for infinitary, which means that \( P \) and \( \Phi^{A \oplus W}_e \) agree on longer and longer initial segments infinitely often). Below the outcome \( \infty \), there are infinitely many \( \mathcal{S} \)-strategies, called **children strategies**, working together to define a p.c. functional \( \Delta_e \). Without loss of generality, we assume that \( \beta \) has outcome \( \infty \), and describe how to satisfy a single \( \mathcal{S}_{e,i} \)-requirement.

Let \( \eta_i \) be an \( \mathcal{S}_{e,i} \)-strategy below \( \beta \)'s outcome \( \infty \). \( \eta_i \) again has two outcomes, \( \infty \) and \( f \), corresponding to whether \( \varphi_i \) is total or not, respectively. If \( \eta_i \) has outcome \( f \) (i.e. \( \varphi_i \) is not total), then we will define \( \Delta^W_e(i, x) = 0 \) for almost all \( x \). If \( \eta_i \) has outcome \( \infty \) (\( \varphi_i \) is total), then we will define \( \Delta^W_e(i, x) = 1 \) for almost all \( x \). If we define \( \Delta^W_e(i, x) \) as 0 at a previous stage under outcome \( f \), and now we want to redefine it as 1 as we see that \( \varphi_i \) converges on more arguments, then we need to force \( W_e \) to have a corresponding change to undefine \( \Delta^W_e(i, x) \) first. For this purpose, before we define \( \Delta^W_e(i, x) \), we first choose a number \( z \) and keep \( z \) out of \( P \), and after we see that \( \Phi^{A \oplus W}_e \) converges to 0, we define \( \Delta^W_e(i, x) \) with use \( \delta_e(i, x) \) bigger than the use \( \varphi_e(z) \). So now if we want to redefine \( \Delta^W_e(i, x) \), we first put \( z \) into \( P \), and wait for \( W_e \) to change below \( \varphi_e(z) \), and hence below \( \delta_e(i, x) \). If there is no such a change, then we will have a global win for \( \mathcal{R} \) as

\[
P(z) = 1 \not= 0 = \Phi^{A \oplus W}_e(z).
\]

(Net that we are assuming that no small numbers have been enumerated into \( A \).) Otherwise, we get a wanted \( W_e \)-change to undefine \( \Delta^W_e(i, x) \).

Note that the idea above is fairly similar to the noncuppable degree construction. In that construction, we need to construct a noncomputable set \( B \) such that for any c.e. set \( W \), \( Q_{e,W} : P = \Phi^{B \oplus W}_e \), then \( W \) also computes \( \emptyset' \) via a p.c. functional \( \Delta \), i.e. \( \emptyset' = \Delta^W \).
In that construction, to make $B$ noncomputable, for any $\varphi_j$, we need find a number $y_j$ to make $B(y_j) \neq \varphi_j(y_j)$, which involves in enumerating numbers $y_j$ into $B$. We need to ensure that such enumerations do not injure those $Q$-strategies with higher priority. That is, the following scenario should be avoided: a number enumerated into $B$ can change the computation $\Phi_e^{B,W}(x_n)$ and lead the new use $\varphi_e(x_n)$ to a number bigger than the $\delta$-use, $\delta(n)$ say, and now $n$ enters $\emptyset'$, our enumeration of $x_n$ into $P$ can force $W$ to change below the new use $\varphi_e(x_n)$, but not below $\delta(n)$, which do not undefine $\Delta^W(n)$ as wanted. To avoid this, whenever we want to put a number $y_j$ into $B$, we put numbers, like $z$ in our previous discussion, into $P$ first, to force $W$ to have wanted changes, and hence to undefine $\Delta^W(n)$, and we put $y_j$ into $B$ only after we see such a $W$-change. Such a process delays the diagonalization, but is consistent with the strategies of making $B$ noncomputable, as once $\varphi_j(y_j)$ converges to 0, it converges to 0 at any further stage.

Our construction here also has this “delayed enumeration” feature, which is more complicated, as when we want to satisfy a $P$-requirement, we see a computation $\Phi_e^C(x)$ converges to 0, if we do not enumerate $\gamma(k)$ into $A$, but instead, we put a number into $P$ to force a $W_e$-change, then after we see a $W_e$-change, this computation $\Phi_e^C(x)$ may have been changed, and we need to wait for $\Phi_e^C(x)$ to converge to 0 again. (Note that we enumerate $x$ into $E$ only when a $W_e$-change is found, and $\Phi_e^C(x)$ still converges to 0, in which case $\gamma(k)$ is enumerated into $A$ to ensure that this computation $\Phi_e^C(x)$ is clear of the $\gamma$-uses.) Such a process can repeat infinitely often, as the $G$-strategy has the highest priority. But if so, then this $P$-strategy is actually satisfied as $\Phi_e^C(x)$ diverges.

A $P$-strategy has three outcomes, $d <_L w < L s$, where $w$ denotes the outcome that we are waiting for $\Phi_e^C(x)$ to converge to 0, $s$ denotes the outcome that eventually we succeed in putting a number into $E$ for the diagonalization, and $d$ for the outcome that $\Phi_e^C(x)$ converges and diverges alternatively infinitely often, in the way described above.

2.2. Interactions between strategies. In this section, we consider interactions between strategies at various circumstances.

First we suppose that $\eta$ is an $S$-strategy working below its mother strategy $\beta$’s infinitary outcome $\infty$. In the construction, before we define $\Delta^W_e(i,x)$, we associate a parameter $u_\eta$ to $\eta$, and if $\eta$ is visited at stage $s$, and $s$ is an $\eta$-expansionary stage, then we enumerate $u_\eta$ into $P$, to force a $W_e$-change to happen. This $W_e$-change will undefine those $\Delta^W_e(i,x)$, which have been defined under $\eta$’s finitary outcome $f$, and therefore, we can redefine $\Delta^W_e(i,x)$ as 1, at the next $\eta$-expansionary stage.

So in the construction, whenever $\eta$ changes its outcome from $f$ to $\infty$, $u_\eta$ is enumerated into $P$, and after $u_\eta$ is put into $P$, we update its value with a big number, and wait for a $\beta$-expansionary stage with the length agreement between $P$ and $\Phi_e^{A\oplus W_e}$ bigger than this new value of $u_\eta$. As $\eta$ assumes that $\beta$ has infinitary outcome, such a delay does not affect the $\eta$-strategy. We call the parameter $u_\eta$ the outcome-agitator of $\eta$. Note that if $\eta$ has outcome $\infty$, then it may happen that $u_\eta$ will be enumerated into $P$, and hence, will be updated infinitely many times.

In the construction, at an $\eta$-expansionary stage, we always define $\Delta^W_e(i,x)$ as 1, we can let the use $\delta_e(i,x)$ to be $-1$, as we never want to undefine it in the remainder of the construction. This means that we only care about those $\delta$-uses, $\delta_e(i,x)$ say, when we define $\Delta^W_e(i,x)$ as 0, under the finitary outcome of $\eta$.

One problem we need to specify here is that if we have two (or more) $S$-strategies working below $\beta$’s infinitary outcome $\infty$, $\eta_1$ and $\eta_2$ say, with $\eta_1^\infty \subseteq \eta_2$. Then $\eta_1$’s actions described above can always force $W_e$ to change successfully, and such changes can undefine $\Delta^W_e(i_2, x)$ infinitely many
times, so if \( \eta_2 \) has outcome \( f \), then \( \eta_1 \)'s action may drive \( \Delta_{e_2}^{W_e}(i_2, x) \) to diverge, which contradicts our idea on \( \eta_2 \). To avoid this, we require that even though a \( W_e \)-change can undefine \( \Delta_{e_2}^{W_e}(i_2, x) \), at the next \( \eta_2 \)-stage, if \( \eta_2 \) has outcome \( f \), which means that \( u_{\eta_2} \) has not been enumerated into \( P \), when we define \( \Delta_{e_2}^{W_e}(i_2, x) \), we need to check whether the computation \( \Phi_{e_2}^{A\oplus W_e}(u_{\eta_2}) \) has been changed or not. If this computation is also changed, then we define \( \Delta_{e_2}^{W_e}(i_2, x) = 0 \) with use bigger than the current \( \varphi_e(u_{\eta_2}) \). On the other hand, the computation \( \Phi_{e_2}^{A\oplus W_e}(u_{\eta_2}) \) keeps the same as before, then we redefine \( \Delta_{e_2}^{W_e}(i_2, x) = 0 \) with use the same as before, which is again bigger than the use \( \varphi_e(u_{\eta_2}) \).

Note that if \( \eta_2 \) has outcome \( \infty \), then the parameter \( u_{\eta_2} \) will be updated infinitely many times, and hence have no final value. In this case, at \( \eta_2 \)-expansionary stages (infinitely many), \( \Delta_{e_2}^{W_e}(i_2, x) \) will all be defined as 1 with use -1. This ensures that \( \lim_{s} \Delta_{e_2}^{W_e}(i_2, x) = 1 \) as wanted.

\( \eta \) has another associated parameter \( q_\eta \), which is designed to ensure that \( \eta \)'s work in defining \( \Delta_{e_2}^{W_e} \) can be undone whenever \( \eta \) is initialized. \( q_\eta \) is defined to be less than \( u_\eta \). While \( u_\eta \) can be updated many times, \( q_\eta \) will be kept the same, unless \( \eta \) is initialized.

Suppose that \( \eta \) is initialized, then \( q_\eta \) is enumerated into \( P \), and if no number is enumerated into \( A \) (which is guaranteed, as at this stage, no link is attached on \( \beta \)), then at the next \( \beta \)-expansionary stage, a \( W_e \)-change appears, which will undefine \( \Delta_{e_2}^{W_e}(i, x) \) if it is defined as 0. Note that \( \eta \) can define \( \Delta_{e_2}^{W_e}(i, x) \) as 1, with use -1, which means that it can never be undefined later, even though \( q_\eta \) is enumerated into \( P \). It will not matter as we are looking at whether \( \varphi_i \) is total or not. If it is total, then \( \Delta_{e_2}^{W_e}(i, x) \) should be defined as 1, even though it is not defined by the \( S_{e_2,1} \)-strategy on the true path. If \( \varphi_i \) is not total, then there are only finitely many stages where a \( S_{e_2,1} \)-strategy will want to define \( \Delta_{e_2}^{W_e}(i, x) = 1 \) for some \( x \).

Now we consider the interaction between a \( P_e \)-strategy \( \alpha \) and other \( R \)-strategies. Without loss of generality, assume that \( \alpha \) works below two \( R \)-strategies \( \beta_1, \beta_2 \) with \( \beta_1 \sim \infty \subseteq \beta_2 \sim \infty \subseteq \alpha \). After \( \alpha \) sees that \( \Phi_{e_2}^C(x) \) converges to 0 at stage \( s_0 \), it first creates a link between \( \alpha \) and \( \beta_2 \), and a number \( p_{\alpha, \beta_2} \) is enumerated into \( P \). Thus, at the next \( \beta_2 \)-expansionary stage \( s_2 \), a \( W_{e(\beta_2)} \)-change occurs, and this change undefines those \( \Delta_{e_2}^{W_{e(\beta_2)}}(i, x) \) defined by \( \eta \)-strategies between \( \alpha \) and \( \beta_2 \). That is, if now we enumerate \( \gamma(k) \) into \( A \), this enumeration will not cause incorrectness of \( \Delta_{e_2}^{W_{e(\beta_2)}} \). So at stage \( s_2 \), the previous link between \( \beta_2 \) and \( \alpha \) is cancelled, and we check whether the computation \( \Phi_{e_2}^C(x) \) has been changed since stage \( s_1 \), due to the coding of \( \emptyset' \) into \( A \oplus C \). If yes, then let \( \alpha \) have outcome \( d \). Otherwise, a link between \( \alpha \) and \( \beta_1 \) is created, and a number \( p_{\alpha, \beta_1} \) is enumerated into \( P \). At the next \( \beta_1 \)-expansionary stage \( s_3 \), the link between \( \alpha \) and \( \beta_1 \) is cancelled, and again, a \( W_{e(\beta_1)} \)-change occurs, which undefines those \( \Delta_{e_2}^{W_{e(\beta_1)}}(i, x) \) defined by \( \eta' \)-strategies between \( \alpha \) and \( \beta_1 \). Now check whether the computation \( \Phi_{e_2}^C(x) \) has been changed since stage \( s_2 \). If yes, then let \( \alpha \) have outcome \( d \). If the computation \( \Phi_{e_2}^C(x) \) is the same as before, then let \( \alpha \) perform the diagonalization by putting \( x \) into \( E \) and \( \gamma(k) \) into \( A \). This enumeration will not cause incorrectness of \( \Delta_{e_2}^{W_{e(\beta_2)}} \) and \( \Delta_{e_2}^{W_{e(\beta_1)}} \).

Thus besides the two parameters \( k(\alpha) \) and \( x(\alpha) \), \( \alpha \) also has parameters \( p_{\alpha, \beta} \), where \( \beta \subset R \) is any \( R \)-strategy active at \( \alpha \) (we will define it soon). Assume that \( \alpha \) cannot be initialized after a stage \( s \), then after \( k(\alpha) \) is defined, it will be kept the same. \( x(\alpha) \) can be cancelled for finitely many times,
when $\alpha$ is reset, due to the changes of $\Psi'$. A parameter $p_{\alpha, \beta}$ can be updated infinitely many times, in which case shows that $\Phi^C_{e(\alpha)}(x_\alpha)$ diverges ($\alpha$ has outcome $d$).

Finally we remark that $\alpha$ will not, and in fact cannot worry about ensuring the correctness of $\Delta^{W_{e(\beta)}}_{e(\beta_1)}(i, x)$ defined by $\eta$-strategies extending $\alpha$. This is because if $\alpha$ were to try and change $P$ each time it sees $\Phi^C_{e(\alpha)}(x_\alpha) \downarrow$ to force $\Delta^{W_{e(\beta)}}_{e(\beta_1)}(i, x)$ to diverge, then it may be the case that $\alpha$ does this infinitely often and thus drive the use of $\Delta^{W_{e(\beta)}}_{e(\beta_1)}(i, x)$ to infinity for some $x$. Thus whenever $\alpha$ needs to change $A$, it might now cause $\Delta^{W_{e(\beta)}}_{e(\beta_1)}(i, x)$ to be incorrect, since $\eta \supset \alpha$ is unable to force changes below $\delta(i, x)$. If $\alpha$ is never again initialized so that $k$ is stable, then $\eta$ will only be incorrect at $\Delta^{W_{e(\beta)}}_{e(\beta_1)}(i, x)$ for finitely many $x$. On the other hand if $\alpha$ is initialized then we will also ensure that the entry of $k(\alpha)$ into $P$ will also correct these $\Delta^{W_{e(\beta)}}_{e(\beta_1)}(i, x)$ values. This is where we need the fact that we are only required to make $W_{e}$ high (instead of Turing complete, as in the noncuppable case).

2.3. Construction. We are now ready to give the full construction of $A, C, E$ and $P$. Before describing the construction, we define the priority tree, $T$ say, effectively.

**Definition 2.2.** (i) We define the priority ranking of the requirements as follows:

$$G < \mathcal{P}_0 < \mathcal{R}_0 < \mathcal{S}_{0,0} < \mathcal{P}_1 < \mathcal{R}_1 < \mathcal{S}_{0,1} < \mathcal{P}_2 < \cdots < \mathcal{P}_n < \mathcal{R}_n < \mathcal{S}_{0,n} < \cdots < \mathcal{S}_{n,n} < \mathcal{P}_{n+1} < \cdots,$$

where for any requirements, $\mathcal{X}, \mathcal{Y}$ say, if $\mathcal{X} < \mathcal{Y}$, then $\mathcal{X}$ has higher priority than $\mathcal{Y}$;

(ii) A $\mathcal{P}$-strategy has three possible outcomes, $d, w, s$, with $d < L w < L s$;

(iii) An $\mathcal{R}$-strategy has two possible outcomes, $\infty, f$, with $\infty < L f$, to denote infinitary and finitary outcomes, respectively;

(iv) An $\mathcal{S}$-strategy has two possible outcomes, $\infty, f$, with $\infty < L f$, to denote infinitary and finitary outcomes, respectively.

The construction will proceed on a priority tree $T$, which is defined inductively as follows:

**Definition 2.3.** Given $\xi \in T$.

(i) Requirement $\mathcal{P}_e$ is satisfied at $\xi$, if there is a $\mathcal{P}_e$-strategy $\alpha$ with $\alpha \subset \xi$;

(ii) Requirement $\mathcal{R}_e$ is satisfied at $\xi$, if there is an $\mathcal{R}_e$-strategy $\beta$ with $\beta \prec f \subseteq \xi$; Requirement $\mathcal{R}_e$ is active at $\xi$, if there is an $\mathcal{R}_e$-strategy $\beta$, with $\beta \prec \infty \subseteq \xi$, in which case $\mathcal{R}_e$ is said to be active at $\xi$ via $\beta$;

(iii) Requirement $\mathcal{S}_{e,i}$ is satisfied at $\xi$, if either $\mathcal{R}_e$ is satisfied at $\xi$, or $\mathcal{R}_e$ is active at $\xi$ via $\beta$, and there is an $\mathcal{S}_{e,i}$-strategy $\eta$, with $\beta \prec \infty \subseteq \eta \subset \xi$.

