Union Theorems in Type-2 Computation (full version draft)

Chung-Chih Li

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

Abstract. The union theorem [12] indicates that, informally, almost all natural complexity classes at type-1 such as PTIME, PSAPCE, EXPTIME, EXPSPACE, and so on, fit the precise definition of complexity classes given by Hartmanis and Stearns in [3]. In other words, according to the theorem, the rigorous definition of complexity classes in terms of computable resource bounds is indeed broad enough to encompass most natural complexity classes. However, when we lift the computation to type-2 using oracle Turing machines, the union theorem doesn't hold without further strengthening some necessary conditions. In [8] we prove a non-union theorem under a less considered cost model known as unit-cost model. In this paper, we emphasize on a more popular cost model known as answer-length-cost model and give a full treatment of this powerful theorem at type-2. We prove and disprove several nontrivial variations of the union theorem based on our framework.

1 Introduction

Let $\mathbf{N}$ be the set of natural numbers. By computable we mean Turing machine computable. A function is said to be recursive if it is total and computable. Let $\mathcal{R}$ denote the set of recursive functions. We use φ_e to denote the function computed by the e^{th} Turing machine. Thus, when we say the computation of φ_e, we refer it to the computation of the e^{th} Turing machine. Let Φ_e denote Blum's complexity measure [1] associated with the computation of φ_e. Clearly, there are infinitely many different Turing machines that compute the same function, φ_e, with different complexity. In their seminal paper [3], Hartmanis and Stearns give a precise definition of complexity classes as follows. For each $t \in \mathcal{R}$, the complexity class $\mathcal{C}(t)$ is defined by:

$$\mathcal{C}(t) = \left\{ f \in \mathcal{R} \,\middle|\, \exists e \left[\varphi_e = f \text{ and } \overset{\infty}{\forall} x \left(\Phi_e(x) \le t(|x|) \right) \right] \right\}, \tag{1}$$

where $\overset{\infty}{\forall} x$ is understood as "*for all but finitely many*" and $|x|$ is the length of the bit representation of $x \in \mathbf{N}$. Despite the fact that the definition in (1) has provided a solid foundation for the study of complexity theory, we prefer to characterize computational complexity classes according to the properties of the resources bounds, not just to name a class by a single function t as shown in (1). For example, PTIME is a complexity class in which every problem can be solved by some Turing machine within a number of steps bounded by some polynomial. In other words, "being polynomial" is the property required for the time-bounds for all problems in PTIME. Therefore, we define,

$$\text{PTIME} \;=\; \left\{ f \;\middle|\; f \in \text{DTIME}(p) \text{ and } p \text{ is a polynomial} \right\},$$

where DTIME(p) follows the definition in (1). Thus, we could better understand PTIME as follows:

$$\text{PTIME} \;=\; \bigcup_{k \in \mathbf{N}} \text{DTIME}(n^k). \tag{2}$$

Clearly, the union set in (2) gives us a more intuitive idea about what PTIME is. However, it is not at all obvious that PTIME is indeed a complexity class under the formal definition in (1). Is there a recursive function that determines exactly the same complexity class, PTIME? The same question can be asked elsewhere, e.g., the big-O notation in algorithm analysis, which can be understood as

$$O(f) = \bigcup_{k \in \mathbf{N}} \text{DTIME}(k \cdot f).$$

Is $O(f)$ a rigorously defined complexity class? The powerful union theorem provides a positive answer to this kind of questions we just asked. The theorem is proven by McCreight and Meyer [12], which is the first theorem in complexity theory proven by using a priority argument with finite injuries.

Theorem 1 (The Union Theorem [12]). *Given any sequence of recursive functions $f_0, f_1, f_2, \ldots$ such that,*

(i) $\lambda i, x. f_i(x)$ is recursive, and
(ii) for all $i, x \in \mathbf{N}$, $f_i(x) \leq f_{i+1}(x)$,

there is a recursive function g such that $\mathcal{C}(g) = \bigcup_{i \in \mathbf{N}} \mathcal{C}(f_i)$. ■

According to the union theorem, there is $g \in \mathcal{R}$ such that DTIME(g) = PTIME. Likewise, we can apply the theorem to $O(f)$, PSPACE, EXPTIME, etc. and claim that they are indeed complexity classes. Clearly, not any arbitrary collection of resource bounds satisfied the two conditions (i) and (ii) in the union theorem. For example, there is no such uniformly effective enumeration that can cover all computable bounds. Thus, we cannot use the union theorem to argue that the class of recursive functions is a complexity class. In fact, Blum [1] proves that given any $t \in \mathcal{R}$, there always exists a recursive function g such that, $g \notin$ DTIME(t). The simplicity of the two premises required in the union theorem above allows us to apply the theorem to most natural complexity classes. However, we shall argue that we cannot expect the same simplicity at type-2.

The most widely studied type-2 "complexity class" is $\mathbf{BFF}_2$ (Basic Feasible Functional at type-2). With Cook and Kapron's second-order polynomials [2, 5, 6], $\mathbf{BFF}_2$ seems to be a natural type-2 analog of PTIME. Is $\mathbf{BFF}_2$ a type-2 complexity class under some notion similar to (1)? Unfortunately, since there is no generally accepted machine model for type-2 complexity theory, we are not able to answer this question without a reasonable and workable framework to begin with. The framework must include the selection of computing formalism (i.e., an abstract machine such as the oracle Turing machine), the cost model for such machines, type-2 asymptotical notations, type-2 complexity measures, time bounds and a clocking scheme, and a precise definition of type-2 complexity classes. In the following section we shall give necessary terminology and notation in order to describe our union theorems at type-2. Details about our framework and concerns are discussed in [7–11]. Also, detailed proofs of our theorems in this paper are given in Appendix C for verification.

2 Necessary Background for Type-2 Complexity Classes

We will try to keep our notation minimal due to the space constraints. A complete list of our notations can be found in Appendix A. Let $\mathcal{F}$ and $\mathcal{T}$ denote the set of finite functions and total functions, respectively, over $\mathbf{N}$. With a fixed coding method for $\mathcal{F}$, we can assume that $\mathcal{F} \subset \mathbf{N}$ and treat any finite function as a natural number. Let $\sigma \in \mathcal{F}$. We use $\sigma \subset f$ to denote that f is an extension of σ. In [8] we define $\mathbf{T}_2\mathbf{TB}$ (Type-2 Time Bounds) as a class of functions to be used as time bounds for clocking OTM (Oracle Turing Machines). OTM is considered as our formal computing device for type-2 computation[1]. For convenience, we repeat the definition of $\mathbf{T}_2\mathbf{TB}$ in the following.

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 $\beta(\sigma, a) = \beta(\tau, a)$; We use $\beta(\sigma, a) \downarrow$ to denote that β converges at (σ, 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. ■

With an appropriate clocking scheme, a precise notion of type-2 complexity classes can be given. Recall from the classical complexity theory, the constructibility property imposed on resource bounds guarantees a basic hierarchy

[1] We refer the reader to Appendix B or [8] for details about OTM, a clocking scheme, and the two cost models. More properties of $\mathbf{T}_2\mathbf{TB}$ are discussed in [8]. The topology and its compact sets used in the definition of type-2 complexity classes in terms of resource bounds can be found in [7]

among classes (see [13], pages 68, 82-85). Intuitively, a time constructible function is an efficiently computable function that is large enough to be used as a time bound for some Turing machines to operate. The classical definition of constructibility is rather intuitive and straightforward. This, however, is not the case at type-2. Much of the difficulty is caused by the cost of making oracle queries and reading the answers returned from the oracle. In other words, making queries and taking answers may used up the resource granted by the resource bound. Note that, at type-1, the union theorem has no concern about constructibility. But at type-2, without a reasonable notion of constructibility, we can use a trivial counterexample to disprove the union theorem. Moreover, under the unit-cost OTM model, a rather strong non-union theorem can be proven (Theorem 5 in [8]) where the OTM is not required to read every bit of the oracle answers. In this paper, we will emphasize on the answer-length-cost model, which is a cost model that requires the OTM to read every bit of the answer returned from the OTM. However, we have to distinguish the two models in some definitions and theorems. If it is necessary, we use OTM^a (M^a_e) and OTM^u (M^u_e) to denote the OTM (with index e) under answer-length-cost model and unit-cost model, respectively. Similarly, for a result obtained based on a certain cost model, we use an "a" or "u" in superscription to indicate the concerned model.

In the following discussion, we will give some notions that are similar to the classical notion of time constructibility. However, we hesitate to consider any of these notions a type-2 analog of time-constructibility because at the present moment it is not clear how do these notions affect the time-hierarchy at type-2; they just serve the purpose of obtaining a reasonable union theorem at type-2. We first rule out those type-2 time bounds that are too small for any OTM to make queries. To successfully query $f(q)$, an OTM^u needs at least $|q|+1$ steps to place q onto the query tape, whereas an OTM^a needs another $|f(q)|+1$ steps to read the answer, $f(q)$. Let $\|\mathsf{dom}(\sigma)\| = \sum_{i \in \mathsf{dom}(\sigma)}(|i|+1)$. Therefore, $\|\mathsf{dom}(\sigma)\|$ is the minimum number of steps an OTM^u needs to query the entire domain of σ. We abuse the notation by $\|\sigma\| = \sum_{i \in \mathsf{dom}(\sigma)}(|i| + |\sigma(i)| + 2)$. Thus, $\|\sigma\|$ is the minimum number of steps for an OTM^a to query the entire domain of σ and read their answers. Let $M^u_{e,\beta}$ denote the machine obtained from clocking M^u_e with $\beta \in \mathbf{T}_2\mathbf{TB}$, and let $\varphi^u_{e,\beta}$ be the functional computed by $M^u_{e,\beta}$ (same as $M^u_{e,\beta}$ and $\varphi^u_{e,\beta}$). Moreover, let $\varphi^u_{e,\beta}(f,x) \Downarrow$ denote that the computation of $M^u_{e,\beta}(f,x)$ terminates and the value is the same as $\varphi^u_e(f,x)$. In other words, $\varphi^u_{e,\beta}(f,x) \Downarrow$ means $M^u_e(f,x)$ can finish its computation under β.

Definition 2 *Let $\beta \in \mathbf{T}_2\mathbf{TB}$ and $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$.*

1. *We say that (σ, a) is **β-queriable**, if there is $M^u_{e,\beta}$, such that on every $(f, a) \in \mathcal{T} \times \mathbf{N}$ with $\sigma \subset f$, $M^u_{e,\beta}$ can successfully query $\mathsf{dom}(\sigma)$ in some order. We say that (σ, a) is **β-queriable witnessed** by OTM^u M^u_e.*
2. *We say that (σ, a) is **β-checkable**, if there is $M^u_{e,\beta}$, such that, on every $(f, x) \in \mathcal{T} \times \mathbf{N}$, $\varphi^u_{e,\beta}(f,x) \Downarrow$, and*

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

*We say that (σ, a) is **β-checkable witnessed** by OTM^u M_e^u.* ■

Since every $\beta \in \mathbf{T}_2\mathbf{TB}$ must be convergent, it is clear that that not every $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$ is β-checkable or β-queriable. Suppose that (σ, a) is β-queriable witnessed by M_e^u. Although $M_{e,\beta}^u$ can gain budget by simply querying $\mathsf{dom}(\sigma)$, the budget however is based on information of σ. Thus, not for every $\tau \in \mathcal{F}$ with $\mathsf{dom}(\sigma) = \mathsf{dom}(\tau)$, (τ, a) is also β-queriable witnessed by some OTM^u. For a β-queriable (σ, a), β will provide enough budget for an OTM^u to print out $\mathsf{dom}(\sigma)$ in some order, but may not be enough for any OTM^a to do the same. If (σ, a) is β-checkable, then σ can be printed in some order by a β-clocked OTM^a on every (f, a) with $\sigma \subset f$. We further define two properties in the following with which the time bounds are more useful for our purposes.

