Draft: Time Bounds and The Unit Cost Model for Clocking Type-2 Computation

Chung-Chih Li
cli2@ilstu.edu

School of Information Technology
Illinois State University, Normal, IL 61790, USA

Abstract. In [15] we defined a class of functions called Type-2 Time Bounds (henceforth $\mathbf{T_2TB}$) for clocking the Oracle Turing Machine (henceforth OTM) in order to capture the long missing notion of complexity classes at type-2. In the present paper we adopt the same notion and further advance this apt type-2 complexity theory along the line of the classical one. Albeit the OTM is mostly accepted as a natural computing device for type-2 computation, the complexity theorems based on the OTM are highly sensitive to the convention of the machine. In particular, the cost model in dealing with the oracle answers turns out to be a crucial factor in the theory. Almost all existent literatures on the machine characterization for type-2 computation are based on a cost model known as *answer-length cost* model. In this paper we present a reasonable alternative called *unit cost* model, and examine how this model shapes the outlook of the type-2 complexity theory. We prove two theorems that are opposite to the classical union theorem and gap theorem under our unit cost model (the results are also opposite to ones under the answer-length cost model [13, 15]). Without the rather bizarre gap phenomena, our formulation for type-2 computation may be considered as a better model that fits our intuitive notion of computation. We also investigate some properties of $\mathbf{T_2TB}$ including a very useful theorem stating that there is an effective operator to convert any $\beta \in \mathbf{T_2TB}$ into an equivalent one that is *locking-detectable*. With this theorem, we can simplify our proofs without loss of generality by assuming any concerned $\beta \in \mathbf{T_2TB}$ locking detectable.

Outline: We organize the present paper as follows. Section 1 contains a very brief introduction to the problem and necessary notations used in this paper. Sections 2 and 3 include definitions of our type-2 time bounds and the clocking scheme. We put the main results of our type-2 complexity theory in Section 4. In Section 5 we discuss the properties of type-2 time bounds. Section 6 concludes our investigation with prospects of future study. We put all proofs in the appendixes.

1 Introduction

Let $\mathbf{N}$ denote the set of natural numbers. Consider an *acceptable programming system*, $\langle \varphi_i \rangle_{i \in \mathbf{N}}$, and let $\langle \Phi_i \rangle_{i \in \mathbf{N}}$ be a *complexity measure* associated to $\langle \varphi_i \rangle_{i \in \mathbf{N}}$.

Simply put, one may consider φ_i as the function computed by the i^{th} Turing machine. A formal definition for an *acceptable programming system* can be found in [21, 20]. For a complexity measure, one may consider $\Phi_i(x)$ as the amount of resource needed to compute φ_i on x. We use $\varphi_i(x) \downarrow = y$ to denote that the computation of φ_i on x is converged and its value is y. Similarly, $\Phi_i(x) \downarrow = m$ means the cost of computing φ_i on x is converged to m. Two landmark papers, Hartmanis and Stearns [8] and Blum [1], initiated an important study now known as *abstract complexity theory* in theoretical computer science. In [8] Hartmanis and Stearns gave a precise definition for complexity classes as follows:

$$\mathbf{C}(t) = \left\{ \varphi_i \;\middle|\; i \in \mathbf{N}, \Phi_i \leq^* t \right\},$$

where $\Phi_i \leq^* t$ means that the relation, $\Phi_i(x) \leq t(x)$, should hold on all but finitely many values of x. Within two years, Blum proposed two axioms as the basic requirements for any reasonable dynamic complexity measures to meet. The two requirements are straightforward: for any $i, x, m \in \mathbf{N}$, we require

1. $\varphi_i(x) \downarrow$ if and only if $\Phi_i(x) \downarrow$, and
2. $\Phi_i(x) = m$ is effectively decidable.

The two axioms had successfully lifted the study of complexity theory to an abstract level with rich results that are independent from any specific machine models.

It is obvious that the complexity theory should be extended into type-2 (a.k.a. second-ordered) computation. This inquiry can be traced back to Constable's 1973 paper [6] in which he asked what should a type-2 complexity theory look like? However, only a few scattered works had been done in the past three decades due to the difficulty of having a generally accepted abstraction for type-2 computation. Traditionally, we may use the OTM as our standard formalism for type-2 computation together with Blum's two axioms as the framework of the theory. But, these are not enough to materialize a robust complexity theory at type-2 due to many other problems. Our first question is that, what is a reasonable notion of asymptotical behaviors of type-2 computation? Such notion is needed in order to simplify our analysis without loss of generality; e.g., the use of "$\leq^*$" in the definition of Hartmanis and Stearns's classes is such a notion. The second question is, how do we clock an OTM with a resource bound so we can shutdown the machine for good reasons? Also, the property of the resource bound should properly reflect the complexity property of the machine being clocked. Moreover, the clocking scheme should satisfy Blum's two axioms in a natural way, i.e., we do not want to impose too many nonsensical restrictions on the resource bounds just to satisfy the two axioms. All of these do not come easy, and that explains why the body of type-2 complexity theory is still so thin after all these years when the heyday of abstract complexity theory is long past.

Recently, we introduced a notion of type-2 asymptotical behaviors in [14] to catch the idea of its type-1 counterpart – *for all but finitely many*. Using the notion of type-2 asymptotical behaviors we answer the first question stated in the previous paragraph and lift the complexity classes into type-2. To our second

question, in [15] we proposed a class of type-2 time bounds and a clocking scheme as a partial answer. In the present paper, we further study the properties of our type-2 time bounds and point out that the type-2 complexity theory is highly sensitive to the actual *cost model* used in the clocking scheme. We believe that our investigation initiates a sound framework for theorists to further speculate a more complete machine-independent complexity theory for type-2 computation.

Notations: We first introduce some necessary notations. By convention, the natural numbers are taken as type-0 objects and type-1 objects are functions over natural numbers. Type-2 objects are *functionals* that take as inputs and produce as outputs type-1 objects. Let type-0 $\subset$ type-1 $\subset$ type-2. We are only interested in total functions of type $\mathbf{N} \to \mathbf{N}$ when they are taken as inputs of type-2 functionals. For convenience, we use $\mathcal{T}$ to denote the set of total functions and $\mathcal{P}$ to denote the set of partial functions. Also, we use $\mathcal{F}$ to denote the set of *finite* functions, which means $\sigma \in \mathcal{F}$ if and only if $\mathsf{dom}(\sigma) \subset \mathbf{N}$ and $\mathsf{card}(\sigma) \in \mathbf{N}$. We fix a canonical indexing for $\mathcal{F}$, and hence we are free to treat any function in $\mathcal{F}$ as a number when it is taken as the input of a type-1 function. Let $\langle \cdot, \cdot \rangle$ be the standard pairing function defined in [20]. Thus, $\langle \sigma, x \rangle \in \mathbf{N}$ for every $\sigma \in \mathcal{F}$ and $x \in \mathbf{N}$. Unless stated otherwise, we let a, b, x, y, z range over $\mathbf{N}$, f, g, h range over $\mathcal{T}$, and F, G, H range over type-2 functionals.

Here we present some typical examples of type-2 functionals: 1. $F(f, x) = f(x)$; 2. $G(f, x) = f(f(x))$; 3. $H(f, x) = \sum_{i=0}^{x} f(i)$; 4. $\Gamma(f) = f \circ f$, where $\circ$ is function composition. F, G, and H are type-2 functionals of type $\mathcal{T} \times \mathbf{N} \to \mathbf{N}$, while Γ is of type $\mathcal{T} \to \mathcal{T}$. With λ-abstraction, we have $\Gamma(f) = \lambda x G(f, x)$. Since some complexity theorems at type-2 can be easily proven by the same tricks used in the original proofs, we therefore keep a type-0 input in order to take this advantage. In some cases, however, the type-0 input seems redundant if the type-1 input is the main character in the proof. We also note that $\mathcal{T} \cong \mathcal{T} \times \mathbf{N}$ via, for example, $f \mapsto (f', f(0))$, where $f'(x) = f(x+1)$. Thus, we do not lose generality when we restrict type-2 functionals to our standard type $\mathcal{T} \times \mathbf{N} \rightharpoonup \mathbf{N}$. With this restriction, we can take the OTM as our standard formal computing device as follows: The numerical input is prepared from the beginning of the tape, and the functional input is prepared as a function oracle attached to the machine. During the course of computation, when the OTM needs a value from the function, the OTM shall query the oracle in a standard manner.[1] Moreover, we fix a programming system $\langle \widehat{\varphi} \rangle_{i \in \mathbf{N}}$ associated with a complexity measure $\langle \widehat{\Phi} \rangle_{i \in \mathbf{N}}$ for our OTM's. Conventionally, we take the number of steps an OTM performed as our time complexity measure. Once a resource bound and a proper

[1] An OTM has two extra tapes: one query tape and one answer tape. The OTM has to prepare the query in the query tape, and then transits to a special state called query state in which the oracle will place the answer to the query in the answer tape in one step, no matter how big the answer might be. Note that the steps needed for the OTM to write the query and read the answer returned from the oracle are counted as a part of the computational cost of the OTM.

clocking scheme are given, we can further restrict our functionals of interest to be total.

2 Type-2 Complexity Classes and Time Bounds

In [23] Seth followed Hartmanis and Stearns's notion [8] and proposed the following two definitions for type-2 complexity classes. Let $|n|$ be the length of the presentation of $n \in \mathbf{N}$.