Now we construct the priority tree $T$ as follows:

**Definition 2.4.** (i) Let the root node, $\lambda$ say, be a $\mathcal{P}_0$-strategy.

(ii) The immediate successors of a node are the possible outcomes of the corresponding strategy.

(iii) For $\xi \in T$, $\xi$ works for the highest priority requirement which has neither been satisfied, nor been active at $\xi$.

(iv) Continuing the inductive steps above, we have built our priority tree $T$. 
The following standard definition of length of agreement functions applies to both $\mathcal{R}$ and $\mathcal{S}$-strategies.

**Definition 2.5.** If $\beta$ is an $\mathcal{R}_e$-strategy, then the length of agreement function between $\Phi^A_{\epsilon W_e}$ and $P$ is:

$$l(\beta, s) = \max \{x < s : \forall y < x [P(y)[s] = \Phi^A_{\epsilon W_e}(y)[s]\},$$

$$m(\beta, s) = \max \{l(\beta, t) : t < s \text{ and } t \text{ is a } \beta\text{-expansionary stage}\}.$$ 

A stage $s$ is $\beta$-expansionary if $s = 0$ or $l(\beta, s) > m(\beta, s)$ and $l(\beta, s)$ is bigger than any number with requests from $\mathcal{P}$-strategies below $\beta$’s outcome $\infty$.

**Definition 2.6.** If $\eta$ is an $\mathcal{S}_{e, i}$-strategy, then the length of convergence function is:

$$l(\eta, s) = \max \{x < s : \forall y < x [\varphi_i(y)[s] \downarrow\},$$

$$m(\eta, s) = \max \{l(\eta, t) : t < s \text{ and } t \text{ is an } \eta\text{-expansionary stage}\}.$$ 

A stage $s$ is $\eta$-expansionary if $s = 0$ or $l(\eta, s) > m(\eta, s)$.

A $\mathcal{P}$-strategy $\alpha$ has several parameters: one is $x(\alpha)$, a candidate for the diagonalization, one is $k(\alpha)$, the threshold of $\alpha$, and the others are numbers associated to $\mathcal{R}$-strategies with higher priority, which are active at $\alpha$. Unlike $x(\alpha)$ and $k(\alpha)$, a parameter associated to $\mathcal{R}$-strategies may have infinitely many many numbers during the construction.

An $\mathcal{S}_{e, i}$-strategy $\eta$ has one initialization parameter, $q_\eta$, and one outcome parameter, $u_\eta$. Once $\eta$ is initialized, $q_\eta$ will be put into $P$ automatically. $u_\eta$ will be enumerated into $P$ whenever $\eta$ sees an $\eta$-expansionary stage and hence if $\alpha$ has outcome $\infty$, then $u_\eta$ will be updated infinitely many times.

In the construction, when a strategy $\xi$ is initialized, then all the strategies with lower priority will be also initialized automatically, and all the parameters of $\xi$ will be cancelled. When a strategy $\xi$ is reset, then all the strategies with lower priority will be also initialized automatically, and all parameters of $\xi$, except for $k_\xi$, will be cancelled.

We assume that $\emptyset'$ is enumerated at odd stages, and that exactly one element can be enumerated into $\emptyset'$ at each odd stage. We will construct a p.c. functional $\Gamma$ at odd stages and strings $\sigma_s$ to approximate the true path at even stages. The full construction is as follows.

**Stage 0:** Let $A_0 = C_0 = E_0 = P_0 = \emptyset$, and initialize all nodes on $T$.

**Stage $s + 1 = 2n + 1$:** Let $k \in \emptyset'_{s+1} - \emptyset'_s$. There are three cases.

1. For any strategy $\xi$, if $k(\xi)$ is defined and $k \leq k(\xi)$, then we reset $\xi$.
2. If $\Gamma^{A \oplus C}(k)[s] \downarrow$, then enumerate $\gamma(k)[s]$ into $C$. $\Gamma^{A \oplus C}(x)[s]$, $x \geq k$, are all undefined because of this enumeration.
3. Otherwise, find the least $x$ such that $\Gamma^{A \oplus C}(x)[s] \uparrow$. Define $\Gamma^{A \oplus C}(x)[s + 1] = \emptyset'_{s+1}(x)$ with $\gamma(x)[s + 1]$ fresh. Go to the next stage.

**Stage $s + 1 = 2n + 2$:** This stage has two phases.

**Phase I:** Phase I is divided into several substages, attempts to approximate the true path.

**Substage 0:** Let $\sigma_{s+1} \uparrow 0 = \lambda$, the root of the priority tree.

**Substage t:** Given $\sigma_{s+1} \uparrow t = \xi$. If $t = s + 1$, then let $\sigma_{s+1} = \xi$ and initialize all the nodes with priority lower than $\sigma_{s+1}$, and go to Phase II.
If \( t < s + 1 \), there are three cases:

**Case 1.** \( \xi = \alpha \) is a \( P_e \)-strategy. There are four subcases.

(\( \alpha 1 \)) If \( k(\alpha) \) is not defined, then we choose a fresh number as \( k(\alpha) \). Let \( \sigma_{s+1} = \alpha \), initialize all the nodes with priority lower than \( \sigma_{s+1} \) and go to Phase II.

(\( \alpha 2 \)) If \( k(\alpha) \) is defined, but \( x(\alpha) \), and \( p_{\alpha,\beta} \), where \( \beta \) is an \( R \)-strategy active at \( \alpha \), are not defined, then choose fresh numbers for these parameters with \( x(\alpha) < p_{\alpha,\beta} \) for all \( R \)-strategies \( \beta \) active at \( \alpha \), and \( p_{\alpha,\beta_1} < p_{\alpha,\beta_2} \) if \( \beta_1 \) has priority higher than \( \beta_2 \). Request that a later stage \( s' \) is a \( \beta \)-expansionary stage if \( s' \) is \( \beta \)-expansionary stage in the standard sense, and also \( l(\beta, s') \) is bigger than \( p_{\alpha,\beta} \).

Let \( \sigma_{s+1} = \alpha \), initialize all the nodes with priority lower than \( \sigma_{s+1} \) and go to Phase II.

(\( \alpha 3 \)) If \( k(\alpha) \), \( x(\alpha) \), and also \( p_{\alpha,\beta} \), where \( \beta \) is an \( R \)-strategy active at \( \alpha \), are all defined, and \( \Phi_{e(\alpha)}^C(x(\alpha)) \downarrow = 0 \), then among those \( R \)-strategies active at \( \alpha \), choose \( \beta \) with the lowest priority, create a link between \( \alpha \) and \( \beta \), and put \( p_{\alpha,\beta} \) into \( P \). Let \( \sigma_{s+1} = \alpha \sim w \), initialize all the nodes with priority lower than \( \sigma_{s+1} \) and go to Phase II. We say that \( \alpha \) requires attention at stage \( s + 1 \).

(\( \alpha 4 \)) If \( \alpha \) is satisfied (i.e. \( \alpha \) has already received attention), then let \( \sigma_{s+1} \uparrow (t + 1) = \alpha \sim s \). Go to the next substage.

(\( \alpha 5 \)) Otherwise, let \( \sigma_{s+1} \uparrow (t + 1) = \alpha \sim w \), and go to the next substage.

**Case 2.** \( \xi = \beta \) is an \( R_e \)-strategy. There are two subcases.

(\( \beta 1 \)) If \( s + 1 \) is not \( \beta \)-expansionary (it may happen that \( l(\beta, s + 1) \) is bigger than \( l(\beta, s') \), where \( s' \) is a \( \beta \)-stage less than \( s + 1 \), but \( l(\beta, s + 1) \) is still less than a number requested by a strategy below \( \beta \sim \infty \), in which case, we still treat this stage not \( \beta \)-expansionary), then let \( \sigma_{s+1} \uparrow (t + 1) = \beta \sim f \), and go to the next substage.

(\( \beta 2 \)) If \( s + 1 \) is \( \beta \)-expansionary, and no link between \( \beta \) and a \( P \)-strategy \( \alpha \) below \( \beta \sim \infty \) exists, then let \( \sigma_{s+1} \uparrow (t + 1) = \beta \sim \infty \). Go to the next substage.

(\( \beta 3 \)) If \( s + 1 \) is \( \beta \)-expansionary, and a link between \( \beta \) and a \( P \)-strategy \( \alpha \) below \( \beta \sim \infty \) exists, then cancel this link, and check whether the computation \( \Phi_{e(\alpha)}^C(x(\alpha)) \) has been changed since it requires attention. If yes, then let \( \sigma_{s+1} \uparrow (t + 1) = \alpha \sim d \) and go to the next substage. If no, then check whether there is an \( R \)-strategy \( \beta' \) with \( \beta' \sim \infty \subseteq \beta \), i.e., the corresponding requirement \( R \) is active at \( \beta \).

If there is such a \( \beta' \), then choose \( \beta' \) with the lowest priority, and create a link between \( \alpha \) and \( \beta' \), let \( \sigma_{s+1} = \alpha \sim w \), initialize all the nodes with priority lower than \( \sigma_{s+1} \) and go to Phase II.

If there is no such a \( \beta' \), then enumerate \( x(\alpha) \) into \( E \), and \( \gamma(k(\alpha)) \) into \( A \). Let \( \sigma_{s+1} = \alpha \sim f \), and initialize all the nodes with priority lower. Go to Phase II. We say that \( \alpha \) receives attention at stage \( s + 1 \).

**Case 3.** \( \xi = \eta \) is an \( S_{e,i} \)-strategy. If \( q_\eta \) and \( u_\eta \) are not defined, then define them as two big numbers, and request that a later stage \( s' \) is \( \beta \)-expansionary, where \( \beta \) is the mother node of \( \eta \), then \( l(\beta, s') \) is bigger than \( q_\eta \) and \( u_\eta \). Let \( \sigma_{s+1} = \eta \sim f \), and initialize all the nodes with priority lower. Go to Phase II.

If \( p_\eta \) is defined, then check whether \( s + 1 \) is an \( \eta \)-expansionary stage.
(η1) If $s + 1$ is an $\eta$-expansionary stage, then let $\sigma_{s+1} \mid (t + 1) = \eta^\infty$. Enumerate $p_\eta$ into $P$ and go to the next substage.

(η2) If $s + 1$ is not an $\eta$-expansionary stage, then let $\sigma_{s+1} \mid (t + 1) = \eta^f$. Go to the next substage.

**Phase II**

Having $\sigma_{s+1}$, for $\xi \subseteq \sigma_{s+1}$, do as follows, and then go to the next stage. Recall that for those $S$-strategies $\eta$ say, being initialized at this stage, $u_\eta$ is enumerated into $P$ automatically.

1. If $\xi = \alpha$ is a $P$-strategy, and $p_{\alpha,\beta}$ is enumerated into $P$ during Phase I, then assign a fresh number to $p_{\alpha,\beta}$.

2. If $\xi = \eta$ is an $S$-strategy, and $u_\eta$ is enumerated into $P$ during Phase I (so $s + 1$ is an $\eta$-expansionary stage), then assign a fresh number to $u_\eta$.

If $s + 1$ is an $\eta$-expansionary stage, then extend the definition of $\Delta^W_\beta$ to all arguments $(i(\eta), x)$ with $x < l(\eta, s + 1)$ such that if $\Delta^W_\beta(i(\eta), x)$ is not defined yet, then define $\Delta^W_\beta(i(\eta), x) = 1$ with use $-1$.

If $s + 1$ is not an $\eta$-expansionary stage, then extend the definition of $\Delta^W_\beta$ to all arguments $(i(\eta), x)$ with $x < s + 1$ such that if $\Delta^W_\beta(i(\eta), x)[s + 1]$ is not defined, then see whether $\Delta^W_\beta(i(\eta), x)$ has been defined so far, after the current $u_\eta$ is selected. If no, then define $\Delta^W_\beta(i(\eta), x) = 0$ with use $\delta_\beta(x) = s + 1$. Otherwise, check whether the computation $\Phi^{A \oplus W_\epsilon_\beta}(u_\eta)$ has changed from the stage when $\Delta^W_\beta(i(\eta), x)$ was defined last time.

If the computation keeps the same, then define $\Delta^W_\beta(i(\eta), x) = 0$ with use the same as before. Otherwise, define $\Delta^W_\beta(i(\eta), x) = 0$ with use $\delta_\beta(x) = s + 1$

In this case, $\eta$ has outcome $f$, and $u_\eta$ is not enumerated into $P$ at this stage.

This completes the whole construction.

### 2.4. Verification

We now verify that the construction described above satisfies all the requirements and hence Theorem 2.1 is proved. At first, we will prove that the true path $TP = \liminf_s \sigma_{2s}$ is infinite.

**Lemma 2.7.** Let $\sigma$ be any node on $TP$, then

1. $\sigma$ can only be initialized or reset at most finitely often;
2. $\sigma$ has an outcome $O$ with $\sigma^\sim O$ on $TP$;
3. $\sigma$ can initialize the strategy $\sigma^\sim O$ at most finitely many times.

Therefore, $TP$ is infinite.

**Proof.** We prove this lemma by induction on the length of $\sigma$. When $\sigma = \lambda$, the root node of $T$, $\sigma$ is a $P_0$-strategy, (1) is obviously true, as it can never be initialized, and can be reset at most $k(\lambda) + 1$ many times, once $k(\lambda)$ is selected. Let $x(\lambda)$ be the final candidate selected by $\lambda$ (by (1), such an $x(\lambda)$ exists), and without loss of generality, suppose that $\lambda$ cannot be reset after a stage $s_0$, and $\Phi^{C_0}_0(x(\lambda))$ converges to 0 at a stage $s > s_0$, then at this stage, $x(\lambda)$ is enumerated into $E$ and also $\gamma(k(\lambda))$ is enumerated into $A$. By the choice of $s_0$, and the enumeration of $\gamma(k(\lambda))$, the computation $\Phi^{C_0}(x(\lambda))[s]$ will be preserved forever, and hence $\lambda$ is satisfied at any stage after $s$. 

Thus, \( \lambda \) is on \( TP \), and (2) is true. Note that after stage \( s \), \( \lambda \) will not initialize other strategies, and (3) is true.

For any non-root node \( \sigma \in TP \), let \( \sigma^- \) be the immediate predecessor of \( \sigma \). By the induction hypothesis, suppose that the conclusions are true for \( \sigma^- \). In the following, we will prove that the conclusions in the lemma also hold for \( \sigma \).

By the induction hypothesis, let \( s_0 \) be the least stage after which \( \sigma^- \) can neither be initialized or reset nor initialize \( \sigma \). As \( \sigma^- \) has only finitely outcomes, and \( \sigma \) is on \( TP \), there is a stage \( s_1 \geq s_0 \) such that no strategy on the left of \( \sigma \) can be visited, and hence after this stage \( \sigma \) cannot be initialized by higher priority strategies. If \( \sigma \) is a \( P \)-strategy, then after stage \( s_1 \), once threshold \( k(\sigma) \) is defined, it can never be cancelled. Therefore, \( \sigma \) can be reset at most \( k(\sigma) + 1 \) times more, and (1) is true for \( \sigma \).

Now we show that (2) and (3) are also true for \( \sigma \).

The case when \( \sigma = \alpha \) is a \( P_e \)-strategy is the most complicated case. Let \( s_2 \geq s_1 \) be the last stage at which \( \alpha \) is reset. Then at a stage \( s_3 > s_2 \), \( x(\alpha) \) will be defined, which will witness \( E \neq \Phi^C_{e(\alpha)} \). If there is a stage, \( s_4 \) say, at which \( x(\alpha) \) is enumerated into \( E \), then \( E(x(\alpha)) = 1 \neq 0 = \Phi^C_{e(\alpha)}(x(\alpha)) \), as the computation \( \Phi^C_{e(\alpha)}(x(\alpha))|_{s_4} \) is protected forever (by the choice of \( s_2 \), and the enumeration of \( \gamma(k(\alpha)) \) into \( A \) at stage \( s_4 \)), making \( \Phi^C_{e(\alpha)}(x(\alpha)) = 0 \). In this case, \( \alpha \) will be on the true path, and after \( s_4 \), \( \alpha \) will take no further actions. (2) and (3) are true for \( \alpha \). So we assume that \( x(\alpha) \) is not enumerated into \( E \) in the construction.

Without loss of generality, we assume that there are infinitely many \( \alpha \)-stages at which \( \Phi^C_{e(\alpha)}(x(\alpha)) \) converges to 0, as otherwise \( \alpha \subseteq w \) will be on \( TP \), and (2), (3) are obviously true. In this case, we create links between \( \alpha \) and those \( R \)-strategies \( \beta \) with \( \beta \subseteq \alpha \) infinitely many times, and the associated computations \( \Phi^C_{e(\alpha)}(x(\alpha)) \) always change before the last link was cancelled, due to the enumeration of the \( \gamma \)-uses into \( C \) (note that no restraint is imposed before all links are created and cancelled). We thus have \( \Phi^C_{e(\alpha)}(x(\alpha)) \) diverges, which provides that \( E(x(\alpha)) = 0 \neq \Phi^C_{e(\alpha)}(x(\alpha)) \), and \( P_e \) is satisfied at \( \alpha \). In this case, \( \alpha \subseteq d \) will be on the true path, and (2) is true for \( \alpha \). Note that in the construction, after \( x(\alpha) \) is selected finally, \( \alpha \) never initialize \( \alpha \subseteq d \), and (3) is true for \( \alpha \).

