

# A NONLOW<sub>2</sub> C.E. DEGREE WHICH BOUNDS NO DIAMOND BASES

ANGSHENG LI, THEODORE A. SLAMAN AND YUE YANG

ABSTRACT. A computably enumerable (c.e.) Turing degree is a diamond base if and only if it is the bottom of a diamond of c.e. degrees with top  $\mathbf{0}'$ . Cooper and Li [3] showed that no low<sub>2</sub> c.e. degree can bound a diamond bases. In the present paper, we show that there exists a nonlow<sub>2</sub> c.e. degree which does not bound a diamond base. Thus, we refute an attractive natural attempt to define the jump class low<sub>2</sub>.

## 1. INTRODUCTION

Let  $\mathcal{R}$  denote the structure of computably enumerable (c.e.) degrees with Turing reducibility  $\leq_T$ . One of the central questions yet to be resolved about  $\mathcal{R}$  is to characterize those relations on the c.e. degrees which are first order definable in  $\mathcal{R}$ . The problem remains open, but some significant partial results are known. Recently, Nies, Shore and Slaman [6] showed that, except for *low*, all of the jump classes are definable in  $\mathcal{R}$ . Recall that a c.e. degree  $\mathbf{a}$  is *low<sub>n</sub>* if  $\mathbf{a}^{(n)} = \mathbf{0}^{(n)}$ ;  $\mathbf{a}$  is *high<sub>n</sub>* if  $\mathbf{a}^{(n)} = \mathbf{0}^{(n+1)}$  (for  $n \geq 1$ ). If  $n = 1$ , we call  $\mathbf{a}$  *high* and *low* respectively. Let  $\mathbf{H}_n$  and  $\mathbf{L}_n$  denote the jump classes of all high<sub>n</sub> and low<sub>n</sub> c.e. degrees respectively. The result by Nies, Shore and Slaman can be stated as follows.

**Theorem 1.1.** *For any  $n \geq 1$ ,  $\mathbf{H}_n$  and  $\mathbf{L}_{n+1}$  are definable in  $\mathcal{R}$ .*

However, the proof of Theorem 1.1 uses first order representations of standard models of arithmetic within  $\mathcal{R}$ . There is still a special interest in obtaining more degree theoretic definitions, especially ones with lower logical complexity (i.e. the so called *natural* definitions), as stated in Shore [7]:

**Question 1.2.** *Are any of the jump classes  $\mathbf{H}_n$  and  $\mathbf{L}_{n+1}$  naturally definable in  $\mathcal{R}$ ? In particular, what about  $\mathbf{L}_2$  and  $\mathbf{H}_1$  where we have the*

---

1991 *Mathematics Subject Classification.* 03D25.

Li was partially supported by EPSRC Research Grant “Turing Definability”, No. GR/M 91419 (UK), by NSF grant No. 69973048, by NSF major grant No. 19931020 (P. R. CHINA), and by a visiting research fellowship of the National University of Singapore (NUS). Slaman was partially supported by National Science Foundation Grant DMS-9988644 and also by a visiting professorship at NUS. All of the authors wish to thank Lei Qian for the discussions they had with him while working on this paper.

*strongest connections with rates of growth and significant techniques already developed to exploit the known characterizations of these classes.*

Clearly, finding a natural definition of any of the jump classes would illuminate the role of the jump in  $\mathcal{R}$ . It could be even more interesting to show that there is no logically simple formula which defines one of the jump classes in  $\mathcal{R}$ . In either case, we need a clearer understanding of the order theoretic properties of c.e. degrees and the relationships between these properties and the jump.

One canonical starting point in the order theoretic investigation of  $\mathcal{R}$  is to consider the existence of least upper bounds and greatest lower bounds for sets of degrees. As is well known, every pair of c.e. degrees  $\mathbf{a}$  and  $\mathbf{b}$  has a least upper bound ( $\mathbf{a} \vee \mathbf{b}$ ) just as every pair of subsets of natural numbers has a computable join. By a theorem of Lachlan [5] and Yates [10], there is a pair of c.e. degrees with a greatest element ( $\mathbf{a} \wedge \mathbf{b}$ ) below both. In fact, it is possible for  $\mathbf{a} \wedge \mathbf{b} = \mathbf{0}$ . But Lachlan and Yates also proved that there are c.e. degrees  $\mathbf{a}, \mathbf{b} > \mathbf{0}$  such that there is no greatest  $\mathbf{x}$  such that  $\mathbf{a}, \mathbf{b} \geq \mathbf{x}$ , so  $\mathcal{R}$  is not a lattice.

Lachlan was the first to investigate the interactions between meet and the jump. He showed that there are high c.e. degrees  $\mathbf{a}$  and  $\mathbf{b}$  such that  $\mathbf{a} \wedge \mathbf{b} = \mathbf{0}$ . So there is no direct connection between the jump of  $\mathbf{a}$  and  $\mathbf{b}$  and the fact that  $\mathbf{a} \wedge \mathbf{b} = \mathbf{0}$ . We note that there are other dynamic consequences for  $\mathbf{a}$  and  $\mathbf{b}$  when  $\mathbf{a} \wedge \mathbf{b} = \mathbf{0}$ ; see Ambos-Spies, Jockusch et al [2].

Lachlan did find a connection between the jump and properties involving  $\vee$  and  $\wedge$  in  $\mathcal{R}$ .

**Definition 1.3.** *We say that a c.e. degree  $\mathbf{a}$  is a diamond base, if there are c.e. degrees  $\mathbf{a}_0, \mathbf{a}_1$  such that  $\mathbf{a} < \mathbf{a}_0, \mathbf{a}_1 < \mathbf{0}'$ ,  $\mathbf{a}_0 \vee \mathbf{a}_1 = \mathbf{0}'$  and  $\mathbf{a}_0 \wedge \mathbf{a}_1 = \mathbf{a}$ . We use  $\mathbf{DB}$  to denote the set of all diamond bases.*

Lachlan showed that  $\mathbf{0} \notin \mathbf{DB}$ .

**Theorem 1.4** (Lachlan Nondiamond Theorem). *There are no c.e. degrees  $\mathbf{a}, \mathbf{b}$  such that  $\mathbf{0} < \mathbf{a} < \mathbf{0}'$ ,  $\mathbf{a} \vee \mathbf{b} = \mathbf{0}'$  and  $\mathbf{a} \wedge \mathbf{b} = \mathbf{0}$ .*

More importantly to this discussion, Lachlan also observed that  $\mathbf{DB} \cap \mathbf{L}_1 = \emptyset$  (see Ambos-Spies [1]). Cooper and Li [3] further improve the result.

**Theorem 1.5** (Cooper and Li). (i)  $\mathbf{DB} \cap \mathbf{L}_2 = \emptyset$ , that is, every  $\text{low}_2$  c.e. degree bounds no diamond bases.  
(ii)  $\mathbf{DB} \cap \mathbf{L}_3 \neq \emptyset$ .

Could Theorem 1.5 be half of a proof that  $\text{low}_2$  is definable, say by  $\mathbf{x} \in \mathbf{DB}$  if and only if  $\mathbf{x}$  does not bound a diamond base? In Theorem 1.6, we show that the answer is no.

**Theorem 1.6.** *There exists a non $\text{low}_2$  c.e. degree  $\mathbf{a}$ , which bounds no diamond bases.*

The remaining sections are devoted to the proof of Theorem 1.6. In Section 2, we introduce the requirements and describe the naive version of strategies; in Section 3, we study the conflicts between requirements and modify the strategies; in Section 4, we define the priority tree  $T$  and describe the full construction, and finally in Section 5, we verify the correctness of the construction.

Notation and terminology are standard and generally follow Soare [1987]. The basic knowledge of tree constructions in computability theory is assumed. A *Turing functional*  $\Phi$  is a c.e. set of consistent quadruples  $\langle x, y, P, N \rangle$  (called *axioms*) where  $x$  and  $y$  are numbers and  $P$  and  $N$  are (codes of) finite sets. We write  $\Phi(A; x) = y$  if there are finite sets  $P$  and  $N$  such that  $\langle x, y, P, N \rangle \in \Phi$  and  $P \subset A$  and  $N \cap A = \emptyset$ . We use the corresponding lower case letter  $\varphi(A; x)$  to denote the use function for  $\Phi(A; x)$ . The words “define  $\Lambda(A; x) = y$  with use  $\lambda$ ” means that we enumerate the quadruple  $\langle x, y, A \upharpoonright (\lambda + 1), \bar{A} \upharpoonright (\lambda + 1) \rangle$  into  $\Lambda$ . If the Turing functional  $\Phi$  applies to the join of two sets  $X$  and  $Y$ , we will write  $\Phi(XY)$  instead of  $\Phi(X \oplus Y)$ . During the course of a construction, whenever we define a parameter as *fresh*, we mean that it is defined as the least natural number which is greater than any number mentioned so far.

## 2. THE REQUIREMENTS AND BASIC MODULES

Fix a complete c.e. set  $K_0$ . We build a c.e. set  $A$  such that:

- (1)  $A$  is nonlow<sub>2</sub>;
- (2)  $A$  is noncuppable; and
- (3) for any c.e. sets  $X$  and  $Y$ , if  $X \not\leq_T A$ ,  $Y \not\leq_T A$  and  $X \oplus Y \equiv_T K_0$ , then there is a c.e. set  $Z$  such that  $Z \leq_T X$ ,  $Z \leq_T Y$  and  $Z \not\leq_T A$ .

Notice that conditions (2) and (3) imply that  $A$  does not bound any diamond base  $D$ : For any possible candidates  $X, Y$  such that  $X, Y, D$  and  $K_0$  form a diamond, if neither  $X$  nor  $Y$  is below  $A$ , then by (3)  $D \not\leq_T A$ , which says that the base is not below  $A$ ; if one of  $X$  or  $Y$  is below  $A$ , then by (2) the other is complete, which says that it is not a proper diamond.

We now look at each individual requirement. In Section 4, we will define a priority tree  $T$ , on which each node is labelled with a requirement. The strategies for a requirement  $O$  described below is corresponding to the  $\alpha$ -version of  $O$ , where  $\alpha$  is a node on  $T$  labelled  $O$ .

**2.1. Description of nonlow<sub>2</sub> requirements.** We use the following typical requirements to make  $A$  not low<sub>2</sub>, (see Cooper, Li and Yi [4] or Shore and Yang [8]). Fix an effective enumeration  $U_e$  of all  $\Sigma_3$  sets

$$U_e = \{x : \exists m \forall y \exists z (\theta_e(x, m, y, z) \downarrow)\}.$$

We build a single functional  $\Lambda(A; x, m)$  and guarantee (via requirements  $P_e$ ) that for each  $e$ , there is an  $x$  such that the set

$$V_x^A = \{m : \Lambda(A; x, m) \downarrow\}$$

is infinite if and only if  $x \notin U_e$ . The point here is that  $\{x : V_x^A \text{ is infinite}\}$  is a  $\Pi_2^A$  set and these requirements will guarantee that it is not a  $\Sigma_3$  set. Thus  $A$  cannot be  $\text{low}_2$ .

- $P_e$ : There is an  $x$  such that  $V_x^A = \{m : \Lambda(A; x, m) \downarrow\}$  is infinite if and only if  $x \notin U_e$ .

$P_e$  begins by choosing a witness  $x$  for diagonalization. As time goes by and  $P_e$  is accessible, it enumerates axioms for  $\Lambda(A; x, m)$  for each  $m$ .  $P_e$  has only one outcome  $\infty$ .  $P_e$  has subrequirements  $Q_{e,m}$  for each  $m$ . They test for  $m$  being a witness to the  $\Sigma_3$  fact that  $x \in U_e$  by looking for verifications of the associated  $\Pi_2$  fact  $\forall y \exists z (\theta_e(x, m, y, z)) \downarrow$  for successive  $y$ 's. The outcomes of such a subrequirement are either  $\infty$  or 0 depending on whether the  $\Pi_2$  fact is true or not. Each time it finds a witness for a new  $y$ , it tries to make  $\Lambda(A; x, n) \uparrow$  for each  $n \geq k$  for some fixed  $k$  by enumerating  $\lambda(A; x, k)$  into  $A$ . We will call the number  $k$  the *killing point*, and the choice of  $k$  depends on  $Q_{e,m}$ 's priority.