Definition 1 (Seth [23]). *Let $t : \mathbf{N} \to \mathbf{N}$ be recursive. Let DTIME(t) denote the set of type-2 functionals such that, for every functional $F \in DTIME(t)$, F is total and there is an OTM $\widehat{M}_e$ that computes F and, on* every *$(f, x) \in \mathcal{T} \times \mathbf{N}$, $\widehat{M}_e$ halts within $t(m)$ steps, where $m = |\mathsf{max}(\{x\} \cup Q)|$ and Q is the set of all answers returned from the oracle during the course of the computation.*

Definition 2 (Seth [23]). *Let $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ be computable. Let DTIME(H) denote the set of type-2 functionals such that, for every functional $F \in DTIME(H)$, F is total and there is an OTM $\widehat{M}_e$ that computes F and, on* every *$(f, x) \in \mathcal{T} \times \mathbf{N}$, $\widehat{M}_e$ halts within $H(f, x)$ steps.*

Clearly, the key idea behind the two definitions is directly lifted from Hartmanis and Stearns's [8]. In fact, the same idea can also be found in other works such as [9, 10, 22, 17] along the line of machine characterizations for type-2 complexity classes. However, the two introduce new problems that do not exist in type-1 computation. In Definition 1, Q is not computable in general, and hence not every complexity class has a recursive presentation. Alternatively, we can update the bound dynamically upon each oracle query made during the course of the computation, e.g., Cook's POTM [7] is an OTM bounded by a polynomial in this manner. However, this introduces another problem, i.e., a clocked OTM may not be total as Cook himself has noted; even if the bound t is as small as in $O(n^2)$ [13]. For Definition 2, we can find a direct implementation from Kapron and Cook's works on type-2 Basic Feasible Functionals (BFF), where the bound H in the definition is a so-called second-ordered polynomial [9, 10]. With a proper notation of type-2 almost everywhere relation such as $\leq_{\mathbf{2}}^{*}$ in [14], a more workable type-2 complexity class defined by a computable $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ is given as follows:

$$DTIME(H) = \left\{ F \mid \exists e \left[\widehat{\varphi}_e = F \text{ and } \widehat{\Phi}_e \leq_{\mathbf{2}}^{*} H \right] \right\}. \tag{1}$$

In type-1, the complexity measure is computed based on the *size* of the input, i.e., the cost of computing $\varphi_e(x)$ is $\Phi_e(|x|)$, where $|x|$ denotes the length of the bit string of x's binary representation. We emphasize that the value of $|x|$ is much smaller than x itself. For type-2 computation, Kapron and Cook [9, 10] introduced the *length* for type-1 function f as follows:

$$|f| = \lambda a.\mathsf{max}(\{y \mid y = |f(x)|, |x| \leq a\}).$$

These all seem well for BFF's, but in general, however, we do not think using a type-2 functional as a resource bounds together with a length function is proper for two reasons. First of all, as a matter of fact, only a finite part of the type-1 input contributes to the computation, it seems unnecessary to let the resource bound "know" the entire type-1 input. In other words, the resource bound should not depend on information that is irrelevant to the computation. Secondly, the length function is difficult to compute, i.e., computing $(f, x) \mapsto |f|(|x|)$ is not basic feasible. Consequently, a clocked OTM will become inefficient if the cost of computing the bound is considered. To avoid these problems, we give a class of functions called Type-2 Time Bounds as explicit time bounds for clocking OTMs [15]. Consider the following definitions, where $\beta : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$ is a function that takes as inputs a finite function and a numerical number and produce a number as output:

Definition 3 (Type-2 Time Bounds). *Let $\beta : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$. We say that:*

1. *β is nontrivial, if for every $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$, $\beta(\sigma, a) \geq |a| + 1$;*
2. *β is bounded, if for every $(f, x) \in \mathcal{T} \times \mathbf{N}$, $\sigma \in \mathcal{F}$, and $\sigma \subset f$, we have $\beta(\sigma, x) \leq \lim_{\tau \to f} \beta(\tau, x)$;*
3. *β is convergent, if for every $(f, a) \in \mathcal{T} \times \mathbf{N}$, there exists $\sigma \in \mathcal{F}$ with $\sigma \subset f$ such that, for all τ with $\sigma \subseteq \tau$, we have $\beta(\sigma, a) = \beta(\tau, a)$;*
4. *β is $\mathcal{F}$-monotone, if for every $a \in \mathbf{N}$ and $\sigma, \tau \in \mathcal{F}$ with $\sigma \subseteq \tau$, we have $\beta(\sigma, a) \leq \beta(\tau, a)$.*
5. *β is a type-2 time bound if β is computable, nontrivial, bounded, and convergent;*
6. *β is a strong type-2 time bound if β is an $\mathcal{F}$-monotone type-2 time bound.*

We use $\mathbf{T_2TB}$ to denote the set of all type-2 time bounds. In [15] we used WB to denote the set of type-2 time bounds and SB to denote the set of strong type-2 time bounds. Clearly, SB $\subset$ WB and SB $\neq$ WB. Consider the following example: Define $\beta : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$ by

$$\beta(\sigma, x) = \sum_{0 \leq i \leq x \ \wedge \ i \in \mathsf{dom}(\sigma)} \sigma(i) + |x| + 1. \tag{2}$$

One can verify that β in (2) is computable, nontrivial, bounded, and convergent, and hence $\beta \in \mathbf{T_2TB}$. Also, one can verify that β is strong.

Since $\mathbf{T_2TB}$ is designed for clocking OTM's, we formulate the properties listed in Definition 3 to reflect our intuition about what should be a resource bound in computation. Nontriviality assures that the resource is at least enough for every OTM to scan the entire type-0 input at the beginning of its computation. Boundedness requires the value of β to be bounded by its value in the limit. In other words, the maximal value of β exists in the limit. Finally, if β is convergent, then the value of β must converge to a number and hence every OTM clocked with β must terminate on any input sooner or later. It is clear that a strong β must be bounded, since a strong β never shrinks during the course of the computation. Our original proposal require a type-2 time bound to be

$\mathcal{F}$-monotone instead of just bounded. But we later found that most interesting theorems sustain with this weaker version of $\mathbf{T_2TB}$.

By a standard diagonalization, one can prove that $\mathbf{T_2TB}$ is not *recursively enumerable*. This indeed is an uneasy fact, since we therefore can't enumerate $\mathbf{T_2TB}$ when needed in some proofs. Being able to effectively decide whether a β has converged seems to be a powerful assumption. It is easy to prove that to decide whether or not a β has converged can be reduced to the halting problem, and this seems to suggest that we cannot make this strong assumption in our theorems. However, we have a very positive theorem stating that making such a strong assumption in our proofs does not lose the generality of our theorems. We first formalize some terminologies in the following definition.

Definition 4 (Locking Detectors). *Let $\beta \in \mathbf{T_2TB}$. Suppose $\ell : \mathcal{F} \times \mathbf{N} \to \{0,1\}$ is computable with the following three properties:*

1. *ℓ is $\mathcal{F}$-monotone;*
2. *$\forall (\sigma, x) \in \mathcal{F} \times \mathbf{N} \left[\ell(\sigma, x) = 1 \implies \beta(\sigma, x) \downarrow\right]$, where $\beta(\sigma, x) \downarrow$ means $\forall \tau \supseteq \sigma[\beta(\sigma, a) = \beta(\tau, a)]$.*
3. *$\forall (f, x) \in \mathcal{T} \times \mathbf{N} \left[\lim_{\sigma \to f} \ell(\sigma, x) = 1\right]$.*

Then, we say that ℓ is a locking detector of β.

If $\beta \in \mathbf{T_2TB}$ has a locking detector, we say that β is locking detectable. One can verify that the β defined in (2) is locking detectable. Let ℓ be such a locking detector of β. We say that $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$ is a *locking fragment* of β if $\ell(\sigma, x) = 1$. Since the converse of property 2 in Definition 4 above needs not to be true, it follows that a minimal locking fragment of ℓ may not be a minimal one of β. More properties of $\mathbf{T_2TB}$ will be shown in Section 5, where we will argue that not every type-2 time bound is locking detectable, but as we mentioned earlier, we will show that the theorems proven with the assumption of locking detectability are still valid in the general case.

3 A Clocking Scheme and Two Cost Models

Here we present an OTM clocking scheme using our $\mathbf{T_2TB}$. This scheme is implied in the works of Kapron and Cook's [9, 10], Seth's [23], and Royer's [22].