Definition 3 *Let $\beta \in \mathbf{T}_2\mathbf{TB}$.*

1. *We say that β is **accessible** if and only if there is an OTM^u M_e^u such that, all minimal locking fragments of β are β-queriable witnessed by M_e^u.*
2. *We say that β is **useful** if and only if there is an OTM^u M_e^u such that, all minimal locking fragments of β are β-checkable witnessed by M_e^u.* ■

Clearly, if β is useful, then it is also accessible. The reason we want β to be useful is as follows. Suppose OTM M_e can be computed under β. If β is useful, then we can patch e on some finitely many (τ, a) under the same budget provided by β as long as (τ, a) is not a locking fragment of β. We will see why we need this later. We say that β is locking detectable if there is a computable function to decide whether β will converge on (σ, x). A locking detectable β is not necessarily useful or accessible. For the inverse, we have the following lemma.

Lemma 1. *If $\beta \in \mathbf{T}_2\mathbf{TB}$ is accessible, then β is locking detectable.* ■

One can easily verify the following properties. We omit the proofs. Note that, the inverse of properties in 3 and 4 do not hold, because the set of the minimal locking fragments of β may not be recursively enumerable.

Properties: Let $\beta \in \mathbf{T}_2\mathbf{TB}$.

1. For every $a \in \mathbf{N}$, $(\emptyset, a)$ is β-checkable.
2. For every $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$, if (σ, a) is β-checkable, then $\beta(\sigma, a) \geq \|\sigma\| + |a| + 1$.
3. If β is accessible, then for every minimal locking fragment, (σ, a), of β, we have $\forall \tau \subseteq \sigma \, [\beta(\tau, a) \geq \|\mathsf{dom}(\tau)\| + |a| + 1]$.
4. If β is useful, then for every minimal locking fragment, (σ, a), of β, we have $\forall \tau \subseteq \sigma \, [\beta(\tau, a) \geq \|\tau\| + |a| + 1]$. ■

Let $\beta_1 \leq \beta_2$ denote that, for every $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$, $\beta_1(\sigma, x) \leq \beta_2(\sigma, x)$. Now, we are in position to define the properties of a sequence of type-2 time bounds required in the union theorems.

Definition 4 *Let $\langle \beta_i \rangle$ denote a sequence of type-2 time bounds $\beta_0, \beta_1, \beta_2, \ldots$.*

1. *We say that $\langle \beta_i \rangle$ is **uniform** if and only if $\lambda i, \sigma, x.\beta_i(\sigma, x)$ is recursive.*
2. *We say that $\langle \beta_i \rangle$ is **ascending** if and only if, for all $i \in \mathbf{N}$, $\beta_i \leq \beta_{i+1}$.*
3. *We say that $\langle \beta_i \rangle$ is **useful** if and only if, for all $i \in \mathbf{N}$, β_i is useful.*
4. *We say that $\langle \beta_i \rangle$ is **convergent** if and only if, for every $(f, x) \in \mathcal{T} \times \mathbf{N}$, there is a $\sigma \subset f$ such that, $\beta_i(\sigma, x) \downarrow$ for every $i \in \mathbf{N}$.*
5. *We say that $\langle \beta_i \rangle$ is **uniformly convergent** if and only if, for every $n \in \mathbf{N}$ and $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$, if $\beta_n(\sigma, x) \downarrow$, then for all $i \in \mathbf{N}$, $\beta_i(\sigma, x) \downarrow$.*
6. *We say that $\langle \beta_i \rangle$ is **strongly convergent** if and only if $\langle \beta_i \rangle$ is uniformly convergent and there is a recursive locking detector for β_0.* ■

Let $\langle \beta_i \rangle$ be strongly convergent and let ℓ be a locking detector for β_0. By definition, $\langle \beta_i \rangle$ is uniformly convergent. Thus, we can use ℓ to detect the convergence of the entire sequence. That is,

$$[\ell(\sigma, x) = 1] \implies \forall i \in \mathbf{N}[\beta_i(\sigma, x) \downarrow],$$

and, for all $(f, x) \in \mathcal{T} \times \mathbf{N}$, $\lim_{\sigma \to f} \ell(\sigma, x) = 1$.

Examples: For every $i \in \mathbf{N}$ and $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$, define

$$\alpha_i(\sigma, x) \ = \ \begin{cases} \sigma(x) + |x|^{i+1} + 1 & \text{if } x \in \mathsf{dom}(\sigma); \\ |x|^{i+1} + 1 & \text{otherwise.} \end{cases}$$

$$\beta_i(\sigma, x) \ = \ \begin{cases} \sigma(x+i) + |x|^{i+1} + 1 & \text{if } (x+i) \in \mathsf{dom}(\sigma); \\ |x|^{i+1} + 1 & \text{otherwise.} \end{cases}$$

One can see that $\langle \alpha_i \rangle$ and $\langle \beta_i \rangle$ above are two sequences of type-2 time bounds. Clearly, $\langle \alpha_i \rangle$ is *uniform*, *ascending*, and *strongly convergent*, while $\langle \beta_i \rangle$ is *uniform*, but neither *ascending* nor *convergent*. Moreover, all type-2 time bounds in $\langle \alpha_i \rangle$ and $\langle \beta_i \rangle$ are useful. ■

Let F_β denote the limit functional determined by $\beta \in \mathbf{T}_2\mathbf{TB}$. That is, for every $(f, x) \in \mathcal{T} \times \mathbf{N}$, $F_\beta(f, x) = \lim_{\sigma \to f} \beta(\sigma, x)$.

Lemma 2. *Given uniform and ascending $\langle \beta_i \rangle$, if there exists a total continuous functional $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ such that, for every $i \in \mathbf{N}$, $F_{\beta_i} \leq H$, then $\langle \beta_i \rangle$ is convergent.* ■

Given any $\beta \in \mathbf{T}_2\mathbf{TB}$, define $\langle \beta_i \rangle$ as, for each $i \in \mathbf{N}$, let $\beta_i = i\beta$. It is clear that such $\langle \beta_i \rangle$ is a counterexample of the inverse of Lemma 2. Referring to the discussing in [7], for any two continuous $F, G : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$, if the set $\{(f, x) \mid F(f, x) > G(f, x)\}$ is compact in $\mathbb{T}(F, G)$, we say that F is almost everywhere less than G, denoted as $F \leq_{\mathbf{2}}^{*} G$. If we relax $F_{\beta_i} \leq H$ in Lemma 2 to $F_{\beta_i} \leq_{\mathbf{2}}^{*} H$, then we have the following lemma which is stronger than the inverse of Lemma 2 in the sense that we do not require $\langle \beta_i \rangle$ to be convergent.

Lemma 3. *For any uniform and ascending $\langle \beta_i \rangle$, there is a total continuous functional $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ such that, for every $i \in \mathbf{N}$, $F_{\beta_i} \leq_{\mathbf{2}}^{*} H$.* ■

Note that the functional H in Lemma 3 is not necessarily computable unless we can effectively determine when does each β_i converge. If we can, then H is computable since $\langle\beta_i\rangle$ is uniform and, for every $(f, x) \in \mathcal{T} \times \mathbf{N}$, the minimal locking fragment (τ, x) and $F_{\beta_{(x+\|\tau\|)}}(f, x)$ can be effectively obtained.

3 Non-union Theorems

Let $\mathbf{C}(\beta)$ denote the type-2 complexity class determined by $\beta \in \mathbf{T_2TB}$ [8]. Similarly, let $\mathbf{C}(\langle\beta_i\rangle)$ denote the union class $\bigcup_{i\in\mathbf{N}} \mathbf{C}(\beta_i)$. According to Theorem 2 in [8], if $\langle\beta_i\rangle$ is ascending, then, for every $i \in \mathbf{N}$, $\mathbf{C}(\beta_i) \subseteq \mathbf{C}(\beta_{i+1})$. Clearly, if $\langle\beta_i\rangle$ is strongly convergent with a locking detector ℓ, then each β_i is a strong type-2 time bound because each β_i can share the same locking detector ℓ. The strong convergence of $\langle\beta_i\rangle$ is strong property that turns out to be one of the necessary hypotheses in our type-2 analog of the union theorem. The following theorem indicated that $\mathbf{BFF}_2$ can be described by some $\langle\beta_i\rangle$. The proof uses some results in Cook and Kapron's [2, 5, 6].

Theorem 1 *There is a uniform and ascending $\langle\beta_i\rangle$ such that, $\mathbf{C}(\langle\beta_i\rangle) = \mathbf{BFF}_2$.* ■

The theorem above implies that there is a programming system for $\mathbf{BFF}_2$. Similar to PTIME, $\mathbf{BFF}_2$ can be viewed as a union of complexity classes where each is determined by a second order polynomial. However, we will see later that $\mathbf{BFF}_2$ is not a type-2 complexity class determined by any $\beta \in \mathbf{T_2TB}$. We first observe that, for any $\langle\beta_i\rangle$ such that $\mathbf{C}(\langle\beta_i\rangle) = \mathbf{BFF}_2$, $\langle\beta_i\rangle$ is not convergent. This is easy to see since the depth of a second-order polynomial can be arbitrarily deep. Thus, any locking fragment will not be enough for some second-order polynomial with deeper depth to compute.

Just as with the type-1 theory, in general, the union of two arbitrary complexity classes is not always a complexity class. We will see in the next section that some conditions are needed in order to obtain a type-2 union theorem.

Theorem 2 (Weak Type-2 Non-union Theorem) *There exist $\beta_1 \in \mathbf{T_2TB}$ and $\beta_2 \in \mathbf{T_2TB}$ such that, $\forall\alpha \in \mathbf{T_2TB}, \mathbf{C}(\alpha) \neq \mathbf{C}(\beta_1) \cup \mathbf{C}(\beta_2)$.* ■

Let $\mathbf{C}^a(\beta)$ denotes the complexity class determined by β under the answer-length-cost model, and $\mathbf{C}^u(\beta)$ the complexity class under the unit-cost model. In contexts where the difference between the two models is of no importance, we then simply use $\mathbf{C}(\beta)$.

Theorem 3 (Type-2 Non-Union Theorem) *There is a uniform, ascending, useful, and convergent $\langle\beta_i\rangle$, such that $\mathbf{C}^a(\langle\beta_i\rangle)$ is not a type-2 complexity class.* ■

These negative results (non-union theorems) help us to find and justify our rather strong hypotheses for obtaining a type-2 union theorem. For example, convergence is a rather strong hypothesis, but the theorem above shows that it is not sufficient to have a union theorem. Thus, we have to further strengthen the hypothesis by including *uniform convergence.* Similarly, if we drop the *usefulness* in the hypotheses, then we can modify the proof of Theorem 3 and have the following negative result.

Corollary 1 *There is a uniform, ascending, and uniformly convergent $\langle\beta_i\rangle$, such that $\mathbf{C}^a(\langle\beta_i\rangle)$ is not a type-2 complexity class.* ■

Thus, the *usefulness* of $\langle\beta_i\rangle$ should be added as a necessary condition in our union theorem. However, it is unclear that usefulness together with uniform convergence are sufficient to obtain a type-2 union theorem.

Conjecture 1 *There is a uniform, ascending, useful, and uniformly convergent $\langle\beta_i\rangle$, such that $\mathbf{C}^a(\langle\beta_i\rangle)$ is not a type-2 complexity class.* ■

The following two lemmas are straightforward. We omit the proof.