**2.2. Description of noncuppable requirements.** The noncuppable requirements guarantee that for all c.e. set  $W$ ,  $A \oplus W$  is complete implies that  $W$  is complete. Fix an effective enumeration of Turing functionals  $\Gamma_e$  ( $e \in \omega$ ). For each pair of natural numbers  $e_1$  and  $e_2$  and its code  $e = \langle e_1, e_2 \rangle$ , we build a Turing functional  $\Delta_e$  such that if  $\Gamma_{e_1}(AW_{e_2}) = D$ , then  $\Delta_e(W_{e_2}) = K_0$ , where  $D$  is an auxiliary set built by us. For simplicity, we would like to view  $K_0$  as a subset of  $D$ , so that any number enumerated into  $K_0$  is enumerated into  $D$  automatically. Thus let us assume that  $K_0$  ( $D$ , respectively) is a subset of even (odd, respectively) numbers and  $K$  is the (disjoint) union of  $K_0$  and  $D$ . The noncuppable requirements  $N_e$  are as follows.

- $N_e$ : If  $\Gamma_{e_1}(AW_{e_2}) = K$ , then  $\Delta_e(W_{e_2}) = K_0$ .

From now on, when we define the value  $\Delta_e(W_{e_2}; p)$ , we always assume that  $p$  is an even number; we may assume that if  $q$  is an odd number, then  $\Delta_e(W_{e_2}; q) = 0$  with empty use. We also assume that the candidates targeting  $D$  are chosen from odd numbers.

In order to coordinate with other requirements, in particular, to distribute a potentially infinitary restraint into infinitely many finitary ones, we have subrequirements  $M_{e,p}$  ( $p$  is an even natural number) working for  $N_e$ , each  $M_{e,p}$  is responsible for defining  $\Delta_e(W_{e_2}; p)$ . From now on, the letter  $p$  is reserved for even numbers. The concern of consistency of  $\Delta_e$  will be discussed later.

- $M_{e,p}$ : If  $\Gamma_{e_1}(AW_{e_2}) = K$ , then  $\Delta_e(W_{e_2}; p)$  is defined and equal to  $\Gamma_{e_1}(AW_{e_2}; p)$ .

Let  $\alpha$  be a node labelled  $N_e$ . The ( $\alpha$ -th version of) noncuppable strategy  $N_e$  works as follows. We omit the index  $e$  during the discussion if there is no confusion. Define the length of agreement function  $l(\alpha, s)$  between  $\Gamma(AW)$  and  $K$  as usual,

$$l(\alpha, s) = \mu y(\Gamma(AW; y) \neq K(y)[s]).$$

We say that  $s$  is  $\alpha$ -*expansionary* if  $\alpha$  is accessible at stage  $s$  and  $l(\alpha, s) > l(\alpha, t)$  for all  $t < s$  at which  $\alpha$  was accessible.  $\alpha$  has two possible outcomes:  $\infty$  for infinitely many  $\alpha$ -expansionary stages; and 0 for finitely many ones.

About the outcome  $\infty$ ,  $\alpha$  has substrategies  $M_{e,p}$ . The naive version of strategy for  $M_{e,p}$  works as follows. We first check if the use  $\gamma(AW; p)$  has been moved since the stage at which it was accessible for the last time. If yes, then  $M_{e,p}$  has outcome  $\infty$ , because it indicates that  $\Gamma(AW; p)$  is partial; otherwise, we define  $\Delta(W; p) = \Gamma(AW; p)$  with use  $\gamma(AW; p)$  and restrain  $A$  up to  $\gamma(AW; p)$ . Restraining  $A$  allows us to tie  $\Delta(W)$  with  $\Gamma(AW)$  so that we can ignore the changes in  $K_0$ ; in other words, if  $\Gamma(AW; p) = K_0(p)$  then  $\Delta(W; p) = K_0(p)$ : Suppose that the number  $p$  enters  $K_0$  at some stage  $t > s$ , where  $s$  is the stage at which  $\Delta(W; p)$  is defined. If  $W$  does not change below  $\gamma(AW; p)$ , then the restraint on  $A$  ensures that  $\Gamma(AW) \neq K$ , so that  $N$  is satisfied by the failure of premise. On the other hand if  $W$  changes below  $\gamma(AW; p)$ , then  $\Delta(W; p)$  will be redefined too, so that the equality  $\Delta(W; p) = \Gamma(AW; p)$  can be preserved.

If  $M_{e,p}$  has outcome  $\infty$  then we have a global win for  $N_e$ , because  $\gamma(AW; p)$  goes to infinity. Thus above the outcome  $\infty$  of  $M_{e,p}$ , we do not need to consider other subrequirements of  $N_e$ . If  $M_{e,p}$  has outcome 0, then we move on to the next  $M$ -subrequirement.

We now discuss the consistency of the Turing functional  $\Delta$ . Obviously, above  $\alpha$  there can only be one definition of  $\Delta(W; p)$ , in other words, after  $\Delta(W; p)$  is defined, we cannot have another definition of  $\Delta(W; p)$  unless  $W$  changes below  $\gamma(AW; p)$ . On the other hand, we must keep the restraint  $\gamma(AW; p)$  on  $A$  as long as  $\Delta(W; p)$  is defined. Clearly there are conflicts with other positive requirements, because those  $\Delta(W; p)$  can be defined at a node to the right of the true path; it is impossible to obey all restraints imposed by nodes to the right.

To overcome this difficulty, we will ensure that if a node  $\tau$  above  $\alpha$  is accessible at stage  $s$ , then every  $\Delta(W; p)$ , which was defined by a node strictly to the right of  $\tau$  at some stage  $t < s$ , is undefined at stage  $s$ . To achieve this, we must be able to make  $\Delta(W; p)$  undefined. The plan is to force a  $W$ -change by attacking the  $D$ -side of  $K$ . If  $W$  does not respond, then  $\Gamma(AW) \neq K$  and we have an easy win for  $N$ . If  $W$  responds then  $\Delta(W; p)$  becomes undefined.

To make  $\delta(W; p)$  undefined, we need to force a  $W$ -change on some number smaller than  $\delta(W; p)$ . The plan is to select a number (called a

*flip point*)  $d$  not yet in  $D$ , and force  $W$  to change by putting  $d$  in  $D$ . The necessary condition on  $d$  is that  $\gamma(AW; d) < \delta(W; p)$ . The modified  $M_{e,p}$ -strategy is as follows. First we pick a fresh flip point  $d$ , keep it outside  $D$ . We delay the definition of  $\Delta(W; p)$  until  $\Gamma(AW; d)$  is defined; then define  $\Delta(W; p) = \Gamma(AW; p)$  with use  $\gamma(AW; d)$  and restrain  $A$  up to  $\gamma(AW; d)$ . When we need to make  $\Delta(W; p)$  undefined, we put  $d$  into  $D$  and wait for a stage at which  $W$  changes below  $\gamma(AW; d)$ . Under the assumption that  $\Gamma(AW; d) = D(d)$ ,  $W$  must change below  $\gamma(AW; d)$ . Since  $\gamma(AW; d) = \delta(W; p)$ ,  $\Delta(W; p)$  becomes undefined.

We use the following example to illustrate the point. Let us consider a node  $\tau$  above  $\alpha \hat{\infty}$  which has outcomes  $o_1 <_L o_2$ . At current stage  $s$ ,  $\tau$  is accessible, and  $\tau$  would have outcome  $\tau \hat{o}_1$ , thus make  $\tau \hat{o}_1$  accessible. However there are  $\Delta(W; p)$ 's which have been defined by some nodes extending  $\tau \hat{o}_2$ . We will refer this situation as “ $\alpha$  stops  $\tau$  having outcome  $o_1$ ”.

Let us first assume that  $\alpha$  is the only node which stops  $\tau$  having outcome  $o_1$ . At the node  $\tau$ , before we declare that  $\tau \hat{o}_1$  is accessible, we pick the smallest  $p$  such that  $\Delta(W; p)$  is defined at some node extending  $\tau \hat{o}_2$ , let  $d$  be the flip point for  $p$ . Put  $d$  into  $D$  and wait for the next  $\alpha$ -expansionary stage  $t > s$ . At stage  $t$ , we can declare that  $\tau \hat{o}_1$  is accessible.

It should be pointed out that when we see a  $W$ -change at  $\alpha$ , we need to act immediately without delay, otherwise, more axioms might be enumerated into  $\Delta$  at some node to the right of  $\tau \hat{o}_1$  before  $\tau \hat{o}_1$  is accessible again. Thus when we put  $d$  into  $D$  at  $\tau$ , we set up a link  $(\alpha, \tau)$ , at the next  $\alpha$ -expansionary stage, we travel the link to  $\tau$  and make  $\tau \hat{o}_1$  accessible.

If there are more than one  $N$ -requirements below  $\tau$ , then we have to deal with them one by one. For example, assume  $N_0$  and  $N_1$  are the only two  $N$ -requirements which are assigned to  $\alpha_0$  and  $\alpha_1$  respectively such that

$$\alpha_0 \hat{\infty} \subseteq \alpha_1 \hat{\infty} \subseteq \tau.$$

Then we deal with  $N_1$  by putting the corresponding flip point  $d_1$  into  $D$  and setting up a link  $(\alpha_1, \tau)$ . At the next  $\alpha_1$ -expansionary stage, we travel the link to  $\tau$  and cancel the link  $(\alpha_1, \tau)$ . Next we deal with  $N_0$  by putting its flip point  $d_0$  into  $D$  and setting up a new link  $(\alpha_0, \tau)$ . As long as the link exists,  $N_1$  is bypassed so that we do not define more  $\Delta_1$  axioms for  $N_1$ . At the next  $\alpha_0$ -expansionary, we travel through the link  $(\alpha_0, \tau)$  to  $\tau$  and  $\tau \hat{o}_1$  can be accessible now.

**2.3. Description of nondiamond requirements.** We modify slightly the requirements used in Cooper and Li [3], which is essentially a tree version of the proof presented in Soare [9]. For each pair of c.e. sets  $X$  and  $Y$  which are candidates of sides of a diamond with bottom below  $A$ , we build two c.e. sets  $B$  and  $C$ , such that if neither  $X$  nor  $Y$  is

below  $A$ , then one of  $B$  and  $C$  is below both  $X$  and  $Y$  but not below  $A$ .

Fix effective enumerations of c.e. sets  $X_e$  and  $Y_e$  ( $e \in \omega$ ) and effective enumerations of Turing functionals  $\Phi_e$  and  $\Psi_e$  ( $e \in \omega$ ). We have the following requirements  $R_e$  and their subrequirements  $S_{e,i,j}$  for  $e, i, j \in \omega$ .

- $R_e$ : If  $E = \Phi_{e_0}(X_{e_1}Y_{e_2})$  then  $B_e \leq_T X_{e_1}, Y_{e_2}$ ,  $C_e \leq_T X_{e_1}, Y_{e_2}$  and for all  $i, j \in \omega$ ,  $S_{e,i,j}$  is satisfied (in the sense explained below), where  $e$  is the code of the triple  $\langle e_0, e_1, e_2 \rangle$  and  $E$  is an auxiliary c.e. set built by us.
- $S_{e,i,j}$ :  $B_e \neq \Psi_i(A)$  or  $C_e \neq \Psi_j(A)$  or  $X_{e_1} \leq_T A$  or  $Y_{e_2} \leq_T A$  or  $(\exists n)[\Phi_{e_0}(X_{e_1}Y_{e_2}; n) \uparrow]$ .

Fix an ordering  $\prec$  on pairs. We say that  $S_{e,i,j}$  is satisfied if either

- (a) for some  $(i', j') \prec (i, j)$ ,  $S_{e,i',j'}$  is satisfied (in the usual sense) by winning either  $X_{e_1} \leq_T A$  or  $Y_{e_2} \leq_T A$  or  $(\exists n)[\Phi_{e_0}(X_{e_1}Y_{e_2}; n) \uparrow]$  (in other words, we have had a global win for  $R_e$  so it is not necessary to consider any further  $S_{e,i,j}$ ); or
- (b) there is  $j' < j$  such that  $S_{e,i,j'}$  is satisfied by winning  $B_e \neq \Psi_i(A)$  (in other words, we made a progress on  $B_e$ ); or
- (c)  $S_{e,i,j}$  is satisfied by winning  $C_e \neq \Psi_j(A)$ .