Definition 5 (Clocked OTM). *Let $\beta \in \mathbf{T_2TB}$ and $\widehat{M}_e$ be an OTM with index e. We say that $\widehat{M}_e$ is clocked by β if and only if $\widehat{M}_e$ is simulated by the procedure shown in Figure 1. Such a clocked OTM is denoted by $\widehat{M}_{e,\beta}$ computing a functional denoted by $\widehat{\varphi}_{e,\beta}$.*

Consider the procedure in Figure 1. The *budget* provided by β is computed based on the answers to the oracle queries during the course of the simulation of $\widehat{M}_e$ on (f, x). If the simulation has overrun the budget, then the simulation will be terminated at the line marked ($\Uparrow$). In this case we say that $\widehat{M}_e$ is *clipped down* by β on (f, a). We denote this situation by $\widehat{\varphi}_{e,\beta}(f, a) \Uparrow$. On the other hand, if the

```
Program for Clocked OTM M̂_{e,β} :
    input (f, x) ∈ T × N;
    var σ ∈ F; q, y, expense, budget ∈ N;                    /* variable declaration */
    σ ⟵ ∅; expense ⟵ 0; budget ⟵ β(σ, x);                    /* initialization */
    Simulate M̂_e on (f, x) step by step and upon each step completed do:
        expense ⟵ expense + 1;
        if (expense > budget)                                /* check budge */
            then output ⊥ and stop;                          /* ⊥ is the bottom symbol. (⇑) */
        if (M̂_e halts with the output y)
            then output y and stop;                          /* simulation completed. (⇓) */
        if (the step just simulated completes an oracle query)
            then do
                q ⟵ current query;
                σ ⟵ σ ∪ {(q, f(q))};                         /* update query-answer set */
                budget ⟵ β(σ, x);                            /* update budget */
            end-do;
    Resume the simulation;
End program
```

Fig. 1. A Clocking Scheme for OTM's

simulation reaches the line marked (⇓), which means that the simulation of $\widehat{M}_e$ on (f, a) is successfully completed, then we denote this situation by $\widehat{\varphi}_{e,\beta}(f, a) \Downarrow$, or we say $\widehat{\varphi}_{e,\beta}(f, a)$ converges to value $\widehat{\varphi}_e(f, a)$. Since β is convergent, it follows that the simulation of $\widehat{M}_e$ on (f, a) will either complete or eventually be clipped down by the clock. Therefore, for any $\beta \in \mathbf{T_2TB}$, $\widehat{\varphi}_{e,\beta}$ is a total computable functional of the type $\mathcal{T} \times \mathbf{N} \to \mathbf{N}$. This remove POTM's nonterminating situation.

Arbitrarily Complex $\widehat{\varphi}$-program Here we show an easy application in terms of classical recursion theory. Although the locking fragment of β in general is undecidable (see Section 5), we need not to know the value of β in the limit in order to construct an arbitrarily complex $\widehat{\varphi}$-program for any given computable type-2 functional. This is similar to the ordinary type-1 computation.

Theorem 1. *Given any computable $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ and $\beta \in \mathbf{T_2TB}$, there is a $\widehat{\varphi}$-program e for F such that $\widehat{\varphi}_e \neq \widehat{\varphi}_{e,\beta}$.*

Proof: Let $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ be computable. With some proper versions of type-2 S-m-n and recursion theorems such as ones proposed in [11], we can construct a $\widehat{\varphi}$-program e such that, for every $(f, x) \in \mathcal{T} \times \mathbf{N}$,

$$\widehat{\varphi}_e(f, x) = \begin{cases} F(f, x) & \text{if } \widehat{\varphi}_{e,\beta}(f, x) \Uparrow, \\ \widehat{\varphi}_e(f, x) + 1 & \text{otherwise.} \end{cases}$$

It is clear that that $\widehat{\varphi}_e = F$ because the otherwise-case will never occur. □

A slightly more involved theorem is the type-2 version of Rabin Theorem [19], which states that, given any $\beta \in \mathbf{T_2TB}$ there is a 0-1 valued computable

functional $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ that cannot be computed by any $\widehat{\varphi}_{e,\beta}$. We prove a versions in [14] where the bound function is simply a computable type-2 functional. The proof is perfectly valid under the present clocking scheme with $\mathbf{T_2TB}$.

Unfortunately, the properties of $\beta \in \mathbf{T_2TB}$ and our intuitive clocking scheme are not sufficient to standardize a framework for type-2 complexity theory. The way an OTM handles the answers returned from the oracle does matter. We have the following two conventions under our clocking scheme.

Definition 6. (Two Cost Models for OTM's)

1. *Answer-Length Cost Model: Under this model, whenever the oracle returns an answer to the oracle query, the machine is required to read every bit of the answer.*
2. *Unit Cost Model: Under this model, the machine needs not to read any bit of the oracle answer unless the machine decides to do so.*

In [9, 10, 22, 23], the answer-length cost model is assumed for the underlying machine. In [15] the outline of a type-2 complexity theory is also based on the answer-length cost model, in which we mention the unit cost model as a possible alternative without further elaboration. Nevertheless, we do not think the unit cost model is an unusual convention. For example, a computation may need to know if the answer to the query is odd or even. In this case, only the first bit of the answer needs to be scanned. However, the controversial part is that, under the unit cost model, the computation can aggressively gain some budget by just querying the oracle without reading the answers. This adds difficulty in working with the the unit cost model and the complexity theory tends to be much flatter than the theory under the answer-length model. For example, there exist certain versions of Union Theorems [15] and Gap Theorems [13] under the answer-length cost model, but they fail to hold under the unit cost model. In the next section we will discuss some interesting complexity phenomena under the unit cost model.

4 Complexity Theory under The Unit Cost Model

We differentiate the unit cost model from the answer-length cost model by using a superscript u as follows: $\widehat{M}^u_e$, $\widehat{\varphi}^u_e$, $\widehat{\Phi}^u_e$, $\widehat{M}^u_{e,\beta}$, $\widehat{\varphi}^u_{e,\beta}$, and $\widehat{\Phi}^u_{e,\beta}$. For example, $\widehat{M}^u_e$ is the unit cost model OTM with index e, and $\widehat{\varphi}^u_{e,\beta}$ denotes the functional computed by $\widehat{M}^u_e$ with clock β. Since the two models do not differ in computability, we have $\widehat{\varphi}_e = \widehat{\varphi}^u_e$ for every e. On the other hand, $\widehat{\Phi}_e \neq \widehat{\Phi}^u_e$, and hence $\widehat{\varphi}_{e,\beta} \neq \widehat{\varphi}^u_{e,\beta}$ in general. Unless stated otherwise, if the superscript is omitted from the statement of some theorem, we mean that the theorem holds under both cost models. We adapt the notion of type-2 complexity classes proposed in [13–15] but alter the cost model to the unit cost model. The *exception set*, $E^u_{e,\beta}$ is defined as follows: For every $\beta \in \mathbf{T_2TB}$ and $e \in \mathbf{N}$, let

$$E^u_{e,\beta} = \left\{ (f,x) \in \mathcal{T} \times \mathbf{N} \;\middle|\; \widehat{\varphi}^u_{e,\beta}(f,x) \Uparrow \right\}.$$

We also adapt the topology introduced in [14]. That is, given a fixed continuous functional $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$, $\mathbb{T}(F)$ is the topology defined by taking the set of total extensions of every *minimum* locking fragment of F as a basic open set. By locking fragment of F, we mean $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$ such that, for any $f, g \in \mathcal{T}$ with $\sigma \subset f$ and $\sigma \subset g$, we have $F(f, x) = F(g, x)$. Since each functional $\widehat{\varphi}_e$ is continuous, the topology $\mathbb{T}(\widehat{\varphi}_e)$ is well defined. It's also clear that such $\mathbb{T}(F)$ is induced from the Baire topology. Note that, we require σ to be the minimum locking fragment, otherwise $\mathbb{T}(F)$ will inflate to the Baire topology in which every open set can be obtained by taking the union of the total extensions of some locking segments of F. We define our type-2 complexity classes as follows.

Definition 7 (The Type-2 Time Bound Class). *Let $\beta \in \mathbf{T}_2\mathbf{TB}$. Define the set of computable type-2 functionals $\mathbf{C}^u(\beta)$ as*

$$\mathbf{C}^u(\beta) = \left\{ \widehat{\varphi}_e : \mathcal{T} \times \mathbf{N} \to \mathbf{N} \;\middle|\; e \in \mathbf{N} \text{ and } E^u_{e,\beta} \text{ is compact in } \mathbb{T}(\widehat{\varphi}_e) \right\}.$$

4.1 Inclusion property

In [14] we pointed out that it doesn't seem likely to have a reasonable notion for type-2 asymptotic relation, $\leq^*_{\mathbf{2}}$, that is transitive due to the topological constraints. Thus, if a type-2 complexity class is defined by some type-2 functional in the classical manner such as (1), a bigger resource bound does not always promise a bigger complexity class. Our type-2 time bounds and the clocking scheme can easily fix this problem. This adds another reason why we do not think using type-2 functionals as resource bounds for type-2 computation is appropriate. Note that the theorem below holds under both cost models, and thus we drop the superscript, u.

Theorem 2. *For every $\beta_1, \beta_2 \in \mathbf{T}_2\mathbf{TB}$, $[\beta_1 \leq \beta_2] \implies [\mathbf{C}(\beta_1) \subseteq \mathbf{C}(\beta_2)]$.*

Proof: Suppose $\widehat{\varphi}_e \in \mathbf{C}(\beta_1)$. According to the clocking scheme, we have that: if $\beta_1 \leq \beta_2$ then for any $e \in \mathbf{N}$, $E_{e,\beta_2} \subseteq E_{e,\beta_1}$. According to a lemma in [14], if $E_{e,\beta_2} \subseteq E_{e,\beta_1}$ and E_{e,β_1} is compact in $\mathbb{T}(\widehat{\varphi}_e)$, so is the subset E_{e,β_2}. It follows that $\widehat{\varphi}_e \in \mathbf{C}(\beta_2)$. □

4.2 Patchability

Since the value of a continuous functional on a compact set is bounded, it follows that, intuitively, if $\widehat{\varphi}_e \in \mathbf{C}(\beta)$ then we need only some constant extra budget to let $\widehat{\varphi}_e$ finish its computation on every points in $E_{e,\beta}$. This intuition does work for the unit cost model. Thus, we have, if $F \in \mathbf{C}^u(\beta)$, then there exist $c, e \in \mathbf{N}$ such that $\widehat{\varphi}^u_e = F$ and $E^u_{e,\beta+c} = \emptyset$. On the contrary, this intuition doesn't work for the answer-length cost model. In [13] we argued that the best we can have is: If $F \in \mathbf{C}(\beta)$, then there exist $c, e \in \mathbf{N}$ such that $\widehat{\varphi}^u_e = F$ and $E_{e,2\beta+c} = \emptyset$ under the answer-length cost model.