Lemma 4. *Let $\langle\beta_i\rangle$ be useful. If there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^a(\langle\beta_i\rangle) = \mathbf{C}^a(\alpha)$, then $\langle\beta_i\rangle$ is convergent.* ■

Lemma 5. *Let $\langle\beta_i\rangle$ be useful. If there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^a(\langle\beta_i\rangle) \subseteq \mathbf{C}^a(\alpha)$, then $\langle\beta_i\rangle$ is convergent.* ■

If we allow $\langle\beta_i\rangle$ to be not useful, then Lemma 4, can be disproved by constructing a trivial $\langle\beta_i\rangle$. For example, let $\mathbf{C}^a(\beta_0) = \mathbf{C}^a(\beta_1) = \cdots$ where each β_i delays its convergence until an inaccessible point is reached. Thus, no OTM^a clocked by any β_i can query the inaccessible point. In such a way, each β_i in the sequence determines the same complexity class and hence $\mathbf{C}^a(\beta_0) = \mathbf{C}^a(\langle\beta_i\rangle)$ but the convergence of $\langle\beta_i\rangle$ breaks if we choose a different inaccessible point for each β_i to converge. Based on the discussion in this section, we have the following theorem as our conclusion.

Theorem 4 *There is no $\beta \in \mathbf{T}_2\mathbf{TB}$ such that, $\mathbf{C}(\beta) = \mathbf{BFF}_2$.* ■

Using Lemma 5 we can further prove that, there is no $\beta \in \mathbf{T}_2\mathbf{TB}$ such that, $\mathbf{BFF}_2 \subseteq \mathbf{C}^a(\beta)$. These negative non-union results imply that a straightforward type-2 analog of the Union Theorem does not exist. In the next section we show how to strengthen the hypotheses in order to have a type-2 Union Theorem under answer-length-cost model.

4 Union Theorems

According to Lemma 4, the convergence of $\langle\beta_i\rangle$ is a necessary condition for $\mathbf{C}(\langle\beta_i\rangle)$ to be a complexity class. However, Theorem 3 states that convergence together with uniformity, ascendancy, and usefulness are not sufficient to obtain a union theorem. Strong convergence turns out to be one of the necessary conditions as indicated in the following theorem. We use a priority argument with finite injuries to the theorem.

Theorem 5 (Type-2 Union Theorem) *Suppose that $\langle\beta_i\rangle$ is (i) uniform, (ii) ascending, (iii) useful, and (iv) strongly convergent. Then, there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that, $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle\beta_i\rangle)$.* ■

Both uniform and strong convergence are very strong conditions in the sense that, for every $(f, x) \in \mathcal{T} \times \mathbf{N}$, every β_i has to refer to the same fragment of f. At the moment, we do not see any reasonable way to get rid of this requirement of convergence. Here we discuss an unsuccessful try. We observe that the sample $\langle\beta_i\rangle$ constructed in the proof of the Type-2 Non-Union Theorem (Theorem 3) is not bounded, i.e., $\lim_{i\to\infty} F_{\beta_i}(f, x) = \infty$. We may ask, if $\langle\beta_i\rangle$ is bounded by some continuous functional, can we have a union theorem without requiring $\langle\beta_i\rangle$ to be uniformly convergent? The next corollary gives a negative result.

Corollary 2 *There exist a continuous functional $F : \mathcal{T}\times\mathbf{N} \to \mathbf{N}$ and a uniform, ascending, and useful $\langle\beta_i\rangle$ such that, for every $i \in \mathbf{N}$, $F_{\beta_i} \leq F$, and $\mathbf{C}^a(\langle\beta_i\rangle)$ is not a type-2 complexity class.* ■

Note that if $\langle\beta_i\rangle$ is bounded by a total continuous functional, then, by Lemma 2, $\langle\beta_i\rangle$ is convergent but not necessarily uniformly convergent.

Recall that a *strong* type-2 time bound is an $\mathcal{F}$-monotone one, i.e., for every $\sigma, \tau \in \mathcal{F}$ and $a \in \mathbf{N}$, $\sigma \subseteq \tau \Rightarrow \beta(\sigma, a) \leq \beta(\tau, a)$. We say that $\langle\beta_i\rangle$ is strong if and only if every β_i in $\langle\beta_i\rangle$ is $\mathcal{F}$-monotone. Computations clocked with such kind of time bounds have an intuitive advantage that the budget provided by the clock will never shrink during the courses of the computations. Thus, we may want the type-2 time bound α constructed in the proof of the type-2 Union Theorem to be strong. However, we are strongly skeptical about this. We have the following conjecture.

Conjecture 2 *There is a uniform, ascending, and strong $\langle\beta_i\rangle$ such that, if there is $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle\beta_i\rangle)$, then α is not strong.* ■

The Type-2 big-O Notation: The big-O notation is a key tool in algorithm analysis. A natural type-2 analog of the big-O notation can be defined as follows.

Definition 5 (Type-2 big-O Notation) *Given $\beta \in \mathbf{T}_2\mathbf{TB}$, define*

$$\mathbf{O}(\beta) = \left\{\varphi_e \;\middle|\; \varphi_e \in \mathbf{C}^a(c\beta + d) \text{ for some } c, d \in \mathbf{N}\right\}. \qquad ■$$

In fact, one of our primary motivations to have a type-2 union theorem is to examine whether $\mathbf{O}(\beta)$ is a well-defined type-2 complexity class. In our opinion, if the conditions in the our union theorem do not rule out $\mathbf{O}(\beta)$ to be a type-2 complexity class, we should consider the conditions reasonable, no matter how strong they are. Clearly, if the β is locking detectable, the the sequence β_i defined in $\mathbf{O}(\beta)$ is *strongly convergent*. Thus, by Theorem 5, we can prove the following corollary:

Corollary 3 *Let $\beta \in \mathbf{T}_2\mathbf{TB}$. If β is locking detectable and useful, then there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^a(\alpha) = \mathbf{O}(\beta)$.* ■

Note that, although we have Theorem 9 in [8] asserting that there is an effective operator Θ_L such that, $\Theta_L(\beta)$ is locking detectable and equivalent to β, but

$$[\mathbf{C}^a(\beta) = \mathbf{C}^a(\Theta_L(\beta))] \not\Rightarrow [\mathbf{C}^a(i\beta + i) = \mathbf{C}^a(i\Theta_L(\beta) + i)].$$

On the other hand, if we define $\beta_i = \Theta_L(i\beta + i)$, the strong convergence of $\langle \beta_i \rangle$ may not hold. This is because, if $i \neq j$, the inaccessible points of β_i and β_j are different. Thus, locking detectability of β is required in Corollary 3. We can easily prove the following two addition corollaries.

Corollary 4 *Let $\alpha, \beta \in \mathbf{T_2TB}$. If α and β are locking detectable and useful, then $\mathbf{O}(\alpha + \beta)$ is a type-2 complexity class.* ■

The following corollary states that we can drop the less significant term in the big-O notation. We omit the proof since it is straightforward.

Corollary 5 *Let $\alpha, \beta \in \mathbf{T_2TB}$. Suppose that both α and β are locking detectable and useful. If $\alpha \leq^* \beta$, then $\mathbf{O}(\alpha + \beta) = \mathbf{O}(\beta)$.* ■

5 Conclusion

For decades type-2 complexity theory using a machine model remains an untouched territory. This paper is added to a series of our previous ones devoted to building up this theory from scratch. As the framework becomes clearer due to our specific clocking scheme for OTM and the precise definition of type-2 complexity classes, we decided to push the theory further by proving a union theorem. Based on the theorem, as its type-1 counterpart, we can characterize some intuitive complexity classes in a precise way. Unfortunately, the most familiar $\mathbf{BFF}_2$ fails to pass the test, i.e., it is not a type-2 complexity class under our definition. This result on the one hand indicates that our framework may not be broad enough to encompass this intuitive type-2 complexity class. On the other hand, it may provide another legitimate reason to argue that $\mathbf{BFF}_2$ is not precise enough for further investigation on a theoretical base. The hindsight of our investigation in this paper may be that, we give a type-2 analog of the big-O notation and, according to the union theorem we proved, we can argue that it is a well-defined type-2 complexity class under our framework.

References

1. Manuel Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14(2):322–336, 1967.
2. Stephen A. Cook and Bruce M. Kapron. Characterization of the basic feasible functions of finite type. *Proceedings of the 30th Annual IEEE Symposium on the Foundations of Computer Science*, pages 154–159, 1989.
3. J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transitions of the American Mathematics Society*, pages 285–306, May 1965.

4. Robert J. Irwin, Bruce M. Kapron, and James S. Royer. On characterizations of the basic feasible functionals: Part I. *Journal of Functional Programming*, 2001. (to appear).
5. Bruce M. Kapron. Feasible computation in higher types. Ph.d. dissertation, University of Toronto, 1991.
6. Bruce M. Kapron and Stephen A. Cook. A new characterization of type 2 feasibility. *SIAM Journal on Computing*, 25:117–132, 1996.
7. Chung-Chih Li. Asymptotic behaviors of type-2 algorithms and induced Baire topologies. In *Proceedings of the Third International Conference on Theoretical Computer Science*, pages 471–484, Toulouse, France, August 2004.
8. Chung-Chih Li. Clocking type-2 computation in the unit cost model. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *Proceedings of Computability in Europe, CiE 2006: Logical Approaches to Computational Barriers, CSR 7-2006*, pages 182–192, Swansea, UK, 2006.
9. Chung-Chih Li. Speed-up theorems in type-2 computation. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *Proceedings of Computability in Europe, CiE 2007: Computation and Logic in the Real World*, pages 478–487, Siena, Italy, June 2007. Springer, LNCS 4497.
10. Chung-Chih Li. Query-optimal oracle Turing machines for type-2 computations. In *Proceedings of Computability in Europe, CiE 2008: Logic and Theory of Algorithms*, pages 293–303, Athens, Greece, June 2008.
11. Chung-Chih Li and James S. Royer. On type-2 complexity classes: Preliminary report. In *Proceedings of the Third International Workshop on Implicit Computational Complexity*, pages 123–138, Aarhus, Denmark, May 2001.
12. 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.
13. Piergiorgio Odifreddi. *Classical Recursion Theory, Volume II*, volume 143 of *Studies in Logic and the Foundations of Mathematics.* Elsevier Science Publishing, North-Holland, Amsterdam, 1999.
14. 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.
15. Anil Seth. Complexity theory of higher type functionals. Ph.d. dissertation, University of Bombay, 1994.

Appendix

A Notations and Conventions

1. $\mathbf{N} \equiv_{\text{def}}$ The set of all natural numbers.
2. $\mathcal{R} \equiv_{\text{def}} \mathbf{N} \to \mathbf{N}$, the set of all recursive functions.
3. $\mathcal{P} \equiv_{\text{def}} \mathbf{N} \rightharpoonup \mathbf{N}$, the set of all partial functions.
4. $\mathcal{T} \equiv_{\text{def}} \mathbf{N} \to \mathbf{N}$, the set of all total functions.
5. $\mathcal{PR} \equiv_{\text{def}} \mathbf{N} \rightharpoonup \mathbf{N}$, the set of all partial recursive functions.
6. $\mathcal{F} \equiv_{\text{def}}$ The set of all finite functions from $\mathbf{N}$ to $\mathbf{N}$. That is, for each $\sigma \in \mathcal{F}$, the domain of σ is a finite subset of $\mathbf{N}$. Unless stated otherwise, τ and σ range over $\mathcal{F}$.