The ( $\alpha$ -th version of) strategy for  $R_e$  is as follows. We drop the index  $e$  in the discussion if there is no confusion.  $\alpha$  measures the length of agreement  $l(\alpha, s)$  between  $E$  and  $\Phi_{e_0}(X_{e_1}Y_{e_2})$ , where

$$l(\alpha, s) = \mu y (\Phi_{e_0}(X_{e_1}Y_{e_2}; y) \neq E(y)[s]).$$

We define  $\alpha$ -expansionary stages as usual. At  $\alpha$  we build the sets  $B$  and  $C$  computable from  $X$  and  $Y$  by permitting argument.  $\alpha$  has two possible outcomes:  $\infty$  for infinitely many  $\alpha$ -expansionary stages and  $0$  for finitely ones. The main jobs are distributed to the subrequirements  $S_{i,j}$ .

The strategy for subrequirements  $S_{i,j}$  working for  $R$  is as follows. The basic task for  $S_{i,j}$  is to search for an appropriate pair of numbers  $(u, v)$  such that  $u$  is a diagonalization witness targeting  $B$  for  $\Psi_i(A; u) \neq B(u)$  and  $v$  targeting  $C$  for  $\Psi_j(A; v) \neq C(v)$ . The main concern is the permitting which makes  $B$  and  $C$  computable from both  $X$  and  $Y$ .

First we look at the initial set up and some easy cases. To begin with, we pick a fresh *switch point*  $n$ , wait for a stage  $s$  at which  $\Phi(XY; n) \downarrow = 0$  and keep  $n$  outside  $E$ . If there is no such stage, then we have an easy win. We use  $-1$  to denote this outcome. The purpose of having a switch point  $n$  is as follows. We will pick  $v$  and  $u$  larger than the use  $\varphi(XY; n)[s]$ . Thus by putting  $n$  into  $E$ , we can force either an  $X$ -change below  $u$  or a  $Y$ -change below  $v$ , which will be used in the permitting argument.

If  $\Phi(XY; n) = 0$ , but the use  $\varphi(XY; n)$  has moved, that is, there are consecutive accessible stages  $s$  and  $t$ ,  $\varphi(XY; n)[s] \neq \varphi(XY; n)[t]$ ,

then we restart  $S_{i,j}$ . Let us also use  $-1$  to indicate this outcome. Clearly, if  $-1$  is the true outcome, then we have a global win for  $R$  by demonstrating that either  $\Phi(XY; n) \downarrow \neq E(n)$  or  $\Phi(XY; n) \uparrow$ .

We now look at the normal cases, where we work under the assumption that  $\varphi(XY; n)$  is defined and never moves.

Step 1: Searching for  $v$ . During the searching process, various candidates  $c$  are tested.

Initially we pick a fresh number  $c$ . We use a list  $L_y$  to store the numbers waiting for  $Y$ 's permission. Initially set  $L_y = \emptyset$ .

Wait for a stage  $s$ , at which  $\Psi_j(A; c) \downarrow = 0$ . If no such  $s$  exists, then we get an easy win for  $S_{i,j}$  by showing  $C(c) = 0 \neq \Psi_j(A; c)$ . We use outcome  $0_c$  to indicate this outcome.

Suppose that there is a stage  $s$  at which  $\Psi_j(A; c) \downarrow = 0$ , then we preserve the computation and put  $c$  in the list  $L_y$  and wait for the permission from  $Y$ . We also define a Turing functional  $\Theta_{i,j}(A; z) = Y(z)$  for all  $z < c$  with use  $\psi_j(A; c)$ . If  $Y$  never permits any element in  $L_y$ , then we have a global win for  $R$  by demonstrating that  $Y \leq_T A$  via functional  $\Theta_{i,j}$ . We use  $\infty_y$  to indicate this outcome. Otherwise, that is,  $Y$  permits some element in  $L_y$ , then let  $v$  be the least such element. Set  $L_y = \emptyset$  and restart the functional  $\Theta_{i,j}$ .

We pause to point out that in order to make  $C$  computable from  $Y$ , we should have some control on the fate of  $v$  at  $R$ . For each  $v$  obtained in step 1, we execute step 2 to search for  $u$ . If  $u$  is found at the same stage, then we set a link to  $R$  and  $R$  is aware that there is a delay concerning  $v$  (more will be discussed after step 3); otherwise, i.e.,  $u$  is not found at the same stage, then we discard  $v$  forever and go back to step 1.

Step 2: Searching for  $u$ . We search for  $u$  only when we have found a  $v$ .

The searching process is similar to step 1. We use a list  $L_x$  to store the numbers waiting for the permission from  $X$ . We begin with picking a fresh  $b$ . Wait for a stage  $s$ , at which  $\Psi_i(A; b) \downarrow = 0$ . If no such  $s$  exists, then we win by showing  $B(b) = 0 \neq \Psi_i(A; b)$ . We use outcome  $0_b$  to indicate this outcome.

Suppose that there is a stage  $s$  at which  $\Psi_i(A; b) \downarrow = 0$ , then we preserve the computation, put  $b$  in the list  $L_x$  and wait for a permission from  $X$ . Define functional  $\Xi_{i,j}(A; z) = X(z)$  for all  $z < b$  with use  $\psi_i(A; b)$ . If  $X$  never permits any element in  $L_x$ , then we win  $R$  by showing that  $X \leq_T A$  via functional  $\Xi_{i,j}$ . We use  $\infty_x$  to indicate this outcome. Otherwise, that is,  $X$  permits some element in  $L_x$ , then let  $u$  be the least such element. Go to step 3.

Step 3: Switching.

Puts  $n$  into  $E$  and wait for an  $X$ - or a  $Y$ -change below  $\varphi(XY; n)$ . If there is no such change, then  $\Phi(XY) \neq E$ , thus we get a global win for  $R$  ( $R$  will have outcome 0 forever). Thus let us assume that either  $X$  or

$Y$  changes below the use  $\varphi(XY; n)$ . If  $X$  changes, then  $v$  is permitted by  $X$ , so we can enumerate  $v$  into  $C$ , discard  $u$  forever and win  $S_{i,j}$  by showing  $\Psi_j(A) \neq C$ ; if  $Y$  changes, then  $u$  is permitted by  $Y$ , so we can enumerate  $u$  into  $B$ , discard  $v$  forever and win  $S_{i,j}$  by showing  $\Psi_i(A) \neq B$ . In both cases, we add a finitary restraint on  $A$  to preserve the computation  $\Psi_j(A; v)$  or  $\Psi_i(A; u)$  respectively. We use  $1_c$  or  $1_b$  to indicate the outcome respectively.

Notice that after we reach step 3, we will make progress either on  $B \neq \Psi_i(A)$  or on  $C \neq \Psi_j(A)$ , provided that  $\Phi(XY) = E$ . We will arrange  $S_{i,j}$  on each path of the priority tree  $T$  in the following way. If  $S_{i,j}$  is satisfied by showing  $B \neq \Psi_i(A)$  (that is, having outcome  $0_b$  or  $1_b$ ) then we move to  $S_{i+1,0}$  and never consider  $S_{i,j'}$  for  $j' > j$ ; if  $S_{i,j}$  is satisfied by showing  $C \neq \Psi_j(A)$  (that is, having outcome  $0_c$  or  $1_c$ ) then we move to  $S_{i,j+1}$ .

In order to make  $B$  and  $C$  computable from  $X$  and  $Y$ , the fate of the numbers  $v$  and  $u$ , specified at the end of step 2, should be determined after some delay. The control of the delay is at the node  $R$  and it is done by setting up a link between  $R$  and  $S$  when we puts  $n$  into  $E$ . Later whenever  $R$  reaches an expansionary stage again, we travel the link to  $S_{i,j}$  and act as described in step 3 and the link gets canceled. This is similar to the one used in the proof of Lachlan's Nonbounding Theorem presented in Soare [9].

### 3. CONFLICTS AND MODIFIED STRATEGIES

Let us study the conflicts between the strategies.

**3.1. Conflicts between noncuppable and nondiamond strategies.** As both strategies have negative effects on the set  $A$ , there is no direct conflicts between them. However there are some concerns about their coordinations. First, the clearing  $\Delta$  procedure might delay the action of  $S$ ,  $R$  should not lost the control on the fate of some witnesses  $v$  and  $u$ . Secondly, the links set up by  $S$  for permitting should not interfere with the ones set up for clearing the  $\Delta$ . These can be solved by the following observations.

- The action at  $S$  has no conflict with  $M$ , thus we can execute the  $S$ -strategy at  $S$  (in particular, the fate of  $v$  will be determined at  $S$ ). This is different from  $Q$ -strategies, where we must clear all the axioms of  $\Delta$  to the right before we put a number into  $A$ . After executing the  $S$ -strategy, we can wait for the clearance of  $\Delta$  before we access the node at next level.
- When  $S$  wants to set up a link at stage  $s$ , it will have outcome either  $1_b$  or  $1_c$ , which was never accessible before  $s$ , otherwise,  $S$  would have been satisfied already. Thus there is no definition of  $\Delta$  above the outcome  $1_b$  or  $1_c$ . Therefore, the link involving  $S$  at any stage is unique.

- Suppose a link between  $S$  and  $R$  has been set up at stage  $s$  and  $t > s$  is the next  $R$ -expansionary stage at which  $R^\infty$  is accessible. We argue that we can travel this link at stage  $t$ , so that the fate of  $u$  and  $v$  is determined at stage  $t$ . The reason is that at stage  $s$  at which  $S$  is accessible, all  $\Delta$  to the right of  $S$  has been cleared. When  $R^\infty$  is accessible, all  $\Delta$  to the right of  $R^\infty$  is cleared, thus all  $\Delta$  to the right of  $S$  is cleared, because the link covers all  $N$  between  $R$  and  $S$ .

### 3.2. Conflicts between nonlow<sub>2</sub> and noncuppable strategies.

The introduction of  $M$ -nodes makes the restraint spread out along the tree, thus the conflicts between the nonlow<sub>2</sub> requirements  $P_j$  and  $Q_{j,m}$  and the noncuppable ones  $N_e$  and  $M_{e,p}$  become less serious. Since the actions are done at  $Q$  and  $M$ , let us concentrate on them and ignore the main requirements  $P$  and  $N$ .

Suppose that  $M$  has higher priority than  $Q$ . Observe that the restraint imposed by  $M$  is always finitary:  $M$  has restraint 0 if it has outcome  $\infty$  and has restraint  $\gamma(AW; d)$  if it has outcome 0. Therefore, we can execute the lower priority  $Q$ -strategy by picking a sufficient large killing point.

Suppose that  $Q$  has higher priority than  $M$ . If  $Q$  has outcome 0, then it will not act eventually; if  $Q$  has outcome  $\infty$ , then the lower priority  $M$  can use believable computations.

### 3.3. Conflicts between nonlow<sub>2</sub> and nondiamond strategies.

Given the nonlow<sub>2</sub> requirements  $P_e$  and  $Q_{e,m}$  and the nondiamond ones  $R_d$  and  $S_{d,i,j}$ . We drop the indices if there is no confusion. If  $S_{d,i,j}$  has finitary outcomes, namely,  $0_b, 0_c, 1_b, 1_c$  or  $-1$ , then  $S$  only imposes a finitary restraint, which is easy to handle. Let us assume that  $S_{d,i,j}$  has infinitary outcomes  $\infty_x$  or  $\infty_y$ . In this case, we need to preserve the totality of functionals  $\Xi_{i,j}(A)$  (demonstrating  $X$  is computable from  $A$ ) or  $\Theta_{i,j}(A)$  (demonstrating  $Y$  is computable from  $A$ ). As the nonlow<sub>2</sub> requirements allow us to spread out the uses, the preservation of  $\Xi_{i,j}(A)$  or  $\Theta_{i,j}(A)$  is eventually successful. We consider the following typical cases to illustrate how to preserve the totality of  $\Xi_{i,j}(A)$ . Other cases are easier hence skipped.

