Canad. J. Math. 2026, pp. 1–26
http://dx.doi.org/10.4153/S0008414X25101922
© The Author(s), 2025. Published by Cambridge University Press on behalf of Canadian Mathematical Society

The subTuring degrees

Takayuki Kihara and Keng Meng Ng

Abstract. In this article, we introduce a notion of reducibility for partial functions on the natural numbers, which we call subTuring reducibility. One important aspect is that the subTuring degrees correspond to the structure of the realizability subtoposes of the effective topos. We show that the subTuring degrees (i.e., the realizability subtoposes of the effective topos) form a dense non-modular (thus, non-distributive) lattice. We also show that there is a nonzero join-irreducible subTuring degree (which implies that there is a realizability subtopos of the effective topos that cannot be the meet of two larger realizability subtoposes of the effective topos).

1 Introduction

1.1 Background

The analysis of the degrees of non-computability is a central subject in computability theory [4, 13]. The key notion used to compare degrees of non-computability of *total* functions on the natural numbers is *Turing reducibility*, and a huge amount of research has been done on it. The best known reducibility for *partial* functions on the natural numbers is Kleene's relative partial computability or nondeterministic Turing reducibility [4, Chapter 11], which coincides with *enumeration reducibility* for the graphs of partial functions. Therefore, it has been widely believed that the study of degrees of partial functions can be absorbed into the theory of enumeration degrees – another field that has developed greatly, like the theory of Turing degrees (see the recent survey [15]).

However, unlike the case of total functions, other candidates for the notion of reducibility for partial functions have been proposed. Sasso [14] used a type of relative computation with sequential (non-parallel) access to an oracle to introduce a reducibility notion between partial functions. This degree structure, which Sasso called the T-degrees of partial functions, has been little studied since then in degree theory, but the importance of Sasso's degree structure has been highlighted in a context quite different from that of degree theory.

Received by the editors November 23, 2024; revised September 5, 2025; accepted September 5, 2025.
Published online on Cambridge Core December 17, 2025.
The first-named author was supported by JSPS KAKENHI (Grant Numbers 21H03392, 22K03401, and 23K28036). The second-named author was supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MOE-T2EP20222-0018) and Academic Research Fund Tier 1 (RG104/24).
AMS subject classification: **03D30**, 03D80.
Keywords: Turing degree, realizability, partial function.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

In constructive mathematics, the pioneering work was done by Goodman [7], who used a model of relative computation similar to Sasso's to show the so-called Goodman's theorem: The system HA^ω of finite-type Heyting arithmetic plus the axiom of choice AC is conservative over Heyting arithmetic HA. The proof uses the realizability interpretation relative to a generic partial choice function (see also [2, 16]). The notion of computability relative to a partial function used here is precisely the notion of *subTuring reducibility* (Definition 2.1), which is a slight modification of Sasso's definition (introduced by van Oosten [19], Madore [12], and later by Kihara [9]).

The exact same reducibility has also been used in van Oosten's semantical proof of de Jongh's theorem [17], for example: For a formula A in intuitionistic predicate calculus IQC, IQC proves A iff HA proves every arithmetical substitution instance of A. The proof of backward direction is done by replacing predicate symbols with mutually generic arithmetical formulas. The tool used here is a pca-valued sheaf (pca stands for partial combinatory algebra). Each pca used in the proof is of the form $K_1[f]$, where $K_1[f]$ represents a model of computability relative to a partial function f. To be more precise, K_1 stands for Kleene's first algebra; that is, the standard model of computability on the natural numbers, and the relative computability is precisely subTuring reducibility.

Later, van Oosten [18] used the same relative computability, which he called a dialogue, to introduce the type structure of finite-type functionals based on partial functions. Given a partial combinatory algebra (pca) A, the dialogue for a partial function f yields a new algebra $A[f]$ (see, e.g., [19]). As one of the notable results, Faber–van Oosten [6, Corollary 2.13] showed that a realizability subtopos (i.e., a realizability topos which is a subtopos) of the effective topos $\mathcal{E}ff$ is nothing but the realizability topos over $K_1[f]$ for some partial function f. This shows that the structure of the subTuring degrees of partial functions on the natural numbers corresponds exactly to the structure of the realizability subtoposes of the effective topos $\mathcal{E}ff$. This extends Hyland's earlier result [8] that there is an embedding of the Turing degrees into the reverse order of subtoposes of the effective topos.

In short, a modification of Sasso's model of sequential computation, i.e., subTuring reducibility, was essential in various works on constructive mathematics, realizability topos theory, etc. For these reasons, Kihara [9] has isolated this notion of subTuring reducibility, and raised the analysis of its structure as an important problem. The issue of structural analysis of subTuring degrees has also been raised by Madore [12].

This article addresses the problem [9, 12] of analyzing the structure of the subTuring degrees (in other words, the structure of the realizability subtoposes of $\mathcal{E}ff$). In this article, we only aim to perform a pure degree-theoretic analysis. As a pure degree theory, our research complements the perspective on partial functions that classical computability theorists have overlooked for many years. Our work provides novel techniques for computability theory and is interesting in its own right as a pure degree theory. As a (semi)constructive mathematics, in the subsequent article [11], we will apply the powerful techniques we have developed in this article to various model constructions in (semi)constructive mathematics.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

1.2 Summary of results

In Section 2, we introduce the notion of subTuring reducibility (Definition 2.1). In Section 3, we show that the subTuring degrees form a lattice (Proposition 3.1), but are not σ-complete. Indeed, no strictly increasing sequence of subTuring degrees has a supremum (Proposition 3.4). Similarly, no strictly decreasing sequence of subTuring degrees has an infimum (Proposition 3.5). We introduce the notion of subTuring jump, and show that the subTuring jump has a fixed point (Theorem 3.9). In Section 4, we show that the subTuring degrees are dense (Theorem 4.1); indeed, any nonempty interval of subTuring degrees has continuum many incomparable degrees (Theorem 4.4). We introduce the notion of quasiminimal degree, and we prove the quasiminimal jump inversion theorem for subTuring degrees; that is, any subTuring degree above the halting problem is the subTuring jump of a quasiminimal subTuring degree (Theorem 4.5). We show that the subTuring meet of partial functions cannot be the degree of a total function (Theorem 4.6). In particular, no minimal pair of subTuring degrees of total functions exists (Corollary 4.8). We show that the lattice of the subTuring degrees is not distributive (Theorem 4.9); indeed, not modular (Theorem 4.10). Hence, the realizability subtoposes of the effective topos form a dense, non-distributive, non-modular lattice. We finally show that there is a nonzero join-irreducible subTuring degree (Theorem 4.12), which implies that there is a realizability subtopos of the effective topos that cannot be the meet of two larger realizability subtoposes of the effective topos (with respect to the ordering by geometric inclusions).

2 Preliminaries

2.1 Notations

For the basics of computability theory, see [4, 13]. Let ω denote the set of all natural numbers, and $\omega^{<\omega}$ denote the set of all finite strings on ω. We use the notation $f :\subseteq X \to Y$ to indicate that f is a partial function from X to Y; that is, the domain of f is a subset of X and the codomain is Y. Here, the domain, $\mathrm{dom}(f)$, is the set of all $x \in X$ such that $f(x)$ is defined. If $x \in \mathrm{dom}(f)$, we often write $f(x)\downarrow$; otherwise, $f(x)\uparrow$. For a set $A \subseteq X$, let $f \upharpoonright A$ denote the restriction of f up to A; that is, $\mathrm{dom}(f \upharpoonright A) = \mathrm{dom}(f) \cap A$, and $(f \upharpoonright A)(x) = f(x)$ for any $x \in \mathrm{dom}(f \upharpoonright A)$. We often identify n with $\{0, 1, \ldots, n-1\}$, and for $f :\subseteq \omega \to \omega$, we use the notation $f \upharpoonright n$ to mean $f \upharpoonright \{0, 1 \ldots, n-1\}$.

2.2 Definition

In Sasso's model of relative computation, access to an oracle is always sequential, so parallel access is not allowed. In other words, if we make a query to an oracle during a computation, we must wait until the oracle replies. However, if the oracle is a partial function, it is possible that the oracle will forever fail to return, in which case the computation will be stuck. Therefore, a query we make must always be a value in the domain of the oracle if we want the computation to halt.

Let us formulate this Sasso's idea rigorously. For a partial computable function $\Phi :\subseteq \omega^{<\omega} \to 2 \times \omega$, the g-relative sequential computation $\Phi[g]$ is defined as follows:

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

(1) Assume that n is given as input, the oracle's responses $(a_0, \dots, a_{s-1})$ are given by the sth round, and the computation has not yet halted.
(2) If $\Phi(n, a_0, \dots, a_{s-1})$ does not halt, then neither does $\Phi[g](n)$ (written $\Phi[g](n)\uparrow$). Otherwise, we have $\Phi(n, a_0, \dots, a_{s-1})\downarrow = \langle i_s, q_s\rangle$.
   (a) If $i_s = 1$, the computation halts and q_s is the output of this computation.
   (b) If $i_s = 0$, proceed to the next step (3).
(3) If $q_s \in \mathrm{dom}(g)$, the computation continues with $a_s = g(q_s)$. If $q_s \notin \mathrm{dom}(g)$, the computation never halts.

If the computation arrives at (2a) above, we write $\Phi[g](n)\downarrow = q_s$. If the computation goes to (2b), we call q_s a query. Note that this sequential computation has the following monotonicity: $f \subseteq g$ implies $\Phi[f] \subseteq \Phi[g]$.

Definition 2.1 For partial functions $f, g :\subseteq \omega \to \omega$, we say that *$f$ is subTuring reducible to g* (written $f \leq_{\mathrm{subT}} g$) if there exists a partial computable function Φ such that $f \subseteq \Phi[g]$; that is, for any $n \in \mathrm{dom}(f)$, we have $\Phi[g](n)\downarrow = f(n)$.

We write $f \equiv_{\mathrm{subT}} g$ if $f \leq_{\mathrm{subT}} g$ and $g \leq_{\mathrm{subT}} f$. As usual, each equivalence class of the equivalence relation $\equiv_{\mathrm{subT}}$ is called a *subTuring degree*.

Roughly speaking, $f \leq_{\mathrm{subT}} g$ if there exists a Turing functional Φ which computes $\Phi^g(n)\downarrow = f(n)$ without making a query outside of $\mathrm{dom}(g)$, whenever an input n is given from $\mathrm{dom}(f)$. Hereafter, $\Phi[g]$ is sometimes written as Φ^g.

Remark 2.1 Sasso's Turing reducibility $f \leq_T g$ is defined as $f = \Phi[g]$; that is, it requires the condition $\mathrm{dom}(f) = \mathrm{dom}(\Phi[g])$. Thus, $f \leq_{\mathrm{subT}} g$ if and only if f has an extension $\hat{f}$ with $\hat{f} \leq_T g$.

In particular, if we consider only total functions, the subTuring reducibility agrees exactly with the usual Turing reducibility.

Observation 2.2 *If g is total, then $g \leq_T f$ if and only if $g \leq_{\mathrm{subT}} f$.*

Remark 2.3 The order $\leq_{\mathrm{subT}}$ coincides with $\leq_{\mathrm{K}_1}$ in the sense of van Oosten [19, Remark 2.5]. This also coincides with the restriction of generalized Weihrauch reducibility to partial single-valued functions on ω (see [9]). The problem of structural analysis of the subTuring degrees was first posed by Madore [12].

2.3 Basics

Restricting to binary functions does not change the structure.

Observation 2.4 *Any partial function $f :\subseteq \omega \to \omega$ is subTuring equivalent to a partial binary function $G_f :\subseteq \omega \to 2$.*

Proof Let G_f be the graph of f in the following sense:

$$G_f(n, m) = \begin{cases} 1 & \text{if } f(n)\downarrow = m \\ 0 & \text{if } f(n)\downarrow \neq m \\ \uparrow & \text{if } f(n)\uparrow . \end{cases}$$

For $G_f \leq_{\mathrm{subT}} f$, to compute $G_f(n, m)$, just ask for the value of $f(n)$, where $(n, m) \in \mathrm{dom}(G_f)$ implies $n \in \mathrm{dom}(f)$. For $f \leq_{\mathrm{subT}} G_f$, to compute $f(n)$, ask for the values

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

of $G_f(n,0), G_f(n,1), G_f(n,2), \ldots$, where $n \in \mathrm{dom}(f)$ implies $(n,m) \in \mathrm{dom}(G_f)$ for any m. If $n \in \mathrm{dom}(f)$, then the answer $G(n,k) = 1$ will be returned at some point, so we can output k at that time. ∎

First, the minimum requirement for a reducibility notion is that it be preordered.

