FINITE FINAL SEGMENTS OF THE D.C.E. TURING DEGREES

STEFFEN LEMPP, YIQUN LIU, YONG LIU, KENG MENG NG, GUOHUA WU, AND CHENG PENG

Abstract. We prove that every finite distributive lattice is isomorphic to a final segment of the d.c.e. Turing degrees (i.e., the degrees of differences of computably enumerable sets). As a corollary, we are able to infer the undecidability of the ∃∀∃-theory of the d.c.e. degrees in the language of partial ordering.

1. Introduction

A set A is *ω-c.e.* if there are computable functions f and g such that for each x, we have $A(x) = \lim_s f(x,s)$, $f(x,0) = 0$, and $|\{s \mid f(x,s+1) \neq f(x,s)\}| \leq g(x)$. A is *c.e.* if we can choose $g(x) = 1$; A is *d.c.e.* if we can choose $g(x) = 2$. A Turing degree is d.c.e. if it contains a d.c.e. set.

The study of the d.c.e. Turing degrees goes back more than half a century, to the work of Ershov [Er68a, Er68b, Er70] in his development of what is now called the Ershov hierarchy. Cooper [Co71] first established that there is a properly d.c.e. degree, i.e., a Turing degree containing a d.c.e. set but no c.e. set. A breakthrough in the study of the d.c.e. degrees was achieved in the Nondensity Theorem [CHLLS91] quoted below, which established the existence of a maximal incomplete d.c.e. degree, or, equivalently, that the two-element linear order can be embedded into the d.c.e. degrees as a finite final segment. This naturally leads to the question of the isomorphism types of all finite final segments of the d.c.e. degrees. If we restrict our attention to final segments which have a least element, then these must be finite lattices. In this paper, we will show that all finite distributive lattices can be realized as final segments of the d.c.e. degrees, leading also to a new and sharper proof of the undecidability of the theory of the d.c.e. degrees (in the language of partial ordering); in particular, we are able to show that the ∀∃∀-theory of the d.c.e. degrees is undecidable.

Our presentation will assume that the reader is familiar with priority arguments on a tree of strategies, in particular $\mathbf{0}'''$-constructions, as presented in, e.g., Soare [So87]. Familiarity with the proof of the D.C.E. Nondensity Theorem would be helpful to understand the present construction, which builds on this.

2010 *Mathematics Subject Classification.* 03D28.

Key words and phrases. d.c.e. degrees, final segments.

Lempp's research was partially supported by NSF grant DMS-160022 and AMS-Simons Foundation Collaboration Grant 626304. He wishes to thank National University of Singapore and Nanyang Technological University for their hospitality during the time much of this research was carried out. Ng was supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MOE-T2EP20222-0018). Peng's research was partially supported by NSF of China No. 12271264. Wu was supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (RG102/23).

We begin by reviewing some definitions and some notation.

Definition 1.1. Let $\mathcal{L}$ be a finite distributive lattice, and denote the upper semilattice of the d.c.e. degrees by $\mathbf{D}_2$. We say $\mathbf{j} : \mathcal{L} \to \mathbf{D}_2$ *embeds* $\mathcal{L}$ *into* $\mathbf{D}_2$ *as a final segment* if the following holds:

(i) $\mathbf{j}(1) = \mathbf{0}'$;
(ii) $a \leq b$ implies $\mathbf{j}(a) \leq \mathbf{j}(b)$;
(iii) $a \not\leq b$ implies $\mathbf{j}(a) \not\leq \mathbf{j}(b)$; and
(iv) for any d.c.e. degree $\mathbf{u}$, there is some $a \in L$ such that $\mathbf{j}(0) \cup \mathbf{u} = \mathbf{j}(a)$.

The D.C.E. Nondensity Theorem can now be stated as follows:

Theorem 1.2 (D.C.E. Nondensity Theorem (Cooper, Harrington, Lachlan, Lempp, Soare [CHLLS91]))**.** *There is a maximal incomplete d.c.e. Turing degree* $\mathbf{d}$*; in particular, the d.c.e. Turing degree are not densely ordered.*

In other words, the 2-element lattice $\mathcal{L} = \{0, 1\}$ can be embedded into the d.c.e. degrees as a final segment (see Definition 1.1). The case when $\mathcal{L}$ is a Boolean algebra was covered by Lempp, Yiqun Liu, Yong Liu, Ng, Peng, and Wu in [Li17]. This suggests that Theorem 1.2 can be generalized to more general finite lattices. To this end, we prove the following theorem in this paper:

Theorem 1.3. *If $\mathcal{L}$ is a finite distributive lattice, then $\mathcal{L}$ can be embedded into d.c.e. degrees as a final segment.*

We conjecture that this theorem can be extended to a much wider class of finite lattices, in particular to the join-semidistributive or even the so-called interval dismantlable lattices introduced by Adaricheva, Hyndman, Lempp, and Nation [AHLN18]; at this point, extending it to all finite lattices appears out of reach but not impossible.

We note an immediate important consequence of Theorem 1.3, a sharper undecidability result for the first-order theory of the d.c.e. degrees in the language of partial ordering; this theory had previously been proven to be undecidable by Cai/Shore/Slaman [CSS12]; a closer analysis of their proof yields the undecidability of the $\forall\exists\forall\exists$-theory. Our result strengthens this:

Theorem 1.4. *The $\exists\forall\exists$-theory of the d.c.e. degrees in the language of partial ordering is undecidable.*

Proof. The proof is almost exactly the same as the proof for the Turing degrees by Lachlan [La68, Section 3] (see Lerman [Le83, Theorem VI.4.6] for a textbook exposition): Let S be the set of the $\leq$-sentences true of all distributive lattices, and F the set of the $\leq$-sentences true of all finite distributive lattices; then by Ershov/Taitslin [ET63], there is no computable set R with $F \subseteq R \subseteq S$. Let $\theta(x)$ be a $\leq$-formula stating that the interval $[x, 1]$ is a distributive lattice; then by Theorem 1.3, the set H of $\leq$-sentences of the form $\forall x\, (\theta(x) \to \varphi(x))$ is a set of $\exists\forall\exists$-sentences and satisfies $F \subseteq H \subseteq S$. □

We now recall the definitions of Boolean algebras and distributive lattices in Section 2, where we also discuss other useful properties.

2. Basics on Lattice Theory

In this section, we define and cover the relevant basic notions and properties of lattices. We follow [Gr11] for the basic definitions. We restrict ourselves to *finite* lattices with least element 0 and greatest element 1.

A lattice $\mathcal{L}$ can be thought of as a poset $(L, \leq)$, where any finite subset has an infimum and a supremum, or as an algebraic structure $(L, \vee, \wedge)$, where $x \leq y$ iff $x \wedge y = x$ iff $x \vee y = y$.

A lattice $\mathcal{L}$ is finite if $|L|$ is finite. A lattice $\mathcal{L}$ is *distributive* if it satisfies the following for all $a, b, c \in L$:

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c),$$
$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c).$$

For $a \leq b$, we call the set $[a, b] = \{x \in L \mid a \leq x \leq b\}$ an *interval*. In particular, an interval is always a sublattice of $\mathcal{L}$ (note that we do not require $0, 1 \in [a, b]$, i.e., the least and greatest element of $[a, b]$ and of $\mathcal{L}$ need not be the same, respectively).

We write $a \prec b$ if $a < b$ and there is no element c such that $a < c < b$. We call a an *atom* if $0 \prec a$. We call $a \in L$ *join-irreducible* in $\mathcal{L}$ if $a \neq 0$ and $a = b \vee c$ implies $a = b$ or $a = c$. If a is join-irreducible, then we let a_* denote the unique element such that $a_* \prec a$. Let $\mathrm{Ji}(\mathcal{L})$ be the set of all join-irreducibles of $\mathcal{L}$. Let $\mathrm{spec}(a) = \{b \in \mathrm{Ji}(\mathcal{L}) \mid b \leq a\}$.

Let $\mathrm{DownJi}(\mathcal{L})$ be the collection of sets $A \subseteq \mathrm{Ji}(\mathcal{L})$ such that if $x, y \in \mathrm{Ji}(\mathcal{L})$ with $x < y$ and $y \in A$, then $x \in A$. Observe that $(\mathrm{DownJi}(\mathcal{L}), \cap, \cup)$ is a distributive lattice with least element $\varnothing$ and greatest element $\mathrm{Ji}(\mathcal{L})$. The following theorem is worth mentioning.

Theorem 2.1 (see [Gr11])**.** *Let $\mathcal{L}$ be a finite distributive lattice. Then the map*

$$\mathrm{spec} : a \mapsto \mathrm{spec}(a)$$

is an isomorphism between $\mathcal{L}$ and $\mathrm{DownJi}(\mathcal{L})$*.*

In particular, a useful fact to keep in mind is that $\mathrm{spec}(a \vee b) = \mathrm{spec}(a) \cup \mathrm{spec}(b)$. As one of the equivalent definitions of finite Boolean algebras, we say that a finite lattice $\mathcal{L}$ is a *Boolean algebra* if $\mathcal{L}$ is a finite distributive lattice such that for any $c \in \mathrm{Ji}(\mathcal{L})$, $0 \prec c$.

We continue with definitions. An element a is a *complement* of b if $a \vee b = 1$ and $a \wedge b = 0$. (The complement of a need not always exist, even in finite distributive lattices.) An element a^* is a *pseudocomplement* of a if for all x, $a \wedge x = 0$ iff $x \leq a^*$. Unlike the complement of a, if a^* exists, it is unique; but in general, $a^{**} \neq a$. If for all $a \in L$, a^* exists, we call $\mathcal{L}$ *pseudocomplemented*.

The *pseudocomplement of a relative to b* is the (unique) element $a * b$ such that for all x, $a \wedge x \leq b$ iff $x \leq a * b$. If for all $a, b \in L$, $a * b$ exists, we call $\mathcal{L}$ *relatively pseudocomplemented*.

Observe that if $\mathcal{L}$ is a finite distributive lattice, then $\mathcal{L}$ is relatively pseudocomplemented. To see this, let $S = \{x \in L \mid a \wedge x \leq b\}$; this set is nonempty since $b \in S$, and by distributivity, S is closed under join and thus has a greatest element, namely, $a * b$. Note that $a^* = a * 0$, and hence a finite distributive lattice is also pseudocomplemented. It is not difficult to show that if $\mathcal{L}$ is relatively pseudocomplemented, then $\mathcal{L}$ is distributive.

A subset $C = \{a_0, a_1, \ldots, a_n\} \subseteq L$ is a *chain of length* $n+1$ if $a_0 \prec a_1 \prec \cdots \prec a_n$.

Theorem 2.2 (see [Gr11])**.** *Let $\mathcal{L}$ be finite distributive lattice. Then every maximal chain in $\mathcal{L}$ is of length $|\operatorname{Ji}(\mathcal{L})|+1$.*

Thus, every chain from 0 to 1 is of the same length. For a non-example, any maximal chain in the five-element modular nondistributive lattice M_3 is of length 3, but $|\operatorname{Ji}(M_3)|+1=4$. (Here, $M_3=\{0,a_1,a_2,a_3,1\}$ with $0<a_i<1$ and $a_i\not\leq a_j$ for distinct $i,j\in\{1,2,3\}$.)

Moreover, we can define the *rank* of $a\in L$ by setting $\operatorname{rank}(a)=n$ if some chain from 0 to a has length $n+1$ but no chain has length $n+2$. In particular, if $\mathcal{L}$ is finite distributive, then $\operatorname{rank}(a)=|\operatorname{spec}(a)|$. An element a is an atom iff $\operatorname{rank}(a)=1$.

We list some useful facts.

Lemma 2.3. *Let $\mathcal{L}$ be a finite distributive lattice and $a,b,c\in L$.*

(1) $a\leq b*a$.
(2) $a=\bigvee\operatorname{spec}(a)$.
(3) If $a\leq b\leq c$, *then* $(b^*)^{[a,c]}=(b*a)\wedge c$.[1]
(4) If $c\leq a$, *then* $a\wedge(a*c)=c$.
(5) If $a,b\in\operatorname{Ji}(\mathcal{L})$ *and* $a\leq b$, *then* $a*a_*\leq b*b_*$ *and* $b\not\leq a*a_*$.

Proof. (1) Since $b\wedge a\leq a$, we have $a\leq b*a$ by definition.

(2) Clearly, $\bigvee\operatorname{spec}(a)\leq a$. Suppose that $b=\bigvee\operatorname{spec}(a)<a$ and consider $A=\{x\leq a\mid x\not\leq b\}$, so $a\in A$. Any $x\in A$ is join-reducible, hence $x=y\vee z$ for some $y,z<x$, where at least one of them must be in A. Therefore, inductively, we see that A is infinite, a contradiction.

(3) Since $(b*a)\wedge c\in[a,c]$, we work in $[a,c]$ and see that for any $x\in[a,c]$, $b\wedge x=a$ iff $x\leq(b*a)\wedge c$, which follows directly from the definition of $b*a$ and the fact that $x\in[a,c]$.

(4) Since $c\leq a\wedge(a*c)\leq c$.

(5) Observe that $b\wedge(a*a_*)\leq b$. Suppose that $b\wedge(a*a_*)=b$, then $b\leq a*a_*$, which implies $a=a\wedge b\leq a_*$, a contradiction. Hence $b\wedge(a*a_*)\leq b_*$ and so $a*a_*\leq b*b_*$. Suppose that $b\leq a*a_*$, then $a=a\wedge b\leq a_*$, again a contradiction. ☐

3. The Requirements and Conflicts

We introduce a preliminary version of our requirements and show that they suffice for our theorem in Section 3.2. Then we rephrase the S-requirements in Section 3.3 and simplify the R-requirements in Section 3.4. After achieving this final (more useful) version of our requirements, we give three examples in Section 3.5. Finally, we discuss the conflicts between requirements in Section 3.6

3.1. The setup.

Recall Definition 1.1: Given a finite distributive lattice $\mathcal{L}$, our job is to exhibit an embedding $\mathbf{j}:\mathcal{L}\to\mathbf{D}_2$ onto a final segment of the d.c.e. degrees. We will define a map $j:\mathcal{L}\to D_2$ (into the set D_2 of d.c.e. sets) and then let $\mathbf{j}(a)=\deg_T(j(a))$.

For each element $a\in L$, we will construct a d.c.e. set A and map a to $j(a)=A$. The set assigned to 0 will be called E. Now our j will be the following map:

$$j:a\mapsto E\oplus(\bigoplus\{B\mid b\in\operatorname{spec}(a)\})$$

[1] Here, $(b^*)^{[a,c]}$ is the pseudocomplement of b computed in $[a,c]$.

Of course, whether $\mathbf{j}$ embeds $\mathcal{L}$ into $\mathbf{D}_2$ as a final segment depends on our choice of the set E and of the sets A for each $a \in \mathrm{Ji}(\mathcal{L})$. We will construct our sets meeting certain requirements.

We start with a global requirement

$$G : K = \Theta^{j(1)},$$

where Θ is a functional we build. This ensures Definition 1.1(i) that $j(1) \equiv_T K$.

Definition 1.1(ii) is automatic, because if $a \le b$, then $\mathrm{spec}(a) \subseteq \mathrm{spec}(b)$, and so by the definition of j, $j(a) \le_T j(b)$. It is worth noting that $\mathrm{spec}(a \vee b) = \mathrm{spec}(a) \cup \mathrm{spec}(b)$, so we have $j(a \vee b) \equiv_T j(a) \oplus j(b)$. Therefore, $\mathrm{im}(\mathbf{j})$ is a finite upper semilattice with least and greatest element. (Since $\mathcal{L}$ is finite and has a least element, and since $\mathbf{j}$ will be onto a final segment of $\mathbf{D}_2$, $\mathrm{im}(\mathbf{j})$ will also automatically be a lattice.)

Next, we explain how to ensure Definition 1.1(iii) and (iv).

3.2. **Initial version of our requirements.** In order to facilitate notation, we will agree on the following: For an interval $S = [a, b] \subseteq L$, we define

$$j[S] = \{U \in D_2 \mid j(a) \le_T U \le_T j(b)\}.$$

$\mathbf{j}[S]$ is the degree version of $j[S]$. In particular, if $S = \{a\}$ is a singleton, then $\mathbf{j}[S] = \{\deg_T(j(a))\} = \mathbf{j}(a)$.

For all d.c.e. sets U, we have a set of requirements $S_\sigma(U)$, indexed by a finite set of nodes $\sigma \in 2^{<\omega}$. For each partial computable functional Ψ, we also have a set of requirements $R_\sigma(\Psi)$, indexed by the same finite set of nodes $\sigma \in 2^{<\omega}$. For now, our requirements are defined recursively as follows.

Let $L_\lambda = L = [0, 1]$. Suppose that we have defined $L_\sigma = [p_\sigma, q_\sigma]$. If $p_\sigma = q_\sigma$, then we stop. Otherwise, we choose $p_{\sigma 0}$ such that $p_\sigma \prec p_{\sigma 0} \le q_\sigma$. We now let $q_{\sigma 0} = q_\sigma$, $p_{\sigma 1} = p_\sigma$, and $q_{\sigma 1} = (p^*_{\sigma 0})^{L_\sigma}$. Set $L_{\sigma 0} = [p_{\sigma 0}, q_{\sigma 0}]$ and $L_{\sigma 1} = [p_{\sigma 1}, q_{\sigma 1}]$ and continue.

Definition 3.1. We let $T_\mathcal{L} = \{\sigma \mid L_\sigma \text{ is defined}\}$ and $T'_\mathcal{L} = \{\sigma \mid \sigma \text{ not a leaf of } T_\mathcal{L}\}$.

We then state our local requirements for now in preliminary form as follows, for each $\sigma \in T'_\mathcal{L}$, d.c.e. set U and partial computable functional Ψ:

$$S_\sigma(U) : j(p_\sigma) \oplus U \in j[L_\sigma] \to j(p_{\sigma 0}) = \Gamma_\sigma^{j(p_\sigma)\oplus U} \vee U = \Delta_\sigma^{j(q_{\sigma 1})}$$
$$R_\sigma(\Psi) : j(p_{\sigma 0}) \ne \Psi^{j(q_{\sigma 1})}$$

This finishes the definition of our requirements. Thus we have, for each d.c.e. set U and each $\sigma \in T'_\mathcal{L}$, a requirement $S_\sigma(U)$. For each partial computable functional Ψ and each $\sigma \in T'_\mathcal{L}$, we have a requirement $R_\sigma(\Psi)$. When the oracle U is clear from the context, we say that Γ_σ is *total and correct* if $j(p_{\sigma 0}) = \Gamma_\sigma^{j(p_\sigma)\oplus U}$, and that Δ_σ is *total and correct* if $U = \Delta_\sigma^{j(q_{\sigma 1})}$.

In the rest of the section, we prove the following lemma.

Lemma 3.2. *If all requirements are met, then $\mathbf{j} : \mathcal{L} \to \mathbf{D}_2$ embeds $\mathcal{L}$ into $\mathbf{D}_2$ as a final segment.*

We assume that all requirements are met throughout the rest of the section in a sequence of lemmas.

We call $S_\sigma(U)$ *active* if $j(p_\sigma) \oplus U \in j[L_\sigma]$. Otherwise $S_\sigma(U)$ is satisfied trivially.

Lemma 3.3. *Let $S_\sigma(U)$ be active.*

(1) *If Γ_σ is total, then* $j(p_\sigma) \oplus U \equiv_T j(p_{\sigma 0}) \oplus U \in j[L_{\sigma 0}]$*. So if* $|L_{\sigma 0}| > 1$*, then* $S_{\sigma 0}(U)$ *is active.*
(2) *If Δ_σ is total, then* $j(p_\sigma) \oplus U \equiv_T j(p_{\sigma 1}) \oplus U \in j[L_{\sigma 1}]$*. So if* $|L_{\sigma 1}| > 1$*, then* $S_{\sigma 1}(U)$ *is active.*

Proof. (1) We have $j(p_{\sigma 0}) \leq_T j(p_{\sigma 0}) \oplus U \leq_T j(p_\sigma) \oplus U \leq_T j(q_\sigma) = j(q_{\sigma 0})$, where the second reducibility follows from the fact that Γ_σ is total, and the third reducibility follows from the fact that $S_\sigma(U)$ is active. But $p_\sigma \leq p_{\sigma 0}$ implies $j(p_\sigma) \oplus U \leq_T j(p_{\sigma 0}) \oplus U$, so the second reduction is in fact an equivalence.

(2) Δ_σ states that $U \leq_T j(q_{\sigma 1})$. Since $p_{\sigma 1} = p_\sigma \leq q_{\sigma 1}$, we have $j(p_{\sigma 1}) \leq_T j(p_{\sigma 1}) \oplus U \leq_T j(q_{\sigma 1})$. □

Lemma 3.4. *For* $\sigma \in T'_{\mathcal{L}}$*, we have* $L_\sigma = L_{\sigma 0} \sqcup L_{\sigma 1}$ *and* $\mathbf{j}[L_\sigma] = \mathbf{j}[L_{\sigma 0}] \sqcup \mathbf{j}[L_{\sigma 1}]$*.*

Proof. For the first equation, $L_{\sigma 0} \cup L_{\sigma 1} \subseteq L_\sigma$ is obvious.

Given $a \in L_\sigma$, if $p_{\sigma 0} \leq a$, then $a \in L_{\sigma 0}$. Otherwise, $p_{\sigma 0} \wedge a = p_\sigma$, so $a \leq q_{\sigma 1}$ and thus $a \in L_{\sigma 1}$. Hence $L_\sigma = L_{\sigma 0} \cup L_{\sigma 1}$.

Now suppose that $b \in L_{\sigma 0} \cap L_{\sigma 1}$. Then $p_{\sigma 0} \leq b \leq q_{\sigma 1}$, contradicting $q_{\sigma 1} = (p^*_{\sigma 0})^{L_\sigma}$. Hence $L_\sigma = L_{\sigma 0} \sqcup L_{\sigma 1}$.

For the second equation, $\mathbf{j}[L_{\sigma 0}] \cup \mathbf{j}[L_{\sigma 1}] \subseteq \mathbf{j}[L_\sigma]$ is trivial.

Given $\deg_T(E \oplus U) \in \mathbf{j}[L_\sigma]$, we have that the degree of $E \oplus U \equiv_T j(p_\sigma) \oplus U$ is in $\mathbf{j}[L_\sigma]$. So $S_\sigma(U)$ is active. If Γ_σ is total, then

$$E \oplus U \equiv_T j(p_\sigma) \oplus U \equiv_T j(p_{\sigma 0}) \oplus U \in j[L_{\sigma 0}].$$

If Δ_σ is total, then

$$E \oplus U \equiv_T j(p_\sigma) \oplus U \equiv_T j(p_{\sigma 1}) \oplus U \in j[L_{\sigma 1}].$$

Therefore $\mathbf{j}[L_\sigma] = \mathbf{j}[L_{\sigma 0}] \cup \mathbf{j}[L_{\sigma 1}]$.

Now suppose that $\deg_T(E \oplus U) \in \mathbf{j}[L_{\sigma 0}] \cap \mathbf{j}[L_{\sigma 1}]$. Then we would have

$$j(p_{\sigma 0}) \leq_T E \oplus U \leq_T j(q_{\sigma 1}),$$

contradicting the $R_\sigma(\Psi)$-requirements. □

Lemma 3.5. $\mathbf{j}$ *is injective.*

Proof. Suppose that $a \neq b$. Let σ be the longest such that $a, b \in L_\sigma$. So $|L_\sigma| \geq 2$, and thus $L_\sigma = L_{\sigma 0} \sqcup L_{\sigma 1}$. Without loss of generality, we assume $a \in L_{\sigma 0}$ and $b \in L_{\sigma 1}$. Hence $\mathbf{j}(a) \in \mathbf{j}[L_{\sigma 0}]$ and $\mathbf{j}(b) \in \mathbf{j}[L_{\sigma 1}]$ and so $j(a) \not\equiv_T j(b)$. □

Lemma 3.6. *If* $a, b \in L$*, then* $a \not\leq b$ *implies* $j(a) \not\leq_T j(b)$*.*

Proof. We suppose towards a contradiction that $j(a) \leq_T j(b)$. Now let $c = b \vee a$. Note that $b < c$ since otherwise we would have $a \leq b$. So $j(b) <_T j(c)$ since $\mathbf{j}$ is injective and order-preserving. But then

$$j(c) \equiv_T j(b \vee a) \equiv_T j(b) \oplus j(a) \equiv_T j(b) <_T j(c),$$

a contradiction. □

Thus we have Definition 1.1(iii).

Definition 3.7. Call L_σ *U-active* if $j(0) \oplus U \in j[L_\sigma]$.

Lemma 3.8.

(1) *If L_σ is U-active, then* $j(0) \oplus U \equiv_T j(p_\sigma) \oplus U$*. So if* $|L_\sigma| \geq 2$*, then* $S_\sigma(U)$ *is active.*

(2) *Suppose that $S_\sigma(U)$ is active. If Γ_σ is total, then $L_{\sigma 0}$ is U-active. If Δ_σ is total, then $L_{\sigma 1}$ is U-active.*

Proof. (1) We have $j(p_\sigma) \leq_T j(0) \oplus U \leq_T j(p_\sigma) \oplus U$, where the first reducibility follows from the fact that L_σ is U-active.

(2) Apply Lemma 3.3 inductively. □

It is easy to see $C'_{\mathcal{L}}(U) := \{\sigma \in T'_{\mathcal{L}} \mid S_\sigma(U) \text{ is active}\}$ is a maximal chain in $T'_{\mathcal{L}}$. We also have that $C_{\mathcal{L}}(U) := \{\sigma \in T_{\mathcal{L}} \mid L_\sigma \text{ is } U\text{-active}\}$ is a maximal chain in $T_{\mathcal{L}}$. Clearly, $C'_{\mathcal{L}}(U)$ is $C_{\mathcal{L}}(U)$ without its longest node.

Lemma 3.9. *For each d.c.e. set U, there exists $a \in L$ such that $j(0) \oplus U \equiv_T j(a)$.*

Proof. Let σ be the longest string in $C_{\mathcal{L}}(U)$, then $L_\sigma = \{a_\sigma\}$ and $j(0) \oplus U \equiv_T j(a_\sigma)$. □

So we have Definition 1.1(iv), and hence Lemma 3.2 has been proved.

3.3. **The S-requirements rephrased.** We need to rephrase our requirements to better fit our construction. Our initial rephrasing of the S-requirements is summarized in the following

Remark 3.10.

(1) If $S_\sigma(U)$ is active, then $j(0) \oplus U \equiv_T j(p_\sigma) \oplus U$.

Since $j(0) = E$, we can replace the disjunct $j(p_{\sigma 0}) = \Gamma^{j(p_\sigma)\oplus U}$ in the conclusion of the $S_\sigma(U)$-requirement by

$$j(p_{\sigma 0}) = \Gamma^{E \oplus U}.$$

(2) Since $p_\sigma \prec p_{\sigma 0}$, there is a unique $c_\sigma \in \mathrm{Ji}(\mathcal{L})$ such that

$$\mathrm{spec}(p_{\sigma 0}) = \mathrm{spec}(p_\sigma) \cup \{c_\sigma\}$$

and

$$p_{\sigma 0} = p_\sigma \vee c_\sigma.$$

Hence

$$j(p_{\sigma 0}) \equiv_T j(p_\sigma) \oplus C_\sigma.$$

So if $S_\sigma(U)$ is active, we can rephrase the $S_\sigma(U)$-requirement as

$$S_\sigma(U) : C_\sigma = \Gamma^{E \oplus U} \vee U = \Delta_\sigma^{j(q_{\sigma 1})}$$

(3) Since $j(p_{\sigma 0}) \equiv_T j(p_\sigma) \oplus C_\sigma$ and $j(p_\sigma) \leq_T j(q_{\sigma 1})$, we have

$$j(p_{\sigma 0}) \not\leq_T j(q_{\sigma 1}) \text{ iff } C_\sigma \not\leq_T j(q_{\sigma 1}).$$

Hence we can rewrite the requirement $R_\sigma(\Psi)$ as

$$R_\sigma(\Psi) : C_\sigma \neq \Psi^{j(q_{\sigma 1})}.$$

Definition 3.11. For $\sigma \in T_L$, we define $F_\sigma(U)$ recursively as follows,

(1) $F_\lambda(U) = \varnothing$,
(2) $F_{\sigma 0}(U) = F_\sigma(U) \cup \{\Gamma_\sigma\}$,
(3) $F_{\sigma 1}(U) = F_\sigma(U) \cup \{\Delta_\sigma\}$,

where Γ_σ stands for the requirement $C_\sigma = \Gamma_\sigma^{E \oplus U}$, and Δ_σ stands for the requirement $U = \Delta_\sigma^{j(q_{\sigma 1})}$.

We say $F_\sigma(U)$ is *satisfied* if all functionals in $F_\sigma(U)$ are total and correct. Given two functionals Λ_σ and Λ_τ in $F_\eta(U)$, we say Λ_σ is *higher than* Λ_τ (and Λ_τ is *lower than* Λ_σ) if $\sigma \subset \tau$. (So $T_{\mathcal{L}}$ is "growing downward".)

Lemma 3.12. *L_σ is U-active iff $F_\sigma(U)$ is satisfied.*

Proof. We proceed by induction on $\sigma \in T_{\mathcal{L}}$. L_λ is obviously U-active and $F_\lambda(U) = \varnothing$.

(1) $L_{\sigma 0}$ is U-active iff L_σ is U-active and Γ_σ is total iff $F_\sigma(U)$ is satisfied and Γ_σ is total iff $F_{\sigma 0}(U)$ is satisfied.

(2) $L_{\sigma 1}$ is U-active iff L_σ is U-active and Δ_σ is total iff $F_\sigma(U)$ is satisfied and Δ_σ is total iff $F_{\sigma 1}(U)$ is satisfied. □

Definition 3.13. Let $[T_{\mathcal{L}}]$ be the set of all leaves of the finite tree $T_{\mathcal{L}}$ (Definition 3.1). We lexicographically order $[T_{\mathcal{L}}]$, denoted by $<$.

Now we can write our $S(U)$-requirements succinctly as follows:

$$S(U) : (\exists \eta \in [T_{\mathcal{L}}])\ F_\eta(U)$$

Note that we stated the $S(U)$-requirement above in a more compact form; it is equivalent to the previous list of requirements for various $\sigma \in T'_{\mathcal{L}}$ by induction on $|\sigma|$.

$j(q_{\sigma 1})$, as it appeared in Δ_σ, will be simplified in Section 3.4 and Section 3.6.

3.4. **The R-requirements rephrased.** We now want to identify $q_{\sigma 1}$ more carefully for each $\sigma \in T'_{\mathcal{L}}$.

Lemma 3.14. *If $a \le p$ and $a \prec c$ and $c \not\le p$, then $(c \vee p) * c = p * a$*

Proof. Let $x = (c \vee p) * c$ and $y = p * a$. Note right away, by Lemma 2.3(1), that $x \ge c$ and $y \ge a$.

(1) To show that $x \le y$, it suffices to show that $p \wedge x \le a$. Since $p \wedge x \ne c$ (since otherwise $c \le p$), it suffices to show $p \wedge x \le c$ (since then $a \prec c$). Now

$$c = (c \vee p) \wedge x = (c \wedge x) \vee (p \wedge x) = c \vee (p \wedge x),$$

where the first equality follows from Lemma 2.3(4).

Hence $p \wedge x \le c$ as desired.

(2) To show that $y \le x$, we need to show that $(c \vee p) \wedge y \le c$. But

$$(c \vee p) \wedge y = (c \wedge y) \vee (p \wedge y) = (c \wedge y) \vee a = (c \vee a) \wedge (y \vee a) = c \wedge y \le c,$$

where the second equality follows from Lemma 2.3(4). □

Lemma 3.15. *If $a \le p$, $a \prec c$, $p \prec p \vee c$, and c is a join-irreducible, then $(p \vee c) * p = c * a$.*

Proof. Pick $a = a_0 \prec a_1 \prec \cdots \prec a_n = p$. Let $b_i = a_i \vee c$. Then we claim:

(1) $c = b_0 \prec b_1 \prec \cdots \prec b_n = p \vee c$,
(2) $a_i \prec b_i$,
(3) $b_i = a_i \vee b_{i-1}$ for $i \ge 1$.

By an easy induction argument, observe that $\mathrm{spec}(b_i) = \mathrm{spec}(a_i) \cup \{c\}$ for all $i \le n$. This immediately gives (1), (2) and (3).

Now, applying the Lemma 3.14 repeatedly, we obtain

$$\begin{aligned}(p \vee c) * p &= b_n * a_n \\ &= (a_n \vee b_{n-1}) * a_n \\ &= b_{n-1} * a_{n-1} = \cdots = c * a.\end{aligned}$$

□

Now we can compute $q_{\sigma 1}$ explicitly.

Now we have

$$\begin{aligned}q_{\sigma 1} &= (p^*_{\sigma 0})^{L_\sigma} \\ &= (p_{\sigma 0} * p_\sigma) \wedge q_\sigma \\ &= ((p_\sigma \vee c_\sigma) * p_\sigma) \wedge q_\sigma \\ &= (c_\sigma * c_{\sigma,*}) \wedge q_\sigma.\end{aligned}$$

Here, the last equality uses Lemma 3.15 and the fact that $c_{\sigma,*} \leq p_\sigma$.

Definition 3.16. For $\sigma \in T_L$, σ is *special* if $L_\sigma = [p_\sigma, 1]$ (i.e., if $q_\sigma = 1$).

Obviously, σ is special iff $\sigma 0$ is special (since being special is just another way of saying that σ is a string where every bit is 0).