When \( \sigma \) is an \( R \)-stratgy or an \( S \)-strategy, (2) and (3) are also true, as in this case, \( \sigma \) has outcome \( f \) on \( TP \) only when after a stage large enough, no further \( \sigma \)-stage can be \( \sigma \)-expansionary. Otherwise, \( \sigma \) will have outcome \( \infty \) on \( TP \). Here we note that if \( \sigma \) is an \( R \)-strategy, a strategy below \( \sigma \) can request that a stage \( s \) is \( \sigma \)-expansionary only when \( l(\sigma, s) \) is bigger than the associated numbers. This kind of requests do not affect the outcome of \( \sigma \) on \( TP \) as these requests can only be imposed by strategies below \( \sigma \), which can only be visited at \( \sigma \)-expansionary stages. (2) is true for \( \sigma \).

As in the construction, the \( R \)-strategies and \( S \)-strategies never initialize strategies with lower priority. (3) is also true for \( \sigma \).

In the construction, we may assign many (perhaps, infinitely many) numbers to a parameter \( u_\eta \) (if \( \eta \) has outcome \( \infty \)) or a parameter \( p_{\alpha, \beta} \) (when \( \alpha \) is a \( P \)-strategy with outcome \( d \)) and such kind of actions do not need \( \sigma \) to initialize other strategies. One more point is that if \( \sigma \) is the \( R \)-strategy with \( \sigma \subseteq \infty \) on the true path with the highest priority, then infinitely many times, a link between \( \sigma \) and a \( P \)-strategy \( \alpha \) below \( \sigma \subseteq \infty \) is cancelled, an enumeration action done by \( \alpha \) is followed. Again, \( \sigma \) does not initialize other strategies in this situation, while \( \alpha \) will do it.

By induction, we know that the lemma is true for all the nodes on the true path. This completes the proof of Lemma 2.7. \( \square \)
Lemma 2.8. For any $e \in \omega$, let $\alpha$ be the $P_e$-strategy on $TP$, then $P_e$ is satisfied by $\alpha$.

We now show that all $R$-requirements are also satisfied.

Lemma 2.9. For any $e \in \omega$, let $\beta$ be the $R_e$-strategy on $TP$, then $\beta$, together with its substrategies, satisfies $R_e$.

Proof. Fix $e$, and let $\beta$ be the $R_e$-strategy on $TP$. If $\beta$ has outcome $f$ on $TP$, then obviously, $P \neq \Phi_e^{A\oplus W_e}$, and $R_e$ is satisfied. So we assume that $\beta$ has outcome $\infty$ on $TP$. In this case, we need to show that if $P = \Phi_e^{A\oplus W_e}$, then $W_e$ has high degree. $\beta$ has infinitely many substrategies $S_{e,i}$ below $\beta^{\infty}$, working together to construct a p.c. functional $\Delta_\beta$ such that

$$\varphi_i \text{ is total if and only if } \lim_x \Delta_\beta^{W_e}(i, x) = 1.$$ 

Let $\eta$ be an $S_{e,i}$-strategy below $\beta^{\infty}$, and we will show that $\eta$ defines $\Delta_\beta^{W_e}(i, x)$ for almost all $x$ to ensure that $\lim_x \Delta_\beta^{W_e}(i, x) = TOT(i)$. Note that when an $S_{e,i}$-strategy $\eta'$ is initialized, the associated number (for the parameter) $q_{\eta'}$ is enumerated into $P$, and if no number is enumerated into $A$, then at the next $\beta$-expansionsry stage, $W_e$ must have a change on small numbers, which will undefine all $\Delta_\beta^{W_e}(i, x)$ defined by $\eta'$. This means that $\eta$, the $S_{e,i}$-strategy on $TP$, will define $\Delta_\beta^{W_e}(i, x)$ for almost all $x$. So we need to show that between the stage when $q_{\eta'}$ is enumerated into $P$ and the next $\beta$-expansionsry stage, no small number is enumerated into $A$. It is actually clear — without loss of generality, we assume that a number is enumerated into $A$ at the next $\beta$-expansionsry stage, whose length is required to be bigger than $u_{\eta'}$. In this case, when $\eta'$ is initialized, a link between some $P$-strategy $\alpha$ and $\beta$ is created, but no number is enumerated into $A$ at this stage. As before the next $\beta$-expansionsry stage, which is required to be bigger than $q_{\eta'}$, every $\beta$-stage is also a $\beta^{\infty}$-stage, and hence no small number, in particular, no number less than $\varphi_e(q_{\eta'})$, is enumerated into $A$. This means that the $A$-part of the computation $\Phi_e^{A\oplus W_e}(p_\eta)$ is protected, so if a new $\beta$-expansionsry stage appears, it must be true that $W_e$ changes below $\varphi_e(q_{\eta'})$, and this change definitely undefines $\Delta_\beta^{W_e}(i, x)$ defined by $\eta'$.

We now see how $\eta$ defines $\Delta_\beta^{W_e}(i, x)$ for almost all $x$. By the discussion given above, we can assume that all $\Delta_\beta^{W_e}(i, x)$ defined by those $S_{e,i}$-strategies with lower priority are undefined automatically whenever $\eta$ is visited. Let $s_1$ be the last stage at which $\eta$ was initialized.

A similar argument as the one described above will be applied. So at the next $\eta$-stage, $s_2$ say, $\eta$ selects numbers for $q_\eta$ and $u_\eta$, and requests that a further stage $s_3$ is $\beta$-expansionsry, then $l(\beta, s_3) > q_\eta, u_\eta$. From now on, in the construction, the number $u_\eta$ is enumerated into $P$ only when $\eta$ sees an $\eta$-expansionsry stage, which means all $\Delta_\beta^{W_e}(i, x)$ defined under the outcome $f$ between the last $\eta$-expansionsry stage and the current stage (not inclusive, of course) are undefined, and hence at the current $\eta$-expansionsry stage, $\eta$ can extend the definition of $\Delta_\beta^{W_e}(i, x)$ on new arguments properly. So if $\eta$ has outcome $\infty$ on the true path, then the parameter $u_\eta$ will be updated infinitely many times, and each number associated will be enumerated into $P$. In this case, all $\Delta_\beta^{W_e}(i, x)$ defined after stage $s_1$ have value $1$, and hence $\lim_x \Delta_\beta^{W_e}(i, x) = 1$. On the other hand, if $\eta$ has outcome $f$ on the true path, then after a large enough stage $s_4 \geq s_3$, there is no more $\eta$-expansionsry stage, and hence from then on, $\eta$ will define $\Delta_\beta^{W_e}(i, x) = 0$ with use $\delta_\beta(i, x)$ bigger than $\varphi_e(u_\eta)$, and
such a use can be changed only when \( W_e \) changes below \( \varphi_e(u_\eta) \). So if \( \Phi_e^{A \oplus W_e}(u_\eta) \) converges, then the \( W_e \)-part of the computation \( \Phi_e^{A \oplus W_e}(u_\eta) \) will be fixed, and hence \( \Delta^W_\beta(i, x) \) will be also defined. In this case, \( \lim_x \Delta^W_\beta(i, x) = 0 \). Thus, we have that \( TOT(i) = \lim_x \Delta^W_\beta(i, x) \), and \( S_{e,i} \) is satisfied.

We are now ready to show that \( R_e \) is satisfied at \( \beta \). This is true as \( \Delta^W_\beta \) is well-defined, according to our discussion given above. Therefore, \( TOT \leq T W_e' \), and \( W_e \) has high degree.

This completes the proof of Lemma 2.9. □

The next lemma shows that the global requirement \( G \) is also satisfied, which completes the proof of Theorem 2.1.

**Lemma 2.10.** The \( G \)-requirement is satisfied.

**Proof.** By the rectification actions performed at odd stages, if \( \Gamma^{A \oplus C}(x) \) is defined, then it must compute \( \emptyset'(x) \) correctly. Fix \( x \in \omega \), and we show that \( \Gamma^{A \oplus C}(x) \) is defined.

First note that once \( \Gamma^{A \oplus C}(x) \) is defined, it can be undefined only when one of the following occurs:

(i) a small number \( \leq x \) enters \( \emptyset' \), or
(ii) \( \gamma(k) \), where \( k \leq x \) is a threshold of a \( \mathcal{P} \)-strategy, is enumerated into \( A \) when this strategy performs diagonalization.

Obviously, by induction on \( x \), we can see that each of these two cases can happen at most finitely many times, which implies that \( \Gamma^{A \oplus C}(x) \) must be defined eventually. As a consequence, \( \Gamma^{A \oplus C} \) is total, and computes \( \emptyset' \) correctly. □

This completes the proof of Theorem 2.1.

### 3. Every promptly simple set can be cupped by a low and array computable partner

If \( A \) is promptly simple, then we know that there is a low cupping partner for \( A \). In which Turing lower cones can we find a low cupping partner for \( A \)? If \( C \) bounds a low cupping partner for \( A \), then clearly \( C \) must also cup \( A \), and so we cannot expect to find a low cupping partner for \( A \) below every noncomputable c.e. set. This is because there are promptly simple \( K \)-trivial sets. On the other hand, Theorem 3.1 says that the next best thing is true. Namely, a c.e. set \( C \) bounds (in fact, wtt-bounds) a low cupping partner of \( A \) if and only if \( C \) itself also cups \( A \). We might also view this as an analogue of the continuity of cupping.

**Theorem 3.1.** Suppose \( A \) is promptly simple, and \( C \) is such that \( \emptyset' \leq_T A \oplus C \). Then, there is a low set \( B \leq_{wtt} C \) such that \( \emptyset' \leq_T A \oplus B \).

A corollary to this is that every promptly simple set can be cupped by a low array computable set. This answers a question raised in Downey, Greenberg, Miller and Weber [3].

**Corollary 1.** If \( A \) is promptly simple, then there is a low array computable set \( B \) such that \( \emptyset' \leq_T A \oplus B \).

**Corollary 2.** There is a c.e. set which is both low and array computable, but not superlow.
3.1. Requirements and notations. We are given a promptly simple set \( A \), a set \( C \) and a Turing functional \( \Phi \) such that \( \psi = \Phi^{A \oplus C} \). Our job is to build the low c.e. set \( B \) and ensure that \( B \leq_{wtt} C \). We also need to build the Turing functional \( \Gamma \) and ensure that \( \psi' = \Gamma^{A \oplus B} \). To ensure the lowness of \( B \), we will try and preserve the computation \( J^B(e) \) each time it converges, where \( J^B(e) \) is the universal partial \( B \)-computable function. We suppress all mention of the stage number from the parameters if the context is clear. We will append \([s]\) to an expression to denote the value of the expression at stage \( s \). The use of the functionals \( \Phi, J \) and \( \Gamma \) are denoted respectively by \( \varphi, j \) and \( \gamma \). Since we get to build the functional \( \Gamma \), we think of \( \gamma(e) \) as a marker pointing at some number \( x \notin \gamma(e) \). Since we get to build the functional \( \Gamma \), we think of \( \gamma(e) \) as a c.e. set of axioms of the form \( \langle \Delta, i \rangle \). We make use of the prompt simplicity of \( A \). By the Recursion Theorem again, there is an infinite computable list of numbers \( \{i_0, i_1, \cdots\} \) where we are able to control \( \psi'(i_k) \) for all \( k \). That is, we are constructing a c.e. set \( F \) where \( F \) is a column of the Halting problem. These numbers are called agitators. For each \( e \), in order for us to set the uses of the computation \( \Gamma^{A \oplus B}(e) \) correctly, we need to pick an agitator from the list. We denote this appointed agitator by \( ag(e) \). If we use up the appointed agitator (i.e. we enumerate it into \( \psi' \) to force a change in \( A \oplus C \)), we will then appoint a fresh one for \( ag(e) \), since the old one can no longer be used.

As mentioned previously we think of \( \Gamma \) as a c.e. set of axioms of the form \( \langle x, y, \sigma \oplus \tau \rangle \) where \( x \) represents the input, \( y \) represents the output and \( \sigma, \tau \) represents the \( A \) and \( B \) use respectively. During the construction we will occasionally set \( \Gamma^{A \oplus B}(e)[s] = r \) with use \( u \). What this means is that we enumerate the axioms \( \langle e, r, \sigma \oplus B_e \upharpoonright u \rangle \) for every \( \sigma \) of length \( u \), such that \( \sigma \supset A_e \upharpoonright \varphi(\psi(\psi(e))) \). Thus this axiom remains applicable until either \( B \upharpoonright u \) or \( A \upharpoonright \varphi(\psi(e)) \) changes. If \( s < t \) are two stages such that \( \Phi^{A \oplus C}(x)[s] \upharpoonright \), then we say that the computation \( \Phi^{A \oplus C}(x) \) is stable from \( s \) to \( t \), if \( A \upharpoonright \varphi(x)[s] = A \upharpoonright \varphi(x)[t] \) and \( C \upharpoonright \varphi(x)[s] = C \upharpoonright \varphi(x)[t] \) holds.

3.2. Description of strategy. The basic idea is similar to the proof of the continuity of cupping in Ambos-Spies, Lachlan and Soare \[\]. We describe the plan here briefly. We want to build a wtt reduction \( B = \Delta^C \) (\( \Delta \) is not built explicitly in the actual construction; it is used here solely for illustrative purposes). Every marker \( \gamma(e) \) is associated with an agitator number \( ag(e) \), and we always ensure that we keep \( \gamma(e) > \varphi(\psi(e)) \), so that in the event of any \( A \)-change, the \( \gamma(e) \)- marker is lifted and we may benefit from it. Once the appropriate \( \gamma(e) \)-uses are set we will define \( B(\gamma(e)) = \Delta^C(\gamma(e)) \) with \( C \)-use \( \varphi(\psi(e)) \). The following is a helpful illustration of the situation.
There are two things we need to do in this construction: to code $\emptyset'$ and to ensure the lowness of $B$. If $e$ ever enters $\emptyset'$ and coding needs to be performed, we can simply enumerate the agitator $ag(e)$ into $\emptyset'$ (that is, into the part of $\emptyset'$ we control). The opponent will either respond with an $A$-change (and coding is automatically done for us) or he responds with a $C$-change (in which case we are now allowed to enumerate $\gamma(e)$ into $B$ for the sake of coding). Note we do not require the prompt simplicity of $A$ to carry out this step.

The second thing we need to do is to ensure that $\exists s(\exists s(J^B(e)[s] \downarrow) \Rightarrow J^B(e) \downarrow)$ for every $e$. To this end we will ensure that each time $J^B(e)[s]$ converges with use $j(e)[s]$, we will try and lift $\gamma(e)$ above $j(e)[s]$ by causing an $A$-change below $\gamma(e)[s]$. We do so by requesting for a prompt change in $A \uparrow \varphi(ag(e))$. If the prompt change is given then the marker $\gamma(e)$ would have been lifted above $j(e)$ successfully. It is only when prompt permission is denied, do we enumerate the agitator $ag(e)$ into $\emptyset'$. The result would be either an $A$-change (in which case $\gamma(e)$ is lifted successfully) or a $C$-change. If the latter happens we will need to enumerate $\gamma(e)$ into $B$ to set the marker $\gamma(e)$ above $\varphi(z)$ for a new agitator $z$. This destroys the computation $J^B(e)$, but in turn ensures that the auxiliary set $U_e$ increases in size each time we request for prompt permission. Since prompt permission will eventually be given, it follows that $\gamma(e)$ will eventually be lifted above $j(e)$, which means that $J^B(e)$ will eventually be preserved forever provided no smaller $\gamma$-marker moves.

Note that the prompt simplicity of $A$ is crucial in ensuring that the coding of $\emptyset' \leq_T A \oplus B$ is compatible with the lowness of $B$ (as we might expect is the case). This is because if we rely solely on agitators to try and lift the marker $\gamma(e)$ above $j(e)$, the opponent can simply respond with a $C$-change every time we use up an agitator (this can be the case if $C$ is merely incomplete). Also it is impossible to make $B$ superlow because even after $\gamma(e)$ is lifted above $j(e)$ successfully, smaller $\gamma(e')$-markers may still move (unpredictably) due to $j(e')$.

### 3.3 The construction.
At stage $s = 0$ we make all parameters undefined. Since $\Phi^{A \oplus C} = \emptyset'$, at the end of each stage $s$, we may wait until $\Phi^{A \oplus C}(z) \models \emptyset'(z)$ for every $z \leq$ the largest number used so far, before starting the next stage $s + 1$. At stage $s > 0$ we pick the least $e < s$ that requires attention, i.e. one of the following holds:

- **(A1)** $ag(e)$ undefined,
- **(A2)** no axioms in $\Gamma^{A \oplus B}(e)$ currently applies,
- **(A3)** the computation $\Phi^{A \oplus C}(ag(e)[s^-])$ is not stable from $s^-$ to $s$, where $s^- < s$ is the stage where the current axiom in $\Gamma^{A \oplus B}(e)$ was set,
- **(A4)** $\Gamma^{A \oplus B}(e) \not\models \emptyset'(e)$,
- **(A5)** $J^B(e) \downarrow$ with use $j(e) \geq \gamma(e)$.
We then act for e; the action to be taken depend on the first item in the list above that applies to e. We perform the required action, and end the stage.