7. $\mathsf{dom}(\sigma) \ \equiv_{\mathrm{def}}$ The domain of σ.
8. $\mathsf{card}(\sigma) \ \equiv_{\mathrm{def}}$ The cardinality of $\mathsf{dom}(\sigma)$.
9. $\langle\cdot,\cdot\rangle : \mathbf{N}\times\mathbf{N}\to\mathbf{N} \ \equiv_{\mathrm{def}}$ A fixed pairing function. That is, $\langle\cdot,\cdot\rangle$ is a fixed computable bijection between $\mathbf{N}\times\mathbf{N}$ and $\mathbf{N}$.
10. $\mathbb{N} \ \equiv_{\mathrm{def}}$ The discrete topology on the space $\mathbf{N}$.
11. $\mathbb{T} \ \equiv_{\mathrm{def}}$ The Baire topology on the space $\mathcal{T}$.
12. $\mathbb{T}\times\mathbb{N} \ \equiv_{\mathrm{def}}$ The product topology of $\mathbb{T}$ and $\mathbb{N}$.
13. $\mathbb{T}(F) \ \equiv_{\mathrm{def}}$ The induced topology of $\mathbb{T}\times\mathbb{N}$ determined by the continuous functional $F : \mathcal{T}\times\mathbf{N}\to\mathbf{N}$.
14. $\leq_{\mathbf{2}}^{*} \equiv_{\mathrm{def}}$ The type-2 almost-everywhere less than or equal to relation.
15. OTM $\equiv_{\mathrm{def}}$ Oracle Turing Machine.
16. $\mathrm{OTM}^{a} \equiv_{\mathrm{def}}$ Oracle Turing Machine under the answer-length-cost model.
17. $\mathrm{OTM}^{u} \equiv_{\mathrm{def}}$ Oracle Turing Machine under the unit-cost model.
18. $POTM \ \equiv_{\mathrm{def}}$ Polynomial-time computable Oracle Turing Machine.
19. **BFF** $\equiv_{\mathrm{def}}$ Basic Feasible Functional.
20. For $n\in\mathbf{N}$, $|n| \ \equiv_{\mathrm{def}}$ The length of the presentation of n.
21. For $f\in\mathcal{T}$, $|f| \ \equiv_{\mathrm{def}}$ The length function of f defined by

$$|f| = \lambda n\cdot\mathsf{max}(\{|f(x)| : |x|\leq n, x\in\mathsf{dom}(f)\}).$$

Note that f does not have to be a total function.
22. $\|\sigma\| = \sum\limits_{i\in\mathsf{dom}(\sigma)} (|i| + |\sigma(i)| + 2)$.
23. $\sigma^{\sim 0} \ \equiv_{\mathrm{def}}$ The zero extension of σ. That is, for every $x\in\mathbf{N}$,

$$\sigma^{\sim 0}(x) = \begin{cases}\sigma(x) & \text{if } x\in\mathsf{dom}(\sigma);\\ 0 & \text{otherwise.}\end{cases}$$

24. $\sigma^{-} \ \equiv_{\mathrm{def}} \ \sigma^{-}\subset\sigma$, and $\mathsf{dom}(\sigma^{-}) = \mathsf{dom}(\sigma) - \{\mathsf{max}(\mathsf{dom}(\sigma))\}$, where $\sigma\neq\emptyset$.
25. $f_{[n]} \ \equiv_{\mathrm{def}}$ The initial n-element segment of f, where f is a total function.
26. $\sigma_{[n]} \ \equiv_{\mathrm{def}}$ The initial n-element segment of σ, where $\mathsf{dom}(\sigma)$ is an initial segment of $\mathbf{N}$. By convention, if $\mathsf{max}(\mathsf{dom}(\sigma))\leq n$, then $\sigma_{[n]} = \sigma$.
27. $(\!(\sigma, x)\!) = \big\{(f,x) \ \big| \ \sigma\subset f\big\}$.
28. $\varphi_e \ \equiv_{\mathrm{def}}$ The function computed by the Turing Machine with index e.
29. $\varphi_e \ \equiv_{\mathrm{def}}$ The functional computed by the OTM with index e.
30. $\varphi_e(x)\downarrow^{s} \equiv_{\mathrm{def}}$ The Turing Machine with index e halts in s steps on input $x\in\mathbf{N}$.
31. $\varphi_e(f,x)\downarrow^{s} \equiv_{\mathrm{def}}$ The Oracle Turing Machine with index e halts in s steps on input $(f,x)\in\mathcal{T}\times\mathbf{N}$.
32. $\mathbf{T_2TB} \ \equiv_{\mathrm{def}}$ The set of Type-2 Time-Bounds of type $\mathcal{F}\times\mathbf{N}\to\mathbf{N}$. Unless stated otherwise, we let α,β, and γ range over $\mathbf{T_2TB}$.
33. $F_\beta \ \equiv_{\mathrm{def}}$ The limit functional determined by $\beta\in\mathbf{T_2TB}$.
34. $\varphi_{e,\beta} \ \equiv_{\mathrm{def}}$ The functional computed by the Oracle Turing Machine with index e clocked with $\beta\in\mathbf{T_2TB}$.
35. $\varphi_{e,\beta}(f,x)\Uparrow \equiv_{\mathrm{def}}$ The β-clocked Oracle Turing Machine with index e is clipped by the clock β during the course of computation on $(f,x)\in\mathcal{T}\times\mathbf{N}$.
36. $E_{e,\beta} \ \equiv_{\mathrm{def}} \ \big\{(f,x) \ \big| \ \varphi_{e,\beta}(f,x)\Uparrow\big\}$.

37. $\Phi_e(f,x)$ $\equiv_{\text{def}}$ The expenses needed to compute $\varphi_e(f,x)$. We use step-counting as our complexity measure.
38. $\widetilde{\Phi}_e(\sigma,x) : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$ is defined by

$$\widetilde{\Phi}_e(\sigma,x) = \begin{cases} \Phi_e(\sigma^{0\sim},x), \text{ if } \varphi_e(\sigma^{0\sim},x)\downarrow, \text{ and all said queries are in } \mathsf{dom}(\sigma); \\ \text{The expenses of the computation of } \varphi_e(\sigma^{0\sim},x) \text{ right after the} \\ \quad\text{first query outside } \mathsf{dom}(\sigma) \text{ is completed.} \end{cases}$$

[Note that $\widetilde{\Phi}_e$ is not necessarily recursive.]
39. $\langle\beta_i\rangle$ $\equiv_{\text{def}}$ A sequence of type-2 time bounds $\beta_0, \beta_1, \beta_2, \ldots$.
40. $\mathbf{C}(\beta)$ $\equiv_{\text{def}}$ The type-2 complexity class determined by $\beta \in \mathbf{T_2TB}$.
41. $\mathbf{C}(\langle\beta_i\rangle)$ $\equiv_{\text{def}}$ $\bigcup_i \mathbf{C}(\beta_i)$.

B OTM, A Clocking Scheme, and Two Cost Models

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 $\langle\varphi_i\rangle_{i\in\mathbf{N}}$ associated with a complexity measure $\langle\Phi_i\rangle_{i\in\mathbf{N}}$ 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.

In the following, we present a clocking scheme using our $\mathbf{T_2TB}$. This scheme is used implicitly in some works such as Kapron and Cook's [6], Seth's [15], and Royer's [14].

Definition 1 (Clocked OTM). *Let $\beta \in \mathbf{T_2TB}$ and M_e be an OTM with index e. We say that M_e is clocked by β if M_e is simulated by the procedure shown in Figure 1. Such a clocked OTM is denoted by $M_{e,\beta}$ and the functional computed by $M_{e,\beta}$ is denoted by $\varphi_{e,\beta}$.*