Note that if σ is special, then $q_{\sigma 1} = c_\sigma * c_{\sigma,*}$, which is immediate from $q_\sigma = 1$ for special σ, and which implies the following

Lemma 3.17. *If $c_\tau = c_\sigma$ where σ is special, then $q_{\tau 1} \leq q_{\sigma 1}$.* □

We next show the following

Lemma 3.18. $\mathrm{Ji}(\mathcal{L}) = \{c_\sigma \mid \sigma \in T'_{\mathcal{L}}$ *is special*$\}$.

Proof. For $\sigma \in T_{\mathcal{L}}$, if σ is special, then $\sigma 0$ is special, and inductively we have

$$p_{\sigma 0} = p_\sigma \vee c_\sigma = c_\lambda \vee \cdots \vee c_\sigma.$$

Let σ be the longest special σ in $T'_{\mathcal{L}}$, then

$$p_{\sigma 0} = 1 = c_\lambda \vee \cdots \vee c_\sigma.$$

Therefore $\mathrm{Ji}(\mathcal{L}) = \mathrm{spec}(1) = \{c_\sigma \mid \sigma \in T'_{\mathcal{L}}$ is special$\}$. □

Now consider the requirement $R_\tau(\Psi) : C_\tau \neq \Psi^{j(q_{\tau 1})}$. By Lemma 3.18, there exists a special σ such that $c_\tau = c_\sigma$. Also note that $q_{\tau 1} \leq q_{\sigma 1}$; therefore we only need to keep the requirements

$$R_\sigma(\Psi) : C_\sigma \neq \Psi^{j(q_{\sigma 1})}$$

for the special nodes $\sigma \in T'_{\mathcal{L}}$. But if σ is special, then we have $q_{\sigma 1} = c_\sigma * c_{\sigma,*}$, so we can rewrite our R-requirements as follows:

$$R_c(\Psi) : C \neq \Psi^{j(c * c_*)}$$

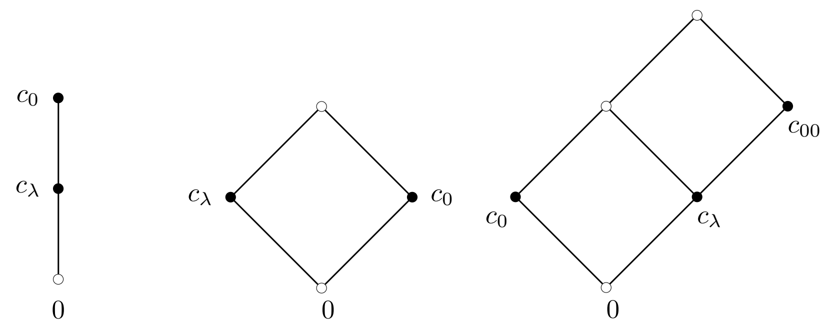
where $c \in \mathrm{Ji}(\mathcal{L})$. (Note here that we are switching from the notation R_{c_σ} to the notation R_c for brevity.)

The final version of our requirements is thus the following:

$$\begin{aligned}G &: K = \Theta^{j(1)} \\ S(U) &: (\exists \eta \in [T_{\mathcal{L}}])\ F_\eta(U) \\ R_c(\Psi) &: C \neq \Psi^{j(c * c_*)} \text{ (for each } c \in \mathrm{Ji}(\mathcal{L}))\end{aligned}$$

where U ranges over all d.c.e. sets and Ψ ranges over all Turing functionals, and where $F_\eta(U)$ was defined in Section 3.3.

3.5. **Three examples.** We give three examples (see Figure 1) and their requirements in this section. In each lattice $\mathcal{L}$, join-irreducible elements are marked by $\bullet$ and also labeled according to $T_{\mathcal{L}}$, the other elements are marked by $\circ$ with the least element labeled by 0. The S-requirements for each lattice are generated as in Figure 2.



(a) 3-element chain (b) diamond lattice (c) 6-element lattice

FIGURE 1. Pictures of lattices

We remark that in Figures 1b and 2b, we write $C_0 = \Gamma_1^{E\oplus U}$ instead of $C_1 = \Gamma_1^{E\oplus U}$ because $c_1 = c_0$. In the same manner, in Figures 1c and 2c, we replace C_1 and C_{01} by C_0 and C_{00}, respectively.

Using Figure 2, we give the requirements for each lattice in Figure 1.

(a) The requirements for the three-element chain (Figure 1a):

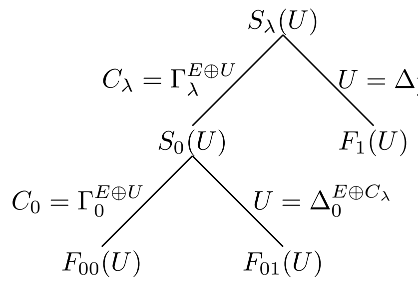
$$\begin{aligned} G &: K = \Theta^{E\oplus C_\lambda\oplus C_0}, \\ S(U) &: F_{00}(U) \vee F_{01}(U) \vee F_1(U), \\ R_{c_\lambda}(\Psi) &: C_\lambda \neq \Psi^E, \\ R_{c_0}(\Psi) &: C_0 \neq \Psi^{E\oplus C_\lambda}. \end{aligned}$$

(b) The requirements for the diamond (Figure 1b):

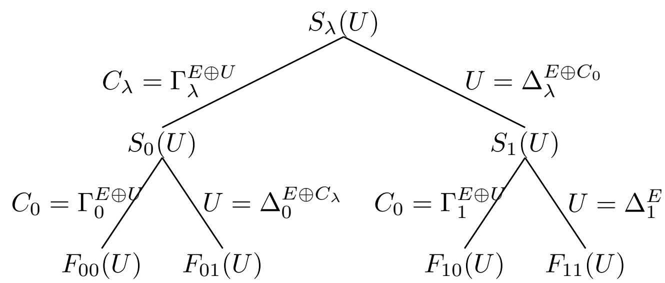
$$\begin{aligned} G &: K = \Theta^{E\oplus C_\lambda\oplus C_0}, \\ S(U) &: F_{00}(U) \vee F_{01}(U) \vee F_{10}(U) \vee F_{11}(U), \\ R_{c_\lambda}(\Psi) &: C_\lambda \neq \Psi^{E\oplus C_0}, \\ R_{c_0}(\Psi) &: C_0 \neq \Psi^{E\oplus C_\lambda}. \end{aligned}$$

(c) The requirements for the six-element lattice (Figure 1c):

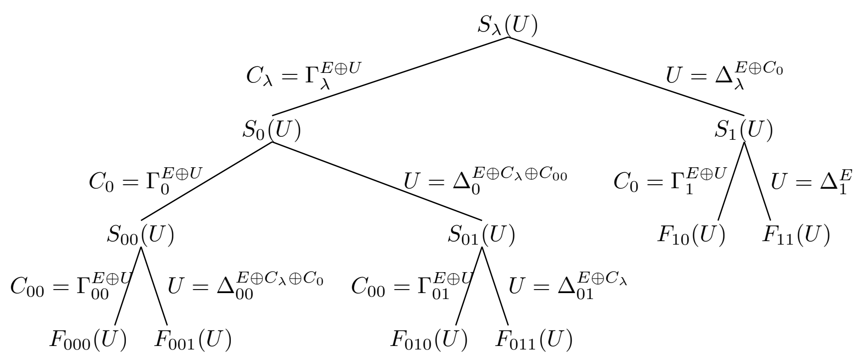
$$\begin{aligned} G &: K = \Theta^{E\oplus C_\lambda\oplus C_0\oplus C_{00}}, \\ S(U) &: F_{000}(U) \vee F_{001}(U) \vee \cdots \vee F_{11}(U), \\ R_{c_\lambda}(\Psi) &: C_\lambda \neq \Psi^{E\oplus C_0}, \\ R_{c_0}(\Psi) &: C_0 \neq \Psi^{E\oplus C_\lambda\oplus C_{00}}, \\ R_{c_{00}}(\Psi) &: C_{00} \neq \Psi^{E\oplus C_\lambda\oplus C_0}. \end{aligned}$$



(a) S-requirements for the 3-element chain



(b) S-requirements for the diamond



(c) S-requirements for the 6-element lattice

FIGURE 2. S-requirements for each lattice

3.6. **A first discussion of the potential conflicts.** Considering the three examples in Figure 1 and their S-requirements in Figure 2, we observe the following properties:

(1) If $c_\sigma < c_\tau$, then C_σ appears above C_τ. For example, in Figure 2c we have C_λ appears above C_{00} but not necessarily above C_0. (See Lemma 3.20.)
(2) The oracle of each Δ_σ is the join of the set E and the sets C_τ with $\tau 0 \subseteq \sigma$ or $\sigma 1 \subseteq \tau$. (See Lemmas 3.21, 3.22, and 3.23.)

These properties are crucial, so we will formally prove them now.

Lemma 3.19. *For $\sigma \in T_L$,* $\operatorname{spec}(p_\sigma) = \{c_\tau \mid \tau 0 \subseteq \sigma\}$.

Proof. The lemma holds for λ since $p_\lambda = 0$ and so $\operatorname{spec}(p_\lambda) = \varnothing$.

Suppose that it holds for σ; then it holds for $\sigma 0$ and $\sigma 1$ because

$$\begin{aligned}\operatorname{spec}(p_{\sigma 0}) &= \operatorname{spec}(p_\sigma) \cup \{c_\sigma\} = \{c_\tau \mid \tau 0 \subseteq \sigma 0\}, \text{ and}\\ \operatorname{spec}(p_{\sigma 1}) &= \operatorname{spec}(p_\sigma) = \{c_\tau \mid \tau 0 \subseteq \sigma\} = \{c_\tau \mid \tau 0 \subseteq \sigma 1\}.\end{aligned}$$

□

Lemma 3.20. *If c is join-irreducible in $\mathcal{L}$ and $c < c_\sigma$, then there is some τ such that $\tau 0 \subseteq \sigma$ and $c = c_\tau$.*

Proof. Recall that $\operatorname{spec}(p_{\sigma 0}) = \operatorname{spec}(p_\sigma) \cup \{c_\sigma\}$. If $c < c_\sigma$, then $c \in \operatorname{spec}(p_{\sigma 0}) \setminus \{c_\sigma\} = \operatorname{spec}(p_\sigma)$, so $c \leq p_\sigma$. Now apply Lemma 3.19. □

Lemma 3.21. *Let $\sigma, \tau \in T_{\mathcal{L}}$. If $\tau 0 \subseteq \sigma$, then $c_\tau \leq q_{\sigma 1}$.*

Proof. By Lemma 3.19, we have $c_\tau \leq p_\sigma \leq q_{\sigma 1}$. □

Lemma 3.22. *Let $\sigma, \tau \in T_{\mathcal{L}}$. If $\sigma 1 \subseteq \tau$, then $c_\tau \leq q_{\sigma 1}$.*

Proof. Since L_τ is a sublattice of $L_{\sigma 1}$, we have $c_\tau \leq p_{\tau 0} \leq q_\tau \leq q_{\sigma 1}$. □

Lemma 3.23. *Let $\sigma \in T_{\mathcal{L}}$ and $\eta = \sigma 10 \cdots 0 \in [T_L]$. Then*

$$\operatorname{spec}(q_{\sigma 1}) = \{c_\tau \mid \tau 0 \subseteq \eta\} = \{c_\tau \mid \tau 0 \subseteq \sigma\} \cup \{c_\tau \mid \sigma 1 \subseteq \tau\}.$$

Proof. Observe that

$$\{c_\tau \mid \tau 0 \subseteq \eta\} \subseteq \{c_\tau \mid \tau 0 \subseteq \sigma\} \cup \{c_\tau \mid \sigma 1 \subseteq \tau\} \subseteq \operatorname{spec}(q_{\sigma 1}),$$

where the second inclusion follows from Lemmas 3.21 and 3.22. So we only need to show $\operatorname{spec}(q_{\sigma 1}) \subseteq \{c_\tau \mid \tau 0 \subseteq \eta\}$. Let $L_{\sigma 1} = [p_{\sigma 1}, q_{\sigma 1}]$. Then $L_\eta = [q_{\sigma 1}, q_{\sigma 1}]$. Hence, $p_\eta = q_{\sigma 1}$, and so $\operatorname{spec}(q_{\sigma 1}) = \operatorname{spec}(p_\eta) = \{c_\tau \mid \tau 0 \subseteq \eta\}$ by Lemma 3.19. □

From Lemma 3.23, we have the following

Lemma 3.24. *Let $\sigma, \tau \in T_{\mathcal{L}}$. If $\sigma 1 \subseteq \tau$, then* $\operatorname{spec}(q_{\tau 1}) \subseteq \operatorname{spec}(q_{\sigma 1})$. □

Next, we consider R-requirements and analyze what happens when an R-requirement tries to diagonalize. In order to give some intuition, we will have to talk about *killing* or *correcting* a functional, the *use* of a computation, and a *witness* of a requirement in the usual sense, but the formal definitions of these are postponed until Section 4.

We will first illustrate these using the example of the six-element lattice in Figure 1c, the S-requirements shown in Figure 2c, and a particular R-requirement.

Suppose that $R_{c_{00}}$ has a witness x and a computation with use $\psi(x)$.

Case 0: $F_{000}(U) = \{\Gamma_\lambda, \Gamma_0, \Gamma_{00}\}$. Then $R_{c_{00}}$ has no conflicts with Γ_λ or Γ_0 since $R_{c_{00}}$ wants to preserve C_λ and C_0, so $R_{c_{00}}$ will not trigger any Γ_λ- or Γ_0-correction. But $R_{c_{00}}$ has a conflict with Γ_{00} since when $R_{c_{00}}$ enumerates x into C_{00}, then $\Gamma_{00}^{E \oplus U}(x)$, intending to compute C_{00}, may require correction by a small number entering E, possibly injuring the computation of $R_{c_{00}}$.

Case 1: $F_{001}(U) = \{\Gamma_\lambda, \Gamma_0, \Delta_{00}\}$. Then, as in Case 0, $R_{c_{00}}$ has no conflicts with Γ_λ or Γ_0. But $R_{c_{00}}$ has a conflict with Δ_{00} since $q_{001} \leq c_{00} * c_{00,*}$ (i.e., the sets appearing in the oracle of Δ_{00} appear also in the oracle of $R_{c_{00}}$), and any correction made by Δ_{00} via a number $\leq \psi(x)$ will injure $R_{c_{00}}$.

Case 2: $F_{010}(U) = \{\Gamma_\lambda, \Delta_0, \Gamma_{01}\}$. Then, as in Case 0, $R_{c_{00}}$ has no conflict with Γ_λ. Also, $R_{c_{00}}$ has no conflict with Δ_0 since C_{00} can be used to correct Δ_0, and $R_{c_{00}}$ itself wants to change C_{00} as well. Finally, $R_{c_{00}}$ has a conflict with Γ_{01} because $c_{01} = c_{00}$.

Case 3: $F_{011}(U) = \{\Gamma_\lambda, \Delta_0, \Delta_{01}\}$. Then, as in Cases 0 and 2, respectively, $R_{c_{00}}$ has no conflict with Γ_λ or Δ_0. Analogously to Case 1 (with Δ_{00}), $R_{c_{00}}$ has a conflict with Δ_{01} since $c_{01} = c_{00}$ and $q_{011} \leq c_{01} * c_{01,*} = c_{00} * c_{00,*}$, where the $\leq$ follows from the calculation above Definition 3.16.

Case 4: $F_{10}(U) = \{\Delta_\lambda, \Gamma_1\}$. Then $R_{c_{00}}$ has a conflict with Δ_λ since the oracle of Δ_λ also appears in the oracle of $R_{c_{00}}$. (To be a little more general, we can show that $c_\lambda < c_{00}$ implies that $q_1 \leq c_{00} * c_{00,*}$ by Lemma 2.3(5) with $a = c_\lambda$ and the calculation above Definition 3.16.) But $R_{c_{00}}$ has no conflict with Γ_1 since $R_{c_{00}}$ wants to preserve $C_1 = C_0$, so $R_{c_{00}}$ will not trigger any Γ_1-correction.

Case 5: $F_{11}(U) = \{\Delta_\lambda, \Delta_1\}$. As in Case 4, $R_{c_{00}}$ has a conflict with Δ_λ; note that $R_{c_{00}}$ also has a conflict with Δ_1 since Δ_1 is lower than Δ_λ and by Lemma 3.24. But in this case, $R_{c_{00}}$ only takes care of Δ_λ since if $R_{c_{00}}$ cannot ensure the correctness of Δ_λ, then Δ_1 won't matter.

In summary,

- $R_{c_{00}}$ has a conflict with Γ_σ iff $c_\sigma = c_{00}$ iff $\sigma = 00$ or $\sigma = 01$.
- In Case 2, since $R_{c_{00}}$ has a conflict with $\Gamma_{01} \in F_{010}(U)$, it has no conflict with Δ_0; this is because $c_{00} = c_{01} \leq q_{01}$ and Δ_0 has oracle $j(q_{01})$.
- In Cases 1, 3, 4, and 5, $R_{c_{00}}$ has no conflict with any Γ, but $R_{c_{00}}$ has conflicts with Δ_{00}, Δ_{01}, and both Δ_λ and Δ_1, respectively. Note here also that $c_{00}, c_{01}, c_\lambda \leq c_{00}$.

We now want to show how to generalize these properties to arbitrary finite distributive lattices. We first formulate the first and third property as a definition.

Definition 3.25 (conflicts)**.**

(1) R_c has *a conflict* with Γ_σ iff $c_\sigma = c$.
(2) R_c has *a conflict* with Δ_τ iff $q_{\tau 1} \leq c * c_*$.

By Lemma 3.20, for any join-irreducible element c in $\mathcal{L}$, if there is some $\Gamma_\tau \in F_\eta(U)$ such that $c \leq c_\tau$ (and so R_c's diagonalization might be destroyed by Γ_τ-correction), then there is some $\Gamma_\sigma \in F_\eta(U)$ with $c_\sigma = c$. Therefore, if for any $\Gamma \in F_\eta(U)$ computing a set C_τ with $c \leq c_\tau$, then R_c has a conflict with some $\Gamma_\sigma \in F_\eta(U)$ with $\sigma \subseteq \tau$; so it suffices to consider this Γ_σ in the definition of conflict.

Given $F_\eta(U)$, suppose that R_c has a conflict with $\Delta_\sigma \in F_\eta(U)$; then for all $\Delta_\tau \in F_\eta(U)$ which are lower than Δ_σ, R_c also has a conflict with Δ_τ because $\mathrm{spec}(q_{\tau 1}) \subseteq \mathrm{spec}(q_{\sigma 1})$ by Lemma 3.24. Therefore, if R_c has a conflict with any $\Delta \in F_\eta(U)$, we need to only consider the highest such Δ_σ.

The following lemma is crucial to our argument.

Lemma 3.26. *For $c \in \mathrm{Ji}(\mathcal{L})$ and $\eta \in [T_L]$, we have:*

(1) If R_c has a conflict with some (necessarily unique) $\Gamma_\tau \in F_\eta(U)$, then for all $\Delta_\sigma \in F_\eta(U)$, $c \leq q_{\sigma 1}$. (Therefore, such Δ_σ can be corrected via the set C.)
(2) Otherwise, there is some $\Delta_\sigma \in F_\eta(U)$ with which R_c has a conflict. For the highest $\Delta_\sigma \in F_\eta(U)$ with which R_c has a conflict, $c_\sigma \leq c$. For all $\Delta_\tau \in F_\eta(U)$ with which R_c has no conflict, we have $c \leq q_{\tau 1}$.

Proof. (1) Recall the definition of $F_\eta(U)$, we have $\tau 0 \subseteq \eta$ and $\sigma 1 \subseteq \eta$. Therefore, we either have $\sigma 1 \subseteq \tau 0$ or $\tau 0 \subseteq \sigma$. In either case, by Lemma 3.23, we have $c = c_\tau \leq q_{\sigma 1}$.

(2) We will proceed by induction on $\sigma \subseteq \eta$ and show:
(a) If $\sigma 1 \subseteq \eta$ and $q_{\sigma 1} \leq c * c_*$, then $c_\sigma \leq c$. We stop the induction.

(b) If $\sigma 1 \subseteq \eta$ and $q_{\sigma 1} \not\leq c * c_*$, then $c \leq q_{\sigma 1}$ and continue with $\sigma 1$.
(c) If $\sigma 0 \subseteq \eta$, then $c_\sigma \neq c$, $c \leq q_{\sigma 0}$ and continue with $\sigma 0$.

Case (a). Suppose towards a contradiction that $c_\sigma \not\leq c$, so $c_\sigma \wedge c \leq c_{\sigma,*}$. Therefore $c \leq c_\sigma * c_{\sigma,*}$. Because of the induction hypothesis in Cases (b) and (c), we have $c \leq q_\sigma$. Together we have

$$c \leq (c_\sigma * c_{\sigma,*}) \wedge q_\sigma = q_{\sigma 1} \leq c * c_*,$$

where the second equality uses the calculation above Definition 3.16, and the last inequality is the assumption of Case (a). Therefore, we obtain $c \leq c * c_*$, contradicting Lemma 2.3(5).

Case (b). Since $q_{\sigma 1} \not\leq c * c_*$, we have $c \wedge q_{\sigma 1} \not\leq c_*$. But $c \wedge q_{\sigma 1} \leq c$, so the only possibility is that $c \wedge q_{\sigma 1} = c$, and hence $c \leq q_{\sigma 1}$.

Case (c). $c \leq q_\sigma = q_{\sigma 0}$ where the first inequality is the induction hypothesis in Cases (b) and (c) and the last equality is by the definition of $q_{\sigma 0}$. $c_\sigma \neq c$ is our assumption for (2) that R_c has no conflicts with any $\Gamma \in F_\eta(U)$.

Now suppose that we never reach Case (a), and so we have $c \leq q_\eta = p_\eta$. By Lemma 3.19, we have $c = c_\tau$ for some $\tau 0 \subseteq \eta$, so R_c would have a conflict with $\Gamma_\tau \in F_\eta(U)$, a contradiction. Thus the induction will end with Case (a), and thus (2) is proved. □

Suppose that R_c has a witness x, a computation with use $\psi(x)$, and a conflict with $\Gamma \in F_\eta(U)$. The strategy that R_c takes would be the following. It will actively kill Γ by enumerating $\gamma(x)$ (possibly $\gamma(x) \leq \psi(x)$) into E and request that $\gamma(x)$ be redefined as fresh (so the new $\gamma(x) >$ the current $\psi(x)$ next time). What R_c hopes for is a point $z \leq \gamma(x)$ such that $U(z)$ changes, and in this case, we are allowed to take the old $\gamma(x)$ out so that the computation of R_c is restored and Γ can be corrected using the latest $\gamma(x)$, not injuring $\psi(x)$. Note that after R_c kills Γ_σ, we will attempt to make $F_{\sigma 100\cdots 0}(U)$ satisfied.

Suppose that R_c finds no conflict with any $\Gamma \in F_\eta(U)$. Then let Δ_σ be the highest one with which R_c has a conflict. R_c would like to know who is responsible for building this Δ_σ. By the preceding paragraph, R_{c_σ} must be responsible for this, where we also have $c_\sigma \leq c$.

This motivates the following definition.

Definition 3.27. Recall that for each $c \in \mathrm{Ji}(\mathcal{L})$, we have R_c-requirements. For each R_c-requirement, based on the conflicts explained above, we define the nondecreasing map $R_c : [T_\mathcal{L}] \to [T_\mathcal{L}]$ (Definition 3.1) by

$$R_c(\eta) = \begin{cases} \sigma 10\dots 0 & \text{if } R_c \text{ has a conflict with } \Gamma_\sigma \in F_\eta(U), \\ \eta & \text{otherwise.} \end{cases}$$

Examples are given in Table 1. In this table, we list $[T_L]$ lexicographically as $\eta_0 < \eta_1 < \cdots < \eta_{|\mathcal{L}|}$. We have, for example, $(R_{c_0} \circ R_{c_\lambda} \circ R_{c_{00}} \circ R_{c_0} \circ R_{c_{00}})(\eta_0) = \eta_5$ for the six-element lattice.

This completes the discussion of the lattice-theoretic aspects of our construction.

		η_0	η_1	η_2	η_3	η_4	η_5
3-element chain	R_{c_λ}	η_2	η_2	η_2			
	R_{c_0}	η_1	η_1	η_2			
diamond	R_{c_λ}	η_2	η_2	η_2	η_3		
	R_{c_0}	η_1	η_1	η_3	η_3		
6-element lattice	R_{c_λ}	η_4	η_4	η_4	η_4	η_4	η_5
	R_{c_0}	η_2	η_2	η_2	η_3	η_5	η_5
	$R_{c_{00}}$	η_1	η_1	η_3	η_3	η_4	η_5

Table 1. The R-maps for our three examples

4. A Single U-set

4.1. **Introduction.** Let $\mathcal{L}$ be the finite distributive lattice with least element 0 and greatest element 1. Recall that our requirements take the following form:

$$G : K = \Theta^{j(1)}$$
$$S(U) : \exists \eta \in [T_{\mathcal{L}}], F_\eta(U)$$
$$R_c(\Psi_e) : C \neq \Psi_e^{j(c*c_*)}$$

where U ranges over all d.c.e. sets, Ψ_e is the e-th Turing functional and e ranges over ω, c ranges over $\mathrm{Ji}(\mathcal{L})$, and G is a single global requirement.

The goal of this section is to present the conflicts of the G-requirement, one single $S(U)$-requirement (for an arbitrary fixed d.c.e. set U), and all R-requirements. We note one unusual feature of our construction: Unlike in other constructions, we will have to try to meet the same requirement repeatedly without any apparent gain until we succeed, simply to have a sufficient number of strategies in the right arrangement.

We will denote the number of potential tries by m for now. Note that the optimal value of m depends on the lattice $\mathcal{L}$ only. A careful analysis into the lattice structure of $\mathcal{L}$ may give us the optimal value. $m = 1$ is optimal for Boolean algebras, $m = 2$ is optimal for the 3-element chain (Figure 1b), but setting $m = |\mathcal{L}| + 1$ is always sufficient for our construction.

This section will introduce the computability-theoretic aspects of our construction (using a general finite distributive lattice instead of examples) in the simplest combinatorial setting, preparing us for the more challenging full setup with all sets U_j.

4.2. **The priority tree.** For each $c \in \mathrm{Ji}(\mathcal{L})$, recall the map $R_c : [T_{\mathcal{L}}] \to [T_{\mathcal{L}}]$ from Definition 3.27; for now, the only property of R_c that we need is that it is a nondecreasing map. We also fix a computable list $\{R^e\}_{e\in\omega}$ of all R-requirements $\{R_c(\Psi_j)\}_{j\in\omega, c\in \mathrm{Ji}(\mathcal{L})}$.

The functionals in $F_\xi(U)$ for each $\xi \in [T_{\mathcal{L}}]$ will be referred to as *U-functionals.* A node α on the tree will be called an *R-node* if it is assigned to an R-requirement; and an *S-node* if it is assigned to an S-requirement with pair $(a, \xi) \in \{0, \dots, m-1\} \times [T_{\mathcal{L}}]$. We view an S-node as the a-th copy of one of the S_U-strategies. We order $\{0, \dots, m-1\} \times [T_{\mathcal{L}}]$ lexicographically. The intuition behind the notation (a, ξ) is that we work our way through all the necessary $S(U)$-strategies until there is no longer a Γ-functional to attack (with $a = 0$, i.e., the first time). Then we start this whole process again with $a = 1$, with $a = 2$, etc., until we reach $a = m - 1$. At this point, we will be sure that we have a sufficient number of S_U-strategies so that we can deal with any possible U-changes back and forth, as we will have to prove in the end.

An S-node assigned to (a, ξ) has only one outcome, 0. An R-node α has a w-outcome, a d-outcome, possibly one of two types of U-outcomes, and finally a ctr-outcome. The intuition (which will become clear later) for the outcomes is the following: Initially, α keeps visiting its w-outcome while α is looking for some computation to converge. Next, having found a computation, α is now ready to visit the next outcome, the U-outcome, if any, meaning that α is dealing with S_U-functionals. In case that no U-outcome is available for α, the next outcome that α visits is the ctr-outcome, meaning that α has gathered enough information and becomes a controller. If α is successful in its diagonalization, then the d-outcome is said to be active, and α visits the d-outcome. If the d-outcome is inactive, then α will visit its outcomes in sequence: It always starts with the w-outcome, then possibly a U-outcome (if any), and then the ctr-outcome, at which point the ctr-outcome becomes active.

Definition 4.1. We define the priority tree $\mathcal{T}$ by recursion: We assign the root node λ to $(0, \iota)$ (ι is the string $00\cdots0$ of proper length) and call it an S-node.

Suppose that α is an S-node assigned to (a, ξ). We determine the least e such that there is no R^e-node $\beta \subset \alpha$ with $\beta^\frown w \subseteq \alpha$ or $\beta^\frown d \subseteq \alpha$, and we assign $\alpha^\frown 0$ to R_e.

Suppose that α is an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$. Let $\beta \subset \alpha$ be the longest S-node, which is assigned to (a, ξ), say. We add outcomes to α in the following sequence:

(1) We add a U-outcome as its *first outcome*.
   (a) If $\xi < R_c(\xi)$, then we say the U-outcome is a *Type I* outcome, and we assign $\alpha^\frown U$ to $(a, R_c(\xi))$. The *next* outcome is placed just to the left of this U-outcome.
   (b) If $\xi = R_c(\xi)$, then we say the U-outcome is a *Type II* outcome. If $a < m-1$, then we say that this U-outcome is *GREEN*, and we assign $\alpha^\frown U$ to $(a+1, \iota)$. If $a = m-1$, then we say that this U-outcome is *RED*, and we do not assign $\alpha^\frown U$ to any requirement (so it is a terminal node on the tree). In either case, the *next* outcome is placed just to the right of this U-outcome. (We caution the reader here that the definition of GREEN and RED is *not* static. A GREEN outcome can become RED and vice versa. The information below the RED outcome will be frozen for a while, waiting for another piece of information to wake up, at which time the RED turns GREEN again. We will never visit a RED outcome directly.)
(2) We add a ctr-outcome as the *second outcome*. We do not assign $\alpha^\frown$ ctr to any requirement (so it is a terminal node on the tree).
(3) Finally, we add w- and d-outcomes to the right of all existing outcomes and assign both $\alpha^\frown w$ and $\alpha^\frown d$ to (a, ξ). (Note that this will introduce "dummy" S_U-strategies that are not strictly needed; however, this will no longer be possible when we have infinitely many sets U_j).

The order of the outcomes is $\text{ctr} < U < w < d$ if the U-outcome is Type I, and $U < \text{ctr} < w < d$ if the U-outcome is Type II. For $\alpha, \beta \in \mathcal{T}$, α has *higher priority* than β (denoted by $\alpha <_P \beta$) if α is to the left of β or α is a proper initial segment of β.

This finishes the definition of the priority tree $\mathcal{T}$.

We need some additional auxiliary notions:

- The ctr- and d-outcomes can be *active* or *inactive*.
- α is a *controller* iff $\alpha^\frown \text{ctr}$ is active.
- Only the *Type II* outcome can be GREEN or RED. A GREEN outcome can become RED and vice versa.
- α^- is the longest node such that $\alpha^- \subsetneq \alpha$.
- For an S-node α assigned to (a, ξ), we set $\text{seq}(\alpha) = (a, \xi)$, $\text{seq}_0(\alpha) = a$, and $\text{seq}_1(\alpha) = \xi$. (Note that if α is an R-node, then α^- will always be an S-node in our priority tree.)
- For an R-node α with $\text{seq}_0(\alpha^-) = a$, we say α is *dealing with the a-th copy of the U-functionals*. (The b-th copy of the U-functionals is irrelevant to α if $b \neq a$.)
- For an R-node, the *next outcome* of a particular outcome is well-defined (except for ctr-, w- and d-outcomes): It is the next outcome added to this R-node after the particular outcome is added.
- For each R-node β, we have a threshold point $\text{threshold}(\beta)$ associated to β, and the diagonalizing witness associated to the w-outcome of β, denoted by $\text{witness}(\beta)$.

An example of the priority tree for the three-element lattice (see Figures 1a and 2a) can be found in Figure 3 with $m = 2$. $[T_{\mathcal{L}}]$ is listed lexicographically as $\eta_0 < \eta_1 < \eta_2$. In Figure 3, the S-node assigned to (a, η_j) is denoted by aj for short.

Some of the w- and d-outcomes are hidden, and so are the labels of U-outcomes. A Type II U-outcome is denoted by a thick line. The U-outcomes of R^3, R^4, and R^7 are RED. A terminal node is denoted by a $\bullet$. A ctr-outcome is denoted by a slim line and a $\bullet$.

Remark 4.2. (1) One may not want to add ctr-outcomes in the first place as we do not even assign $\alpha^\frown \text{ctr}$ to any requirements. However, adding ctr-outcomes reveals its priority clearly. Suppose that $\alpha^\frown \text{ctr}$ becomes active. If $\text{ctr} < U$, then nodes below $\alpha^\frown U$ will be initialized. If $U < \text{ctr}$, nodes below $\alpha^\frown U$ *must not* be initialized. If α and β are two controllers, then α has higher priority than β if $\alpha^\frown \text{ctr} <_P \beta^\frown \text{ctr}$.

(2) The reason why we add a RED U-outcome to α even though we do not even assign $\alpha^\frown U$ to any requirement is to make the construction more uniform. For example, in Figure 3, when the U-outcome of R^1 becomes RED, the strategy of R^1 will be the same as R^3.