4.3 Enumerability

It is easy to show that the finite invariant closure of a type-1 complexity class is *recursively enumerable* [3]. However, not every complexity class itself can be recursively enumerated. When the cost bound function t is too small, the complexity class determined by t is unlikely to be recursively enumerable [3, 12]. On the other hand, if t is big enough to bound all *finite support functions*[2] almost everywhere, then the complexity class determined by t is recursively enumerable. In particular, if $t(x) \geq |x| + 1$ for all $x \in \mathbf{N}$, then all finite support functions are contained in the complexity class, $C(t)$ (see [4], section 9.4). An intuitive explanation to this is that: if the bound t allows to compute every finite support function almost everywhere, then we can have a program patched at finitely many points with cost bounded by t. In such a way, we can exactly enumerate the complexity class determined by t.

Recall that every $\beta \in \mathbf{T_2TB}$ is nontrivial. This property, unlike at type-1, does not suffice for us to enumerate $\mathbf{C}(\beta)$. Our difficulty is that, given $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$, we may not be able to test if it is the case that $\sigma \subset f$ for any input $f \in \mathcal{T}$ under the the answer-length cost model, since querying the oracle outside the domain of the locking fragment of β is dangerous as the returned answer may be very huge and the machine may use up its budget in scanning the oracle answer as required under the answer-length cost model. To solve this problem, we imposed two rather strong assumptions on β and prove that if β satisfies the two assumptions, then $\mathbf{C}(\beta)$ is recursive enumerable [13]. We also conjectured that there is a $\beta \in \mathbf{T_2TB}$ such that $\mathbf{C}(\beta)$ is not recursively enumerable. Under the unit cost model, on the other hand, we do not need any extra assumptions on β because the cost of querying the oracle is more manageable. We have the following theorem.

Theorem 3. *For every $\beta \in \mathbf{T_2TB}$, $\mathbf{C}^u(\beta)$ is recursively enumerable.*

Proof: Assume that $\beta \in \mathbf{T_2TB}$ is locking detectable. (According to Theorem 10 in Section 5, this assumption does not hurt the generality of our theorem). The reason that we can remove the conditions in the same theorem under the answer-length cost model [13] and makes the present proof a bit easier is that, under the unit cost model, it is possible to let a machine that computes some functional in $\mathbf{C}^u(\beta)$ query the oracle at some points outside the domain of the locking fragment of β. In other words, the cost of testing whether $\tau \subset f$ is manageable.

It is straightforward to obtain an S-m-n theorem on type-0 arguments as follows: There is a recursive function s such that, for every $f \in \mathcal{T}$ and $x, y \in \mathbf{N}$, we have

$$\widehat{\varphi}_{s(e,x)}(f, y) = \widehat{\varphi}_e(f, \langle x, y \rangle).$$

[2] A function f is finite support if the value of f is 0 almost everywhere.

Then, we can use this S-m-n theorem to formulate the recursive function g as follows:

$$\widehat{\varphi}_{g(e,a,b)}(f,x) = \begin{cases} \widehat{\varphi}_e(\tau^{\sim 0},x) \text{ if } \forall\langle\sigma,y\rangle \leq a \left[\widehat{\varphi}^u_{e,\beta+b}(\sigma^{\sim 0},y)\Downarrow\right] \text{ and} \\ \qquad\qquad \forall\langle\sigma,y\rangle \left[(a<\langle\sigma,y\rangle\leq\langle\tau,x\rangle \ \wedge\ \forall\eta\subseteq\sigma(a<\langle\eta,y\rangle)) \right. \\ \qquad\qquad\qquad \left. \Rightarrow \widehat{\varphi}^u_{e,\beta}(\sigma^{\sim 0},y)\Downarrow\right], \text{where } \tau = f_{[2^{F_\beta(f,x)}]}; \\ 0 \qquad\quad \text{otherwise.} \end{cases} \tag{3}$$

In (3), $\tau^{\sim 0}$ denotes the zero extension of τ and $f_{[n]}$ denotes the initial segment of f with length n. Since β is locking detectable, it follows that F_β is computable (see Definition 8 for F_β). In stead of enumerating the set of all finite invariants[3] of $\mathbf{C}^u(\beta)$, we argue in the following that g exactly enumerates all indies for functional in $\mathbf{C}^u(\beta)$.

If $F\in\mathbf{C}^u(\beta)$, then there is a $\widehat{\varphi}$-program e for F such that, $E^u_{e,\beta}$ is compact in $\mathbb{T}(F)$. Thus, if we choose sufficiently large $a,b\in\mathbf{N}$, we have $F=\widehat{\varphi}_{g(e,a,b)}$ for some e. Note that, since β is locking detectable, we can effectively compute $F_\beta(f,x)$, and hence we can find $\tau = f_{[2^{F_\beta(f,x)}]}$. It is clear that any machine queries the oracle beyond τ cannot be bounded by β (since the query is too big to place down). Thus, the information beyond τ is not necessary, and if the condition of the if statement in (3) holds, we have $\widehat{\varphi}_e(f,x)=\widehat{\varphi}_e(\tau^{\sim 0},x)$.

Next, we shall argue that, for every $e,a,b\in\mathbf{N}$, we have $\widehat{\varphi}_{g(e,a,b)}\in\mathbf{C}^u(\beta)$. Fix $e,a,b\in\mathbf{N}$. Let $F=\widehat{\varphi}_{g(e,a,b)}$ and $S\subseteq\mathcal{T}\times\mathbf{N}$ be the set on which the $\widehat{\varphi}$-program $g(e,a,b)$ uses $\widehat{\varphi}_e(\tau^{\sim 0},x)$ as its output, i.e., $\widehat{\varphi}$-program $g(e,a,b)$ on every $(f,x)\in S$, the condition of the first clause in the if statement in (3) holds. We have two cases: (1) S is compact in $\mathbb{T}(F)$ and (2) S is not compact in $\mathbb{T}(F)$. For convenience, let $(\!(\sigma,x)\!) = \{(f,x)\ \big|\ f\in\mathcal{T} \text{ and } \sigma\subset f\}$.

Case 1: S is compact in $\mathbb{T}(F)$.

In this case, we fix a finite open cover for S as follows:

$$\mathcal{O} = \{(\!(\sigma_0,x_0)\!),(\!(\sigma_1,x_1)\!),\dots,(\!(\sigma_n,x_n)\!)\}, \tag{4}$$

such that, for each $(\!(\sigma_i,x_i)\!)\in\mathcal{O}$, (σ_i,x_i) is a minimal locking fragment of F. By the construction, $\sigma_i\subseteq\tau$ for some minimal locking fragment of β, i.e., $\beta(\tau,x_i)\downarrow$. Let $t:\{0,1,\dots,n\}\to\mathbf{N}$ be a finite function such that, for each $i\leq n$, $t(i)=\widehat{\varphi}_e(\sigma_i^{\sim 0},x_i)$, where $(\!(\sigma_i,x_i)\!)\in\mathcal{O}$. Then, we construct a

[3] Enumerating the set of all finite invariants is a classical approach to attacking this problem at type-1. At type-2, we may use the topological approach introduced in [14] to defining finite invariants as two type-2 computable functionals that differ on some compact set. To enumerate finite invariants is easy under the unit cost model, but we need a sort of *linear speedup theorem* to complete our proof, and we don't know yet what should be a linear speedup theorem in type-2 context.

$\widehat{\varphi}$-program e' as follows:

$$\widehat{\varphi}_{e'}(f,x) = \begin{cases} t(i) \text{ if there exists } (\!(\sigma_i, x_i)\!) \in \mathcal{O} \text{ such that} \\ \quad \sigma_i \subset f \text{ and } x_i = x; \\ 0 \quad \text{otherwise.} \end{cases} \tag{5}$$

Clearly, $\widehat{\varphi}_{e'} = F$. Since we are under the unit cost model, for every $(f,x) \in \mathcal{T} \times \mathbf{N}$, we can check the condition in (5) to decide whether 0 or $t(i)$ to be returned for some $i \leq n$ with cost bounded by β. Therefore, $F \in \mathbf{C}^u(\beta)$ in this case.