- If (A1) applies, pick a fresh agitator for ag(e). If (A2) applies, we set \( \Gamma^{A \oplus B}(e)[s] \downarrow = \emptyset'(e) \) with fresh use \( \gamma(e) \). If (A3) applies then enumerate \( \gamma(e) \) into \( B \) to make the \( \Gamma^{A \oplus B}(e) \)-computation not applicable.
- If (A4) applies then enumerate \( ag(e) \) into \( \emptyset' \) and pick a fresh \( ag(e) \). Wait for \( C \upharpoonright \varphi(ag(e)) \) to change. If the latter happens first, do nothing else. If the former happens first, we enumerate \( \gamma(e) \) into \( B \).
- Finally if (A5) applies we enumerate \( \varphi(ag(e)) \) into \( U_e \) and request for prompt permission. If we are given permission (i.e. \( A \uparrow \varphi(ag(e)) \) changes), then do nothing else. If we are denied prompt permission, then enumerate \( ag(e) \) into \( \emptyset' \) and pick a fresh \( ag(e) \). Again wait for \( C \upharpoonright \varphi(ag(e)) \) or \( A \uparrow \varphi(ag(e)) \) to change. If the latter happens first, do nothing else. If the former happens first, we enumerate \( \gamma(e) \) into \( B \).

3.4. Verification. Firstly note that we never wait forever at some step of the construction. Next, we show that each e only requires attention finitely often. Suppose no \( e' < e \) requires attention anymore. Clearly no \( \gamma(e') \) is enumerated into \( B \) anymore for \( e' < e \). Suppose on the contrary, e requires attention infinitely often. Hence there are infinitely many stages \( s_0 < s_1 < \cdots \) such that at each of these stages \( s_i, ag(e)[s_i] \) is enumerated into \( \emptyset' \) and a new agitator is picked. This is because if there were only finitely many such stages, then \( y = \lim_i ag(e)[s_i] \) exists and e will no longer require attention after \( \Phi^{A \oplus C} \uparrow y + 1 \) becomes stable.

Suppose \( s_i \) is large, such that \( e \in \emptyset'[s_i] \) iff \( e \notin \emptyset' \). Furthermore at the end of each stage \( s_i \), we must have \( \Gamma^{A \oplus B}(e) \uparrow \) because \( ag(e)[s^-] = ag(e)[s_i] \) and \( \Phi^{A \oplus C}(ag(e)) \) is stable from \( s^- \) to \( s_i \), where \( s^- \) is the stage where the \( \Gamma[s_i]-\)axioms were set. Consequently at stages \( s_j \) for all \( j > i \) we must have (A5) applies. If prompt permission is never given at any of these stages then the set \( U_e \) is infinite (since each time we reset \( ag(e) \) fresh after being denied prompt permission), and this produces a contradiction. On the other hand if prompt permission is given at some stage \( s_j \) then the \( \gamma(e) \) use will be lifted above \( j(e) \), and so \( J^B(e)[s_j] \downarrow \) on the correct use by the induction hypothesis, and e does not need to act at stage \( s_{j+1} \), another contradiction.

So, each e only requires attention finitely often, and each \( \gamma(e) \) settles. Consequently for each e, we have \( \Gamma^{A \oplus B}(e) \uparrow = \emptyset'(e) \) (\( \Gamma \) is clearly a consistent set of axioms, by the convention that \( \varphi(x)[t] \leq \varphi(x)[s] \) if \( t < s \)). Also we have \( \exists s(J^B(e)[s] \downarrow) \Rightarrow J^B(e) \downarrow \), so that \( B \) is low. Lastly we have to show that \( B \leq_{wtt} C \): fix an x, and run the construction until stage x. If x is not yet picked as a \( \gamma \)-use, then \( x \notin B \) (since uses are picked fresh). Otherwise there is some stage \( s < x \) such that \( x \) is picked to be \( \gamma(e)[s] \) for some e; at stage s we must have \( y = \varphi(ag(e))[s] \) defined. Go to a stage \( t > s \) where \( C_t \uparrow y = C \uparrow y \), such that no \( e' < e \) requires attention at t and \( ag(e)[t] \downarrow \). Then it is not hard to see that we have \( x \in B \) iff \( x \in B_{t+1} \).

4. Every c.e. set which is low\(_2\)-cuppable can also be cupped by an array computable set

In this section we show that every c.e. set which is low\(_2\)-cuppable can be cupped by an array computable set.

**Theorem 4.1.** Suppose \( \emptyset' \leq_T A \oplus C \) where \( C \) is low\(_2\). Then, there is an array computable set \( B \) such that \( \emptyset' \leq_T A \oplus B \).
4.1. **The method of exploiting low$_2$-ness.** The most common method of exploiting the low$_2$-ness of $C$ is to use it to help us guess whether a certain $\Gamma^C$ is total. Since $C$ is low$_2$ the totality of $\Gamma^C$ can be approximated by a $\Delta^0_3$ procedure. One can coordinate this with other requirements on the construction tree, and at certain nodes the leftmost outcome will give the true answer to the totality of $\Gamma^C$. We present here a slightly different way of using the fact that $C$ is low$_2$, by developing an analogue of Robinson’s technique for low sets.

Recall that a c.e. set is low$_2$ if and only if there is some $f \leq_T \emptyset'$ such that for every total $g \leq_T A$, we have $g(x) < f(x)$ for almost all $x$. The construction can be viewed as a game between us and the opponent. The opponent has control over the function $f$ and the low$_2$ set $C$. He is responsible for giving us a computable approximation $f(x)[s]$ of $f(x)$. We will at the same time be building a reduction $\Delta^C$ (a different reduction will be built at each requirement), and we will ensure that if we are unable to keep $\Delta^C$ total, then we would have an automatic win at the requirement. We will try and keep $\Delta^C$ total. Each axiom $\langle \sigma, x, y \rangle$ we enumerate into $\Delta$ will represent a challenge to the opponent, which can be viewed as a request for the certification of the correctness of $\sigma$ as an initial segment of $C$. If $C$ is low (instead of merely low$_2$), then as in the Robinson’s technique the opponent has to respond promptly to each of our challenges. This means that he has to either demonstrate that $\sigma \not\subset C$ by changing $C$ below $|\sigma|$, or else he has to give us certification that $\sigma \subset C$. One of these two scenarios must occur if $C$ is low, and we could wait for one of them to happen before proceeding with the construction. Since $C$ might be noncomputable, the opponent could of course first certify that $\sigma \subset C$, and then later change $C$ below $|\sigma|$, but however he can only do so finitely often.

Now consider the situation we are in, that is, $C$ is now merely low$_2$. Low$_2$-ness is a weak form of lowness, and the opponent cannot after all change $C$ too often. What kind of challenges can we issue the opponent so that he has to give us usable certification of the correctness of $C$-segments? Each time we issue him a challenge by extending the domain of $\Delta^C(x)$, through the enumeration of an axiom $\langle \sigma, x, y \rangle$, he can now respond with one of the following:

(i) demonstrate $\sigma \not\subset C$ by changing $C \upharpoonright |\sigma|$, 
(ii) certify that $\sigma \subset C$, in the sense that he switches the approximation of $f(x)$ to become larger than $y$,
(iii) do nothing at all for the time being.

The third outcome is annoying, and unlike the case where $C$ is low, we now have to alter our strategies to live with the fact that the opponent can choose not to give us a definite answer (i) or (ii). Hence, as we proceed in the construction, we have to make sure that our future actions do not undo or destroy our previous attempts, because these might be pending the opponent’s response in the future. The low$_2$-ness of $C$ will force him to respond with (i) or (ii) at almost every challenge we issue, but possibly with a huge delay in between, conditional of course, on the fact that we keep $\Delta^C$ total.

The above description of the use of low$_2$-ness is slightly different from usual, which would normally entail using a $\Delta^0_3$-guessing process to determine the totality of $C$-computable functions. However, the intuition behind our method is clear, in the sense that we want to force the opponent to show us “delayed certifications” of $C$-segments. By doing so, we are able to directly exploit the fact that “$C$ should not change very often”.

4.2. **A related result.** We will first discuss a related result of Downey, Greenberg, Miller and Weber [3]. The authors proved the following.
Theorem 4.2 (Downey, Greenberg, Miller and Weber [3]). If $A$ is promptly simple, there is an array computable set $B$ such that $\emptyset' \leq_T A \oplus B$.

In [2], Ambros-Spies, Jockusch, Shore and Soare showed that prompt simplicity and low-cuppability were the same. Hence they were able to demonstrate an interplay between the dynamic property of a set (which describes how fast elements enter a set), with the traditional concept of lowness as an oracle. From Theorem 4.2, we know that if $\emptyset' = \Gamma^{A \oplus B}$ where $C$ is low, then we should be able to construct an array computable set $B$ and a functional $\Phi$ such that $\emptyset' = \Phi^{A \oplus B}$. How would such a proof go?

By Ishmukhametov [4], we know that for c.e. sets, the array computable sets are exactly the same as c.e. traceable sets. Here, we say that a set $B$ is c.e. traceable, if there is a computable, nondecreasing and unbounded function $h$, such that for every total $B$-computable function $\Psi^B_e$, there is a uniformly c.e. sequence of sets $\{T_x\}_{x \in \mathbb{N}}$ such that for every $x$, $|T_x| \leq h(x)$ and $\Psi^B_e(x) \in T_x$. We also call $\{T_x\}_{x \in \mathbb{N}}$ a c.e. trace for $\Psi^B_e$ with bound $h$. We also call a computable, nondecreasing and unbounded function an order function. The choice of the order function $h$ in the definition of c.e. traceability is not important. That is, if $B$ is c.e. traceable with respect to some order $h$, then it is c.e. traceable with respect to every other order $h$.

Suppose now we are given a low $C$ and $\Gamma$ such that $\emptyset' = \Gamma^{A \oplus B}$. We want to build a c.e. traceable set $B$ and a $\Phi$ such that $\emptyset' = \Phi^{A \oplus B}$. We need to satisfy the requirements

$$R_e : \text{ if } \Psi^B_e \text{ is total, then build a c.e. trace } \{T_x\}_{x \in \mathbb{N}} \text{ for } \Psi^B_e.$$ 

to ensure that $B$ is c.e. traceable. The reduction $\emptyset' = \Phi^{A \oplus B}$ is built by movable markers. We maintain a set of markers $\varphi(0), \varphi(1), \cdots$ which should be viewed as the use of the functional $\Phi^{A \oplus B}$. When we enumerate a new axiom $\langle \sigma, x, y \rangle$ into the functional, we are actually setting the marker value $\varphi(x) \downarrow = |\sigma|$. The marker value (and the corresponding computation) persists until there is an $A$ or $B$-change below $\varphi(x)$. When that happens, the previously enumerated computation becomes nonapplicable, and $\varphi(x)$ becomes undefined. When this happens, we can move or lift the marker $\varphi(x)$ to a fresh value. Since all sets concerned are c.e., once a computation for input $x$ becomes nonapplicable, it will stay undefined until we decide to enumerate a new axiom and give a new marker value for input $x$. In the actual construction, we will need two separate sets of markers $\{\varphi_A(x)\}$ and $\{\varphi_B(x)\}$ for the $A$ and $B$-use; however for simplicity we will assume that they are the same for the time being.

The global coding requirements work in the following way. Whenever some $x$ enters $\emptyset'$, we will need to cancel our previously set axiom $\Phi^{A \oplus B}(x)$. We do this by enumerating $\varphi(x)$ into $B$, since this is the only thing we have direct control over. The strategy $R_e$ wants to make $B$ c.e. traceable. Hence, it wants to restraint $B$, each time after we had committed some current value $\Psi^B_e(x)[s]$ into the trace $T_x$. Each $R_e$ will not be able to directly restraint the global coding requirements, so it will have to ensure that $B$ does not change too often, by “disengaging” dangerous $\varphi$-markers, from below the $B$-use of $\Psi^B_e$-computations.

The construction takes place on a binary tree of strategies. The $e^{th}$ level is devoted to the requirement $R_e$. Each node $\alpha$ divides its main strategy into substrategies, which are run separately by individual $\alpha$-modules $M^\alpha_0, M^\alpha_1, \cdots$. We divide $\mathbb{N}$ into infinitely many partitions, $zone^\alpha_0, zone^\alpha_1, \cdots$, and the module $M^\alpha_k$ looks after all the computations $\Psi^B_k(x)$ for all $x \in zone^\alpha_k$. At the beginning, we set $zone^\alpha_0 = \{x\}$ for all $x$. If $x$ is currently in $zone^\alpha_k$, then the module $M^\alpha_k$ will only trace the value $\Psi^B_k(x)[s]$ into $T_x$, if it manages to disengage $\varphi(k)$ from below the use of $\Psi^B_k(x)[s]$ (i.e. it manages to
make $\psi_s(x)[s < \varphi(k)]$. If the computation is later on injured (either due to higher priority requirements acting or the global coding actions), we will have to lower the tolerance of $x$, and transfer control of $x$ to a higher priority $\alpha$-module. In particular, we will now set $\text{zone}^\alpha_j = \text{zone}^\alpha_{j+1}[s] = \text{zone}^\alpha_j[s]$ for every $j \geq k$. Thus, $x$ previously had tolerance number $k$, but after the injury it would have a tolerance number of $k-1$, and it will now be handled by $M^\alpha_{k-1}$. When $x$ reaches a tolerance number of 0, then it would only be traced if $M^\alpha_0$ manages to disengage $\varphi(0)$, so $\Psi^B(x)$ will be preserved forever. Hence, each trace $T_x$ will have size at most $x$. There are two issues here, and we will describe how they are handled at a later stage; firstly we have to ensure that each $\varphi(k)$ is moved only finitely often so that $\Phi^{A \oplus B}$ is total. Secondly, we have to ensure that for almost all $x$, we have the $\Psi^B(x)$ values traced; after all it is useless if, for instance, every $x$ reaches a tolerance number of 0.

**Atomic strategy of a single $M^\alpha_k$:** the subscript $\alpha$ is dropped if the context is clear. We first describe how a single $M^\alpha_k$ intends to disengage $\varphi(k)$ from below the use $\psi(x)$ of every $x \in \text{zone}^\alpha_k$. We reserve infinitely many indices called agitators (by the Recursion Theorem), where we can specify if an index $ag^\alpha_k$ is in $\emptyset'$. We wait for $\Gamma_{A\oplus C}(ag^\alpha_k)[s] \downarrow$ and $\Psi^B(x)[s] \downarrow$, such that $\gamma(ag^\alpha_k) < \varphi(k)$ (in the latter case, we say that $M^\alpha_k$ is set correctly). If $M^\alpha_k$ is not set correctly, we will simply enumerate $\varphi(k)$ into $B$ to clear the $\Phi$-axioms, and move the $\varphi(k)$ marker above $\gamma(ag^\alpha_k)$ in order to set $M^\alpha_k$ correctly. Our ultimate aim is to force an $A \upharpoonright \varphi(k)$-change so that we can lift the marker $\varphi(k)$ without damaging the $\Psi^B(x)$-computation. This is analogous to “requesting for prompt permission”. In this case, we enumerate a challenge $\Delta^C(p)[s] \downarrow$ with $C$-use $\sigma = C_s \upharpoonright \gamma(ag^\alpha_k)$. As discussed in Section 4.1, the opponent has to either demonstrate that $\sigma \notin C$, or else he has to certify that $\sigma \subset C$. If the latter happens we could enumerate $ag^\alpha_k$ into $\emptyset'$ and wait for a $A$ or $C$-change below $\gamma(ag^\alpha_k)$. If an $A$-change is given, then we could lift the $\varphi(k)$-marker (analogously, prompt permission has been given). If the opponent responds in any other way (i.e. prompt permission has been denied), then we would enumerate $\varphi(k)$ into $B$ to kill the current $\Psi^B(x)$-computation, and wait for it to converge again. The point is that prompt permission can only be denied finitely often before $M^\alpha_k$ is finally successful, where we can then trace the $\Psi^B(x)$-computations.