(3) For those who are familiar with the proof of D.C.E. Nondensity Theorem [CHLLS91], our notion of controller is dynamic and generalizes theirs.

(4) The letter U in *U-outcome* really has the same meaning as the letter U in the S_U-node just above α. In Section 5 when we have indexed sets U_j, we will have U_j-outcomes, or simply j-outcomes.

The following lemma states that all requirements are represented by some node along any infinite path through $\mathcal{T}$.

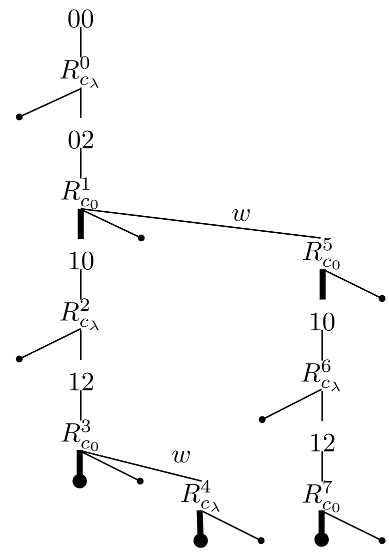


FIGURE 3. The priority tree for 3-element chain

Lemma 4.3. *Let p be an infinite path through $\mathcal{T}$. Then*

(1) *there is an S-node α such that for each S-node β with $\alpha \subset \beta \subset p$, we have* $\operatorname{seq}(\alpha) = \operatorname{seq}(\beta)$, *and*
(2) *for each e, there is an R^e-node α such that either $\alpha^\frown d \subset p$ or $\alpha^\frown w \subset p$.*

Proof. (1) Consider $\operatorname{seq}(\alpha)$ for all S-nodes $\alpha \subset p$, which is nondecreasing for $\alpha \subset p$; now note that $m \times [T_\mathcal{L}]$ is finite.

(2) We proceed by induction on e. Suppose that α is assigned to R^e. Suppose that for all R^e-nodes β with $\alpha \subseteq \beta \subset p$, we have $\beta^\frown U \subset p$, then the value of $\operatorname{seq}(\gamma)$ for all S-nodes γ with $\alpha \subset \gamma \subset p$ would be strictly increasing. But $m \times [T_\mathcal{L}]$ is finite. □

4.2.1. *Functionals manipulated at S-nodes and at R-nodes.* An S-node β assigned to (b, ξ) intends to ensure that the b-th copies of all functionals in $F_\xi(U)$ are correct and total. However, not all functionals in $F_\xi(U)$ are maintained at β. β only builds and maintains each Γ-functional in $\operatorname{Maintain}(\beta, U)$, defined as follows.

Definition 4.4. Let β be an S-node with $\operatorname{seq}(\beta) = (b, \xi)$ and $\alpha = (\beta^-)^-$, if it exists, with $\operatorname{seq}(\alpha) = (a, \eta)$.

- If α does not exist, then $\operatorname{Maintain}(\beta, U) = F_\xi(U)$. (In fact, $\xi = 00 \cdots 0$ and $b = 0$.)
- If $a = b$ and $\eta = \xi$, then $\operatorname{Maintain}(\beta, U) = \varnothing$.
- If $a = b$ and $\eta < \xi$, then for some σ we have $\sigma 0 \subseteq \eta$ and $\sigma 100 \cdots 0 = \xi$, and we let $\operatorname{Maintain}(\beta, U) = \{\Gamma_\tau \mid \sigma 1 \subseteq \tau \subseteq \xi\}$.
- If $a + 1 = b$, then $\operatorname{Maintain}(\beta, U) = F_\xi(U)$. (In fact, $\xi = 00 \cdots 0$.)

An R-node β with $\operatorname{seq}(\beta^-) = (a, \eta)$ ensures that some of the functionals in $F_\eta(U)$ are properly killed (see Section 4.3 for details), and that possibly one Δ-functional is correct and total, depending on which outcome β is visiting.

Definition 4.5. Let β be an R-node with $\text{seq}(\beta^-) = (a, \eta)$, and suppose, if $\beta^\frown U$ is not a terminal node, that $\text{seq}(\beta^\frown U) = (b, \xi)$.

- If $a = b$, then $\text{Kill}(\beta, U) = F_\eta(U) \setminus F_\xi(U)$ and $\text{Maintain}(\beta, U) = \{\Delta\}$ for the unique $\Delta \in F_\xi(U) \setminus F_\eta(U)$.
- If $a < b$, then $\text{Kill}(\beta, U) = F_\eta(U)$ and $\text{Maintain}(\beta, U) = \varnothing$.

If $\beta^\frown U$ is a terminal node, then $\text{Kill}(\beta, U) = \text{Maintain}(\beta, U) = \varnothing$. We also define $\text{Kill}(\beta, d) = \text{Maintain}(\beta, d) = \text{Kill}(\beta, w) = \text{Maintain}(\beta, w) = \varnothing$.

In summary, an R-node β visiting its U-outcome has to kill each functional in $\text{Kill}(\beta, U)$, and build and maintain the Δ-functional, if any, in $\text{Maintain}(\beta, U)$.

4.2.2. *β^* and $\beta^\sharp$.* Now we are ready to formulate the crucial definition for our construction. Consider an R_c-node β. Suppose that $\text{seq}(\beta^-) = (a, \eta)$ and that $\eta = R_c(\eta)$. By Lemma 3.26(2), we know that there is a highest $\Delta_\sigma \in F_\eta(U)$ with which R_c has a conflict. Let α be the R_{c_σ}-node such that $\alpha^\frown U \subseteq \beta^-$ and α builds (the a-th copy of) this Δ_σ (along this U-outcome). We then define

$$\beta^* = \alpha.$$

Two properties will be used throughout the paper: The first one is that $c_\sigma \leq c$; The second one is that the set C does not appear in the oracle of R_{c_σ} (Lemma 2.3 5).

Suppose that $\text{seq}(\beta^-) = (a, \eta)$ with $a \neq 0$. Let α be the R_d-node with $\alpha \subset \alpha^\frown U \subseteq \beta$, where the U-outcome is a Type II outcome and where $seq(\alpha^\frown U) = (a, \iota) \in m \times [T_{\mathcal{L}}]$ ($\iota = 00 \cdots 0$). We define

$$\beta^\sharp = \alpha.$$

If $a = 0$ then we do not define $\beta^\sharp$.

Remark 4.6.

(1) In fact, we should have written β^{*U} instead of β^* because we are discussing U-functionals. Once we consider several sets U_j, we will write β^{*j} instead. If the index is clear from the context, we simply write β^*. We proceed analogously for $\beta^\sharp$.

(2) If $\beta^\sharp$ is defined, then so is $(\beta^\sharp)^*$.

(3) The intuition is the following: Let β, an R_c-node, be a controller without a Type I U-outcome. Then β^*, an R_d-node, is defined and $\beta^* \subsetneq \beta$. Suppose that β and β^* are R_c-strategies for the same c. Informally, if there is a U-change, we make $\beta^{*\frown} d$ active; otherwise, $\beta^\frown d$ is active. Suppose that they are R-strategies with different c; then we will wait for the stage such that $\alpha = (\beta^*)^\sharp$ becomes a controller and check if α and α^* are R_c-strategies for the same c. When $(\beta^*)^\sharp$ is not defined and the process cannot be continued, then some R-node with a Type I U-outcome is ready to become a controller (and this will be discussed later).

(4) As an illustration, in Figure 3, we have $(R^4_{c_\lambda})^* = (R^3_{c_0})^* = R^2_{c_\lambda}$, $(R^2_{c_\lambda})^\sharp = R^1_{c_0}$, $(R^1_{c_0})^* = R^0_{c_\lambda}$, and $(R^0_{c_\lambda})^\sharp$ is not defined.

4.3. **Preliminaries.** We cover some standard notions and shorthand notations in this section. Given a set X of natural numbers, we usually think of it as an infinite binary string. $X \restriction l = \sigma$ if the length of σ is l and for each $n < l$, $X(n) = \sigma(n)$. As usual, $X_0 \oplus \cdots \oplus X_{k-1}(kn + i) = X_i(n)$ for each $i < k$. However, by a slight abuse of notation, we let $(X_0 \oplus \cdots \oplus X_{k-1}) \restriction l = X_0 \restriction l \oplus \cdots \oplus X_k \restriction l$, and the length of $(X_0 \oplus \cdots \oplus X_{k-1}) \restriction l$ is defined to be l.

For a d.c.e. set $U = W_i \setminus W_j$ for some c.e. sets W_i and W_j, we write U_s for $W_{i,s} \setminus W_{j,s}$.

We will use $\Gamma, \Delta, \Phi, \Psi$ to denote Turing functionals. A Turing functional Φ is a c.e. set consisting of "axioms" of the form (x, i, σ), where $x \in \omega$ is the input, $i \in \{0, 1\}$ the output, and $\sigma \in 2^{<\omega}$ the oracle use; so $(x, i, \sigma) \in \Phi$ denotes that $\Phi^\sigma(x) = i$. Furthermore, if $(x, i, \sigma), (x, j, \tau) \in \Phi$ for comparable σ and τ, we require $i = j$ and $\tau = \sigma$. For $X \subseteq \omega$, we write $\Phi^X(x) \downarrow = i$ if $(x, i, X \restriction l) \in \Phi$ for some l (the *use function* $\varphi(x)$ is defined to be the least such l); $\Phi^X(x) \uparrow$ if for each i and l we have $(x, i, X \restriction l) \notin \Phi$. x is a *divergent point* of Φ^X if $\Phi^X(x) \uparrow$. If Φ_s is a c.e. enumeration of Φ, where each Φ_s is a finite subset of Φ, these notions apply accordingly to Φ_s.

For a Turing functional Φ that is constructed by us stage by stage, suppose that our intention is ensure $\Phi^{X_0 \oplus \cdots \oplus X_{k-1}} = Y$ where X_i or Y could be either a set with given fixed enumeration or a set that is to be constructed by us. At stage s, we say that $\Phi_s(n)$ (omitting the oracles and the set Y) is *correct* if $\Phi_s^{X_{0,s} \oplus \cdots \oplus X_{k-1,s}}(n) \downarrow = Y_s(n)$; *incorrect* if $\Phi_s^{X_{0,s} \oplus \cdots \oplus X_{k-1,s}}(n) \downarrow \neq Y_s(n)$; and *undefined* if $\Phi_s^{X_{0,s} \oplus \cdots \oplus X_{k-1,s}}(n) \uparrow$. Suppose that $\Phi_s^{X_{0,s} \oplus \cdots \oplus X_{k-1,s}}(n)$ is undefined; then *defining* $\Phi(n)$ *with use* u means that we enumerate $(n, Y_s(n), (X_{0,s} \oplus \cdots \oplus X_{k-1,s}) \restriction u) \in \Phi_s$ so that $\Phi_s^{X_{0,s} \oplus \cdots \oplus X_{k-1,s}}(n)$ becomes correct. Note that whether $\Phi_s(n)$ is correct, incorrect, or undefined depends on a particular substage of stage s, but this is usually clear from the context.

4.4. **Use blocks.** Use blocks are the main source of both verbal and mathematical complexity of our construction.

Consider a Γ-functional that belongs to Maintain(α, U) for some S-node α (Definition 4.4) and intends to ensure $\Gamma^{E \oplus U} = C$ for some $c \in \text{Ji}(\mathcal{L})$. Suppose α is being visited at stage s, we define $\Gamma(n)$ *with use block* $\mathsf{B} = [u - l, u)$ means that we define $\Gamma(n)$ with use u and reserve the *use block* B for future use. We also say that B is *defined for* $\Gamma(n)$ *by* α at stage s; B *belongs to* Γ; B is *for* $\Gamma(n)$; B is *maintained by* α. If the use block B is a *fresh* use block, we define $\mathbf{B}_{\langle s|}(\gamma, n) = \mathsf{B}$ and Created$(\mathsf{B}) = s$.

Remark 4.7. It will be seen (Section 4.15: visit(α) for S-node) that $\mathbf{B}_{\langle s|}(\gamma, n)$ is well defined because we will not define $\Gamma(n)$ twice at a single stage.

The use block $\mathsf{B} = [u - l, u)$ is viewed as a *potential subset* of E. Enumerating (or extracting) a point k with $u - l \leq k < u$ in B means letting $E(k) = 1$ ($E(k) = 0$, respectively). Similar to the use function, the *use block function* $\mathbf{B}_s(\gamma, n)$ is defined to be the use block B with $\gamma_s(n) = \max \mathsf{B} + 1$ if $\gamma_s(n) \downarrow$; undefined if $\gamma_s(n) \uparrow$. Different from the notion $\mathbf{B}_{\langle s|}(\gamma, n)$, the notion $\mathbf{B}_s(\gamma, n)$ depends on a particular substage of stage s; conventionally, it is usually evaluated when α is being visited and will be clear from the context.

A use block B defined at stage $t < s$ ($\mathsf{B} = \mathbf{B}_{\langle t|}(\gamma, n)$ for some n) is *available for correcting* n at stage s if $\mathsf{B} = \mathbf{B}_s(\gamma, n)$. The general idea is the following: If $\Gamma_s^{E \oplus U}(n) = j \neq C_s(n)$ and B is the use block that is available for correcting n, then we enumerate an unused point into B and so immediately $\Gamma_s^{E \oplus U}(n) \uparrow$, then we can redefine $\Gamma_s^{E \oplus U}(n) = 1 - j$ with the *same* use block B.

A use block B for $\Gamma(n)$ can be *killed* (Section 4.7) by some node β at stage s, and we define Killed$(\mathsf{B}) = s$ and write $\mathsf{B} = \mathbf{B}^{\beta}_{|s\rangle}(\gamma, n)$. A killed use block can still be available for correcting n in the future in which case we have to show certain bad things will not happen to it. A use block which cannot be available for correcting n is good in the sense that it will not add any complexity to our construction and such use block will be called *permanently killed*. Since a permanently killed use block will never concern us, we are not using additional notation. For convenience, a permanently killed use block is also said to be killed.

Let $\mathsf{B}_0 = \mathbf{B}_{\langle s_0|}(\gamma, n)$ and $\mathsf{B}_1 = \mathbf{B}_{\langle s_1|}(\gamma, n)$ be two use blocks with $s_0 < s_1$ such that at stage s_1, B_0 is killed and a point x is in the use block. Suppose that x is extracted at $s_2 > s_1$, then B_1 will never be available for correcting n in the future as x will never be enumerated back into B_0. In such case, B_1 is called permanently killed. This phenomenon to B_1 will be handled *tacitly*. The other situation when we permanently kill a use block is in (2b) in Section 4.6.

Remark 4.8. To call a use block killed or permanently killed is to request a certain functional, say, Γ, to (re)define $\Gamma(n)$ with a *fresh* use block when necessary.

$\mathbf{B}_s(\gamma, n)$ is an interval and $\gamma_s(n)$ is a natural number. However, they are closely related. In a slight abuse of notation, if $\mathbf{B}_s(\gamma, n) = [u - l, u)$ and $u = \gamma_s(n)$, we write $y < \gamma_s(n) < z$ if $y < u - l$ and $u \leq z$. A *fresh* use block $\mathsf{B} = [u - l, u)$ is one with $u - l$ fresh and l sufficiently large. As we are either defining a functional with the same use block or a fresh one, it turns out that all these use blocks are pairwise disjoint and we can also leave sufficiently large spaces between adjacent use blocks for diagonalizing witnesses picked by R -nodes or points enumerated by G-requirements.

Remark 4.9. We need the size of B to be sufficiently large so that there is always an unused element whenever we need one. This phenomenon occurs also in the construction of a maximal incomplete d.c.e. degree. See Lemma 4.27.

Next, consider a Δ-functional that belongs to Maintain(β, U) for some R-node β and intends to ensure $\Delta^{E \oplus C_0 \oplus \cdots C_{k-1}} = U$, possibly without any $c_i \in \mathrm{Ji}(\mathcal{L})$. It is defined in essentially the same way except that Δ has additional oracle sets built by us besides E. The use block B is therefore a potential subset of $E \oplus C_0 \oplus \cdots C_{k-1}$. We say that B *crosses over* E and C_i for each $i < k$. To enumerate (or extract) a point k into (or from) B via $X \in \{E, C_0, \ldots, C_{k-1}\}$ is to let $X(k) = 1$ (or $X(k) = 0$, respectively); by default, $X = E$ if it is not explicitly mentioned. Likewise, we have $\mathbf{B}_s(\delta, n)$, $\mathbf{B}_{\langle s|}(\delta, n)$, and $\mathbf{B}^{\eta}_{|s\rangle}(\delta, n)$ defined.

Suppose that B crosses over X. The use block B is X*-restrained* if $X \restriction B$ is restrained, in which case we are not allowed to enumerate a point into or extract a point from B via X. B is *restrained* if B is X-restrained for some X. B is X*-free* if B is not X-restrained. As we will see, if B is E-restrained, then there will be a set C for some $c \in \mathrm{Ji}(\mathcal{L})$ such that B is C-free.

If it is not available for correcting n at stage s, the use block does not come into play at stage s and we therefore do not worry about it. However, if it is available for correcting n at stage s, the use block can be killed or E-restrained in which case we have to be cautious.

As a summary of the notations and also as a preview of what can happen to a use block in the construction, let us consider a use block B.

(1) If B is permanently killed, it will never be available for correcting.
(2) If B is available for correcting, and not restrained, we can do whatever we want to the use block B to make the correction.

(3) If B is available for correcting $\Gamma(n)$ and E-restrained, we will show that $\Gamma(n)$ is in fact correct and needs no additional correction.
(4) If B is available for correcting $\Delta^{E\oplus C_1\oplus\cdots\oplus C_{k-1}}(n)$ and E-restrained, we will show that there will be some $i < k$ such that B is C_i-free so that we can use C_i to correct $\Delta(n)$.

We remark that the set C for each $c \in \mathrm{Ji}(\mathcal{L})$ will be built as a c.e. set. Therefore if we enumerate a point into a use block via C, we will not extract it. In fact, (2b) in Section 4.6 takes advantage of this.

4.5. **The S-strategy.** Let β be an S-node with $\mathrm{seq}(\beta) = (b, \xi)$. The idea of the S-strategy is straightforward: It builds and keeps each Γ-functional that belongs to $\mathrm{Maintain}(\beta, U)$ (Definition 4.4) correct and total. At stage s, for each $\Gamma^{E\oplus U} = C$ that belong to $\mathrm{Maintain}(\beta, U)$ and for each $x \leq s$, β keeps $\Gamma^{E\oplus U}(x)$ defined and correct according to the following
Correcting Strategy:

(1) Suppose $\Gamma_s^{E\oplus U}(x)\downarrow = C_s(x)$. β does nothing.
(2) Suppose $\Gamma_s^{E\oplus U}(x)\downarrow \neq C_s(x)$ with use block $\mathsf{B} = \mathbf{B}_s(\gamma, x)$.
  (a) If B is killed and not E-restrained, then β enumerates an unused point, referred to as a *killing point*, into B via E. Then we go to (3) immediately.
  (b) If B is not killed and not E-restrained, β enumerates an unused point, referred to as a *correcting point*, into B via E. Then we redefine $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with the *same* use block B.
(3) Suppose $\Gamma_s^{E\oplus U}(x)\uparrow$. If each $\mathbf{B}_{\langle t|}(\gamma, x)$ with $t < s$ has been *killed* (see Section 4.7 below), then β will pick a fresh use block B′ and define $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with use block B′ (hence $\mathsf{B}' = \mathbf{B}_{\langle s|}(\gamma, x)$); if otherwise, we define $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with the use block that is not killed (there will be at most one such use block).

Remark 4.10.
- We will show that if B is E-restrained, then we will not have Case (2) in the correcting strategy.
- A correcting strategy never extracts a point from a use block.

4.6. **The R-strategy and the Δ-functional.** Let β be an R-node. If β decides to visit its U-outcome, it needs to build and keep the Δ-functional, if any, that belongs to $\mathrm{Maintain}(\beta, U)$ correct in essentially the same way as an S-node.

Let Δ, if any, belong to $\mathrm{Maintain}(\beta, U)$. Without loss of generality, we assume that the Δ-functional is $\Delta^{E\oplus C_0\oplus\cdots\oplus C_{k-1}} = U$ for some $c_i \in \mathrm{Ji}(\mathcal{L})$. For each $x \leq s$, β keeps $\Delta^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x)$ defined and correct according to the following
Correcting Strategy:

(1) Suppose that $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x)\downarrow = U_s(x)$. β does nothing.
(2) Suppose that $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x)\downarrow \neq U_s(x)$ with use block $\mathsf{B} = \mathbf{B}_s(\delta, x)$.
  (a) If B is killed and not E-restrained, then β enumerates an unused point, referred to as a *killing point*, into B via E. Then we go to (3) immediately.
  (b) If B is killed and E-restrained, we let C_i for some $i < k$ be a set such that B is C_i-free (we will show that such C_i exists) and then β enumerates an unused point, referred to as a *killing point*, into B via C_i. B

is then *permanently killed* (as C_i will be a c.e. set). Then we go to (3) immediately.

(c) If B is not killed and not restrained, then β enumerates an unused point, referred to as a *correcting point*, into B via E. Then we define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x) = U_s(x)$ with the same use block B.

(d) If B is not killed but E-restrained, we let C_i for some $i < k$ be a set such that B is C_i-free (we will show such C_i exists), and then β enumerates an unused point, referred to as a *correcting point*, into B via C_i. Then we define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_k}(x) = U_s(x)$ with the same use block B.

(3) Suppose that $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_k}(x)\uparrow$. If for each $t < s$, $\mathbf{B}_{\langle t|}(\delta, x)$ is killed, then β will choose a fresh use block B' and define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_k}(x) = U_s(x)$ with use block B' (hence $\mathsf{B}' = \mathbf{B}_{\langle s|}(\delta, n)$); otherwise, we will define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_k}(x) = U_s(x)$ with the use block that is not killed (there will be at most one such use block).

At stage s, if the U-outcome is visited, then β will follow the above instructions for each $x \le s$ and we have $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_k}(x)$ defined and correct.

Note that β only enumerates correcting points into the use block whenever a correction is needed. Thus, if $U(x)$ changes twice in a row, we will have two correcting points in the use block.

4.7. **The R-strategy and the killing-strategy.** Let β be an R-node. β will first pick a *threshold point* denoted by threshold(β). If β decides to visit its U-outcome, it needs to *kill* each functional that belongs to $\mathrm{Kill}(\beta, U)$, if any. The intention is that the use of each of these functionals at the point threshold(β) should go to ∞ as s goes to ∞. To be precise:

Let $\Gamma^{E\oplus U} = C$ (a Δ-functional is dealt with similarly) belong to $\mathrm{Kill}(\beta, U)$ and $v = \mathrm{threshold}(\beta)$. We suppose that $\mathrm{seq}(\beta^-) = (b, \xi)$ and note that at stage s, when β is visited, (the b-th copy of) each functional in $F_\xi(U)$ is defined and correct by the correcting strategies. Hence $\mathbf{B}_s(\gamma, x)\downarrow$ for each $x \le s$ after β^- finishes its job. Then β executes the following

Killing strategy: For each x with $v \le x \le s$, let $\mathsf{B}_x = \mathbf{B}_s(\gamma, x)$ be the use block. We enumerate an unused point, referred to as a *killing point*, into B_x and declare that B_x is *killed* (hence $\mathsf{B}_x = \mathbf{B}^{\beta}_{|s\rangle}(\gamma, x)$).

One easily sees that if B_x contains a killing point, then B_x will never be available for correcting x (Section 4.4).

4.8. **The R-strategy and its computation with slowdown condition.** Let β be an R_c-node, where $c \in \mathrm{Ji}(\mathcal{L})$ and $\mathrm{spec}(c * c_*) = \{c_1, \ldots, c_k\}$. Suppose that β is assigned to the requirement $R_c(\Psi) : C \neq \Psi^{E\oplus C_1\oplus\cdots\oplus C_k}$. Let $v = \mathrm{threshold}(\beta)$ and $x > v$ be a diagonalizing witness.

Suppose that at stage s, we have

$$\Psi_s^{E\oplus C_1\oplus\cdots C_k}(x)\downarrow= 0$$

with use $\psi_s(x)$. Let $\sigma = (E \oplus C_1 \oplus \cdots \oplus C_k)[s] \restriction \psi_s(x)$. (Again, we will tacitly actually define $\sigma = (E[s] \restriction \psi_s(x)) \oplus (C_1[s] \restriction \psi_s(x)) \oplus \cdots \oplus (C_k[s] \restriction \psi_s(x))$.) Now, abusing notation, we will let $y = \psi_s(x) - 1$ refer not only to the number $\psi_s(x) - 1$, but also to the string σ. We say y is *the computation for β at stage s*, while the diagonalizing witness x is understood from context. At stage $t > s$, y is *restored* if

$(E \oplus C_1 \oplus \cdots \oplus C_k)[t] \upharpoonright y + 1 = \sigma$; *injured* if otherwise. Usually, we will focus on part of the whole computation. Given a use block B, $y \upharpoonright B$ is *restored* if for each $x \in B$, $(E \oplus C_1 \oplus \cdots \oplus C_k)[t] \upharpoonright B = \sigma \upharpoonright B$; *injured* if otherwise. If y is restored eventually and $C(x) = 1$, then $\Psi_s^{E \oplus C_1 \oplus \cdots C_k}(x) \downarrow= 0 \neq C(x)$ and the requirement $R_c(\Psi)$ is satisfied.

For a technical reason, we introduce a slowdown condition when finding a computation. This delay feature is also used in the proof of the original D.C.E. Nondensity Theorem.

Definition 4.11. For any node α, a set X is *relevant* to α if X is either K, E, or C for some $c \in \mathrm{Ji}(\mathcal{L})$, or $X = U$ if an $S(U)$-requirement is assigned to some node $\gamma \subseteq \alpha$. (Of course, in the current section, we have only one set U.)

Clearly, there are only finitely many sets that are relevant to a fixed node.

Definition 4.12. Let X be a set, y be a number, and $t \leq s$ be two stages. We define

- $\mathrm{same}(X, y, t, s)$ iff $X_t \upharpoonright (y + 1) = X_s \upharpoonright (y + 1)$.
- $\mathrm{diff}(X, y, t, s)$ iff $\neg\, \mathrm{same}(X, y, t, s)$.
- $\mathrm{SAME}(X, y, t, s)$ iff for all s' with $t \leq s' \leq s$, we have $\mathrm{same}(X, y, s', s)$.

SAME() checks if a set is stable, same() and diff() will tell us which computations will be restored (as will be seen later).

Definition 4.13 (computation with slowdown condition)**.** Let β be an R-node, s the current stage, and $s^* < s$ be the last β-stage. Let y be the computation for β at stage s. y is the computation *with slowdown conditions* if the following is satisfied:

(1) For each X that is relevant to β, we have
$$\mathrm{SAME}(X, y, s^*, s).$$
(If s^* is not defined, then $\mathrm{SAME}(X, y, s^*, s)$ is defined to be false.)
(2) If n is a point enumerated into some use block $\mathsf{B} = [u-l, u)$ by a node $\alpha \subsetneq \beta$ at the same stage, then the computation y should also satisfy $y < u - l$.
(3) $y < s^*$.

If β does not find a computation with slowdown condition, then β simply visits its w-outcome. Clearly, imposing the slowdown condition only delays β for finitely many stages if the computation actually converges.

In the rest of the paper, a computation always refers to a computation with slowdown condition.

4.9. **The R-strategy and the $\varnothing$-data.** Let β be an R-node. If a computation is not found by β, we should visit the w-outcome of β. If a computation y is found, then we might be ready to visit the U-outcome and make some progress. As we will recursively collect a bunch of computations found at various R-nodes as our *data*, we put the single computation y into the same package as our base step.

Definition 4.14 ($\varnothing$-data)**.** Let β be an R-node. Suppose that a computation y is found at stage s. Let the *$\varnothing$-data* of β, denoted by $\mathcal{E}_s^{\varnothing}(\beta)$, consist of the following:

(1) a set of nodes $\mathcal{E}_s^{\varnothing}(\beta) = \{\beta\}$ (slightly abusing notation),
(2) the computation y for β.

If there is no confusion, we might drop the subscript s of $\mathcal{E}_s^{\varnothing}(\beta)$.

Before we have a long discussion on the U-outcome, let us have a quick overview of the strategy of an R-node β. If the d-outcome is *activated*, then β visits this d-outcome, claiming that the R-requirement is satisfied by doing nothing. In all other cases, after threshold(β) is defined and a diagonalizing witness $x >$ threshold(β) is picked, β tries to obtain the $\varnothing$-data. If it fails to obtain the $\varnothing$-data, then β visits the w-outcome, claiming (eventually) that a disagreement has been found and the R-requirement is therefore satisfied. If it obtains the $\varnothing$-data, then β *encounters each of the other outcomes in order* (Definition 4.1) and decides what to do next and which one of the outcomes to visit. In the current section 4, the first outcome is a U-outcome, and the second outcome is a ctr-outcome.

4.10. **The R-strategy and the U-outcome.** Suppose that the current stage is s and β is an R-node with $\varnothing$-data $\mathcal{E}_s^{\varnothing}(\beta)$. Now we encounter the first outcome of β, which is always a U-outcome. The action we take depends on whether this U-outcome is Type I, GREEN, or RED. Let us assume that $\mathrm{seq}(\beta^-) = (a, \eta) \in m \times [T_{\mathcal{L}}]$ and $\mathrm{seq}(\beta^\frown U) = (b, \xi)$, if the latter is defined. Let $v = \mathrm{threshold}(\beta)$.

If the U-outcome is Type I, then we visit it. By visiting this outcome, we kill each functional that belongs to $\mathrm{Kill}(\beta, U)$ (see Section 4.7 for the killing strategy), and we define and keep the Δ-functional that belongs to $\mathrm{Maintain}(\beta, U)$ correct (see Section 4.6 for the correcting strategy). $\mathcal{E}_s^{\varnothing}(\beta)$ is *not* discarded.

If the U-outcome is GREEN, then we visit it. We also kill each functional that belongs to $\mathrm{Kill}(\beta, U)$ by the killing strategy. In this case, there is no Δ-functional to define, and $\mathcal{E}_s^{\varnothing}(\beta)$ *is* discarded.

If the U-outcome is RED, then we do *not* visit it. Recall from Section 4.2.2 that β^* is defined. Note also that s is a β^*-stage visiting the U-outcome of β^* with its $\varnothing$-data $\mathcal{E}_s^{\varnothing}(\beta^*)$. We now combine $\mathcal{E}_s^{\varnothing}(\beta^*)$ and $\mathcal{E}_s^{\varnothing}(\beta)$ and add some information as follows.

Definition 4.15 (U-data)**.** At stage s, suppose that $\mathcal{E}_s^{\varnothing}(\beta)$ is obtained. If the first U-outcome for β is RED, we let β^* be defined as in Section 4.2.2. Let y_{β^*} and y_β be the computations for β^* and β, respectively. Let the *U-data* $\mathcal{E}_s^U(\beta)$ consist of the following:

(1) a set of nodes $\mathcal{E}_s^U(\beta) = \mathcal{E}_s^{\varnothing}(\beta) \cup \mathcal{E}_s^{\varnothing}(\beta^*)$,
(2) a U-reference stage s for each $\xi \in \mathcal{E}_s^U(\beta)$,
(3) for each $\xi \in \mathcal{E}_s^{\varnothing}(\beta)$, a *$U$-condition* $\mathrm{same}(U, y_\xi, s, t)$ with *U-reference length* y_ξ and variable t, and
(4) for each $\xi \in \mathcal{E}_s^{\varnothing}(\beta^*)$, a *$U$-condition* $\mathrm{diff}(U, y_\beta, s, t)$ with *U-reference length* y_β and variable t.

We denote the *U-condition* for each $\xi \in \mathcal{E}_s^U(\beta)$ by $\mathrm{Cond}^U(\xi, t)$ where t is a variable. If there is no confusion, we might drop the subscript s of $\mathcal{E}_s^U(\beta)$.

The reference stages for $\mathcal{E}_s^U(\beta)$, $\mathcal{E}_s^{\varnothing}(\beta)$, and $\mathcal{E}_s^{\varnothing}(\beta^*)$ happen to be the same for now. We are very careful about the reference length in the above definition to reflect the dependence of each parameter: the reference length in (3) will follow the same idea when we discuss multiple $S(U)$-requirements in Section 5, while the reference length in (4) still needs to be modified.

To demonstrate this situation, let us look at an example and see how $\mathcal{E}_s^U(\beta)$ can be helpful. This example reminds the reader of the essential idea in the proof of the original D.C.E. Nondensity Theorem, where we embed the 2-element chain, but in a more general setting.