Case 2: S is not compact in $\mathbb{T}(F)$. Let

$$\mathcal{O} = \{(\!(\sigma_0, x_0)\!), (\!(\sigma_1, x_1)\!), \ldots\}$$

be any open cover of S such that, for each $(\!(\sigma_i, x_i)\!) \in \mathcal{O}$, (σ_i, x_i) is a minimal locking fragment of F. According to the condition of the if statement in (3), if there are infinitely many (τ, x) such that,

$$\forall \langle \sigma, y \rangle \left[(a < \langle \sigma, y \rangle \leq \langle \tau, x \rangle \ \wedge \ \forall \eta \subseteq \sigma (a < \langle \eta, y \rangle)) \Rightarrow \widehat{\varphi}^u_{e,\beta}(\sigma^{\sim 0}, y) \Downarrow \right] \tag{6}$$

is true, then it follows that, for all but finitely many $\langle \tau, x \rangle$, (6) is true. Thus, $E^u_{e,\beta}$ is compact. Moreover, there are only finitely many (τ, x) such that, on $(f,x) \in (\!(\tau, x)\!)$, the $\widehat{\varphi}$-program $g(e,a,b)$ outputs 0 instead of $\widehat{\varphi}_e(f,x)$. Thus, the set

$$A = \left\{ (f,x) \;\middle|\; F(f,x) \neq \widehat{\varphi}^u_e(f,x) \right\}$$

is also compact in $\mathbb{T}(F)$. Thus, we can patch e on A and the computational cost of the patched $\widehat{\varphi}$-program on A does not cause it to be removed from $\mathbf{C}^u(\beta)$. Therefore, $F \in \mathbf{C}^u(\beta)$. □

4.4 Non-Union Theorem

The Union Theorem [16] is one of the most fascinating theorems in classical complexity theory. Not just because the technique used in the proof was new to complexity theory back then, but also the theorem told us that most *natural* complexity classes have clear boundaries in terms of Hartmanis and Stearns's notion of complexity classes. In other words, we can use one computable function to exactly bound a natural complexity class in most cases. Although an arbitrary collection of computable functions (or, union of complexity classes) is not necessarily a complexity class in general, but we only need a very weak condition to have the following so-called union theorem at type-1.

Theorem 4. (McCreight and Meyer [16]) *Let the sequence of recursive functions, f_0, f_1, f_2, ..., be recursive and $f_i(x) \leq f_{i+1}(x)$ for all $i, x \in \mathbf{N}$. Then, there is a recursive function g such that $C(g) = \bigcup_{i \in \mathbf{N}} C(f_i)$.* □

According to the theorem, a complexity class such as $PTIME$, $PSPACE$, etc., each can be exactly named by one recursive function; same to the set of computable functions bounded by computable functions in $O(f)$ (the big-O notation). At type-2, the union theorem seems to break down. For example, the class of type-2 basic feasible functionals is not a complexity class [13]. Nevertheless, in [15, 13] we imposed some quite strong but yet reasonable conditions on the sequence of type-2 time bounds to have a type-2 union theorem under the answer-length cost model. As a result, if we define a type-2 big-O notation as $\mathbf{O}(\beta) = \bigcup_{a,b\in\mathbf{N}} \mathbf{C}(a\beta + b)$, then for each $\beta \in \mathbf{T_2TB}$ we have some $\gamma \in \mathbf{T_2TB}$ such that $\mathbf{C}(\gamma) = \mathbf{O}(\beta)$ under the answer-length cost model. However, the conditions are not sufficient under the union cost model. The union theorem does not hold under the unit cost mode unless the sequence of the type-2 time bounds tends to trivial. We have the following non-union theorem.

For convenience, we define $\|\sigma\| = \sum_{i\in\mathsf{dom}(\sigma)}(|i| + |\sigma(i)| + 2)$ for $\sigma \in \mathcal{F}$ and $\sigma \neq \emptyset$. By convention, let $\|\sigma\| = 2$ for $\sigma = \emptyset$.

Theorem 5 (Non-Union Theorem). *For any $\alpha, \beta \in \mathbf{T_2TB}$, we have*

$$\mathbf{C}^u(\alpha) \neq \bigcup_{c\in\mathbf{N}} \mathbf{C}^u(c\beta).$$

Proof: Unlike the quite involved proof for the union theorem under the answer-length cost model, the proof for the non-union theorem above is rather straightforward. We simply construct a counterexample.

Fix any two $\alpha, \beta \in \mathbf{T_2TB}$ and $(g, a) \in \mathcal{T} \times \mathbf{N}$. Let $\tau \subset g$ and suppose that (τ, a) is a minimal locking fragment of α and $\alpha(\tau, a) = k$. Consider the functional $G^k_{\tau,a}$ defined as follows:

$$G^k_{\tau,a}(f, x) \;=\; \begin{cases} 2^{k+1} & \text{if } x = a, \tau \subset f, \text{ and } f(\mathsf{max}(\mathsf{dom}(\tau)) + 1) \text{ is odd,} \\ 0 & \text{otherwise.} \end{cases}$$

Let d be the least point beyond the locking fragment of α on (g, a), i.e., $d = \mathsf{max}(\mathsf{dom}(\tau)) + 1$.

Claim 1: $G^k_{\tau,a} \notin \mathbf{C}^u(\alpha)$.

For each $i \in \mathbf{N}$, define

$$\sigma_i = \tau \cup \{(d, 2i + 1)\}.$$

For each $i \in \mathbf{N}$ and $f \in \mathcal{T}$, if $\sigma_i \subset f$, then any $\widehat{\varphi}_e$ that computes $G^k_{\tau,a}$ will be clipped by α during the course of the computation on (f, a). Let $X = \bigcup_{i\in\mathbf{N}} (\!(\sigma_i, a)\!)$. Clearly, $X \subseteq E_{e,\alpha}$. Moreover, X is not compact in $\mathbb{T}(\widehat{\varphi}_e)$ because

$$\mathcal{O} = \{(\!(\sigma_0, a)\!), (\!(\sigma_1, a)\!), \ldots\},$$

is an open cover for X without finite subcover. According to Lemma 2 in the full version of [14], since $X \subseteq E_{e,\alpha}$, it follows that $E_{e,\alpha}$ is not compact in $\mathbb{T}(\widehat{\varphi}_e)$. Therefore, $G^k_{\tau,a} \notin \mathbf{C}^u(\alpha)$.

Claim 2: $G^k_{\tau,a} \in \bigcup_{c\in\mathbf{N}} \mathbf{C}^u(c\beta)$.

We construct an OTM, $\widehat{M}^u_e$, that computes $G^k_{\tau,a}$ as follows: On any input $(f,x) \in \mathcal{T} \times \mathbf{N}$, the machine will, (1) scan x and check if $(x = a)$, then, (2) check if $\tau \subset f$, (3) check if $f(d)$ is odd, and (4) output 0 or 2^{k+1} based on the results in (1), (2), and (3).
Under the unit cost model, the cost for an OTM to compare $\tau(i)$ and $f(i)$, where $f \in \mathcal{T}$ and $i \in \mathsf{dom}(\tau)$, is bounded by $c(|i| + |\tau(i)|)$ for some constant $c > 1$, because the OTM only has to scan at most $2 \cdot |\tau(i)|$ many bits to judge if the two are identical. Note that in the case that $|f(i)| > |\tau(i)|$, $\widehat{M}^u_e$ does not have to scan all of $f(i)$. Likewise, the machine can just scan the least significant bit of $f(d)$ to know whether $f(d)$ is odd. Thus, the cost of the computation of $\widehat{M}^u_e$ on any (f,x) is bounded as follows:

$$\widehat{\Phi}^u_e(f,x) \leq c(|x| + \|\tau\| + |d| + k + 1).$$

Since $||\tau|| + |d| + k + 1$ is also a constant, there is a constant $c' \in \mathbf{N}$ such that,

$$\widehat{\Phi}^u_e(f,x) \leq c'|x| + c'.$$

Since β is a type-2 time bound, which is nontrivial, it follows that, for every $(\sigma,x) \in \mathcal{F} \times \mathbf{N}$, $\beta(\sigma,x) \geq |x| + 1$, and thus,

$$c'\beta(\sigma,x) \geq c'|x| + c'.$$

That is, if $\widehat{M}^u_e$ is clocked by $c'\beta$, $\widehat{M}^u_e$ will get enough budget from $c'\beta$ during the course of the computation on every (f,x). Thus, $G^k_{\tau,a} \in \mathbf{C}^u(c'\beta)$.

Therefore, from claims 1 and 2, $G^k_{\tau,a} \in \bigcup_{c\in\mathbf{N}} \mathbf{C}^u(c\beta) - \mathbf{C}^u(\alpha)$, i.e., $\mathbf{C}^u(\alpha) \neq \bigcup_{c\in\mathbf{N}} \mathbf{C}^u(c\beta)$. □

Note that the union in Theorem 5 is a type-2 analog of the big-O notation. Thus, $\mathbf{O}(\beta)$ is not a complexity class under the unit cost model.

Also note that the sequence we proposed above, $\beta, 2\beta, 3\beta, \ldots$, is very conservative in a sense that the sequence is uniformly convergent, i.e., every one in the sequence converges at the same locking fragment, which is a very strong condition. Thus, in order to sustain the union theorem under the unit cost model, the condition must be further strengthened. For example, we may require the value of the sequence to be bounded on any (f,x), i.e., $\lim_{(i\to\infty,\sigma\to f)} \beta_i(\sigma,x) \in \mathbf{N}$. However, we consider such a union theorem trivial.

4.5 Anti-Gap Theorem

When people tried to find an effective operation to enlarge a type-1 complexity class, the gap phenomena were discovered [2, 3, 5, 24]. We've learned that it is impossible to have such effective operation unless some "nice" property is assumed. We state a stronger version of gap theorems known as the Operator Gap Theorem in the following.

Theorem 6. (Constable [5], Young [24]) *Given any total effective operator Θ, we can effectively find an arbitrarily large recursive function t such that $\mathcal{C}(t) = \mathcal{C}(\Theta(t))$.*

In other words, we can always find resource bound t such that the given effective operator fails to enlarge the complexity class determined by t. Some properties such as *time-constructibility* and *honesty* are those commonly mentioned "nice" ones to dismiss the gap phenomena. Three major theorems in classical complexity theory – Compression theorem, Gap theorem, and Honesty theorem – form a wonderful trilogy telling a full story along this line.