Observation 2.5 *$\leq_{\mathrm{subT}}$ is transitive.*

Proof Assume that $f \leq_{\mathrm{subT}} g \leq_{\mathrm{subT}} h$. Then, we get $f \subseteq \Phi[g]$ and $g \subseteq \Psi[h]$. By monotonicity, we get $f \subseteq \Phi[g] \subseteq \Phi[\Psi[h]]$. By composing the two sequential computations, we get $f \leq_{\mathrm{subT}} h$. ∎

Observation 2.6 *The least subTuring degree consists of exactly those functions with a partial computable extension.*

A subTuring degree is *total* if it contains a total function. For partial functions $f, g :\subseteq \omega \to \omega$, the join $f \oplus g$ is defined as usual:

$$(f \oplus g)(2n) = f(n); \qquad (f \oplus g)(2n+1) = g(n).$$

Observation 2.7 *If the domain of $f :\subseteq \omega \to \omega$ is c.e. relative to a total function $g : \omega \to \omega$, then $f \oplus g$ has a total subTuring degree.*

Proof Let D_s be the stage s approximation of the domain of f with respect to its g-computable enumeration. Define $\tilde{f}(n,s) = f(n) + 1$ if $n \in D_s$; otherwise, $\tilde{f}(n,s) = 0$. Then, $\tilde{f} \oplus g$ is total. Obviously, $\tilde{f} \oplus g \leq_{\mathrm{subT}} f \oplus g$. For $f \oplus g \leq_{\mathrm{subT}} \tilde{f} \oplus g$, to compute $f(n)$, search for s such that $\tilde{f}(n,s) > 0$. ∎

In particular, if the domain of $f :\subseteq \omega \to \omega$ is c.e., then f has a total subTuring degree. Considering $g = \chi_{\mathrm{dom}(f)}$, one also sees that if the domain of $f :\subseteq \omega \to \omega$ is computable relative to f, i.e., $\chi_{\mathrm{dom}(f)} \leq_{\mathrm{subT}} f$, then f has a total subTuring degree.

2.4 Realizability theory

To emphasize the importance of subTuring degrees, as mentioned in Section 1.1, this is not only related to pure computability theory, but also to realizability theory [20]. The purpose of Section 2.4 is to explain this connection, and since it will not be used in the remaining sections, readers who are only interested in pure computability theory may skip this section.

Definition 2.2 A pca is a set equipped with a partial binary operation which is combinatory complete (see [20]). For a partial function $f :\subseteq \omega \to \omega$, we consider the pca $K_1[f] = (\omega, *_f)$ of f-relative computability; that is, the partial binary operation $*_f :\subseteq \omega^2 \to \omega$ is defined as follows:

$$e *_f n \downarrow = m \iff \Phi_e[f](n) \downarrow = m,$$

where Φ_e is the eth partial computable function.

Let $\mathsf{RT}(A)$ be the realizability topos induced by a pca A.

Fact 1 [6, Corollary 2.13] *Every realizability subtopos of the effective topos is equivalent to one of the form $\mathsf{RT}(K_1[f])$ for some partial function f on the natural numbers.*

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Moreover, there is an inclusion of toposes $\mathsf{RT}(K_1[f]) \hookrightarrow \mathsf{RT}(K_1[g])$ *if and only if* $g \leq_{\mathrm{subT}} f$.

Assuming the former assertion, the latter assertion can also be derived from the correspondence between the subTuring degrees and some suborder of the poset of Lawvere–Tierney topologies (see also [9, 10]). The important point is that this is based on subTuring reducibility, and not on nondeterministic Turing reducibility (enumeration reducibility). In this way, the theory of subTuring degrees can be seen as a classification theory of the realizability subtoposes of the effective topos, or a classification theory of K_1-based pcas.

3 Basic operations on subTuring degrees

3.1 Lattice structure

Let us examine the basic structure of the subTuring degrees. Most degree structures studied in computability theory are upper semilattice, but they often do not have a meet (consider the Turing degrees, the many-one degrees, the truth-table degrees, the enumeration degrees, and so on).

Degree structures with meet are also known, but they are degree structures for problems with many solutions (e.g., the Medvedev degrees, the Muchnik degrees, and the Weihrauch degrees), and in the context of this article, they correspond to degree structures for multi-valued functions. Surprisingly, the subTuring degree structure has a meet even though it deals with single-valued functions.

Proposition 3.1 *The subTuring degrees form a lattice.*

Proof The join is given by $f \oplus g$. The meet $f \cap g$ is defined as follows:

$$(f \cap g)(d, e, n) = \begin{cases} \Phi_d[f](n) & \text{if } \Phi_d[f](n)\downarrow = \Phi_e[g](n)\downarrow \\ \uparrow & \text{otherwise.} \end{cases}$$

For $f \cap g \leq_{\mathrm{subT}} f$, to compute $(f \cap g)(d, e, n)$, simulate $\Phi_d[f](n)$. If $(d, e, n) \in \mathrm{dom}(f \cap g)$, then $\Phi_d[f](n)\downarrow$ by definition, so any queries made during the computation are contained within the domain of f. The same applies to $f \cap g \leq_{\mathrm{subT}} g$.

To show that $f \cap g$ is the greatest lower bound of f and g, assume $h \leq_{\mathrm{subT}} f, g$. Then, there exist d and e such that $h \subseteq \Phi_d[f], \Phi_e[g]$. This means that $n \in \mathrm{dom}(h)$ implies $\Phi_d[f](n)\downarrow = \Phi_e[g](n)\downarrow = h(n)$, so we have $(d, e, n) \in \mathrm{dom}(f \cap g)$ and $(f \cap g)(d, e, n) = h(n)$. This shows $h \leq_{\mathrm{subT}} f \cap g$. ∎

Thus, the subTuring degrees have the property of being a lattice, which is very rare in degree theory. A few other degree structures that are lattices, such as Medvedev and Weihrauch degrees, are known to be distributive lattices. Is the subTuring lattice also distributive? Quite interestingly, the subTuring lattice is non-distributive! The result that it is a lattice, and that it is not distributive despite being a lattice, implies that it is not elementarily equivalent to other known degree structures.

The proof of non-distributivity is a bit technical, so it is better to get used to the basic proof techniques first, so we leave the proof for later (Theorem 4.9).

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Remark 3.2 As a reviewer have pointed out, the strong Weihrauch degrees of multifunctions also form a non-distributive lattice [5]; however, the reverse Medvedev degrees embed into the strong Weihrauch degrees as an initial segment, so the strong Weihrauch degrees are not dense. In contrast, we show that the subTuring degrees are dense (Theorem 4.1). Hence, the subTuring lattice is not elementarily equivalent to the strong Weihrauch lattice.

Remark 3.3 Combining Fact 1 and Proposition 3.1 shows that the realizability subtoposes of the effective topos form a non-distributive lattice. On the other hand, the fact that the subtoposes of a topos forms a distributive lattice (in fact, a coHeyting algebra) shows that the lattice operations on the realizability subtoposes differs from those on the whole subtoposes. Indeed, the lattice operations on the latter are extensions of those on the Weihrauch lattice (see [10]). Comparing the explicit descriptions, we see that while the subTuring/Weihrauch joins are the same, the meets differ.

3.2 Completeness

Next, let us discuss countable joins and meets. There is no nontrivial countable supremum in several degree structures, including the Turing degrees. Spector's proof uses the notion of an exact pair, which leads to the absence of a meet, but is not applicable to the subTuring degrees by Proposition 3.1. Nevertheless, it is easy to prove the absence of nontrivial countable supremum in the subTuring degrees using a similar method.

Proposition 3.4 *No strictly increasing ω-sequence of subTuring degrees has a supremum.*

Proof Assume that we are given a countable sequence $(g_n)_{n\in\omega}$ of partial functions of increasing subTuring degrees. Let us check that given any upper bound h of $(g_n)_{n\in\omega}$, we can construct another upper bound f such that $h \not\leq_{\mathrm{subT}} f$. For this purpose, we construct a strictly increasing sequence $(a_n)_{n\in\omega}$ and guarantee that $f(a_n, x) = g_n(x)$. Also, let $f(b, x)$ be undefined if $b \neq a_n$ for any $n \in \omega$. In this case, it is clear that $g_n \leq_{\mathrm{subT}} f$ for any $n \in \omega$, so we want to guarantee $h \not\leq_{\mathrm{subT}} f$ by choosing a_n appropriately.

We describe a strategy to ensure $h \not\leq_{\mathrm{subT}} f$ via e, i.e., $h \notin \Phi_e[f]$. Assume that $(a_n)_{n\leq e}$ has already been constructed and that $f(b, x)$ is determined for any $b \leq a_e$. At stage e, perform the following actions: Ask if the computation of $\Phi_e[f](n)$ for some input $n \in \mathrm{dom}(h)$ makes a query (b, x) for some $b > a_e$ and $x \in \omega$.

Case (1): If yes, declare $f(b, x)\uparrow$. In other words, put $a_{e+1} = b + 1$ and proceed to the next stage $e + 1$. In this case, the computation of $\Phi_e[f](n)$ for some $n \in \mathrm{dom}(h)$ makes a query to $(b, x) \notin \mathrm{dom}(f)$, which implies $\Phi_e[f](n)\uparrow$. In particular, we have $h \notin \Phi_e[f]$.

Case (2): If no, then do nothing and proceed to the next stage $e + 1$. In this case, only queries whose first coordinates are at most a_e are asked to f during the computation of $\Phi_e[f](n)$ for any $n \in \mathrm{dom}(h)$. By our definition of f, using $g_0 \oplus \cdots \oplus g_e$, one can easily compute $f(b, x)$ for any $b \leq a_e$ and $x \in \omega$. In other words, $f \upharpoonright (a_e + 1) \times \omega \leq_{\mathrm{subT}} g_0 \oplus \cdots \oplus g_e \leq_{\mathrm{subT}} g_e$. By assumption, we have $h \not\leq_{\mathrm{subT}} g_e$, so $h \not\leq_{\mathrm{subT}} f \upharpoonright (a_e + 1) \times$

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

ω. However, the computation of $\Phi_e[f]$ is performed by accessing only to $f \upharpoonright (a_e + 1) \times \omega$, so computing h must have failed somewhere. Hence, we get $h \not\in \Phi_e[f]$. ■

By a somewhat dual argument, the absence of nontrivial countable infimum can also be easily proved.

Proposition 3.5 *No strictly decreasing ω-sequence of subTuring degrees has an infimum.*

Proof Assume that we are given a countable sequence $(g_n)_{n\in\omega}$ of partial functions of decreasing subTuring degrees. Let us check that given any lower bound h of $(g_n)_{n\in\omega}$, we can construct another lower bound $f \not\leq_{\mathrm{subT}} h$. Now, instead of g_n, consider $g_n^* = g_0 \cap \cdots \cap g_n$. For simplicity, we can assume that an input of g_n^* is of the form $(e_0, \dots, e_n, x)$. That is, $g_n^*(e_0, \dots, e_n, x)$ is the common value of $\Phi_{e_0}[g_0](x), \dots, \Phi_{e_n}[g_n](x)$ if it is defined. In particular, we have $\mathrm{dom}(g_n^*) \subseteq \omega^{n+2}$, so we can assume that the domains of g_n^*'s are pairwise disjoint. We construct a strictly increasing sequence $(a_n)_{n\in\omega}$ such that $a_n \in \mathrm{dom}(g_n^*)$ and define $f(a_n) = g_n^*(a_n)$. Also, let $f(b)$ be undefined if $b \neq a_n$ for any $n \in \omega$.

First, let us check $f \leq_{\mathrm{subT}} g_n^*$ no matter how we choose $(a_n)_{n\in\omega}$. For each $m \geq n$, if a_m is of the form $\langle e_0, \dots, e_m, x\rangle$ and $f(a_m)\downarrow$, then the value of $f(a_m)$ is the common value of $\Phi_{e_0}[g_0](x), \dots, \Phi_{e_m}[g_m](x)$, which is especially the common value of $\Phi_{e_0}[g_0](x), \dots, \Phi_{e_n}[g_n](x)$. Therefore, it follows that $f(a_m) = g_m^*(e_0, \dots, e_m, x) = g_n^*(e_0, \dots, e_n, x)$. In this way, using g_n^*, one can compute f except for finitely many values, so we obtain $f \leq_{\mathrm{subT}} g_n^*$. Now, we want to guarantee $f \not\leq_{\mathrm{subT}} h$ by choosing a_n appropriately.