Example 4.16. Let us consider the case in Figure 3. Recall from Figure 2a that $F_{00}(U)$ consists of $C_\lambda = \Gamma_\lambda^{E\oplus U}$ and $C_0 = \Gamma_0^{E\oplus U}$, $F_{01}(U)$ consists of $C_\lambda = \Gamma_\lambda^{E\oplus U}$ and $U = \Delta_0^{E\oplus C_\lambda}$, and $F_1(U)$ consists of $U = \Delta_\lambda^E$. For the easy case, let us ignore $R^3_{c_0}$ and consider $\beta = R^4_{c_\lambda}$ and $\beta^* = R^2_{c_\lambda}$. Let y_2 and y_4 denote the computation for $R^2_{c_\lambda}$ and $R^4_{c_\lambda}$, respectively, and $\mathcal{E}^U_s(R^4_{c_\lambda})$ be the U-data currently obtained when $R^4_{c_\lambda}$ encounters the RED U-outcome at stage s. As we can assume that y_2 is larger than it actually is as long as our construction allows restoring those extra digits, we assume that $y_4 < y_2$. Note that y_2 is injured at the end of stage s by the killing strategy of $R^2_{c_\lambda}$. It is also natural for us to enumerate the diagonalizing witnesses x_4 and x_2 with $x_4 < x_2$ for $R^4_{c_\lambda}$ and $R^2_{c_\lambda}$, respectively, into C_λ. Then the S-node assigned 10 potentially has to make corrections to $\Gamma_\lambda^{E\oplus U}(x_4)$ and $\Gamma_\lambda^{E\oplus U}(x_2)$ in the future, which potentially injures y_2. Let $s^* < s$ be the last stage when $R^4_{c_\lambda}$ is visited. With slowdown condition, we have $\mathrm{SAME}(U, y_4, s^*, s)$. Let $w = \mathrm{threshold}(R^2_{c_\lambda})$.

With the U-data and the above observations, we now discuss under which conditions we can restore y_2 or y_4.

Suppose that $t > s$ is a stage when we have $\mathrm{diff}(U, y_4, s, s_0)$ and are visiting the S-node assigned 10. We intend to explain why y_2 can be restored while this S-node can keep Γ_λ and Γ_0 correct. We consider a use block $\mathsf{B} < y_2$ that looks different before and after the moment when y_2 is found. The goal is to show that B is either not available for correcting, or does not need any correction. There are three kinds of such use blocks, $\mathbf{B}^{R^2}_{|s\rangle}(\gamma_\lambda, n)$ for some $n \ge w$, $\mathbf{B}^{R^2}_{|s\rangle}(\gamma_0, n)$ for some $n \ge w$, and $\mathbf{B}_{\langle s'|}(\delta_\lambda, n)$ for some n and some $s' \le s$, where the first two are similar.

For a use block $\mathsf{B} = \mathbf{B}^{R^2}_{|s\rangle}(\gamma_\lambda, n)$ for some $n \ge w$, we note that $\mathrm{Created}(\mathsf{B}) > s^*$ because otherwise it would have been killed at s^*. By $\mathrm{SAME}(U, y_4, s^*, s)$ and $\mathrm{diff}(U, y_4, s, t)$, we realize that even if we restore $y_2 \upharpoonright \mathsf{B}$ at stage t, B is not available for correcting n. Therefore restoring y_2 and keeping $\Gamma_\lambda(n)$ correct at the S-node 10 creates no conflicts. The same argument applies to a use block $\mathbf{B}^{R^2}_{|s\rangle}(\gamma_0, n)$ for each $n \ge w$. For the use block $\mathsf{B}' = \mathbf{B}_{\langle s'|}(\delta_\lambda, n)$ for some n and some $s' \le s$, we simply restore $y \upharpoonright \mathsf{B}'$ because if we have restored y_2, we are going to claim satisfaction of $R^2_{c_\lambda}$, and we are not going to visit the U-outcome of $R^2_{c_\lambda}$ anymore and hence we do not have to keep Δ_λ correct. By the way, if we have $\mathrm{diff}(U, y_4, s, t)$, each use block $< y_2$ will be E-restrained at stage t.

Suppose that at stage $t > s$, we have $\mathrm{same}(U, y_4, s, t)$ and are visiting $R^2_{c_\lambda}{}^\frown U$. We intend to explain why y_4 can be restored. We will only consider the use block $\mathsf{B} < y_4$ that looks different before and after the moment when y_4 is found. First of all, since $y_4 < B^{R^2}_{|s\rangle}(\gamma_\lambda, w)$ and $y_4 < B^{R^2}_{|s\rangle}(\gamma_0, w)$ by the Slowdown Condition (Definition 4.13 (2)), we will only consider a use block $\mathsf{B} = \mathbf{B}_{\langle s'|}(\delta, n) < y_4$ for some $s' \le s^*$ and some $n < y_4$. $\Delta_\lambda(n)$ is correct at stage s^* as otherwise we would enumerate a correcting point into B and hence $y_4 < \mathsf{B}$ by the Slowdown Condition. As we have $\mathrm{SAME}(U, y_4, s^*, s)$, no point will be enumerated into B during each stage s'' with $s^* \le s'' \le s$. For each $t' > s$, if we have $\mathrm{diff}(U, y_4, s, t')$, we restore y_2 and do not maintain this use block B; if $\mathrm{same}(U, y_4, s, t')$, then $\Delta_\lambda(n)$ is correct as it was at stage s^*. In other words, we will never enumerate a point into B and y_4 will never be injured. Therefore, it is safe for us to restore y_4 and activate the d-outcome of $R^4_{c_\lambda}$, claiming that $R^4_{c_\lambda}$ is satisfied. By the way, if we have $\mathrm{same}(U, y_4, s, t)$, this use block B will be E-restrained at stage t.

For each $t > s$, we have either $\text{diff}(U, y_4, s, t)$ or $\text{same}(U, y_4, s, t)$. This gives us a *decision map* $\mathcal{D}_t(R^4_{c_\lambda}) = \xi$ when $\text{Cond}^U(\xi, t)$ (Definition 4.15) holds. According to the decision map, we decide which computation is to be restored at each stage.

From this Example 4.16, we see that $\mathcal{E}^U_s(R^4_{c_\lambda})$ contains all information to decide whether y_2 or y_4 will be restored at each stage $t > s$, and they can really be restored while functionals that belong to $\text{Maintain}(01, U)$ or $\text{Maintain}(R^2_{c_\lambda}, U)$ can be kept correct depending on which computation is restored. This motivates that we should stop collecting more data and get ready to make some progress. This is exactly what the ctr-outcome suggests: After encountering a RED outcome and obtaining $\mathcal{E}^U_s(\beta)$, we encounter the second outcome of β, which is a ctr-outcome. β is ready to become a *controller*.

4.11. **The R-strategy and the controller, part 1.** As a ctr-outcome is never the first outcome, it will be clear from our construction that whenever an R-node β encounters the ctr-outcome, it must have obtained the U-data $\mathcal{E}^U(\beta)$ (Definition 4.15).

Definition 4.17 (controller)**.** At stage s, suppose that β is an R-node encountering the ctr-outcome with U-data $\mathcal{E}^U(\beta) = \mathcal{E}^\varnothing(\beta) \cup \mathcal{E}^\varnothing(\alpha)$ for some α (it will be shown that β and α are related in a certain way). Let $\mathcal{E}^{\text{ctr}}(\beta) = \mathcal{E}^U(\beta)$ (a modification will be needed in Section 5). Suppose $\text{seq}(\beta^-) = (b, \xi)$. We say that β becomes a *U^b-controller* (or *controller* for simplicity). The controller β inherits the priority from the terminal node $\beta^\frown \text{ctr}$ on the priority tree $\mathcal{T}$. Suppose that β and α are R_c- and R_d-nodes, respectively, for some $c, d \in \text{Ji}(\mathcal{L})$ with $d \leq c$ (we will not need to consider the case when $d > c$).

(1) If $d = c$, then we say that the β has *no U^b-problem*.
(2) If $d < c$ (see Section 4.2.2), then we say that α is the *U^b-problem* (or U-problem for short) for β. (See Example 4.20 below.)

In both cases, we let $\text{Cond}^U_\beta(\xi, t)$ be $\text{Cond}^U(\xi, t)$ for each $\xi \in \mathcal{E}^U(\beta)$. Let s^{ctr}_β denote current stage s.

While β is not initialized, $\hat{C} \upharpoonright s^{\text{ctr}}_\beta$ is restrained for each $\hat{c} \in \text{Ji}(\mathcal{L})$ with $\hat{c} \neq c$.

Definition 4.18 (decision map)**.** Let β become a controller with $\mathcal{E}^{\text{ctr}}(\beta)$. For each $s \geq s^{\text{ctr}}_\beta$, the *decision map* is defined by setting $\mathcal{D}_s(\beta) = \xi$ for the longest $\xi \in \mathcal{E}^{\text{ctr}}(\beta)$ with $\text{Cond}^U_\beta(\xi, t)$. The controller β *changes its decision* (at stage s) if $\mathcal{D}_s(\beta) \neq \mathcal{D}_{s-1}(\beta)$. When s is clear from context, we write $\mathcal{D}(\beta) = \xi$ for short.

If $\mathcal{D}_s(\beta) = \xi$, we would like to show that y_ξ can be restored, and we also put a restraint on $E \upharpoonright y_\xi$ at stage s.

Definition 4.19 (noise)**.** Let β be a controller with $\mathcal{E}^{\text{ctr}}(\beta)$. At the beginning of stage $s > s^{\text{ctr}}_\beta$, if there is some set X that is relevant to β (Definition 4.11) such that

$$\text{diff}(X, s^{\text{ctr}}_\beta, s-1, s),$$

then β *sees some noise* at stage s.

While it is not initialized, a controller β can see at most finitely much noise and hence changes its decision at most finitely many times. Therefore, if β sees some noise, we can safely initialize all nodes to the right of $\beta^\frown \text{ctr}$. In this Section 4, "longest" in Definition 4.18 does not matter as there will be a unique choice; it will matter in Section 5. From $\mathrm{Cond}^U(\beta, t)$ and $\mathrm{Cond}^U(\beta^*, t)$ defined in Definition 4.15 and from $\mathcal{E}^{\mathrm{ctr}}(\beta) = \mathcal{E}^U_s(\beta)$, we see that at each $s \geq s^{\mathrm{ctr}}_\beta$, $\mathcal{D}_s(\beta)$ is always defined.

If β becomes a controller at stage s, we stop the current stage. Example 4.16 is Case (1) in Definition 4.17. We now discuss Case (2) in the next example.

Example 4.20. Different than in Example 4.16, we now consider $R^3_{c_0}$ instead of $R^4_{c_\lambda}$. Suppose that $R^3_{c_0}$ is encountering its ctr-outcome with data $\mathcal{E}^U(\beta) = \mathcal{E}^\varnothing(R^3_{c_0}) \cup \mathcal{E}^\varnothing(R^2_{c_\lambda})$ (we drop the subscript s if there is no confusion). We let $\mathcal{E}^{\mathrm{ctr}}(\beta) = \mathcal{E}^U(\beta)$ but realizing that $R^2_{c_\lambda}$ is a U^1-problem for $R^3_{c_0}$ (Definition 4.172). Let $s_* = s^{\mathrm{ctr}}_{R^3_{c_0}}$.

Why is $R^2_{c_\lambda}$ called a problem? Because its diagonalizing witness x_2 should be enumerated into C_λ while C_λ also belongs to the oracle of $R^3_{c_0}$. This is the potential conflict. (A plausible attempt is to assume $y_3 < x_2$ by patiently waiting so that enumerating x_2 does not injure y_3. However, this does not work if we have multiple S-requirements.) The solution we take here is that we enumerate only x_3 into C_0 and keep x_2 out of C_λ. Note that enumerating x_3 into C_0 does not injure y_2 because C_0 does not belong to the oracle of $R^2_{c_\lambda}$ (this is not a coincidence – see Section 4.2.2).

To emphasize the fact to keep x_2 out of C_λ, the controller $R^3_{c_0}$ puts a restraint on $C_\lambda \restriction s_*$ (also for the sake of the G-requirement discussed later). The restraint is not canceled unless the controller $R^3_{c_0}$ is initialized.

Suppose $\mathrm{same}(U, y_3, s_*, s)$, then we have $\mathcal{D}(R^3_{c_0}) = R^3_{c_0}$ and $E \restriction y_3$ is restrained. Similar to Example 4.16, we can restore y_3 and activate the d-outcome of $R^3_{c_0}$ in this situation.

Suppose $\mathrm{diff}(U, y_3, s_*, s)$, we now have a different situation than in Example 4.16. First of all, x_2 is not enumerated, so we are not attempting to activate the d-outcome of $R^2_{c_\lambda}$. However, we still want to restore y_2. In fact, this can still be done and it is done in a very crude way: we simply restore y_2 and ignore the impact on each use block $\mathsf{B} < y_2$. Then we turn the GREEN U-outcome of $R^1_{c_0}$ $(= (R^2_{c_\lambda})^\sharp)$ into a RED outcome. By doing so, we will not visit the S-node assigned 10 and hence we do not need to keep those functionals correct and ignoring each use block $\mathsf{B} < y_2$ is legitimate. In this case, we also declare that $E \restriction y_2$ is restrained although it is not necessary. If $R^3_{c_0}$ never changes its decision, then our construction will run until $R^1_{c_0}$ becomes a controller at $s_{**} = s^{\mathrm{ctr}}_{R^1_{c_0}}$.

At each stage $s > s_{**}$, we will encounter the following cases. If $R^3_{c_0}$ sees some noise (Definition 4.19), $R^1_{c_0}$ is initialized ($R^1_{c_0}$ has lower priority than $R^3_{c_0}$ as the node $(R^1_{c_0})^\frown \mathrm{ctr}$ is to the right of $(R^3_{c_0})^\frown \mathrm{ctr}$). If $\mathcal{D}(R^1_{c_0}) = R^1_{c_0}$, we can have $R^1_{c_0}$ restored and activate the d-outcome of $R^1_{c_0}$. However, if $\mathcal{D}(R^1_{c_0}) = R^0_{c_\lambda}$, we cannot mimic what $R^3_{c_0}$ did because $(R^0_{c_\lambda})^\sharp$ is not defined this time. We will discuss this situation in the next example.

Before leaving this section, we give some definitions that will be used throughout the rest of the paper. Recall from Section 4.8 that a computation y is restored at (a substage of) stage t if $(E \oplus C_1 \oplus \cdots C_k)[t] \restriction y + 1 = y$; and is injured otherwise. By U-restoring y_β at the beginning of stage s, we mean that we set $(E_s \oplus C_{1,s} \oplus \cdots \oplus C_{k,s}) \restriction \mathsf{B} = y_\beta \restriction \mathsf{B}$ (viewing y_β as a string) for each use block $\mathsf{B} < y_\beta$ that belongs to a U-functional. In this section, each use block belongs to some U-functional, and therefore U-restoring y_β is the same as restoring y_β.

Definition 4.21 (U-restorable)**.** Let β be given and y_β be its computation. We say that y_β is (U, C)*-restorable* at stage s if, after we U-restore y_β and set $E \upharpoonright y_\beta$ to be restrained at the beginning of stage s, for each $\alpha \subseteq \beta$ and for each use block $\mathsf{B} < y_\beta$ (hence B is E-restrained) maintained by α, we have the following:

(1) If B is a use block for $\Gamma^{E\oplus U}(n) = D$, where $n > \text{threshold}(\beta)$, $\Gamma \in \text{Maintain}(\alpha, U)$ (so α is an S-node), and D is not necessarily different from C, then either
   (a) B is available for correcting n and $\Gamma^{E\oplus U}(n) = D_s(n)$ (i.e., $\Gamma(n)$ is correct and hence needs no correction), or
   (b) B is not available for correcting n.

(2) If B is a use block for $\Delta^{E\oplus\cdots}(n) = U(n)$ for some $n > \text{threshold}(\beta)$ and $\Delta \in \text{Maintain}(\alpha, U)$ (so α is an R-node), then either
   (a) B is available for correcting n and $\Delta^{E\oplus\cdots}(n) = U_s(n)$ (i.e., $\Delta(n)$ is correct and hence needs no correction), or
   (b) B crosses over C.

Suppose that y_β is (U, C)-restorable and β is an R_c-node. Then y_β is *weakly U-restorable* if either

- the witness x_β is not enumerated into C, or
- for some use block B that (2a) fails (hence (2b) holds), we have that B is C-restrained.

(In fact, as we will see later, if x_β is allowed to be enumerated into C, then B should be C-free and vice versa.) y_β is *U-restorable* in other cases.

If β is an R_c-node and U-restorable at stage s, then enumerating a point into C will not injure y_β. Therefore we can enumerate its diagonalizing witness x_β into C and the use block in Case (2b) in Definition 4.21 can help correcting $\Delta(n)$. Hence, y_β can be restored at the beginning of s and will not be injured by the end of stage s.

In Example 4.16, we showed that if $\text{diff}(U, y_4, s, t)$, then y_2 is U-restorable at t; if $\text{same}(U, y_4, s, t)$, then y_4 is U-restorable at t. In Example 4.20, if $\text{same}(U, y_3, s_*, s)$, then y_3 is U-restorable at s; if $\text{diff}(U, y_3, s_*, s)$, then y_2 is (U, C_λ)-restorable (in fact, no use block $\mathsf{B} < y_2$ satisfies (2) in Definition 4.21) and weakly U-restorable at s as the witness x_2 is not enumerated.

4.12. **The R-strategy and the controller, part 2.** In Example 4.20, it was shown that each controller β has some action to take unless it is a U^0-controller and $\mathcal{D}(\beta)$ is a U^0-problem (Definition 4.17(2)). In this section, we discuss this situation.

Example 4.22. We continue with Example 4.20. At stage s_{**}, $R^1_{c_0}$ becomes a U^0-controller and $R^0_{c_\lambda}$ is a U^0-problem (Definition 4.17). Suppose that at $s > s_{**}$ we have $\mathcal{D}_s(R^1_{c_0}) = R^0_{c_\lambda}$. Notice that we also have $\mathcal{D}_s(R^3_{c_0}) = R^2_{c_\lambda}$, a U^1-problem for the U^1-controller $R^3_{c_0}$. We have *two* U-problems, and they are both R_{c_λ}-nodes. This is not a coincidence.

Recall that this example is about embedding the 3-element chain (Figure 1b), and if α is a U-problem, then α must be an R_{c_λ}-node. By setting $m = 2$ (Figure 3), if we run into a U^0-problem, we must also have a U^1-problem, both of which are R_{c_λ}-nodes. Note that y_2 and y_0 are both weakly U-restorable, meaning that if the restraint on C_λ is dropped then their witnesses can be enumerated and therefore they should become U-restorable under certain conditions. This is our plan.

At stage s, when $\mathcal{D}(R^1_{c_0}) = R^0_{c_\lambda}$, we do the following two things and stop stage s:

(1) We obtain the U-data $\mathcal{E}^U_s(R^2_{c_\lambda}) = \mathcal{E}^\varnothing_{s_*}(R^2_{c_\lambda}) \cup \mathcal{E}^\varnothing_{s_{**}}(R^0_{c_\lambda})$. We add the new U-conditions for both $R^2_{c_\lambda}$ and $R^0_{c_\lambda}$ as follows:
   - $\mathrm{Cond}^U(R^0_{c_\lambda}, t)$ is $\mathrm{diff}(U, y_2, s, t)$, and
   - $\mathrm{Cond}^U(R^2_{c_\lambda}, t)$ is $\mathrm{same}(U, y_2, s, t)$.

   We also add the U-reference stage s for each $\xi \in \mathcal{E}^U(R^2_{c_\lambda})$. (As the notation suggests, this will be the data that $R^2_{c_\lambda}$ needs in order to encounter the second outcome of $R^2_{c_\lambda}$.)

(2) We establish a *link*, connecting the root of the tree and the ctr-outcome of $R^2_{c_\lambda}$. That is, whenever the root is visited, we *directly* encounter the ctr-outcome of $R^2_{c_\lambda}$ (and $R^2_{c_\lambda}$ will become a controller as we will see).

At the beginning of stage $s+1$, the next stage, if one of the controllers $R^3_{c_0}$ and $R^1_{c_0}$ sees some noise, we discard the U-data $\mathcal{E}^U(R^2_{c_\lambda})$ obtained at stage s and we also destroy the link, and then we wait for another stage when $\mathcal{D}(R^1_{c_0}) = R^0_{c_\lambda}$; otherwise, we *immediately* travel the link, and $R^2_{c_\lambda}$ encounters the ctr-outcome with the data $\mathcal{E}^U_s(R^2_{c_\lambda})$. Before we let $R^2_{c_\lambda}$ obtain $\mathcal{E}^{\mathrm{ctr}}(R^2_{c_\lambda})$, let us first analyze the data $\mathcal{E}^U_s(R^2_{c_\lambda})$ and see why $R^2_{c_\lambda}$ is ready to become a controller.

Firstly, recall that $\mathrm{Cond}^U_{R^3_{c_0}}(R^2_{c_\lambda}, t)$ is $\mathrm{diff}(U, y_3, s_*, t)$ and that $\mathrm{Cond}^U_{R^1_{c_0}}(R^0_{c_\lambda}, t)$ is $\mathrm{diff}(U, y_1, s_{**}, t)$, where s_* denotes the U-reference stage for $R^2_{c_\lambda}$ stored in $\mathcal{E}^{\mathrm{ctr}}_{s_*}(R^3_{c_0})$, and s_{**} is the U-reference stage for $R^0_{c_\lambda}$ stored in $\mathcal{E}^{\mathrm{ctr}}_{s_{**}}(R^1_{c_0})$. Let $s_0 > s_*$ be the last stage when $R^3_{c_0}$ sees some noise (Definition 4.19), then we have $\mathrm{SAME}(U, s_*, s_0, s)$. Note that we can assume, without loss of generality (by assuming that each y_i is as large as possible), that

$$y_3 < y_2 < s_* < s_0 < y_1 < y_0 < s_{**} < s.$$

Notice that we also have $\mathcal{D}_{s_0}(R^3_{c_0}) = R^2_{c_\lambda}$, or equivalently, $\mathrm{diff}(U, y_3, s_*, s_0)$.

For a stage $t > s$, from $\mathrm{SAME}(U, s_*, s_0, s)$ and $\mathrm{diff}(U, y_3, s_*, s_0)$ and $y_3 < y_2 < s_*$, we deduce that

$$\mathrm{same}(U, y_2, s, t) \Rightarrow \mathrm{diff}(U, y_3, s_*, t),$$

which implies that y_2 is weakly U-restorable at stage t; from $\mathrm{SAME}(U, s_*, s_0, s)$ (particularly from $\mathrm{same}(U, y_2, s_{**}, s)$) and $y_2 < y_1$, we deduce that

$$\mathrm{diff}(U, y_2, s, t) \Rightarrow \mathrm{diff}(U, y_1, s_{**}, t),$$

which implies that y_0 is weakly U-restorable at stage t.

Now, as we are encountering and then visiting the ctr-outcome, which is to the left of both of the controllers $R^3_{c_0}$ and $R^1_{c_0}$, we are safe to initialize both of them. More importantly, the restraint on C_λ is dropped. Therefore when $R^2_{c_\lambda}$ becomes a controller at $s^{\mathrm{ctr}}_{R^2_{c_\lambda}} = s+1$, we are allowed to enumerate x_2 and x_0, the diagonalizing witnesses for y_2 and y_0, respectively, into C_λ. Being weakly U-restorable is now being U-restorable. By the way, we also put a restraint on $C_0 \upharpoonright s_{R^2_{c_\lambda}}$.

We remark that in the above example, visiting the second outcome of $R^2_{c_\lambda}$ requires a lot of work — we have to obtain $\mathcal{E}^U(R^2_{c_\lambda})$ in a very time-consuming way. In this example, the next outcome is a ctr-outcome, and we are lucky that $R^2_{c_\lambda}$ can immediately become a controller and the U-data $\mathcal{E}^U_s(R^2_{c_\lambda})$ is not wasted. However, this will not generally be true in Section 5. The U-data can be wasted (in the same manner that $\varnothing$-data can be wasted) and both of the previous controllers are initialized. It seems that we have gained nothing, but encountering the second outcome one more time is a bit of progress.

Now we make the procedure of obtaining $\mathcal{E}^U(R^2_{c_\lambda})$ as in Example 4.22 formal. We set $m = |\mathrm{Ji}(\mathcal{L})| + 1$ to keep the argument simple.

Definition 4.23 (strong U-data)**.** Suppose $m = |\mathrm{Ji}(\mathcal{L})|+1$. Suppose that each α_i, $i < m$, is a U^i-problem for the controller β_i, where $\alpha_0 \subseteq \cdots \subseteq \alpha_{m-1}$. For each α_i, its computation is y_{α_i}, and $\mathrm{Cond}^U_{\beta_i}(\alpha_i, s)$ is $\mathrm{diff}(U, z_{\alpha_i}, s_i, s)$, where z_{α_i} is the U-reference length (in fact, $z_{\alpha_i} = y_{\beta_i}$ in this section) and s_i is the U-reference stage (in fact, $s_i = s^{\mathrm{ctr}}_{\beta_i}$ in this section). Let s be the current stage when $\mathcal{D}_s(\beta_0) = \alpha_0 \neq \mathcal{D}_{s-1}(\beta_0)$. By the Pigeonhole Principle, we have for some $0 \leq i < j < m$ and some $c \in \mathrm{Ji}(\mathcal{L})$ such that both α_i and α_j are R_c-nodes.

The U-data $\mathcal{E}^U_s(\alpha_j)$ consists of the following:

(1) a set of nodes $\mathcal{E}^U_s(\alpha_j) = \mathcal{E}^\varnothing(\alpha_j) \cup \mathcal{E}^\varnothing(\alpha_i)$, where $\mathcal{E}^\varnothing(\alpha_i)$ belongs to $\mathcal{E}^{\mathrm{ctr}}(\beta_i)$ and $\mathcal{E}^\varnothing(\alpha_j)$ belongs to $\mathcal{E}^{\mathrm{ctr}}(\beta_j)$ (the subscripts of $\mathcal{E}^\varnothing(\alpha_i)$ and $\mathcal{E}^\varnothing(\alpha_j)$ can be deduced from $\mathcal{E}^{\mathrm{ctr}}(\beta_i)$ and $\mathcal{E}^{\mathrm{ctr}}(\beta_j)$, respectively, and hence can be omitted),

(2) for each $\xi \in \mathcal{E}^\varnothing(\alpha_j)$, a U-condition $\mathrm{Cond}^U(\xi, t) = \mathrm{same}(U, y_\xi, s, t)$ with *reference length* y_ξ, *reference stage* s, and variable t,

(3) for each $\xi \in \mathcal{E}^\varnothing(\alpha_i)$, a U-condition $\mathrm{Cond}^U(\xi, t) = \mathrm{diff}(U, y_{\alpha_j}, s, t)$ with *reference length* y_{α_j}, *reference stage* s, and variable t,

This U-data is *strong U-data*. U-data that is not strong U-data (Definition 4.15) is called *weak U-data*.

The *U-link* connects the root of the priority tree and the ctr-outcome of α_j, in the sense that whenever the root is visited, we skip the actions to U-functionals and directly travel the link and let α_j encounter the ctr-outcome with the U-data $\mathcal{E}^U_s(\alpha_j)$.

Encountering the ctr-outcome will make α_j a controller following Definition 4.17. Note that in Definition 4.17, it does not matter whether the U-data $\mathcal{E}^U(\beta)$ is strong or not. However, if it is a strong U-data, it will not have a U-problem.

As in Example 4.22, we of course hope that $\mathrm{Cond}^U(\alpha_j, s)$ implies that α_j is restorable at stage s and that $\mathrm{Cond}^U(\alpha_i, s)$ implies that α_i is restorable at stage s. This will be proved in the verification section.

The following example continues Example 4.22 and demonstrates the situation after $R^2_{c_\lambda}$ becomes a controller with the strong U-data $\mathcal{E}^U(R^2_{c_\lambda}) = \{R^2_{c_\lambda}, R^0_{c_\lambda}\}$.

Example 4.24. Suppose $R^2_{c_\lambda}$ becomes a controller with strong U-data $\mathcal{E}^U(R^2_{c_\lambda}) = \{R^2_{c_\lambda}, R^0_{c_\lambda}\}$ and that $\mathcal{D}(R^2_{c_\lambda}) = R^2_{c_\lambda}$. In this case, we are not going to skip any nodes below $R^0_{c_\lambda}$ and directly visit $(R^2_{c_\lambda})^\frown d$. For example, we might have $R^1_{c_0}$ visiting its w-outcome for a very long time. Meanwhile, $R^7_{c_0}$ becomes a controller with $\mathcal{E}^{\mathrm{ctr}}(R^7_{c_0}) = \{R^7_{c_0}, R^6_{c_\lambda}\}$ in the same sense as $R^3_{c_0}$. Then $\mathcal{D}(R^7_{c_0}) = R^6_{c_\lambda}$. We will turn the GREEN U-outcome of $R^5_{c_0}$ RED. Later $R^5_{c_0}$ becomes a controller with $\mathcal{E}^{\mathrm{ctr}}(R^5_{c_0}) = \{R^5_{c_0}, R^0_{c_\lambda}\}$. Note that this $R^0_{c_\lambda}$ has a new computation and a new diagonalizing witness, which is different from the data stored in $\mathcal{E}^{\mathrm{ctr}}(R^3_{c_0})$. Then $\mathcal{D}(R^5_{c_0}) = R^0_{c_\lambda}$ and we obtain strong U-data $\mathcal{E}^U(R^6_{c_\lambda}) = \{R^6_{c_\lambda}, R^0_{c_\lambda}\}$, and then $R^6_{c_\lambda}$ becomes a controller. It can be the case that $\mathcal{D}(R^2_{c_\lambda}) = R^2_{c_\lambda}$ and $\mathcal{D}(R^6_{c_\lambda}) = R^0_{c_\lambda}$. Whenever $R^2_{c_\lambda}$ sees some noise, all nodes to the right of $(R^2_{c_\lambda})^\frown\mathrm{ctr}$ are initialized, including the controller $R^6_{c_\lambda}$.

One last remark is that (2b) in Definition 4.21 can actually happen. However, we have to consider a more complicated lattice, for example, the diamond lattice in Figure (1b) or the 6-element lattice in Figure (1c). For this reason, we let the readers sort out the details, and we will see in the verification that if (2a) fails, then (2b) must hold.

4.13. **The G-strategy.** Recall that we have a global requirement

$$G : K = \Theta^{j(1)},$$

where we assume $j(1) = E \oplus C_0 \oplus C_1 \oplus \cdots C_{|\mathrm{Ji}(\mathcal{L})|-1}$ where $\{c_0, c_1, \ldots, c_{|\mathrm{Ji}(\mathcal{L})|-1}\} = \mathrm{Ji}(\mathcal{L})$. $\Theta(x)$ is always defined with fresh large use $\theta(x) + 1$ the first time, which never changes. (In order to prevent coding by this requirement to interfere with witnesses and killing points, we agree that no use block for a U-functional will be allowed to contain any Θ-uses.)

Now, when x is enumerated into K at stage s, we choose C_k for some k and simply enumerate $a = \theta(x)$ into it. The correct set C_k will be denoted by $\chi_s(a)$, and the following describes how we decide it:

List all controllers in decreasing order of priority as

$$\beta_0, \beta_1, \ldots, \beta_{n-1}.$$

(Recall that β_i has higher priority than β_j if $\beta_i^\frown \mathrm{ctr} <_P \beta_j^\frown \mathrm{ctr}$.) We assume that each β_i is an $R_{c_{\beta_i}}$-node for some $c_{\beta_i} \in \mathrm{Ji}(\mathcal{L})$, so it is restraining the set $\hat{C} \restriction s^{\mathrm{ctr}}_{\beta_i}$ for each $\hat{c} \neq c_{\beta_i}$. If $i < j$, then β_j becomes a controller after β_i, so $s^{\mathrm{ctr}}_{\beta_i} < s^{\mathrm{ctr}}_{\beta_j}$. Now, we let β_i be the controller of highest priority, if any, such that $a < s^{\mathrm{ctr}}_{\beta_i}$. If such β_i exists, we let $\chi_s(a) = C_{\beta_i}$; otherwise, we let $\chi_s(a) = E$.

We remark that if $\chi_s(\theta(x)) = E$, then enumerating $\theta(x)$ into E does not affect any controller. If $\chi_s(\theta(x)) = C_{\beta_i}$, then enumerating $\theta(x)$ into C_{β_i} does not affect $\beta_0, \beta_1, \ldots, \beta_i$ since $\theta(x)$ is relatively large for $\beta_0, \ldots, \beta_{i-1}$ and β_i has no restraint on C_{β_i}. The controllers $\beta_{i+1}, \ldots, \beta_{n-1}$ will be simply initialized. In fact, as $x < \theta(x) < s^{\mathrm{ctr}}_{\beta_i}$, this controller β_i sees some noise at stage s (Definition 4.19) and hence all nodes, including β_j with $j > i$, to the right of $\beta_i^\frown \mathrm{ctr}$ are initialized in the first place.

4.14. **The threshold point and diagonalizing witness.** Let β be an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$. The threshold point is denoted by threshold(β), and the diagonalizing witness is denoted by witness(β) (see Section 4.2). As usual, we should define witness(β) $>$ threshold(β). Note that threshold(β) is associated to β and is undefined only when β is initialized. witness(β) is associated with the w-outcome of β and will become undefined whenever a node/outcome (for example, the U-outcome of β) to the left of it is visited. Of course, the next time we visit β and witness(β) becomes undefined, we define it to be a fresh number. That being said, each time β visits its U-outcome, it has a different computation with a different witness. Suppose the w-outcome is the *true outcome* (the leftmost outcome of β that is visited infinitely often); then witness(β) will become stable.

Let β be a controller with β^* as its U^a-problem (where $a > 0$). Note that the w-outcome of an R-node $\alpha \subseteq \beta$ is to the right of $\beta^\frown \mathrm{ctr}$ and therefore witness(α) becomes undefined at s^{ctr}_β. In particular, such α will pick their diagonalizing witnesses larger than s^{ctr}_β next time, and so they are free to enumerate them without worrying about the restraint set by β.