*Case 1.*  $P_e < R_d < S_{d,i,j} < Q_{e,m}$ .

In the  $\infty$  outcome for  $Q_{e,m}$ , we get a win on  $P_e$  by guaranteeing that  $\Lambda(A; x, n) \uparrow$  for all sufficiently large  $n$ , the cost here is an infinite computable set is put into  $A$  and its members are chosen at  $P_e$ . The computations  $\Xi_{i,j}(A; b)$  for some fixed  $b$  at  $S$  might be injured when  $Q_{e,m}$  puts numbers into  $A$ . To overcome this conflict, above the  $\infty$  outcome of  $Q$ , we restart all requirements of lower priority than  $P_e$ . In particular, the  $R_d$  and  $S_{d,i,j}$  will be restarted.

In the finitary outcome for  $Q_{e,m}$ , obviously  $Q$  itself cannot injure  $S$  infinitely often. To prevent the combined effect on  $S$  by infinitely many different finitary  $Q$ 's, each  $Q$  should pick its own killing point  $k$  such that  $Q$  only injures the computations  $\Xi_{i,j}(A; b)$  when  $b$  is larger than  $k$ . Thus for a fixed computation  $\Xi_i(A; b)$ , there are only finitely many  $Q$  requirements which can injure it and each of them only injures it finitely many times.

*Case 2.*  $R_d < P_e < S_{d,i,j} < Q_{e,m}$ .

In this case,  $R_d$  is of higher priority than  $P_e$ . Suppose that the subrequirement  $S_{d,i,j}$  has an infinitary outcome. Then  $R_d$  is satisfied. We restart  $P_e$  at an appropriate later node by choosing a new witness  $x'$  and putting the corresponding subrequirements  $Q_{e,m'}$  on the tree below the version of  $P_e$  with witness  $x'$ .

*Case 3.*  $P_e < Q_{e,m} < R_d < S_{d,i,j}$ .

When  $Q_{e,m}$  has outcome  $\infty$ , it will put infinitely many numbers into  $A$ , hence may injure the computation of the form  $\Xi_{i,j}(A; b) = 0$  which  $S$  wants to preserve. As usual, this situation is handled by so-called believable computations. Suppose that  $S_{d,i,j}$  is assigned to the node  $\alpha$  on  $T$ . We say that a computation  $\Xi_{i,j}(A; z) = 0$  is  $\alpha$ -believable at stage  $s$  if for all  $\beta$  labelled with a  $Q$ -requirement such that  $\beta \hat{\infty} \subseteq \alpha$ , the use  $\xi_{i,j}(A; z)$  is less than the number  $Q$  wants to put into  $A$ . Thus the action of  $\beta$  will not injure any  $\alpha$ -believable computations.

## 4. CONSTRUCTION

**4.1. Description of priority tree.** Fix a computable priority list of the requirements and subrequirements such that the subrequirements  $Q_{e,m}$  ( $M_{e,p}$ ,  $S_{e,i,j}$ , respectively) appear after  $P_e$  ( $N_e$ ,  $R_e$ , respectively).

We define the priority tree  $T$  and label it inductively in the usual manner. We label each node on  $T$  with a requirement or a subrequirement. We assume that the tree  $T$  grows upwards. The root node on  $T$  is labelled  $P_0$ . Suppose that  $\tau$  is a node on  $T$ . If  $\tau$  is labelled  $P_e$  then  $\tau$  has a unique outgoing edge labelled  $\infty$ ; if  $\tau$  is labelled  $S_{e,i,j}$ , then  $\tau$  has seven outgoing edges labelled  $-1, \infty_x, 0_b, \infty_y, 0_c, 1_b$  and  $1_c$  from left to right; otherwise, that is, if  $\tau$  is labelled  $Q_{e,m}$ ,  $N_e$ ,  $M_{e,p}$  or  $R_e$ , then  $\tau$  has two outgoing edges labelled  $\infty$  and  $0$ , with  $\infty$  to the left of  $0$ .

Let  $\tau$  be a node on  $T$  and  $\alpha \subset \tau$  be a node labelled  $S_{e,i,j}$ . We say that  $S_{e,i,j}$  is  $\Sigma_3$ -injured at  $\tau$  if there is a node  $\beta$  labelled  $P_i$ , and a node  $\eta$  labelled  $Q_{i,m}$  such that either

$$\beta \subset \alpha \hat{\infty}_x \subseteq \eta \hat{\infty} \subseteq \tau;$$

or

$$\beta \subset \alpha \hat{\infty}_y \subseteq \eta \hat{\infty} \subseteq \tau.$$

(Namely,  $S_{e,i,j}$  is between the pair  $P_i$  and  $Q_{i,m}$ , where  $Q_{i,m}$  demonstrates that  $P_i$  has a  $\Sigma_3$ -outcome.)

We say that  $P_e$  is  $\Sigma_3$ -injured at  $\tau$  if there is a node  $\beta$  labelled  $R_d$  and a node  $\eta$  labelled  $S_{d,i,j}$  such that either

$$\beta \hat{\infty} \subseteq \alpha \subset \eta \hat{\infty}_y \subseteq \tau;$$

or

$$\beta \hat{\infty} \subseteq \alpha \subset \eta \hat{\infty}_x \subseteq \tau.$$

(Note that we are not worried about the injury of  $P_e$ . Instead, we do not want a substrategy  $Q_{e,m}$  after  $S_{d,i,j}$  to injure  $S_{d,i,j}$ .)

We say that a requirement  $N_e$  is *satisfied* at  $\tau$  if there is a node  $\alpha \subset \tau$  labelled  $N_e$  such that either  $\alpha \hat{0} \subseteq \tau$  or there is an  $\eta$  labelled with a subrequirement  $M_{e,p}$  working for  $N_e$  such that  $\eta \hat{\infty} \subseteq \tau$ . (Namely, we see a global win for  $N_e$  at  $\eta$ .) If  $N_e$  is satisfied at  $\tau$  then all its subrequirements are satisfied at  $\tau$ .

We say that a requirement  $R_e$  is *satisfied* at  $\tau$  if there is a node  $\alpha \subset \tau$  labelled  $R_e$  such that either  $\alpha \hat{0} \subseteq \tau$  or  $\alpha$  is not  $\Sigma_3$ -injured at  $\tau$  and there is an  $\eta$  labelled with a subrequirement  $S_{e,i,j}$  working for  $R_e$  such that  $\eta \hat{\infty}_x$  (or  $\infty_y$ , or  $-1$ )  $\subseteq \tau$ . (Namely, we see a global win for  $R_e$  at  $\eta$ .) If  $R_e$  is satisfied at  $\tau$  then all its subrequirements are satisfied at  $\tau$ .

Suppose that  $R_e$  is not satisfied at  $\tau$  and a subrequirement  $S_{e,i,j}$  is assigned at node  $\alpha \subset \tau$ . we say that  $S_{e,i,j}$  is *satisfied* at  $\tau$  if either

- there is a node  $\beta \subset \tau$  labelled  $S_{e,i,j'}$  for some  $j' < j$  (working for the same  $R_e$ ) such that  $S_{e,i,j'}$  is not  $\Sigma_3$ -injured at  $\tau$  and  $\beta \hat{0}_b$  (or  $1_b$ )  $\subseteq \tau$ ; or
- $\alpha$  is not  $\Sigma_3$ -injured at  $\tau$  and  $\alpha \hat{0}_c$  or  $1_c \subseteq \tau$ .

Similarly we can define  $P_e$  and its subrequirements  $Q_{e,m}$  being satisfied at  $\tau$  as follows. We say that a requirement  $P_e$  is *satisfied* at  $\tau$  if  $P_e$  is not  $\Sigma_3$ -injured at  $\tau$  and there is an  $\eta$  labelled with a subrequirement  $Q_{e,m}$  working for  $P_e$  such that  $\eta \hat{\infty} \subseteq \tau$ . If  $P_e$  is satisfied at  $\tau$  then all its subrequirements are satisfied at  $\tau$ .

Continuing the inductive definition of  $T$ , if all  $\alpha \subset \tau$  have been labelled, then  $\tau$  is labelled with the highest priority  $O$  such that  $O$  is either a requirement which never appeared before or all its copies are  $\Sigma_3$ -injured at  $\tau$  or  $O$  is a unsatisfied new subrequirement.

**4.2. Conventions and environments.** In the remaining of this section, all computations at node  $\alpha$  are  $\alpha$ -believable ones. When we say that  $\alpha$  is accessible for the first time, we mean that  $\alpha$  is accessible for the first time after it was initialized. When we define a Turing functional  $\Lambda(A; x, n) = 1$ , we always pick use  $\lambda$  from the set  $\omega_x = \{\langle x, y \rangle : y \in \omega\}$  and enumerate the axiom of the form

$$\langle \langle x, n \rangle, 1, A_x \upharpoonright (\lambda + 1), (\bar{A}_x) \upharpoonright (\lambda + 1) \rangle,$$

into  $\Lambda$ , where  $A_x = A \cap \omega_x$  and  $\bar{A}_x = \omega_x - A_x$ . The purpose is that different  $P$ -strategies (even different versions of the same  $P$ -strategy) will not interfere with each other.

Let  $\alpha$  be a node on  $T$ . It is convenient to have a collection of parameters related to  $\alpha$ , used as a guide in the construction. We call this collection of parameters the *environment* of  $\alpha$ . Nodes with different labels have different environments. Natural words are chosen when we name the parameters, because of the consideration of readability in the proof and verification. Strictly speaking, we should write  $p(\alpha, s)$  for the value of parameter  $p$  at the beginning of stage  $s$  in the environment of  $\alpha$ , however, when there is no confusion, we will simply write  $p$ . We may also drop the indices of the requirements.

- (1) If  $\alpha$  is labelled  $P_e$ , then its environment contains a number  $x$  as its witness for diagonalization.
- (2) If  $\alpha$  is labelled  $Q_{e,m}$ , then its environment contains
  - a node  $\sigma$  below  $\alpha$  labelled  $P_e$ , for which  $Q_{e,m}$  is working, we will call  $\sigma$  the *head of  $\alpha$* ;
  - a number  $x$  for diagonalization chosen at  $\sigma$ ;
  - a set INFTY-S-NODE of nodes  $\beta$  labelled with  $S$  which are not  $\Sigma_3$ -injured at  $\alpha$  and either  $\sigma \subset \beta \hat{\infty}_y \subseteq \alpha$  or  $\sigma \subset \beta \hat{\infty}_x \subseteq \alpha$ ;
  - a set FIN-Q-NODE of nodes  $\beta$  labelled with  $Q_{e,m'}$  also working for  $\sigma$ , which are not  $\Sigma_3$ -injured at  $\alpha$ , and  $\sigma \subset \beta \hat{0} \subseteq \alpha$ ;
  - a number FIN-RESTRAINT which is the maximal restraint imposed on  $\alpha$  by a node to the left of  $\alpha$  or by a node below  $\alpha$  which has finitary outcome such as  $0$ ,  $1_b$  and  $1_c$ .
- (3) If  $\alpha$  is labelled  $N_e$ , then no parameter is needed.
- (4) If  $\alpha$  is labelled  $M_{e,p}$ , then its environment contains a flip point  $d$  for  $\Delta(W; p)$ .
- (5) If  $\alpha$  is labelled  $R_e$ , then no parameter is needed.
- (6) If  $\alpha$  is labelled  $S_{e,i,j}$ , then its environment contains
  - a node  $\sigma$  below  $\alpha$  labelled  $R_e$ , for which  $S_{e,i,j}$  is working, we will call  $\sigma$  the *head of  $\alpha$* ;
  - a stage LAST-ACC-STAGE at which  $\alpha$  was accessible for the last time;
  - a label LAST-OUTCOME which is the outcome of  $\alpha$  at stage LAST-ACC-STAGE;
  - a number  $n$  called *switch point*;
  - a number C-WITNESS, which is the witness for the diagonalization of  $\Psi_j(A) \neq C$ ;
  - a number B-WITNESS, which is the witness for the diagonalization of  $\Psi_i(A) \neq B$ ;
  - a list  $L_y$  which are the numbers waiting for  $Y$ 's permission;
  - a list  $L_x$  which are the numbers waiting for  $X$ 's permission;

- a number Y-PERMITTED-NUM which has passed  $Y$ 's permission;
- a number X-PERMITTED-NUM which has passed  $X$ 's permission.