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 of 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 M_e is *clipped down* by β on (f,a) denoted by $\varphi_{e,\beta}(f,a)\Uparrow$. On the other hand, if the simulation reaches the line marked ($\Downarrow$), which means that the simulation of M_e on (f,a) is successfully completed, then we say that $\varphi_{e,\beta}(f,a)$ converges to value $\varphi_e(f,a)$. We denote this situation by $\varphi_{e,\beta}(f,a)\Downarrow$. Since β is convergent, it follows that the

```
Program for Clocked OTM $M_{e,\beta}$ :
        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 $M_e$ 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 ($M_e$ 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

simulation of M_e on (f, a) will either complete or eventually be clipped down by the clock. Therefore, for any $\beta \in \mathbf{T_2TB}$, $\varphi_{e,\beta}$ is a total computable functional of type $\mathcal{T} \times \mathbf{N} \to \mathbf{N}$. This removes the problem of POTM.

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 oracle answers does matter. We have the following two conventions under our clocking scheme.

Definition 2 (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 [6, 14, 15] are the answer-length cost model. Also, the outline of a type-2 complexity theory given in [11] 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, i.e., many interesting theorems in classical complexity theory do not exist under unit cost model.

C Proofs of Lemmas and Theorems

Lemma 1 *If $\beta \in \mathbf{T_2TB}$ is accessible, then β is locking detectable..*

Proof: Let $\beta \in \mathbf{T_2TB}$ be accessible. Fix any OTM^u M_e^u that witnesses β being accessible. Clearly, given any $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$, we can effectively decide if a given (σ, a) is β-queriable. If not, (σ, a) must be a locking fragment of β. Thus, we can have $\ell : \mathcal{F} \times \mathbf{N} \to \{0, 1\}$ such that, ℓ, on every (σ, x), returns 1 if (σ, x) is not β-queriable, or 0 otherwise. (Note that $\varphi_e = \varphi_e^u$.) □

Lemma 2 *Given uniform and ascending $\langle \beta_i \rangle$, if there exists a total continuous functional $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ such that, for every $i \in \mathbf{N}$, $F_{\beta_i} \leq H$, then $\langle \beta_i \rangle$ is convergent.*

Proof: Suppose by contradiction that $\langle \beta_i \rangle$ is not convergent. Thus, there is a $(f, x) \in \mathcal{T} \times \mathbf{N}$ such that, there is no (σ, x) with $\sigma \subset f$ that makes every β_i to converge on it. That is, for each $i \in \mathbf{N}$, if $\sigma_i \subset f$ and (σ_i, x) is a minimal locking fragment of β_i, then there exists $j > i$ such that, (σ_i, x) is not a locking fragment of β_j. Let σ_i, i and j be such. Since $\langle \beta_i \rangle$ is ascending, we have

$$\beta_i(\sigma_i, x) = F_{\beta_i}(f, x) \leq \beta_j(\sigma_i, x) < F_{\beta_j}(f, x).$$

By assumption, $\lim_{i\to\infty} F_{\beta_i}(f, x) \leq H(f, x)$, and hence $\lim_{i\to\infty} F_{\beta_i}(f, x)$ is not bounded. But, since H is a total continuous functional, $H(f, x) \downarrow = y$ for some finite number y. This leads a contradiction, and hence $\langle \beta_i \rangle$ cannot be convergent. □

Lemma 3 *For any uniform and ascending $\langle \beta_i \rangle$, there is a total continuous functional $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ such that, for every $i \in \mathbf{N}$, $F_{\beta_i} \leq_{\mathbf{2}}^{*} H$.*

Proof: Define $H : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ by

$$H(f, x) = F_{\beta_{(x+\|\tau\|)}}(f, x),$$

where $\tau \subset f$ and (τ, x) is a minimal locking fragment of β_x. It is clear that such H is total and continuous.

Fix any $i \in \mathbf{N}$. We shall prove that $F_{\beta_i} \leq_{\mathbf{2}}^{*} H$. Fix $(f, x) \in \mathcal{T} \times \mathbf{N}$ and let $\tau \subset f$ be such that (τ, x) is a minimal locking fragment of β_x. We have two cases: $i \leq x$ and $x < i$.

Case 1: $i \leq x$. Since $\langle \beta_i \rangle$ is ascending, it is clear that,

$$F_{\beta_i}(f,x) \leq F_{\beta_x}(f,x) \leq F_{\beta_{x+\|\tau\|}}(f,x) = H(f,x).$$

Case 2: $x < i$. Suppose that $H(f,x) < F_{\beta_i}(f,x)$, i.e.,

$$H(f,x) = F_{\beta_{x+\|\tau\|}}(f,x) < F_{\beta_i}(f,x).$$

There are finitely many different (τ, x)'s such that, (τ, x) is a locking fragment of β_i and $\|\tau\| \leq i-x$. Therefore, $\{(f,x) | F_{\beta_i}(f,x) > H(f,x)\}$ is compact in $\mathbb{T}(F_{\beta_i}, H)$, and hence $F_{\beta_i} \leq_{\mathbf{2}}^{*} H$. □

Theorem 1 *There is a uniform and ascending $\langle \beta_i \rangle$ such that,* $\mathbf{C}(\langle \beta_i \rangle) = \mathbf{BFF_2}$.

Proof: For each $k \in \mathbf{N}$, let p_k be the two-variable polynomial defined by

$$p_k(m,n) = k(m+n+1)^k.$$

Here we follow the definition of $q^{d,k}$ in [4]. For every $d, k \in \mathbf{N}$, we recursively define the second-order polynomial $q^{d,k}$ as follows.

For every $g : \mathbf{N} \to \mathbf{N}$ and $y \in \mathbf{N}$,

$$\begin{aligned} q^{0,k} =& p_k(0,y). \\ q^{d+1,k} =& p_k(g(q^{d,k}(g,y)), y). \end{aligned}$$

Recall that $|x|$ is a natural number that denotes the length of the bits representation of $x \in \mathbf{N}$. We use the same notation for $f \in \mathcal{T}$. For for $f \in \mathcal{T}$, $|f|$ is called the length function defined as follows.

$$|f| = \lambda n \cdot \mathsf{max}(\{|f(x)| : |x| \leq n, x \in \mathsf{dom}(f)\}).$$

By a straightforward argument, we have that, for every second-order polynomial $\boldsymbol{q}$ over $f : \mathbf{N} \to \mathbf{N}$ and $x \in \mathbf{N}$, if d is the depth of $\boldsymbol{q}$, then there is a $k \in \mathbf{N}$ such that, for every f and x,

$$\boldsymbol{q}(|f|, |x|) \leq q^{d,k}(|f|, |x|).$$

For each $i \in \mathbf{N}$, define $\beta_i : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$ by

$$\beta_i(\sigma, x) = \mathsf{max}(|x|+1, q^{i,i}(|\sigma^{\sim 0}|, |x|)).$$

Since, for each $i \in \mathbf{N}$ and $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$, $q^{i,i}(|\sigma^{\sim 0}|, |x|)$ is computable, β_i is computable and hence $\langle \beta_i \rangle$ is uniform. It is also clear that β_i is nontrivial, strong $\mathcal{F}$-monotone, and convergent. Moreover, since if $i \leq j$, then $q^{i,i} \leq q^{j,j}$, it follows that $\langle \beta_i \rangle$ is ascending.

If (σ, x) is a locking fragment of β_i, then, for all $f \supset \sigma$, we have

$$F_{\beta_i}(f,x) = \mathsf{max}(|x|+1, q^{i,i}(|f|, |x|)).$$

Thus, F_{β_i} is a second-order polynomial of depth i. For every second-order polynomial $\boldsymbol{q}$ such that, for every $(f,x) \in \mathcal{T} \times \mathbf{N}$, $\boldsymbol{q}(|f|,|x|) \leq q^{d,k}(|f|,|x|)$, we have $\boldsymbol{q} \leq q^{d,k} \leq F_{\beta_i}$, where $i = \mathsf{max}(d,k)$. By the Lemma for the *continuity of bounds* in [4], if the functional F can be computed in time bounded by F_{β_i}, then $F \in \mathbf{C}(\beta_i)$. Thus, F is bounded by some β_i if and only if there is an OMT M_e that computes F and its runtime is bounded by a second-order polynomial. $\square$

Theorem 2 (Weak Type-2 Non-union Theorem) *There are two type-2 time bounds, β_1 and β_2 such that,*

$$\forall \alpha \in \mathbf{T_2TB}, \mathbf{C}(\alpha) \neq \mathbf{C}(\beta_1) \cup \mathbf{C}(\beta_2).$$

Proof: Define $F_1, F_2, F_3 : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$, respectively, by

$$F_1(f,x) = \begin{cases} 2^{3\cdot|f(x)|} & \text{if } x \text{ is odd;} \\ 2^{|f(x)|} & \text{otherwise,} \end{cases}$$

$$F_2(f,x) = \begin{cases} 2^{3\cdot|f(x)|} & \text{if } x \text{ is even;} \\ 2^{|f(x)|} & \text{otherwise,} \end{cases}$$

$$F_3(f,x) = 2^{2\cdot|f(x)|}.$$

It is clear that we can choose e_1, e_2, and e_3 so that $\varphi_{e_1} = F_1, \varphi_{e_2} = F_2, \varphi_{e_3} = F_3$, and the complexity of each program is bounded as follows, for each $(f,x) \in \mathcal{T} \times \mathbf{N}$:

$$\Phi_{e_1}(f,x) \leq \begin{cases} |x| + 3\cdot|f(x)| + c & \text{if } x \text{ is odd;} \\ |x| + |f(x)| + c & \text{otherwise,} \end{cases}$$

$$\Phi_{e_2}(f,x) \leq \begin{cases} |x| + 3\cdot|f(x)| + c & \text{if } x \text{ is even;} \\ |x| + |f(x)| + c & \text{otherwise,} \end{cases}$$

$$\Phi_{e_3}(f,x) \leq |x| + 2\cdot|f(x)| + c,$$

where $c \geq 1$ is some constant. Define $\beta_1, \beta_2 : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$ by:

$$\beta_1(\sigma,x) = \begin{cases} |x| + 3\cdot|\sigma(x)| + c & \text{if } x \text{ is odd and } x \in \mathsf{dom}(\sigma); \\ |x| + |\sigma(x)| + c & \text{if } x \text{ is even and } x \in \mathsf{dom}(\sigma); \\ |x| + c & \text{otherwise,} \end{cases}$$

$$\beta_2(\sigma,x) = \begin{cases} |x| + 3\cdot|\sigma(x)| + c & \text{if } x \text{ is even and } x \in \mathsf{dom}(\sigma); \\ |x| + |\sigma(x)| + c & \text{if } x \text{ is odd and } x \in \mathsf{dom}(\sigma); \\ |x| + c & \text{otherwise.} \end{cases}$$

Clearly, $\beta_1, \beta_2 \in \mathbf{T_2TB}$. It is also clear that $\varphi_{e_1} \in \mathbf{C}(\beta_1)$ and $\varphi_{e_2} \in \mathbf{C}(\beta_2)$. Since there are infinitely many odd x such that, for some $f \in \mathcal{T}$,

$$F_{\beta_2}(f,x) = |x| + |f(x)| + c < \Phi_{e_1}(f,x) \leq |x| + 3\cdot|f(x)| + c,$$

we have that $\varphi_{e_1} \notin \mathbf{C}(\beta_2)$. Similarly, $\varphi_{e_2} \notin \mathbf{C}(\beta_1)$, and $\varphi_{e_3} \notin \mathbf{C}(\beta_1) \cup \mathbf{C}(\beta_2)$.

By contradiction, suppose that there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that, $\mathbf{C}(\alpha) = \mathbf{C}(\beta_1) \cup \mathbf{C}(\beta_2)$. That means, $F_1, F_2 \in \mathbf{C}(\alpha)$, and $F_3 \notin \mathbf{C}(\alpha)$. By definition, $E_{e_1,\alpha}$ and $E_{e_2,\alpha}$ are compact in $\mathbb{T}(\varphi_{e_1})$ and $\mathbb{T}(\varphi_{e_2})$, respectively, and $E_{e_3,\alpha}$ is noncompact in $\mathbb{T}(\varphi_{e_3})$. Because F_1, F_2, and F_3 have the same minimal locking fragment on every $(f, x) \in \mathcal{T} \times \mathbf{N}$, we have that $\mathbb{T}(\varphi_{e_1}) = \mathbb{T}(\varphi_{e_2}) = \mathbb{T}(\varphi_{e_3})$. For simplicity, let $\mathbb{T}$ denote $\mathbb{T}(\varphi_{e_1})$, $\mathbb{T}(\varphi_{e_2})$, or $\mathbb{T}(\varphi_{e_3})$, without confusion.

Define $\mathcal{B}_o, \mathcal{B}_e \subseteq \mathcal{F} \times \mathbf{N}$ by:

$$\begin{aligned}\mathcal{B}_o &= \left\{ (\!(\sigma, x)\!) \;\middle|\; x \text{ is odd and } \mathsf{dom}(\sigma) = \{x\} \right\}, \\ \mathcal{B}_e &= \left\{ (\!(\sigma, x)\!) \;\middle|\; x \text{ is even and } \mathsf{dom}(\sigma) = \{x\} \right\}.\end{aligned}$$

Let $\mathcal{B} = \mathcal{B}_o \cup \mathcal{B}_e$. Thus, $\mathcal{B}$ is the basis of $\mathbb{T}$ consisting of minimal locking fragments of φ_{e_1}, φ_{e_2}, and φ_{e_3}. Let $\mathcal{O} \subseteq \mathbb{T}$ be an open cover for $E_{e_3,\alpha}$ without any finite subcover. Suppose

$$\mathcal{O} = \{(\!(\sigma_1, x_1)\!), (\!(\sigma_2, x_2)\!), \ldots\},$$

