Clocking Type-2 Computation in The Unit Cost Model

Chung-Chih Li

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

Abstract. In [12] we defined a class of functions called Type-2 Time Bounds (henceforth $\mathbf{T_2TB}$) for clocking the Oracle Turing Machine (OTM) in order to capture the long missing notion of complexity classes at type-2. In the present paper we further advance the type-2 complexity theory under our notion of type-2 complexity classes. We have learned that the theory is highly sensitive to how the oracle answers are handled. We present a reasonable alternative called *unit cost* model, and examine how this model shapes the outlook of the type-2 complexity theory. Under the unit cost model we prove two theorems opposite to the classical union theorem and gap theorem. 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*. The existence of such operator allows us to simplify many proofs without loss of generality.[1]

1 Introduction

Let $\langle \varphi_i \rangle_{i \in \mathbf{N}}$ be an *acceptable programming system* and $\langle \Phi_i \rangle_{i \in \mathbf{N}}$ be a *complexity measure* associated to $\langle \varphi_i \rangle_{i \in \mathbf{N}}$, where $\mathbf{N}$ is the set of natural numbers. 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 [16, 15]. For the 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 that the cost of computing φ_i on x is converged to m. In [7] Hartmanis and Stearns gave a precise definition for complexity classes as follows: $C(t) = \{ f \mid \exists i[\varphi_i = f \text{ and } \Phi_i \leq^* t] \}$, where $\Phi_i \leq^* t$ means that the relation, $\Phi_i(x) \leq t(x)$, holds on *all but finitely many* values of x. Within two years, Blum proposed two axioms in [1] 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 (i) $\varphi_i(x) \downarrow$ if and only if $\Phi_i(x) \downarrow$, and (ii) $\Phi_i(x) = m$ 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

[1] We've omitted all detailed proofs due to the space constraints. For a full version: `http://www.itk.ilstu.edu/faculty/chungli/mypapers/Full_UnitCost.pdf`.

any specific machine models. These two landmark papers initiated an important study now known as *abstract complexity theory* in theoretical computer science.

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 [5] 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. In particular, we face a fundamental problem that there is no Church-Turing thesis at type-2. As a result, the notion of asymptotical behavior at type-2 and the way of clocking whatever type-2 computing devices become not quite as intuitive as ordinary type-1 computation. Recently, we introduced a notion of type-2 asymptotical behavior in [11] to catch the idea of its type-1 counterpart – *for all but finitely many.* Using this notion and the clocking scheme with type-2 time bounds proposed in [12], we describe a natural notion of type-2 complexity classes that seems to be a solid ground for type-2 complexity theory to take off. 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 fix some necessary notations. By convention, natural numbers are taken as type-0 objects and functions over natural numbers are type-1 objects. 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 interested only in total functions of type $\mathbf{N} \to \mathbf{N}$ when they are taken as inputs of type-2 functionals. For convenience, let $\mathcal{T}$ denote the set of total functions of type $\mathbf{N} \to \mathbf{N}$ and $\mathcal{P}$ the set of partial functions of type $\mathbf{N} \rightharpoonup \mathbf{N}$. Also, let $\mathcal{F}$ denote the set of *finite* functions, i.e., $\mathcal{F} \subset \mathcal{P}$ and $\sigma \in \mathcal{F}$ if and only if $\mathsf{dom}(\sigma) \subset \mathbf{N}$ and $\mathsf{card}(\sigma) \in \mathbf{N}$. We fix a canonical indexing for $\mathcal{F}$ so we can treat any function in $\mathcal{F}$ as a natural number when it is taken as the input of some type-1 function. Let $\langle \cdot, \cdot \rangle$ be the standard pairing function defined in [15]. Thus, for every $\sigma \in \mathcal{F}$ and $x \in \mathbf{N}$, there is a unique $\langle \sigma, x \rangle \in \mathbf{N}$. Let $|n|$ be the length of the presentation of $n \in \mathbf{N}$. 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. Without loss of generality we restrict type-2 functionals to our standard type $\mathcal{T} \times \mathbf{N} \rightharpoonup \mathbf{N}$. Thus, we can follow the tradition by using the OTM as our standard formalism for type-2 computation. We also fix some necessary conventions for OTM's in the following paragraph.