**4.3. Construction.** We now describe the stage by stage construction. At stage  $s$ , we first specify a string  $TP_s$  of length less than or equal to  $s$ , called the *accessible string*, then act along the accessible string.

We define the accessible string inductively from the root. The root of the tree is always accessible.

At the inductive step, suppose that the node  $\alpha$  is accessible. If the length of  $\alpha$  is equal to  $s$  then we let  $\alpha = TP_s$  and go to the next stage.

Suppose that the length of  $\alpha$  is less than  $s$ . Then we first determine the outcome  $o$  of  $\alpha$ . Before we declare that  $\alpha \hat{\ } o$  is accessible, we check if there is any  $N$ -requirement below  $\alpha$  which stops  $\alpha$  having outcome  $o$ . If yes, then we stop defining the accessible string, start the procedure below, referred as *clearing  $\Delta$  for  $o$  at  $\alpha$*  and delay all actions except  $\alpha$  is labelled  $S$ ; otherwise, we let  $\alpha \hat{\ } o$  be accessible and take actions accordingly.

The procedure of clearing  $\Delta$  for  $o$  at  $\alpha$  is as follows.

Given  $\alpha$  and an outcome  $o$ . Ask if there is a pair of requirements  $N_e$  and  $M_{e,p}$ , such that (1)  $N_e$  is assigned to some node  $\beta$  below  $\alpha$  and (2)  $M_{e,p}$  is a subrequirement for  $N_e$ ,  $M_{e,p}$  is assigned to some node  $\tau$  to the right of  $\alpha \hat{\ } o$  and at  $\tau$  we defined  $\Delta_e(W; p)$ . If yes, then let  $\beta_0$  be the longest such  $\beta$ ,  $p_0$  be the smallest  $p$  for  $\beta$  and  $d_0$  be the flip point for  $p_0$ . Put  $d_0$  into  $D$  and set up a link  $(\beta_0, \alpha)$ . Initialize all nodes not labelled  $M$  which are to the right of  $\alpha \hat{\ } o$ , that is, cancel all actions desired by these nodes; cancel all parameters in the environment; cancel all restraint imposed by these nodes; start over the definition of Turing functionals which we are building; and cancel all links involving the node.

We now continue the definition of outcome of  $\alpha$  and the next accessible node. We consider the two cases based on whether or not there is a link starting from  $\alpha$ .

*Case 1.* There is a link of the form  $(\alpha, \tau)$ .

*Subcase 1.1.*  $\alpha$  is labelled with an  $N$ -requirement.

Then it is necessary that the link  $(\alpha, \tau)$  was set up when we clear  $\Delta$  for some outcome  $o$  at  $\tau$ . Check if the stage  $s$  is  $\alpha$ -expansionary.

- If no, then let  $\alpha \hat{\ } 0$  be accessible (Since 0 is the rightmost outcome of  $\alpha$ , no clearing is needed).
- If yes, then go to  $\tau$  and cancel the link  $(\alpha, \tau)$ . Check if there is an  $N$ -requirement which stops  $\tau$  having outcome  $o$ .
  - If yes, then repeat the procedure of clearing  $\Delta$  for  $o$  at  $\tau$ ;

- otherwise, let  $\tau \hat{o}$  be accessible and acts as described in the construction below.

*Subcase 1.2.*  $\alpha$  is labelled with an  $R$ -requirement, say  $R_e$ .

Then it is necessary that  $\tau$  is labelled with an  $S$ -requirement, say  $S_{e,i,j}$ . Check if the stage  $s$  is  $\alpha$ -expansionary.

- If no, then let  $\alpha \hat{0}$  be accessible (as discussed in Subcase 1.1, no clearing is needed).
- If yes, then go to  $\tau$  and cancel the link  $(\alpha, \tau)$ . Now it is necessary that either  $X_{e_1}$  or  $Y_{e_2}$  changed below  $\varphi(X_{e_1}Y_{e_2}; n)$ , where  $n$  is the switch point; and the values of Y-PERMITTED-NUM and X-PERMITTED-NUM have been defined, say  $v$  and  $u$  respectively.
  - If  $X_{e_1}$  changed below  $\varphi(X_{e_1}Y_{e_2}; n)$ , then enumerate  $v$  into  $C_e$ , and add restraint  $\psi_j(A; v)$  on  $A$ , cancel all other witnesses in  $L_x$  and  $L_y$  (including  $u$ ). Go to the next stage and initialize all nodes above  $\tau \hat{1}_c$ .
  - Otherwise, then  $Y_{e_2}$  must have changed below  $\varphi(X_{e_1}Y_{e_2}; n)$ . Enumerate  $u$  into  $B_e$ , add restraint  $\psi_i(A; u)$  on  $A$ , cancel all other witnesses in  $L_x$  and  $L_y$  (including  $v$ ). Go to the next stage and initialize all nodes above  $\tau \hat{1}_b$ .

Case 2. There is no link of the form  $(\alpha, \tau)$ .

Then we first decide the outcome  $o$  of  $\alpha$ . If  $\alpha$  is labelled with an  $S$ -requirement, then take the actions as described below now. Then check if there is any  $N$  which stops  $\alpha$  having outcome  $o$ . If yes, then start the procedure of clearing the  $\Delta$ -definitions for  $o$  as described earlier; if no, take the actions described below for requirements other than  $S$ .

We decide the outcome  $o$  and take the actions based on the label of  $\alpha$  as follows.

- (1)  $\alpha$  is labelled  $P_e$ . Then  $o = \infty$ , which is the only leaving edge anyway.

*Actions.* Let  $x$  be the witness of  $\alpha$  for  $P_e$  (if  $x$  has not been defined, then pick a fresh one). Let  $n$  be the least number at which  $\Lambda(A; x, n)$  is undefined. Pick a fresh use  $\lambda$ , and define  $\Lambda(A; x, n) = 1$  with use  $\lambda$ .

- (2)  $\alpha$  is labelled  $Q_{e,m}$ .

First we pick a killing point  $k$  as follows. For each  $\beta$  in INFTY-S-NODE, let  $\psi_\beta$  be the biggest  $\psi$ -use at  $\beta$  for numbers  $z$  less than or equal to  $|\alpha|$ . For each node  $\beta$  in FIN-Q-NODE, let  $\lambda_\beta$  be the  $\Lambda$ -use selected by  $\beta$  for  $\beta$ 's killing point. Calculate the value

$$m^* = \max(\{ \psi_\beta : \beta \in \text{INFTY-S-NODE} \} \cup \{ \lambda_\beta : \beta \in \text{FIN-Q-NODE} \} \cup \{ k' : k' \text{ is the killing point chosen at } \alpha' \subset \alpha \} \cup \{ \text{FIN-RESTRAINT} \}).$$

If the killing point for  $\alpha$  is undefined or it is defined but now it is less than  $m^*$ , then discard the old killing point and pick a fresh number  $k$  (in particular  $k$  is larger than  $m^*$ ); otherwise, keep the old killing point. (We assume that  $\lambda(A; x, k)$ , the  $\Lambda$ -use of  $k$  is larger than  $k$ . Thus if  $k > m^*$  then the relevant computation will not be injured when we put  $\lambda(A; x, k)$  into  $A$ .)

Let  $y$  be the least number such that

$$(\forall z < s_0)\theta_e(x, m, y, z) \uparrow,$$

where  $s_0$  is the stage at which  $\alpha$  was accessible for the last time. If  $\theta_e(x, m, y, s) \downarrow$  then  $o = \infty$ ; otherwise,  $o = 0$ .

*Actions.* We act only when  $\alpha$  has outcome  $\infty$ . Put  $\lambda(A; x, k)$  into  $A$ .

- (3)  $\alpha$  is labelled  $N_e$ .

Check if  $s$  is an  $\alpha$ -expansionary stage. If yes, then  $o = \infty$ ; otherwise  $o = 0$ . No action is required since the jobs are distributed to the subrequirements  $M_{e,p}$ .

- (4)  $\alpha$  is labelled  $M_{e,p}$ .

Let  $d$  be the flip point for  $\alpha$  (if  $d$  is undefined, then pick it fresh). Check if  $\gamma_{e_1}(AW_{e_2}; d)[s] \neq \gamma_{e_1}(AW_{e_2}; d)[t]$ , where  $t$  is the stage at which  $\alpha$  was accessible for the last time. If yes, then  $o = \infty$ ; otherwise  $o = 0$ .

*Actions.* We take action only when  $\alpha$  has outcome 0. Check if  $\Delta_e(W_{e_2}; p)$  is defined at stage  $s$ .

- If yes, then do nothing;
- otherwise, define

$$\Delta_e(W_{e_2}; p) = \Gamma_{e_1}(AW_{e_2}; p)$$

with use  $\gamma_{e_1}(AW_{e_2}; d)$  and set a restraint on  $A$  of amount  $\gamma_{e_1}(AW_{e_2}; d)$ .

- (5)  $\alpha$  is labelled  $R_e$ . If  $s$  is an  $\alpha$ -expansionary stage, then  $o = \infty$ ; otherwise,  $o = 0$ . No action is taken since the jobs are distributed to the subrequirements  $S_{e,i,j}$ .

- (6)  $\alpha$  is labelled  $S_{e,i,j}$ . We drop the index  $e$  in the discussion if there is no confusion.

If  $s$  is the first stage  $\alpha$  is accessible after it is initialized, then  $o = -1$ . Pick a fresh switch point  $n$  if it is undefined. *This takes care of the initial case.*

If there is a  $u \in B_e$  such that  $\Psi_i(A; u) = 0$  then  $o = 1_b$ ; if there is a  $c \in C_e$  such that  $\Psi_j(A; c) = 0$ , then let  $o = 1_c$ . No action is required. *This takes care of the case when  $S_{e,i,j}$  has been satisfied by successful diagonalization.*

We now look at the normal cases. Let  $n$  be the switch point. If  $\Phi(XY; n) \neq 0[s]$  or  $(\Phi(XY; n) \downarrow = 0[s] \text{ and } \varphi(XY; n)[s] \neq \varphi(XY; n)[t])$ , where  $t$  is LAST-ACC-STAGE, then  $o = -1$ .

*Actions.* Cancel the parameters C-WITNESS, B-WITNESS,  $L_x$ ,  $L_y$ , Y-PERMITTED-NUM and X-PERMITTED-NUM; restart the functionals  $\Theta_{e,i,j}$  and  $\Xi_{e,i,j}$ .

We determine the next accessible node based on LAST-OUTCOME as follows.

- Suppose that LAST-OUTCOME is  $-1$ . If  $\Phi(XY; n)[s] = 0$ , then  $o = 0_c$ .

*Action.* Pick C-WITNESS fresh.

- Suppose that LAST-OUTCOME is  $0_c$ . Let C-WITNESS be  $c$ . If  $\Psi_j(A; c) = 0[s]$ , then  $o = \infty_y$ ; otherwise  $o = 0_c$  be accessible.

*Actions.* We act only when  $\alpha$  has outcome  $\infty_y$ . Put  $c$  in the list  $L_y$ ; define  $\Theta_{i,j}(A; z) = Y(z)$  for all undefined  $z \leq c$  with use  $\psi_j(A; c)$ .

- Suppose that LAST-OUTCOME is  $\infty_y$ . If there is a  $w$  in the list  $L_y$  such that  $Y \upharpoonright w[s] \neq Y \upharpoonright w[t]$  where  $t$  is LAST-ACC-STAGE, then  $o = 0_b$ .

*Actions.* Cancel the parameter C-WITNESS, set  $L_y = \emptyset$  and B-WITNESS fresh. Restart the definition of  $\Theta_{i,j}$ . Let Y-PERMITTED-NUM  $v$  be the least such  $w$ .