where (σ_i, x_i) is a minimal locking fragment of φ_{e_3} for every $i \in \mathbf{N}$. Note that, for every $i \in \mathbf{N}$, $\mathsf{dom}(\sigma_i) = \{x_i\}$, and (σ_i, x_i) is also a minimal locking fragment of φ_{e_1} and φ_{e_2}. By the assumption on $\mathcal{O}$, there are infinitely many $(\!(\sigma_i, x_i)\!) \in \mathcal{O}$ such that, the clock α cannot provide enough budget for the computation of $\varphi_{e_3,\alpha}$ on $(\sigma_i^{\sim 0}, x_i)$. Since $\Phi_{e_3}(\sigma_i^{\sim 0}, x_i) \leq |x| + 2 \cdot |\sigma_i(x_i)| + c$, we have $\alpha(\sigma_i, x_i) < |x| + 2 \cdot |\sigma_i(x_i)| + c$. In other words,

$$\overset{\infty}{\exists}\ (\!(\sigma, x)\!) \in \mathcal{B}\ \ \left[\alpha(\sigma, x) < |x| + 2 \cdot |\sigma(x)| + c\right]. \tag{3}$$

Since $E_{e_1,\alpha}$ is compact in $\mathbb{T}$, it follows that, for all but finitely many $(\!(\sigma, x)\!) \in \mathcal{B}$, we have $\Phi_{e_1}(\sigma^{\sim 0}, x) \leq \alpha(\sigma, x)$. Thus,

$$\overset{\infty}{\forall}\ (\!(\sigma, x)\!) \in \mathcal{B}_o\ \ \left[|x| + 3 \cdot |\sigma(x)| + c \leq \alpha(\sigma, x)\right]. \tag{4}$$

Similarly, consider $E_{e_2,\alpha}$. We have

$$\overset{\infty}{\forall}\ (\!(\sigma, x)\!) \in \mathcal{B}_e\ \ \left[|x| + 3 \cdot |\sigma(x)| + c \leq \alpha(\sigma, x)\right]. \tag{5}$$

From (4) and (5), we have

$$\overset{\infty}{\forall}\ (\!(\sigma, x)\!) \in \mathcal{B}\ \ \left[|x| + 3 \cdot |\sigma(x)| + c \leq \alpha(\sigma, x)\right]. \tag{6}$$

Clearly, (6) and (3) is a contradiction. Therefore, no such α exists. □

Theorem 3 (Type-2 Non-Union Theorem) *There is a uniform, ascending, useful, and convergent $\langle \beta_i \rangle$, such that $\mathbf{C}^a(\langle \beta_i \rangle)$ is not a type-2 complexity class.*

Proof: Define $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ by

$$F(f, x) = \begin{cases} 1 & \text{if } f(0) = f(1) = \cdots = f(x) = 0; \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, F is computable and total. Let e be a φ-program for F such that, on every (f, x), e does not make any unnecessary query outside $\{0, 1, \ldots, x\}$, and

$$\Phi_e(f, x) \leq \|f_{[x+1]}\| + |x| + c,$$

for some constant $c \geq 1$. It is clear that such an e exists. Define $\langle \beta_i \rangle$ by:

$$\beta_i(\sigma, x) = \begin{cases} |x| + c & \text{if } x \geq i; \\ \|\sigma_{[x+2]}\| + |x| + i & \text{otherwise.} \end{cases}$$

Clearly, $\langle \beta_i \rangle$ is uniform, ascending, and useful. For convergence, given $(f, x) \in \mathcal{T} \times \mathbf{N}$, it follows that $f_{[x+2]}$ is a locking fragment for all β_i. Note that every β_i is locking detectable, but $\langle \beta_i \rangle$ is neither strongly nor uniformly convergent. For example, the minimal locking fragment for β_0 is $(\emptyset, x)$, while for β_i with $i > x$ the minimal locking fragment is (σ, x) where $\mathsf{dom}(\sigma) = \{0, 1, \ldots, x + 1\}$.

Claim 1: $F \notin \mathbf{C}^a(\langle \beta_i \rangle)$.
Fix any $i \in \mathbf{N}$. For every $(f, x) \in \mathcal{T} \times \mathbf{N}$, if $x \geq i$, then $\varphi_{e,\beta_i}(f, x) \Uparrow$. Thus, for any $i \in \mathbf{N}$, $F \notin \mathbf{C}^a(\beta_i)$, and hence $F \notin \mathbf{C}^a(\langle \beta_i \rangle)$.
Claim 2: For all $\alpha \in \mathbf{T_2TB}$, $\mathbf{C}^a(\alpha) \neq \mathbf{C}^a(\langle \beta_i \rangle)$.
By contradiction, suppose that $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle \beta_i \rangle)$. It follows that $F \notin \mathbf{C}^a(\alpha)$. Fix any $(g, a) \in \mathcal{T} \times \mathbf{N}$ such that, $\varphi_{e,\alpha}(g, a) \Uparrow$. Thus, there must exist $\sigma \subseteq g_{[a+1]}$ such that,

$$\alpha(\sigma, a) < \|\sigma\| + |a| + c.$$

Otherwise, $\varphi_{e,\alpha}(g, a) \Downarrow$. Note that $a + 1 \notin \mathsf{dom}(\sigma)$.
Consider the follow computable functional $F' : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$,

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

If e' is a φ-program for F', then, for every $f \in \mathcal{T}$ with $\sigma \subset f$, we have

$$\Phi_{e'}(f, a) > \|\sigma\| + |a| + |f(a + 1)|.$$

Clearly, no φ-program can compute F' with complexity less than $\|\sigma\| + |a| + |f(a + 1)|$ on $(\!(\sigma, a)\!)$. Thus,

$$\forall (f, a) \in (\!(\sigma, a)\!) \; [\varphi_{e',\alpha}(f, a) \Uparrow],$$

and hence $(\!(\sigma, a)\!) \subseteq E_{e',\alpha}$. Since (σ, a) is not a locking fragment of $\varphi_{e'}$, it follows $(\!(\sigma, a)\!)$ is noncompact in $\mathbb{T}(\varphi_{e'})$. Since if X is compact in $\mathbb{T}(\varphi_{e'})$, then

every $S \subseteq X$ is also compact in $\mathbb{T}(\varphi_{e'})$, by contrapositive, we conclude that $E_{e',\alpha}$ is noncompact in $\mathbb{T}(\varphi_{e'})$. Therefore, $F' \notin \mathbf{C}^a(\alpha)$.
On the other hand, we can have a φ-program e' for F' such that, for every $(f, x) = \mathcal{T} \times \mathbf{N}$, we have

$$\Phi_{e'}(f, x) \leq \begin{cases} |x| + c & \text{if } x \neq a, \\ \|f_{[a+2]}\| + |a| + c & \text{if } x = a. \end{cases}$$

Thus, if we select a sufficiently large $i \geq \mathsf{max}(c, a)$, then $\varphi_{e'} \in \mathbf{C}^a(\beta_i)$ and hence $F' \in \mathbf{C}^a(\langle\beta_i\rangle)$. This contradicts our assumption that $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle\beta_i\rangle)$. Therefore, $\mathbf{C}^a(\alpha) \neq \mathbf{C}^a(\langle\beta_i\rangle)$. □

Corollary 1 *There is a uniform, ascending, and uniformly convergent $\langle\beta_i\rangle$, such that $\mathbf{C}^a(\langle\beta_i\rangle)$ is not a type-2 complexity class.*

Sketch of Proof: We can use the exact arguments for Theorem 3 with the following modification on $\langle\beta_i\rangle$. For every $i, x \in \mathbf{N}$ and $\sigma \in \mathcal{F}$, define

$$\beta_i(\sigma, x) = \begin{cases} \|\sigma_{[x+2]}\| + |x| + i & \text{if } x < i; \\ |x| + c + 1 & \text{if } x \geq i \text{ and } \sigma(0) = \sigma(1) = \cdots = \sigma(x+1) = 0; \\ |x| + c & \text{otherwise.} \end{cases}$$

It is clear that every β_i is to converge at the same fragment. Note that, $\langle\beta_i\rangle$ is no longer useful. □

Theorem 5 (Type-2 Union Theorem) *Suppose that $\langle\beta_i\rangle$ is (i) uniform, (ii) ascending, (iii) useful, and (iv) strongly convergent. Then, there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that, $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle\beta_i\rangle)$.* ■

Proof: Let $\ell : \mathcal{F} \times \mathbf{N} \to \{0, 1\}$ be a locking detector for $\langle\beta_i\rangle$ as in Definition 4. Let $\langle\cdot,\cdot\rangle : \mathcal{F} \times \mathbf{N} \to \mathbf{N}$ be our standard 1-1 pairing function with the extra property that: For all $x \in \mathbf{N}$ and $\sigma, \tau \in \mathcal{F}$, if $\sigma \subseteq \tau$, then $\langle\sigma, x\rangle \leq \langle\tau, x\rangle$. Such a $\langle\cdot,\cdot\rangle$ can be constructed easily.

The following requirements are naively lifted from the original arguments in the proof of the Union Theorem[2],

$$P_n^* : \overset{\infty}{\forall} (\sigma, x)[\beta_n(\sigma, x) \leq \alpha(\sigma, x)].$$

These would certainly be easy to satisfy, but they are not sufficient for our purposes. The problem is that, we have

$$\overset{\infty}{\forall} (\sigma, x)[\beta(\sigma, x) \leq \alpha(\sigma, x)] \;\not\Rightarrow\; \mathbf{C}^a(\beta) \subseteq \mathbf{C}^a(\alpha).$$

[2] See [13], page 55

We therefore use the following requirements for our priority argument.

> P_n**:** $\mathbf{C}^a(\beta_n) \subseteq \mathbf{C}^a(\alpha)$.
> N_n**:** $[\varphi_n \notin \mathbf{C}^a(\langle\beta_i\rangle)] \;\Rightarrow\; [\varphi_n \notin \mathbf{C}^a(\alpha)]$.

The priority ordering of these is:

$$N_0 > P_0 > N_1 > P_1 > \cdots .$$

Clearly, if we can construct an $\alpha \in \mathbf{T}_2\mathbf{TB}$ that satisfies all of these requirements, then the theorem follows.

We construct a type-2 time bound α in stages. We will make sure that every P_n will be satisfied after finitely many stages. Each N_n will be satisfied in the limit. That is, no finitary actions can guarantee that N_n is satisfied, but it will be so if we take certain actions infinitely often. Moreover, we will see in a moment how a P_i can be *injured* when we try to satisfy some N_j with $j \leq i$ in finitely many stages, although there are infinitely many stages in which some actions will be taken in order to satisfy each such N_j.

We maintain an array, *guess*, which is global to every stage. At the beginning of each stage $k > 0$, the values of $guess(0), guess(1), \ldots, guess(k-1)$ are natural numbers defined from the previous stage. Intuitively, $guess(e) = n$ means that we guess the functional φ_e is in $\mathbf{C}^a(\beta_n)$. Each stage e will consider φ-programs $0, 1, \ldots, e$. In stage e, when φ-program e is first considered, we guess that $\varphi_e \in \mathbf{C}^a(\beta_e)$ and set $guess(e) = e$. In a later stage k, if we have evidence that φ_e should not be in $\mathbf{C}^a(\beta_{guess(e)})$, then the value of $guess(e)$ will be changed to k.

Actions of The Algorithm for α: The algorithm for α is given in Figure 2 below. Here we discuss some of the strategies behind this algorithm. Using the definition in item 38 of the list in Appendix A, $\widetilde{\Phi}_e(\sigma, x)$ is the cost of computing $\varphi_e(\sigma^{\sim 0}, x)$ up to the completion of the first query outside $\mathsf{dom}(\sigma)$, or $\widetilde{\Phi}_e(\sigma, x)$ is simply the cost of computing $\varphi_e(\sigma^{\sim 0}, x)$ if no query outside $\mathsf{dom}(\sigma)$ made during the course of the computation.

In stage $k = \langle \sigma, x\rangle$ the algorithm for α takes σ and x as its inputs. If there exists $\tau \subset \sigma$ such that $\ell(\tau, x) = 1$, we force α to converge on (σ, x) and no further action will be taken. Otherwise, the algorithm checks programs $e = 0, 1, \ldots, k$ and takes some proper actions according to the following. If $\widetilde{\Phi}_e(\sigma, x) \leq \beta_{guess(e)}(\sigma, x)$, then this means that we don't have enough evidence to refute the current guess, $\beta_{guess(e)}$, for program e in this stage. In this case we shall let α provide enough budget for the computation of $\varphi_e(\sigma^{\sim 0}, x)$ to finish either completely or else up to the point where the first query outside $\mathsf{dom}(\sigma)$ is made. On the other hand, if $\beta_{guess(e)}(\sigma, x) < \widetilde{\Phi}_e(\sigma, x)$, then it is doubtful that $\varphi_e \in \mathbf{C}^a(\beta_{guess(e)})$. We thus force α to clip the computation of $\varphi_e(\sigma^{\sim 0}, x)$ and give program e a bigger guess, k, for another chance. For convenience, define:

$$\begin{aligned} A_k &= \left\{ e \;\middle|\; 0 \leq e \leq k, \widetilde{\Phi}_e(\sigma, x) \leq \beta_{guess(e)}(\sigma, x) \right\}. \\ B_k &= \left\{ e \;\middle|\; 0 \leq e \leq k, \beta_{guess(e)}(\sigma, x) < \widetilde{\Phi}_e(\sigma, x) \right\}. \end{aligned}$$

We use **A1** and **A2** to denote the following two actions.