We describe a strategy to ensure $f \not\leq_{\mathrm{subT}} h$ via e, i.e., $f \not\in \Phi_e[h]$. Assume that $(a_n)_{n<e}$ has already been constructed. At stage e, perform the following actions: Find $a_e \in \mathrm{dom}(g_e)$ such that $a_e > a_{e-1}$ and $\Phi_e[h](a_e) \neq g_e^*(a_e)$. Since $g_e^* \not\leq_{\mathrm{subT}} h$ by assumption, such a_e must exist. Then proceed to the next stage $e+1$. Note that $a_e \in \mathrm{dom}(g_e)$ implies $a_e \in \mathrm{dom}(f)$ by definition. Therefore, our strategy ensures $\Phi_e[h](a_e) \neq f(a_e)$, which implies $f \not\in \Phi_e[h]$. ■

3.3 Jump operator

For Turing degrees, there is an important operator called the Turing jump, which is defined as the relative halting problem. There is also the enumeration jump operator for the enumeration degrees, which is an extension of the Turing jump operator. It is hoped, then, that the notion of jump for the subTuring degrees can also be introduced. It is natural to require, for example, the following for the subTuring jump operator J:

(1) Inflationary: $f \leq_{\mathrm{subT}} J(f)$.
(2) Monotone: $f \leq_{\mathrm{subT}} g \implies J(f) \leq_{\mathrm{subT}} J(g)$.
(3) Conservative: f total $\implies J(f) \equiv_{\mathrm{subT}} f'$.
(4) Strict: $J(f) \not\leq_{\mathrm{subT}} f$.

Here, f' denotes the Turing jump of f. However, currently, no operator J has been found that satisfies all of these properties. For example, the most obvious candidate

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

for J is probably the relative halting problem, as usual:

$$K_0(f)(e) = \begin{cases} 1 & \text{if } \Phi_e[f](e)\downarrow \\ 0 & \text{if } \Phi_e[f](e)\uparrow . \end{cases}$$

However, this is not appropriate for a jump operator in the following sense.

Observation 3.6 *K_0 is not monotone.*

Proof Observe $\chi_{\mathrm{dom}(f)} \leq_{\mathrm{subT}} K_0(f)$. This is because if $\Phi_{d(n)}$ is a computation that asks the query n and halts regardless of the solution, then being $n \in \mathrm{dom}(f)$ and $\Phi^f_{d(n)}(d(n))$ halting are equivalent, which is equivalent to $d(n) \in K_0(f)(d(n))$.

If f is a total function, then $f \upharpoonright A \leq_{\mathrm{subT}} f$ for any A. Let us take A such that $\chi_A \not\leq_{\mathrm{subT}} K_0(f)$. If K_0 is monotone, then we get $\chi_A \leq_{\mathrm{subT}} K_0(f \upharpoonright A) \leq_{\mathrm{subT}} K_0(f)$, which is a contradiction. ∎

However, we cannot give up the monotonicity necessary to guarantee that the jump operator is well-defined on the subTuring degrees. Considering what went wrong, we just defined $K_0(f)$ as the relative halting problem, but we must not forget that in our relative computation, there are two kinds of reasons for "not halting." That is, one is simply that a machine (a Turing functional Φ) does not halt, and the other is that the query was made outside the domain of an oracle and we are waiting forever for the oracle's response. Thus, it is reasonable to define the relative halting problem as follows:

$$K(f)(e) = \begin{cases} 1 & \text{if } \Phi_e[f](e)\downarrow \\ 0 & \text{if } \Phi_e[f](e)\uparrow \text{ without making a query outside of } \mathrm{dom}(f), \\ \uparrow & \text{if } \Phi_e[f](e)\uparrow \text{ because of making a query outside of } \mathrm{dom}(f). \end{cases}$$

This is similar to the way in which a feedback Turing machine [1] distinguishes between non-halting and freezing computations, but we have not yet figured out the formal correspondence. This relative halting problem K has the expected properties.

Proposition 3.7 *K satisfies the following conditions:*

(1) *Uniformly inflationary: $f \leq_{\mathrm{subT}} K(f)$ via a single index.*
(2) *Uniformly monotone: $f \leq_{\mathrm{subT}} g \implies K(f) \leq_{\mathrm{subT}} K(g)$ uniformly.*
(3) *Uniformly conservative: f total $\implies K(f) \equiv_{\mathrm{subT}} f'$ via a single index.*

Proof (1) One can easily find an index $i(n)$ such that $\Phi_{i(n)}[f](x)$ simulates $f(n)$ no matter what f and x are. Then, we have $f(n) = K(f)(i(n))$; hence, $f \leq_{\mathrm{subT}} K(f)$ via an index of $\lambda n.i(n)$.

(2) Assume $f \leq_{\mathrm{subT}} g$ via d; that is, $f \subseteq \Phi_d[g]$. If $e \in \mathrm{dom}(K(f))$, $\Phi_e[f](e)$ makes queries only within the domain of f. Therefore, the replies to these queries by f and $\Phi_d[g]$ are identical. That is, $\Phi_e[f](e)$ and $\Phi_e[\Phi_d[g]](e)$ perform the same computation, especially whether they halt or not. One can effectively find an index $b(d,e)$ such that $\Phi_{b(d,e)}[g](x)$ simulates $\Phi_e[\Phi_d[g]](e)$ no matter what g and x are. Then, we get $K(f)(e) = K(g)(b(d,e))$. Consequently, $f \leq_{\mathrm{subT}} g$ via d implies $K(f) \leq_{\mathrm{subT}} K(g)$ via an index of $\lambda e.b(d,e)$.

(3) If f is total, then so is $K(f)$. Thus, $K(f) = K_0(f) = f'$. ∎

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

From these properties, we can automatically derive the strictness of K on hyperarithmetical functions. We say that a partial function f is hyperarithmetical if it extends to a total function whose graph is Δ^1_1 (see, e.g., [3] for Δ^1_1).

Proposition 3.8 *If J is uniformly inflationary, uniformly monotone, and uniformly conservative, then $f <_{\mathrm{subT}} J(f)$ for any partial hyperarithmetical function f. Indeed, $f \equiv_{\mathrm{subT}} J(f)$ implies $\varnothing^{(\alpha)} <_{\mathrm{subT}} f$ for any computable ordinal α.*

Proof Assume $J(f) \leq_{\mathrm{subT}} f$. Let $\mathcal{O}$ be Kleene's system of ordinal notations [3], and let $a \in \mathcal{O}$. We use $\varnothing^{(a)}$ to denote the iterated Turing jump of $\varnothing$ along the notation a. If $a = 2^b$ and $\varnothing^{(b)} \leq_{\mathrm{subT}} f$ via p_b, then by monotonicity and conservativity, we have $\varnothing^{(a)} \equiv_{\mathrm{subT}} J(\varnothing^{(b)}) \leq_{\mathrm{subT}} J(f) \leq_{\mathrm{subT}} f$ via some index p_a. By uniformity, $p_b \mapsto p_a$ is computable. If $a = 3 \cdot 5^e$ and $\varnothing^{(\varphi_e(n))} \leq_{\mathrm{subT}} f$ via $p_{\varphi_e(n)}$ for any $n \in \omega$, then whenever the sequence $(p_{\varphi_e(n)})_{n\in\omega}$ is computable, we have $\varnothing^{(a)} = \bigoplus_{n\in\omega} \varnothing^{(\varphi_e(n))} \leq_{\mathrm{subT}} f$ via some index p_a, where $(p_{\varphi_e(n)})_{n\in\omega} \mapsto p_a$ is computable. Hence, by effective transfinite induction, we get $\varnothing^{(a)} \leq_{\mathrm{subT}} f$ for any $a \in \mathcal{O}$. ∎

Unfortunately, K does not satisfy strictness in general. In other words, K has a fixed point.

Theorem 3.9 *There exists a partial function f such that $K(f) \equiv_{\mathrm{subT}} f$.*

Proof Let θ^Σ_e and θ^Π_e be the eth Σ^1_1 sentence and the eth Π^1_1 sentence, respectively. Define ψ as follows:

$$\psi(a,b) = \begin{cases} 1 & \text{if both } \theta^\Sigma_a \text{ and } \theta^\Pi_b \text{ are true} \\ 0 & \text{if both } \theta^\Sigma_a \text{ and } \theta^\Pi_b \text{ are false} \\ \uparrow & \text{otherwise.} \end{cases}$$

We claim $K(\psi) \leq_{\mathrm{subT}} \psi$. To see this, we simulate the computation of $\Phi_e[\psi](e)$ for given $e \in \mathrm{dom}(K(\psi))$. The history of the computation of $\Phi_e[\psi](e)$ of length s is a sequence $(i_0, q_0, a_0, i_1, q_1, a_1, \ldots, i_s, q_s)$ such that, for any $k < s$, $i_k = 0$, and $\Phi_e(e, a_0, \ldots, a_k)\downarrow = \langle i_{k+1}, q_{k+1}\rangle$, and if $q_k = \langle u, v\rangle$, then

(3.1) either (θ^Σ_u is true and $a_k = 1$) or (θ^Π_v is false and $a_k = 0$).

This gives a Σ^1_1-description of being a history of some computation. If $e \in \mathrm{dom}(K(\psi))$, then we must have $q_k = \langle u, v\rangle \in \mathrm{dom}(\psi)$, which means that the truth values of θ^Σ_u and θ^Π_v are the same. Thus, the condition (3.1) can be changed to the following:

(3.2) either (θ^Π_v is true and $a_k = 1$) or (θ^Σ_u is false and $a_k = 0$).

This gives a Π^1_1-description of being a history of some computation. Now, $\Phi_e[\psi](e)$ halts if and only if there exists a sequence which is the history of $\Phi_e[\psi](e)$ of some length s such that $i_s = 1$. This is described just by adding existential quantifiers on natural numbers, so this also has Σ^1_1-description $\theta^\Sigma_{c(e)}$ and Π^1_1-description $\theta^\Pi_{d(e)}$. Therefore, for $e \in \mathrm{dom}(K(\psi))$, whether $\Phi_e[\psi](e)$ halts or not can be decided by looking at the value of $\psi(c(e), d(e))$. This verifies the claim. ∎

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Note that such a fixed point f cannot be a partial hyperarithmetical function since, by Propositions 3.7 and 3.8, our operator K is strict on partial hyperarithmetical functions.

Question 1 *Does there exist a uniformly inflationary, uniformly monotone, uniformly conservative, strict operator on the subTuring degrees?*

4 Degree-theoretic properties

4.1 Density

Many degree structures, including the Turing degrees, are not globally dense, even though they may have some local substructures which are dense. Interestingly, in contrast to those degree structures, the structure of the subTuring degrees is globally dense.

Theorem 4.1 *The subTuring degrees are dense; that is, if $g <_{\mathrm{subT}} f$, then there is h such that $g <_{\mathrm{subT}} h <_{\mathrm{subT}} f$.*

In order to prove this, let us introduce a very useful "anti-cupping" lemma that will be the source of many results. In order to describe a statement, we need the following notion: For a partial function $f :\subseteq \omega \to \omega$, a set $A \subseteq \omega$ is *f-c.e.* if either it is empty or there exists a total function $g \leq_{\mathrm{subT}} f$ such that $A = \{g(n) : n \in \omega\}$. A set $A \subseteq \omega$ is *f-immune* if A has no f-c.e. subset.

Lemma 4.2 *For $\alpha, \beta : \omega \to \omega$ and $f :\subseteq \omega \to \omega$, let $A \subseteq \omega$ be an $(\alpha \oplus \beta \oplus f)$-immune set. For any $g :\subseteq \omega \to \omega$, if $\mathrm{dom}(g)$ is α-c.e., and $g \leq_{\mathrm{subT}} (f \upharpoonright A) \oplus \beta$, then $g \leq_{\mathrm{subT}} \beta$.*

Proof Assume that $g \leq_{\mathrm{subT}} (f \upharpoonright A) \oplus \beta$ via a partial computable function Φ. By monotonicity, we have $g \subseteq \Phi[(f \upharpoonright A) \oplus \beta] \subseteq \Phi[f \oplus \beta]$, so there is no difference between the computation of $\Phi[(f \upharpoonright A) \oplus \beta]$ and $\Phi[f \oplus \beta]$ for input $n \in \mathrm{dom}(g)$. Now, look at the queries $q_0, q_1, \dots$ made during the computation of $\Phi[f \oplus \beta](n)$, take out the even queries $2p_0, 2p_1, \dots$ (i.e., the queries asking to f), and put $Q_n = \{p_0, p_1, \dots\}$, which is the set of queries asking to f. If $\Phi[f \oplus \beta](n) \downarrow$, the set Q_n is clearly $(f \oplus \beta)$-c.e. uniformly in n. Furthermore, since $g \subseteq \Phi[(f \upharpoonright A) \oplus \beta]$, the queries to f that we make for input $n \in \mathrm{dom}(g)$ must be contained in $\mathrm{dom}(f \upharpoonright A) = A$; that is, if $n \in \mathrm{dom}(g)$, then $Q_n \subseteq A$. Consider $Q = \bigcup\{Q_n : n \in \mathrm{dom}(g)\}$, which is an $(\alpha \oplus \beta \oplus f)$-c.e. subset of A. By immunity, Q is finite. Since this is all the queries to f during the computation of g by Φ, we obtain $g \subseteq \Phi[(f \upharpoonright Q) \oplus \beta]$. However, since $f \upharpoonright Q$ is a finite function, it is computable, so we get $g \leq_{\mathrm{subT}} \beta$ as desired. ∎

Lemma 4.2 suggests that all nonzero degrees have properties close to the so-called strong anticupping property. Here, for an upper semilattice L, we say that an element $a \in L$ has the *strong anticupping property* if there exists $b \in L$ such that $a \leq b \vee c$ implies $a \leq c$ for any $c \in L$. Note that A depends on β in Lemma 4.2, so it does not show the strong anticupping property. However, it is powerful enough to help prove various results discussed below.