Otherwise, that is, for all  $w \in L_y$ ,  $Y \upharpoonright w[s] = Y \upharpoonright w[t]$ , where  $t$  is LAST-ACC-STAGE, then  $o = 0_c$ .

*Actions.* Check if there is a  $w \in L_y$ ,  $\Theta_{i,j}(A; w)$  is undefined. If yes, let  $c$  be the least such  $w$ , delete all number larger than or equal to  $c$  from the list  $L_y$  and set C-WITNESS to be  $c$ ; otherwise, that is, for all  $w \in L_y$ ,  $\Theta_{i,j}(A; w)$  is defined and  $Y \upharpoonright w[s] = Y \upharpoonright w[t]$ , where  $t$  is LAST-ACC-STAGE, then set C-WITNESS a fresh value.

- Suppose that LAST-OUTCOME is  $0_b$ . Let B-WITNESS be  $b$ . If  $\Psi_i(A; b) = 0[s]$  then  $o = \infty_x$ ; otherwise,  $o = 0_b$ .

*Actions.* We take actions only when  $\alpha$  has outcome  $\infty_x$ . Put  $b$  in the list  $L_x$ ; define  $\Xi_{i,j}(A; z) = Y(z)$  for all undefined  $z \leq b$  with use  $\psi_i(A; b)$ .

- Suppose that LAST-OUTCOME is  $\infty_x$ . If there is a  $w'$  in the list  $L_x$  such that such that  $\Xi_{i,j}(A; w')$  is defined at stage  $s$  and  $X \upharpoonright w'[s] \neq X \upharpoonright w'[t]$  where  $t$  is LAST-ACC-STAGE, then stop the definition of accessible string.

*Actions.* Let X-PERMITTED-NUM  $u$  be the least such  $w'$ , restart functional  $\Xi_{i,j}$ , put  $n$  into  $E$ . Set link  $(\beta, \alpha)$  where  $\beta$  is the head of  $\alpha$ . Go to next stage.

Otherwise, that is, for all  $w' \in L_x$ ,  $X \upharpoonright w'[s] = X \upharpoonright w'[t]$ , where  $t$  is LAST-ACC-STAGE, then  $o = 0_c$ .

*Actions.* Check if there is a  $w' \in L_x$ ,  $\Xi_{i,j}(A; w')$  is undefined. If yes, let  $b$  be the least such  $w$ , delete all number larger than or equal to  $b$  from the list  $L_x$ , set B-WITNESS to be  $b$ ; if for all  $w' \in L_x$ ,  $\Xi_{i,j}(A; w')$  is defined and  $X \upharpoonright w'[s] = X \upharpoonright w'[t]$ , where  $t$  is LAST-ACC-STAGE, then set B-WITNESS a fresh value.

At the end of the stage, we *initialize* all nodes to the right of  $TP_s$ . This finishes the construction.

## 5. VERIFICATION

We now verify that the construction works. We begin with the routine lemma which shows that the true path exists.

**Lemma 5.1.** *For any  $e \in \omega$ , there is a unique node  $\alpha$  on  $T$  such that  $\alpha$  is the leftmost one of length  $e$  which is accessible (and not covered by any link) infinitely often.*

*Proof.* We do an induction on  $e$ . We only need to show that there is a node which is accessible infinitely often. Since the priority tree  $T$  is finite branching, there is always a leftmost one.

Suppose true for  $e$ . Let  $\alpha$  be the string of length  $e$  which is accessible infinitely often. Fix  $s$ , we need to show that there is  $t > s$  and there is outcome  $o$  such that  $\alpha \hat{\ } o$  is accessible at  $t$ .

It suffices to rule out the following cases.

*Case 1.* At some stage when  $\alpha$  is accessible, we did not continue the definition of accessible string, instead we set up a link of the form  $(\beta, \alpha)$  for some node  $\beta$ .

At the stage  $t_0 > s$ , when  $\alpha$  is accessible again, the link is traveled hence get canceled.

*Subcase 1.1.* If the link was set because some  $N$ -requirement stops  $\alpha$  having outcome  $o$ , then the link can only transfer into another link  $(\beta', \alpha)$  for some  $\beta' \subset \beta$ . Since there is only finitely many such  $\beta'$ , eventually  $\alpha$  will be accessible and no link of the form  $(\beta, \alpha)$  exists.

*Subcase 1.2* If  $\alpha$  is labelled with an  $S$  requirement, then at any stage  $t > t_0$  when  $\alpha$  is accessible again, it will have outcome either  $1_b$  or  $1_c$ .

Note that the two subcases are mutually disjoint, because the outcomes  $1_b$  and  $1_c$  are mutually exclusive and they are the rightmost outcomes of  $S$ , there is no need to clear  $\Delta$  for  $1_b$  or  $1_c$  at  $\tau$ .

*Case 2.* There is a link  $(\alpha, \tau)$  for some node  $\tau$ . Therefore, we travel to  $\tau$  via the link, instead of accessing  $\alpha \hat{\ } o$ .

Then  $\alpha$  is labelled with either  $N$  or  $R$ , which has two outgoing edges  $\infty$  and  $0$ . If there is a stage  $t > s$  at which  $\alpha$  is accessible and  $t$  is not  $\alpha$ -expansionary, then  $\alpha \hat{\ } 0$  will be accessible at  $t$ . Suppose that for all stage  $t > s$  at which  $\alpha$  is accessible,  $t$  is  $\alpha$ -expansionary. By the

argument in case 1, there is a stage  $t' > t$  at which  $\alpha$  is accessible again and no  $N$  will stop  $\alpha$  having outcome  $\infty$ . At stage  $t'$ , the link  $(\alpha, \tau)$  is traveled and canceled. Thus at the stage  $t'' > t'$  at which  $\alpha$  is accessible again and no  $N$  stops  $\alpha$  having outcome  $\infty$ ,  $\alpha \hat{\infty}$  will be accessible at  $t''$ .

This finishes the proof of Lemma 5.1.  $\square$

Let  $TP$  be the *true path* in  $T$ , that is,  $TP$  is the leftmost path which is accessible infinitely often. By Lemma 5.1,  $TP$  exists.

Next we show that every requirement is represented along the true path.

**Lemma 5.2.** *Let  $TP$  be the true path on  $T$ . Then for any requirement  $P_e$ ,  $N_e$  and  $R_e$ , and subrequirement  $Q_{e,m}$ ,  $M_{e,p}$  and  $S_{e,i,j}$ , there is a unique node  $\alpha$  on  $TP$  such that either for all  $n$  greater than the length of  $\alpha$ , the requirement is never  $\Sigma_3$ -injured at  $TP \upharpoonright n$  or for all  $n$  greater than the length of  $\alpha$ , it is satisfied at  $TP \upharpoonright n$ .*

*Proof.* The Lemma follows from the assignment of requirements by a routine induction on  $e$ . In fact, it is true for any infinite path  $\mathcal{P}$  on  $T$ .  $\square$

We argue by induction along  $TP$  that every requirement is satisfied. We split it into two lemmas.

**Lemma 5.3.** *Let  $\alpha$  be a node on  $TP$  and  $O$  be the label of  $\alpha$ . Then*

- (a) *Suppose that  $O$  is  $P_e$ . Then the parameter  $x$  in the environment of  $\alpha$  is eventually fixed. (Note that this  $x$  is only the witness for the version of  $P_e$  at  $\alpha$ , we do not claim that  $x$  is the final witness for  $P_e$  as  $\alpha$  might be  $\Sigma_3$ -injured.) Let  $\beta$  be a node on  $TP$  labelled  $Q_{e,m}$  working for  $\alpha$ . Then the killing point  $k$  at  $\beta$  is eventually fixed. Moreover,*
  - (a1) *if  $\beta \hat{\ } 0 \subset TP$ , then for all  $n \leq k$ ,  $\Lambda(A; x, n)$  is defined;*
  - (a2) *if  $\beta \hat{\ } \infty \subset TP$ , then there is a stage  $s$ , such that every number in the sequence  $\langle \lambda(A; x, k)[t] : t > s \rangle$  (specified at  $\alpha$ ) is enumerated into  $A$ , consequently, for all  $n \geq k$ ,  $\Lambda(A; x, n)$  is undefined.*
- (b) *Suppose that  $O$  is  $N_e$ . Then  $\alpha \hat{\ } \infty \subset TP$  if and only if there are infinitely many  $\alpha$ -expansionary stages. Let  $\beta$  be node on  $TP$  labelled  $M_{e,p}$  working for  $\alpha$ . Then the flip point  $d$  at  $\beta$  is eventually fixed. Moreover,*
  - (b1) *if  $\beta \hat{\ } 0 \subset TP$ , then  $\Delta_e(W_{e_2}; p)$  is defined;*
  - (b2) *if  $\beta \hat{\ } \infty \subset TP$ , then  $\Gamma_{e_1}(AW_{e_2}; p) \uparrow$ .*
- (c) *Suppose that  $O$  is  $R_e$ . Then  $\alpha \hat{\ } \infty \subset TP$ , if and only if there are infinitely many  $\alpha$ -expansionary stages. Let  $\beta$  be the node on  $TP$  labelled  $S_{e,i,j}$  working for  $\alpha$ . If  $R_e$  is not  $\Sigma_3$ -injured, then*
  - (c1) *if  $\beta \hat{\ } (-1) \subset TP$ , then  $\Phi_{e_0}(X_{e_1}Y_{e_2}) \neq E$ ;*

- (c2) if  $\beta^{\wedge} \infty_x \subset TP$ , then  $L_x$  is an infinite set and for each  $w' \in L_x$ ,  $\Psi_{e,i}(A; w') = 0$  and  $\Xi_{e,i,j}(A) = X_{e_1}$ ;
- (c3) if  $\beta^{\wedge} 0_b \subset TP$ , then  $\Psi_{e,i}(A) \neq B_e$ ;
- (c4) if  $\beta^{\wedge} \infty_y \subset TP$ , then  $L_y$  is an infinite set and for each  $w \in L_y$ ,  $\Psi_{e,j}(A; w) = 0$  and  $\Theta_{e,i,j}(A) = Y_{e_2}$ ;
- (c5) if  $\beta^{\wedge} 0_c \subset TP$ , then  $\Psi_{e,j}(A) \neq C_e$ ;
- (c6) if  $\beta^{\wedge} 1_b \subset TP$ , then  $\Psi_{e,i}(A) \neq B_e$ ;
- (c7) if  $\beta^{\wedge} 1_c \subset TP$ , then  $\Psi_{e,j}(A) \neq C_e$ .

*Proof.* We prove (a)-(c) by simultaneous induction. We begin with statement (a).

Suppose that  $\alpha$  is labelled  $P_e$  and  $x$  is the witness for diagonalization chosen at  $\alpha$ . Since  $\alpha$  only changes its parameter when it is initialized,  $x$  is eventually fixed.

Let  $\beta$  be the node labelled  $Q_{e,m}$  as described in (a). We first argue that the killing point  $k$  used by  $\beta$  is eventually fixed. As  $\beta$  is on the true path, by induction, the statements (a) and (c) apply to the nodes below  $\beta$ . We go through the definition of the killing point  $k$ . For any  $\eta$  in INFTY-S-NODE,  $\eta$  is not  $\Sigma_3$ -injured. Without loss of generality, let us assume that  $\eta^{\wedge} \infty_y \subseteq \beta$ . By statement (c4), the computations  $\Psi_{e,j}(A; w) = 0$  at  $\eta$  on numbers  $w < |\beta|$  in  $L_y$  are eventually preserved. Hence the uses  $\psi_{e,j}(A; w)$  are eventually fixed. For any  $\eta$  in FIN-Q-NODE, by statement (a), the killing point at  $\eta$  is eventually fixed, hence the number  $\lambda_\eta$  chosen by  $\eta$  is eventually fixed. Clearly the parameter FIN-RESTRAINT is eventually fixed. Finally, by induction hypothesis, the killing point  $k'$  for some node  $\beta'$  below  $\beta$  is eventually fixed. Consequently, the killing point  $k$  at  $\beta$  is eventually fixed.