Oracle Turing Machines: In addition to the standard I/O tape of a TM, an OTM has two extra tapes called query tape and answer tape. The type-0 numerical input is prepared at the beginning of the I/O tape and the type-1 functional input is prepared as a function oracle attached to the machine before the computation begins. During the course of computation, if the OTM needs some value from the function oracle, the OTM have to place the quetion to the query tape and then transit to a special state called query state. Then, the oracle will place the answer to the answer tape in one step; no matter how big the answer might be. As for the classical complexity theory, we fix a programming system $\langle\widehat{\varphi}_i\rangle_{i\in\mathbf{N}}$ associated with a complexity measure $\langle\widehat{\Phi}_i\rangle_{i\in\mathbf{N}}$ for our OTM's. Conventionally, we take the number of steps an OTM performed as our time complexity measure. Note that the steps for the OTM to prepare the query and read the answer are counted as a part of the computational cost.

2 Type-2 Complexity Classes and Time Bounds

Seth followed Hartmanis and Stearns's notion to define type-2 complexity classes in [18] where he proposed two alternatives:

1. Given recursive $t : \mathbf{N} \to \mathbf{N}$, let $\mathit{DTIME}(t)$ denote the set of type-2 functionals such that, for every functional $F \in \mathit{DTIME}(t)$, 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.
2. Given computable $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$, let $\mathit{DTIME}(H)$ denote the set of type-2 functionals such that, for every functional $F \in \mathit{DTIME}(H)$, 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.

The key idea behind Seth's complexity classes is directly lifted from [7]. In fact, the same idea can also be found in other works such as [8, 17] along the line of machine characterizations. However, we face some problems that do not exist in type-1 computation. For example, in one of Seth's definitions, the resource bound is determined by the set of all answers returned from the oracle; but this set in general is not computable and hence it can't be available before the computation completes. Alternatively, we should update the resource bound dynamically upon each answer returned from the oracle during the course of the computation. But if we do so, there is no guarantee that a clocked OTM must be total. For example, Cook's POTM [6] is an OTM bounded by a polynomial in this manner but a POTM may run forever. Kapron and Cook's proposed their remedies in the context of feasible functionals in [8] and gave a very neat characterizations of type-2 Basic Feasible Functionals (BFF), where the so-called second-ordered polynomial is used as the resource bound. We may adapt all these ideas with our $\leq_2^*$ defined in [11] and extend the second-ordered polynomial to a general type-2 computable functional to have the following complexity class:

$$\mathit{DTIME}(H) = \{F \mid \exists e[\widehat{\varphi}_e = F \text{ and } \widehat{\Phi}_e \leq_2^* H]\}. \tag{1}$$

$\mathit{DTIME}(H)$ seems to be a perfect analog of classical $DTIME$. However, we do not think using a type-2 functional as a resource bound is proper because the bound should not depend on information that is irrelevant to the computation. In other words, we prefer the clocking scheme of POTM. To avoid the problem of POTM we mentioned, we give a class of functions called Type-2 Time Bounds denoted by $\mathbf{T_2TB}$ in order to properly clock OTMs [12].

Definition 1 (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 $\beta(\sigma, a) \leq \beta(\tau, a)$.*

If β is computable, nontrivial, bounded, and convergent, we say that β is a type-2 time bound. Moreover, if β is $\mathcal{F}$-monotone, we say that β is strong.[2]

The properties listed in Definition 1 are formulated so to catch our intuition about what a resource bound should be in clocking the OTM (see the full version for details). By a standard diagonalization, one can prove that $\mathbf{T_2TB}$ is not *recursively enumerable*. This indeed is an uneasy fact, since being able to enumerate $\mathbf{T_2TB}$ is a property that can make many proofs possible or easier. Let $\beta(\sigma, x)\downarrow$ denote the situation that $\forall \tau \supseteq \sigma[\beta(\sigma, a) = \beta(\tau, a)]$. If $\beta(\sigma, x)\downarrow$, we say that (σ, x) is a *locking fragment* of β.