A partial function $f :\subseteq \omega \to \omega$ is *quasiminimal* if f is non-computable (in the sense that it cannot be extended to a partial computable function), but any total function $g \leq_{\mathrm{subT}} f$ is computable.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Proposition 4.3 *Every nonzero subTuring degree bounds a quasiminimal subTuring degree.*

Proof Let $f :\subseteq \omega \to \omega$ be a non-computable function. By Lemma 4.2, if A is f-immune then every total function $g \leq_{\mathrm{subT}} f \upharpoonright A$ is computable. We also have $f \upharpoonright A \leq_{\mathrm{subT}} f$. Therefore, it suffices to construct an f-immune set A such that $f \upharpoonright A$ is not computable.

At stage e, perform the following actions with auxiliary parameters u_e and r_e:

(1) Assume that $A \upharpoonright r_e$ has already been determined. Search for $n > r_e$ such that either $\varphi_e(n) \uparrow$ or $\varphi_e(n) \neq f(n)$. Note that such an n always exists; otherwise, $\varphi_e(n) \downarrow = f(n)$ for any $n > r_e$, which implies that f is computable, a contradiction. Choose such an n, declare $n \in A$, and put $u_e = n + 1$.

(2) Search for $n > u_e$ such that $n \in W_e^f$, where W_e^f is the eth c.e. set relative to f. If W_e^f is infinite, such an n exists. For such an n, declare $n \notin A$, and put $r_{e+1} = n + 1$. If there is no such n, put $r_{e+1} = u_e$.

By the action (1), for any e, we have $\varphi_e(n) \neq f(n)$ for some $n \in A$, so $f \upharpoonright A$ is non-computable. By the action (2), the final obtained A is f-immune. Thus, by the argument above, $f \upharpoonright A$ is quasiminimal. ∎

Now, let us show the density of the subTuring degrees.

Proof Let $f, g :\subseteq \omega \to \omega$ be such that $g <_{\mathrm{subT}} f$. Assume that the domain of g is α-c.e. As in the proof of Proposition 4.3, one can easily construct an $(\alpha \oplus f)$-immune set A such that $f \upharpoonright A \not\leq_{\mathrm{subT}} g$ (by using the assumption $f \not\leq_{\mathrm{subT}} g$). Then, we have $f \not\leq_{\mathrm{subT}} (f \upharpoonright A) \oplus g$; otherwise, by Lemma 4.2, $f \leq_{\mathrm{subT}} (f \upharpoonright A) \oplus g$ implies $f \leq_{\mathrm{subT}} g$, a contradiction. Consequently, we get $g <_{\mathrm{subT}} (f \upharpoonright A) \oplus g <_{\mathrm{subT}} f$. ∎

The above proof actually shows the existence of a strong quasiminimal cover in any nonempty interval of subTuring degrees. Here, we say that $\mathbf{a}$ is a *strong quasiminimal cover* of $\mathbf{b}$ if $\mathbf{b} < \mathbf{a}$ and, for any total degree $\mathbf{c}$, $\mathbf{c} \leq \mathbf{a}$ implies $\mathbf{c} \leq \mathbf{b}$. This is because if $h \leq_{\mathrm{subT}} (f \upharpoonright A) \oplus g$ is total, then $h \leq_{\mathrm{subT}} g$; hence, $(f \upharpoonright A) \oplus g$ is a strong quasiminimal cover of g.

Many degree structures, including the Turing degrees, are known to be locally countable, i.e., any given degree can bound only countably many different degrees. In contrast, the structure of subTuring degrees is not locally countable. In fact, any nonzero subTuring degree can bound continuum many different degrees.

Theorem 4.4 *Any nonempty interval of subTuring degrees contains an antichain of continuum many different degrees.*

In other words, if $g <_{\mathrm{subT}} f$, then there exists $\{h_\alpha\}_{\alpha < 2^{\aleph_0}}$ such that $g <_{\mathrm{subT}} h_\alpha <_{\mathrm{subT}} f$ and $h_\alpha \not\leq_{\mathrm{subT}} h_\beta$ whenever $\alpha \neq \beta$.

Proof The proof is almost the same as the proof of Proposition 4.3. Let $f, g :\subseteq \omega \to \omega$ be such that $g <_{\mathrm{subT}} f$, and assume that the domain of g is α-c.e. We construct a family $\{A_x\}_{x \in 2^\omega}$ of continuum many subsets of ω such that each A_x is $(\alpha \oplus A_y \oplus f)$-immune for any $y \neq x$. Moreover, we ensure that $f \upharpoonright A_x \not\leq_{\mathrm{subT}} g$ for any $x \in 2^\omega$. Assuming that we could construct such a family, consider $h_x = (f \upharpoonright A_x) \oplus g$ for any $x \in 2^\omega$. If $x \neq y$, by Lemma 4.2, $h_y \leq_{\mathrm{subT}} h_x$ would imply $f \upharpoonright A_y \leq_{\mathrm{subT}} g$, which is impossible

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

by our assumption on $\{A_x\}_{x\in 2^\omega}$. Hence, $\{h_x\}_{x\in 2^\omega}$ gives an antichain of continuum many subTuring degrees. Moreover, as in Theorem 4.1, one can also see that $g <_{\mathrm{subT}} h_x <_{\mathrm{subT}} f$. Therefore, it suffices to construct such a family $\{A_x\}_{x\in 2^\omega}$. What we actually construct is $\{A_\sigma\}_{\sigma\in 2^{<\omega}}$, and then we define $A_x = \bigcup_{\sigma<x} A_\sigma$.

At stage e, perform the following actions with auxiliary parameters u_e and r_e:

(1) Assume that $A_\sigma \upharpoonright r_e$ has already been determined for each $\sigma \in 2^e$. Search for $n \in \mathrm{dom}(f)$ with $n > r_e$ such that either $\Phi_e[g](n)\uparrow$ or $\Phi_e[g](n) \neq f(n)$. Note that such an n always exists; otherwise, $\Phi_e[g](n)\downarrow = f(n)$ for any $n \in \mathrm{dom}(f)$ with $n > r_e$, which implies that $f \leq_{\mathrm{subT}} g$, a contradiction. Choose such an n, declare $n \in A_\sigma$ for any $\sigma \in 2^e$, and put $u_e = n+1$.

(2) There are 2^{e+1} substages in the strategy to guarantee relative immunity. At substage 0, put $u_e^0 = u_e$. At substage $t < 2^{e-1}$, we focus on the t-th string $\sigma^\frown i \in 2^{e+1}$. Search for a finite string ρ extending $A_\sigma \upharpoonright u_e^t$ and a number $n > u_e^t$ such that $n \in W_e^{\alpha\oplus\rho\oplus f}$. If such ρ and n exist, then declare that $A_{\sigma^\frown i}$ extends ρ, and protect $A_{\sigma^\frown i} \upharpoonright |\rho|+1$. Moreover, declare $n \notin A_\tau$ for any string $\tau \neq \sigma^\frown i$ of length $e+1$. This means that we put $u_e^{t+1} = \max\{|\rho|, n\}+1$. If there exist no such ρ and n, put $A_{\sigma^\frown i} = A_\sigma$ and $u_e^{t+1} = u_e$. If $t = 2^{e+1}-1$, then put $r_e = u_e^{t+1}$.

Finally, we define $A_x = \bigcup_{\sigma<x} A_\sigma$ for any $x \in 2^\omega$. By the action (1), we get $f \upharpoonright A_\sigma \notin \Phi_e[g]$ for any $\sigma \in 2^e$, so $f \upharpoonright A_x \notin \Phi_e[g]$ since $A_\sigma \subseteq A_x$ for any $x > \sigma$; hence, $f \upharpoonright A_x \not\leq_{\mathrm{subT}} g$ for any $x \in 2^\omega$. By the action (2), since A_x extends $A_{x\upharpoonright e+1} \upharpoonright u_e^t$ for a suitable t, if $W_e^{\alpha\oplus A_x\oplus f}$ is infinite, we always find ρ and $n > u_e^t$ such that $n \in W_e^{\alpha\oplus\rho\oplus f}$. If $x \upharpoonright e \neq y \upharpoonright e$, then the action (2) ensures $n \notin A_y$, so we obtain $W_e^{\alpha\oplus A_x\oplus f} \not\subseteq A_y$. If $x \neq y$, then there is k such that $x \upharpoonright k \neq y \upharpoonright k$, so given e, consider $d \geq k$ such that $W_d^B = W_e^B$ for any B (e.g., add meaningless lines to the program e to pad its size). By the above argument, we get $W_e^{\alpha\oplus A_x\oplus f} = W_d^{\alpha\oplus A_x\oplus f} \not\subseteq A_y$. Consequently, A_y is $(\alpha\oplus A_x\oplus f)$-immune. ∎

Note that h_x in Theorem 4.4 is a strong quasiminimal cover of g for any $x \in 2^\omega$.

4.2 Jump inversion

In Turing degree theory, the Friedberg jump inversion theorem states that for every total function $h \geq_T \varnothing'$, there is a total function f such that $f' \equiv_T h$. In enumeration degree theory, this has been improved to the quasiminimal jump inversion theorem. Although the proof is different from the usual method, one can also prove the (quasiminimal) jump inversion theorem for the subTuring degrees. Here, we adopt the jump operator K introduced in Section 3.3.

Theorem 4.5 *For any partial function $h \geq_{\mathrm{subT}} \varnothing'$, there exists a quasiminimal partial function f such that $K(f) \equiv_{\mathrm{subT}} h \equiv_{\mathrm{subT}} f \oplus \varnothing'$.*

Proof It suffices to show that, for any partial function h, there exists a quasiminimal f such that $K(f) \leq_{\mathrm{subT}} h \oplus \varnothing' \leq_{\mathrm{subT}} f \oplus \varnothing'$ since $\varnothing' \leq_{\mathrm{subT}} K(\varnothing) \leq_{\mathrm{subT}} K(f)$ by monotonicity and conservativity. Let us construct a $\varnothing'$-computable increasing sequence $(a_k)_{k\in\omega}$ of natural numbers. This is the coding location of h in f; that is, we define $f(a_k) = h(k)$ for each $k \in \omega$ and let f be undefined except on $\{a_k\}_{k\in\omega}$.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Assume that $(a_k)_{k<e}$ has already been defined. For each partial function σ defined only on $\{a_k\}_{k<e}$, ask $\varnothing'$ if the computation of $\Phi_e[\sigma](e)$ makes a query greater than a_{e-1}. If so, let q_σ be the first such query plus 1, or else let $q_\sigma = a_{e-1}+1$. Similarly, ask $\varnothing'$ if there exists n such that the computation of $\Phi_e[\sigma](n)$ makes a query greater than a_{e-1}. If so, let r_σ be the first such query plus 1, or else let $r_\sigma = a_{e-1}+1$. Then, let a_e be the maximum value of these q_σ and r_σ's.

This completes the construction. We claim that the computation of $\Phi_e[f](e)$ is determined by $\Phi_e[f\upharpoonright a_{e-1}+1](e)$. Otherwise, this computation creates a query to f greater than a_{e-1}. The computation is performed by $\Phi_e[f\upharpoonright a_{e-1}+1](e)$ until just before the first such query q is created. Note that the domain of $\sigma := f\upharpoonright a_{e-1}+1$ is $\{a_k\}_{k<e}$, so our construction ensures $a_{e-1}<q<q_\sigma\le a_e$, which implies $q\notin\mathrm{dom}(f)$. Therefore, $\Phi_e[f](e)$ does not halt.

Observe that $(a_k)_{k\in\omega}$ is $\varnothing'$-computable. Thus, it is clear from the definition of f that $h\le_{\mathrm{subT}} f\oplus\varnothing'$ and $f\le_{\mathrm{subT}} h\oplus\varnothing'$ hold. Indeed, the graph of f is computable in $h\oplus\varnothing'$. We show that $K(f)\le_{\mathrm{subT}} h\oplus\varnothing'$. To compute $K(f)(e)$, first use $h\oplus\varnothing'$ to recover $(a_k)_{k<e}$ and the graph of $f\upharpoonright a_{e-1}+1$. Ask $\varnothing'$ which of the following is true:

(1) $\Phi_e[f\upharpoonright a_{e-1}+1](e)$ makes a query $\le a_{e-1}$ that is different from any of $\{a_k\}_{k<e}$, or makes a query that is greater than a_{e-1}.
(2) $\Phi_e[f\upharpoonright a_{e-1}+1](e)$ halts.
(3) Otherwise. That is, $\Phi_e[f\upharpoonright a_{e-1}+1](e)$ does not halt even though it only makes queries in $\{a_k\}_{k<e}$.