We now prove statement (a1). Suppose  $\beta^{\wedge} 0 \subset TP$ . Let  $s$  be the stage after which no nodes to the left of  $\beta^{\wedge} 0$  are accessible, and the killing point  $k$  at  $\beta$  is fixed. When  $\alpha$  is accessible after stage  $s$ , we will define  $\Lambda(A; x, k)$  with a fresh use  $\lambda(A; x, k)$  if it has not yet been defined. We claim that this axiom  $\Lambda(A; x, k)$  will never be injured, which also implies that for all  $n \leq k$ ,  $\Lambda(A; x, n)$  is defined. The reason is as follows. By our convention on selection of  $\Lambda$ -uses, only the nodes  $\beta'$  working for the same  $\alpha$  could possibly injure  $\Lambda(A; x, k)$ . If  $\beta'$  is to the right of  $\beta$ , then it is taken care of by initialization. If  $\beta'$  is between  $\alpha$  and  $\beta$ , then  $\beta'$  necessarily has outcome 0, then  $\beta'$  will not act after stage  $s$ , since otherwise, the node  $\beta'^{\wedge} \infty$  which is to the left of  $\beta$  would be accessible. If  $\beta^{\wedge} 0 \subseteq \beta'$ , then by their choices of killing points,  $\beta'$  will not injure  $\Lambda(A; x, k)$ . Therefore  $\Lambda(A; x, k)$  is defined.

We now prove statement (a2). Suppose that  $\beta^{\wedge} \infty \subset TP$ . Let  $s$  be the stage after which no node to the left of  $\beta^{\wedge} \infty$  is accessible and the killing point  $k$  at  $\beta$  is fixed. Let  $\lambda(A; x, k)$  be the use chosen at  $\alpha$  at some stage  $t > s$ . Since  $\beta^{\wedge} \infty$  is on the true path, there is a stage  $t' > t$  at which  $\beta^{\wedge} \infty$  is accessible. At stage  $t'$ ,  $\lambda(A; x, k)$  will be enumerated into  $A$ . Consequently,  $\Lambda(A; x, n)$  is undefined for all  $n \geq k$ .

This establishes statement (a).

Next we prove statement (b). Suppose that  $\alpha$  is labelled  $N_e$ . If  $\alpha \hat{\infty} \subset TP$ , then obviously there are infinitely many  $\alpha$ -expansionary stages. Suppose that  $\alpha \hat{0} \subset TP$  and  $s$  is the stage after which no nodes to the left of  $\alpha \hat{0}$  are accessible. If there is an  $\alpha$ -expansionary stage  $t$  after  $s$ , then after clearing  $\Delta$  for the outcome  $\infty$  at  $\alpha$ , say at  $t' > t$ , we still make  $\alpha \hat{\infty}$  accessible even if  $t'$  is not  $\alpha$ -expansionary. Thus at stage  $t'$ ,  $\alpha \hat{\infty}$  is accessible, contradicting the choice of  $s$ . Hence there are only finitely many  $\alpha$ -expansionary stages.

Let  $\beta$  be node on  $TP$  labelled  $M_{e,p}$  working for  $\alpha$ . As  $\beta$  only change its flip point when it is initialized, the flip point  $d$  at  $\beta$  is eventually fixed.

We now prove statement (b1). Suppose that  $\beta \hat{0} \subset TP$ . Let  $s$  be the stage at which  $\beta \hat{0}$  is accessible and after which no node to the left of  $\beta \hat{0}$  is accessible. Then by construction, either  $\Delta_e(W_{e_2}; p)$  has been defined before  $s$  or we defined it at stage  $s$  with use  $\gamma_{e_1}(AW_{e_2}; d)[s]$ . As  $\gamma_{e_1}(AW_{e_2}; d)[s]$  is fixed (otherwise,  $\alpha \hat{\infty}$  would be accessible),  $\Delta_e(W_{e_2}; p)$  will never be injured after stage  $s$ , since its use will be of the form  $\gamma_{e_1}(AW_{e_2}; d')$  for some  $d' < d$ . This establishes statement (b1). Statement (b2) follows from condition of  $\beta$  having outcome  $\infty$ .

We now prove statement (c).

For notational simplicity, we drop the index  $e$  in the discussions below.

The argument about  $R_e$  is similar to the one about  $N$  in (b).

Suppose  $\alpha \hat{\infty} \subset TP$  and  $\alpha$  is never  $\Sigma_3$ -injured at any node on  $TP$ . Let  $\beta$  be the node on  $TP$  labelled  $S_{i,j}$ , which is working for  $\alpha$ . Fix a stage  $s_0$  after which no node to the left of  $\beta$  is accessible and  $\beta$  is never initialized. Then the switch point  $n$  is fixed after  $s_0$ .

Let us consider the statements (c1) — (c7). We prove only (c1), (c6), (c5) and (c4), because the proof of (c7), (c3) and (c2) are similar to (c6), (c5) and (c4) respectively.

We begin with (c1). Suppose  $\beta \hat{(-1)} \subset TP$ . Clearly for infinitely many stages  $t > s_0$ , either  $\Phi(XY)(n) \neq 0[t]$ ; or  $\Phi(XY)(n) \downarrow [t] = 0$ , but with a use different from previous one. In both cases,  $\Phi(XY)(n) \neq 0 = E(n)$ , hence  $\Phi(XY) \neq E$ .

Next we consider (c6). Suppose  $\beta \hat{1}_b \subset TP$ . Let  $s_1$  be the least stage at which  $\beta \hat{1}_b$  is accessible and after which no node to the left of  $\beta \hat{1}_b$  is accessible. At stage  $s_1$ , by construction, there is a  $u \in B$  such that  $\Psi_i(A; u) = 0$ . It remains to argue that this computation is preserved forever. At stage  $s_1$ , all nodes extending or to the right of  $\beta \hat{1}_b$  get initialized, hence no nodes from there can injure  $\Psi_i(A; u) = 0$ . After stage  $s_1$  no node to the left of  $\beta \hat{1}_b$  is accessible. Finally the nodes below  $\beta$  cannot injure  $\Psi_i(A; u) = 0$ , because the computation is believable. This establishes (c6).

We now move to (c5). Suppose  $\beta \hat{0}_c \subset TP$ . Let  $s_1$  be the stage after which no node to the left of  $\beta \hat{0}_c$  is accessible. Then after  $s_1$ , the parameter C-WITNESS is fixed, say  $c$ . Thus for all  $t > s_1$  at which  $\beta \hat{0}_c$  is accessible,  $\Psi_j(A; c) \neq 0[t]$ , since otherwise  $\beta \hat{\infty}_y$  would be accessible. Clearly  $c \notin C$ , hence  $\Psi_j(A) \neq C$ .

Finally we consider (c4). Suppose  $\beta \hat{\infty}_y \subset TP$ . We need to show that  $\Theta_{i,j}(A) = Y$ .

Let  $s_1$  be a least stage at which  $\beta \hat{\infty}_y$  is accessible and after which no node to the left of  $\beta \hat{\infty}_y$  is accessible. We show that there is a sequence  $\langle w_k : k \in \omega \rangle$  such that for each  $k \in \omega$ ,  $\Psi_j(A; w_k) = 0$ .

We do an induction on  $k$ . Let  $w_0$  be the only element in  $L_y[s_1]$ . We show that  $\Psi_j(A; w_0) = 0$ . It suffices to argue that the computation  $\Psi_j(A; w_0)$  can only be injured finitely often after stage  $s_1$ . Let us look at the nodes on different regions on the tree. We only need to consider those nodes  $\eta$  labelled with a  $Q$  requirements, because only those nodes put numbers into  $A$ . If  $\eta$  is to the left of  $\beta \hat{\infty}_y$ , it is never accessible after  $s_1$ . If  $\eta$  is strictly to the right of  $\beta \hat{\infty}_y$ , it is initialized whenever  $\beta \hat{\infty}_y$  is accessible. If  $\eta$  is below  $\beta$ , then the action of  $\eta$  will not injure the computation  $\Psi_j(A; w_0)$ , since otherwise it would not be believable. The only remaining worry is those  $\eta$  above  $\beta \hat{\infty}_y$  labelled with  $Q$ , say  $Q_{d,m}$ , which are subrequirements of some  $P_d$ . First, let us consider the case that  $P_d$  is assigned to a node  $\tau$  above  $\beta \hat{\infty}_y$ . When  $\Psi_j(A; w_0)$  is defined at some stage after  $s_1$ , there are only finitely many such  $P$ -requirements having chosen their  $\lambda$ 's, each of them can injure  $\Psi_j(A; w_0)$  once. The reason is as follows. Suppose that at stage  $s$   $Q_{d,m}$  puts  $\lambda$  into  $A$  destroying the computation  $\Psi_j(A; w_0)$ . When  $\tau$ , which is above  $\beta \hat{\infty}_y$ , selects its new use  $\lambda'$ ,  $\Psi_j(A; w_0)$  has been redefined, hence  $\lambda'$  will be larger than the new use for  $\Psi_j(A; w_0)$ . Thus  $Q_{d,m}$  will not injure  $\Psi_j(A; w_0)$  twice. Finally, we look at the case that  $P_d$  is assigned to a node  $\mu$  below  $\beta$ . By the assignment of requirements,  $\mu$  is below  $\alpha$ , because  $\beta$  is  $\Sigma_3$ -injured if and only if  $\alpha$  is.

- If  $|\eta| \leq w_0$ , then  $Q_{d,m}$  can only act finitely often, otherwise, the version of  $Q_{d,m}$  on the true path would have outcome  $\infty$ , (because they all test the same  $\Pi_2$  condition  $(\forall y)(\exists z)\theta(x, m, y, z)$ ), which implies  $R$  is  $\Sigma_3$ -injured by the pair  $\mu$  and  $\eta$ . Let  $s_2$  be the stage after which all these  $\eta$ 's will not act.
- If  $|\eta| > w_0$ , if  $\beta$  is not  $\Sigma_3$ -injured at  $\eta$ , then  $Q_{d,m}$  will not injure  $\Psi_j(A; w_0)$  by the choice of killing point. If  $\beta$  is  $\Sigma_3$ -injured at  $\eta$ , let  $(\mu_0, \eta_0)$  be the pair of  $P$  and  $Q$  requirements, which  $\Sigma_3$ -injures  $\beta$  and no other such pairs are between  $\mu_0$  and  $\eta_0$ . After stage  $s_2$ , the  $\eta$ 's having length less than  $w_0$  will not act, thus we may assume that  $|\eta_0| > w_0$ . Then  $\eta_0$ 's killing point  $k_0$  is too large to injure  $\Psi_j(A; w_0)$ , so is the killing point  $k$  for  $\eta$ , since  $k > k_0$ .

Thus we have only finitely many  $Q_{d,m}$  and each of them can act finitely often, consequently  $\Psi_j(A; w_0)$  will be preserved. This establishes the base case. The proof for inductive case is similar, hence omitted.

We now show that  $\Theta_{i,j}(A) = Y$ . First observe that at any stage  $t > s_1$ , for any  $z \in L_y[t]$  (not necessarily in the final sequence  $\langle w_k : k \in \omega \rangle$ ), if  $\Theta_{i,j}(A; z)$  is defined at stage  $t$  then  $\Theta_{i,j}(A; z) = Y(z)$  (otherwise, we will see a  $Y$ -change, thus some node to the left of  $\beta^\wedge \infty_y$  would be accessible). Next let  $\langle w_k : k \in \omega \rangle$  be the sequence defined above. As  $\Psi_j(A; w_k)$  is eventually defined and for all  $w_{k-1} < z \leq w_k$ ,  $\Theta_{i,j}(A; z)$  is defined with use  $\psi_j(A; w_k)$ . Hence  $\Theta_{i,j}(A; z)$  is eventually defined and equal to  $Y$ . This established (c4) and Lemma 5.3.  $\square$

Finally we show that all requirements are satisfied.

**Lemma 5.4.** *For each  $e, i, j$  in  $\omega$  all requirements  $P_e, N_e, R_e$  and  $S_{e,i,j}$  are satisfied.*

*Proof.* First we argue that the requirements  $P_e$  are satisfied.