**Coordination between different nodes:** since each node $\alpha$ may make infinitely many enumerations into $B$ (though only finitely often for each module), we have to equip $\alpha$ with two outcomes, $\infty$ to the left of $f$. A node $\beta_1 \supset \alpha \neg \neg f$ is not allowed to injure $\alpha$, so at each stage when $\beta_1$ is visited, only the modules $M^\beta_j$ for $j > \alpha$ the previous $\alpha$-expansionary stage, are allowed to act. However, a node $\beta_0 \supset \alpha \neg \neg f$ will wait for enough $\alpha$-modules to be successful. That is, it will wait for all of $M^\alpha_0, \cdots, M^\alpha_k$ to successfully trace every computation they are looking after, before $\beta_0$ begins to act for $M^\beta_k$. By coordinating the actions of $\alpha$ and $\beta_0$ in this way, we ensure that $\alpha$ never injures any $\beta_0$-module, although it might be the case for instance, that $M^\beta_0$ acts and injures $M^\alpha_{k+1}, M^\alpha_{k+2}, \cdots$ with current traced computations. Note that $M^\alpha_k$ is safe from the actions of $M^\beta_0$, because every computation in $\text{zone}^\alpha_k$ has been successfully traced (and would have disengaged $\varphi(k)$). In the meantime, $\alpha$ will now start with the actions of $M^\alpha_{k+1}, M^\alpha_{k+2}, \cdots$. Therefore if $M^\beta_k$ itself is not injured anymore (remember that injury to $M^\beta_0$ can only come from below and never from above), it follows that $M^\alpha_{k+1}$ will eventually be successful and preserves its traced computations forever. We arrange for nodes of each length $k$ to agree never to move $\varphi(k)$. In this way, we can ensure that each $\varphi(k)$ is only moved finitely often (since there are only finitely many nodes on the construction
4.3. Incorporating low\textsubscript{2}-ness into the strategy. In the proof above, \( C \) was low. The advantage was that we could run the atomic strategy for each \( \alpha \)-module sequentially. That is, we could start with \( M_0^\alpha \), wait for it to successfully trace every computation in \( \text{zone}_{0}^\alpha \), before starting \( M_1^\alpha \). If at any point in time, \( M_k^\alpha \) is injured, we would drop back and start again sequentially beginning with \( M_k^\alpha \). Now suppose \( C \) was low\textsubscript{2}. As discussed in Section 4.1, the opponent now has a third option available to him each time we issue him a challenge. He could choose to do nothing until a much later stage (he can wait for as long as he likes). There is still hope for us, since we do not actually need to build the c.e. trace for \( B \) in a sequential manner. We will now need to go through the modules in two passes. The first pass is done sequentially, where we go through \( M_0^\alpha, M_1^\alpha, \ldots \) and issue a challenge to the opponent on behalf of each module. After the first pass through a module \( M_k^\alpha \), it will be \textit{waiting} for the opponent to respond to the challenge. We cannot go through all the modules without the opponent responding to any of the challenges; if this happens we want to make sure that \( \Delta^C \) is total which will then contradict the low\textsubscript{2}-ness of \( C \). On the other hand if the opponent goes back and responds to the challenge on some \( M_k^\alpha \), we will start the second phase and try to force a change in \( A \upharpoonright \varphi(k) \). If the opponent allows us to start the second phase on a module \( M_k^\alpha \), we will be guaranteed to at least make some progress at the opponent’s expense. Note that the opponent is of course not obliged to respond to our challenges in a sequential fashion (even if \( \Delta^C \) should turn out to be total). He could respond in a more or less random fashion, and he only has to make sure that in the limit, he responds to all but finitely many of our challenges.

Since the opponent does not have to respond promptly, an different situation arises. He could delay responding to the challenge at \( M_k^\alpha \), and allow us to first successfully trace the computations in \( M_j^\alpha \) for some \( j > k \). Suppose later on he responds to the challenge at \( M_k^\alpha \). He could force us to enumerate \( \varphi(k) \) into \( C \) by certifying an incorrect \( C \)-segment (though he can only do this finitely often), which would in turn injure all the modules larger than \( k \). Consequently, all the computations previously in \( \cup_{j>k} \text{zone}_{j}^\alpha \) will have to lower their tolerance. In particular, \( \text{zone}_{k}^\alpha \) itself might receive new values of \( x \) which were previously in \( \text{zone}_{k+1}^\alpha \), but whose tolerance had been reduced. This is alright, because after all \( M_k^\alpha \) has not yet been successful, and we could wait for \( \Psi^B(x) \) to converge for all the new \( x \in \text{zone}_{k}^\alpha \), before issuing a new challenge to the opponent. As mentioned earlier, the opponent can only respond incorrectly at each module finitely often, so each module can first become successful, and then only to have its traced computations injured by smaller modules later on, only finitely often.

There is a problem with this approach, for other requirements might be enumerating numbers into \( B \), while \( M_k^\alpha \) is waiting for the opponent to respond to an issued challenge. The reason why a promptly simple set \( A \) is so useful when combined with permitting, is because we can freeze the actions of all other requirements while waiting for the opponent to respond to an issued challenge. The point is that we will always benefit from any resulting \( A \)-change. We expect this to be a main issue here. Note that \( \psi_{\alpha}(x) \) (for \( x \in \text{zone}_{k}^\alpha \)) could converge with a very large use, so that after a challenge is issued by \( M_k^\alpha \), we could have changed \( B \) below \( \psi_{\alpha}(x) \). Now if \( A \upharpoonright \gamma(\text{ag}_{k}^\alpha) \) changes before \( \Psi^B_{\alpha}(x) \) next converges, we would be unable to benefit from the \( A \)-change. The opponent could then certify that \( C \) was correct (on the old \( \gamma(\text{ag}_{k}^\alpha) \)-use), and never change \( C \) below that. This is bad for \( M_k^\alpha \), for it would have lost all progress it had previously made on the test \( \Delta^C(p) \), and \( M_k^\alpha \) would have to pick a new \( p \) for the purpose of issuing new challenges to the opponent.
This might happen infinitely often, since $\Psi^B(x)$ can converge on ever increasing use. Even though locally, $\alpha$ requires no definite action ($\Psi^B_\alpha$ being not total), the module $M^\alpha_k$ will end up moving the global marker $\varphi(k)$ infinitely often.

To deal with the above problem, we will do the following. We ensure that every time $M^\alpha_k$ issues a challenge on behalf of every $x \in \text{zone}_k^\alpha$, and then later on some computation in $\text{zone}_k^\alpha$ is injured (we cannot prevent these types of injuries), we will also have the corresponding challenge reset so that $M^\alpha_k$ can reuse the same test $\Delta^C_\alpha(p)$. To be more specific, we wait for $\Psi^B_\alpha(x)$ to converge believably (we will explain what it means to be a believable computation) for every $x \in \text{zone}_k^\alpha$, and then challenge the opponent (via $\Delta^C_\alpha(p)$) to certify that $C \upharpoonright \gamma(\alpha g_j^\beta)$ is correct, where $p$ is the largest such that $\varphi(p) < \psi_\alpha(x)$ for any $x \in \text{zone}_k^\alpha$. We will arrange for all $\beta \supset \alpha$ to respect all computations in $\text{zone}_k^\alpha$. Hence any action which can injure the computations in $\text{zone}_k^\alpha$ will have to be either due to the global actions (which are finite in nature), or due to the actions of some $M^\beta_j$ where $\beta \subset \alpha$ and $j \leq p$. We say that a computation is $M^\alpha_k$-believable, if all modules $M^\beta_j$ where $\beta \subset \alpha$ and $j \leq p$ are set correctly for the relevant $p$. Therefore, the only reason why a module $M^\beta_j$ enumerates $\varphi(j)$ into $B$ and destroys the computation, must be due to one of the following two reasons:

(i) $M^\beta_j$ is no longer set correctly, and it enumerates $\varphi(j)$ into $B$ in an attempt to set itself correctly again. In this case, there must have been some change in $C$ which now makes $\Delta^C_\alpha(p)$ now undefined, because of $M^\alpha_k$-believability.

(ii) the opponent responded to some challenge put forward by $M^\beta_j$, where we would have enumerated $\alpha g^\beta_j$ in response, and observed a resulting $C \upharpoonright \gamma(\alpha g^\beta_j)$-change. In this case $\Delta^C_\alpha(p)$ is now undefined as well.

In either of the two cases above, $M^\alpha_k$ can now reuse the test $\Delta^C_\alpha(p)$ on the same $p$, so that previous progress on this test location is not wasted. Even if this happens infinitely often, $M^\alpha_k$ will only use finitely many agitators and hence only move $\varphi(k)$ finitely often. In this way, the actions of a larger module $M^\alpha_j$ may affect $M^\alpha_k$, but no real injury is inflicted on $M^\alpha_k$. Any $\Psi^B_x$-computation which is true, will eventually become believable. For each $k$, we need to arrange for two outcomes - $-k\infty$ to the left of $kf$. Since $\alpha$ does not know what the nodes extending it are currently doing, we cannot require $M^\alpha_k$-believability to include nodes $\beta_1 \supset \alpha^{-k}f$. Hence $\beta_1$ has to ensure that it never damages a convergent $\Psi^B_x$-computation which is pending the opponent’s response, for every $x \in \text{zone}_k^\alpha$. Because of the possibility of $\psi_\alpha(x)$ going to $\infty$ for some $x \in \text{zone}_k^\alpha$, we need to have the outcome $k\infty$. A node $\beta_0 \supset \alpha^{-k}\infty$ is only visited if $\Delta^C_\alpha(p)$ is undefined, so $M^\alpha_k$ does not care what $\beta_0$ does during the $\alpha^{-k}\infty$-stages.

4.4. **Requirements and conventions.** We are given the sets $A$ and $C$ and Turing functional $\Gamma$ such that $\emptyset' = \Gamma^{A \sqsupset C}$. Our job is to build the c.e. set $B$ and Turing functional $\Phi$ such that $\emptyset = \Phi^{A \sqsupset B}$. Since $C$ is loww, hence there is a function $f \leq_T \emptyset$ such that for every total $g \leq_T A$, we have $g(x) < f(x)$ for almost all $x$. We let $\lim_s f(x)[s] = f(x)$ be a computable approximation to $f$. We make $B$ c.e. traceable by satisfying the requirements $R_e$ for every $e$. If a set is c.e. traceable, the choice of the order does not matter. Hence we will make all the traces we build have a bound of identity size (any arbitrary choice will do), i.e. $|T_x| \leq x$.

The usual convention regarding stage numbers and notations applies. The use of the functionals $\Gamma, \Psi$ and $\Phi$ are denoted respectively by $\gamma, \psi$ and $\varphi$. The functional $\Phi$ is build as a c.e. set of axioms,
in the same way as in Theorem 3.1. We also assume the usual convention that if $\Gamma^{A\otimes C}(x)$ (similarly $\Psi^B(x)$) converges at stage $s$, then the use satisfies $x < \gamma(x) < s$. During the actual construction, there may be several actions taken one after another in a single stage $s$. It is sometimes convenient to break down a single stage into substages where a single action is taken at each substage. In this construction we will not bother with distinguishing between a stage and its substages; when we refer to a stage $s$ we actually mean the instance within the stage $s$ (or the substage of $s$) where the action is taken.

We introduce a minor difference in notations for the $\Phi$-use. We will have separate notations for the $A$ and $B$-use, denoted respectively by $\varphi_A(e)$ and $\varphi_B(e)$. One can thus think of the actual $\Phi$-use as being $2 \max\{\varphi_A(e), \varphi_B(e)\} + 1$, where extra axioms are enumerated into $\Phi$ by padding.

4.5. The construction tree. The construction takes place on an infinite branching tree. Nodes at level $e$ are devoted to the requirement $R_e$. There are two groups of outcomes, and we will alternate between the two groups in the ordering: $1\infty < L 1f < L 2\infty < L 2f < L 3\infty < L 3f < L \cdots$. The first group of outcomes $1\infty, 2\infty, \cdots$ are called infinite outcomes, in order to distinguish these from the second group $1f, 2f, \cdots$ which we will call the finite outcomes. The choice of these names have little to do with the frequency of actions and when the respective outcomes are played; in the actual construction the finite outcomes do have infinitary actions, however they are tagged as “finite” simply because certain associated $\Psi^B$-computations converge in the limit.

The ordering amongst nodes are denoted by $\alpha <_L \beta$, which means that $\alpha$ is strictly to the left of $\beta$ (i.e. there is some $i < \min\{||\alpha||, ||\beta||\}$ such that $\alpha \upharpoonright i = \beta \upharpoonright i$ and $\alpha(i) <_L \beta(i)$). We use $\alpha \subset \beta$ to mean that $\beta$ extends $\alpha$ properly, and $\alpha \subseteq \beta$ to mean $\alpha \subset \beta$ or $\alpha = \beta$. We also say that $\alpha \subset \infty \beta$ if $\alpha \cap \infty \subseteq \beta$ for some $n$, and $\alpha \subseteq \infty \beta$ if a similar situation holds with $f$ instead of $\infty$. We also write $\alpha \subseteq \infty \beta$ and $\alpha \subseteq_f \beta$ to have the obvious meaning.

We say that $\alpha$ is an $R_e$-node, if $\alpha$ is assigned the requirement $R_e$. As usual if $\alpha$ is an $R_e$-node then we write $\Psi^B\alpha$ and $\Psi^B\alpha$ to be the same thing. Each node $\alpha$ measures the totality of $\Psi^B\alpha$. As described previously, $\alpha \cap k\infty$ will be visited if $M^\alpha_k$ has its current test reset, i.e. $\Delta^C_\alpha(p)$ undefined, so that $M^\alpha_k$ does not care what happens in the region $[\alpha \cap k\infty]$. Whenever $\alpha$ plays this outcome, we initialize all nodes $\beta \supset \alpha \cap o$ for any outcome $o$ to the right of $k\infty$. Whenever $M^\alpha_k$ progresses in its atomic strategy, we will visit the outcome $k\infty$, and initialize all nodes extending an outcome $o >_L k\infty$. Any node $\beta \supset \alpha \cap k\infty$ will coordinate its actions with $\alpha$ as described previously. That is, $M^\beta_j$ will wait until all the modules $M^\alpha_{k, 1}, \cdots, M^\alpha_{j}$ are successful, before $M^\beta_j$ is allowed to act (this prevents $\alpha$-modules from injuring $\beta$-modules). To ensure the true path of construction exists, whenever a module $M^\alpha_{k}$ becomes successful, we will visit the outcome $k'f$ where $k'$ is the least place $\leq k$ where a run of successful modules starts. For instance, using $s$ to denote success, and $w$ otherwise, we could have initially:

<table>
<thead>
<tr>
<th>Module</th>
<th>$M^\alpha_0$</th>
<th>$M^\alpha_1$</th>
<th>$M^\alpha_2$</th>
<th>$M^\alpha_3$</th>
<th>$M^\alpha_4$</th>
<th>$M^\alpha_5$</th>
<th>$M^\alpha_6$</th>
<th>$M^\alpha_7$</th>
<th>$M^\alpha_8$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>

Outcome played $= 0f$. Then we have:

<table>
<thead>
<tr>
<th>Module</th>
<th>$M^\alpha_0$</th>
<th>$M^\alpha_1$</th>
<th>$M^\alpha_2$</th>
<th>$M^\alpha_3$</th>
<th>$M^\alpha_4$</th>
<th>$M^\alpha_5$</th>
<th>$M^\alpha_6$</th>
<th>$M^\alpha_7$</th>
<th>$M^\alpha_8$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>$w$</td>
<td>$w$</td>
<td>$s$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>

Outcome played $= 2f$.

<table>
<thead>
<tr>
<th>Module</th>
<th>$M^\alpha_0$</th>
<th>$M^\alpha_1$</th>
<th>$M^\alpha_2$</th>
<th>$M^\alpha_3$</th>
<th>$M^\alpha_4$</th>
<th>$M^\alpha_5$</th>
<th>$M^\alpha_6$</th>
<th>$M^\alpha_7$</th>
<th>$M^\alpha_8$</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>$w$</td>
<td>$w$</td>
<td>$s$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>
Outcome played = 5f.

<table>
<thead>
<tr>
<th>Module</th>
<th>$M_0^\alpha$</th>
<th>$M_1^\alpha$</th>
<th>$M_2^\alpha$</th>
<th>$M_3^\alpha$</th>
<th>$M_4^\alpha$</th>
<th>$M_5^\alpha$</th>
<th>$M_6^\alpha$</th>
<th>$M_7^\alpha$</th>
<th>$M_8^\alpha$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>w</td>
<td>w</td>
<td>s</td>
<td>w</td>
<td>w</td>
<td>s</td>
<td>w</td>
<td>s</td>
<td>s</td>
<td>...</td>
</tr>
</tbody>
</table>

Outcome played = 8f.

<table>
<thead>
<tr>
<th>Module</th>
<th>$M_0^\alpha$</th>
<th>$M_1^\alpha$</th>
<th>$M_2^\alpha$</th>
<th>$M_3^\alpha$</th>
<th>$M_4^\alpha$</th>
<th>$M_5^\alpha$</th>
<th>$M_6^\alpha$</th>
<th>$M_7^\alpha$</th>
<th>$M_8^\alpha$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>w</td>
<td>w</td>
<td>s</td>
<td>w</td>
<td>w</td>
<td>s</td>
<td>w</td>
<td>s</td>
<td>s</td>
<td>...</td>
</tr>
</tbody>
</table>

Outcome played = 5f.

<table>
<thead>
<tr>
<th>Module</th>
<th>$M_0^\alpha$</th>
<th>$M_1^\alpha$</th>
<th>$M_2^\alpha$</th>
<th>$M_3^\alpha$</th>
<th>$M_4^\alpha$</th>
<th>$M_5^\alpha$</th>
<th>$M_6^\alpha$</th>
<th>$M_7^\alpha$</th>
<th>$M_8^\alpha$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>w</td>
<td>w</td>
<td>s</td>
<td>w</td>
<td>w</td>
<td>s</td>
<td>w</td>
<td>s</td>
<td>s</td>
<td>...</td>
</tr>
</tbody>
</table>

Outcome played = 4f.