```
Program for $\alpha$
        input   $\sigma : \mathcal{F}, x : \mathbf{N}$;
        $k \longleftarrow \langle \sigma, x \rangle$;                    /* We call this stage $k$.   */
/* $guess$ is an array of natural numbers global to all stages. */
/* The initial values of $guess(0)$,$guess(1), \ldots, guess(k-1)$ for this stage $k$ are
   obtained by running $\alpha$ on every $(\sigma', x')$ with $0 \leq \langle \sigma', x' \rangle < k$. */

        for $\langle \sigma', x' \rangle = 0$ to $k - 1$ Run $\alpha(\sigma', x')$;

/* Guess that the new $\varphi$-program $k$ is bounded by $\beta_k$. */
        $guess(k) \longleftarrow k$;

/* We check if $\alpha$ has to converge at this point. */
$\triangle$      if $\exists \tau [\tau \subset \sigma$ and $\ell(\tau, x) = 1)]$
            then return max$(\{\alpha(\tau, x) | \tau \subset \sigma\})$;

/* A1: Since max$(guess(0), guess(1), \ldots, guess(k-1)) < k$ and $\langle \beta_i \rangle$ is as-
   cending, we set $ceiling$ to $\beta_k(\sigma, x)$ as the maximal value for $\alpha(\sigma, a)$ in such
   a way the action A1 can be satisfied. */
        $ceiling \longleftarrow \beta_k(\sigma, x)$;