Consider Figure 3 with the w-outcome of $R^3_{c_0}$ changed to the d-outcome. Suppose that $R^3_{c_0}$ becomes a controller at s_* but at each $s > s_*$, we have $\mathcal{D}_s(R^3_{c_0}) = R^3_{c_0}$. The previous diagonalizing witness $x_2 = \text{witness}_{s_*}(R^2_{c_\lambda})$ is not enumerated yet but it is prepared by $R^2_{c_\lambda}$ to become a controller (as in Example 4.22). We should avoid using x_2 in other places. By our convention, $R^2_{c_\lambda}$ should pick a new diagonalizing witness $x'_2 = \text{witness}_s(R^2_{c_\lambda}) > s_*$ next time, and perhaps $R^4_{c_\lambda}$ becomes a controller at s. In this situation, we have both $\mathcal{E}^{\varnothing}_{s_*}(R^2_{c_\lambda}) \subseteq \mathcal{E}^{\text{ctr}}(R^3_{c_0})$ and $\mathcal{E}^{\varnothing}_{s}(R^2_{c_\lambda}) \subseteq \mathcal{E}^{\text{ctr}}(R^4_{c_\lambda})$, each of which has a distinct diagonalizing witness with a corresponding computation.

We have the usual conflicts between a threshold point and a computation. In our construction, all $\varnothing$-data will be discarded by the end of each stage unless there is a controller β collecting them. Suppose that β is a controller with $\mathcal{E}^{\text{ctr}}(\beta) = \{\beta, \alpha\}$ with computations y_α and y_β. Let $k = \text{threshold}(\beta) \geq \text{threshold}(\alpha)$ (we can assume $\text{threshold}(\alpha) \leq \text{threshold}(\beta)$ if $\alpha \subseteq \beta$). Whenever there is a set X relevant to β such that $\text{diff}(X, k, s-1, s)$, we initialize $\beta^\frown \text{ctr}$, that is, we discard $\mathcal{E}^{\text{ctr}}(\beta)$ and β is no longer a controller. Note that we do not directly initialize an R-node, so $\text{threshold}(\beta)$ and $\text{threshold}(\alpha)$ remain defined. Therefore such an initialization to the ctr-outcome happens only finitely often to a fixed controller.

4.15. **The construction.** We can initialize not only a node but also an outcome. As we will always have that the o-outcome of α is initialized iff the node $\alpha^\frown o$ is initialized, we simply write $\alpha^\frown o$ for both events.

Definition 4.25 (initialization)**.** An S-node α is *initialized* by canceling all functionals are defined by α. An R-node α is *initialized* by canceling $\text{threshold}(\alpha)$, all parameters stored at each outcome of α, and all functionals (if any) that are defined by α. $\alpha^\frown w$ is *initialized* by canceling $\text{witness}(\alpha)$. $\alpha^\frown d$ is *initialized* by making it inactive. $\alpha^\frown U$ is *initialized* by canceling the Δ-functional (if any) that belongs to $\text{Maintain}(\alpha, U)$. $\alpha^\frown$ ctr is *initialized* by discarding $\mathcal{E}^{\text{ctr}}(\alpha)$ and making α no longer a controller.

If α is initialized, then we also tacitly initialize all outcomes and nodes to the right of α.

The following is a special case of Definition 4.19.

Definition 4.26 (threats)**.** Let β be a controller with $\mathcal{E}^{\text{ctr}}(\beta)$. At stage $s > s^{\text{ctr}}_\beta$ (the stage at which β becomes a controller), if there is some X that is relevant (Definition 4.11) to β such that

$$\text{diff}(X, \text{threshold}(\beta), s-1, s),$$

then β *sees some threats.* (We are assuming that $\alpha \subseteq \beta$ implies $\text{threshold}(\alpha) \leq \text{threshold}(\beta)$.)

Construction. At stage s, we first run the controller strategy (see below) and then the G-strategy (see below). Then we perform $\text{visit}(\lambda)$ (see below), where λ is the root of the priority tree $\mathcal{T}$. We stop the current stage whenever we perform $\text{visit}(\alpha)$ for some α with $|\alpha| = s$.

$\text{visit}(\alpha)$ *for an S-node:* Suppose that there is some U-link connecting α and $\beta^\frown o$ for some o-outcome of β, then we perform $\text{encounter}(\beta, o)$.

Suppose that there is no link. For each $\Gamma^{E \oplus U} = C$ (for some $c \in \text{Ji}(\mathcal{L})$) that belongs to $\text{Maintain}(\alpha, U)$ and for each $x \leq s$,

(1) Suppose $\Gamma_s^{E\oplus U}(x)\downarrow= C_s(x)$. β does nothing else.
(2) Suppose $\Gamma_s^{E\oplus U}(x)\downarrow\neq C_s(x)$ with use block $\mathsf{B} = \mathbf{B}_s(\gamma, x)$.
   (a) If B is killed and not E-restrained, then β enumerates an unused point, referred to as a *killing point*, into B via E. Then we go to (3) immediately.
   (b) If B is not killed and not E-restrained, then β enumerates an unused point, referred to as a *correcting point*, into B via E. Then we redefine $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with the *same* use block B.
(3) Suppose $\Gamma_s^{E\oplus U}(x)\uparrow$. If each $\mathbf{B}_{\langle t|}(\gamma, x)$ with $t < s$ has been *killed* (see Section 4.7), then β picks a fresh use block B' and defines $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with use block B' (hence $\mathsf{B}' = \mathbf{B}_{\langle s|}(\gamma, x)$); otherwise, we define $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with the use block that is not killed (there will be at most one such use block).

Then we stop the current substage and perform visit($\alpha^\frown 0$) for the R-node $\alpha^\frown 0$.

visit(α) *for an R-node:* If threshold(α) is not defined, we define it with a fresh number. Then we perform encounter(α, d).

Without loss of generality, we assume that α is assigned an $R_c(\Phi)$-requirement for some $c \in \mathrm{Ji}(\mathcal{L})$ and Φ.

encounter(α, d)*:* If d is active, then we perform visit($\alpha^\frown d$). If d is inactive, we perform encounter(α, w).

encounter(α, w)*:* If witness(α) is not defined, then we pick a fresh number $x >$ threshold(α) and define witness(α) $= x$.

(1) If a computation y is found by α with slowdown condition (Definition 4.13), we obtain $\varnothing$-data $\mathcal{E}_s^{\varnothing}(\alpha)$ (Definition 4.14) and perform encounter(α, U), where U is the first outcome (recall from Definition 4.1 that we add outcomes in order).
(2) If no computation is found, then we perform visit($\alpha \frown w$).

encounter(α, U)*:* Notice that we must have obtained $\mathcal{E}^{\varnothing}(\alpha)$.

(1) If the U-outcome is Type I, then let $v =$ threshold(α). For each functional Γ (Δ is dealt with similarly) that belongs to Kill(α, U) and for each x with $v \leq x \leq s$, let $\mathsf{B}_x = \mathbf{B}_s(\gamma, x)$ be the use block. We enumerate an unused point (*killing point*) into B_x and say B_x is *killed*.

   Let Δ belong to Maintain(α, U). Without loss of generality, we assume that this functional is to ensure $\Delta^{E\oplus C_0\oplus\cdots\oplus C_{k-1}} = U$ (allowing for $k = 0$, i.e., that there are no C_i). For each $x \leq s$, we do the following:
   (a) Suppose that $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x)\downarrow= U_s(x)$. Then β does nothing else.
   (b) Suppose we have $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x)\downarrow\neq U_s(x)$ with use block $\mathsf{B} = \mathbf{B}_s(\delta, x)$.
      (i) If B is killed and E-free, then β enumerates an unused point, referred to as a *killing point*, into B via E. Then we go to (3) immediately.
      (ii) If B is killed and E-restrained, we let C_i ($i < k$) be the set such that B is C_i-free (we will show such a C_i exists); then β enumerates an unused point, referred to as a *killing point*, into B

via C_i. B is then *permanently killed* (as C_i will be a c.e. set). Then we go to (3) immediately.

(iii) If B is not killed and E-free, then β enumerates an unused point, referred to as a *correcting point*, into B via E. Then we define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x) = U_s(x)$ with the same use block B.

(iv) If B is not killed and E-restrained, we let C_i for some $i < k$ be the set such that B is C_i-free (we will show such a C_i exists); then β enumerates an unused point, referred to as a *correcting point*, into B via C_i. Then we define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x) = U_s(x)$ with the same use block B.

(c) Suppose that $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x)\uparrow$. If for each $t < s$, $\mathbf{B}_{\langle t|}(\delta, x)$ is killed, β chooses a fresh use block B' and defines $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x) = U_s(x)$ with use block B' (hence $\mathsf{B}' = \mathbf{B}_{\langle s|}(\delta, n)$); otherwise, we will define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{k-1}}(x) = U_s(x)$ with the use block that is not killed (there will be at most one such use block).

Then we stop the current substage and perform visit($\alpha^\frown U$) for the S-node $\alpha^\frown U$.

(2) If the U-outcome is GREEN, then let $v = \text{threshold}(\alpha)$. For each functional Γ (Δ is dealt with similarly) that belongs to $\text{Kill}(\alpha, U)$ and for each x with $v \le x \le s$, let $\mathsf{B}_x = \mathbf{B}_s(\gamma, x)$ be the use block. We enumerate an unused point (*killing point*) into B_x and say B_x is *killed*. Then we stop the current substage and perform visit($\alpha^\frown U$) for the S-node $\alpha^\frown U$.

(3) If the U-outcome is RED, then we obtain the weak U-data $\mathcal{E}_s^U(\alpha)$ (Definition 4.15) and perform encounter(α, ctr).

encounter(α, ctr)*:* Notice that we must have obtained $\mathcal{E}^U(\alpha)$. Suppose that α is an R_c-node for some $c \in \text{Ji}(\mathcal{L})$.

(1) Let $\mathcal{E}^{\text{ctr}}(\alpha) = \mathcal{E}^U(\alpha)$ (Definition 4.17), and let α become a controller.
(2) We enumerate the diagonalizing witness for each $\xi \in \mathcal{E}^{\text{ctr}}(\alpha)$ into the set C if ξ is not a U-problem (see Definition 4.17).
(3) While α is a controller, we put a restraint on $\hat{C} \upharpoonright s_\alpha^{\text{ctr}}$ for each $\hat{c} \ne c$.

We then stop the current stage.

controller-strategy: Let β (if any) be a controller of highest priority such that β sees some noise (Definition 4.19). We initialize all nodes to the right of $\beta^\frown$ ctr. Suppose that β is an R_c-node, $\text{seq}_0(\beta) = b$, and $\mathcal{E}^{\text{ctr}}(\beta) = \mathcal{E}^U(\beta)$.

(1) If β sees also some threats (Definition 4.26), then we also initialize $\beta^\frown$ ctr.
(2) If β does not changes its decision (Definition 4.18), then we do nothing.
(3) If β changes its decision and $\mathcal{D}_s(\beta) = \xi$, then we set $E \upharpoonright y_\xi$ to be restrained (so each use block $\mathsf{B} < y_\xi$ is E-restrained) until the next time β changes its decision. Furthermore:
  (a) If ξ is not a U^b-problem of β, then we restore y_ξ and let $\xi^\frown d$ be active.
  (b) If ξ is a U^b-problem of β and $b > 0$, we restore y_ξ and turn the GREEN U-outcome of $\xi^\sharp$ into a RED U-outcome (once β changes its decision or is initialized, it turns back to GREEN).
  (c) If ξ is a U^0-problem, then we restore y_ξ and pick (as per Lemma 4.28) $\alpha_i \subsetneq \alpha_j$ such that they are U-problems and are both R_d-nodes for some $d \in \text{Ji}(\mathcal{L})$. We obtain the strong U-data $\mathcal{E}^U(\alpha_j)$ (Definition 4.23)

and establish a U-link connecting the root of the priority tree and the ctr-outcome of α_j (the U-link will be destroyed once traveled or β changes its decision).

G-strategy: Suppose $K = \Theta^{j(1)} = \Theta^{E\oplus C_0\oplus\cdots\oplus C_{|\mathcal{L}|-1}}$ where $\mathrm{Ji}(\mathcal{L}) = \{c_0, \ldots, c_{|\mathcal{L}|-1}\}$. For each $x \leq s$,

(1) if $\Theta^{j(1)}(x)$ has never been defined (so $x = s$), we define $\Theta^{j(1)}_s(x) = K_s(x)$ with a fresh use $\theta(x) + 1$ (which never changes).
(2) If $\Theta^{j(1)}(x) \downarrow\neq K_s(x)$, we enumerate $\theta(x)$ into the set $\chi_s(\theta(x))$ (see Section 4.13). Then we go to (3) immediately.
(3) If $\Theta^{j(1)}(x) \uparrow$, we define $\Theta^{j(1)}_s(x) = K_s(x)$ with the same use $\theta(x) + 1$.

We remark that we did not explicitly mention yet how big a use block should be to avoid being distracted by this technical issue (see Lemma 4.27).

4.16. **The verification.** First of all, one has to show that the use block is sufficiently large and also justify the controller strategy (3c) so that the construction will not terminate unexpectedly.

Lemma 4.27 (Block size)**.** *Each use block can be chosen sufficiently large.*

Proof. Let $\mathsf{B} = [a, b)$ be a use block. Such B can interact with a controller β with $\mathcal{E}^{\mathrm{ctr}}(\beta) = \{\beta, \alpha\}$ in the following way: If $\mathcal{D}(\beta) = \alpha$ with $y_\alpha > \mathsf{B}$, we will possibly extract a point from B. In this case, we say that B is *injured*. If $\mathcal{D}(\beta) = \beta$ with $y_\beta < \mathsf{B}$, we will possibly enumerate a point into B when the node which maintains B is visited. In this case, we say that B is *restored*. We only have to consider each use block B with $y_\beta < \mathsf{B}$ (and $\mathsf{B} < y_\alpha$) as otherwise restoring either y_β or y_α makes no changes to B. Therefore, we need to count how many times B can potentially be injured and then restored.

According to Definition 4.18, $\mathcal{D}_s(\beta) = \xi$ for the longest ξ such that $\mathrm{Cond}^U(\xi, s)$ and this is determined by $U_s \restriction y_\beta$ (where $y_\beta < a$). Therefore the number of times that $\mathsf{B} = [a, b)$ can be injured and then restored depends on the number of changes that $U \restriction a$ can have, i.e., the size of

$$S = \{s \mid \mathrm{diff}(U, a, s-1, s)\}.$$

It is clear that the number of S can be bounded by a computable function $p(a)$ since U is a d.c.e. set.

Therefore, when we define $\Gamma(x)$ with a fresh use block, we pick a fresh number a and let the use block be $[a, a + p(a))$. Defined in this way, a use block is sufficiently large.

It remains to show that a use block cannot interact with two controllers: Suppose that B interacts with β with $\mathcal{E}^{\mathrm{ctr}}(\beta) = \{\beta, \alpha\}$. Then a controller β' of lower priority believes that B never changes again (if it changes it, then β sees some noise and β' is initialized). For a controller β'' of higher priority, we can assume that $s^{\mathrm{ctr}}_{\beta''} < y_\beta < \mathsf{B}$, so B does not interact with β''. □

Lemma 4.28. *Let β be a controller such that $\mathcal{D}_s(\beta) = \xi$ where ξ is a U^0-problem, then there exists $\alpha_i \subsetneq \alpha_j$ both of which are R_d-nodes for some $d \in \mathrm{Ji}(\mathcal{L})$.*

Proof. Let $\beta_0 = \beta$. This $\mathcal{E}^U(\beta)$ must be weak U-data as strong U-data has no U-problem. Therefore the U-outcome of β must be turned RED by another (unique) controller β_1 with $\alpha_1 = \mathcal{D}(\beta_1)$ as a U-problem of β_1. Continuing this fashion, we find

$$\xi = \alpha_0 \subsetneq \alpha_1 \subsetneq \cdots \subsetneq \alpha_{m-1}$$

where each α_i is a U^i-problem of some controller β_i. Note that we assume $m = |\mathrm{Ji}(\mathcal{L})| + 1$. By the Pigeonhole Principle, there are α_i, α_j, and some $d \in \mathrm{Ji}(\mathcal{L})$ such that α_i and α_j are both R_d-nodes. □

Considering weak U-data, we are going to put each use block into one of several categories.

Definition 4.29. Let $\mathcal{E}^U(\beta) = \{\beta, \beta^*\}$ be weak U-data (Definition 4.15) obtained at stage s. Let $\mathcal{A}_1 = \{\eta \mid \eta \subsetneq \beta^*\}$, $\mathcal{A}_2 = \{\eta \mid \beta^* \subseteq \eta \subsetneq \beta\}$, and $\mathcal{A}_3 = \{\eta \mid \beta \subseteq \eta\}$. For a use block B that is maintained by a node in $\mathcal{A}_i$ and killed by a node in $\mathcal{A}_j$ at stage s, we define $\mathcal{Q}^U_{\mathcal{E}^U(\beta)}(\mathsf{B}) = (i, j)$ (for $i \le j$); if B is not killed, then $\mathcal{Q}^U_{\mathcal{E}^U(\beta)}(\mathsf{B}) = (i, \infty)$.

We write $\mathcal{Q}$ for $\mathcal{Q}^U_{\mathcal{E}^U(\beta)}$ if there is no confusion.

To tell whether a computation y is restorable or not, we only care about those blocks B with $\mathsf{B} < y$. The slowdown conditions (Definition 4.13) allow us to exclude some of the blocks from consideration:

Lemma 4.30. *Let $\mathcal{E}^U(\beta) = \{\beta, \beta^*\}$ be weak U-data (Definition 4.15) obtained at stage s. Suppose that β is an R_c-node and β^* is an R_d-node for some $d \le c \in \mathrm{Ji}(\mathcal{L})$. Given a use block B, if $\mathcal{Q}(\mathsf{B}) = (1, 1)$, then $\mathsf{B} > y_{\beta^*}$; if $\mathcal{Q}(\mathsf{B}) = (i, j)$ with $i \le j \in \{1, 2\}$, then $\mathsf{B} > y_\beta$.* □

Therefore, to tell whether y_{β^*} is (U, D)-restorable, we consider only those blocks B with $\mathcal{Q}(\mathsf{B}) = (1, j), j \in \{2, 3, \infty\}$ (and $\mathsf{B} < y_{\beta^*}$); to tell whether y_β is (U, C)-restorable, we consider only those blocks B with $\mathcal{Q}(\mathsf{B}) = (i, j), i \in \{1, 2\}, j \in \{3, \infty\}$ (and $\mathsf{B} < y_\beta$).

Lemma 4.31. *Let $\mathcal{E}^U(\beta) = \{\beta, \beta^*\}$ be weak U-data obtained at stage s. Suppose that β is an R_c-node and β^* is an R_d-node for some $d \le c \in \mathrm{Ji}(\mathcal{L})$. Suppose that B belongs to a Γ-functional.*

(1) If $\mathcal{Q}(\mathsf{B}) = (1, j)$ with $j \in \{3, \infty\}$, then this Γ-functional computes a set $\hat{C}$ with $\hat{c} \not\geq d$ (hence $\hat{c} \not\geq c$ since $d \le c$).
(2) If $\mathcal{Q}(\mathsf{B}) = (2, j)$ with $j \in \{3, \infty\}$, then this Γ-functional computes a set $\hat{C}$ with $\hat{c} \not\geq c$.

Suppose that B belongs to a Δ-functional.

(3) If $\mathcal{Q}(\mathsf{B}) = (1, j)$ with $j \in \{2, 3, \infty\}$, then B crosses over D.
(4) If $\mathcal{Q}(\mathsf{B}) = (1, j)$ with $j \in \{3, \infty\}$, then B crosses over C.

Proof. (1) Suppose $\hat{c} \ge d$, $\mathrm{seq}((\beta^*)^-) = (b, \xi)$ and that $\Gamma^{E \oplus U} = \hat{C}$ and so also $\Gamma^{E \oplus U} = D$ belongs to $F_\xi(U)$. Recall Definition 3.11 and Lemma 3.20. Setting $c_\sigma = \hat{c}$, there is some τ such that $\tau 0 \subseteq \sigma 0$ and $c_\tau = d$. Since $\Gamma^{E \oplus U} = D$ belongs to $\mathrm{Kill}(\beta^*, U)$, we have by Definition 4.5 that $\Gamma^{E \oplus U} = \hat{C}$ also belongs to $\mathrm{Kill}(\beta^*, U)$, but this implies $\mathcal{Q}(\mathsf{B}) = (1, 2)$, a contradiction.

(2) Suppose $\hat{c} \ge c$ and $\mathrm{seq}((\beta)^-) = (b, \xi)$. Note that $\Gamma^{E \oplus U} = \hat{C}$ belongs to $F_\xi(U)$, so $\Gamma^{E \oplus U} = C$ also belongs to $F_\xi(U)$ by Lemma 3.20. Hence the U-outcome should be Type I. But weak U-data $\mathcal{E}^U(\beta)$ can only be obtained when the U-outcome is RED, a contradiction.

(3) By Lemma 3.26 (1).
(4) By Lemma 3.26 (2).

□

Lemma 4.32. *Let $\mathcal{E}^U(\beta) = \{\beta, \beta^*\}$ be weak U-data obtained at stage s. Suppose that β is an R_c-node and β^* is an R_d-node for some $d \leq c \in \mathrm{Ji}(\mathcal{L})$. At each stage $t > s$ (independent of whether $\mathcal{E}^U(\beta)$ is discarded or not),*

(1) if $\mathrm{diff}(U, y_\beta, s, t)$ and $\mathrm{SAME}(\hat{D}, y_{\beta^*}, s, t)$ for each $\hat{d} \not\geq d$, then y_{β^*} is (U, D)-restorable (Definition 4.21) at stage t;

(2) if $\mathrm{same}(U, y_\beta, s, t)$ and $\mathrm{SAME}(\hat{C}, y_\beta, s, t)$ for each $\hat{c} \not\geq c$, then y_β is (U, C)-restorable at stage t.

Proof. Let $t > s$. We assume that we U-restore y_{β^*} or y_β at the beginning of the stage t.

(1) For y_{β^*}, we consider each B with $\mathcal{Q}(\mathsf{B}) = (1, j)$ where $j \in \{2, 3, \infty\}$ by Lemma 4.30. If B belongs to a Δ-functional, then B crosses over D by Lemma 4.31(3). Hence Definition 4.21(2b) holds for this use block B.

If B belongs to a Γ-functional and $\mathcal{Q}(\mathsf{B}) = (1, j)$ with $j \in \{3, \infty\}$, then by Lemma 4.31(1) we have that $\Gamma^{E \oplus U} = \hat{D}$ for some $\hat{d} \not\geq d$. At stage s when β^* found its computation y_{β^*}, we have $\Gamma_s^{E \oplus U}(x) = \hat{D}_s(x)$ if the former is defined. Then $\mathrm{SAME}(\hat{D}, y_{\beta^*}, s, t)$ tells us that in particular $\hat{D}_s(x) = \hat{D}_t(x)$. If B is available for correcting x at stage t (Section 4.4), as $\Gamma(x)$ is correct, we conclude that Definition 4.21(1a) holds for this use block B.

If B belongs to a Γ-functional and $\mathcal{Q}(\mathsf{B}) = (1, 2)$, we let s^* be the last stage when we visit β. Therefore, by the slowdown condition of β, we have $\mathrm{SAME}(U, y_\beta, s^*, s)$. Note that we also have $s^* < \mathrm{Created}(\mathsf{B}) \leq s$. Therefore, if we have $\mathrm{diff}(U, y_\beta, s, t)$, then B is not available for correcting even if we restore $E \upharpoonright \mathsf{B}$ to $y_{\beta^*} \upharpoonright \mathsf{B}$ at the beginning of stage t. We conclude that Definition 4.21(1b) holds for this use block B.

Hence y_{β^*} is (U, D)-restorable.

(2) For y_β, we consider each B with $\mathcal{Q}(\mathsf{B}) = (i, j)$ where $i \in \{1, 2\}, j \in \{3, \infty\}$ by Lemma 4.30.

If B belongs to a Γ-functional, we have by Lemma 4.31(1)(2) and by $\mathrm{SAME}(\hat{C}, y_\beta, s, t)$ for each $\hat{c} \not\geq c$ that $\Gamma^{E \oplus U}$ is correct and therefore Definition 4.21(1a) holds for this use block B.

If B belongs to a Δ-functional and $\mathcal{Q}(\mathsf{B}) = (1, j)$ with $j \in \{3, \infty\}$, by Lemma 4.31(4), B crosses over C. Hence Definition 4.21(2b) holds for this use block.

If B belongs to a Δ-functional and $\mathcal{Q}(\mathsf{B}) = (2, j)$ with $j \in \{3, \infty\}$, then $\mathrm{same}(U, y_\beta, s, t)$ says $\Delta_t(x) = U_s(x) = U_t(x)$, that is, $\Delta(x)$ is correct at stage t. Therefore Definition 4.21(2a) holds for this use block.

Hence y_β is (U, C)-restorable.

□

If β becomes a controller at stage s_β^{ctr} with $\mathcal{E}^{\mathrm{ctr}}(\beta) = \mathcal{E}^U(\beta) = \{\beta, \beta^*\}$ and β is an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$ then for each $\hat{C}$ with $\hat{c} \not\geq c$, we have $\mathrm{SAME}(U, s_\beta^{\mathrm{ctr}}, s_\beta^{\mathrm{ctr}}, t)$ at each $t > s_\beta^{\mathrm{ctr}}$ (while β is not initialized) since we have a restraint on $\hat{C}$. β is

U-restorable while β^* might be weakly U-restorable if it is an R_d-node for some $d < c$.

Lemma 4.33. *Let $\mathcal{E}^U_s(\alpha_j) = \mathcal{E}^{\varnothing}_{s_*}(\alpha_j) \cup \mathcal{E}^{\varnothing}_{s_{**}}(\alpha_i)$ be strong U-data (Definition 4.23), where $\alpha_i \subsetneq \alpha_j$ are U-problems of controller β_j and β_i, respectively. Suppose that β_j and β_i are $R_{c_{\beta_j}}$- and $R_{c_{\beta_i}}$-nodes and α_j and α_i are both R_d-nodes with $d < c_{\beta_i}$ and $d < c_{\beta_j}$. We let $s_* = s^{\mathrm{ctr}}_{\beta_j}$, $s_{**} = s^{\mathrm{ctr}}_{\beta_i}$, $s_0 > s_*$ be the stage when $\mathcal{D}_{s_0}(\beta_j) = \alpha_j$ (or equivalently, $\mathrm{diff}(U, y_{\beta_j}, s_*, s_0)$) and $\mathrm{SAME}(U, s_*, s_0, s)$, and s_{00} be the stage when $\mathcal{D}_{s_{00}}(\beta_i) = \alpha_i$ (or equivalently, $\mathrm{diff}(U, y_{\beta_i}, s_{**}, s_{00})$) and $\mathrm{SAME}(U, s_{**}, s_{00}, s)$. Without loss of generality, we may assume*

$$y_{\beta_j} < y_{\alpha_j} < s_* < s_0 < y_{\beta_i} < y_{\alpha_i} < s_{**} < s_{00} < s.$$

Recall that $\mathrm{Cond}^U(\alpha_j, t)$ is $\mathrm{same}(U, y_{\alpha_j}, s, t)$ and $\mathrm{Cond}^U(\alpha_i, t)$ is $\mathrm{diff}(U, y_{\alpha_j}, s, t)$.

(1) If $\mathrm{same}(U, y_{\alpha_j}, s, t)$ and $\mathrm{SAME}(\hat{D}, y_{\alpha_j}, s, t)$ for each $\hat{d} \ngeq d$, then y_{α_j} is (U, D)-restorable at stage t.

(2) If $\mathrm{diff}(U, y_{\alpha_j}, s, t)$ and $\mathrm{SAME}(\hat{D}, y_{\alpha_j}, s, t)$ for each $\hat{d} \ngeq d$, then y_{α_i} is (U, D)-restorable at stage t.

Proof. (1) From $\mathrm{diff}(U, y_{\beta_j}, s_*, s_0)$ and $\mathrm{SAME}(U, s_*, s_0, s)$, we deduce

$$\mathrm{same}(U, y_{\alpha_j}, s, t) \Rightarrow \mathrm{diff}(U, y_{\beta_j}, s_*, t).$$

Notice that from s_*, the set $\hat{D} \upharpoonright s_*$ for each $\hat{d} \neq c_{\beta_j}$ is restrained by the controller β_j. Since $d < c_{\beta_j}$, $\hat{D} \upharpoonright s_*$ is restrained for each $\hat{d} \ngeq d$. Therefore we have $\mathrm{SAME}(\hat{D}, y_{\alpha_j}, s_*, t)$ for each $\hat{d} \ngeq d$. By Lemma 4.32(1) we conclude that y_{α_j} is (U, D)-restorable.

(2) From $\mathrm{diff}(U, y_{\beta_i}, s_{**}, s_{00})$ and $\mathrm{SAME}(U, s_{**}, s_{00}, s)$, we deduce

$$\mathrm{diff}(U, y_{\alpha_j}, s, t) \Rightarrow \mathrm{diff}(U, y_{\beta_i}, s_{**}, t).$$

A similar argument as above shows that $\mathrm{SAME}(\hat{D}, y_{\beta_i}, s_{**}, t)$ holds for each $\hat{d} \ngeq d$. By Lemma 4.32(1) we conclude that y_{α_i} is (U, D)-restorable.

□

As the sets are properly restrained by the controllers, we have the following

Lemma 4.34. *Let β be a controller with $\mathcal{E}^{\mathrm{ctr}}(\beta) = \mathcal{E}^U(\beta) = \{\beta, \alpha\}$ where $\mathcal{E}^U(\beta)$ is either strong or weak. Suppose that β is an R_c-node and α is an R_d-node with $d \le c$. At each stage $s > s^{\mathrm{ctr}}_\beta$, if $\mathcal{D}_s(\beta) = \beta$, then y_β is U-restorable; if $\mathcal{D}_s(\beta) = \alpha$, then y_α is U-restorable if $d = c$ and weakly U-restorable if $d < c$.* □

Recall that if $\mathcal{E}^U(\beta)$ is strong U-data, then $d = c$. We summarize what we have proved in the following

Lemma 4.35. *Let β be a controller with $\mathcal{E}^{\mathrm{ctr}}(\beta) = \mathcal{E}^U(\beta)$. If $\mathcal{D}_s(\beta) = \xi$ and ξ is U-restorable, then we can restore y_ξ at the beginning of stage s and activate the d-outcome of ξ; y_ξ remains restored at all substages of stage s.* □

Lemma 4.36 (decision)**.** *Let β be a controller with $\mathcal{E}^{\mathrm{ctr}}(\beta) = \{\beta, \alpha\}$ (with $\alpha \subsetneq \beta$). Then for each $s > s^{\mathrm{ctr}}_\beta$, $\mathcal{D}_s(\beta)$ is defined.*

Proof. Either via Definition 4.15 or Definition 4.23, we have that $\mathrm{Cond}^U(\beta, s)$ iff $\neg\,\mathrm{Cond}^U(\alpha, s)$. Thus $\mathcal{D}_s(\beta)$ is always defined. □

Given the construction, we define the *true path* $p \in [\mathcal{T}]$ by induction: We first specify $\lambda \subseteq p$ for the root λ of $\mathcal{T}$. Suppose $\sigma \subseteq p$ is specified; then we say that the o-outcome of σ is the *true outcome* of σ if it is the leftmost outcome, if any, that is visited infinitely often, and we specify $\sigma^\frown o \subseteq p$. This completes the definition of p. That p is infinite follows from the next

Lemma 4.37 (Finite Initialization Lemma)**.** *Let p be the true path. Each node $\alpha \in p$ is initialized finitely often, and p is infinite.*

Proof. The root of the priority is never initialized. Using induction, we consider $\alpha \subseteq p$ and suppose that for each $\beta \subseteq \alpha$, β is initialized finitely often. We first show that α has a true outcome, say o-outcome, then we show that $\alpha^\frown o$ is initialized finitely often.

Suppose that α is an S-node. Note that if some U-link, established by a controller β, is traveled at some α-stage s, the U-link is destroyed and the controller β is also initialized (since β is to the right of the U-link). Establishing another U-link at the beginning of the next α-stage requires a controller to the left of β. However, there are potentially only finitely many of them by slowdown condition (an R-node which is visited for the first time cannot become a controller at the same stage). Therefore, there will be infinitely many stages when 0-outcome of α is visited.

Suppose that α is an R-node and s_0 is the stage after which α is not initialized. If there exists some stage $s_1 > s_0$ when α becomes a controller, then for each α-stage $s > s_1$, we must have $\mathcal{D}_s(\alpha) = \alpha$ and $\alpha^\frown d$ is visited. That is, the d-outcome is the true outcome. If such s_1 does not exist, then α will visit either the d-, the w-, or the U-outcome of α. The true outcome of α is therefore well-defined for α.

Let o-outcome be the true outcome of α and s_0 be the stage after which we do not visit any node to the left of the o-outcome. There are at most finitely many controllers to the left of $\alpha^\frown o$, each of which sees at most finitely much noise. Therefore, there is some $s_1 > s_0$ after which $\alpha^\frown o$ is not initialized.

Hence, p is infinite and each $\alpha \subseteq p$ is initialized finitely often. □

The following lemmas argue along the true path.

Lemma 4.38. *Each R^e-requirement is satisfied for each $e \in \omega$.*

Proof. Let p be the true path. By Lemma 4.3, let $\alpha \subset p$ be the longest R-node assigned an R^e-requirement, say, it is an $R_c(\Psi)$-requirement. Suppose the w-outcome is the true outcome. By Lemma 4.37, there are s_0 and x such that for each $s > s_0$, the only outcome of α that we visit is the w-outcome and witness$(\alpha) = x$. Then we claim that $\Psi^{E\oplus U}(x) \neq C(x)$: Otherwise, we will find a computation y and hence obtain $\mathcal{E}^{\varnothing}(\alpha)$, and so we will perform encounter(α, U) and then visit an outcome to the left of w-outcome, contradicting the choice of s_0.