Let  $\alpha$  be the unique node on the true path labelled  $P_e$  and defined as in Lemma 5.2. Let  $x$  be the witness chosen for  $P_e$  at  $\alpha$ . This  $x$  is the final witness for  $P_e$ , since  $P_e$  only changes  $x$  when the old version of it is  $\Sigma_3$ -injured.

We consider two cases based on whether  $P_e$  has a global  $\Sigma_3$ -outcome.

*Case 1.* There is a node  $\beta$  on  $TP$  labelled  $Q_{e,m}$  working for  $\alpha$ , such that  $\beta^\wedge \infty \subset TP$ .

At each of the infinitely many stages when  $\beta^\wedge \infty$  is accessible,  $\beta$  finds a new  $y$  such that  $(\exists z)\theta(x, m, y, z)$ . Therefore, the  $\Pi_2$ -formula  $(\forall y)(\exists z)\theta(x, m, y, z)$  holds. Hence  $m$  is the  $\Sigma_3$ -witness that  $x$  is in  $U_e$ . On the other hand, by statement (a) in Lemma 5.3,  $\Lambda(A; x, n)$  undefined for all  $n \geq k$ . It follows that the set  $V_x^A$  is finite.

*Case 2.* For all nodes  $\beta_m$  on  $TP$  labelled  $Q_{e,m}$  working for  $\alpha$ ,  $\beta_m^\wedge 0 \subset TP$ .

In this case, we argue that  $x$  is not in  $U_e$  and  $V_x^A = \omega$  by showing that for all  $m$  in  $\omega$ ,  $(\exists y)(\forall z)\neg\theta(x, m, y, z)$  and  $\Lambda(A; x, m) \downarrow$ .

Fix  $m$ . Let  $s_m$  be the stage after which no node to the left of  $\beta_m$  is accessible. Let  $y_m$  be the least number such that  $(\forall z < s_m)\neg\theta(x, m, y_m, z)$ . Then for all  $z \geq s_m$  we still have  $\neg\theta(x, m, y_m, z)$ , since otherwise  $\beta_m^\wedge \infty$  would be accessible. Since it is true for all  $m$ ,  $x$  is not in  $U_e$ . On the other hand, by statement (a) in Lemma 5.3, all  $\Lambda(A; x, m)$  is eventually defined.

In both cases, the requirement  $P_e$  is satisfied.

We now prove that  $N_e$  is satisfied.

First notice that the set  $D$ , which we built, is c.e. If there is a c.e. set  $W_{e_2}$  such that  $A \oplus W_{e_2}$  is complete, then there is a Turing functional  $\Gamma_{e_1}$  such that  $\Gamma_{e_1}(AW_{e_2}) = K$ . Let  $e$  be the code of the pair  $\langle e_1, e_2 \rangle$  and  $\alpha$  be the unique node labelled  $N_e$  on  $TP$ . We show that  $\Delta_e(W_{e_2})$  is total and equal to  $K_0$ .

Fix a stage  $s_0$ , after which  $\alpha$  never gets initialized. Clearly, if  $\alpha \hat{\ } 0$  is not on  $TP$ , otherwise  $\Gamma_{e_1}(AW_{e_2}) \neq K$ . Thus  $\alpha \hat{\ } \infty$  is on  $TP$ , we consider two cases based on whether  $N_e$  has a global  $\Sigma_3$ -outcome.

*Case 1.* There is a node  $\beta$  on  $TP$  labelled  $M_{e,p}$  working for  $\alpha$ , such that  $\beta \hat{\ } \infty \subset TP$ .

Then by statement (b2) in Lemma 5.3,  $\Gamma_{e_2}(AW_{e_1}; p)$  is undefined,  $N_e$  is satisfied trivially.

*Case 2.* For all nodes  $\beta_p$  on  $TP$  labelled  $M_{e,p}$  working for  $\alpha$ ,  $\beta_p \hat{\ } 0 \subset TP$ .

In this case, by statement (b1) in Lemma 5.3, the Turing functional  $\Delta_e(W_{e_2})$  is total.

Fix a number  $p$ . We argue  $\Delta_e(W_{e_2}; p)$  is equal to  $\Gamma_{e_1}(AW_{e_2}; p)$ . Let  $s$  be the stage after which no node to the left of  $\beta_p$  is accessible. As argued in the proof of statement (b1),  $\Delta_e(W_{e_2}; p)$  will not change after  $s$ . Let  $s^-$  be the last stage before  $s$  at which we define  $\Delta_e(W_{e_2}; p)$ , say at node  $\beta^-$ . Then at stage  $s^-$ ,  $\Delta_e(W_{e_2}; p)[s^-] = \Gamma_{e_1}(AW_{e_2}; p)[s^-]$ . After  $s^-$ , no node to the left of  $\beta^-$  is accessible, otherwise this  $\Delta$  would be cleared. Therefore, the flip point  $d^-$  at  $\beta^-$  is fixed and the finite restraint of amount  $\gamma_{e_1}(AW_{e_2}; d^-)[s^-]$  is permanent on  $A$ . By the assumption on  $s^-$ ,  $W_{e_2}$  will not change below  $\gamma_{e_1}(AW_{e_2}; d^-)[s^-]$  after  $s^-$ . Therefore

$$\begin{aligned} \Delta_e(W_{e_2}; p) &= \Delta_e(W_{e_2}; p)[s^-] \\ &= \Gamma_{e_1}(AW_{e_2}; p)[s^-] \\ &= \Gamma_{e_1}(AW_{e_2}; p). \end{aligned}$$

Thus  $N_e$  is satisfied.

Finally we consider the nondiamond requirements. First notice that the set  $E$  is c.e. Suppose that  $X_{e_1}$  and  $Y_{e_2}$  are candidates for a diamond with base below  $A$ . Since  $X_{e_1} \oplus Y_{e_2}$  is complete, there is an  $e_0$  such that  $\Phi_{e_0}(X_{e_1} Y_{e_2}) = E$ . Consider the requirement  $R_e$  where  $e = \langle e_0, e_1, e_2 \rangle$ . Let  $\alpha$  be the unique node on  $TP$  labelled  $R_e$  as in Lemma 5.2. Then by statement (c) in Lemma 5.3,  $\alpha \hat{\ } \infty \subset TP$ . For any subrequirement  $S_{e,i,j}$ , let  $\beta_{i,j}$  be the unique node on  $TP$  working for  $R_e$ . By statement (c1) in Lemma 5.3,  $\beta_{i,j} \hat{\ } (-1) \notin TP$ .

Again we consider two cases based on whether  $R_e$  has a global  $\Sigma_3$ -outcome.

*Case 1.* There is a node  $\beta_{i,j}$  such that  $\beta_{i,j} \hat{\infty}_x \subset TP$  or  $\beta_{i,j} \hat{\infty}_y \subset TP$ .

Then by statements (c2) and (c4) in Lemma 5.3, either  $X_{e_1} \leq_T A$  or  $Y_{e_2} \leq_T A$ .

*Case 2.* The negation of Case 1.

Then by the assignment of requirements, either there is an  $i$  such that for all  $j$ ,  $S_{e,i,j}$  is satisfied via outcome  $0_c$  or  $1_c$ ; or for all  $i$  there is a  $j$  such that  $S_{e,i,j}$  is satisfied via outcome  $0_b$  or  $1_b$ . In the former case, by statements (c5) and (c7) in Lemma 5.3,  $C_e \not\leq_T A$ ; in the latter case, by statements (c4) and (c6) in Lemma 5.3,  $B_e \not\leq_T A$ .

It remains to show that the set  $B_e$  and  $C_e$  are computable from  $X_{e_1}$  and  $Y_{e_2}$ . Let  $s_0$  be the stage after which  $R_e$  is never initialized. All stage below are larger than  $s_0$ . We drop the indices during the discussion.

Fix a number  $z$ . First notice that we can compute whether or not  $z$  is chosen as a witness: We just wait for a stage at which some number  $z' > z$  appeared in the construction; if by that stage  $z$  has not been chosen as a witness, then  $z$  will never be, since we always choose witness fresh. If  $z$  is not chosen as a witness, then  $z$  will not be in  $C$  or  $B$ . Without loss of generality, let us assume that  $z$  is a witness chosen at node  $\tau$  and targeting the correct set.

To decide whether or not  $z$  is in  $C$  from  $X$ , just wait until a stage  $t$  at which  $X \upharpoonright z = X_t \upharpoonright z$ , then  $z \in C$  if and only if  $z \in C_t$ .

To decide whether or not  $z$  is in  $C$  from  $Y$ , wait until stage  $t$  at which  $Y \upharpoonright z = Y_t \upharpoonright z$ . At stage  $t$ , if no link  $(\alpha, \tau)$  exists at stage  $t$ , then  $z \in C$  if and only if  $z \in C_t$ . If the link  $(\alpha, \tau)$  exists at stage  $t$ , then wait for the next  $\alpha$ -expansionary stage  $t'$ , which exists since  $\alpha \hat{\infty} \subset TP$ . Then  $z \in C$  if and only if  $z \in C_{t'}$ .

To decide whether or not  $z$  is in  $B$  from  $Y$ , just wait until a stage  $t$  at which  $Y \upharpoonright z = Y_t \upharpoonright z$ , then  $z \in B$  if and only if  $z \in B_t$ .

To decide whether or not  $z$  is in  $B$  from  $X$ , wait until stage  $t$  at which  $X \upharpoonright z = X_t \upharpoonright z$ . At stage  $t$ , if no link  $(\alpha, \tau)$  exists at stage  $t$ , then  $z \in B$  if and only if  $z \in B_t$ . If the link  $(\alpha, \tau)$  exists at stage  $t$ , then wait for the next  $\alpha$ -expansionary stage  $t'$ , which exists since  $\alpha \hat{\infty} \subset TP$ . Then  $z \in B$  if and only if  $z \in B_{t'}$ .  $\square$

We end this paper with the following open problem:

**Open Problem:** Is there a high c.e. degree which bounds no diamond base?

## REFERENCES

- [1] K. Ambos-Spies. An extension of the non-diamond theorem in classical and  $\alpha$ -recursion theory. *J. Symbolic Logic*, 49: 586–607, 1984.

- [2] K. Ambos-Spies, C. G. Jockusch, Jr., R. A. Shore, and R. I. Soare. An algebraic decomposition of the recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees. *Trans. Amer. Math. Soc.*, 281(1): 109–128, 1984.
- [3] S. Barry Cooper and Angsheng Li. A nondiamond theorem, and diamond base. preprint.
- [4] S. Barry Cooper, Angsheng Li, and Xiaoding Yi. On the distribution of Lachlan nonsplitting bases. *Arch. for Math. Logic*, to appear.
- [5] A. H. Lachlan. Lower bounds for pairs of recursively enumerable degrees. *Proc. London Math. Soc. (3)*, 16: 537–569, 1966.
- [6] André Nies, Richard A. Shore, and Theodore A. Slaman. Interpretability and definability in the recursively enumerable degrees. *Proc. London Math. Soc. (3)*, 77(2):241–291, 1998.
- [7] Richard A. Shore. Natural definability in degree structures. In M. Lerman P. Cholak, S. Lempp and R. A. Shore, editors, *Computability Theory and Its Applications: Current Trends and Open Problems*, volume 257 of *Contemporary Mathematics*, pages 255–271, Providence RI, 2000. AMS.
- [8] Richard A. Shore and Yue Yang. A nonlow<sub>2</sub> r.e. degree with the extension of embeddings properties of a low<sub>2</sub> degree. preprint.
- [9] Robert I. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, Heidelberg, 1987.
- [10] C. E. M. Yates. A minimal pair of recursively enumerable degrees. *J. Symbolic Logic*, 31:159–168, 1966.

INSTITUTE OF SOFTWARE, ACADEMIA SINICA, BEIJING, CHINA.

*Current address:* Department of Pure Mathematics, School of Mathematics, University of Leeds, Leeds, LS2 9JT, U.K.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF CALIFORNIA, BERKELEY, CA 94720, U.S.A.

DEPARTMENT OF MATHEMATICS, FACULTY OF SCIENCE, NATIONAL UNIVERSITY OF SINGAPORE, LOWER KENT RIDGE ROAD, SINGAPORE 119260.