/* A2: In the following, we check $\varphi$-programs $e = 0, 1, \ldots, k$. If $\varphi_e$ is not
   bounded by $\beta_{guess(e)}$ on $(\sigma, x)$, then we take the action A2 by lowering
   $ceiling$ and assign the new guess, $k$, for $e$.                                    */
        for $e = 0$ to $k$ do
            if $\widetilde{\Phi}_e(\sigma, x) > \beta_{guess(e)}(\sigma, x)$ do              /* $e$ is picked on.   */
                $ceiling \longleftarrow$ min$(ceiling, \beta_{guess(e)}(\sigma, x))$;
                $guess(e) \longleftarrow k$;
            end if-do;
        end for-do;
        return $ceiling$;
End program
```

Fig. 2. An Algorithm for the Type-2 Union Theorem

A1: α is defined so as to provide enough budget allowing the computations of the φ-programs in A_k on $(\sigma^{\sim 0}, x)$ to finish either completely or else up to the point where the first query outside $\mathsf{dom}(\sigma)$ is made. This action explicitly tries to satisfy requirements $P_{guess(0)}, P_{guess(1)}, \ldots,$ and $P_{guess(k)}$.

A2: α is defined so as to clip the computations of the φ-programs in B_k on $(\sigma^{\sim 0}, x)$, and the algorithm will give these programs a new guess, k. This action is intended to satisfy requirements $N_0, N_1, \ldots, N_k$ but it may also injure some $P_{guess(e)}$, where $e \in A_k$.

We shall argue in a moment that every requirement will be satisfied eventually as the stages progressed. In other words, each P_e can be injured by some N_k at most finitely many times. Note that, for every $n \in \mathbf{N}$, $P_{n+1} \Rightarrow P_n$. Thus, if P_k is satisfied, so is every P_n with $n \leq k$.

Terminology and Notation: Suppose $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$ and $\langle \sigma, x \rangle = k$.

1. Let $guess_k$ denote the state of the array, *guess*, as of the beginning of stage k. The variable *ceiling* : $\mathbf{N}$ that is local to every stage holds the maximal value of $\alpha(\sigma, x)$ for the action **A2**. Let $ceiling_k$ denote the value of *ceiling* as of the end of stage k. Thus, $\alpha(\sigma, x) = ceiling_k$.
2. We say that the φ-program $e \leq k$ ***participates*** in lowering *ceiling* in stage k if and only if $\widetilde{\Phi}_e(\sigma, x) > \beta_{guess_k(e)}(\sigma, x)$ and, for all $\tau \subset \sigma$, $\ell(\tau, x) = 0$.
3. We say that α ***picks on*** φ-program e at (σ, x) if and only if
$$0 \leq e \leq k \text{ and } \widetilde{\Phi}_e(\sigma, x) > \alpha(\sigma, x).$$
According to the definition of $\widetilde{\Phi}_e$, for every $\sigma, \tau \in \mathcal{F}$ with $\sigma \subset \tau$ and $x \in \mathbf{N}$, we have $\widetilde{\Phi}_e(\sigma, x) \leq \widetilde{\Phi}_e(\tau, x)$. It is clear that, if α *picks on* φ-program e at (σ, x), then, for all $f \supset \sigma$, the clock α will clip the computation of M_e on (f, x), i.e., $\varphi_{e,\alpha}(f, x) \Uparrow$. However, since (σ, x) is not necessarily a locking fragment of φ_e and it is possible that $\alpha(\sigma, x) \leq \alpha(\tau, x)$, it follows that there may exist some other φ-program e' for φ_e such that, at the beginning of the computation, φ-program e' queries the oracle on $\mathsf{dom}(\tau)$ in order to gain the budget $\alpha(\tau, x)$ first. In such a way, we may have $\varphi_{e',\alpha}(f, x) \Downarrow$.
4. We say that φ-program e is a ***casualty*** of α at (σ, x) in stage k, if and only if there exists some $i \leq k$ such that
$$\begin{aligned} \beta_{guess_k(i)}(\sigma, x) &< \widetilde{\Phi}_i(\sigma, x), \text{ and} \\ \beta_{guess_k(i)}(\sigma, x) &< \widetilde{\Phi}_e(\sigma, x) \leq \beta_{guess_k(e)}(\sigma, x). \end{aligned}$$
On (σ, x), α does not intend to *pick on* e, but when α tries to *pick on* some other φ-program (in the case above, φ-program i), it lowers *ceiling* enough so that the clock α may clip some computations of M_e on (f, x) as a casualty, where $\sigma \subset f$.

Unlike the construction for the original proof of the Union Theorem [12, 13], in some cases of $\varphi_e \in \mathbf{C}^{\alpha}(\langle \beta_i \rangle)$, the value of $guess(e)$ may change infinitely often due to the fact that a compact set may be infinite. The difference between $\varphi_e \in \mathbf{C}^a(\langle\beta_i\rangle)$ and $\varphi_e \notin \mathbf{C}^a(\langle\beta_i\rangle)$ is the following. In case that $\varphi_e \in \mathbf{C}^a(\langle\beta_i\rangle)$, if α should change its guess for program e on infinitely many (σ, x)'s, then all but finitely many of these are locking fragments of φ_e but *not minimal ones.* That is, the set $(\!(\sigma, x)\!)$ is not $\mathbb{T}(\varphi_e)$-open, and there are finitely many basic $\mathbb{T}(\varphi_e)$-open sets that cover all of such infinitely many $(\!(\sigma, x)\!)$'s. On the other hand, if $\varphi_e \notin \mathbf{C}^a(\langle\beta_i\rangle)$, α will change its guess on infinitely many minimal locking fragments of φ_e.

Claim 1: $\alpha \in \mathbf{T}_2\mathbf{TB}$. We shall prove that α is computable, nontrivial, $\mathcal{F}$-monotone, and convergent.

1. Computability. Since $\langle\beta_i\rangle$ is strongly convergent, the locking detector ℓ is recursive. Also, it is clear that $\widetilde{\Phi}_e(\sigma, x) > \beta_{guess(e)}(\sigma, x)$ is recursively decidable. Thus, α is recursively computable.
2. Nontriviality. For any (σ, x), the value of $\alpha(\sigma, x)$ is based on the value of $\beta_i(\sigma, x)$ for some $i \in \mathbf{N}$. Thus, α must be nontrivial.
3. $\mathcal{F}$-monotonicity. Since whenever α decides to converge on (σ, x), α outputs $\mathsf{max}(\{\alpha(\tau, x) | \tau \subset \sigma\})$, it is clear that α is $\mathcal{F}$-monotonicity.
4. Convergence. Fix $(f, x) \in \mathcal{T} \times \mathbf{N}$. Since $\langle\beta_i\rangle$ is strongly convergent and ℓ is its locking detector, there is a $\sigma \in \mathcal{F}$ with $\sigma \subset f$ such that the condition of the if statement in line marked $\triangle$ is true. Thus, for all $\tau \supseteq \sigma$, $\alpha(\tau, x) = \alpha(\sigma, x)$ and hence α is convergent.

Therefore, $\alpha \in \mathbf{T}_2\mathbf{TB}$.

Note that although we have the assumption that $\langle\beta_i\rangle$ is useful, α may not be useful because it is possible that $\ell(\sigma, x) = 0$ and $\beta_i(\sigma, x)\downarrow$, and

$$\alpha(\sigma, x) = \beta_i(\sigma, x) < \|\sigma\| + |x| + 2.$$

We will see later that this introduces a difficulty when we need to patch some φ-program e that is clipped by α at a point (σ, x) in case (σ, x) is not a locking fragment of α.

Claim 2: $\bigcup_{n=0}^{\infty} \mathbf{C}^a(\beta_n) \subseteq \mathbf{C}^a(\alpha)$.

Fix $n \in \mathbf{N}$ and suppose that $F \in \mathbf{C}^a(\beta_n)$. Moreover, suppose that e is a φ-program for F and E_{e,β_n} is compact in $\mathbb{T}(\varphi_e)$. We shall prove that $F \in \mathbf{C}^a(\alpha)$. Note that we cannot prove that $E_{e,\alpha}$ is compact in $\mathbb{T}(\varphi_e)$. Instead, we will prove that there is another φ-program p for F such that $E_{p,\alpha}$ is compact in $\mathbb{T}(\varphi_p)$. Define:

$$I_e = \left\{(\sigma, x) \in \mathcal{F} \times \mathbf{N} \;\middle|\; \alpha(\sigma, x) < \widetilde{\Phi}_e(\sigma, x)\right\}. \tag{7}$$

$$O_e = \left\{(\sigma, x) \in \mathcal{F} \times \mathbf{N} \;\middle|\; (\!(\sigma, x)\!) \text{ is open in } \mathbb{T}(\varphi_e)\right\}. \tag{8}$$

$$X_e = I_e \cap O_e. \tag{9}$$

I_e is the set on which φ-program e is picked on by α. O_e is the set of the minimal locking fragments of φ_e and the sub-fragments of the minimal locking fragments of φ_e. In other words, for every $(\sigma, x) \in O_e$, there exists

some $\tau \supseteq \sigma$ such that, $(\tau, x) \in O_e$ and (τ, x) is a minimal locking fragment of φ_e.
We first prove that X_e is finite. Then, we show that there is another φ-program p for F such that $E_{p,\alpha}$ is compact in $\mathbb{T}(\varphi_p)$. Note that if p is another φ-program for F, then, by definition, $\mathbb{T}(\varphi_p) = \mathbb{T}(\varphi_e) = \mathbb{T}(F)$, although φ-programs p and e may make different queries. We will carefully distinguish between $\mathbb{T}(\varphi_p)$ and $\mathbb{T}(\varphi_e)$ when talking about the compactness of $E_{p,\alpha}$ or $E_{e,\alpha}$ in order to avoid confusion.

Claim 2.1: X_e is finite.
In any stage we say that the guess for e is a *bad guess* if $guess(e) < n$. Also, we say that (f, x) is a *bad input* if $(f, x) \in E_{e,\beta_n}$. Without loss of generality, we may assume that, if $(f, x) \in E_{e,\beta_n}$, then for every $i \in \mathbf{N}$ $\varphi_{e,\beta_i}(f, x) \Uparrow$.
Fix any $(\sigma, x) \in O_e$. In the course of the computation of α on (σ, x), there are three possible reasons for (σ, x) to be in I_e: (*i*) the ceiling is too low due to a bad guess from some previous stage for e, (*ii*) e is a casualty when α tries to pick on some other φ-programs at (σ, x), and (*iii*) $\sigma \subset f$ and (f, x) is a bad input, i.e., $(f, x) \in E_{e,\beta_n}$. Note that these three reasons are not exclusive. (See Fig. 3.)

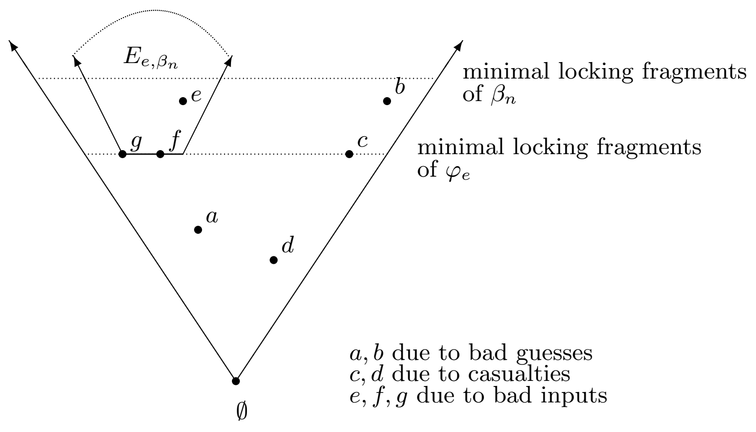
E_{e,β_n}
e
b
minimal locking fragments of β_n
g
f
c
minimal locking fragments of φ_e
a
d
a, b due to bad guesses
c, d due to casualties
e, f, g due to bad inputs
$\emptyset$

Fig. 3. Points on which α clips the computation of φ-program e.

1. **Bad Guess:** According to the algorithm, whenever e participates in lowering *ceiling*, the value of $guess(e)$ will be changed to a bigger one. Thus, only a finite number of bad guesses could have been made before $guess(e)$ becomes equal to or bigger than n. Therefore, only finitely many (σ, x)'s could be introduced into X_e due to bad guesses.
2. **Casualty:** For some $e' \in \mathbf{N}$, α tries to pick on the φ-program e' at (σ, x) by lowering *ceiling* in stages $\langle \sigma, x \rangle$, but the ceiling is too low so that the computation of φ-program e will be clipped by α as a casualty. One can see that such e' cannot be bigger than n, otherwise the guess for e' is always bigger than n and the ceiling will never get too low to for e. There

are only finitely many such φ-programs with indices smaller that n. For each such e', it does not matter whether $\varphi_{e'}$ is in $\mathbf{C}^a(\langle\beta_i\rangle)$, the guess for e' will become bigger and bigger until it gets a correct one or passes n. Moreover, since $\langle\beta_i\rangle$ is ascending, it follows that this φ-program e' introduces at most finitely many (σ, x)'s into X_e.

3. **Bad Inputs:** Suppose $(\sigma, x) \in X_e$ not because of the two reasons discussed above. It must be the case that $\sigma \subset f$ for some $(f, x) \in E_{e,\beta_n}$. One can see that (σ, x) must be a minimal locking fragment of φ_e, otherwise E_{e,β_n} cannot be compact in $\mathbb{T}(\varphi_e)$. Thus, $(\!(\sigma, x)\!)$ is a basic open set of E_{e,β_n}. By the assumption that E_{e,β_n} is compact in $\mathbb{T}(\varphi_e)$, we only have finitely many such (σ, x).

Therefore, X_e is finite. Next, we shall argue that, $\varphi_e \in \mathbf{C}^a(\alpha)$.

Claim 2.2: $\varphi_e \in \mathbf{C}^a(\alpha)$.
It is clear that if we clock e with α we have

$$E_{e,\alpha} = \left\{ (\!(\sigma, x)\!) \;\middle|\; (\sigma, x) \in X_e \right\}.$$

However, $E_{e,\alpha}$ may not be compact in $\mathbb{T}(\varphi_e)$ because there may be some $(\sigma, x) \in X_e$ that is not a locking fragment of φ_e due to a bad guess or casualty as discussed above. Let

$$P_e = \left\{ (\sigma, x) \;\middle|\; (\sigma, x) \in X_e \text{ and } (\sigma, x) \text{ is not a locking fragment of } \varphi_e. \right\}$$

Our goal is to show that we can patch φ-program e on those $(\sigma, x) \in P_e$ under the budget provided by α.
At first, we show that, for every $(\sigma, x) \in P_e$,

$$\alpha(\sigma, x) \geq \|\sigma\| + |x| + 2.$$

Note that α may not be useful. Suppose $(\sigma, x) \in P_e$ and, for some $i \in \mathbf{N}$,

$$\beta_i(\sigma, x) = \alpha(\sigma, x).$$

Since (σ, x) is not a locking fragment of φ_e, it follows that, for every $f \supset \sigma$, φ-program e on (f, x) must query the oracle outside $\mathsf{dom}(\sigma)$. Thus, β_n cannot converge at (σ, x), otherwise E_{e,β_n} is not compact in $\mathbb{T}(\varphi_e)$ under the answer-length-cost model. By the assumptions that $\langle\beta_i\rangle$ is uniformly convergent and that β_n does not converge at (σ, x), we know that neither does β_i converge at (σ, x). Since β_i is useful, it follows that

$$\beta_i(\sigma, x) = \alpha(\sigma, x) \geq \|\sigma\| + |x| + 2.$$

Similarly, for every $\eta \subseteq \sigma$, there is some $j \in \mathbf{N}$ such that,

$$\beta_j(\eta, x) = \alpha(\eta, x) \geq \|\eta\| + |x| + 2. \tag{10}$$

Let q be the first query outside $\mathsf{dom}(\sigma)$ made during the course of the computation of $\widetilde{\Phi}_e(\sigma, x)$. Clearly, for all $\tau \in \mathcal{F}$ such that $\mathsf{dom}(\tau) - \mathsf{dom}(\sigma) = \{q\}$, we have $(\tau, x) \in O_e$. Suppose $(\tau, x) \notin I_e$ and $\langle\tau, x\rangle = k$. We have

$$\widetilde{\Phi}_e(\tau, x) \leq \beta_{guess_k(e)}(\tau, x) \leq ceiling_k = \alpha(\tau, x).$$

Thus, we can have a φ-program e' to check $\sigma \subset f$ at the beginning of the computation on any $(f, x) \in \mathcal{T} \times \mathbf{N}$. If $\sigma \subset f$ is false, e' computes $F(f, x)$ exactly same as e does. If $\sigma \subset f$ is true, e' queries $[f(q) =?]$. This can be done by an OTM^a clocked by α because of (10), and in case the extra query $[f(q) =?]$ is needed, the budget, $\alpha(\sigma, x)$, must be enough to make the extra query because β_i does not converge at (σ, x) and β_i is useful. It is clear that

$$X_{e'} = X_e - \{(\sigma, x)\}.$$

Since P_e is finite, we can use the same idea to remove every $(\sigma, x) \in P_e$ and result in a patched φ-program p for φ_e such that,

$$E_{p,\alpha} = \left\{ (\!(\sigma, x)\!) \;\middle|\; (\sigma, x) \in (X_e - P_e) \right\}.$$

Clearly, $E_{p,\alpha}$ is compact in $\mathbb{T}(\varphi_p)$. Since $F = \varphi_e = \varphi_p$, it follows that $F \in \mathbf{C}^a(\alpha)$.

Claim 3: $\mathbf{C}^a(\alpha) \subseteq \bigcup_{n=o}^{\infty} \mathbf{C}^a(\beta_n)$.

Suppose $\varphi_n \notin \mathbf{C}^a(\langle\beta_i\rangle)$. It is easy to see that any guess for φ-program n cannot satisfy n forever, and thus α will lower down the ceiling to pick on φ-program n at infinitely many locking fragments of φ_n. Since X_n is not finite, we cannot patch φ-program n under a fixed cost in the same way discussed in Claim 2.2. Thus, $\varphi_n \notin \mathbf{C}^a(\alpha)$.

Since $\mathbf{C}^a(\langle\beta_i\rangle) = \bigcup_{n=o}^{\infty} \mathbf{C}^a(\beta_n)$, by Claims 2 and 3, we therefore have $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle\beta_i\rangle)$ □

Discussion Consider the proof of the theorem above. The construction for α does not guarantee that α is $\mathcal{F}$-monotone. It is possible that α shrinks on some $(\sigma, a) \in \mathcal{F} \times \mathbf{N}$ in order to pick on some φ-program e as in the following case. Suppose *guess*(e) contains an index i such that $\varphi_e \notin \mathbf{C}^a(\beta_i)$ and α needs to pick on e on input (σ, a). Moreover, on every (τ, a) with $\tau \subset \sigma$, α does not pick on e because e queries outside $\mathsf{dom}(\tau)$ and β_i's budget is enough for e to complete the queries. Thus, it is possible that α on some (τ, a) with $\tau \subset \sigma$ outputs a value bigger than $\beta_i(\sigma, a)$. Thus, we need to explicitly maintain a floor in order to prevent α from shrinking. However, such floor creates a difficulty in trying to prove $\mathbf{C}^a(\alpha) \subseteq \mathbf{C}^a(\langle\beta_i\rangle)$. This is because we have to argue that: If $F \notin \mathbf{C}^a(\langle\beta_i\rangle)$, then there is no φ-program e for F such that, every time when α tries to pick on e, e is protected by the floor. That is, we need to prove that there is no such φ-program e for F such that, for every stage $k = \langle\sigma, x\rangle$ when α tries to pick on e at (σ, x), the following situation occurs: floor $\geq \widetilde{\Phi}_e(\sigma, x) > \beta_{guess(e)}(\sigma, x)$.

At the moment we are skeptical about the existence of a way to fix this problem and hence we propose the following conjecture.

Conjecture 2 *There is a uniform, ascending, and strong $\langle\beta_i\rangle$ such that, if there is $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle\beta_i\rangle)$, then α is not strong.* ■

Corollary 2 *There exist a continuous functional $F : \mathcal{T} \times \mathbf{N} \to \mathbf{N}$ and a uniform, ascending, and useful $\langle \beta_i \rangle$ such that, for every $i \in \mathbf{N}$, $F_{\beta_i} \leq F$, and $\mathbf{C}^a(\langle \beta_i \rangle)$ is not a type-2 complexity class.*

Sketch of Proof: Note that the $\langle \beta_i \rangle$ given in the proof of Theorem 3 is not bounded. We modify it as follows: let $d \geq \mathsf{max}(c, c', a)$, where c, c', and a are the same constants in the arguments of the proof of Theorem 3. Define

$$\beta_i(\sigma, x) = \begin{cases} |x| + c & \text{if } x \geq i; \\ \|\sigma_{[x+2]}\| + |x| + d & \text{otherwise.} \end{cases}$$

It is clear that, for every $(f, x) \in \mathcal{T} \times \mathbf{N}$ and $i \in \mathbf{N}$, β_i is bounded as

$$F_{\beta_i}(f, x) \leq F(f, x) = \|f_{[x+2]}\| + |x| + d.$$

Then, we use the exact arguments in the proof of Theorem 3 except the definition for $\langle \beta_i \rangle$ to prove this corollary. $\square$

Corollary 3 *Let $\beta \in \mathbf{T}_2\mathbf{TB}$. If β is locking detectable and useful, then there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^a(\alpha) = \mathbf{O}(\beta)$.*

Proof: Let $c, d \in \mathbf{N}$ and $i = \mathsf{max}(c, d)$. By the basic hierarchy, we have $\mathbf{C}^a(c\beta + d) \subseteq \mathbf{C}^a(i\beta + i)$. Thus,

$$\mathbf{O}(\beta) = \mathbf{C}^a(\langle \beta_i \rangle), \text{ where } \beta_i = i\beta + i.$$

Clearly, $\langle \beta_i \rangle$ is uniform and ascending. Suppose β is locking detectable and useful. Then, for every $i > 0$, $i\beta + i$ is also useful. Moreover, if ℓ is a locking detector for β, then ℓ is a locking detector for $\langle \beta_i \rangle$. Thus, $\langle \beta_i \rangle$ is strongly convergent. By Theorem 5, there is an $\alpha \in \mathbf{T}_2\mathbf{TB}$ such that $\mathbf{C}^a(\alpha) = \mathbf{C}^a(\langle \beta_i \rangle) = \mathbf{O}(\beta)$. $\square$

Corollary 4 *Let $\alpha, \beta \in \mathbf{T}_2\mathbf{TB}$. If α and β are locking detectable and useful, then $\mathbf{O}(\alpha + \beta)$ is a type-2 complexity class.*

Sketch of Proof: Let ℓ_α, ℓ_β be locking detectors of α and β, respectively. For every $(\sigma, x) \in \mathcal{F} \times \mathbf{N}$, define

$$\gamma(\sigma, x) = \alpha(\sigma, x) + \beta(\sigma, x).$$

Clearly, γ is a useful type-2 time bound if both α and β are. For locking detectability, define $\ell : \mathcal{F} \times \mathbf{N} \to \{0, 1\}$ by

$$\ell(\sigma, x) = \begin{cases} 1 & \text{if } \ell_\alpha(\sigma, x) = \ell_\beta(\sigma, x) = 1; \\ 0 & \text{otherwise.} \end{cases}$$

Thus, ℓ is a locking detector for γ. By Corollary 3, $\mathbf{O}(\alpha + \beta)$ is a type-2 complexity class. $\square$