In [13] we gave a preliminary idea for type-2 time-constructibility, but we still do not fully understand what should be the meaning of a type-2 honest functional. Under the answer-length cost model, the gap phenomena are inherited from the type-1 computation, i.e., the gap phenomena are caused by the type-1 part of the computation. We observe that no "pure" type-2 computation is possible under the answer-length cost model because every oracle query must be followed by an inevitable type-1 computation (i.e., reading the answer that can go arbitrarily huge). On the other hand, under the unit cost model, the type-2 computation becomes "purer" and the gap phenomena disappear. We see this as a positive result because we can uniformly enlarge a complexity class. The only condition is that, β has to be strong.

Theorem 7 (Anti-Gap Theorem). *Suppose $g : \mathbf{N} \to \mathbf{N}$ is recursive and, for every $x \in \mathbf{N}$, $g(x) \geq 3x$. Then, for every strong $\beta \in \mathbf{T_2TB}$, $\mathbf{C}^u(\beta) \subset \mathbf{C}^u(g \circ \beta)$.*

Proof: We say that (σ, a) is ***β-checkable***, if there is an OTM under the unit cost model, $\widehat{M^u_e}$ such that, on every $(f, x) \in \mathcal{T} \times \mathbf{N}$, $\widehat{\varphi}^u_{e,\beta}(f, x) \Downarrow$, and

$$\widehat{\varphi}^u_{e,\beta}(f, x) = \begin{cases} 1 \text{ if } \sigma \subset f \text{ and } x = a; \\ 0 \text{ otherwise.} \end{cases}$$

Fix a $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$ such that, there is an OTM clocked by β under the unit cost model, $M^u_{e,\beta}$, that can verify (σ, a) but cannot verify any (τ, a) with $\sigma \subset \tau$. I.e., (σ, a) is the biggest β-checkable fragment with respect to some (f, a). Such (σ, a) must exist. Consider the following two cases.

Case 1: $\sigma = \emptyset$. Consider the following functional, $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$,

$$F(f, x) = \begin{cases} 1 \text{ if } x = a \text{ and } f(0) \text{ is odd;} \\ 0 \text{ otherwise.} \end{cases}$$

On input (f, a) where $f \in \mathcal{T}$ and $f(0)$ is odd, the minimum cost of any OTM that computes F is a constant dominated by the size of a. Let the cost be k. Clearly, k is the same as the cost of checking $(\{(0, 1)\}, a)$, because only the least significant bit of the value of $f(0)$ has to be scanned. Thus,

$$k = |a| + 1 + \|\{(0, 1)\}\| = |a| + 5.$$

But, $(\{(0,1)\},a)$ is not β-checkable, so we have

$$|a|+5 > \beta(\{(0,1)\},a) \geq |a|+1.$$

In other words, the budget provided by β is not enough for any OTM, M_e, that computes F to scan the first bit of the value of $f(0)$ in case that $f(0)$ is odd. Thus, $\widehat{\varphi}_{e,\beta}(f,a) \Uparrow$. A similar argument can be used to the case that $f(0)$ is even to conclude that $\widehat{\varphi}_{e,\beta}(f,a) \Uparrow$. Therefore, for each $y \in \mathbf{N}$, $(\!(\{(0,y)\},a)\!) \subset E_{e,\beta}$ and, moreover, $(\!(\{(0,y)\},a)\!)$ is an open set in $\mathbb{T}(\widehat{\varphi}_e)$. It follows that $E_{e,\beta}$ is noncompact in $\mathbb{T}(\widehat{\varphi}_e)$ and hence $F \notin \mathbf{C}^u(\beta)$.
Clearly, if $|a| \geq 2$, then

$$g(\beta(\{(0,1)\},a)) \geq 3(|a|+1) > |a|+5.$$

Thus, we can have an OTM clocked by $g \circ \beta$ that computes F. Hence $F \in \mathbf{C}^u(g \circ \beta)$.

Case 2: $\sigma \neq \emptyset$. Let $n \in \mathbf{N}$ be the least number not in $\mathsf{dom}(\sigma)$. Consider the following functional, $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$,

$$F(f,x) = \begin{cases} 1 \text{ if } x = a, \sigma \subset f, \text{ and } f(n) \text{ is odd;} \\ 0 \text{ otherwise.} \end{cases}$$

On input (f,a) where $f \in \mathcal{T}, \sigma \subset f$, and $f(n)$ is odd, the minimum cost of any OTM that computes F is a constant, k, dominated by $|a|$ and $\|\sigma\|$, and k is the same as the cost of checking $(\sigma \cup \{(n,1)\},a)$, which is

$$k = \|\sigma \cup \{(n,1)\}\| + |a| + 1 = \|\sigma\| + |a| + |n| + 3.$$

By the assumption, (σ,a) is β-checkable and all its extensions are not, and since β is strong $\mathcal{F}$-monotone,

$$\|\sigma\| + |a| + |n| + 3 > \beta(\sigma \cup \{(n,1)\},a) \geq \beta(\sigma,a) \geq \|\sigma\| + |a| + 1.$$

Since n is the least number not in $\mathsf{dom}(\sigma)$, it follows that $|n| \leq \|\sigma\| + 1$. Also, since $|a| \geq 1$, we have

$$g(\beta(\sigma \cup \{(n,1)\},a)) \geq g(\beta(\sigma,a)) \geq 3(\|\sigma\| + |a| + 1) \geq \|\sigma\| + |a| + |n| + 3.$$

Similarly, when $f(n)$ is even. For $(f,x) \in \mathcal{T} \times \mathbf{N}$ and $\sigma \not\subset f$ or $x \neq a$, since (σ,a) is β-checkable, $\widehat{M}^u_e$ clocked by β, on such (f,x), can finish the checking and output 0. Thus, $F \in \mathbf{C}^u(g \circ \beta)$.

What remains to argue is that $F \notin \mathbf{C}^u(\beta)$. Fix any OTM, $\widehat{M}^u_e$, that computes F. Suppose that q is the last query asked among $\mathsf{dom}(\sigma) \cup \{n_0\}$ throughout the course of computation of $\widehat{M}^u_e$ on (f,a) where $\sigma \subset f$. If $q = n$, we can slightly modify the same argument in the previous case to conclude that $F \notin \mathbf{C}^u(\beta)$. If $q \in \mathsf{dom}(\sigma)$, consider τ_y for each $y \in \mathbf{N}$ defined in the following.

$$\tau_y = (\sigma - \{(q,\sigma(q))\}) \cup \{(q, \sigma(q) + y \cdot 2^{|\sigma(q)|}), (n,1)\}.$$

Thus, for every $y \in \mathbf{N}$ and $f \in \mathcal{T}$, if $\tau_y \subset f$, then $\widehat{\varphi}_{e,\beta}(f,a) \Uparrow$. This is because $(\sigma \cup \{(n,1)\}, a)$ is not β-checkable, and after the first $|\sigma(q)|$ bits of the value of $f(q)$ scanned, the budget for $\widehat{M}_e^u$ must be used up. Thus, for every $y \in \mathbf{N}$, $(\!(\tau_y, a)\!) \subset E_{e,\beta}$ and is an open set in $\mathbb{T}(\widehat{\varphi}_e)$. Therefore, $F \notin \mathbf{C}^u(\beta)$. □

Note that Theorem 7 above does not hold if β is not strong (i.e., not $\mathcal{F}$-monotone). An intuitive explanation is that, if β is not strong, then it can shrink the budget to the bottom (i.e., $|x|+1$) until it receives a locking fragment that is too long to be seen under the budget provided by $g \circ \beta$.

5 Properties of Type-2 Time Bounds

In this section we study the relation between type-2 time bounds and type-2 functionals. Due to the properties of type-2 time bounds, each of them determines a limit functional as follows. (More details about limit functionals can be found in Rogers' [20].

Definition 8. *Let $\beta \in \mathbf{T_2TB}$. Define $F_\beta : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ by*

$$F_\beta = \lambda f, x. \left(\lim_{\sigma \to f} \beta(\sigma, x) \right).$$

Some obvious properties of the Limit functional, $]F_\beta$, come directly from the properties of β. For example, F_β is a continuous functional and total on $\mathcal{T} \times \mathbf{N}$. Following the type-2 almost-everywhere relation, $\leq_{\mathbf{2}}^{*}$, defined in [14], we can prove that,

$$\widehat{\varphi}_e \in \mathbf{C}(\beta) \implies \widehat{\Phi}_e \leq_{\mathbf{2}}^{*} F_\beta. \tag{7}$$

However, the converse of (7) is false because the history of requesting budget from β does matter. Thus, $F_{\beta_1} = F_{\beta_2}$ does not imply that $\mathbf{C}(\beta_1) = \mathbf{C}(\beta_2)$, which means that the budget provided by β in the limit may not be useful for the computation. This causes the major difference between complexity classes defined by $\mathbf{T_2TB}$ and type-2 functionals. However, we may want to have a certain computable operation on β's to force the budget in the limit to be used earlier during the computation; in such a way all computations with complexity bounded by F_β can be finished. We argue that such an effective operation is impossible. We state this in the following theorem.