<table>
<thead>
<tr>
<th>Module</th>
<th>$M_0^\alpha$</th>
<th>$M_1^\alpha$</th>
<th>$M_2^\alpha$</th>
<th>$M_3^\alpha$</th>
<th>$M_4^\alpha$</th>
<th>$M_5^\alpha$</th>
<th>$M_6^\alpha$</th>
<th>$M_7^\alpha$</th>
<th>$M_8^\alpha$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>w</td>
<td>w</td>
<td>s</td>
<td>w</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>w</td>
<td>...</td>
</tr>
</tbody>
</table>

Outcome played = 2f.

Hence if $\Psi_\alpha^B$ is total, then $\Delta_\alpha^C$ is total and the opponent has to respond at all but finitely many of the modules. In between, infinitary outcomes may be played from time to time, as different $\Psi_\alpha$-computations are injured, but the above will describe the situation in the limit. In the above example we will visit $\alpha$-outcome $2f$ infinitely often, and so a node $\beta \supset \alpha \sim 2f$ can wait patiently to start any $\beta$-module. On the other hand if some infinite outcome $k\infty$ is played infinitely often then it must be that some $x \in \text{zone}_k^\alpha$ is divergent and $\Delta_\alpha^C$ is not total, so that the opponent doesn’t have to respond at all. However this is an automatic win for us at $\alpha$.

4.6. Notations for the formal construction. Each $\alpha$ builds a c.e. trace $\{T_x^\alpha\}_{x \in \mathbb{N}}$, and ensures that for all $x$, $|T_x^\alpha| \leq x$. For each $\alpha$ and $x$, we have a parameter $t_x^\alpha$ which gives keeps track of an upperbound for $x - |T_x^\alpha[s]|$. At the beginning (each time $\alpha$ is initialized) we start with $t_x^\alpha = x$. Every time a value $\Psi_\alpha^B(x)[s]$ is traced in $T_x^\alpha$ and later $B \upharpoonright \psi_\alpha(x)$ is changed, we will decrease $t_x^\alpha$ by 1. When $t_x^\alpha$ reaches 1 (if ever), then any current value traced must be preserved forever. This ensures that $|T_x^\alpha| \leq x$. During the construction we will sometimes say that we lower $t_x^\alpha$. This simply means that we decrease the value of $t_x^\alpha$ by 1. When we lower $t_x^\alpha$, we also lower $t_y^\alpha$ for every $y > x$ at the same time. The inverse of the parameter $t_x^\alpha$ is denoted by the parameter $\text{zone}_x^\alpha$, which is the set of all numbers $x$ such that $t_x^\alpha = k$. Each node $\alpha$ will be building a Turing functional $\Delta_\alpha$, which is used as a test. Following conventions, the use of currently applicable computations $\Delta_\alpha(x)$ is denoted by $\delta_\alpha(x)$.

Every $x$ in the same zone has the same goals; they all want to “disengage” the same $\varphi_B$-markers from below their use $\psi_B(x)$. That is, every $x$ in $\text{zone}_x^\alpha$ wants to ensure that $\psi_B(x) < \varphi_B(k)$. The $k$th module $M_k^\alpha$ will act on behalf of all the $x \in \text{zone}_x^\alpha$, and will elect a number $p_k^\alpha$ called a follower. The idea is that $M_k^\alpha$ will be enumerating computations for $\Delta_\alpha(p_k^\alpha)$. It also appoints a number $ag_k^\alpha$ called an agitator, and as the name suggests, will be used to force an $A \oplus C$-change.

We will introduce what we call global agitators, denoted by $Ag_k$. That is, $Ag_k$ will be used to help in the coding of $\emptyset(k)$. The primary aim of the global agitator is to ensure that if $k$ enters $\emptyset'$ and we need to code, we will only change $B \upharpoonright \varphi_B(k)$ if we have an accompanying $C \upharpoonright \gamma(Ag_k)$-change. Since we only need to use up $Ag_k$ if coding occurs, we may choose and fix the values for the global agitators in advance.

Each module $M_k^\alpha$ is in a particular state at any point in time. It can either be unstarted, ready, waiting or successful. Roughly speaking, being in the state unstarted means that the module
has just been initialized and needs to pick a new follower and agitator. The state \( \text{ready} \) represents the fact that we are waiting for an appropriate chance to define \( \Delta_C^\alpha(p_k^\alpha) \). When the module passes to state \( \text{waiting} \), we have defined \( \Delta_C^\alpha(p_k^\alpha) \) and is now waiting for the opponent to respond with \( f(p_k^\alpha) > \Delta_C^\alpha(p_k^\alpha) \). Lastly the state \( \text{successful} \) represents the fact that we have managed to disengage all dangerous markers, and have increased the trace size by 1 for all \( x \in \text{zone}_k^\alpha \). The state of a module will retain its assigned value until we assign a new state to it. There is one exception to this: if a module \( M_k^\alpha \) has state \( \text{waiting} \) and a \( C \)-change occurs so that we have \( \Delta_C^\alpha(p_k^\alpha) \) becomes undefined, then we set the module to have state \( \text{ready} \) (this is assumed to be a background task which is done implicitly, we do not mention this step in the construction). The reason why we want to implement this is to be consistent with the fact that

\[
\begin{align*}
M_k^\alpha \text{ has state } \text{ready} &\implies \Delta_C^\alpha(p_k^\alpha) \text{ is undefined,} \\
M_k^\alpha \text{ has state } \text{waiting} &\implies \Delta_C^\alpha(p_k^\alpha) \text{ is defined.}
\end{align*}
\]

We say that the module \( M_k^\alpha \) is \textit{set correctly}, if its state is not \textit{unstarted}, and \( \gamma(ag_k^\alpha) \leq \varphi_A(k) \), with everything mentioned here defined. That is, a module is said to be set correctly if its agitator has its use in the correct place. Similarly, the global agitator \( Ag_k \) is said to be set correctly, if \( \gamma(Ag_k) \leq \varphi_A(k) \). We define \( i_k^\alpha \) to be the module ending the longest run of consecutive \( \alpha \)-modules starting with \( M_k^\alpha \), that are in the state \( \text{successful} \). That is, \( i_k^\alpha = \text{least number } j \geq k \text{ such that } M_j^\alpha \text{ is not in the state } \text{successful} \).

**Definition 4.3 (Active modules).** With each node \( \alpha \) we associate a collection of \( \alpha \)-modules which we call \textit{active modules}. At stage \( s \), a module \( M_k^\alpha \) is said to be active, if the following holds:

1. \( k > |\alpha| \),
2. for every \( \beta \subset \alpha \) and \( n \) such that \( \beta \cap f \subseteq \alpha \), we have \( n < k < i_n^\beta \),
3. for every \( \beta \subset \alpha \) and \( n \) such that \( \beta \cap \infty \subseteq \alpha \), we have \( n < k \) and for every \( n \leq k' \leq k \), we have \( M_k^\beta \) is either \( \text{successful} \) or is set correctly,
4. \( Ag_n \) is set correctly for all \( n \leq k \).

When we visit \( \alpha \) during the construction, only the \( \alpha \)-modules which are currently active gets a chance to act. The idea is that once a module \( M_k^\beta \) enters the state \( \text{successful} \), then it never moves \( \varphi_B(k) \) anymore, and so when \( \alpha \supset \beta \) is visited we only allow \( \alpha \)-modules which are currently active to act. This is to prevent injury to the \( \alpha \)-modules by \( \beta \)-modules.

**Definition 4.4 (Believable computation).** We define what we mean by a believable computation. A computation \( \Psi^B_k(x) \) which converges at a stage \( s \) with \( B \)-use \( u \), is said to be \( \text{believable} \), if for every \( z \geq k \) such that \( \varphi_B(z) \downarrow u \), we have \( M_z^\alpha \) is currently active, and either \( \text{successful} \) or is set correctly.

When we initialize a module \( M_k^\alpha \), we set \( p_k^\alpha \) and \( ag_k^\alpha \) to be undefined, and declare the state of \( M_k^\alpha \) as \( \text{unstarted} \). When we initialize a node \( \alpha \), we first initialize all of its modules. Then, we set \( T_x = \emptyset \) and \( i_x^\alpha = x \) for all \( x \), and set \( \Delta_\alpha = \emptyset \).

4.7. **The ordering amongst modules.** The driving force behind the construction is the action of the individual modules (instead of the actions of nodes). During the construction when a node \( \alpha \) is visited, we will take actions for some \( \alpha \)-module. This action will affect and injure \( \beta \)-modules, possibly for all nodes \( \beta \) comparable with \( \alpha \). Due to these interactions between the modules of different nodes, we will introduce the following two concepts.
We first define a local priority ordering amongst the modules (of different nodes). Note that a module $M_i^\alpha$ only enumerates current $\varphi_B(k)$-marker values into $B$ (under its individual strategy). If $M_i^\alpha$ is an $\alpha$-module, we define the set of modules with a lower local priority, to be all the modules $M_j^\beta$ for some $\beta \subset_f \alpha$, and $k \leq i$. We also include all the modules $M_i^\beta$ for $i > k$ in this list (i.e. larger $\alpha$-modules are also of a lower local priority).

We say that a computation $\Psi_B^\beta(x)[s]$ is a current traced computation at $s$, if $\Psi_B^\beta(x)[s] \downarrow \in T_x^\beta$, and there is no change in $B$ below $\psi_\beta(x)$ since the time the value was traced in $T_x^\beta$. Furthermore we also require that $t_x^\beta$ was not lowered since the time the value was traced. That is, current traced computations are the computations that we should not allow to be injured easily. Another minor technical point to note is the following. We will think of the trace $T$ if this case, we also say that $M_i^\alpha$ has persisted between two stages $s$ and $t$, say that a computation $\Psi_B(x)$ is traced at $s$, where $s = t - s_x^\beta$ is the previous visit to $x$ in $T_x^\beta$. At stage $s$, we will also say that a computation $\Psi_B^\beta(x)$ has persisted for two visits to $\alpha$, if $\Psi_B^\beta(x)[s] \downarrow$ and there has been no change in $B$ below the $\psi$-use between $s^-$ and $s$, where $s^-$ is the previous visit to $\alpha$. We say that a computation has persisted between two stages $s < t$ if the above holds with the obvious modifications. The point of making this definition is to identify the following situation: a module $M_i^\beta$ may have a currently traced computation $\Psi_B^\beta(x)$, but it may be initialized after it had made the trace of $\Psi_B^\beta(x)$ into $T_x^\beta$. Namely, it is possible for a module to have a current traced computation, but it is not successful.

Next, for an $\alpha$-module $M_i^\alpha$, we define the injury set of $M_i^\alpha$ to be the set of modules $M_i^\beta$ where $\beta \supset \alpha$, $i > k$, and such that for some $x \in zone_i^\beta$, we have $\Psi_B^\beta(x)$ is a current traced computation (in this case, we also say that $M_i^\beta$ has a current traced computation). Note that if $M_i^\beta$ is successful, we will always consider it to be in the injury set as well. That is, $M_i^\beta$ is a module in which some $\Psi_B^\beta(x)$-computation which has already been traced, would be injured if $M_i^\alpha$ decides to act and enumerate $\varphi_B(k)$ into $B$. Note that the modules of a lower priority depends only on the layout of the tree and does not change with time, while the injury set varies with time.

4.8. The construction. At stage $s = 0$, initialize all nodes and do nothing. At stage $s > 0$ we define the accessible string $\delta_s$, of length $s$ inductively on its length. A node $\alpha$ is visited at stage $s$, if $\delta_s \supset \alpha$. Suppose $\alpha \subset \delta_s$ has been defined. We state the actions to be taken by $\alpha$ at stage $s$, and its successor along the accessible string. Pick the smallest $k$ (if exists) such that $s^- < k < s$ where $s^- < s$ is the previous stage where $\delta_{s^-} \subset L \alpha$, and $M_k^\alpha$ is currently active and requires attention. We say that $M_k^\alpha$ requires attention, if one of the following holds:

(A1) $M_k^\alpha$ is unstarted.
Otherwise, we have $\Phi_{A\oplus B}(k) \downarrow$, $\Gamma_{A\oplus C}(ag_k^\alpha) \downarrow = \Theta'(ag_k^\alpha)$, and as well as one of the following:

(A2) $M_k^\alpha$ is ready, and let $s^-$ be the previous visit to $\alpha$. There exists some largest $\alpha$-stage $t \leq s^-$ such that after $\alpha$ has acted at $t$, we have $M_k^\alpha$ not ready. We require that for some $x \in zone_k^\alpha$, $\Psi_B^\beta(x)$ has persisted between $t$ and $s^-$, but not between $s^-$ and $s$. In short, some relevant computation has recently been destroyed (and possibly by $\alpha$ itself at $s^-$).

(A3) $M_k^\alpha$ is ready, but is not set correctly.

(A4) $M_k^\alpha$ is ready, and for every $x \in \cup_{j \leq k} zone_j^\alpha$, $\Psi_B^\beta(x)[s] \downarrow$ and is $M_k^\alpha$-believable.

(A5) $M_k^\alpha$ is waiting, and $\gamma(ag_k^\alpha) > \delta_{\alpha}(ag_k^\alpha)[s]$. 


(A6) $M^α_k$ is waiting, and $Δ^C_α(p^α_k)[s] < f(p^α_k)[s]$.

Step 1: we will act for the $α$-module $M^α_k$. At each visit to $α$, at most one $α$-module receives attention. Choose the first item in the list above that applies, and take the corresponding action:

- (A1) applies: pick a fresh follower $p^α_k$ and a fresh agitator $ag^α_k$. Declare $M^α_k$ to be in ready state.
- (A2) applies: do nothing. This is included to ensure that infinite outcomes of $α$ gets a chance to act.
- (A3) applies: enumerate $φ_B(k)$ (if defined) into $B$.
- (A4) applies: let $p =$ the largest such that $φ_B(p) \downarrow ψ_α(x)$ for some $x \in \cup_{j<δ} zone^α_j$ (we always take $p ≥ k$), and let $z = \max\{Ag_0, \cdots, Ag_p\} \cup \{ag_β^J | β \subseteq α$ and $q ≤ p\}$. Note that all the parameters involved must be defined and set correctly, because of $M^β$-believability. Let $m = \max\{δ_α(y)[s] | y ≤ p^α_k\}$. Define $Δ^C_α(y) \downarrow =$ a fresh number, with use $C_s \downarrow γ(z) + m$, for every $y ≤ p^α_k$ where no axioms currently apply for $Δ^C_α(y)[s]$. Declare $M^α_k$ as waiting.
- (A5) applies: declare $M^α_k$ as successful. For every $x \in zone^α_k$, we enumerate the value of $ψ_B^α(x)[s]$ into the trace $T^α_x$.
- (A6) applies: enumerate $ag^α_k$ into $∅'$ and pick a fresh agitator. Wait for either $C \downarrow γ(ag^α_k)$ or $A \downarrow γ(ag^α_k)$ to change (one of the two must change). If $A$ changes do nothing, else if $C$ changes then we enumerate $φ_B(k)$ (if defined) into $B$.

If an enumeration was made into $B$ in Step 1, we say that the module $M^α_k$ was injurious. We separate this case from the rest because these are the “bad actions” which will injure the other modules on the tree.

Step 2: we now determine the effect that our actions in step 1 has (if any) on the other modules in the construction. If $M^α_k$ was not injurious, we do nothing. Otherwise we do the following.

- for every $x \in zone^β_j$ such that $M^β_j$ is of lower local priority (than $M^α_k$) and $j > k$, we lower $t^β_x$.
- we initialize all modules of a lower local priority.
- for every $β ⊃ α$, do the following. Let $i > k$ be the least (if any) such that $M^β_i$ is in the injury set of $M^α_k$. Note that we naturally consider the injury set before step 1 is taken, so that $M^β_i$ is the smallest $β$-module which has a current traced computation being injured by the action in step 1. We then initialize all modules $M^β_j$ for all $j ≥ i − 1$, and we lower $t^β_x$ for every $x \in \cup_{j>i} zone^β_j$.

Step 3: now decide which outcome of $α$ is accessible. Suppose that $M^α_k$ has just received attention (if no module received attention let $k = s$). If (A5) was the action taken, let $δ_s(|α|) = jf$ where $s^- < j ≤ k$ is the largest such that $M^α_j$ is active and $M^α_{j-1}$ is not successful. Otherwise if (A5) was not the action taken, we search for the least $j$ with $s^- < j ≤ k$ such that $M^α_j$ is ready and active, and for some $x \in zone^α_j$, $ψ^α_B(x)$ has not persisted for at least two visits to $α$. If $j$ exists, we let $δ_s(|α|) = j∞$, otherwise we let $δ_s(|α|) = kf$.

Global actions: at the end of stage $s$, we initialize all nodes $β > L δ_s$, and take the global actions for coding: pick the smallest $e < s$ such that either

(i) $Ag_0$ is not set correctly, or
(ii) $φ^{A ∪ B}(e) \not\in ∅(e)$
holds. If (i) holds and $\varphi(e) \downarrow$, we will enumerate $\varphi_B(e)$ into $B$. On the other hand if (ii) holds, then we will enumerate $Ag_\alpha$ into $\emptyset'$, and wait for $A \uparrow \gamma(Ag_\alpha)$ or $C \uparrow \gamma(Ag_\alpha)$ change. If $C \uparrow \gamma(Ag_\alpha)$ changes we will enumerate $\varphi_B(e)$ into $B$. In addition, if we had enumerated $\varphi_B(e)$ into $B$ in the previous step, we will also perform the following: for every node $\alpha$ and every $k \geq e$, we initialize the module $M_k^\alpha$. We also lower $t_z^\alpha$ for every $x \in \text{zone}_z^k$ where $k > e$.