Suppose the d-outcome is the true outcome. Then there is s_0 such that for each $s > s_0$, the only outcome of α that will be visited is the d-outcome. By Lemma 4.35, y_α is restored for each s. Thus, $\Psi^{E\oplus U}(x) = 0 \neq C(x) = 1$. □

Lemma 4.39. *The $S(U)$-requirement is satisfied.*

Proof. By Lemma 4.3, let α be the $S(U)$-node such that for each β with $\alpha \subsetneq \beta \subset p$ we have seq$(\alpha) =$ seq$(\beta) = (b, \xi)$. Note that for such β, we have Maintain$(\beta, U) = \varnothing$ (Definition 4.4). Therefore, each b-th copy of any functional in $F_\xi(U)$ is not killed by an R-node below α. Let s_0 be the stage such that for each $s > s_0$, α is not initialized. Since we never stop an S-node from correcting its functional, (the b-th copy of) the functional in $F_\xi(U)$ is correct and total. Hence, the $S(U)$-requirement is satisfied. □

Lemma 4.40. *The G-requirement is satisfied.*

Proof. By the G-strategy (1) and (3), $\Theta^{j(1)}(x)$ is eventually defined for each x. Since the G-strategy (2) can always act, $\Theta^{j(1)}(x)$ is correct. □

This completes the proof if we only have a single $S(U)$-requirement to satisfy. One can think of this construction as a sub-construction dealing with a single U-set. When we have $S(U_i)$-requirements for $i \leq k$, we have multiple sub-constructions organized in a nested way, each giving a U_i-condition that tells us whether a computation y is U_i-restorable. Now we can simply take the conjunction of U_i-conditions for each $i \leq k$ to have a condition implying that y is restorable. Organizing multiple sub-constructions now only requires (a lot of) patience.

5. The full construction

We will make general definitions for the priority tree, some of the basic strategies, etc., and give examples to demonstrate some of the combinatorics. The three-element chain in Figure 1a will be used for all examples in this section, and we will often restrict ourselves to considering only S_{U_0}- and S_{U_1}-requirements (Figure 4). *Lemma 4.32 and Lemma 4.33 are essential to the validity of our construction, and will be tacitly applied in all examples in this section.* The complexity of the construction increases with the number of join-irreducible elements and therefore the three-element chain will allow us to illustrate concretely the combinatorial ideas used in the general case.

In sections 5.9 and 5.10, we present the general construction and verification. In the verification section we try to strike a delicate balance between readability and being formal. Our formal proofs in the example sections are representative enough so that in these final sections, we will appeal to those examples when there is no loss in generality.

5.1. **The priority tree.** Recall from Section 4.2 that we are considering a space $\{0, 1, \dots, m-1\} \times [T_\mathcal{L}] = m \times [T_\mathcal{L}]$ and nondecreasing maps $R_c : [T_\mathcal{L}] \to [T_\mathcal{L}]$ for each $c \in \mathrm{Ji}(\mathcal{L})$, where $m = |\mathrm{Ji}(\mathcal{L})| + 1$. An S-node α working for $S_{U_0}, \dots, S_{U_{k-1}}$ is assigned to an element $(f_0, \dots, f_{k-1}) \in (m \times [T_\mathcal{L}])^k$, where each $f_i = (a_i, \xi_i) \in m \times [T_\mathcal{L}]$. We let $\mathrm{seq}(\alpha)(i) = f_i$, $\mathrm{seq}_0(\alpha)(i) = a_i$, and $\mathrm{seq}_1(\alpha)(i) = \xi_i$.

Definition 5.1. We define the priority tree $\mathcal{T}$ by recursion: We assign $((0, \iota)) \in (m \times [T_\mathcal{L}])^1$ to the root node λ and call it an S-node (ι is the finite string $00\cdots0$ of the proper length depending on $[T_\mathcal{L}]$).

Suppose that α is an S-node. We determine the least e such that there is no R^e-node $\beta \subset \alpha$ with $\beta^\frown w \subseteq \alpha$ or $\beta^\frown d \subseteq \alpha$, and assign $\alpha^\frown 0$ to R^e.

Suppose α is an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$ and α^- is assigned to $(f_0, \dots, f_{k-1}) \in (m \times [T_\mathcal{L}])^k$ where each $f_i = (a_i, \xi_i)$. We sequentially add U_i-outcomes from $i = k-1$ to $i = 0$, where each U_i-outcome could be of Type I or Type II (and in that case GREEN or RED). Then we add a single ctr-outcome, and finally we add the w- and the d-outcome. The priority order of each outcome, however, varies and is described as follows.

Proceeding from $i = k-1$ down to $i = 0$, we add the U_i-outcomes:

(1) If $\xi_i < R_c(\xi_i)$, then this U_i-outcome is a *Type I* outcome, and we assign $\alpha^\frown U_i$ to

$$(f_0, \dots, f_{i-1}, (a_i, R_c(\xi_i)), (0, \iota), \dots, (0, \iota)) \in (m \times [T_{\mathcal{L}}])^k.$$

The *next* outcome to be added is placed just to the left of this outcome.

(2) If $\xi_i = R_c(\xi_i)$, then this U_i-outcome is a *Type II* outcome. If $a_i < m-1$, then this outcome is GREEN and we assign $\alpha^\frown U_i$ to

$$(f_0, \dots, f_{i-1}, (a_i + 1, \iota), (0, \iota), \dots, (0, \iota)) \in (m \times [T_{\mathcal{L}}])^k.$$

If $a_i = m-1$, then this outcome is RED, and we do not assign $\alpha^\frown U_i$ to any requirement; it is a terminal node. In either case, the *next* outcome to be added is placed just to the right of this outcome.

After the U_0-outcome has been added, the next outcome we add is the ctr-outcome. We add it immediately to the left or right of the U_0-outcome depending on whether the U_0-outcome is a Type I outcome or a Type II outcome. We do not assign the node $\alpha^\frown$ ctr to any requirement; it is a terminal node.

Finally, we add the w- and d-outcomes to the right of all existing outcomes with d the rightmost outcome and assign both $\alpha^\frown w$ and $\alpha^\frown d$ to

$$(f_0, \dots, f_{k-1}, (0, \iota)) \in (m \times [T_{\mathcal{L}}])^{k+1}.$$

By the same argument as in Lemma 4.3, it is clear that along any infinite path through $\mathcal{T}$, all requirements are represented by some node.

Lemma 5.2. *Let p be an infinite path through $\mathcal{T}$.*

(1) For each S_{U_i}-requirement, there is an S-node α such that for all β with $\alpha \subseteq \beta \subset p$, we have $\operatorname{seq}(\alpha)(i) = \operatorname{seq}(\beta)(i)$.

(2) For each e, there is an R^e-node α such that either $\alpha^\frown d \subset p$ or $\alpha^\frown w \subset p$. □

Suppose that we consider only two U-sets. Then we assign both $\alpha^\frown w$ and $\alpha^\frown d$ to $(f_0, f_1) \in (m \times [T_{\mathcal{L}}])^2$ (where $(f_0, f_1) = \operatorname{seq}(\alpha^-)$) instead of assigning them to an element in $(m \times [T_{\mathcal{L}}])^3$ as in Definition 5.1. An example for the three element-lattice (see Figure 1a and Figure 2a) is given in Figure 4. We hide some of the R-nodes with w- or d-outcomes from the tree. A U_i-outcome has label i for short. A Type II outcome is denoted by a thick line. A terminal node is represented by a •. Therefore, we also hide the label of the ctr-outcome (i.e., any outcome not labeled but shown in Figure 4 is a ctr-outcome). An S-node assigned to $((1,1),(0,2))$, for example, is abbreviated as $11, 02$. The only outcome of an S-node is also hidden. Sometimes, to avoid having repeated scenarios and monstrous priority, we replace f_1 by $*$ so that no U_1-functionals will be built, i.e., we do not attempt to satisfy an S_{U_1}-requirement at this node. Depending on whether R^{15} is an R_{c_0}- or an R_{c_λ}-strategy, the priority tree grows in different ways, both of which are worth mentioning and shown in Figure 4, at the left and right bottom, respectively.

Before we delve into Figure 4 and discuss the combinatorics, we still have to explain the basic strategy for each node in order to have some basic idea of how the construction works. (With a little extra effort, we could discuss the strategies in general instead of the special case of considering only S_{U_0}- and S_{U_1}-requirements, but we hope the reader will be able to extrapolate from the case of two U-sets in his/her mind.)

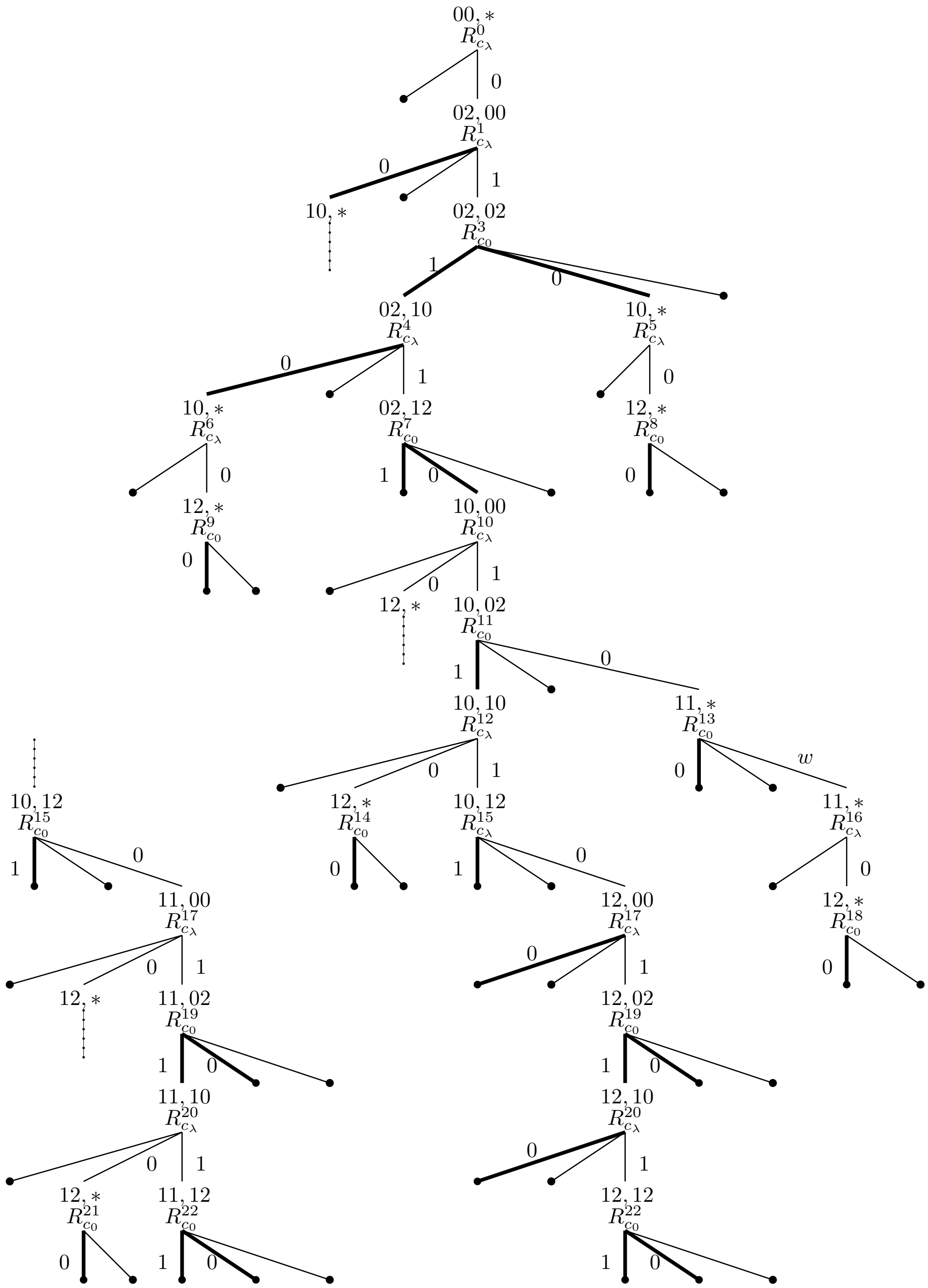


FIGURE 4. The priority tree for the 3-element chain

5.2. **Functionals manipulated at S-nodes and R-nodes.** An S-node β builds and maintains each Γ-functional in Maintain(β) defined as follow.

Definition 5.3. Let β be an S-node with $\mathrm{seq}(\beta) = (f_0, \dots, f_{k-1}) \in (m \times [T_{\mathcal{L}}])^k$ and ι be the string $00 \cdots 0$ of the proper length depending on $[T_{\mathcal{L}}]$.

(1) Suppose β is the root of the priority tree, then in fact $\mathrm{seq}(\beta) = ((0, \iota))$ and we define $\mathrm{Maintain}(\beta) = F_\iota(U_0)$.

(2) Suppose otherwise, we let $\alpha = (\beta^-)^-$. If $\mathrm{seq}(\alpha) = (g_0, \dots, g_{k-2}) \in (m \times [T_{\mathcal{L}}])^{k-1}$, then in fact we have $g_i = f_i$ for $i < k-1$ and $f_{k-1} = (0, \iota)$. We define $\mathrm{Maintain}(\beta) = F_\iota(U_{k-1})$.

If $\mathrm{seq}(\alpha) = (g_0, \dots, g_{k-1}) \in (m \times [T_{\mathcal{L}}])^k$, we let i be the least such that $g_i \neq f_i$. Then for $j > i$ we have $f_j = (0, \iota)$. Suppose $g_i = (a, \eta)$ and $f_i = (b, \xi)$. Recall from Definition 4.4 the definitions of $\mathrm{Maintain}(\beta, U_i)$. Then we define $\mathrm{Maintain}(\beta) = \mathrm{Maintain}(\beta, U_i) \cup F_\iota(U_{i+1}) \cup \cdots \cup F_\iota(U_{k-1})$.

Let $\mathrm{Maintain}(\beta, U_{<i})$ be the subset of $\mathrm{Maintain}(\beta)$ consisting of all U_j-functionals for $j < i$.

An R_c-node β will have to kill some functionals and build and maintain at most one Δ-functional along each U_i-outcome.

Definition 5.4. Let β be an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$. Suppose $\mathrm{seq}(\beta^-) = (f_0, \dots, f_{k-1}) \in (m \times [T_{\mathcal{L}}])^k$ and $f_i = (a_i, \eta_i)$ for each $i < k$. Applying Definition 4.5 to each f_i, we have $\mathrm{Kill}(\beta, U_i)$ and $\mathrm{Maintain}(\beta, U_i)$ defined properly. For each $i < k$, we define $\mathrm{Kill}(\beta, U_{\geq i})$ be the union of $\mathrm{Kill}(\beta, U_i)$ and $F_{\eta_j}(U_j)$ for each $j > i$.

By visiting the U_i-outcome, the R_c-node β kills each functional in $\mathrm{Kill}(\beta, U_{\geq i})$ and builds and maintains the Δ-functional (if any) in $\mathrm{Maintain}(\beta, U_i)$.

5.3. β^{*i} **and** $\beta^{\sharp i}$**.** Let β an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$. Suppose $\mathrm{seq}(\beta^-) = (f_0, \dots, f_{k-1}) \in (m \times [T_{\mathcal{L}}])^k$ with each $f_i = (a_i, \eta_i)$. If U_i $(i < k)$ is a Type II outcome of β, we define β^{*i} by adapting the definition in Section 4.2.2. Similarly for each $i < k$ with $a_i \neq 0$, we define $\beta^{\sharp i}$ by adapting the definition in Section 4.2.2.

5.4. **Basic strategies.** An S-node β builds and maintains each Γ-functional that belongs to $\mathrm{Maintain}(\beta)$ using exactly the same correcting strategy described in Section 4.5. An R-node β visiting its U_i-outcome kills each functional that belongs to $\mathrm{Kill}(\beta, U_{\geq i})$ using exactly the same killing strategy described in Section 4.7, and builds and maintains the Δ-functional (if any) that belongs to $\mathrm{Maintain}(\beta, U_i)$ using exactly the same correcting strategy described in Section 4.6. Collecting $\varnothing$-data when performing $\mathrm{encounter}(\beta, w)$ is the same as in Section 4.9. After the $\varnothing$-data is obtained, we begin by performing $\mathrm{encounter}(\beta, U_{k-1})$, where U_{k-1} is the first outcome added to β. Analogously, performing $\mathrm{encounter}(\beta, U_i)$ for $i < k-1$ now requires U_{i+1}-data $\mathcal{E}^{U_{i+1}}(\beta)$, and performing $\mathrm{encounter}(\beta, \mathrm{ctr})$ requires U_0-data $\mathcal{E}^{U_0}(\beta)$. We remark that a more suggestive notation for $\mathcal{E}^{U_i}(\beta)$ would be $\mathcal{E}^{U_{\geq i}}(\beta)$, but we choose to keep the notation short.

5.5. **Weak U_i-data.** As we are dealing with multiple $S(U)$-requirements, it will be convenient for us to reformulate $\varnothing$-data and U-data that were used in Section 4 to a more general format. For each $\mathcal{E}^{U_i}(\beta)$, we will also define a *U_i-data tree* $\mathcal{S}^{U_i}(\beta)$ to reveal the combinatorics.

For the rest of the paper, we adopt the following:

Convention 1: $\mathcal{E}^{U_i}(\beta)$ is abbreviated as $\mathcal{E}^i(\beta)$.
Convention 2: $\mathcal{S}^{U_i}(\beta)$ is abbreviated as $\mathcal{S}^i(\beta)$.
Convention 3: If $k = |\mathrm{seq}(\beta^-)|$, then $\mathcal{E}^k(\beta) := \mathcal{E}^{\varnothing}(\beta)$.

Definition 5.5 (data)**.** $\mathcal{E} = (Z, y, \mathcal{C})$ is *data* if

(1) Z is a finite set of R-nodes,
(2) $y : \xi \mapsto y_\xi$ for each $\xi \in Z$ such that y_ξ is the computation for ξ, and
(3) $\mathcal{C}$ is a finite set of functions of the form $\mathrm{Cond}^i(\beta) : \omega \to \{0, 1\}$ where $\beta \in Z$ and $i \in \omega$.

The symbol y, now viewed as a function defined on Z, is a bit overused but it should cause no confusion. We also confuse $\mathcal{E}$ with Z. We will write $\mathcal{E} = \{\beta_1, \dots, \beta_k\}$ for $Z = \{\beta_1, \dots, \beta_k\}$ and $\xi \in \mathcal{E}$ for $\xi \in Z$.

Definition 5.6 (data tree)**.** $\mathcal{S} = (S, f, g, h)$ is a *data tree* if

(1) S is a finite binary tree,
(2) $f : S \to \mathrm{Ji}(\mathcal{L})$,
(3) $g : S \to \{0, 1\}$ such that $g(\sigma) = 1$ implies $f(\sigma 0) = f(\sigma 1)$ and that $g(\sigma) = 0$ implies $f(\sigma 0) \leq f(\sigma 1)$, and
(4) $h : S \to \mathcal{T}$.

For $\sigma \in S$, we say σ has *type* c if $f(\sigma) = c$, σ is *strong* if $g(\sigma) = 1$, σ is *weak* if $g(\sigma) = 0$, λ denotes the empty string as usual.

If $\mathcal{S}_0 = (S_0, f_0, g_0, h_0)$ and $\mathcal{S}_1 = (S_1, f_1, g_1, h_1)$ are two data trees, and $l \in \{0, 1\}$, then $\mathcal{S}_0 \otimes_l \mathcal{S}_1 = (S, f, g, h)$ where $S(i\sigma) = S_i(\sigma)$, $f(i\sigma) = f_i(\sigma)$, $g(i\sigma) = g_i(\sigma)$, and $h(i\sigma) = h_i(\sigma)$ for $i \in \{0, 1\}$, $f(\lambda) = f_1(\lambda)$, $g(\lambda) = l$, $h(\lambda) = h_1(\lambda)$.

Here, $f(\sigma) = c$ denotes that $h(\sigma)$ is an R_c-node.

Definition 5.7 ($\varnothing$-data)**.** Let β be an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$. Suppose that the current stage is s. The $\varnothing$-data is $\mathcal{E}^{\varnothing}_s(\beta) = (Z, y, \mathcal{C})$ consisting of

(1) $Z = \{\beta\}$,
(2) a computation y_β for β, and
(3) $\mathcal{C} = \varnothing$.

We also define the $\varnothing$-data tree $\mathcal{S}^{\varnothing}_s(\beta) = (S, f, g, h)$ where $S = \{\lambda\}$, $f(\lambda) = c$, $g(\lambda) = 1$, and $h(\lambda) = \beta$. As usual, the subscript s is often omitted.

We adopt the same notation $\mathcal{E}^{\varnothing}(\beta)$ as in Definition 4.14 even though the definition here is slightly reformulated.

Definition 5.8 (operation on data)**.** For an R_c-node β such that $\mathrm{seq}(\beta^-) = (f_0, \dots, f_{k-1}) \in (m \times [T_{\mathcal{L}}])^k$, we let the U_k-data of β simply be $\mathcal{E}^{\varnothing}_s(\beta)$.

Given two U_{i+1}-data (defined inductively from $\varnothing$-data) $\mathcal{E}^{i+1}(\alpha) = (Z_0, y_0, \mathcal{C}_0)$ and $\mathcal{E}^{i+1}(\beta) = (Z_1, y_1, \mathcal{C}_1)$ (where $Z_0 \cap Z_1 = \varnothing$) and a stage s, we define

$$\mathcal{E}^i(\beta) = \mathcal{E}^{i+1}(\alpha) \otimes_s \mathcal{E}^{i+1}(\beta) = (Z, y, \mathcal{C})$$

as follows:

(1) $Z = Z_0 \cup Z_1$,
(2) $y = y_0 \cup y_1$ (so for $\xi \in Z_i$, we have $y_\xi = y_{i,\xi}$), and
(3) $\mathcal{C}$ consists of
    (a) $\mathcal{C}_0$ and $\mathcal{C}_1$,
    (b) for each $\xi \in Z_1$, $\mathrm{Cond}^i(\xi)(t) = \mathrm{same}(U_i, y_\xi, s, t)$, (In this case, y_ξ is the U_i-reference length and s the U_i-reference stage. $\mathrm{Cond}^i(\xi)$ has *type* same.)

(c) for each $\xi \in Z_0$, $\mathrm{Cond}^i(\xi)(t) = \mathrm{diff}(U_i, y_\gamma, s, t)$, where γ is the shortest node in Z_1 (By our conventions $y_\gamma \geq y_{\gamma'}$ for other $\gamma' \in Z_1$). (In this case, y_γ is the U_i-reference length and s the U_i-reference stage. $\mathrm{Cond}^i(\xi)$ has *type* diff.)

(Here we are identifying the predicate $\mathrm{same}(U_i, y_\xi, s, t)$ with its characteristic function, and we will keep doing so for other predicates.) We say that $\mathcal{E}^i(\beta)$ *extends* $\mathcal{E}^{i+1}(\alpha)$ and $\mathcal{E}^{i+1}(\beta)$, and that $\mathcal{E}^{i+1}(\alpha)$ and $\mathcal{E}^{i+1}(\beta)$ *belongs to* $\mathcal{E}^i(\beta)$.

In the above definition, y_γ is simply the least z such that if $\neg\,\mathrm{same}(U, y_\xi, s, t)$ for each $\xi \in \mathcal{E}^{i+1}(\beta)$, then we have $\mathrm{diff}(U_i, z, s, t)$. Therefore, for a fixed t, if $\mathrm{Cond}^i(\xi)(t) = 0$ for each $\xi \in \mathcal{E}^{i+1}(\beta)$, then $\mathrm{Cond}^i(\xi)(t) = 1$ for each $\xi \in \mathcal{E}^{i+1}(\alpha)$. For a stage t (usually clear from context), $\mathrm{Cond}^i(\xi)$ *holds* if $\mathrm{Cond}^i(\xi)(t) = 1$.

The following is analogous to Definition 4.15.

Definition 5.9 (weak U_i-data)**.** Let β be an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$. Suppose $\mathrm{seq}(\beta^-) = (f_0, \dots, f_{k-1})$ with $f_i = (b_i, \xi_i)$. At stage s, when we perform encounter(β, U_i) and U_i is a RED outcome, β^{*i} is well defined (which is an R_d-node for some $d \leq c$) and β^{*i} is visiting its U_i-outcome with (weak or strong) U_{i+1}-data $\mathcal{E}^{i+1}(\beta^{*i}) = (Z_0, y_0, \mathcal{C}_0)$ and a U_{i+1}-data tree $\mathcal{S}^{i+1}(\beta^{*i})$. β itself also has (weak or strong) U_{i+1}-data $\mathcal{E}^{i+1}(\beta) = (Z_1, y_1, \mathcal{C}_1)$ and a U_i-data tree $\mathcal{S}^{i+1}(\beta)$. We define

$$\mathcal{E}^i_s(\beta) = \mathcal{E}^{i+1}(\beta^{*i}) \otimes_s \mathcal{E}^{i+1}(\beta)$$
$$\mathcal{S}^i_s(\beta) = \mathcal{S}^{i+1}(\beta^{*i}) \otimes_0 \mathcal{S}^{i+1}(\beta)$$

If there is no confusion, we might drop the subscript s of $\mathcal{E}^i_s(\beta)$ and $\mathcal{S}^i_s(\beta)$.

Note that $\mathcal{S}^i_s(\beta)$ defined as above satisfies Definition 5.6(3) and therefore is a valid data tree. It is inductively clear from Lemma 4.32 (and Lemma 4.33 for later discussion) that for each $\xi \in \mathcal{E}^{i+1}(\beta^{*i})$, that $\mathrm{Cond}^i(\xi)(t) = 1$ holds implies that ξ is (U_i, D)-restorable (Definition 4.21); and for each $\xi \in \mathcal{E}^{i+1}(\beta)$, that $\mathrm{Cond}^i(\xi)(t) = 1$ holds implies that ξ is (U_i, C)-restorable. $\mathcal{E}^i(\beta)$ focuses on the recursive aspects of the data and $\mathcal{S}^i(\beta)$ focuses on the combinatorial aspects of the data. They are closed related.

As there will be many stages for the examples, we have the following

Convention: s^o_β denotes the stage when we perform encounter(β, o) *or* visit(β, o) *for an o-outcome.*

Note that s^{ctr}_β defined in Definition 4.17 conforms with this convention.

The next example is for $\mathcal{E}^0(R^{22}_{c_0})$ in Figure 4.

Example 5.10 (U_0-data and U_0-data tree)**.** In the case of $R^{15}_{c_\lambda}$ (the other case $R^{15}_{c_0}$ can be handled similarly), we will implement the basic strategies on the priority tree in Figure 4. We may assume that each node collects its $\varnothing$-data without delay. At stage s, the first node along $R^{22}_{c_0}$ that is encountering a RED outcome is $R^7_{c_0}$, encountering its U_1-outcome ($s^1_7 = s$). We let

$$\mathcal{E}^1(R^7_{c_0}) = \mathcal{E}^\varnothing(R^4_{c_\lambda}) \otimes_s \mathcal{E}^\varnothing(R^7_{c_0}) = \{R^4_{c_\lambda}, R^7_{c_0}\},$$
$$\mathcal{S}^1(R^7_{c_0}) = \mathcal{S}^\varnothing(R^4_{c_\lambda}) \otimes_0 \mathcal{S}^\varnothing(R^7_{c_0}),$$

and $R^7_{c_0}$ immediately visits its U_0-outcome, which is GREEN. So the U_1-data $\mathcal{E}^1(R^7_{c_0})$ is discarded. The next node that is encountering a RED outcome is $R^{15}_{c_\lambda}$

$(s^1_{15} = s)$. We let

$$\mathcal{E}^1(R^{15}_{c_\lambda}) = \mathcal{E}^\varnothing(R^{12}_{c_\lambda}) \otimes_s \mathcal{E}^\varnothing(R^{15}_{c_\lambda}) = \{R^{12}_{c_\lambda}, R^{15}_{c_\lambda}\},$$
$$\mathcal{S}^1(R^{15}_{c_\lambda}) = \mathcal{S}^\varnothing(R^{12}_{c_\lambda}) \otimes_0 \mathcal{S}^\varnothing(R^{15}_{c_\lambda}),$$

with

$$\mathrm{Cond}^1(R^{15}_{c_\lambda})(t) = \mathrm{same}(U_1, y_{15}, s^1_{15}, t),$$
$$\mathrm{Cond}^1(R^{12}_{c_\lambda})(t) = \mathrm{diff}(U_1, y_{15}, s^1_{15}, t).$$

Then $R^{15}_{c_\lambda}$ visits its U_0-outcome $(s^0_{15} = s)$. At the same stage, we will have $R^{22}_{c_0}$ encounter its RED U_1-outcome $(s^1_{22} = s)$. Again, we let

$$\mathcal{E}^1(R^{22}_{c_0}) = \mathcal{E}^\varnothing(R^{20}_{c_\lambda}) \otimes_s \mathcal{E}^\varnothing(R^{22}_{c_0}) = \{R^{20}_{c_\lambda}, R^{22}_{c_0}\},$$
$$\mathcal{S}^1(R^{22}_{c_0}) = \mathcal{S}^\varnothing(R^{20}_{c_\lambda}) \otimes_0 \mathcal{S}^\varnothing(R^{22}_{c_0}),$$

with

$$\mathrm{Cond}^1(R^{22}_{c_0})(t) = \mathrm{same}(U_1, y_{22}, s^1_{22}, t),$$
$$\mathrm{Cond}^1(R^{20}_{c_\lambda})(t) = \mathrm{diff}(U_1, y_{22}, s^1_{22}, t).$$

Then $R^{22}_{c_0}$ encounters its RED U_0-outcome $(s^0_{22} = s)$. Note that $(R^{22}_{c_0})^{*0} = R^{15}_{c_\lambda}$, so we now let

$$\mathcal{E}^0(R^{22}_{c_0}) = \mathcal{E}^1(R^{15}_{c_\lambda}) \otimes_s \mathcal{E}^1(R^{22}_{c_0}) = \{R^{12}_{c_\lambda}, R^{15}_{c_\lambda}, R^{20}_{c_\lambda}, R^{22}_{c_0}\},$$
$$\mathcal{S}^0(R^{22}_{c_0}) = \mathcal{S}^1(R^{15}_{c_\lambda}) \otimes_0 \mathcal{S}^1(R^{22}_{c_0}),$$

with

$$\mathrm{Cond}^0(R^{22}_{c_0})(t) = \mathrm{same}(U_0, y_{22}, s^0_{22}, t),$$
$$\mathrm{Cond}^0(R^{20}_{c_\lambda})(t) = \mathrm{same}(U_0, y_{20}, s^0_{22}, t),$$
$$\mathrm{Cond}^0(R^{15}_{c_\lambda})(t) = \mathrm{diff}(U_0, y_{20}, s^0_{22}, t),$$
$$\mathrm{Cond}^0(R^{12}_{c_\lambda})(t) = \mathrm{diff}(U_0, y_{20}, s^0_{22}, t).$$

(Recall that the choice of γ in Definition 5.8 is $R^{20}_{c_\lambda}$ here.)

As in Section 4, we can introduce *temporarily* (a formal definition will be given later) a decision map $\mathcal{D}_t(R^{22}_{c_0}) = \xi \in \mathcal{E}^0(R^{22}_{c_0})$ for the longest ξ with $\mathrm{Cond}^0(\xi)(t) = \mathrm{Cond}^1(\xi)(t) = 1$. For the decision map we temporarily define here, we note that

- $\mathcal{D}_t(R^{22}_{c_0})$ is defined for all $t > s$,
- if $\mathcal{D}_t(R^{22}_{c_0}) = R^{22}_{c_0}$, then $R^{22}_{c_0}$ is (U_0, C_0)-restorable and (U_1, C_0)-restorable,
- if $\mathcal{D}_t(R^{22}_{c_0}) = R^{20}_{c_\lambda}$, then $R^{20}_{c_\lambda}$ is (U_0, C_λ)-restorable and (U_1, C_λ)-restorable,
- if $\mathcal{D}_t(R^{22}_{c_0}) = R^{15}_{c_\lambda}$, then $R^{15}_{c_\lambda}$ is (U_0, C_λ)-restorable and (U_1, C_λ)-restorable, and
- if $\mathcal{D}_t(R^{22}_{c_0}) = R^{12}_{c_\lambda}$, then $R^{12}_{c_\lambda}$ is (U_0, C_λ)-restorable and (U_1, C_λ)-restorable.

Here we are applying Lemma 4.32 independently to U_0 and U_1. To be precise, we have to state the SAME() condition as in Lemma 4.32 (1) and (2); *we choose to keep it tacit in all remaining examples in this section.*

We note that $f(\sigma 0) \leq f(\sigma 1) = f(\sigma)$ for each $\sigma \in \mathcal{S}^0(R^{22}_{c_0})$. This follows inductively from the properties of β^{*i} and β.