Since $(a_k)_{k<e}$ and $f\upharpoonright a_{e-1}+1$ are finite, this decision can be performed with the unrelativized halting problem $\varnothing'$. By the above construction, if (1) holds, $K(f)(e)\uparrow$; if (2) holds, $K(f)(e)=1$; and if (3) holds, $K(f)(e)=0$. This shows $K(f)\le_{\mathrm{subT}} h\oplus\varnothing'$.

For quasiminimality, if $\Phi_e[f]$ is total, this computation can be simulated by $\Phi_e[f\upharpoonright a_{e-1}+1]$. Otherwise, $\Phi_e[f\upharpoonright a_{e-1}+1](n)$ for some n makes a query q larger than a_{e-1}, but by the same argument as in Claim above, for $\sigma := f\upharpoonright a_{e-1}+1$ is $\{a_k\}_{k<e}$, our construction ensures $a_{e-1}<q<r_\sigma\le a_e$, which implies $q\notin\mathrm{dom}(f)$. Thus, $\Phi_e[f]$ is computable since this computation uses only finite information on f. ∎

4.3 Minimal pair

Next, let us analyze the properties of subTuring meets. Turing degrees do not form a lattice, but there are several pairs of Turing degrees having a meet. The following results lead to the conclusion that this Turing meet and the subTuring meet are never coincident. In fact, every total degree is meet-irreducible within the total degrees.

Theorem 4.6 *For any total functions $f,g,h\colon\omega\to\omega$, if $f\cap g\le_{\mathrm{subT}} h$, then either $f\le_{\mathrm{subT}} h$ or $g\le_{\mathrm{subT}} h$ holds.*

Proof We construct Turing functionals Ψ and Δ fulfilling the following requirement:

$$R_e\colon(\forall f,g\in 2^\omega)\ \Psi^f\cap\Delta^g\subseteq\Phi_e^h\implies f\le_{\mathrm{subT}} h\text{ or }g\le_{\mathrm{subT}} h.$$

Here, $\Psi^f\cap\Delta^g$ is the intersection of Ψ^f and Δ^g when identifying a function with its graph. That is, if (and only if) both $\Psi^f(n)$ and $\Delta^g(n)$ converge to the same value, $(\Psi^f\cap\Delta^g)(n)$ halts at the value. Obviously, $\Psi^f\cap\Delta^g\le_{\mathrm{subT}} f\cap g$, so it is sufficient to

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

satisfy this requirement. Rewriting R_e a little more concretely according to the actual strategy, we will ensure the following:

$$R_e\colon (\forall f, g \in 2^\omega)\ [(\exists n)\ \Psi^f(n)\downarrow = \Delta^g(n)\downarrow \neq \Phi_e^h(n)] \text{ or } f \leq_{\mathrm{subT}} h \text{ or } g \leq_{\mathrm{subT}} h.$$

Strategy: Let us describe the R_e-strategy. The strategy must operate without knowing what f and g are and with all possibilities of $f, g \in 2^\omega$ in mind. This strategy consists of infinitely many substrategies. The (s,t)-th substrategy proceeds as follows:

(1) For any f with $f(s) = i$, put $\Psi^f(e,s,t,i,j) = 0$. Similarly, for any g with $g(t) = j$, put $\Delta^g(e,s,t,i,j) = 0$.

(2) Wait for $\Phi_e^h(e,s,t,i,j)\downarrow = 0$ for some $i, j < 2$.

If no such i, j exists, then proceed to the next stage $e+1$. In this case, $\Psi^f(e,s,t,f(s),g(t)) = \Delta^g(e,s,t,f(s),g(t)) = 0 \neq \Phi_e^h(e,s,t,f(s),g(t))$, so the requirement R_e is satisfied.

If such i and j exist, fix the first pair i, j found.

(3) For any f with $f(s) = 1-i$, put $\Psi^f(e,s,t,i,j) = 1$. Similarly, for any g with $g(t) = 1-i$, put $\Delta^g(e,s,t,i,j) = 1$.

Note that for any f, g with $\langle f(s), g(t)\rangle = \langle 1-i, 1-j\rangle$, then $\Psi^f(e,s,t,i,j) = \Delta^g(e,s,t,i,j) = 1 \neq 0 = \Phi_e^h(e,s,t,i,j)$, so the requirement R_e is satisfied.

(4) Wait for $\Phi_e^h(e,s,t,1-i,j)\downarrow = 0$. While waiting, invoke the $\langle s+1, t\rangle$-th substrategy.

If we wait forever, for any f, g with $\langle f(s), g(t)\rangle = \langle 1-i, j\rangle$, then by (1), $\Psi^f(e,s,t,1-i,j) = \Delta^g(e,s,t,1-i,j) = 0 \neq \Phi_e^h(e,s,t,1-i,j)$, so the requirement R_e is satisfied. Thus, together with (3), we see that R_e is satisfied for any value of $g(t)$ when $f(s) = 1-i$.

(5) Suppose that, after waiting for a while, $\Phi_e^h(e,s,t,1-i,j)\downarrow = 0$ occurs. While waiting, the R_e-strategy is still running, so suppose that the substrategy currently in operation is (s',t) (the (s,t)-strategy will be abandoned when t increases, so we may assume that the second coordinate remains at t).

Once we recognize that we reach (5), if $f(s) = i$, then put $\Psi^f(e,s,t,1-i,j) = 1$; similarly, if $g(s) = 1-j$, then put $\Delta^g(e,s,t,1-i,j) = 1$. Then stop the currently running substrategy and start a new substrategy $(s',t+1)$.

In this case, if $\langle f(s), g(s)\rangle = \langle i, 1-j\rangle$, then $\Psi^f(e,s,t,1-i,j) = \Delta^g(e,s,t,1-i,j) = 1 \neq 0 = \Phi_e^h(e,s,t,1-i,j)$, so the requirement R_e is satisfied. Together with (3), we see that R_e is satisfied for any value of $f(s)$ when $g(t) = 1-j$. However, since the fulfillment of the requirement in (4) with $\langle f(s), g(s)\rangle = \langle 1-i, j\rangle$ has been withdrawn, R_e is not satisfied when $g(t) = j$.

Verification: Let us check that the requirement R_e is satisfied by this construction.

Case 1: If we wait forever for (2) in either substrategy, the requirement R_e is clearly satisfied.

Hereafter, we write i and j chosen in the state (2) of the (s,t)-substrategy as $i_{s,t}$ and $j_{s,t}$, respectively. Note that $(s,t) \mapsto i_{s,t}, j_{s,t}$ is h-computable since the substrategy proceeds in an h-computable process.

Case 2: Consider the case where all substrategies pass through (3) but t converges to a finite value. In this case, for some s_0, the substrategies after (s_0,t) do not reach (5) but wait at (4) and continue to activate the next substrategy. As described in (4),

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

the requirement R_e is satisfied for any value of $g(t)$ when $f(s) = 1 - i_{s,t}$. That is, the requirement R_e is not yet satisfied only for $f(s) = i_{s,t}$ for any $s > s_0$. However, in this case, as $s \mapsto i_{s,t} = f(s)$ is h-computable, one derives $f \leq_{\mathrm{subT}} h$.

Case 3: Consider the case where t diverges to infinity. In this case, for each t, (5) occurs for $j = j_{s_t,t}$ defined by some substrategy (s_t, t). As described in (5), the requirement R_e is satisfied for any value of $f(s_t)$ when $g(t) = 1 - j_{s_t,t}$. That is, the requirement R_e is not yet satisfied only for $g(t) = j_{s_t,t}$ for any t. Note that one can h-computably recognize what $j_{s_t,t}$ is by waiting for the occurrence of (5), so $t \mapsto j_{s_t,t}$ is h-computable. Thus, in this case, $g \leq_{\mathrm{subT}} h$ is derived and the requirement R_e is satisfied. ■

Corollary 4.7 *The meet of two incompatable total subTuring degrees is nontotal.*

This leads to the observation that, although the existence of a minimal pair is well known for Turing degrees, it does not form a minimal pair in the subTuring degrees. Here, if a lattice L has a bottom element 0, a pair (a, b) of elements in L is called a *minimal pair* if $a, b > 0$ and $a \wedge b = 0$.

Corollary 4.8 *There exists no minimal pair of total subTuring degrees.*

4.4 Distributivity

A lattice is distributive if $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. We now give a proof of non-distributivity of the subTuring degrees.

Theorem 4.9 *The subTuring lattice is not distributive.*

Proof We construct partial functions $f :\subseteq \omega \times 2 \times 2 \to \omega$ and $g, h :\subseteq \omega \to 2$ such that $(f \oplus g) \cap (f \oplus h) \not\leq_{\mathrm{subT}} f \oplus (g \cap h)$. It suffices to fulfill the following requirement:

$$R_e : (f \oplus g) \cap (f \oplus h) \not\subseteq \Phi_e[f \oplus (g \cap h)].$$

Given e, we need to find c, d, x such that $((f \oplus g) \cap (f \oplus h))(c, d, x) \downarrow \neq \Phi_e[f \oplus (g \cap h)](c, d, x)$. Given n, consider the following Turing functionals Ψ_n and Γ_n:

$$\Psi_n[f \oplus g](x) = f(n, g(n), 0) \qquad \Gamma_n[f \oplus h](x) = f(n, h(n), 1).$$

If we write indices of Ψ_n and Γ_n as c_n and d_n, respectively, we actually guarantee the following requirement:

$$R_e : \exists n, x\, [\Psi_n[f \oplus g](x) \downarrow = \Gamma_n[f \oplus h](x) \downarrow \neq \Phi_e[f \oplus (g \cap h)](c_n, d_n, x)].$$

By the definition of the meet $\cap$, it is easy to see that this modified requirement implies the first proposed requirement.

Strategy: At stage e, operate the strategy to satisfy the requirement R_e. Inductively, assume that we have constructed finite functions f_e, g_e, h_e and an auxiliary parameter n_e. We also assume that, for any $t < n_e$, $f_e(t, i, j)$, $g_e(t)$ and $h_e(t)$ have already been determined, and for any $t \geq n_e$, $f_e(t, i, j)$, $g_e(t)$ and $h_e(t)$ are undefined. We refer to a triple of the form $\langle t, i, j \rangle$ as a level t input. Now, let c and d be indices of $\Psi := \Psi_{n_e}$ and $\Gamma := \Gamma_{n_e}$, respectively.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

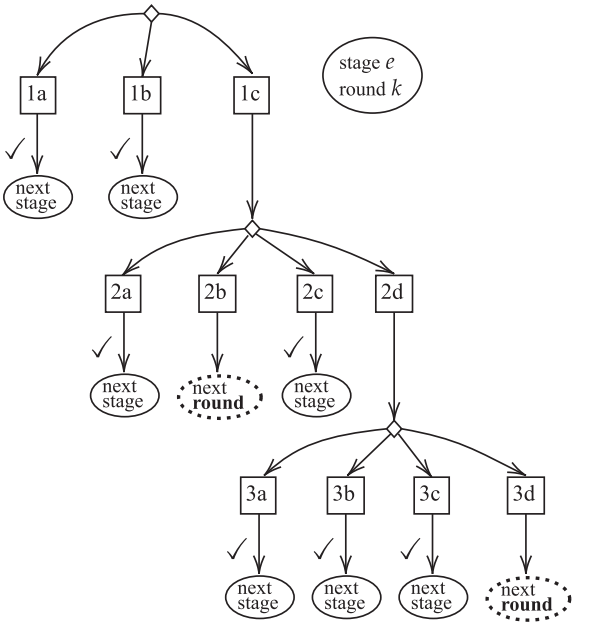


Figure 1: Overview of the R_e-strategy.

This strategy consists of infinitely many rounds. See Figure 1 for the overview of the R_e-strategy. First, regarding the computation of $\Phi_e[f \oplus (g \cap h)](c,d,x)$, we have the following three possibilities:

(1a) $\Phi_e[f \oplus (g \cap h)](c,d,x)$ does not make a query, or makes a query only to f, which is of level $< n_e$.
(1b) $\Phi_e[f \oplus (g \cap h)](c,d,x)$ makes a level $\geq n_e$ query to f.
(1c) $\Phi_e[f \oplus (g \cap h)](c,d,x)$ makes a query to $g \cap h$.

Here, in the $(k+1)$-th round, we ask about the subsequent queries after the query taken up in the k^{th} round. As we will see later, no action is taken when advancing a round, always leaving room for the same action as in the first round. Therefore, follow the discussion as if we were in the first round.

Case (1a): Put $g_{e+1}(n_e) = h_{e+1}(n_e) = 0$, and set the level n_e values of f_{e+1} appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$, whose value is different from $\Phi_e[f_e \oplus (g_e \cap h_e)](c,d,x)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e+1$.