Definition 2 (Locking Detectors). *Let $\beta \in \mathbf{T_2TB}$. We say that ℓ is a locking detector of β if $\ell : \mathcal{F} \times \mathbf{N} \to \{0, 1\}$ is computable and (i) ℓ is $\mathcal{F}$-monotone, (ii) $\forall(\sigma, x) \in \mathcal{F} \times \mathbf{N}[(\ell(\sigma, x) = 1) \Rightarrow \beta(\sigma, x)\downarrow]$, and (iii) $\forall(f, x) \in \mathcal{T} \times \mathbf{N}[\lim_{\sigma \to f} \ell(\sigma, x) = 1]$.*

If $\beta \in \mathbf{T_2TB}$ has a locking detector ℓ, we say that β is locking detectable. If $\ell(\sigma, x) = 1$, then (σ, x) is a *locking fragment* of β. It is clear that whether a given β on some (σ, x) has converged is undecidable. Thus, we cannot simply assume that every type-2 time bound is locking detectable. Nevertheless, we have a very positive theorem allowing us to make that assumption without loss of generality (see Theorem 9 in Section 5).

3 A Clocking Scheme and Two Cost Models

We present a clocking scheme using our $\mathbf{T_2TB}$. This scheme is used implicitly in some works such as Kapron and Cook's [8], Seth's [18], and Royer's [17].

Definition 3 (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 $\widehat{M}_e$ is simulated by the procedure shown in Figure 1. Such a clocked OTM is denoted by $\widehat{M}_{e,\beta}$ and the functional computed by $\widehat{M}_{e,\beta}$ is denoted by $\widehat{\varphi}_{e,\beta}$.*

Consider the procedure in Figure 1. The *budget* provided by β is computed upon every answer returned from the oracle during the course of the simulation

[2] In [12] 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.

```
Program for Clocked OTM $\widehat{M}_{e,\beta}$ :
        input $(f, x) \in \mathcal{T} \times \mathbf{N}$;
        var $\sigma \in \mathcal{F}$; $q, y, expense, budget \in \mathbf{N}$;          /* variable declaration */
        $\sigma \longleftarrow \emptyset$; $expense \longleftarrow 0$; $budget \longleftarrow \beta(\sigma, x)$;          /* initialization */
        Simulate $\widehat{M}_e$ on $(f, x)$ step by step and upon each step completed do:
            $expense \longleftarrow expense + 1$;
            if ($expense > budget$)                                   /* check budge */
                then output $\perp$ and stop;          /* $\perp$ is the bottom symbol. ($\Uparrow$) */
            if ($\widehat{M}_e$ halts with the output $y$)
                then output $y$ and stop;              /* simulation completed. ($\Downarrow$) */
            if (the step just simulated completes an oracle query)
                then do
                    $q \longleftarrow$ current query;
                    $\sigma \longleftarrow \sigma \cup \{(q, f(q))\}$;          /* update query-answer set */
                    $budget \longleftarrow \beta(\sigma, x)$;                    /* update budget */
                end-do;
        Resume the simulation;
End program
```

Fig. 1. A Clocking Scheme for OTM's

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) denoted by $\widehat{\varphi}_{e,\beta}(f, a) \Uparrow$. On the other hand, if the simulation reaches the line marked ($\Downarrow$), which means that the simulation of $\widehat{M}_e$ on (f, a) is successfully completed, then we say that $\widehat{\varphi}_{e,\beta}(f, a)$ converges to value $\widehat{\varphi}_e(f, a)$. We denote this situation by $\widehat{\varphi}_{e,\beta}(f, a) \Downarrow$. 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 $\beta \in \mathbf{T}_2\mathbf{TB}$, $\widehat{\varphi}_{e,\beta}$ is a total computable functional of type $\mathcal{T} \times \mathbf{N} \to \mathbf{N}$. This removes the problem of POTM.

Theorem 1. *Given any computable $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ and $\beta \in \mathbf{T}_2\mathbf{TB}$, there is a $\widehat{\varphi}$-program e for F such that $\widehat{\varphi}_e \neq \widehat{\varphi}_{e,\beta}$.*

The theorem above show that arbitrarily complex $\widehat{\varphi}$-programs exist. The proof is an easy application 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. A slightly more involved theorem is the type-2 version of Rabin Theorem [14] stating that, given any $\beta \in \mathbf{T}_2\mathbf{TB}$, there is a 0-1 valued computable functional that cannot be computed by any $\widehat{\varphi}_{e,\beta}$. We prove a versions in [11] where the bound function is simply a computable type-2 functional. The proof is perfectly valid under the present clocking scheme with $\mathbf{T}_2\mathbf{TB}$.

Unfortunately, the properties of $\beta \in \mathbf{T}_2\mathbf{TB}$ and our intuitive clocking scheme are not sufficient to standardize a framework for type-2 complexity theory. The way an OTM handles the oracle answers does matter. We have the following two conventions under our clocking scheme.