Lemma 5.11. *Let β be an R_c-node with $\mathcal{E}^0(\beta)$ and $\mathcal{S}^0(\beta) = (S, f, g, h)$. For each $\sigma \in S$, we have $f(\sigma 0) \leq f(\sigma 1) = f(\sigma)$; for two leaves of S, if $\sigma <_{lex} \tau$, then $h(\sigma) \subsetneq h(\tau)$.*

Proof. An easy induction on S (using Definition 5.6(3)). □

5.6. **Controllers.** In Example 5.10, is $R^{22}_{c_0}$ ready to be a controller based on $\mathcal{E}^0(R^{22}_{c_0})$? Recall from Definition 4.17 that we have the notion of a U^b-problem. Looking at $\mathcal{E}^0(R^{22}_{c_0})$, we see that $R^{22}_{c_0}$ is an R_{c_0}-node and the others are R_{c_λ}-nodes. This will surely cause problems.

Definition 5.12 (U^b_i-problem)**.** Let β be an R_c-node with $\mathcal{E}^0(\beta)$ and $\mathcal{S}^0(\beta) = (S, f, g, h)$. For a $\sigma \in S$, if $f(\sigma 0) < c$, then we let $\alpha = h(\sigma 0) \in \mathcal{E}^0(\beta)$, $i = |\sigma|$, and $b = \mathrm{seq}_0(\alpha^-)(i)$ (defined in the first paragraph of Section 5.1). We say that α is a *U^b_i-problem* of β, or simply a U_i-problem. If in addition $f(\sigma) = c$, then α is the *critical* U^b_i-problem, or simply a critical U_i-problem.

If $\alpha = h(\tau)$ is a critical U_j-problem for some j, then there exists some σ such that $\tau = \sigma 0$ and in fact $j = |\sigma|$.

Example 5.13 (critical problems)**.** Continuing Example 5.10, we have, in the case of $R^{15} = R^{15}_{c_\lambda}$, that $R^{20}_{c_\lambda}$ is a critical U^1_1-problem, $R^{15}_{c_\lambda}$ is a critical U^1_0-problem, and $R^{12}_{c_\lambda}$ is a (noncritical) U^1_1-problem.

On the other hand, in the case of $R^{15} = R^{15}_{c_0}$, we see that $R^{20}_{c_\lambda}$ is a critical U^1_1-problem, $R^{12}_{c_\lambda}$ is a critical U^1_1-problem, and $R^{15}_{c_0}$ is not a U_0-problem.

The next lemma follows from Definition 5.6(3).

Lemma 5.14. *Let β be an R_c-node with $\mathcal{E}^0(\beta)$ and $\mathcal{S}^0(\beta) = (S, f, g, h)$. Suppose that $\alpha = h(\sigma 0)$ is a critical U^b_i-problem ($i = |\sigma|$ and $b = \mathrm{seq}_0(\alpha^-)(i)$), then $g(\sigma) = 0$.* □

If $\alpha = h(\sigma 0)$ is a critical U^b_i-problem, then $g(\sigma) = 0$ allows us to consider the node $\hat{\alpha} = h(\sigma)$ with the weak U_i-data $\mathcal{E}^i(\hat{\alpha}) = \mathcal{E}^{i+1}(\alpha) \otimes \mathcal{E}^{i+1}(\hat{\alpha})$ where $\alpha = \hat{\alpha}^{*i}$ (as we will see later, we have $g(\sigma) = 0$ if and only if $\mathcal{E}^i(h(\sigma))$ is weak U_i-data defined in Definition 5.9). If $\alpha = h(\sigma 0)$ is a noncritical U^b_i-problem, we might have that $g(\sigma) = 1$ (see Example 5.21).

In Example 5.10, having identified the critical problems for $R^{22}_{c_0}$, we would like to group $R^{15}_{c_\lambda}$ and $R^{12}_{c_\lambda}$ together and ignore the U_1-condition for now and only check if $\mathrm{Cond}^0(R^{15}_{c_\lambda})$ ($= \mathrm{Cond}^0(R^{12}_{c_\lambda})$) holds or not. In case it holds (and both $\mathrm{Cond}^0(R^{22}_{c_0})$ and $\mathrm{Cond}^0(R^{20}_{c_\lambda})$ fail), we can turn the GREEN U_0-outcome of R^7 RED and proceed as usual. For this purpose we have to make some modifications based on $\mathcal{E}^0(R^{22}_{c_0})$ and $\mathcal{S}^0(R^{22}_{c_0})$ to get $\mathcal{E}^{\mathrm{ctr}}(R^{22}_{c_0})$ and $\mathcal{S}^{\mathrm{ctr}}(R^{22}_{c_0})$.

Definition 5.15 (modified data and decision map)**.** Let β be an R_c-node with $\mathcal{E}^0(\beta) = (Z, y, \mathcal{C})$ and $\mathcal{S}^0(\beta) = (S, f, g, h)$. When we perform encounter(β, ctr) at s, we do the following:

(1) Let $\mathcal{S}^{\mathrm{ctr}}(\beta) = (S', f', g', h')$ be a data tree defined recursively as follows: we enumerate λ into S'. Suppose that we have enumerated σ into S', if $h(\sigma)$ is a critical $U_{|\sigma|-1}$-problem, we stop; otherwise we enumerate $\sigma 0$ and $\sigma 1$ into S' and continue. Then for each $\sigma \in S'$, we define $f'(\sigma) = f(\sigma)$, $g'(\sigma) = g(\sigma)$, and $h'(\sigma) = h(\sigma)$

(2) Let $\mathcal{E}^{\mathrm{ctr}}(\beta) = (Z', y', \mathcal{C}')$ consist of the following:
   (a) $Z' = \{h(\sigma) \mid \sigma \in \mathcal{S}^{\mathrm{ctr}}(\beta)\}$.
   (b) If $\alpha = h(\sigma 0) \in Z'$ is a critical $U_{|\sigma|}$-problem, we let $y'_\alpha = \max\{y_{h(\tau)} \mid \sigma 0 \subseteq \tau\} = y_{h(\sigma 00 \cdots 0)}$ (the last equality follows from our assumption that if $\alpha_0 \subsetneq \alpha_1$, then $y_{\alpha_0} > y_{\alpha_1}$). If $\alpha \in Z'$ is not a critical problem, we let $y'_\alpha = y_\alpha$.
   (c) To obtain $\mathcal{C}'$, for each $\alpha \in Z'$, we do the following:
      (i) If α is not a critical problem, then we enumerate $\mathrm{Cond}^j(\alpha) \in \mathcal{C}$ into $\mathcal{C}'$.
      (ii) If α is a critical U_i-problem, then for each $j \leq i$ such that $\mathrm{Cond}^j(\alpha) \in \mathcal{C}$ has type diff, we enumerate $\mathrm{Cond}^j(\alpha)$ into $\mathcal{C}'$.
      (iii) If α is a critical U_i-problem, then for each $j \leq i$ such that $\mathrm{Cond}^j(\alpha)(t) = \mathrm{same}(U_j, y_\alpha, s_*, t) \in \mathcal{C}$ where s_* is the U_j-reference stage, we enumerate $\mathrm{Cond}^j(\alpha)(t) = \mathrm{same}(U_j, y'_\alpha, s_*, t)$ into $\mathcal{C}'$.

   For each $\xi \in Z'$, we sometimes write $\mathrm{Cond}^i_\beta(\xi)$ for $\mathrm{Cond}^i(\xi)$ to emphasize which node is the controller.

Based on $\mathcal{E}^{\mathrm{ctr}}(\beta)$ and $\mathcal{S}^{\mathrm{ctr}}(\beta)$, the *decision map* $\mathcal{D}_s(\beta)$ is defined to be the longest $\xi \in \mathcal{E}^{\mathrm{ctr}}(\beta)$ such that $\mathrm{Cond}^i_\beta(\xi)(s) = 1$ for each i.

Note that $\mathcal{E}^0(\beta)$ and $\mathcal{S}^0(\beta)$ are not discarded yet.

Example 5.16. Continuing Example 5.10, $\mathcal{E}^{\mathrm{ctr}}(R^{22}_{c_0}) = \{R^{15}_{c_\lambda}, R^{20}_{c_\lambda}, R^{22}_{c_0}\}$ with

$$\begin{aligned}
&\mathrm{Cond}^0_{22}(R^{15}_{c_\lambda})(t) = \mathrm{diff}(U_0, y_{20}, s^0_{22}, t),\\
&\mathrm{Cond}^0_{22}(R^{20}_{c_\lambda})(t) = \mathrm{same}(U_0, y_{20}, s^0_{22}, t),\\
&\mathrm{Cond}^0_{22}(R^{22}_{c_0})(t) = \mathrm{same}(U_0, y_{22}, s^0_{22}, t),\\
&\mathrm{Cond}^1_{22}(R^{20}_{c_\lambda})(t) = \mathrm{diff}(U_1, y_{22}, s^1_{22}, t), \text{ and}\\
&\mathrm{Cond}^1_{22}(R^{22}_{c_0})(t) = \mathrm{same}(U_1, y_{22}, s^1_{22}, t).
\end{aligned}$$

Here $s^1_{22} = s^0_{22}$ as in Example 5.10. Note that $\mathcal{D}_s(R^{22}_{c_0})$ is defined at each stage $t > s^{\mathrm{ctr}}_{22}$. $\mathcal{E}^1(R^{15}_{c_\lambda})$, which belongs to $\mathcal{E}^0(R^{22}_{c_0})$, is not discarded; it will be used later.

Let us summarize what we have now. For an R_c-node β (where $c \in \mathrm{Ji}(\mathcal{L})$), encountering its ctr-outcome with $\mathcal{E}^0(\beta)$ and $\mathcal{S}^0(\beta)$, we first obtain $\mathcal{E}^{\mathrm{ctr}}(\beta)$ and $\mathcal{S}^{\mathrm{ctr}}(\beta)$ as in Definition 5.15 and β becomes a controller. While β is a controller, as in Section 4, we will put a restraint on $\hat{C} \upharpoonright s^{\mathrm{ctr}}_\beta$ for each $\hat{c} \neq c$. For a stage $t > s^{\mathrm{ctr}}_\beta(\beta)$, if $\mathcal{D}_t(\beta) = \xi$ is not a problem, then we simply restore y_ξ and activate the d-outcome of ξ; if $\mathcal{D}_t(\beta) = \xi$ is a critical U^b_i-problem where $b > 0$, we restore y_ξ (this is the y'_ξ in Definition 5.15) and turn the GREEN U_i-outcome of $\xi^{\sharp i}$ into a RED outcome; if $\mathcal{D}_t(\beta) = \xi$ is a critical U^b_i-problem where $b = 0$, we restore y_ξ and search for two critical U_i-problems that are both R_d-nodes for the same $d \in \mathrm{Ji}(\mathcal{L})$ in the history and obtain strong U_i-data as in Section 4. We will elaborate on this in the next section.

5.7. **Strong U_i-data and U_i-link.** We begin with an example that requires strong U_1-data.

Example 5.17 (strong U_1-data and U_1-link)**.** Continuing Example 5.16, we suppose that at stage s_* we have $\mathcal{D}(R^{22}_{c_0}) = R^{20}_{c_\lambda}$, the critical U^1_1-problem for $R^{22}_{c_0}$.

Then we restore y_{20} and turn the GREEN U_1-outcome of $R^{19}_{c_0} = (R^{20}_{c_\lambda})^{\sharp 1}$ RED. We also assume that for each $t > s_*$, $R^{22}_{c_0}$ does not see any noise. Hence we have $\text{SAME}(U_i, s^{\text{ctr}}_{22}, s_*, t)$ for $i = 0, 1$ and

$$\text{same}(U_0, y_{20}, s^{\text{ctr}}_{22}, s_*),$$
$$\text{diff}(U_1, y_{22}, s^{\text{ctr}}_{22}, s_*).$$

Let us suppose that at $s^{\text{ctr}}_{19} > s_*$ $R^{19}_{c_0}$ becomes a controller with $\mathcal{E}^{\text{ctr}}(R^{19}_{c_0}) = \{R^{15}_{c_\lambda}, R^{17}_{c_\lambda}, R^{19}_{c_0}\}$ with

$$\begin{aligned}
\text{Cond}^0_{19}(R^{15}_{c_\lambda})(t) &= \text{diff}(U_0, y_{17}, s^{\text{ctr}}_{19}, t)\\
\text{Cond}^0_{19}(R^{17}_{c_\lambda})(t) &= \text{same}(U_0, y_{17}, s^{\text{ctr}}_{19}, t)\\
\text{Cond}^0_{19}(R^{19}_{c_0})(t) &= \text{same}(U_0, y_{19}, s^{\text{ctr}}_{19}, t)\\
\text{Cond}^1_{19}(R^{17}_{c_\lambda})(t) &= \text{diff}(U_1, y_{19}, s^{\text{ctr}}_{19}, t)\\
\text{Cond}^1_{19}(R^{19}_{c_0})(t) &= \text{same}(U_1, y_{19}, s^{\text{ctr}}_{19}, t)
\end{aligned}$$

Suppose that at $s_{**} > s^{\text{ctr}}_{19}$, we have $\mathcal{D}_{s_{**}}(R^{19}_{c_0}) = R^{17}_{c_\lambda}$, a critical U^0_1-problem for $R^{19}_{c_0}$, so we are in the same situation as in Example 4.22. We restore y_{17} and collect

$$\mathcal{E}^1(R^{20}_{c_\lambda}) = \mathcal{E}^{\varnothing}(R^{17}_{c_\lambda}) \otimes_{s_{**}} \mathcal{E}^{\varnothing}(R^{20}_{c_\lambda}),$$

where $\mathcal{E}^{\varnothing}(R^{20}_{c_\lambda})$ belongs to $\mathcal{E}^{\text{ctr}}(R^{22}_{c_0})$ and $\mathcal{E}^{\varnothing}(R^{17}_{c_\lambda})$ from $\mathcal{E}^{\text{ctr}}(R^{19}_{c_0})$. From Definition 5.8, the U_1-Conditions are

$$\text{Cond}^1(R^{20}_{c_\lambda})(t) = \text{same}(U_1, y_{20}, s_{**}, t),$$
$$\text{Cond}^1(R^{17}_{c_\lambda})(t) = \text{diff}(U_1, y_{20}, s_{**}, t).$$

Then we establish a U_1-link starting from the S-node $12, 00$ and ending with the U_0-outcome of $R^{20}_{c_\lambda}$.

From Lemma 4.33, we know that $\text{Cond}^1(R^{20}_{c_\lambda})(t) = \text{same}(U_1, y_{20}, s_{**}, t)$ implies that $\text{diff}(U_1, y_{22}, s^{\text{ctr}}_{22}, t)$ and hence y_{20} is (U_1, C_λ)-restorable at stage t, and $\text{Cond}^1(R^{17}_{c_\lambda})(t) = \text{diff}(U_1, y_{20}, s_{**}, t)$ implies that $\text{diff}(U_1, y_{19}, s^{\text{ctr}}_{19}, t)$ and hence y_{17} is (U_1, C_λ)-restorable at stage t.

We now give the formal definitions of U_i-link and strong U_i-data.

Definition 5.18 (environment)**.** Let α be an R-node. Define $\text{env}^{<i}(\alpha)$ to be the shortest S-node η such that for each $j < i$, $\text{seq}(\eta)(j) = \text{seq}(\alpha^-)(j)$ (if $i = 0$, then η is defined to be the root of $\mathcal{T}$). (This node η will be the starting point of the U_i-link in the next definition.)

We say that α and β *have the same* $<i$*-environment* if $\text{env}^{<i}(\alpha) = \text{env}^{<i}(\beta)$.

A useful observation is the following: If $\beta^{*i} = \alpha$ or $\beta^{\sharp i} = \alpha$, then $\text{env}^{<i}(\alpha) = \text{env}^{<i}(\beta)$.

The following is analogous to Definition 4.23.

Definition 5.19 (strong U_i-data)**.** Suppose $m = |\mathcal{L}| + 1$. Suppose that for each $k < m$, α_a is a U^a_i-problem for the controller β_a with $\alpha_0 \subsetneq \cdots \subsetneq \alpha_{m-1}$ having the same *strong*[2] $<i$-environment. For each $a < m$, let $\mathcal{E}^{i+1}(\alpha_a)$ belong to $\mathcal{E}^{\text{ctr}}(\beta_a)$. Let s be the stage when $\mathcal{D}_s(\beta_0) = \alpha_0$. By the Pigeonhole Principle, we have for

[2]This is a subtle point and can be ignored for now. We refer the reader to Definition 5.26 and to Example 5.25 for the intuition. The existence of such a sequence is proved in Lemma 5.29.

some $0 \leq a < b < m$ and some $c \in \mathrm{Ji}(\mathcal{L})$ such that both α_a and α_b are R_c-nodes. We then define

$$\mathcal{S}^i(\alpha_b) = \mathcal{S}^{i+1}(\alpha_a) \otimes_1 \mathcal{S}^{i+1}(\alpha_b),$$
$$\mathcal{E}^i(\alpha_b) = \mathcal{E}^{i+1}(\alpha_a) \otimes_s \mathcal{E}^{i+1}(\alpha_b)$$

We also establish a U_i*-link* starting from $\mathrm{env}^{<i}(\alpha_a)$ and ending with $(\alpha_b)^\frown U_i$. It will be destroyed immediately after it is traveled.

We will tacitly assume without further proof that Lemma 4.33 applies to the strong data in the remaining examples in this section.

When we visit an S-node α with a U_i-link, we only maintain functionals that belong to $\mathrm{Maintain}(\beta, U_{<i})$ (Definition 5.3) and travel immediately along the U_i-link. Note that in $\mathcal{S}^i(\alpha_b)$ defined in the above definition, we have $g(\lambda) = 1$ and also $f(0) = f(1) = f(\lambda)$. Therefore $\mathcal{S}^i(\alpha_b)$ still satisfies Definition 5.6(3).

The key ingredients of the construction have been covered. Controllers follow the same strategies as in Section 4. Continuing Example 5.17, in the next example we quickly have a controller without problems.

Example 5.20 (Controller $R^{20}_{c_\lambda}$)**.** In Example 5.17, we obtained at stage s_{**} our first strong U_1-data $\mathcal{E}^1(R^{20}_{c_\lambda})$ and established a U_1-link starting from the S-node $12, 00$ and ending with $(R^{20}_{c_\lambda})^\frown U_0$, where the U_0-outcome is a RED outcome.

Suppose that at stage $s > s_{**}$ the U_1-link is traveled and we are encountering the RED U_0-outcome of $R^{20}_{c_\lambda}$. We are then obtain

$$\mathcal{E}^0(R^{20}_{c_\lambda}) = \mathcal{E}^1(R^{15}_{c_\lambda}) \otimes_s \mathcal{E}^1(R^{20}_{c_\lambda}) = \{R^{12}_{c_\lambda}, R^{15}_{c_\lambda}, R^{17}_{c_\lambda}, R^{20}_{c_\lambda}\},$$
$$\mathcal{S}^0(R^{20}_{c_\lambda}) = \mathcal{S}^1(R^{15}_{c_\lambda}) \otimes_0 \mathcal{S}^1(R^{20}_{c_\lambda}).$$

Then we should encounter the next outcome, the ctr-outcome of $R^{20}_{c_\lambda}$. As $R^{12}_{c_\lambda}$, $R^{15}_{c_\lambda}$, $R^{17}_{c_\lambda}$, and $R^{20}_{c_\lambda}$ are all R_{c_λ}-nodes, we have $\mathcal{S}^{\mathrm{ctr}}(R^{20}_{c_\lambda}) = \mathcal{S}^0(R^{20}_{c_\lambda})$ and $\mathcal{E}^{\mathrm{ctr}}(R^{20}_{c_\lambda}) = \mathcal{E}^0(R^{20}_{c_\lambda})$ (Definition 5.15). (Note that since $\mathcal{S}^{\mathrm{ctr}}(R^{20}_{c_\lambda}) = \mathcal{S}^0(R^{20}_{c_\lambda})$, we also have $\mathcal{E}^{\mathrm{ctr}}(R^{20}_{c_\lambda}) = \mathcal{E}^0(R^{20}_{c_\lambda})$.) Then $R^{20}_{c_\lambda}$ becomes a controller at this stage $s = s^{\mathrm{ctr}}_{R^{20}_{c_\lambda}}$, so we enumerate each diagonalizing witness into C_λ and put a restraint on $C_0 \upharpoonright s$. As in Section 4, if $\mathcal{D}_t(R^{20}_{c_\lambda}) = \xi$, then y_ξ is both U_0- and U_1-restorable. Therefore we can safely restore y_ξ if $\mathcal{D}_t(R^{20}_{c_\lambda}) = \xi$ and activate the d-outcome of ξ.

5.8. **A monstrous example.** The next example is a monstrous example in which we see how the combinatorics grows complicated.

Example 5.21. Suppose that we have $R^{15}_{c_0}$ in Figure 4. The first U_0-data would be

$$\begin{aligned}\mathcal{E}^0(R^{22}_{c_0}) &= \mathcal{E}^1(R^{15}_{c_0}) \otimes_s \mathcal{E}^1(R^{22}_{c_\lambda}) \\ &= (\mathcal{E}^{\varnothing}(R^{12}_{c_\lambda}) \otimes_s \mathcal{E}^{\varnothing}(R^{15}_{c_0})) \otimes_s (\mathcal{E}^{\varnothing}(R^{20}_{c_\lambda}) \otimes_s \mathcal{E}^{\varnothing}(R^{22}_{c_0})),\end{aligned}$$

obtained at some stage s. Then we encounter the ctr-outcome of $R^{22}_{c_0}$ and have $\mathcal{S}^{\mathrm{ctr}}(R^{22}_{c_0}) = \mathcal{S}^0(R^{22}_{c_0})$ and hence $\mathcal{E}^{\mathrm{ctr}}(R^{22}_{c_0}) = \mathcal{E}^0(R^{22}_{c_0})$. $R^{22}_{c_0}$ becomes a controller at $s_{22} = s^{\mathrm{ctr}}_{R^{22}_{c_0}} = s$. Let $s'_{22} > s_{22}$ be the stage when $\mathcal{D}_{s'_{22}}(R^{22}_{c_0}) = R^{12}_{c_\lambda}$ and $R^{22}_{c_0}$ sees no more noise from then on. Since y_{12}, the computation for $R^{12}_{c_\lambda}$, is only weakly restorable, we restore it and turn the GREEN U_1-outcome of $R^{11}_{c_0}$ RED.

Next we might reach $R^{18}_{c_0}$ at some stage $s > s'_{22} > s_{22}$ (we recycle the symbol s) through $R^{13}_{c_0}{}^\frown w$ and have

$$\begin{aligned}\mathcal{E}^0(R^{18}_{c_0}) &= \mathcal{E}^1(R^{11}_{c_0}) \otimes_s \mathcal{E}^\varnothing(R^{18}_{c_0})\\ &= (\mathcal{E}^\varnothing(R^{10}_{c_\lambda}) \otimes_s \mathcal{E}^\varnothing(R^{11}_{c_0})) \otimes_s \mathcal{E}^\varnothing(R^{18}_{c_0}),\end{aligned}$$

Again since $\mathcal{S}^{\mathrm{ctr}}(R^{18}_{c_0}) = \mathcal{S}^0(R^{18}_{c_0})$, we have $\mathcal{E}^{\mathrm{ctr}}(R^{18}_{c_0}) = \mathcal{E}^0(R^{18}_{c_0})$ and $R^{18}_{c_0}$ becomes a controller at $s_{18} = s^{\mathrm{ctr}}_{R^{18}_{c_0}} = s$. Let s'_{18} be the stage when $\mathcal{D}_{s'_{18}}(R^{18}_{c_0}) = R^{10}_{c_\lambda}$, we restore $R^{10}_{c_\lambda}$ and notice that $R^{10}_{c_\lambda}$ and $R^{12}_{c_\lambda}$, both of which are R_{c_λ}-nodes and have the same <1-environment (Definition 5.18), are a critical U^0_1-problem and a critical U^1_1-problem for the controller $R^{22}_{c_0}$ and the controller $R^{18}_{c_0}$, respectively. Thus we obtain the following strong U_1-data and U_1-data tree

$$\begin{aligned}\mathcal{E}^1(R^{12}_{c_\lambda}) &= \mathcal{E}^\varnothing(R^{10}_{c_\lambda}) \otimes_{s'_{18}} \mathcal{E}^\varnothing(R^{12}_{c_\lambda}),\\ \mathcal{S}^1(R^{12}_{c_\lambda}) &= \mathcal{S}^\varnothing(R^{10}_{c_\lambda}) \otimes_1 \mathcal{S}^\varnothing(R^{12}_{c_\lambda}),\end{aligned}$$

and establish a U_1-link starting from the S-node $10, 00$ and ending at the U_0-outcome of $R^{12}_{c_\lambda}$.

Let $s > s'_{18} > s_{18} > s'_{22} > s_{22}$ (we recycle the symbol s again) be the stage when we travel the U_1-link and visit $R^{14}_{c_0}$. If we fail to obtain $\mathcal{E}^\varnothing(R^{14}_{c_0})$ (which is always the case by the slowdown condition if this is the first time we visit $R^{14}_{c_0}$), we have to initialize the controllers $R^{22}_{c_0}$ and $R^{18}_{c_0}$ anyway. Wasting a lot of work does not matter as long as we make progress towards some nodes to the left of the controllers $R^{22}_{c_0}$ and $R^{18}_{c_0}$. If we obtain $\mathcal{E}^\varnothing(R^{14}_{c_0})$, then we encounter$(R^{14}_{c_0}, U_0)$ and obtain

$$\begin{aligned}\mathcal{E}^0(R^{14}_{c_0}) &= \mathcal{E}^1(R^{12}_{c_\lambda}) \otimes_s \mathcal{E}^\varnothing(R^{14}_{c_0}),\\ \mathcal{S}^0(R^{14}_{c_0}) &= \mathcal{S}^1(R^{12}_{c_\lambda}) \otimes_0 \mathcal{S}^\varnothing(R^{14}_{c_0}),\end{aligned}$$

where $R^{12}_{c_\lambda} = (R^{14}_{c_0})^{*0}$ and $\mathcal{E}^1(R^{14}_{c_\lambda})$ is strong U_1-data. Another important observation is the following:

(1) If s^* is the last stage when we visit $R^{14}_{c_0}$, then we have

$$s^* < s_{22} < s'_{22} < s_{18} < s'_{18} < s,$$

and by slowdown condition

$$\mathrm{SAME}(U_0, y_{14}, s^*, s).$$

Then we perform encounter$(R^{14}_{c_0}, \mathrm{ctr})$. Notice that we have $\mathcal{S}^{\mathrm{ctr}}(R^{14}_{c_0}) \neq \mathcal{S}^0(R^{14}_{c_0})$ (Definition 5.15) and hence $\mathcal{E}^{\mathrm{ctr}}(R^{14}_{c_0}) = \{R^{12}_{c_\lambda}, R^{14}_{c_0}\}$. *From this, we can see the necessity to introduce the notion of a critical problem:* If we allow $\mathcal{D}(R^{14}_{c_0}) = R^{10}_{c_\lambda}$, we have that $R^{10}_{c_\lambda}$ is a U^0_1-problem for $R^{14}_{c_0}$ but $(R^{10}_{c_\lambda})^{\sharp 1}$ is not defined. We cannot obtain any new strong U_1-data at this point. In this case, we should instead look at the critical U_0-problem $R^{12}_{c_\lambda}$.

Let $s_{14} = s^{\mathrm{ctr}}_{R^{14}_{c_0}}$ be the stage when $R^{14}_{c_0}$ becomes a controller and $s'_{14} > s_{14}$ be the stage when $\mathcal{D}_{s'_{14}}(R^{14}_{c_0}) = R^{12}_{c_\lambda}$. We will verify first that $R^{12}_{c_\lambda}$ and $R^{10}_{c_\lambda}$ are (U_0, C_λ)-restorable under $\mathrm{Cond}^0_{R^{14}_{c_0}}(R^{12}_{c_\lambda})(t) = \mathrm{diff}(U_0, y_{14}, s_{14}, t)$. In fact, this follows directly from Lemma 4.32. However, a cautious reader might be worried about the use block that was killed by the end of s_{22}. Therefore we will show that

$$\mathrm{Cond}^0_{R^{14}_{c_0}}(R^{12}_{c_\lambda})(t) \Rightarrow \mathrm{Cond}^0_{R^{22}_{c_0}}(R^{12}_{c_\lambda})(t) \wedge \mathrm{Cond}^0_{R^{18}_{c_0}}(R^{10}_{c_\lambda})(t).$$

That is,

$$\operatorname{diff}(U_0, y_{14}, s_{14}, t) \Rightarrow \operatorname{diff}(U_0, y_{20}, s_{22}, t) \wedge \operatorname{diff}(U_0, y_{18}, s_{18}, t).$$

This follows clearly from

$$y_{14} < y_{22} < y_{20} < y_{18},$$

and

$$\mathrm{SAME}(U_0, y_{14}, s^*, s_{14}).$$

Let us continue. At stage s'_{14}, we have $\mathcal{D}_{s'_{14}}(R^{14}_{c_0}) = R^{12}_{c_\lambda}$ and therefore we turn the GREEN U_0-outcome of $R^7_{c_0} = (R^{12}_{c_\lambda})^{\sharp 0}$ RED.

Let $s_7 > s'_{14}$ be the stage when $R^7_{c_0}$ becomes a controller with

$$\begin{aligned}
\mathcal{E}^{\mathrm{ctr}}(R^7_{c_0}) &= \mathcal{E}^0(R^7_{c_0}) \\
&= \mathcal{E}^{\varnothing}(R^0_{c_\lambda}) \otimes_{s_7} \mathcal{E}^1(R^7_{c_0}) \\
&= \mathcal{E}^{\varnothing}(R^0_{c_\lambda}) \otimes_{s_7} (\mathcal{E}^{\varnothing}(R^4_{c_\lambda}) \otimes_{s_7} \mathcal{E}^{\varnothing}(R^7_{c_0}))
\end{aligned}$$

We now distinguish the two cases: If $\mathcal{D}_{s'_7}(R^7_{c_0}) = R^0_{c_\lambda}$ for some $s'_7 > s_7$, then we continue in Example 5.22; if $\mathcal{D}_{s'_7}(R^7_{c_0}) = R^4_{c_\lambda}$ for some $s'_7 > s_7$, then we continue in Example 5.23.

Example 5.22 ($\mathcal{D}(R^7_{c_0}) = R^0_{c_\lambda}$)**.** Continuing Example 5.21, we suppose $\mathcal{D}_{s'_7}(R^7_{c_0}) = R^0_{c_\lambda}$. $R^0_{c_\lambda}$ is a critical U^0_0-problem for $R^7_{c_0}$, and $R^{12}_{c_\lambda}$ is a critical U^1_0-problem for $R^{14}_{c_0}$, both have the same < 0-environment. We will now obtain the following strong U_0-data and U_0-data tree:

$$\begin{aligned}
\mathcal{E}^0(R^{12}_{c_\lambda}) &= \mathcal{E}^{\varnothing}(R^0_{c_\lambda}) \otimes_{s'_7} \mathcal{E}^1(R^{12}_{c_\lambda}), \\
\mathcal{S}^0(R^{12}_{c_\lambda}) &= \mathcal{S}^{\varnothing}(R^0_{c_\lambda}) \otimes_1 \mathcal{S}^1(R^{12}_{c_\lambda}),
\end{aligned}$$

where $\mathcal{E}^1(R^{12}_{c_\lambda})$ is the strong U_1-data belonging to $\mathcal{E}^{\mathrm{ctr}}(R^{14}_{c_0})$. Thus for each $\sigma \in \mathcal{S}^0(R^{12}_{c_\lambda})$, we have $g(\sigma) = 1$ and hence $f(\sigma) = c_\lambda$ by Definition 5.6(3). We also establish a U_0-link starting from the root of the tree and ending at the ctr-outcome of $R^{12}_{c_\lambda}$.

At $s_{12} > s'_7$, the U_0-link is traveled and $R^{12}_{c_\lambda}$ becomes a controller with data $\mathcal{E}^{\mathrm{ctr}}(R^{12}_{c_\lambda}) = \mathcal{E}^0(R^{12}_{c_\lambda})$, which has no more problems.

Example 5.23 ($\mathcal{D}(R^7_{c_0}) = R^4_{c_\lambda}$)**.** Continuing Example 5.21, we suppose $\mathcal{D}_{s'_7}(R^7_{c_0}) = R^4_{c_\lambda}$. We will turn the GREEN U_1-outcome of $R^3_{c_0}$ RED. Note that the U_0-outcome of $R^3_{c_0}$ is still GREEN. Let $s_8 > s'_7$ be the stage when $R^8_{c_0}$ becomes a controller with data

$$\mathcal{E}^{\mathrm{ctr}}(R^8_{c_0}) = \mathcal{E}^0(R^8_{c_0}) = \mathcal{E}^{\varnothing}(R^5_{c_\lambda}) \otimes_{s_8} \mathcal{E}^{\varnothing}(R^8_{c_0}).$$

Let $s'_8 > s_8$ be the stage when $\mathcal{D}_{s_8}(R^8_{c_0}) = R^5_{c_\lambda}$. We will turn the GREEN U_0-outcome of $R^3_{c_0}$ RED.

Let $s_3 > s'_8$ be the stage when $R^3_{c_0}$ becomes a controller with data

$$\begin{aligned}
\mathcal{E}^{\mathrm{ctr}}(R^3_{c_0}) &= \mathcal{E}^0(R^3_{c_0}) \\
&= \mathcal{E}^{\varnothing}(R^0_{c_\lambda}) \otimes_{s_3} \mathcal{E}^1(R^3_{c_0}) \\
&= \mathcal{E}^{\varnothing}(R^0_{c_\lambda}) \otimes_{s_3} (\mathcal{E}^{\varnothing}(R^1_{c_\lambda}) \otimes_{s_3} \mathcal{E}^{\varnothing}(R^3_{c_0}))
\end{aligned}$$

Depending on the decision of $R^3_{c_0}$, we again have to split cases and consider the following two examples separately.