Verification: Since $\Phi_e[f_e \oplus (g_e \cap h_e)](c,d,x)$ only makes level $< n_e$ queries to f, the behavior of this computation coincides with $\Phi_e[f \oplus (g \cap h)](c,d,x)$. By the action of the strategy, we have $\Psi[f \oplus g](x) = f(e, g(n_e), 0) = f(e, h(n_e), 1) = \Gamma[f \oplus h](x) \downarrow$, whose value is different from $\Phi_e[f \oplus (g \cap h)](c,d,x)$. Thus, the requirement R_e is satisfied.

Case (1b): There is a query to f of the form (m,i,j) for some $m \geq n_e$. Declare $f_{e+1}(m,i,j) \uparrow$. Put $g_{e+1}(n_e) = h_{e+1}(n_e) = 1 - i$. Since $(m,i,j) \neq (n_e, g_{e+1}(n_e), 0)$ and $(m,i,j) \neq (n_e, h_{e+1}(n_e), 1)$, one can set the level n_e values of f_{e+1} so that

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

$f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = m + 1$, and proceed to stage $e + 1$.

Verification: $\Phi_s[f \oplus (g \cap h)](d, e, x)$ makes a query $(m, i, j) \notin \mathrm{dom}(f)$, so the computation does not halt. We also have $\Psi[f \oplus g](x) = f(n_e, g(n_e), 0) = f(n_e, h(n_e), 1) = \Gamma[f \oplus h](x) \downarrow$. Thus, the requirement R_e is satisfied.

Case (1c): The computation makes a query (a, b, y) to $g \cap h$. By the definition of the meet $g \cap h$, the oracle's response to this query is the common value of $\Phi_a[g](y)$ and $\Phi_b[h](y)$ (if defined). Let us now go into the computation processes of $\Phi_a[g](y)$ and $\Phi_b[h](y)$. We have the following possibilities:

(2a) Either $\Phi_a[g](y)$ or $\Phi_b[h](y)$ does not halt, but makes only small queries less than n_e.
(2b) Either $\Phi_a[g](y)$ or $\Phi_b[h](y)$ halts after making only small queries less than n_e.
(2c) Either $\Phi_a[g](y)$ or $\Phi_b[h](y)$ makes a query larger than n_e.
(2d) None of the above; that is, both $\Phi_a[g](y)$ and $\Phi_b[h](y)$ make a query n_e.

Case (2a): Put $g_{e+1}(n_e) = h_{e+1}(n_e) = 0$, and set the values appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e + 1$.

Verification: Recall that we are in Case (1c), where $\Phi_e[f \oplus (g \cap h)](c, d, x)$ makes a query (a, b, y) to $g \cap h$, and we then enter Case (2a), where $\Phi_a[g_e](c)$ or $\Phi_b[h_e](c)$ is undefined, so $(g \cap h)(a, b, y)$ is also undefined. This means that $\Phi_e[f \oplus (g \cap h)](c, d, x)$ is accessing outside the domain of the oracle, so the computation does not halt. By our action, we have $\Psi[f \oplus g](x) = f(n_e, g(n_e), 0) = f(n_e, h(n_e), 1) = \Gamma[f \oplus h](x) \downarrow$, so the requirement R_e is satisfied.

Case (2b): In this case, the common value of $\Phi_a[g](c)$ and $\Phi_b[h](c)$ is determined by the current finite information g_e or h_e. Then, proceed to the next round $k + 1$; that is, resume the computation of $\Phi_e[f \oplus (g \cap h)](c, d, x)$, and check the conditional branch (1) again for the subsequent queries of the query (a, b, y) taken up this time.

Case (2c): If Φ_a makes a query $m > n_e$ to g, then declare $g(m) \uparrow$. Do the same for Φ_b and h. Put $g_{e+1}(n_e) = h_{e+1}(n_e) = 0$, and set the values appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e + 1$.

Verification: Recall that we are in Case (1c), where $\Phi_e[f \oplus (g \cap h)](c, d, x)$ makes a query (a, b, y) to $g \cap h$, and we then enter Case (2c), where Φ_a makes a query $m > n_e$ to g. Then, our action ensures $m \notin \mathrm{dom}(g)$, so we have $\Phi_a[g](c) \uparrow$. Do the same for the symmetric case. If $\Phi_a[g_e](c)$ or $\Phi_b[h_e](c)$ is undefined, then $(g \cap h)(a, b, y)$ is also undefined, which means that $\Phi_e[f \oplus (g \cap h)](c, d, x)$ is accessing outside the domain of the oracle, so the computation does not halt. By our action, we have $\Psi[f \oplus g](x) = f(n_e, g(n_e), 0) = f(n_e, h(n_e), 1) = \Gamma[f \oplus h](x) \downarrow$, so the requirement R_e is satisfied.

Case (2d): In this case, both $\Phi_a[g](y)$ and $\Phi_b[h](y)$ make only queries less than or equal to n_e. Therefore, depending on the values of $g(n_e)$ and $h(n_e)$, at most four different computations appear. Looking at these four computations, we have the following possibilities:

(3a) Either $\Phi_a[g](y)$ or $\Phi_b[h](y)$ may not halt, depending on the value of $g(n_e)$ or $h(n_e)$.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

(3b) Either $\Phi_a[g](y)$ or $\Phi_b[h](y)$ may make a query greater than n_e, depending on the value of $g(n_e)$ or $h(n_e)$.
(3c) $\Phi_a[g](y)$ or/and $\Phi_b[h](y)$ have the possibility of halting computations of different outputs, depending on the value of $g(n_e)$ or/and $h(n_e)$.
(3d) Otherwise, that is, no matter what values $g(n_e)$ and $h(n_e)$ are set to, $\Phi_a[g](y)$ and $\Phi_b[h](y)$ both halts and return the same output, making only queries less than or equal to n_e.

Case (3a): For instance, if setting $g_{e+1}(n_e) = i$ forces $\Phi_a[g_{e+1}](y)$ not to halt, then set $h_{e+1}(n_e)$ to any value. Do the same for the symmetric case. Set the values appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e + 1$.

Case (3b): For instance, if setting $g_{e+1}(n_e) = i$ forces $\Phi_a[g_{e+1}](y)$ to make a query $m > n_e$, then declare $g_{e+1}(m) \uparrow$, and set $h_{e+1}(n_s)$ to any value. Do the same for the symmetric case. Set the values appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e + 1$.

Case (3c): If setting $g_{e+1}(n_e) = i$ and $h_{e+1}(n_e) = j$ force the condition $\Phi_a[g](y) \neq \Phi_b[h](y)$, then set the values appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e + 1$.

Otherwise, put $g^i_{e+1}(n_e) = i$ for each $i < 2$, and assume $\Phi_a[g^0_{e+1}](y) \neq \Phi_a[g^1_{e+1}](y)$. Then, put $h_{e+1}(n_e) = 0$. Assuming that (3a) does not hold, $\Phi_b[h_{e+1}](y)$ must halt, so $\Phi_b[h_{e+1}](y) \neq \Phi_a[g^i_{e+1}](y)$ for some $i < 2$. Then, put $g_{e+1} = g^i_{e+1}$ for such i. Do the same for the symmetric case. Set the values appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e + 1$.

Verification: The requirement R_e is satisfied in Case (3a) for the same reason as (2a), and Case (3b) for the same reason as (2c). The same reasoning can be applied to Case (3c), since we ensure $\Phi_a[g_{e+1}](y) \neq \Phi_b[h_{e+1}](y)$, which implies $(a, b, y) \notin \mathrm{dom}(g \cap h)$.

Case (3d): In this case, the common value of $\Phi_a[g](y)$ and $\Phi_b[h](y)$ is determined by the current finite information g_e or h_e (since the computations ask only for values of $g \upharpoonright n_e + 1$ and $h \upharpoonright n_e + 1$ by $\neg$(2c), and the results of these computations do not depend on the values of $g(n_e)$ and $h(n_e)$). Then, proceed to the next round $k + 1$; that is, resume the computation of $\Phi_e[f \oplus (g \cap h)](c, d, x)$, and check the conditional branch (1) again for the subsequent queries of the query (a, b, y) taken up this time.

Verification: The next round proceeds only if the strategy reaches (2b) or (3d). In this case, the response to the query (a, b, y) made at (1c) can be computed only from the information of $g_e = g \upharpoonright n_e$ and $h_e = h \upharpoonright n_e$. Moreover, the strategy does not take any action at (2b) or (3c), so nothing interferes with the next round of operation. Also, if the strategy arrives at a state other than (2b) or (3d) at some round, then the requirement R_e is satisfied there.

Infinite rounds: If the strategy keeps arriving only at (2b) or (3d), it means that the round keeps going infinitely, and this also means, in particular that (1c) is passed through infinitely many times. Then, the computation of $\Phi_e[f \oplus (g \cap h)](c, d, x)$ continues to make queries to $g \cap h$ infinitely many times, so $\Phi_e[f \oplus (g \cap h)](c, d, x)$ does not halt. In this case, put $g_{e+1}(n_e) = h_{e+1}(n_e) = 0$, and set the values

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

appropriately so that $f_{e+1}(n_e, g_{e+1}(n_e), 0) \downarrow = f_{e+1}(n_e, h_{e+1}(n_e), 1)$. Put $n_{e+1} = n_e + 1$, and proceed to stage $e+1$. Then, the requirement R_e is satisfied for the same reason as (1a). ■

A lattice L is *modular* if, for any $a, b, c \in L$, $a \leq b$ implies $a \vee (b \wedge c) = b \wedge (a \vee c)$. Obviously, a distributive lattice is always modular. By making a small modification to the proof of Theorem 4.9, it is actually possible to prove that the subTuring lattice is not modular.

Theorem 4.10 *The subTuring lattice is not modular.*

Proof We construct partial functions $f :\subseteq \omega \times 2 \times 2 \to \omega$ and $g, h :\subseteq \omega \to 2$ such that $(f \oplus g) \cap (f \oplus h) \not\leq_{\mathrm{subT}} f \oplus (g \cap h)$. It suffices to fulfill the following requirement:

$$R_e : (f \oplus g) \cap (f \oplus h) \not\subseteq \Phi_e[f \oplus (g \cap (f \oplus h))].$$

Given e, we need to find c, d, x such that $((f \oplus g) \cap (f \oplus h))(c, d, x) \downarrow \neq \Phi_e[f \oplus (g \cap h)](c, d, x)$. For the construction, the description becomes a little more complicated, but the idea of the proof is exactly the same as Theorem 4.9. For example, item (1c) is modified as follows:

(1c) $\Phi_e(f \oplus (g \cap (f \oplus h)))$ makes a query to $g \cap (f \oplus h)$.

If this happens, assume that the computation makes a query (a, b, y) to $g \cap (f \oplus h)$. Then, go into the computation processes of $\Phi_a[g](y)$ and $\Phi_b[f \oplus h](y)$. The discussion that follows is exactly the same as Theorem 4.9, but for example, item (2c) is modified as follows:

(2c) The computation of $\Phi_a[g](y)$ or $\Phi_b[f \oplus h](y)$ makes either a query to g or h which is larger than n_e, or a query to f which is of level $\geq n_e$.

Note that a query to f may be at level n_e. In this case, the requirement is satisfied by exactly the same action as in Case (1b). If there is a query to g or h that is greater than n_e, the situation is the same as Case (2c) in Theorem 4.9.

In Case (2d), both $\Phi_a[g](y)$ and $\Phi_b[f \oplus h](y)$ can only make queries to g and h which are less than or equal to n_e and queries to f of level less than n_e. Therefore, the situation is exactly the same as Case (2d) in Theorem 4.9, so the exact same argument applies. ■

While the structure is not a distributive lattice, we can nonetheless find incomparable subTuring degrees that satisfy the distributive law.

Proposition 4.11 *There are partial functions $f_0, f_1, f_2 :\subseteq \omega \to \omega$ such that their subTuring degrees are pairwise incomparable, and $f_0 \oplus (f_1 \cap f_2) \equiv_{\mathrm{subT}} (f_0 \oplus f_1) \cap (f_0 \oplus f_2)$.*

Similarly, there are partial functions $e_0, e_1, e_2 :\subseteq \omega \to \omega$ such that their subTuring degrees are pairwise incomparable, and $e_0 \cap (e_1 \oplus e_2) = (e_0 \cap e_1) \oplus (e_0 \cap e_2)$.