Next, we extend the domain of $\Phi$. Wait for $\Gamma^{A \uparrow B}(z) \models \emptyset(z)$ for every $z \leq s$ the largest number used so far, and $z < s$. For every $e < s$ for which no axiom is applicable to $\Phi^{A \uparrow B}(e)$, we set $\Phi^{A \uparrow B}(e) \models \emptyset'(e)$ with a fresh $B$-use $\varphi_B(e)$, and $A$-use $\varphi_A(e)$ equals to $\max(\gamma(Ag_\alpha^e) \mid$ for any $\alpha$ such that $ag_\alpha^e \downarrow \cup \{\gamma(Ag_\alpha)\}$. Clearly there is a danger that the $A$-use $\varphi_A(e)$ might get driven to infinity, but we will later show that this cannot be the case.

4.9. Verification. It is easy to verify the following facts.

**Fact 4.1.** Suppose $M_j^\alpha$ is a module which is active at a stage $s$. Then for any $k < j$, either $M_k^\alpha$ is also active at $s$, or else $M_k^\alpha$ will never be active after $s$.

**Fact 4.2.** Observe that we never wait forever at some step of the construction. When a module $M_i^\alpha$ acts, it can only enumerate the current marker value of $\varphi_B(k)$ into $B$. If $M_i^\alpha$ is initialized, then all larger $\alpha$-modules are also initialized by the same action. Furthermore if $M_i^\alpha$ is initialized then at the same time either the node $\alpha$ is initialized, or $\varphi(x)$ becomes undefined (due to changes below the $\varphi$-use on either the $A$ or $B$ side). For any $\alpha$ and $k$, $\text{zone}_k^\alpha$ is never empty. If $M_j^\alpha$ and $M_i^\alpha$ are both active at some stage $s$, and $j < k < i$, then $M_k^\alpha$ is also active at $s$. If $p_j^\alpha \downarrow$ and $p_k^\alpha \downarrow$ for $j < k$, then we have $p_j^\alpha < p_k^\alpha$.

It is also not too difficult to verify the following fact. The nontrivial case to consider is when $M_{k-1}^\alpha$ takes an injurious action, and initializes $M_k^\alpha$. In this situation, $M_{k-1}^\alpha$ itself is not initialized, however its state will be ready.

**Fact 4.3.** If an action is taken to lower $t_z^\alpha$ for some $x \in \text{zone}_k^\alpha$, then $M_k^\alpha$ will also be initialized and $M_{k-1}^\alpha$ will become either unstarted or ready.

**Lemma 4.5.** Suppose $M_i^\alpha$ receives attention at $s$, and $\varphi(k)[s] \downarrow$ and $s^- < s$ is the previous visit to the left of $\alpha$. Then, every module $M_i^\alpha$ which is active at $s$, for $s^- < i < k$ must either be successful or be set correctly at $s$.

**Proof.** $M_i^\alpha$ cannot be unstarted (otherwise $M_i^\alpha$ would have received attention at $s$). If $M_i^\alpha$ is ready then it has to be set correctly (otherwise again, $M_i^\alpha$ would have received attention). This leaves the case $M_i^\alpha$ is waiting. Since $\varphi(i)$ is defined at $s$, hence by considering $\Delta_i^s(p_i^\alpha)$, it is not difficult to see that at $s$ we must have $M_i^\alpha$ set correctly as well. \hfill $\Box$

**Lemma 4.6.** Suppose $M_k^\alpha$ is waiting at stage $s$. Then for every $x \in \cup_{i \leq k} \text{zone}_j^\alpha$, we have $\Psi_{\alpha}(x)[s] \downarrow$. Furthermore $B \uparrow \psi_{\alpha}(x)[s]$ cannot change before $M_k^\alpha$ has a change of state.

**Proof.** Let $s_0 \leq s$ be the largest when (A4) holds to change $M_k^\alpha$ from ready to waiting. By Fact 4.3, $\text{zone}_j^\alpha[s_0] = \text{zone}_j^\alpha[s]$ for every $j \leq k$, it suffices to prove the lemma with $s_0$ in place of $s$. Fix an $x \in \text{zone}_j^\alpha[s_0]$, clearly $\Psi_{\alpha}(x)[s_0] \downarrow$. Suppose that $\varphi_B(z)[s_0] < \psi_{\alpha}(x)[s_0]$ is enumerated into $B$ at some least stage $t > s_0$, and some $z$. Suppose the contrary that $\beta$ has had no state change between $s_0$ and $t$. Suppose firstly that the enumeration was made by the global actions. We have $Ag_\alpha$ is set correctly at $s_0$; this follows from the fact that $M_k^\alpha$ is active at $s_0$ (if $z < k$), and $M_k^\alpha$-believability
if \( z \geq k \). In either case we have \( \delta_s(p^0_k) \) set to be larger than \( \gamma(Aq_z)[s_0] \) at \( s_0 \), and consequently neither \( \gamma(Aq_z) \) nor \( \varphi(z) \) could have changed between \( s_0 \) and \( t \). Hence, the global actions at \( t \) would destroy the \( \Delta^C_\alpha(p^0_k) \)-computation set at \( s_0 \), resulting in a change of state at \( t \).

Next, we want to show that at stage \( s_0 \) the outcome \( kj \) is played when \( \alpha \) is visited: if not then there is some \( j < k \) and \( j > s_0 \) such that \( M_j^\alpha \) is ready and active. For \( x \in \bigcup_{y \leq j} \text{zone}^\alpha_y \), we claim that the \( \Psi^B_\alpha(x)[s_0] \)-computation (besides being \( M_j^\alpha \)-believable) is also \( M_j^\alpha \)-believable at \( s_0 \). This follows by Lemma 4.5 because every module \( M_j^\alpha \) for \( j \leq i < k \) must be active (since \( M_j^\alpha \) and \( M_k^\alpha \) are). Hence \( M_j^\alpha \) would have received attention instead of \( M_k^\alpha \) at \( s_0 \), a contradiction.

Suppose now that the enumeration of \( \varphi_B(z)[s_0] \) was made by the module \( M_j^\beta \) for some node \( \beta \), at stage \( t \). Clearly \( \beta <_L \alpha \), or \( \beta >_L \alpha \) is trivial. Suppose \( \beta \geq \alpha \) for some outcome \( o \). Again \( o >_L kj \) is trivial, and from the fact that \( M_j^\alpha \) is waiting between \( s \) and \( t \), it follows that \( o = kj \) is impossible. Suppose that \( o = k^j \) for some \( k^j < k \). Since \( M_j^\beta \) is active at \( t \) it follows that at \( t \) we have \( z < l^\beta_k \leq k \), which means that after \( M_j^\beta \) acts at \( t \), we would initialize \( M_k^\alpha \) (\( M_k^\alpha \) being of lower local priority). If on the other hand we have \( o = k^j \alpha \) for some \( k^j \leq k \), then stage \( t \) is strictly after \( s_0 \). We may assume \( k^j < k \) otherwise \( M_k^\alpha \) is ready at \( t \). At \( t \) there is some \( x' \in \text{zone}^\alpha_{k^j}[t] \) such that \( \Psi^B_\alpha(x') \) has not persisted for two visits to \( \alpha \). Since \( M_k^\alpha \) had no change in state between \( s_0 \) and \( t \), it follows that \( x' \in \text{zone}^\alpha_{k^j}[s_0] = \text{zone}^\alpha_{k^j}[t] \), which means that \( x' < x \). Since \( \Psi^B_\alpha(x') \) had not persisted between \( s_0 \) and \( t \), this contradicts the minimality of \( t \).

We are now left with the case \( \beta \leq \alpha \). If \( \beta \geq \alpha \) then at \( s_0 \) we have \( z < l^\beta_n \) (again due to either \( M_k^\alpha \) being active or \( M_j^\alpha \)-believability). If \( z < n \) then at \( t \) some outcome to the left of \( nf \) will be played when \( \beta \) acts, resulting in an initialization to \( \alpha \). Suppose therefore, that \( n \leq z < l^\beta_n \), and hence \( M_j^\beta \) is successful at \( s_0 \). The only way for \( M_j^\beta \) to get out of successful, is for \( M_j^\beta \) to be initialized before \( t \), and so by Fact 4.2, either \( \beta \) itself is initialized or \( \varphi_B(z) \) is lifted between \( s_0 \) and \( t \), another contradiction.

Finally we have the case \( \beta \geq \alpha \) or \( \beta = \alpha \). We can conclude (again due to either \( M_k^\alpha \) being active or \( M_j^\alpha \)-believability) that \( M_j^\beta \) is either successful or is set correctly at \( s_0 \) (for \( \beta = \alpha \) and \( z < k \), use Fact 4.1 and Lemma 4.5). A similar argument as the one used above can be applied to show that \( M_j^\beta \) cannot be successful. Hence \( M_j^\beta \) has to be set correctly at \( s_0 \), and furthermore \( \alpha \) at \( s_0 \) will set the use \( \delta_s(p^0_k) \) at \( s_0 \) at least as big as \( \gamma(ag^0_\beta)[s_0] \). Between \( s_0 \) and \( t \) there can be no change in \( A \) below \( \gamma(ag^0_\beta)[s_0] \), because \( M_j^\beta \) was observed to be set correctly at \( s_0 \). There is also no change in \( C \) below \( \gamma(ag^0_\beta)[s_0] \), since \( M_k^\alpha \) was assumed to have no state change. Hence at \( t \) when \( \beta \) gets to act, it must still be that \( M_j^\beta \) is set correctly, and that \( (A6) \) applies, causing us to enumerate \( ag^0_\beta \) and get a \( C \vdash \gamma(ag^0_\beta) \)-change, which would in turn cause \( \Delta^C_\alpha(p^0_k) \) to become undefined after the action at stage \( t \).

\textbf{Lemma 4.7.} At all time, if the module \( M_k^\alpha \) is active, and \( k > \text{stage number of the previous visit to the left of } \alpha \), the following are true:

\begin{enumerate}
  \item \( M_k^\alpha \) is not unstarted \( \iff p^0_k \) and \( ag^\alpha_k \) are both defined.
  \item \( M_k^\alpha \) has state ready \( \Rightarrow \Delta^C_\alpha(p^0_k) \) is undefined.
  \item \( M_k^\alpha \) has state waiting \( \Rightarrow \Delta^C_\alpha(p^0_k) \) is defined.
\end{enumerate}

\textbf{Proof.} (i) and (iii) are obvious.

(ii): suppose on the contrary that \( s \) is a stage such that \( M_k^\alpha \) is ready and is active, but \( \Delta^C_\alpha(p^0_k) \) is has an axiom that applies. Let \( s^- < s \) be the latest action that caused \( M_k^\alpha \) to become ready. Note
that this must be either $M^\alpha_t$ receiving attention under (A1), or due to some $C$-change occurring. In any case it must be that $\Delta^C_\alpha(p^k_\alpha)[s^-]$ is undefined. Hence at some time $t$ where $s^- < t < s$ we have some module $M^j_t$ where $j > k$ taking action under (A4) and enumerating the axiom for $\Delta^C_\alpha(p^k_\alpha)[s]$. Since $M^\alpha_t$ is active at $t$, it follows by Fact 4.1 that $M^k_t$ is also active at $t$, and in fact by Lemma 4.5, every $M^\alpha_t$ has to be active, and has to be either successful or be set correctly at $t$ for all $k \leq i < j$. Hence at stage $t$ when $\alpha$ was visited, it is not hard to see that we have $t^- < k < t$, and also that $M^\alpha_t$ requires attention under (A4), because every relevant computation converges and is $M^k_t$-believable at $t$. Hence it is impossible for $M^\alpha_t$ to act at $t$, a contradiction. □

**Lemma 4.8.** Suppose that $M^\alpha_t$ has just received attention under (A5) at $s$. Then for every $x \in \cup_{j \leq k} zone^\alpha_j[s]$, we have $\Psi^B_\alpha(x)[s] \downarrow$. Furthermore if $\varphi_B(k)[s] \downarrow$, then $\varphi_B(k)[s] < \psi_\alpha(x)[s]$.

**Proof.** Let $s_0 < s$ be the stage where (A4) applies to $M^\alpha_t$ and causes the state changes from ready to waiting. It follows again by Fact 4.3 that $zone^\alpha_j[s_0] = zone^\alpha_j[s]$ for all $j \leq k$, and by Lemma 4.6, we have $\Psi^B_\alpha(x)[s_0] \downarrow$, and this computation is not injured between $s_0$ and $s$. Furthermore at $s_0$ we had set $\Delta^C_\alpha(p^k_\alpha)[s_0] \downarrow$ with $u = \delta_\alpha(p^k_\alpha)$, where we clearly have $M^\alpha_t$ set correctly with $\gamma(a^\alpha_t) < u$. It also follows that the values of $C \upharpoonright u$ and $p^k_\alpha$ did not change between $s_0$ and $s$, so consequently $A \upharpoonright \gamma(a^\alpha_t)[s_0]$ has to change between $s_0$ and $s$ in order for (A5) to hold at $s$. Because $M^\alpha_t$ was set correctly at $s_0$, it follows that $\varphi_B(k)[s]$ is picked fresh and so is larger than $\psi_\alpha(x)[s_0]$ (this is why it is important that $B \upharpoonright \psi_\alpha(x)[s_0]$ did not change between $s_0$ and $s$, so that we can benefit from the $A$-change in between). □

**Lemma 4.9.** Suppose $\varphi_B(x)[s]$ is enumerated into $B$ at stage $s$ (by any action), and for some $\alpha$ and $k > x$, $M^\alpha_t$ is either successful or has a current traced computation at $s$. Then the same action will either

- initialize the node $\alpha$, or
- initialize $M^\alpha_t$, and lower $t^\alpha_x$ for every $x \in zone^\alpha_k$

**Proof.** We proceed by induction on the sequence of actions (or substages) in the construction. If $\varphi_B(x)$ was enumerated by the global actions, then it is clear. So, we assume some $M^\beta_x$ took an injurious action. We must have $\beta$ and $\alpha$ comparable (otherwise it is trivial), and if $\beta \supseteq \alpha$ then it is easy. If $\beta \subset \alpha$ then one can verify that $M^\beta_k$ would be in the injury set of $M^\beta_x$ when $\beta$ acted. The only case left is $\alpha^- n \subseteq \beta$ for some $n$. We show this case is not possible.

At stage $s$ we have $M^\beta_k$ is either successful or has a current traced computation. Clearly we have $k > x > n$. When $\alpha$ was visited earlier in the stage $s$, we had some $y \in zone^\alpha_k[s]$ where $\Psi^B_\alpha(y)$ has not persisted for two visits to $\alpha$. If $M^\beta_k$ has a current traced computation at $s$, then $M^\beta_k$ would have received attention under (A5) and enumerated $\Psi^B_\alpha(z)[s_0]$ into $T^\alpha_x$ at some stage $s_0 \leq s$. Furthermore between $s_0$ and $s$, $t^\alpha_x$ is not lowered. On the other hand if $M^\beta_k$ is successful at $s$, then it also receives attention under (A5) at some stage $s_0 \leq s$. We claim that in either case, we have $t^\alpha_y[s_0] \leq k$.

In the latter case, this follows because of Fact 4.3 and the fact that $k > n$. In the former case, we cannot apply Fact 4.3 directly because $M^\alpha_t$ might be initialized between $s_0$ and $s$. However if $z < y$ then $k = t^\alpha_y[s] \leq t^\alpha_y[s] = n$ is a contradiction, and on the other hand if $z \geq y$ then $k = t^\alpha_y[s_0] \geq t^\alpha_y[s_0] > k$.

In that case it follows by Fact 4.3 and by Lemma 4.6 that $\Psi^B_\alpha(y)[s_0] \downarrow$ and at $s_0$ it had already persisted for two visits to $\alpha$. Therefore the visit to $\alpha$ at $s_0$ and the visit to $\beta$ at $s$ cannot take place
in the same stage. Therefore $B \upharpoonright \psi_\alpha(y)[s_0]$ has to change at some action strictly between $s_0$ and $s$. By Lemma 4.8 this has to be $\varphi_B(x')$ for some $x' < k$. Apply induction hypothesis to get a final contradiction, and hence we conclude that the case $\alpha \subset \infty \beta$ is also not possible. \hfill $\Box$

**Lemma 4.10.** For any $\beta$ and $e$, if $M^\beta_e$ is initialized finitely often, then it only picks finitely many agitators $ag^\beta_e$.