Example 5.24 ($\mathcal{D}(R^3_{c_0}) = R^1_{c_\lambda}$)**.** Continuing Example 5.23, we let s'_3 be the stage when we have $\mathcal{D}_{s'_3}(R^3_{c_0}) = R^1_{c_\lambda}$. Then we will obtain strong U_1-data

$$\mathcal{E}^1(R^4_{c_\lambda}) = \mathcal{E}^{\varnothing}(R^1_{c_\lambda}) \otimes_{s'_3} \mathcal{E}^{\varnothing}(R^4_{c_\lambda}),$$
$$\mathcal{S}^1(R^4_{c_\lambda}) = \mathcal{S}^{\varnothing}(R^1_{c_\lambda}) \otimes_1 \mathcal{S}^{\varnothing}(R^4_{c_\lambda}),$$

where $\mathcal{E}^{\varnothing}(R^4_{c_\lambda})$ belongs to $\mathcal{E}^{\text{ctr}}(R^7_{c_0})$. We also establish a U_1-link starting from the S-node $02, 00$ and ending at the U_0-outcome of $R^4_{c_\lambda}$.

Since the U_0-outcome of $R^4_{c_\lambda}$ is GREEN, our strong U_1-data is discarded when we visit this outcome at stage s. (Here, discarding the data is acceptable as now we are able to visit the nodes below the U_0-outcome, which is a progress.) Let us assume that $R^9_{c_0}$ becomes a controller at $s_9 = s$ with data $\mathcal{E}^{\text{ctr}}(R^9_{c_0}) = \{R^6_{c_\lambda}, R^9_{c_0}\}$. Suppose $\mathcal{D}(R^9_{c_0}) = R^6_{c_\lambda}$ at $s'_9 > s_9$ and therefore the U_0-outcome of $R^4_{c_\lambda}$ becomes RED. Then, we go over the procedures once again starting from Example 5.21 and Example 5.23 to the point when we obtain the strong U_1-data $\mathcal{E}^1(R^4_{c_\lambda})$ and establish the U_1-link starting from the S-node $02, 00$ and ending at the U_0-outcome of $R^4_{c_\lambda}$.

Let $s > s'_3$ be the stage when we travel the link. This time, as the U_0-outcome of $R^4_{c_\lambda}$ is RED, we obtain

$$\mathcal{E}^0(R^4_{c_\lambda}) = \mathcal{E}^{\varnothing}(R^0_{c_\lambda}) \otimes_s \mathcal{E}^1(R^4_{c_\lambda})$$
$$\mathcal{S}^0(R^4_{c_\lambda}) = \mathcal{S}^{\varnothing}(R^0_{c_\lambda}) \otimes_0 \mathcal{S}^1(R^4_{c_\lambda})$$

and then we perform encounter($R^4_{c_\lambda}$, ctr). $R^4_{c_\lambda}$ is then a controller without any more problems. If $R^9_{c_0}$ never sees any noise again, then the U_0-outcome of $R^4_{c_\lambda}$ remains RED forever and we never go back to $R^6_{c_\lambda}$ and $R^9_{c_0}$; if $R^9_{c_0}$ sees some noise, then $(R^4_{c_\lambda})^\frown$ ctr is initialized.

Example 5.25 ($\mathcal{D}(R^3_{c_0}) = R^0_{c_\lambda}$)**.** Continuing Example 5.23, we let s'_3 be the stage when we have $\mathcal{D}_{s'_3}(R^3_{c_0}) = R^0_{c_\lambda}$ and $R^0_{c_\lambda}$ is a critical U^0_0-problem for $R^3_{c_0}$.

Recall that for each $t \geq s'_{14}$ we assume that $\mathcal{D}_t(R^{14}_{c_0}) = R^{12}_{c_\lambda}$, which is a critical U^1_0-problem for $R^{14}_{c_0}$, and also that for each $t \geq s'_8$ we assume $\mathcal{D}_t(R^8_{c_0}) = R^5_{c_\lambda}$, which is also a critical U^1_0-problem for $R^8_{c_0}$. Now we have a choice: which U^1_0-problem do we combine with $R^0_{c_\lambda}$ to get a strong U_0-data? We refer the reader to Example 5.22 in which we do obtain the strong U_0-data by combining $R^0_{c_\lambda}$ and $R^{12}_{c_\lambda}$. However, after a moment of thought, we may prefer $R^5_{c_\lambda}$ over $R^{12}_{c_\lambda}$ in the current situation; we say that $R^5_{c_\lambda}$ and $R^0_{c_\lambda}$ has the same *strong* < 0-environment (see Definition 5.26). (To have a better intuition for our choice, we should *imagine* that the U_0-outcome of $R^3_{c_0}$ is a Type I outcome. Then $R^3_{c_0}$ itself could be a critical U^0_0-problem for some controller below the $(R^3_{c_0})^\frown U_0$. We will naturally search for a U^1_0-problem also below the $(R^3_{c_0})^\frown U_0$ instead of searching for one below the RED U_1-outcome of $R^3_{c_0}$.) Therefore, we obtain

$$\mathcal{E}^0(R^5_{c_\lambda}) = \mathcal{E}^{\varnothing}(R^0_{c_\lambda}) \otimes_s \mathcal{E}^1(R^5_{c_\lambda})$$
$$\mathcal{S}^0(R^5_{c_\lambda}) = \mathcal{S}^{\varnothing}(R^0_{c_\lambda}) \otimes_1 \mathcal{S}^1(R^5_{c_\lambda})$$

and establish a U_0-link starting from the root of the tree and ending at the ctr-outcome of $R^5_{c_\lambda}$.

At s_5, the U_0-link is traveled and $R^5_{c_\lambda}$ becomes a controller with data $\mathcal{E}^{\text{ctr}}(R^5_{c_\lambda}) = \mathcal{E}^0(R^5_{c_\lambda})$, which has no more problems.

This finishes our monstrous example.

Our final remark is the following: Our priority tree is defined in a uniform way and the examples above strictly follow this definition even though we might have used a shortcut in certain cases; but optimizing the priority tree is not our concern.

Definition 5.26. Let $\alpha \subseteq \beta$ be two nodes. We say that α and β have the same *strong* $< i$-environment at stage s if

(1) $\mathrm{env}^{<i}(\alpha) = \mathrm{env}^{<i}(\beta) = \eta$ for some S-node η. (This depends only on the priority tree.)
(2) For each ξ with $\alpha \subseteq \xi^\frown U_j \subseteq \beta$ for some $j > i$, if U_j is Type II, then U_j is GREEN.

If $\mathcal{D}(\beta) = \alpha$ where α is a critical U_i^0-problem, then we should have $\alpha = \alpha_0 \subsetneq \alpha_1 \subsetneq \cdots \subsetneq \alpha_{m-1}$ where each α_j is a U_i^j-problem and has the same strong $<$ i-environment. Then we obtain strong U_i-data according to Definition 5.19. See Lemma 5.29 for a proof.

5.9. **The construction.** At stage s, we first run the controller strategy (see below) and then the G-strategy (see Section 4.15). Then we perform visit(λ) (see below), where λ is the root of the priority tree $\mathcal{T}$. We stop the current stage whenever we perform visit(α) for some α with $|\alpha| = s$.

visit(α) *for an S-node:* Suppose that there is some U_i-link connecting α and $\beta^\frown o$ for some o-outcome of β (in fact, $o = U_{i-1}$ if $i > 0$; $o = \mathrm{ctr}$ if $i = 0$). Then we maintain each Γ-functional in Maintain($\alpha, U_{<i}$) (Definition 5.3) and perform encounter(β, o).

Suppose that there is no link. Then we maintain each Γ-functional in Maintain(α) (Definition 5.3) and stop the current substage and perform visit($\alpha^\frown 0$) for the R-node $\alpha^\frown 0$.

To build and maintain a $\Gamma^{E\oplus U} = C$ (for some $c \in \mathrm{Ji}(\mathcal{L})$) that belongs to Maintain($\alpha, U_{<i}$) or Maintain(α), depending on which case we have, we do the following: For each $x \leq s$:

(1) Suppose $\Gamma_s^{E\oplus U}(x) \downarrow = C_s(x)$. Then β does nothing else.
(2) Suppose $\Gamma_s^{E\oplus U}(x) \downarrow \neq C_s(x)$ with use block $\mathsf{B} = \mathbf{B}_s(\gamma, x)$.
    (a) If B is killed and not E-restrained, then β enumerates an unused point, referred to as a *killing point*, into B via E. Then we go to (3) immediately.
    (b) If B is not killed and not E-restrained, then β enumerates an unused point, referred to as a *correcting point*, into B via E. Then we redefine $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with the *same* use block B.
(3) Suppose $\Gamma_s^{E\oplus U}(x) \uparrow$. If each $\mathbf{B}_{\langle t|}(\gamma, x)$ with $t < s$ has been *killed*, then β will pick a fresh use block B' and define $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with use block B' (hence $\mathsf{B}' = \mathbf{B}_{\langle s|}(\gamma, x)$); otherwise, we define $\Gamma_s^{E\oplus U}(x) = C_s(x)$ with the use block that is not killed (there will be at most one such use block).

visit(α) *for an R-node:* If threshold(α) is not defined, then we define it to be a fresh number. Then we perform encounter(α, d).

Without loss of generality, we assume that α is assigned an $R_c(\Phi)$-requirement for some $c \in \mathrm{Ji}(\mathcal{L})$ and Φ; let $|\mathrm{seq}(\alpha^-)| = k$.

encounter(α, d)*:* If d is active, then we perform visit$(\alpha^\frown d)$. If d is inactive, we perform encounter(α, w).

encounter(α, w)*:* If witness(α) is not defined, then we pick a fresh number $x >$ threshold(α) and define witness$(\alpha) = x$. If a computation y is found by α, then we obtain $\mathcal{E}_s^{\varnothing}(\alpha)$ and $\mathcal{S}_s^{\varnothing}(\alpha)$ (Definition 5.7). Then we perform encounter(α, U_{k-1}), where U_{k-1} is the first outcome (recall from Definition 5.1 that we add outcomes in order).

encounter(α, U_i)*:* Inductively we must have obtained $\mathcal{E}^{i+1}(\alpha)$ (or $\mathcal{E}^{\varnothing}(\alpha)$ if $i = k-1$).

(1) If the U_i-outcome is Type I, then let $v =$ threshold(α). For each functional Γ (where Δ is dealt with similarly) that belongs to Kill$(\alpha, U_{\geq i})$ (Definition 5.4) and for each x with $v \leq x \leq s$, let $\mathsf{B}_x = \mathbf{B}_s(\gamma, x)$ be the use block. We enumerate an unused point (*killing point*) into B_x and declare that B_x is *killed*.

   Let Δ belong to Maintain(α, U_i). Without loss of generality, we assume this functional is to ensure $\Delta^{E\oplus C_0\oplus\cdots\oplus C_{r-1}} = U_i$ for some $c_0, \ldots, c_{r-1} \in$ Ji$(\mathcal{L})$.

   (a) Suppose that $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{r-1}}(x) \downarrow = U_s(x)$. Then β does nothing else.
   (b) Suppose we have $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{r-1}}(x) \downarrow \neq U_s(x)$ with use block $\mathsf{B} = \mathbf{B}_s(\delta, x)$.
      (i) If B is killed and E-free, then β enumerates an unused point, referred to as a *killing point*, into B via E. Then we go to (3) immediately.
      (ii) If B is killed and E-restrained, we let C_i $(i < k)$ be a set such that B is C_i-free (we will show such C_i exists); then β enumerates an unused point, referred to as a *killing point*, into B via C_i. B is then *permanently killed* (as C_i will be a c.e. set). Then we go to (3) immediately.
      (iii) If B is not killed and E-free, then β enumerates an unused point, referred to as a *correcting point*, into B via E. Then we define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{r-1}}(x) = U_s(x)$ with the same use block B.
      (iv) If B is not killed and E-restrained, we let C_i for some $i < k$ be a set such that B is C_i-free (we will show such C_i exists); then β enumerates an unused point, referred to as a *correcting point*, into B via C_i. Then we define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{r-1}}(x) = U_s(x)$ with the same use block B.
   (c) Suppose that $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{r-1}}(x) \uparrow$. If for each $t < s$, $\mathbf{B}_{\langle t|}(\delta, x)$ is killed, then β will choose a fresh use block B' and define
   $$\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{r-1}}(x) = U_s(x)$$
   with use block B' (hence $\mathsf{B}' = \mathbf{B}_{\langle s|}(\delta, n)$); otherwise, we will define $\Delta_s^{E\oplus C_0\oplus\cdots\oplus C_{r-1}}(x) = U_s(x)$ with the use block that is not killed (there will be at most one such use block).

   Then we stop the current substage and perform visit$(\alpha^\frown U_i)$ for the S-node $\alpha^\frown U_i$.

(2) If the U_i-outcome is GREEN, then let $v =$ threshold(α). For each functional Γ (where Δ is dealt with similarly) that belongs to Kill$(\alpha, U_{\geq i})$ and

for each x with $v \leq x \leq s$, let $\mathsf{B}_x = \mathsf{B}_s(\gamma, x)$ be the use block. We enumerate an unused point (*killing point*) into B_x and say B_x is *killed.* Then we stop the current substage and perform visit($\alpha^\frown U_i$) for the S-node $\alpha^\frown U_i$.

(3) If the U_i-outcome is RED, then we obtain $\mathcal{E}^i_s(\alpha)$ and $\mathcal{S}^i_s(\alpha)$ by Definition 5.9. Then we perform encounter(α, U_{i-1}) if $i > 0$; or encounter(α, ctr) if $i = 0$.

encounter(α, ctr)*:* Notice that we must have obtained $\mathcal{E}^0(\alpha)$ and $\mathcal{S}^0(\alpha)$. Suppose that α is an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$.

(1) Let $\mathcal{E}^{\mathrm{ctr}}(\alpha)$ and $\mathcal{S}^{\mathrm{ctr}}(\alpha)$ be obtained by Definition 5.15.
(2) Let α become a controller.
(3) We enumerate, for each $\xi \in \mathcal{E}^{\mathrm{ctr}}(\alpha)$ and each i, the diagonalizing witness into the set C if ξ is not a (critical) U_i-problem (i.e., ξ is also an R_c-node).
(4) While α is a controller, we put a restraint on $\hat{C} \upharpoonright s^{\mathrm{ctr}}_\alpha$ for each $\hat{c} \neq c$, where s^{ctr}_α is the current stage s.

Then we stop the current stage.

controller-strategy: Let β (if any) be a controller of highest priority such that β sees some noise (Definition 4.19). We initialize all nodes to the right of $\beta^\frown$ ctr. Suppose β is an R_c-node.

(1) If β sees also some threats (Definition 4.26), then we also initialize $\beta^\frown$ ctr (i.e., we discard $\mathcal{E}^{\mathrm{ctr}}(\beta)$ and $\mathcal{S}^{\mathrm{ctr}}(\beta)$).
(2) If β does not change its decision, then we do nothing.
(3) If β changes its decision and (by Lemma 4.36) $\mathcal{D}_s(\beta) = \xi$, then we restore y_ξ and set a restraint on $E \upharpoonright y_\xi$ (each use block $\mathsf{B} < y_\xi$ becomes E-restrained) until the next time β changes its decision. Furthermore,
   (a) Suppose that $\xi \in \mathcal{E}^{\mathrm{ctr}}(\beta)$ is not a critical U_i-problem for β. Then we activate the d-outcome of ξ.
   (b) Suppose that ξ is a critical U^b_i-problem for β and $b > 0$. Then we turn the GREEN U_i-outcome of $\xi^{\sharp i}$ RED (once β changes its decision or is initialized, it turns back to GREEN).
   (c) Suppose that ξ is a critical U^0_i-problem for β. Let $\xi = \alpha_0 \subsetneq \cdots \subsetneq \alpha_{m-1}$ be a sequence of nodes where each α_j is a critical U^j_i-problem for some controller and all nodes have the same strong $<i$-environment (see Lemma 5.29). We pick two nodes $\alpha_j \subsetneq \alpha_k$ which are both R_d-nodes for some $d \in \mathrm{Ji}(\mathcal{L})$. We obtain the strong U_i-data $\mathcal{E}^i(\alpha_k)$ and $\mathcal{S}^i(\alpha_k)$ (Definition 5.19) and establish a U_i-link starting from the S-node $\mathrm{env}^{<i}(\alpha_k)$ and ending at the U_{i-1}-outcome of α_k if $i > 0$; or the ctr-outcome if $i = 0$. The U_i-link will be destroyed once traveled or β changes decision.

We remark that we did not explicitly mention how big a use block should be just to avoid getting distracted by this technical issue. See Lemma 5.27.

5.10. **The verification.** We have to show that the size of each use block can actually be chosen sufficiently large so that this construction does not terminate unexpectedly.

Lemma 5.27 (Block size)**.** *Each use block can be chosen sufficiently large.*

Proof. We refer the reader to Lemma 4.27. Consider a use block B interacting with a controller β with $y_\beta < \mathsf{B}$. Recall that the number of times that $B = [a, b)$ is injured and then restored depends on the number of changes that $U \upharpoonright a$ can have. Since we have multiple U-sets now, we need to be more careful. Suppose that B is interacting with a controller β such that $U_0, \dots, U_{k-1}$ are relevant to β. According to the decision function, the number of times that $B = [a, b)$ is injured and then restored is bounded by the size of the following set

$$S = \{s \mid \text{diff}(U_i, a, s-1, s) \text{for some } i < k\},$$

which can be bounded by a computable function $p(a, k)$.

We caution the reader that when we define the use block B at α, we do not know where the controller β is located and which U-sets are relevant to β. In particular, the number $k = |\text{seq}(\beta^-)|$ is unknown to α. Therefore, we have to prepare ahead of time. We simply define B to be $[a, a+p(a, a))$ for some fresh number a and B will be sufficiently large. More precisely, we assume that $y_\beta > k$ by taking $\max\{k+1, y_\beta\}$ as the value y_β. Therefore $k < y_\beta < a < b$ and B is sufficiently large. □

The following lemma justifies the first line of the controller strategy (3), which requires that $\mathcal{D}_s(\beta)$ is always defined.

Lemma 5.28 (decision)**.** *Let β be a controller. For each $s > s^{\text{ctr}}_\beta(\beta)$, $\mathcal{D}_s(\beta)$ is defined.*

Proof. We refer the reader to Definition 5.15 for $\mathcal{E}^{\text{ctr}}(\beta)$. Our goal is to show that for each $s > s^{\text{ctr}}_\beta(\beta)$,

$$\{\xi \in \mathcal{E}^{\text{ctr}}(\beta) \mid \text{Cond}^i(\xi)(s) = 1 \text{ for each } i\} \neq \varnothing,$$

which then implies that $\mathcal{D}_s(\beta)$ is defined.

The proof proceeds by a straightforward induction with the following claim:

Claim. Given $\sigma \in \mathcal{S}^{\text{ctr}}(\beta)$ with $|\sigma| = i$. Let $y_* = \max\{y_\tau \mid \sigma 1 \subseteq \tau \in \mathcal{S}^{\text{ctr}}(\beta)\}$ and s_* be the stage when $\mathcal{E}^i(h(\sigma)) = \mathcal{E}^{i+1}(h(\sigma 0)) \otimes_{s_*} \mathcal{E}^{i+1}(h(\sigma 1))$ is obtained. If $\text{same}(U_i, y_*, s_*, s)$ *holds, then for each τ with $\sigma 1 \subseteq \tau$,* $\text{Cond}^i(h(\tau))(s) = 1$*; if* $\text{diff}(U_i, y_*, s_*, s)$ *holds, then for each τ with $\sigma 0 \subseteq \tau$,* $\text{Cond}^i(h(\tau))(s) = 1$.

Proof of the claim. The proof follows directly from Definition 5.8 and Definition 5.15. □

The following lemma with $v = 0$ justifies the controller strategy (3c).

Lemma 5.29. *Suppose that at stage s we have $\mathcal{D}_s(\beta) = \xi$ for a controller strategy β and that ξ is a critical U^v_i-problem for β (where $v < m$). Then there exists a sequence of nodes*

$$\xi = \alpha_v \subsetneq \alpha_{v+1} \subsetneq \cdots \subsetneq \alpha_{m-1},$$

where each α_k is a critical U^k_i-problem for some controller β_k with $\mathcal{D}_s(\beta_k) = \alpha_k$, and each pair of them has the same strong $< i$-environment (Definition 5.26).

Proof. For $v = m-1$, the lemma holds vacuously. So assume $v < m-1$ and suppose that the lemma holds for $v+1$, we will show that it holds for v.

Let β be the controller as in the hypothesis of the lemma, and $\mathcal{S}^{\text{ctr}}(\beta)$ be the data tree of it. Since $\xi = \alpha_v$ is a critical U^v_i-problem of β, there exists some $\sigma \in \mathcal{S}^{\text{ctr}}(\beta)$ such that $\alpha_v = h(\sigma 0)$ (by the remark after Definition 5.12). Let $\eta = h(\sigma 1) = h(\sigma)$, then $\eta^{*i} = \alpha_v$. Moreover, the data $\mathcal{E}^i(\eta) = \mathcal{E}^{i+1}(\eta) \otimes \mathcal{E}^{i+1}(\alpha_v)$ belongs to $\mathcal{E}^{\text{ctr}}(\beta)$. As a consequence, we have that

(1) $\alpha_v {}^\frown U_i \subseteq \eta \subseteq \beta$;

(2) the U_i-outcome of η is a RED outcome; and
(3) $\eta^\frown U_i$ is to the left of $\beta^\frown$ctr (allowing $\eta = \beta$).

From Item (2), there exists some controller β_{v+1} with $\mathcal{D}(\beta_{v+1}) = \alpha_{v+1}$ such that $\alpha_{v+1}^{\sharp i} = \eta$ and α_{v+1} is a critical U_i^{v+1}-problem for β_{v+1}. By induction hypothesis, we can find a sequence of nodes

$$\alpha_{v+1} \subsetneq \alpha_{v+2} \subsetneq \cdots \subsetneq \alpha_{m-1}$$

such that the lemma holds. By the remark below Definition 5.18, we have that $\mathrm{env}^{<i}(\alpha_v) = \mathrm{env}^{<i}(\alpha_{v+1})$. It remains to show that α_v and α_{v+1} satisfy Item (2) in Definition 5.26.

Suppose towards a contradiction that there exists some ρ such that $\alpha_v^\frown U_i \subseteq \rho^\frown U_j \subseteq \alpha_{v+1}$, where $j > i$ and the U_j-outcome is RED. Let χ be the controller which is responsible for turning this U_j-outcome RED; i.e., $\mathcal{D}(\chi)$ is a critical U_j-problem and $(\mathcal{D}(\chi))^{\sharp j} = \rho$.

By Items (1) and (3), either $\alpha_v^\frown U_i \subseteq \rho^\frown U_j \subseteq \eta \subseteq \beta$ or $\eta^\frown U_i \subseteq \rho^\frown U_j \subseteq \alpha_{v+1}$. In the first case, if $\chi^\frown$ ctr is to the *left* of $\beta^\frown$ ctr, then the moment χ turns the U_j-outcome of ρ RED, we will not visit any node below $\rho^\frown U_j$, and so in particular not β. This contradicts that β is a working controller at the current stage; if $\chi^\frown$ ctr is to the *right* of $\beta^\frown$ ctr, then as soon as $\mathcal{D}(\beta) = \alpha_v$, $\chi^\frown$ ctr is initialized and the U_j-outcome of ρ is reset to GREEN, a contradiction. In the second case, we have both the U_i- and the U_j-outcomes RED. If $\chi^\frown$ ctr is to the *left* of $\beta_{v+1}{}^\frown$ ctr, the moment $\chi^\frown$ ctr turns the U_j -outcome of ρ RED, β_{v+1} is initialized and we will never visit nodes below $\rho^\frown U_j$, in particular never β_{v+1}, contradicting that β_{v+1} is a working controller; if $\chi^\frown$ ctr is to the *right* of $\beta_{v+1}{}^\frown$ ctr, then as soon as $\mathcal{D}(\beta_{v+1}) = \alpha_{v+1}$, we have that χ is initialized and the U_j-outcome of ρ is reset to GREEN, a contradiction.

This completes the proof. □

The following three lemmas justify the restoration part of the controller strategy (3):

Lemma 5.30. *Suppose $\mathcal{E}^i(\beta) = \mathcal{E}^{i+1}(\alpha) \otimes_s \mathcal{E}^{i+1}(\beta)$ (Definition 5.8), where $\alpha \subsetneq \beta$ and α, β are R_d, R_c-nodes for some $d \le c \in \mathrm{Ji}(\mathcal{L})$, respectively.*

(1) For each $\xi \in \mathcal{E}^{i+1}(\alpha)$, $\mathrm{Cond}^i(\xi)(t) = \mathrm{diff}(U_i, y_\gamma, s, t)$ (where y_γ is the U_i-reference length for ξ) and $\mathrm{SAME}(\hat{D}, y_{\beta^*}, s, t)$ for each $\hat{d} \not\ge d$ implies that ξ is (U, D)-restorable (Definition 4.21) at $t > s$.
(2) For each $\xi \in \mathcal{E}^{i+1}(\beta)$, if we have $\mathrm{Cond}^i(\xi)(t) = \mathrm{same}(U_i, y_\xi, s, t)$ and $\mathrm{SAME}(\hat{C}, y_\beta, s, t)$ for each $\hat{c} \not\ge c$, then ξ is (U, C)-restorable.

Proof. If $\mathcal{E}^i(\beta)$ is weak U_i-data (Definition 5.9), then the lemma follows as in Lemma 4.32. If $\mathcal{E}^i(\beta)$ is strong U_i-data (Definition 5.19), then the lemma follows as in Lemma 4.33. □

Lemma 5.31. *Let β, an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$, be a controller with $\mathcal{E}^{\mathrm{ctr}}(\beta)$ and $\mathcal{S}^{\mathrm{ctr}}(\beta)$. Suppose $\mathcal{D}_s(\beta) = \xi$ where $\xi = h(\sigma)$ for some leaf $\sigma \in \mathcal{S}^{\mathrm{ctr}}(\beta)$. If ξ is an R_c-node, then ξ is (U_i, C)-restorable for each $i < |\sigma|$ (in this case, we say that ξ is* restorable*). If ξ is an R_d-node for some $d < c$, then ξ is (U_i, D)-restorable for each $i < |\sigma|$ (in this case, we say that ξ is* weakly restorable*).*

Proof. Induction on $\mathcal{S}^{\mathrm{ctr}}(\beta)$, using Lemma 5.30. □

Stated another way, we have the following

Lemma 5.32. *Let β be a controller where β is an R_c-node for some $c \in \mathrm{Ji}(\mathcal{L})$. Let $\xi \in \mathcal{E}^{\mathrm{ctr}}(\beta)$.*

(1) Suppose that ξ is an R_c-node. If $\mathcal{D}_s(\beta) = \xi$, then we can restore y_ξ at the beginning of stage s and activate the d-outcome of ξ and y_ξ remains restored at each substage of stage s.

(2) Suppose that ξ is an R_d-node with $d < c$. So ξ is a critical U_i^b-problem for some i and some $b > 0$. If $\mathcal{D}_s(\beta) = \xi$, then we can restore y_ξ at the beginning of stage s and turn the GREEN U_i-outcome of $\xi^{\sharp i}$ RED. Furthermore, y_ξ remains restored at each substage of stage s. □

With Lemma 5.2, Lemma 5.27, Lemma 5.28 and Lemma 5.32, the following four lemmas have essentially the same proofs as Lemma 4.37, Lemma 4.38, Lemma 4.39, and Lemma 4.40, respectively.

Lemma 5.33 (Finite Initialization Lemma)**.** *Let p be the true path. Each node $\alpha \in p$ is initialized finitely often and p is infinite.* □

Lemma 5.34. *The R^e-requirement is satisfied for each $e \in \omega$.* □

Lemma 5.35. *The $S(U_i)$-requirement is satisfied for each $i \in \omega$.* □

Lemma 5.36. *The G-requirement is satisfied.* □

6. Final remarks

Lemma 5.27 remains true when our U-sets are ω-c.e. sets. In fact, we can list all ω-c.e. sets (up to Turing degree) as a subsequence of the sets $\{U_e\}_{e\in\omega}$ where $U_e(x) = \lim_{y\to\infty} \Phi_e(x, y)$. U is a valid ω -c.e. set if $|\{y \mid \Phi_e(x, y-1) \neq \Phi_e(x, y)\}| \leq x$; and invalid otherwise. Our construction is almost unaffected — whenever we detect for some x that $|\{y \mid \Phi_e(x, y-1) \neq \Phi_e(x, y)\}| > x$, then $S(U_e)$ wins by a finite outcome (we add this additional outcome to an S-node), claiming that it is an invalid requirement and does not need to be satisfied. This yields

Theorem 6.1. *If $\mathcal{L}$ is a finite distributive lattice, then $\mathcal{L}$ can be embedded into ω-c.e. degrees as a final segment.* □

References

[AHLN18] Adaricheva, Kira V.; Hyndman, Jennifer P.; Lempp, Steffen; Nation, James B., *Interval dismantlable lattices*, Order **35** (2018), 133–-137.

[Ar88] Arslanov, Marat M., *The lattice of the degrees below* $\mathbf{0}'$, Izv. Vyssh. Uchebn. Zaved. Mat., 1988, no. 7, 27–33.

[Ar00] Arslanov, Marat M., *Open questions about the n-c.e. degrees*, in: *Computability theory and its applications (Boulder, CO, 1999)*, Contemp. Math. 257, Amer. Math. Soc., Providence, RI, 2000.

[CSS12] Cai, Mingzhong; Shore, Richard A. and Slaman, Theodore A., *The n-r.e. degrees: undecidability and* Σ_1*-substructures*, J. Math. Log. **12** (2012), paper 1250005.

[Co71] Cooper, S. Barry, *Degrees of Unsolvability*, Ph.D. Thesis, Leicester University, Leicester, England, 1971.

[CHLLS91] Cooper, S. Barry; Harrington, Leo; Lachlan, Alistair H.; Lempp, Steffen; Soare, Robert I., *The d.r.e. degrees are not dense*, Ann. Pure Appl. Logic **55** (1991), 125–151.

[CLW89] Cooper, S. Barry; Lempp, Steffen; Watson, Philip, *Weak density and cupping in the d-r.e. degrees*, Israel J. Math. **67** (1989), 137–152.

[Do89] Downey, Rodney G., *D.r.e. degrees and the nondiamond theorem*, Bull. London Math. Soc. **21** (1989), 43–50.

[Er68a] Ershov, Yuri L., *A certain hierarchy of sets I*, Algebra and Logic **7** no. 1 (1968), 47–74 (English translation pp. 25–43).

[Er68b] Ershov, Yuri L., *A certain hierarchy of sets II*, Algebra and Logic **7** no. 4 (1968), 15–47 (English translation pp. 212–232).

[Er70] Ershov, Yuri L., *A certain hierarchy of sets III*, Algebra and Logic **9** no. 1 (1970), 34–51 (English translation pp. 20–31).

[ET63] Ershov, Yurii L.; Taitslin, Mikhail A., *Undecidability of certain theories*, Algebra i Logika Sem. **2** no. 5 (1963), 37–41.

[Gr11] Grätzer, George, *Lattice Theory: Foundation*, Birkhäuser/Springer Basel AG, Basel, 2011.

[La68] Lachlan, Alistair H., *Distributive initial segments of the degrees of unsolvability*, Z. Math. Logik Grundlagen Math. **14** (1968), 457–472.

[LeLN] Lempp, Steffen, *Lecture Notes on Priority Arguments*, preprint available on line at web site `http://www.math.wisc.edu/~lempp/papers/prio.pdf`.

[Le83] Lerman, Manuel, *Degrees of unsolvability. Local and global theory*, Perspect. Math. Logic, Springer-Verlag, Berlin, 1983, xiii+307 pp.

[Li17] Liu, Yong, *The structure of d.r.e. degrees*, Ph.D. Thesis, National University of Singapore, Singapore, 2017.

[Pu65] Putnam, Hilary W., *Trial and error predicates and the solution to a problem of Mostowski*, J. Symbolic Logic **30** (1965), 49–57.

[So87] Soare, Robert I., *Recursively enumerable sets and degrees*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1987.

(Lempp) Department of Mathematics, University of Wisconsin, 480 Lincoln Drive, Madison, WI 53706-1325, USA

Email address: `lempp@math.wisc.edu`
URL: `http://www.math.wisc.edu/~lempp/`

(Yiqun Liu) Office of the President, National University of Singapore, 21 Lower Kent Ridge Road, Singapore 119077, SINGAPORE

Email address: `liuyq@nus.edu.sg`

(Yong Liu) School of Information Engineering, Nanjing Xiaozhuang University, No. 3601 Hongjing Avenue, Nanjing 211171, CHINA

Email address: `liuyong@njxzc.edu.cn`

(Ng, Wu) Division of Mathematical Sciences, School of Physical and Mathematical Sciences, College of Science, Nanyang Technological University, Singapore 639798, SINGAPORE

Email address, Ng: `kmng@ntu.edu.sg`
URL: `http://www.ntu.edu.sg/home/kmng/`

Email address, Wu: `guohua@ntu.edu.sg`
URL: `http://www3.ntu.edu.sg/home/guohua/`

(Peng) Institute of Mathematics, Hebei University of Technology, No. 5340 Xiping Road Tianjin 300401, CHINA

Email address: `pengcheng@hebut.edu.cn`