Proof We claim that given any subTuring incomparable partial functions g, h such that $g \cap h \not\equiv_{\mathrm{subT}} \varnothing$, we can find sets A and B such that:

(1) $g \upharpoonright A \not\leq_{\mathrm{subT}} h$ and $h \upharpoonright B \not\leq_{\mathrm{subT}} g$.
(2) $g \not\leq_{\mathrm{subT}} (g \upharpoonright A) \oplus (h \upharpoonright B)$ and $h \not\leq_{\mathrm{subT}} (g \upharpoonright A) \oplus (h \upharpoonright B)$.
(3) $g \cap h \not\leq_{\mathrm{subT}} g \upharpoonright A$ and $g \cap h \not\leq_{\mathrm{subT}} h \upharpoonright B$.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

If these conditions are satisfied, we can take $f_0 = g \cap h$, $f_1 = g \upharpoonright A$ and $f_2 = h \upharpoonright B$ and take $e_0 = f_1 \oplus f_2$, $e_1 = g$ and $e_2 = h$. The condition (1) implies $f_1, f_2 \not\leq_{\mathrm{subT}} f_0$, and the condition (3) implies that $f_0 \not\leq_{\mathrm{subT}} f_1, f_2$ and f_1 is incomparable with f_2. Hence, f_0, f_1, f_2 are pairwise subTuring incomparable. The condition (2) implies $e_1, e_2 \not\leq_{\mathrm{subT}} e_0$, and the condition (1) implies that $e_0 \not\leq_{\mathrm{subT}} e_1, e_2$ and e_1 is incomparable with e_2. Hence, e_0, e_1, e_2 are pairwise subTuring incomparable.

Now, we have $f_1 \cap f_2 = (g \upharpoonright A) \cap (h \upharpoonright B) \leq_{\mathrm{subT}} g \cap h = f_0$, so $f_0 \oplus (f_1 \cap f_2) \equiv_{\mathrm{subT}} f_0$. Moreover, $f_0 \leq_{\mathrm{subT}} f_0 \oplus f_1 = (g \cap h) \oplus (g \upharpoonright A) \leq_{\mathrm{subT}} g$, and similarly $f_0 \leq_{\mathrm{subT}} f_0 \oplus f_2 \leq_{\mathrm{subT}} h$. Hence, $f_0 \leq_{\mathrm{subT}} (f_0 \oplus f_1) \cap (f_0 \oplus f_2) \leq_{\mathrm{subT}} g \cap h = f_0$. This shows $f_0 \oplus (f_1 \cap f_2) \equiv_{\mathrm{subT}} f_0 \equiv_{\mathrm{subT}} (f_0 \oplus f_1) \cap (f_0 \oplus f_2)$.

Next, we have $e_0 = (g \upharpoonright A) \oplus (h \upharpoonright B) \leq_{\mathrm{subT}} g \oplus h = e_1 \oplus e_2$, so $e_0 \cap (e_1 \oplus e_2) \equiv_{\mathrm{subT}} e_0$. Moreover, $g \upharpoonright A \leq_{\mathrm{subT}} ((g \upharpoonright A) \oplus (h \upharpoonright B)) \cap g = e_0 \cap e_1 \leq_{\mathrm{subT}} e_0$, and similarly, $h \upharpoonright B \leq_{\mathrm{subT}} e_0 \cap e_2 \leq_{\mathrm{subT}} e_0$. Hence, $e_0 = (g \upharpoonright A) \oplus (h \upharpoonright B) \leq_{\mathrm{subT}} (e_0 \cap e_1) \oplus (e_0 \cap e_2) \leq_{\mathrm{subT}} e_0$. This shows $e_0 \cap (e_1 \oplus e_2) \equiv_{\mathrm{subT}} e_0 \equiv_{\mathrm{subT}} (e_0 \cap e_1) \oplus (e_0 \cap e_2)$.

In order to show the claim, we need to construct $A \upharpoonright s$ and $B \upharpoonright s$ for each s; assume that these have been decided for s, and our next step is to satisfy a requirement of type (1), (2), or (3) above.

To satisfy (1), we pick $n > s$ such that $n \in \mathrm{dom}(g)$ and $g(n) \neq \Phi[h](n)$, where Φ is the next Turing functional we are diagonalizing against. This n must exist lest $g \leq_{\mathrm{subT}} h$. We then set $A \upharpoonright n+1 = (A \upharpoonright s) \cup \{n\}$, which will ensure that $g \upharpoonright A \notin \Phi[h]$. We act similarly to make $h \upharpoonright B \notin \Phi[g]$.

To satisfy (2), we check if there is some $x \in \mathrm{dom}(g)$ such that the computation $\Phi[(g \upharpoonright (A \upharpoonright s)) \oplus (h \upharpoonright (B \upharpoonright s))](x)$ makes a query to the oracle $g(n)$ or $h(n)$ for some $n > s$. If no, then $g \subseteq \Phi[(g \upharpoonright A) \oplus (h \upharpoonright B)]$ implies that $g \subseteq \varphi$ for some partial computable function simulating Φ with the given finite oracle, which is impossible. If the answer is yes, witnessed by some least $n > s$, we take $A \upharpoonright n+1 = (A \upharpoonright s)$ and $B \upharpoonright n+1 = (B \upharpoonright s)$, in particular setting $n \notin A \cup B$, and therefore ensuring $g \notin \Phi[(g \upharpoonright A) \oplus (h \upharpoonright B)]$. We do similarly to make $h \notin \Phi[(g \upharpoonright A) \oplus (h \upharpoonright B)]$.

To satisfy (3), we check if there is some $x \in \mathrm{dom}(g \cap h)$ such that the computation $\Phi[g \upharpoonright (A \upharpoonright s)](x)$ queries $g(n)$ for some $n > s$. If not, then just like in (2) above, we have that $g \cap h \notin \Phi[g \upharpoonright A]$, unless $g \cap h \equiv_{\mathrm{subT}} \varnothing$. If yes, then we just keep $n \notin A$ by taking $A \upharpoonright n+1 = (A \upharpoonright s)$ which will ensure that $g \cap h \notin \Phi[g \upharpoonright A]$. We proceed similarly to make $g \cap h \notin \Phi[h \upharpoonright B]$. In fact, note that we can replace $g \cap h$ by any function which is $\not\equiv_{\mathrm{subT}} \varnothing$. ■

4.5 Irreducibility

For a lattice L, an element $a \in L$ is *join-irreducible* if, for any $b, c \in L$, $a = b \vee c$ implies either $a = b$ or $a = c$.

Theorem 4.12 *There exists a nonzero join-irreducible subTuring degree.*

Proof We construct a noncomputable partial function f such that for any $g, h \leq_{\mathrm{subT}} f$ if $f \leq_{\mathrm{subT}} g \oplus h$ then either $f \leq_{\mathrm{subT}} g$ or $f \leq_{\mathrm{subT}} h$ holds. It suffices to fulfill the

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

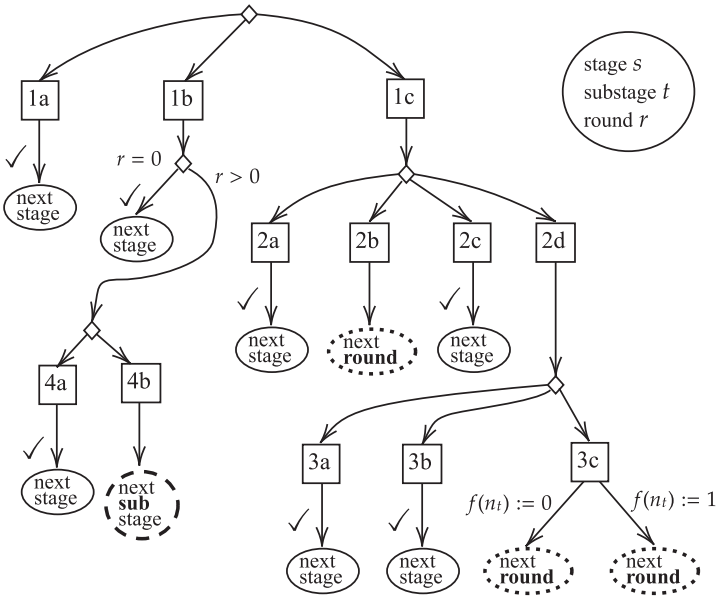


Figure 2: Overview of the $R_{e,d,j}$-strategy.

following requirements:

$$P_e\colon f \notin \varphi_e.$$
$$R_{e,d,j}\colon \text{If } g \subseteq \Phi_e[f] \text{ and } h \subseteq \Phi_d[f] \text{ and } f \subseteq \Phi_j[g \oplus h] \text{ then } f \leq_{\mathrm{subT}} g \text{ or } f \leq_{\mathrm{subT}} h.$$

At the beginning of the stage s, assume that a computable infinite set $I_s \subseteq \omega$ has been constructed, and $f \upharpoonright \min I_s$ has been determined. That is, $f \upharpoonright \min I_s$ is guaranteed not to change after stage s. Each $n \notin I_s$ with $n > \min I_s$ is removed from the domain of f.

P_e-strategy: Assume $s = 2e$. For $n = \min I_s$, one can choose the value of $f(n)$ so that $f(n) \downarrow \neq \varphi_e(n)$. Put $I_{s+1} = \{x \in I_s : x > n\}$. Then, the values of f up to n will be protected. The requirement P_e is clearly satisfied.

$R_{e,k,j}$-strategy: Assume $s = 2\langle e, d, j\rangle + 1$. This strategy consists of infinitely many substrategies. The t^{th} substrategy works with the t-th smallest element n_t of I_s. In particular, $n_0 = \min I_s$, and $f \upharpoonright n_0$ has already been determined. As long as we focus only on this one substrategy, it is somewhat similar to the strategy of Theorem 4.9. In particular, the t^{th} substrategy itself also consists of infinitely many rounds. See Figure 2 for the overview of the $R_{e,d,j}$-strategy.

First, regarding the computation of $\Phi_j[g \oplus h](n_t)$, we have the following possibilities:

(1a) $\Phi_j[g \oplus h](n_t)$ is undefined without making a query.
(1b) $\Phi_j[g \oplus h](n_t)$ halts without making a query.
(1c) $\Phi_j[g \oplus h](n_t)$ makes a query for g or h.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Here, in the k^{th} round (starting with $k = 0$), we ask about the k^{th} query during the computation of $\Phi_j[g \oplus h](n_t)$. For instance, (1b) in the k^{th} round means that this computation has already made k queries (since we have reached the k^{th} round) and halts without making any further queries. That is, the computation halts after making exactly k queries.

Case (1a): In this case, by setting $f(n_t) \downarrow$, one can guarantee that $f(n_t) \neq \Phi_j[g \oplus h](n_t)$. Here, if it is not the first round, set $f(n_t) \downarrow$ to be a value consistent with the current assumption on $f(n_t)$ (made by the previous rounds). Put $I_{s+1} = \{x \in I_s : x > n_t\}$, and proceed to stage $s + 1$.

Case (1b): If we are in the first round, the computation halts without ever creating a query, so the value of $\Phi_j[g \oplus h](n_t)$ is constant regardless of what oracle $g \oplus h$ is. In this case, by setting $f(n_t) \downarrow$ to be an appropriate value, we can guarantee that $f(n_t) \neq \Phi_j[g \oplus h](n_t)$. Put $I_{s+1} = \{x \in I_s : x > n_t\}$ and proceed to stage $s + 1$.

If the round has already progressed, there may already be a restriction on the value of $f(n_t)$, which may prevent diagonalization, so the action in this case is discussed in detail below.

Case (1c): Since the argument is symmetric, assume that the computation has made a query to g. If the assumption $g \subseteq \Phi_e[f]$ is satisfied, then the solution of the query m to g should be obtained from the solution of $\Phi_e[f](m)$. In this case, we have the following possibilities:

(2a) $\Phi_e[f](m)$ does not halt, but makes only small queries less than n_0.
(2b) $\Phi_e[f](m)$ halts after making only small queries less than n_0.
(2c) $\Phi_e[f](m)$ makes a query $n' \geq n_0$ with $n' \neq n_t$.
(2d) None of the above; that is, $\Phi_e[f](m)$ makes a query n_t.

Case (2a): In this case, $m \notin \mathrm{dom}(g)$ is ensured, so $\Phi_j[g \oplus h](n_t) \uparrow$. As in Case (1a), by setting $f(n_t) \downarrow$, one can guarantee that $f(n_t) \neq \Phi_j[g \oplus h](n_t)$. Here, set $f(n_t) \downarrow$ to be a value consistent with the current assumption on $f(n_t)$ (made by the previous rounds). Put $I_{s+1} = \{x \in I_s : x > n_t\}$ and proceed to stage $s + 1$.

Case (2b): It means that the solution to the query m for g has already been determined from the current information in $f \upharpoonright n_0$. In this case, proceed to round $k + 1$ of the t^{th} substrategy.

Case (2c): By setting $f(n') \uparrow$, one can force $g(m) \uparrow$, which also forces $\Phi_j[g \oplus h](n_t) \uparrow$. Since $n' \neq n_t$, as in Case (1a), one can set $f(n_t) \downarrow$ to guarantee that $f(n_t) \neq \Phi_j[g \oplus h](n_t)$. Here, set $f(n_t) \downarrow$ to be a value consistent with the current assumption on $f(n_t)$ (made by the previous rounds). Put $I_{s+1} = \{x \in I_s : x > n_t, n'\}$ and proceed to stage $s + 1$.