*Proof.* Suppose $t_0$ is a stage after which $M^\beta_e$ is never initialized. Then $p = \lim p^\beta_e$ must exist. Since after $t_0$, we only pick a new agitator if (A6) applies, it follows that we may assume that there are infinitely many stages $t_1 < t_2 < \cdots$ larger than $t_0$ such that $M^\beta_e$ receives attention under (A6) at all of these stages. When $M^\beta_e$ next receives attention after each $t_n$, we must have $M^\beta_e$ ready (otherwise it is not hard to see by Lemma 4.7 that (A5) will apply and break the cycle), which means that $\Delta^\beta_e(p)$ receives a fresh axiom between each $t_n$ and $t_{n+1}$. This is contrary to the fact that $f(p)$ reaches a limit. \hfill $\Box$

**Lemma 4.11.** For each $e$, the following are true:

(i) $\varphi(e)$ is moved finitely often,

(ii) there is a stage after which no module $M^\alpha_k$ for any $\alpha$ and $k \leq e$ is injurious.

*Proof.* We prove (i) and (ii) simultaneously by induction on $e$. Let $s_0 > e$ be large enough such that (i) and (ii) holds for all $k < e$, and also the global actions for coding has stopped acting for $\varphi(0), \cdots, \varphi(e)$. That is, the global requirements no longer enumerate $\varphi(0), \cdots, \varphi(e)$. Note that we are not assuming that (i) and (ii) holds for all $k < e$; instead we only require that the global actions enumerate $\varphi(e)$ into $B$ finitely often. Let $\alpha$ be the leftmost node such that $|\alpha| = e - 1$ and $\alpha$ is visited on or after stage $e$ (assume $s_0$ large enough such that this happens before $s_0$). Thus the only modules $M^\alpha_k$ which can receive attention after stage $s_0$, will belong to the nodes $\beta \subset \alpha$, because modules to the right of $\alpha$ are never active after stage $s_0$. We proceed in a series of steps:

**Claim 1:** if $\beta_0 \subset_f \beta_1 \subset \alpha$, then no $\beta_0$-module can take an action which initializes $M^\beta_1$ after stage $s_0$. The only $\beta_0$-module which can do that after $s_0$ is $M^\beta_0$, and so if $\beta_0 \subset_f \beta_1$, then we must have $n \leq e$ (otherwise it is trivial). Let $t_1 > s_0$ be a stage where $M^\beta_0$ acts and initializes $M^\beta_1$. Note that we only need to show that there are finitely many of such stages $t_1$ (since we can choose $s_0$ large enough for the rest of the lemma). Clearly $M^\beta_1$ is in the injury set of $M^\beta_0$ at $t_1$, and consequently we have $M^\beta_1$ is either *successful* at $t_1$, or else it has a current traced computation at $t_1$. In any case we can let $t_0 < t_1$ be the stage where $M^\beta_1$ receives attention under (A5), and sets things up for $t_1$. Note that at $t_1$, we will also have $M^\beta_1$ initialized and every $x \in \text{zone}^{\beta_1}_{t+1}$ gets lowered, but the latter does not happen between $t_0$ and $t_1$. Therefore, if there are infinitely many such stages $t_1$, we may assume that $t_0$ is large (enough for our purpose) as well. At $t_0$, we have $M^\beta_0$ is *successful* (since $M^\beta_1$ is active), and hence between $t_0$ and $t_1$, the module $M^\beta_0$ has to be initialized. Since $t_0$ is large, the module $M^\beta_0$ has to be initialized by the actions of a third module $M^\sigma_0$ for some $\sigma$, which took an injurious action. By Lemma 4.9 this is impossible, by the choice of $t_0$. This contradiction shows that $M^\beta_0$ has to be *successful* at $t_1$ still, and so cannot be injurious towards $M^\beta_1$.
Note that a $\beta$-module can receive attention infinitely often, in which case $\Psi_{\beta}^C$ is not total. However, a module cannot be infinitely injurious; this is important because we want each module to have a finite effect on the rest of the construction:

Claim 2: for any $\beta$, if $M^\beta_e$ is initialized finitely often, then it can be injurious only finitely often.

This follows directly from Lemma 4.10.

Claim 3: for any $\beta$ such that $\beta \subseteq \alpha$, we have $M^\beta_e$ is initialized finitely often. Start with the minimal such node $\beta \subseteq \alpha$, and work downwards inductively by applying Claims 1 and 2.

Claim 4: if $\beta \subseteq \alpha$, then $M^\beta_e$ is also initialized finitely often. This time round, start with the maximal such node $\beta \subseteq \alpha$, and work upwards inductively by applying Claims 1, 2 and 3.

The claims above prove (ii) for $e$. We now show (i). Assume $s_1 > s_0$ is such that no module $M^\sigma_k$ for any $\sigma$ and $k \leq e$ is injurious anymore. Note that if $\sigma \geq L \alpha$, then $ag^\sigma_e$ is undefined after $s_0$, and never receives a definition again. If $\sigma < L \alpha$ then $ag^\sigma_e$ never receives a new value after $s_0$. Let $\sigma \subseteq \alpha$, and by Lemma 4.10, we have $\lim ag^\sigma_e$ exists. Finally, it is not hard to put the various facts together and conclude that (ii) holds for $e$ as well. \hfill \Box

An immediate corollary to Lemma 4.11 is that $\Phi_{A \oplus B}$ is total and correctly computes $\Psi'$. We define the true path of the construction to be the leftmost path visited infinitely often. Before we can show that the true path exists, we start with a preparatory lemma:

**Lemma 4.12.** Suppose $\alpha$ is visited at $s$ and has just finished acting and has decided to visit the outcome $k_0 \alpha \infty$. Then, it is impossible for $M^\alpha_{k_1}$ to be waiting if $k_1 > k_0$.

**Proof.** Let $s^- \leq s$ be the stage where $M^\alpha_{k_1}$ received attention under (A4) to give its current waiting state. Since outcome $k_0 \alpha \infty$ was played at $s$, it follows by Fact 4.1 that $M^\alpha_{k_0}$ would be able to receive attention at $s^-$ if it required to do so. Hence at $s^-$, $M^\alpha_{k_0}$ cannot be unstarted or successful, and by Lemma 4.7(ii) we have $\Delta^C_{\alpha}(p^\alpha_{k_0})[s^-] \downarrow$. Hence when $\delta_\alpha(p^\alpha_{k_1})$ was set at $s^-$, it must be larger than $\delta_\alpha(p^\alpha_{k_0})[s^-]$. Since $M^\alpha_{k_0}$ is ready at $s$ it follows that $C$ must change below $\delta_\alpha(p^\alpha_{k_1}) < \delta_\alpha(p^\alpha_{k_1})$ between $s^-$ and $s$, a contradiction. \hfill \square

Note that in the above lemma, we do not actually need $\alpha$ to have outcome $k_0 \alpha \infty$ at $s$. All we really require is that $k_0 > s^-$ (the previous visit to $\alpha$) and $M^\alpha_{k_0}$ is active and ready at $s$.

**Lemma 4.13.** The true path of construction exists.

**Proof.** Suppose we have defined the true path up till $\alpha$. Hence $\alpha$ is visited infinitely often, and $\delta_\alpha \leq \alpha$ for finitely many $t$. We want to show that some outcome of $\alpha$ is visited infinitely often. Suppose each outcome is visited finitely often; we want to derive a contradiction to the fact that $C$ is low$_2$. Since $\alpha$ is initialized only finitely often, it follows by Lemma 4.11 that each $\alpha$-module is initialized finitely often. We first show the following:

Claim 1: for almost all $k$, $M^\alpha_k$ eventually becomes active and is active at every $\alpha$-stage after that. Fix $k$ large enough. We may assume by induction hypothesis, that the statement of the claim holds for any $\beta \subset \alpha$. Let $s_0$ be large enough such that all parameters for $M^\beta_j$ for $\beta \subset \alpha$ and $j \leq k$ have settled. We can do this because each $\beta$-module is initialized finitely often, and by Lemma 4.10, we know that $ag^\beta_j$ is never refreshed again. We want to show that (1) to (4) of Definition 4.3 holds for $M^\beta_k$ forever after $s_0$. (1) and (4) are obvious, so we consider the other two. First consider some $\beta^-nf \subseteq \alpha$. 
If $M_n^\beta$ receives attention infinitely often, then it is not hard to see that eventually we must have $M_n^\beta$ only receiving attention under (A4). Note that no $\beta$-module smaller than $M_n^\beta$ can receive attention, so only $M_0^\beta$ can be responsible for enumerating $\Delta_{\psi}^\gamma(y)$-axioms for $y \leq p_n^\beta$. Therefore there must be some $x \in \text{zone}_n^\beta$ (which would have settled) such that $\Psi_B^\beta(x)$ is divergent (because otherwise the variables $z$ and $m$ in (A4) would eventually settle). Consequently we must visit $\beta$ at some stage $t$ in which $\Psi_B^\beta(x)[t]$ has not persisted for two $\beta$-stages, and by Lemma 4.6 $M_n^\beta$ would have to be ready at $t$. It follows that (A2) will apply to $M_n^\beta$ at $t$, a contradiction (to the fact that $\beta f$ is the true outcome of $\beta$) follows by applying the induction hypothesis to $\beta$. Therefore, $M_n^\beta$ only receives attention finitely often, and it follows that eventually, at each time $\beta^-nf$ is visited, some module $M_j^\beta$ for $j \geq n$ would have to receive a state change from waiting to successful, and subsequently stays successful forever (assuming of course, that $j \leq k$). If we wait long enough then $t_n^\beta > k$ at every visit to $\alpha$.

Now consider some $\beta^-n\infty \subseteq \alpha$. We want to show that eventually, every module $M_n^\beta, \ldots, M_k^\beta$ is either successful or is set correctly at every $\alpha$-stage. Any $\beta$-module $M_j^\beta$ for $j \leq k$ which receives attention at an $\alpha$-stage, can only have done so if (A2) applies (use Lemma 4.12 for this, and the fact that $ag_j^\beta$ has settled). If every module $M_j^\beta$ for $n \leq j \leq k$ receives attention at finitely many $\alpha$-stages, then by Lemma 4.5, we clearly have what we want. Therefore, we may assume that $j$ is the least such that $M_j^\beta$ receives attention at infinitely many $\alpha$-stages for some $n \leq j \leq k$. Take a large enough $\alpha$-stage $t$ where $M_j^\beta$ receives attention.

Let $t^- < t$ be the previous visit to $\beta$, and $t' \leq t^-$ be the largest $\beta$-stage where we finished $\beta$'s action with $M_j^\beta$ in state waiting. Furthermore, $t'$ and $t^-$ satisfies the conditions described in (A2) of the construction. Hence at stage $t^* \leq t'$ when we bestowed the state waiting upon $M_j^\beta$, we had set $\delta_{\beta}(p_j^\beta) = \gamma(z) + m$ where $z, m$ are as defined in the construction. We may assume $t^* > s_0$. Since no module smaller than $M_j^\beta$ can be responsible for setting $\Delta_{\beta}$-axioms anymore, it follows by an argument similar as above, that $m$ will reach a limit. Hence at stage $t^*$, there must be some $x \in \text{zone}_{j}^\beta[t^*]$ such that $\varphi_B(k + 1)[t^*] < \psi_{\beta}(x)[t^*]$. Since the computation $\Psi_B^\beta(x)[t^*]$ must be $M_j^\beta$-believable at $t^*$, we can also deduce that every module $M_n^\beta, \ldots, M_k^\beta$ is either successful or is set correctly. This concludes the proof of Claim 1.

By Claim 1 it follows that $p_n = \lim p_n^\alpha$ is defined for almost all $n$. Note that each $\alpha$-module is initialized only finitely often, and receives attention finitely often (by assumption). If $\Psi^B_\alpha$ is not total then one can verify using Lemma 4.12 that $\alpha^-n\infty$ will be visited infinitely often for some $n$, a contradiction. On the other hand suppose that $\Psi^B_\alpha$ is total. If $n$ is large enough, what is the final state of the module $M_n^\alpha$? Clearly it cannot be unstarted, and cannot be ready since $\Psi^B_\alpha$ is total. If it is waiting for infinitely many $n$, then $\Delta^C_\alpha(p_n)$ will be defined forever at infinitely many $n$. Consequently $\Delta^C_\alpha$ is defined on almost all inputs, which implies (by low2-ness) that $f(p_n) > \Delta^C_\alpha(p_n)$ for almost all $n$. This in turn implies that for all large enough $n$, $M_n^\beta$ has to become successful and stay in that state forever. Hence some finitary outcome of $\beta$ will be visited infinitely often (depending on where the consecutive run of successful modules start), contrary to assumption. Therefore, there is a leftmost outcome of $\alpha$ visited infinitely often. □

Lemma 4.14. Along the true path of construction, the requirements succeed.
Proof. Let \( \alpha \) be assigned requirement \( R_e \), along the true path of construction, such that \( \Psi^B \) is total. We show that the version of \( \{ T^\alpha_x \} \) built after the final initialization of \( \alpha \) at \( s_0 \) traces \( \Psi^B_e(x) \) correctly for almost all \( x \).

Claim 1: for any \( s \geq s_0 \) and every \( x \), we have \( x + 1 - |T^\alpha_x| \geq t^\alpha_x \). Fix two stages \( s_2 > s_1 > s_0 \) such that two different elements are enumerated into \( T^\alpha_x \) at \( s_1 \) and \( s_2 \). We will be done if we can show that \( t^\alpha_x \) is decreased between \( s_1 \) and \( s_2 \). At \( s_1 \) it must be the case that \( M^\alpha_m \) receives attention under (A5), enumerating \( \Psi^B(x)[s_1] \) into \( T^\alpha_x \) where \( m = t^\alpha_x[s_1] \). By Lemma 4.8, the change in \( B \) \( \psi_\alpha(x)[s_1] \) has to be due to some \( \varphi_B(z) \) entering \( B \) for some \( z < m \). By Lemma 4.9, \( t^\alpha_x \) is certainly lowered, which proves the claim.

From Claim 1 it follows that \( |T^\alpha_x| \leq x + 1 \), because \( \text{zone}^\alpha_0 \) is never lowered. It remains to show that almost every \( \Psi^B_e(x) \) is traced. Firstly, we argue that the true outcome of \( \alpha \) cannot be infinitary. Suppose not, and the true outcome of \( \alpha \) is \( \text{non} \infty \) for some \( n \). Let \( s_1 \) be large enough, so that no module \( M_j^\alpha \) is initialized, and \( \varphi(j) \) has settled, for every \( j \leq n + 1 \). Thus \( \text{zone}^\alpha_j \) has settled for all \( j \leq n \), and we also assume that \( \Psi^B_e(x)[s_1] \) on the correct use for every \( x \) in the final \( \cup_{j \leq n} \text{zone}^\alpha_j \). By Claim 1 of Lemma 4.13 (applied to \( \alpha \Rightarrow \text{non} \infty \)) we may as well assume that all the \( \Psi^B_e(x) \)-computations above are \( M^\alpha_n \)-believable. Observe that \( M^\alpha_n \) cannot remain \( \text{ready} \) forever after \( s_1 \), because otherwise (A4) will eventually apply for \( M^\alpha_n \), and \( M^\alpha_n \) would have received attention at the next stage where \( \alpha \Rightarrow \text{non} \infty \) is visited. So, \( M^\alpha_n \) eventually becomes \( \text{waiting} \) and enumerates some computation \( \Delta^C_\beta(p^\alpha_n) \), with a use that we can assume does not change anymore. This is a contradiction because \( M^\alpha_n \) has to get back to state \( \text{ready} \) in time for the next \( \alpha \Rightarrow \text{non} \infty \) visit.

Thus we let the true outcome of \( \alpha \) be \( \text{nf} \) for some \( n \). Hence \( l^\alpha_n \rightarrow \infty \), because \( M^\alpha_j \Rightarrow \text{nf} \) eventually becomes active for all large enough \( j > n \), following from Claim 1 of Lemma 4.13. Again wait for a stage \( s_1 \) large enough so that all relevant activity in \( M^\alpha_0, \ldots, M^\alpha_{n+1} \) has ceased. Let \( x_0 = \min \cup_{j \geq n+1} \text{zone}^\alpha_j[s_1] \). We claim that \( q = \Psi^B_e(x) \) is traced in \( T^\alpha_x \) for all \( x > x_0 \). Note that \( t^\alpha_x \geq n + 1 \) after \( s_1 \), so let \( m \geq n + 1 \) be the final value attained by \( t^\alpha_x \). Let \( s \) be the time where an action was taken to make \( t^\alpha_x = m \) (note that \( s \) may be smaller than \( s_1 \)). If the action was an initialization to \( \alpha \), then \( M^\alpha_m \) is initialized as well. Otherwise the action was to lower \( t^\alpha_x \) from \( m + 1 \) to \( m \). By Fact 4.3 we know that \( M^\beta_n \) will have to become \( \text{successful} \) after stage \( s \), which means that at some point \( s' > s \), we have (A5) applies for \( M^\beta_n \) and \( \Psi^B_\alpha(x)[s'] \) enumerated into \( T^\alpha_x \). By Lemmas 4.8 and 4.9, we have \( \Psi^B_\alpha(x)[s'] = p \).

\[ \square \]

References


School of Mathematics, Statistics and Operation Research, Victoria University of Wellington, Wellington, New Zealand

*E-mail address*: Noam.Greenberg@mcs.vuw.ac.nz

Department of Mathematics, University of Wisconsin, Madison, WI 53706-1388, USA

*E-mail address*: selwynng@math.wisc.edu

Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371, Singapore

*E-mail address*: guohua@ntu.edu.sg