Definition 4 (Two Cost Models for OTM's).

1. *Answer-Length Cost 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: The machine needs not to read any bit of the oracle answer unless the machine decides to do so.*

In other words, the cost for each answer returned from the oracle under the answer-length cost model is one unit step plus the length of the answer, while the other model is one. The underlying cost model used in [8, 17, 18] are the answer-length cost model. Also, the outline of a type-2 complexity theory given in [12] is also based on the answer-length cost model. The answer-length cost model from many aspects is more manageable. Nevertheless, we do not think the unit cost model is merely a peculiar convention. On the contrary, the unit cost model is rather reasonable in real computation. For example, only the first bit of the answer is needed to decide whether it is odd or even. 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 trick makes the complexity theory under the unit cost model much flatter than the theory under the other model. For example, there exist certain versions of Union Theorems [12] and Gap Theorems [10] under the answer-length cost model, but the theorems fail to hold under the unit cost model.

4 Complexity Theory under The Unit Cost Model

We explicitly make the notations for the unit cost model different 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 [10–12] and alter the cost model to the unit cost model. The *exception set*, $E^u_{e,\beta}$ is defined as follows:

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

We also adopt the topology introduced in [11], i.e., for every continuous functional $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$, $\mathbb{T}(F)$ is the topology obtained by taking the set of total extensions of every *minimum* locking fragment of F as a basic open set. 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.

Definition 5 (Type-2 Complexity Classes). *Let $\beta \in \mathbf{T_2TB}$. Define the set of computable type-2 functionals $\mathbf{C}^u(\beta)$ as*

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

Inclusion property: In [11] 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 clocking scheme and type-2 time bounds can easily fix this problem. This adds another reason to why we do not think using type-2 functionals as resource bounds is appropriate. Note that the theorem below holds under both cost models.

Theorem 2. *For every $\beta_1, \beta_2 \in \mathbf{T_2TB}$, $[\beta_1 \leq \beta_2] \implies [\mathbf{C}(\beta_1) \subseteq \mathbf{C}(\beta_2)]$.*

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 indeed is correct under the unit cost model, i.e., if $F \in \mathbf{C}^u(\beta)$, then there exist $c, e \in \mathbf{N}$ such that $\widehat{\varphi}^u_e = F$ and $E^u_{e,\beta+c} = \emptyset$. However, under the answer-length cost model, we need more than a constant extra budge as shown in [10]: 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.

Enumerability: It is easy to show that the finite invariant closure of a type-1 complexity class is *recursively enumerable* [2]. 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 [2, 9]. On the other hand, if t is big enough to bound all *finite support functions*[3] 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 determined by t (see [3], section 9.4). Although we required every $\beta \in \mathbf{T_2TB}$ to be nontrivial, this requirement is not sufficient for enumerating $\mathbf{C}(\beta)$. The 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, which may cause a huge returned answer and the OTM will use up its budget in scanning the entire answer as required under the answer-length cost model. In [10] we imposed two rather strong conditions to have $\mathbf{C}(\beta)$ being recursively enumerable. We also conjecture that there exists $\beta \in \mathbf{T_2TB}$ such that $\mathbf{C}(\beta)$ is not recursively enumerable. On the contrary, the cost of querying the oracle is more manageable under the unit cost model. As a result, we have the following theorem without any extra condition on β needed.

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

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

Non-Union Theorem: The Union Theorem [13] is one of the most fascinating theorems in classical complexity theory. Not just because the technique used in the proof then was new to complexity theorists, but also the theorem told us that most *natural* complexity classes have clear boundaries in terms of the bounds that determine Hartmanis and Stearns's complexity classes. In other words, we can use one computable function to exactly bound any given natural complexity class. Although any arbitrary union of computable functions is not necessarily a complexity class in general, we only need a very weak condition to have the following theorem known as the union theorem.

Theorem 4 (McCreight & Meyer [13]). *Let the sequence of recursive functions, f_0, f_1, f_2, ..., be recursive and $f_i(x) \le f_{i+1}(x)$ 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 determined 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 [10]. Nevertheless, in [12, 10] 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 $\beta \in \mathbf{T}_2\mathbf{TB}$ there exists $\gamma \in \mathbf{T}_2\mathbf{TB}$ 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.