Theorem 8. *There is no recursive operator $\Theta : \mathbf{T_2TB} \to \mathbf{T_2TB}$ such that, for any $\beta \in \mathbf{T_2TB}$, $\widehat{\varphi}_e : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$, and $(f,x) \in \mathcal{T} \times \mathbf{N}$, we have*

$$\widehat{\Phi}_e(f,x) \leq F_\beta(f,x) \iff \widehat{\varphi}_{e,\Theta(\beta)}(f,x) \Downarrow .$$

Proof: We argue that the negation of the theorem will lead to a solution to the halting problem. Define $F : \mathcal{T} \times \mathbf{N} \rightharpoonup \mathbf{N}$ by

$$F(f,x) = \varphi_x(x).$$

Clearly, we can have a $\widehat{\varphi}$-program e for F such that, for each $(f, x) \in \mathcal{T} \times \mathbf{N}$, if $\varphi_x(x) \downarrow$, then

$$\widehat{\Phi}_e(f, x) \leq \Phi_x(x) + |x| + c,$$

where $c \geq 1$ is some constant depending on the OTM. Define $\beta : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$ by

$$\beta(\sigma, k) = \begin{cases} t + |k| + c & \text{if } i \text{ is the least element of } \mathsf{dom}(\sigma), \text{ provided:} \\ & \{0, 1, \ldots, i\} \subseteq \mathsf{dom}(\sigma) \text{ and } \sigma(i) = \langle k, t \rangle \text{ for some } t; \\ |k| + c & \text{if no such } i \text{ exists.} \end{cases}$$

One can check that β is $\mathcal{F}$-monotone, convergent, and nontrivial. Hence $\beta \in \mathbf{T_2TB}$. Let $K = \{x \mid x \in \mathbf{N}, \varphi_x(x) \downarrow\}$ and $k_0, k_1, \ldots$ be a fixed recursive enumeration of K without repetition. For each $i \in \mathbf{N}$, let $t_i = \Phi_{k_i}(k_i)$. Define $\kappa : \mathbf{N} \to \mathbf{N}$ by

$$\kappa(i) = \langle k_i, t_i \rangle. \tag{8}$$

Clearly, κ is total recursive. For any $k \in K$, there exists $i \in \mathbf{N}$ such that, $k = k_i$ and $\kappa(i) = \langle k, \Phi_k(k) \rangle$, and thus,

$$F_\beta(\kappa, k) = \Phi_k(k) + |k| + c.$$

When $k \in K$, we have

$$\widehat{\Phi}_e(\kappa, k) \leq \Phi_k(k) + |k| + c = F_\beta(\kappa, k).$$

By contradiction, suppose a Θ such as in the statement of the theorem exists. From the definition of F, the OTM $\widehat{M}_e$ would never make any oracle query. Therefore, the overall budget provided by $\Theta(\beta)$ for $\widehat{M}_e$ on (κ, k) is $\Theta(\beta)(\emptyset, k)$. From the assumption, if $x \in K$, then the budget $\Theta(\beta)(\emptyset, x)$ must be enough for computing $\widehat{\varphi}_e(\kappa, x)$. Therefore, we can have a type-1 recursive function to decide if any given x is in K as follows.

$$\varphi_p(x) = \begin{cases} 1 & \text{if } \Phi_x(x) \leq \Theta(\beta)(\emptyset, x), \\ 0 & \text{otherwise.} \end{cases}$$

That gives a recursive characteristic function of K, which is impossible. □

Theorem 9. *There is a $\beta \in \mathbf{T_2TB}$ such that F_β is total, but F_β is not computable.*

The original form of the theorem above is due to Poúr-EL [18]. In Rogers' terms [20], the theorem states that not every limit functional corresponds to a recursive functional that is total on recursive functions. Here we omit the proof. Consider the following corollary.

Corollary 1. *There is $\beta \in \mathbf{T_2TB}$ that is not locking detectable.*

It is clear that if β is locking-detectable, then F_β is computable. Therefore, we immediately obtain the corollary. The corollary seems to be a disadvantage of our formulation, but in the next section we shall argue that it is not the case.

5.1 Locking Detectable Type-2 Time Bounds

Locking detectability probably is the most useful property we want to have in our proofs. For example, the proof of Theorem 3 makes such assumption. In this section we argue that making this seemingly strong assumption in fact does not lose the generality of our proofs under both cost models. Consider the following definition.

Definition 9 (Equivalent Type-2 Time Bounds). *We say that β and α are equivalent if the two determine the same complexity class, i.e., $\mathbf{C}(\beta) = \mathbf{C}(\alpha)$.*

Our approach is to construct an effective operator that converts any $\beta \in \mathbf{T_2TB}$ to an equivalent locking detectable one. We state the main theorem first.

Theorem 10. *There is an effective operator $\Theta_L : \mathbf{T_2TB} \to \mathbf{T_2TB}$ such that, for every $\beta \in \mathbf{T_2TB}$, $\Theta_L(\beta)$ is a locking detectable type-2 time bound equivalent to β. Moreover, if β is strong, then so is $\Theta_L(\beta)$.*

Before we prove the theorem, let's fix some definition for our convenience in the arguments.

Definition 10 (β-Inaccessible Points). *Let $\beta \in \mathbf{T_2TB}$ and $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$ with $\sigma \neq \emptyset$. Suppose that $\mathsf{dom}(\sigma) = \{x_0, x_1, x_2, \ldots, x_n\}$ and $x_0 < x_1 < \cdots < x_n$. For each $x_i \in \mathsf{dom}(\sigma)$, we say that x_i is a β-inaccessible point in (σ, a) if*

1. *$i \neq 0$ and x_{i-1} is a β-inaccessible point in (σ, a), or*
2. *$|x_i| > \beta(\sigma', a)$, for every $\sigma' \subseteq \{x_o, x_1, \ldots, x_{i-1}\}$.*

Also, let $\mathsf{IA}_\beta(\sigma, a)$ denote the set of all β-inaccessible points in (σ, a).

For every $\beta \in \mathbf{T_2TB}$ and $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$, it is clear that $\mathsf{IA}_\beta(\sigma, a)$ is computable. However, we observe that, if there are β-inaccessible points in (σ, a), then there is no β-clocked OTM that can query exactly all $\mathsf{dom}(\sigma)$ to the oracle on any (f, a) with $\sigma \subset f$. Intuitively, we obtain a locking detectable type-2 time bound equivalent to β by removing all β-inaccessible points from input (σ, a). Note that we do not require an OTM to make queries in any particular order; hence having β-inaccessible points in (σ, a) does not mean that, if $\sigma \subset \tau$, then (τ, a) also has β-inaccessible points. In other words, we cannot simply lock our new time-2 time bound to the value of $\beta(\sigma', a)$ for some $\sigma' \subseteq \sigma$. We solve this problem by searching the *base segment* defined as follows.

Definition 11 (Base Segments). *Let $\beta \in \mathbf{T_2TB}$ and $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$. For $\tau \subseteq \sigma$, we say that (τ, a) is the **base segment** in (σ, a) with respect to β if*

1. *$\mathsf{dom}(\tau)$ is a finite initial segment of $\mathbf{N}$,*
2. *$\mathsf{IA}_\beta(\tau, a) = \emptyset$, and*
3. *for all $\tau' \supset \tau$, $\mathsf{IA}_\beta(\tau', a) \neq \emptyset$.*

It is clear that, given any $\beta \in \mathbf{T_2TB}$ and $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$, we can effectively decide whether the base segment in (σ, a) exists. Moreover, if the base segment does exist in some (σ, a), then we can effectively obtain it. We state this observation as follows:

Lemma 1. *Let $\beta \in \mathbf{T_2TB}$. Fix any $(f, a) \in \mathcal{T} \times \mathbf{N}$. There exists a unique $\tau \in \mathcal{F}$ with $\tau \subset f$ such that, (τ, a) is the base segment in (σ, a) with respect to β for all $\sigma \in \mathcal{F}$ with $\tau \subseteq \sigma$. Moreover, if $\tau \subset \sigma$, then there are β-inaccessible points in (σ, a).* □

The lemma above is obvious due to the fact that every $\beta \in \mathbf{T_2TB}$ is convergent. Note that, the uniqueness comes from the requirements that a base segment must be an initial segments of some total function.

Proof of Theorem 10: Our proof is based on the following imple idea: Given any $\beta \in \mathbf{T_2TB}$ and $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$, if the base segment in (σ, a) is found, say (τ, a), then we lock the type-2 time bound with value $\beta(\tau, a)$. The construction of such Θ_L is quite straightforward as in Figure 2.

```
Program for Θ_L(β)

        input σ : F, x : N;
        var τ : F;

        find (τ, x) the base segment in (σ, x) with respect to β;
        if such (τ, x) does not exist
            then τ ⟵ {⟨i, σ(i)⟩ | i ∈ dom(σ) and i ∉ IA_β(σ, x)};
        output β(τ, x);

End program
```

Fig. 2. Program for $\Theta_L(\beta)$

We shall argue that $\Theta_L(\beta)$ is computable, nontrivial, bounded, convergent, locking detectable, and $\mathbf{C}(\beta) = \mathbf{C}(\Theta_L(\beta))$. Let $\Theta_L(\beta) = \alpha$.

Claim 1: $\alpha \in \mathbf{T_2TB}$ and is locking detectable. Let $(f, a) \in \mathcal{T} \times \mathbf{N}$.