Case (2d): In this case, the computation may depend on the value of $f(n_t) \in \{0, 1\}$, so the case is divided again.

(3a) $\Phi_e[f](m)$ may make a query $n' \geq n_0$ with $n' \neq n_t$, depending on the value of $f(n_t)$.
(3b) $\Phi_e[f](m)$ may not halt, depending on the value of $f(n_t)$.
(3c) Otherwise, that is, $\Phi_e[f](m)$, for any value of $f(n_t)$, halts after only making a query $n' < n_0$ or $n' = n_t$.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Here, if the value of $f(n_t)$ has already been determined, we only look at that value.

Cases (3a) and (3b): Set $f(n_t)$ to the corresponding value. In this case, we perform the same action as in Cases (2c) and (2a), respectively, and then proceed to stage $s+1$.

Case (3c): In this case, proceed to round $k+1$ of the t^{th} substrategy. However, in this case, the computation of $\Phi_e[f](m)$ may depend on the value of $f(n_t)$, so we perform the round $k+1$ when $f(n_t)=0$ and when $f(n_t)=1$ in parallel.

Next rounds: The next round proceeds only if the substrategy reaches (2b) or (3c). If the substrategy keeps arriving only at (2b) or (3c) for some value of $f(n_t)$, it means that the round keeps going infinitely, and this also means, in particular that (1c) is passed through infinitely many times. In this case, the computation of $\Phi_j[g\oplus h](n_t)$ continues to make queries infinitely many times, so $\Phi_j[g\oplus h](n_t)$ does not halt. Thus, by setting $f(n_t)\downarrow$, one can guarantee that $f(n_t)\neq\Phi_j[g\oplus h](n_t)$. Here, set $f(n_t)\downarrow$ to be a value consistent with the current assumption on $f(n_t)$. Put $I_{s+1}=\{x\in I_s : x>n_t\}$, and proceed to stage $s+1$.

Note that if, for some value of $f(n_t)$, the substrategy eventually reaches a state other than (1b), (2b), or (3c), then we proceed to the next stage, after satisfying the requirement. Also, as we saw above, if we only arrive at (2b) or (3c) for some value of $f(n_t)$, we also satisfy the requirement and proceed to the next stage. The only remaining case is when we arrive at (1b) no matter how we choose the value of $f(n_t)$. It is necessary to describe the action at (1b) after the round has progressed.

Case (1b): After the round has progressed, the computation may be asking for the value of $f(n_t)$. At this time, there are two different possibilities in Case (1b).

(4a) The value of $\Phi_j[g\oplus h](n_t)$ does not depend on whether $f(n_t)=0$ or $f(n_t)=1$.

(4b) The value of $\Phi_j[g\oplus h](n_t)$ depends on whether $f(n_t)=0$ or $f(n_t)=1$.

Case (4a): In this case, one can choose the value of $f(n_t)$ so that $f(n_t)\neq\Phi_e[g\oplus h](n_t)$. Put $I_{s+1}=\{x\in I_s : x>n_t\}$, and proceed to stage $s+1$.

Case (4b): In this case, start running the $(t+1)$-th substrategy.

Let us analyze the situation when (4b) occurs. First, note that the behavior of each substrategy does not depend on the behavior of the other substrategies. In other words, since the behavior of the t^{th} substrategy depends only on the values of $f\upharpoonright n_0$ and $f(n_t)$, in fact, one can operate all substrategies in parallel. Furthermore, if (4b) occurs, then the computation $\Phi_j[g\oplus h](n_t)$ only makes queries $n\in\mathrm{dom}(f)$ with $n<n_0$ or $n=n_t$. Now, for each $i<2$, let f_i be an extension of $f\upharpoonright n_0$ such that $f_i(n_t)=i$. Dependently, the values of g and h also change, so that $\hat{g}_i=\Phi_e[f_i]$ and $\hat{h}_i=\Phi_d[f_i]$.

Note that (4b) occurs because the solution of a query differs somewhere in the Φ_j-computations for $f(n_t)=0$ and for $f(n_t)=1$. That is, for the queries $m_0,m_1,\dots$ during the computation of $\Phi_j[\hat{g}_0\oplus\hat{h}_0](n_t)$ and the queries $m'_0,m'_1,\dots$ during the computation of $\Phi_j[\hat{g}_1\oplus\hat{h}_1](n_t)$, since we are running the same computation Φ_j, these queries must be the same until the very first difference on the solutions occurs. That is, there exists some $k(t)$ such that $m_i=m'_i$ for each $i\leq k(t)$, but the solutions for $m_{k(t)}=m'_{k(t)}$ are different. For instance, if this is a query to g, then $\hat{g}_0(m_{k(t)})\neq\hat{g}_1(m'_{k(t)})$. Let us now define $\tilde{m}_t:=m_{k(t)}$. The point is that if $\hat{g}=\Phi_e[f]$ and $\hat{h}=\Phi_d[f]$, then one can recover the value of $f(n_t)$ from the value of $\hat{g}(\tilde{m}_t)$ or $\hat{h}(\tilde{m}_t)$.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

Moreover, even if $g \subseteq \hat{g}$ and $h \subseteq \hat{h}$, if $\Phi_j[g \oplus h](n_t)\downarrow$, then the same computation is performed without accessing outside the domains of g and h, so one can recover the value of $f(n_t)$ from the value of $g(\tilde{m}_t)$ or $h(\tilde{m}_t)$ in the same way. This is the case, for example, when $n_t \in \mathrm{dom}(f)$ and $f \leq_{\mathrm{subT}} g \oplus h$ via j.

Infinite substages: Recall that, if a substrategy does not reach (4b) for some value of $f(n_t)$, then we proceed to the next stage. Therefore, the process to proceed to the next stage is not yet described if all substrategies t reach (4b) for any value of $f(n_t)$. Also, a new substrategy is activated only when it reaches (4b). Note that no action is taken at this point.

If each substrategy t reaches (4b), then we can use the information on the finite function $f \upharpoonright n_0$ to compute the sequences of queries $m_0, m_1, \ldots$ and $m'_0, m'_1, \ldots$, and eventually find a query $\tilde{m}_t$ such that $\hat{g}_0(\tilde{m}_t) \neq \hat{g}_1(\tilde{m}_t)$ or $\hat{h}_0(\tilde{m}_t) \neq \hat{h}_1(\tilde{m}_t)$. This procedure $t \mapsto \tilde{m}_t$ is computable using only $f \upharpoonright n_0$. Furthermore, for $\hat{g} = \Phi_e[f]$ and $\hat{h} = \Phi_d[f]$, the value of $f(n_t)$ is determined from $\hat{g}(\tilde{m}_t)$ or $\hat{h}(\tilde{m}_t)$. Let $J_s \subseteq I_s$ be the set of locations n_t such that $\tilde{m}_t$ is a query to the g-side and $\hat{g}_0(\tilde{m}_t) \neq \hat{g}_1(\tilde{m}_t)$. If J_s is infinite, put $I_{s+1} = J_s$, otherwise, $I_{s+1} = I_s \backslash J_s$. We then proceed to stage $s+1$.

Verification: The requirement P_e clearly ensures that f is not computable. Suppose now that $g, h \leq_{\mathrm{subT}} f$ and $f \leq_{\mathrm{subT}} g \oplus h$. Then, $g \subseteq \Phi_e[f]$, $h \subseteq \Phi_d[f]$ and $f \subseteq \Phi_j[g \oplus h]$ for some $e, d, j \in \omega$. If some substrategy of the $R_{e,d,j}$-strategy does not arrive at (4b), then $f \not\subseteq \Phi_j[g \oplus h]$ is guaranteed as discussed above, so all substrategies will always arrive at (4b). Let $s = 2\langle e, d, j\rangle + 1$ and consider $n_0 = \min I_s$. By the construction of f, for any $n \in \mathrm{dom}(f)$, $n \geq n_0$ implies $n \in I_{s+1}$. For any $n_t \in I_{s+1}$, one can compute $\tilde{m}_t$ using information on the finite function $f \upharpoonright n_0$. As noted above, if $n_t \in \mathrm{dom}(f)$, then by our assumption that $f \leq_{\mathrm{subT}} g \oplus h$ via j, the value of $f(n_t)$ can be computed from $g(\tilde{m}_t)$ or $h(\tilde{m}_t)$. If $n_t \in J_s$, then the former, and if $n_t \notin J_s$, then the latter. Thus, we obtain $f \leq_{\mathrm{subT}} g$ if $I_{s+1} = J_s$ and $f \leq_{\mathrm{subT}} h$ if $I_{s+1} = I_s \backslash J_s$. ∎

When translated into the context of realizability theory using Fact 1, this result means that there is a realizability subtopos of the effective topos that cannot be the meet of two larger realizability subtoposes of the effective topos (with respect to the ordering by geometric inclusions).

Acknowledgements The authors wish to thank David Belanger for valuable discussions.

References

[1] N. L. Ackerman, C. E. Freer, and R. S. Lubarsky, *An introduction to feedback Turing computability*. J. Logic Comput. **30**(2020), no. 1, 27–60.
[2] M. Beeson, *Goodman's theorem and beyond*. Pacific J. Math. **84**(1979), no. 1, 1–16.
[3] C. T. Chong and Y. Liang, *Recursion theory: Computational aspects of definability*, volume 8 of De Gruyter Series in Logic and its Applications, De Gruyter, Berlin, 2015. With an interview with Gerald E. Sacks.
[4] S. B. Cooper, *Computability theory*, Chapman & Hall/CRC, Boca Raton, FL, 2004.
[5] D. D. Dzhafarov, *Joins in the strong Weihrauch degrees*. Math. Res. Lett. **26**(2019), no. 3, 749–767.
[6] E. Faber and J. van Oosten, *More on geometric morphisms between realizability toposes*. Theory Appl. Categ. **29**(2014), 874–895.
[7] N. D. Goodman, *Relativized realizability in intuitionistic arithmetic of all finite types*. J. Symbolic Logic **43**(1978), no. 1, 23–44.

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press

[8] J. M. E. Hyland, *The effective topos*. In: A. S. Troelstra and D. van Dalen (eds.), *The L.E.J. Brouwer centenary symposium (Noordwijkerhout, 1981)*, volume 110 of Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1982, pp. 165–216.
[9] T. Kihara, *Rethinking the notion of oracle: A prequel to Lawvere-Tierney topologies for computability theorists*. Preprint, 2022. arXiv:2202.00188.
[10] T. Kihara, *Lawvere-Tierney topologies for computability theorists*. Trans. Amer. Math. Soc. Ser. B **10**(2023), 48–85.
[11] T. Kihara and K. M. Ng, *Church's thesis in subtoposes of the effective topos*. In preparation, 2024.
[12] D. Madore, *Various notions of Turing reduction for partial functions*. 2012. https://mathoverflow.net/questions/112617/various-notions-of-turing-reduction-for-partial-functions
[13] P. Odifreddi, *Classical recursion theory: The theory of functions and sets of natural numbers*, volume 125 of Studies in Logic and the Foundations of Mathematics, North-Holland Publishing Co., Amsterdam, 1989. With a foreword by G. E. Sacks.
[14] L. P. Sasso, *A survey of partial degree*. J. Symbolic Logic **40**(1975), 130–140.
[15] M. I. Soskova, *The theory of the enumeration degrees, definability, and automorphisms*. In: A. Rezuș (ed.), *Contemporary logic and computing*, volume 1 of Landscape Log, College Publications, London, 2020, pp. 706–730.
[16] B. van den Berg and L. van Slooten, *Arithmetical conservation results*. Indag. Math. (N.S.) **29**(2018), no. 1, 260–275.
[17] J. van Oosten, *A semantical proof of de Jongh's theorem*. Arch. Math. Logic **31**(1991), no. 2, 105–114.
[18] J. van Oosten, *A combinatory algebra for sequential functionals of finite type*. In: S. B. Cooper and J. K. Truss (eds.), *Models and computability (Leeds, 1997)*, volume 259 of London Mathematical Society Lecture Note Series, Cambridge University Press, Cambridge, 1999, pp. 389–405.
[19] J. van Oosten, *A general form of relative recursion*. Notre Dame J. Formal Logic **47**(2006), no. 3, 311–318.
[20] J. van Oosten, *Realizability: An introduction to its categorical side*, volume 152 of Studies in Logic and the Foundations of Mathematics, Elsevier B. V., Amsterdam, 2008.

Nagoya University, Japan
e-mail: kihara@i.nagoya-u.ac.jp

Nanyang Technological University, Singapore
e-mail: kmng@ntu.edu.sg

https://doi.org/10.4153/S0008414X25101922 Published online by Cambridge University Press