Theorem 5 (Non-Union Theorem). *For any $\beta \in \mathbf{T}_2\mathbf{TB}$, there is no $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^u(\alpha) = \bigcup_{c \in \mathbf{N}} \mathbf{C}^u(c\beta)$.*

Thus, $\mathbf{O}(\beta)$ is not a complexity class under the unit cost model. 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, the condition must be further strengthened if we want to sustain the union theorem under the unit cost model. 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 a union theorem under such strong condition trivial.

Anti-Gap Theorem: When people tried to find an effective operation to enlarge a type-1 complexity class, the gap phenomena were discovered [2, 4]. We have 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 [4] & Young [19]). *For 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 [10] we gave a preliminary idea for type-2 time-constructibility, but we still do not fully understand what should be the proper meaning of *type-2 honest functionals*. 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 $\beta \in \mathbf{T}_2\mathbf{TB}$, $\mathbf{C}^u(\beta) \subset \mathbf{C}^u(g \circ \beta)$.*

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. It is clear that each type-2 time bound determines a limit functional as follows. (More details about limit functionals can be found in Rogers' [15]).

Definition 6. *Given any $\beta \in \mathbf{T}_2\mathbf{TB}$, define $F_\beta = \lambda f, x.(\lim_{\sigma \to f} \beta(\sigma, x))$.*

Some obvious properties of this limit functional, F_β, come directly from the properties of β. For example, F_β is a continuous functional and total on $\mathcal{T} \times \mathbf{N}$. Taking the type-2 almost everywhere relation, $\leq_{\mathbf{2}}^{*}$, defined in [11], we can prove that, $\widehat{\varphi}_e \in \mathbf{C}(\beta) \Rightarrow \widehat{\Phi}_e \leq_{\mathbf{2}}^{*} F_\beta$. However, the converse 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 $\mathbf{T}_2\mathbf{TB}$ 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 $\beta \in \mathbf{T_2TB}$ and $\widehat{\varphi}_e : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$, $\widehat{\Phi}_e(f,x) \leq F_\beta(f,x) \Leftrightarrow \widehat{\varphi}_{e,\Theta(\beta)}(f,x) \Downarrow$.*

We know that not every limit functional corresponds to a recursive functional that is total on recursive functions [15]. It is clear that if β is locking-detectable, then F_β is computable. Thus, there is a $\beta \in \mathbf{T_2TB}$ such that F_β is total but not computable. We obtain the following corollary.

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

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. We argue that making this seemingly strong assumption in fact does not lose the generality of our proofs under both cost models. 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 theorem as follows.

Theorem 9. *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 $\Theta_L(\beta)$.*

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 the oracle. But as a matter of fact, type-2 complexity theory is a highly underdeveloped area mainly because 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 model called *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. The complexity theory is much more fragile under the unit cost model.

There are apparently many questions yet to be answered along the line of classical complexity theorems. 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. Computational complexity and the existence of complexity gaps. *Journal of the ACM*, 19(1):158–174, 1972.
3. Walter S. Brainerd and Landweber Lawrance H. *Theory of Computation*. John Wiley & Sons, New York, 1974.
4. Robert L. Constable. The operator gap. *Journal of the ACM*, 19:175–183, 1972.
5. Robert L. Constable. Type two computational complexity. In *Proceedings of the 5th ACM Symposium on the Theory of Computing*, pages 108–122, 1973.
6. 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.
7. J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transitions of the American Mathematics Society*, pages 285–306, May 1965.
8. Bruce M. Kapron and Stephen A. Cook. A new characterization of type 2 feasibility. *SIAM Journal on Computing*, 25:117–132, 1996.
9. L.H. Landweber and E.R. Robertson. Recursive properties of abstract complexity classes. *ACM Symposium on the Theory of Complexity*, May 1970.
10. Chung-Chih Li. Type-2 complexity theory. Ph.d. dissertation, Syracuse University, New York, 2001.
11. 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.
12. 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.
13. 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.
14. M.O. Rabin. Degree of difficulty of computing a function and a partial ordering of recursive sets. Technical Report 2, Hebrew University, 1960.
15. Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. First paperback edition published by MIT Press in 1987.
16. James Royer and John Case. *Subrecursive Programming Systems: Complexity & Succinctness*. Birkhäuser, 1994.
17. 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.
18. Anil Seth. Complexity theory of higher type functionals. Ph.d. dissertation, University of Bombay, 1994.
19. Paul Young. Easy construction in complexity theory: Gap and speed-up theorems. *Proceedings of the American Mathematical Society*, 37(2):555–563, February 1973.