1. Computability: Since β and IA_β are computable, it follows that the construction shown in Figure 2 is effective.
2. Nontriviality: The value of α is based on the value of β on the same type-0 input. Thus, α is nontrivial as long as β is.
3. Bounddedness: Since every β-inaccessible points in any $\sigma \subset f$ will be removed, there is no β-inaccessible point in $\tau \subset f$. It follows that, for every $\sigma \subset f$,
$$\alpha(\sigma, a) = \beta(\tau, a) \leq \lim_{\eta \to f} \alpha(\eta, a).$$

4. Convergence: Since β itself must be convergent, it is clear that if we keep extending the finite initial segment σ of f and feed (σ, a) to α, eventually $(f_{[n]}, a)$ will contain the *base segment*, and then α will be locked.
5. Locking detectability: Since we can effectively decide whether the base segment in (σ, a) exists, by Lemma 1, we can construct a locking detector for α.

Claim 2: $\mathbf{C}(\beta) = \mathbf{C}(\alpha)$. We shall prove that, for any $\widehat{\varphi}$-program e and $(f, a) \in \mathcal{T} \times \mathbf{N}$,

$$\widehat{\varphi}_{e,\beta}(f, a) \Downarrow \quad \text{if and only if} \quad \widehat{\varphi}_{e,\alpha}(f, a) \Downarrow .$$

Fix a $\widehat{\varphi}$-program, e, and let $(f, a) \in \mathcal{T} \times \mathbf{N}$. Consider the following two cases.

1. Suppose $\widehat{\varphi}_{e,\beta}(f, a) \Uparrow$. From the construction of α, we know that the only situation in which $\alpha(\sigma, a) \leq \beta(\sigma, a)$ must be the case that there are some β-inaccessible points in (σ, a). But if any OTM attempts to query at a β-inaccessible point, the OTM will be clipped down ($\Uparrow$). Therefore,

$$\widehat{\varphi}_{e,\beta}(f, a) \Uparrow \ \Rightarrow \ \ \widehat{\varphi}_{e,\alpha}(f, a) \Uparrow .$$

2. Suppose $\widehat{\varphi}_{e,\beta}(f, a) \Downarrow$. Let $\mathsf{dom}(\sigma_t)$ be the collection of queries made during the period from the beginning of the computation up to the moment t. It is clear that there must not be any β-inaccessible point in (σ_t, a), because otherwise $\widehat{\varphi}_{e,\beta}(f, a) \Uparrow$. Thus, according to the construction of α, we know that $\beta(\sigma_t, a) = \alpha(\sigma_t, a)$. Therefore, at any moment t during the computation, the budget $\alpha(\sigma_t, x)$ is also enough to support the computation. This proves that $\widehat{\varphi}_{e,\alpha}(f, x) \Downarrow$.

For $\mathcal{F}$-monotonicity, let β be $\mathcal{F}$-monotone, we shall argue that $\Theta_L(\beta)$ is also $\mathcal{F}$-monotone. Let β be $\mathcal{F}$-monotone. Fix any $a \in \mathbf{N}$ and $\sigma_1, \sigma_2 \in \mathcal{F}$ with $\sigma_1 \subseteq \sigma_2$. Since β is strong, we have $\beta(\sigma_1, a) \leq \beta(\sigma_2, a)$.

1. Suppose that the base segment, (τ, a), is found in (σ_1, a). By Lemma 1, the same base segment will also be found in (σ_2, a). Thus, $\alpha(\sigma_1, a) = \alpha(\sigma_2, a) = \beta(\tau, a)$.
2. Suppose that the base segment, (τ, a), is found in (σ_2, a) but not found in (σ_1, a). Since the base segment is the maximal segment without β-inaccessible points, thus $[\mathsf{dom}(\sigma_1) - \mathsf{IA}_\beta(\sigma_1, a)] \subset \mathsf{dom}(\tau)$.
3. Suppose that neither (σ_1, a) nor (σ_2, a) contains the base segment. In this case, we have that $[\mathsf{dom}(\sigma_1) - \mathsf{IA}_\beta(\sigma_1, a)] \subseteq [\mathsf{dom}(\sigma_2) - \mathsf{IA}_\beta(\sigma_2, a)]$.

Since β is strong, it follows that $\alpha(\sigma_1, a) \leq \alpha(\sigma_2, a)$ in cases 2 and 3. Therefore, each of the three cases above concludes that $\alpha(\sigma_1, a) \leq \alpha(\sigma_2, a)$, and hence α is $\mathcal{F}$-monotone. □

6 Conclusion and Futures

Type-2 computation to some extent is a better model for many real-world computations. Just to name some: real computation, real time (interactive) computation, mass database inquiries, machine learning, Web search engine, and so on, where we do not use all available information just as the OTM does not use the entire knowledge of its oracle. But as a matter of fact, type-2 complexity theory is a highly underdeveloped area mainly because there is no type-2 counterpart of the Church-Turing Thesis. A tiny difference between computation models can easily cause manifest discrepancy in the notion of computability. The situation becomes even worse when computational complexity is concerned. Even with the OTM, a widely accepted standard for type-2 computation, the way we treat the answers returned from the oracle radically shapes the outlook of the complexity theory at type-2. The answer-length cost is the most common cost model assumed in the literatures. This obviously is not the only choice (we do not read every entry returned from the Google search engine, do we?). We thus propose a reasonable alternative cost model – *unit cost* model. We have learned that even under the same clocking scheme, this cost model gives us a very different type-2 complexity structure. For example, the union theorem and the gap theorem are much more fragile under the unit cost model than under the answer-length cost model.

Along the line of classical complexity theorems, there are apparently many questions yet to be answered. For example, is there any reasonable version of the Speedup theorem? Hierarchy theorem? What is the meaning of complete-problems at type-2? Is the classical notion of *honesty* necessarily trivial at type-2? If yes, what should it be and what problem it is meant to fix? BFF does not fit our notion of complexity classes, but are there any intuitively feasible classes that do? Etc, etc. All these questions should be answered and we speculate that the clocking scheme together with our type-2 time bounds has provided an accessible approach to explore this long missing piece of a more general complexity theory.

References

1. Manuel Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14(2):322–336, 1967.
2. A. Borodin. Complexity classes of recursive functions and the existence of complexity gaps. *Conference Record of ACM Symposium on the Theory of Computing*, pages 67–78, 1969.
3. A. Borodin. Computational complexity and the existence of complexity gaps. *Journal of the ACM*, 19(1):158–174, 1972.
4. Walter S. Brainerd and Landweber Lawrance H. *Theory of Computation*. John Wiley & Sons, New York, 1974.
5. Robert L. Constable. The operator gap. *Journal of the ACM*, 19:175–183, 1972.
6. Robert L. Constable. Type two computational complexity. In *Proceedings of the 5^{th} ACM Symposium on the Theory of Computing*, pages 108–122, 1973.

7. Stephen A. Cook. Computability and complexity of higher type functions. In Y. N. Mpschovakis, editor, *Logic from Computer Science*, pages 51–72. Springer-Verlag, 1991.
8. J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transitions of the American Mathematics Society*, pages 285–306, May 1965.
9. Bruce M. Kapron and Stephen A. Cook. A new characterization of Mehlhorn's polynomial time functionals. *Proceedings of the 32th Annual IEEE Symposium on the Foundations of Computer Science*, pages 342–347, 1991.
10. Bruce M. Kapron and Stephen A. Cook. A new characterization of type 2 feasibility. *SIAM Journal on Computing*, 25:117–132, 1996.
11. Steve C. Kleene. Recursive functionals and quantifies of finite types I. *Transitions of the American Mathematics Society*, 91:1–52, 1959.
12. L.H. Landweber and E.R. Robertson. Recursive properties of abstract complexity classes. *ACM Symposium on the Theory of Complexity*, May 1970.
13. Chung-Chih Li. Type-2 complexity theory. Ph.d. dissertation, Syracuse University, New York, 2001.
14. Chung-Chih Li. Asymptotic behaviors of type-2 algorithms and induced baire topologies. *Proceedings of the Third International Conference on Theoretical Computer Science*, pages 471–484, August 2004.
15. Chung-Chih Li and James S. Royer. On type-2 complexity classes: Preliminary report. *Proceedings of the Third International Workshop on Implicit Computational Complexity*, pages 123–138, May 2001.
16. E. McCreight and A. R. Meyer. Classes of computable functions defined by bounds on computation. *Proceedings of the First ACM Symposium on the Theory of Computing*, pages 79–88, 1969.
17. Elena Pezzoli. On the computational complexity of type two functionals, Proceedings of the Conference for Computer Science Logic '97. In D. van Dalen and M Mezem, editors, *Lecture Notes in Computer Science*, volume 1414, pages 373–388. Springer-Verlag, 1998.
18. Marian B. Poúr-El. A comparison of five "computable" operators. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:325–340, 1960.
19. M.O. Rabin. Degree of difficulty of computing a function and a partial ordering of recursive sets. Technical Report 2, Hebrew University, 1960.
20. Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. First paperback edition published by MIT Press in 1987.
21. James Royer and John Case. *Subrecursive Programming Systems: Complexity & Succinctness*. Birkhäuser, 1994.
22. James S. Royer. Semantics vs. syntax vs. computations: Machine models of type-2 polynomial-time bounded functionals. *Journal of Computer and System Science*, 54:424–436, 1997.
23. Anil Seth. Complexity theory of higher type functionals. Ph.d. dissertation, University of Bombay, 1994.
24. Paul Young. Easy construction in complexity theory: Gap and speed-up theorems. *Proceedings of the American Mathematical Society*, 37(2):555–563, February 1973.