Compaction of Church Numerals

Isamu Furuya ^{*,†} and Takuya Kida [†]

Graduate School/Faculty of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Sapporo 060-0814, Japan

* Correspondence: furuya@ist.hokudai.ac.jp; Tel.: +81-11-706-7680

† These authors contributed equally to this work.

Received: 28 June 2019; Accepted: 5 August 2019; Published: 8 August 2019

Abstract: In this study, we address the problem of compaction of Church numerals. Church numerals are unary representations of natural numbers on the scheme of lambda terms. We propose a novel decomposition scheme from a given natural number into an arithmetic expression using tetration, which enables us to obtain a compact representation of lambda terms that leads to the Church numeral of the natural number. For natural number n , we prove that the size of the lambda term obtained by the proposed method is $\mathcal{O}((\text{slog}_2 n)^{(\log n / \log \log n)})$. Moreover, we experimentally confirmed that the proposed method outperforms binary representation of Church numerals on average, when n is less than approximately 10,000.

Keywords: loss-less compression; lambda term; higher-order compression

1. Introduction

The goal of this study is to obtain a compact lambda term (λ -term) that leads to the Church numeral of a given natural number. Church numerals are unary representations of natural numbers on λ -terms. Let $\mathcal{C}(n)$ be the Church numeral for natural number n ; then $\mathcal{C}(n)$ is as follows:

$$\mathcal{C}(n) = \lambda f. \lambda x. \overbrace{f (f \cdots (f x) \cdots)}^n.$$

Namely, the length of the Church numeral increases linearly with n . We want a more compact representation for n in the scheme of λ -terms. We refer to this task as the *compaction of Church numerals*. Using a binary representation for n is a natural idea, which enables to obtain a λ -term whose size is $\Theta(\log n)$. Another idea is to decompose n into an arithmetic expression and to represent it as a corresponding λ -term of the expression. This may reduce the size of the λ -term. For example, $n = 500$ can be decomposed into $5 \times 10 \times 10$. The λ -term corresponding to this expression is given as $(\lambda p. \lambda q. \lambda f. \lambda x. p (q (q f)) x) \mathcal{C}(5) \mathcal{C}(10)$, which is much smaller than $\mathcal{C}(500)$. In this idea, *tetration* can be used for the decomposition. Tetration is the next hyperoperation after exponentiation; it is defined as iterated exponentiation, e.g., the third tetration of two $3^2 = 2^{2^2} = 16$. Since a function for calculating ${}^i k$ on Church numerals is written as a λ -term whose size is $\mathcal{O}(i + k)$, the size of the λ -term for $n = {}^i k$ achieves $\mathcal{O}(\log n)$.

In this paper, we propose the *Recursive tetration partitioning* (RTP) method to decompose a natural number using tetration. We also present an algorithm to generate a compact λ -term by using RTP. Moreover, we prove that the size of the obtained λ -term is $\mathcal{O}((\text{slog}_2 n)^{\log n / \log \log n})$, where $\text{slog}_2 n$ is super-logarithm, which is the inverse operation of tetration. It is slightly worse than $\mathcal{O}(\log n)$, but we also prove that the size of the λ -term achieves $\mathcal{O}(\text{slog}_2 n)$ in some cases, and our experimental result shows that our method outperforms a binary representation on λ -term on average, when n is less than approximately 10,000.

The compaction of Church numerals can be applied to data compression. Kobayashi et al. [1] proposed a compression method called higher-order compression that uses extended λ -terms as the data model. Their method translates an input data to a compact λ -term and then encodes the obtained λ -term. When a pattern consecutively appears n times in the input, such a part can be represented as a λ -term with $\mathcal{C}(n)$. Thus efficient compaction of $\mathcal{C}(n)$ will help for higher-order compression when the data contains many repetitive patterns.

Yaguchi et al. [2] recently proposed an efficient algorithm for higher-order compression. They utilized a simply typed λ -term for efficient modeling and encoding. Differing from Kobayashi et al.'s method where each context occurring more than once in an input is extracted, Yaguchi et al.'s method extracts the most frequent context up to a certain size. They state in [2] that their method often achieves better performance than a grammar compression [3] with regard to compression ratio. Of course, their method can also be applied to the compaction of $\mathcal{C}(n)$, while it takes time for repeating context extraction. We confirmed via experiments that our method tends to produce more compact λ -terms for Church numerals than Yaguchi et al.'s method. Note that our method can be easily incorporated into Yaguchi et al.'s method.

Contributions: The primary contributions of this study are as follows.

1. For natural numbers, we propose a novel decomposition scheme called RTP, which enables to obtain a compact representation of λ -terms that leads to the Church numerals of the numbers.
2. By incorporating RTP, we propose an algorithm to perform the compaction of $\mathcal{C}(n)$ for natural number n . Moreover, we prove that the size of the λ -terms constructed by the algorithm is $\mathcal{O}((\text{slog}_2 n)^{\log n / \log \log n})$.
3. We implemented the proposed algorithm and conducted comparative experiments. With regard to the sizes of the obtained λ -terms, the results show that the method is superior to that of Yaguchi et al. [2] and is also superior to a binary representation on λ -term on average, when n is less than approximately 10,000.

Related studies: Better modeling for an input data leads better compression ratio. Since it is rare that we know the exact model for the input well beforehand, we usually use a universal source coding [4] or assume an appropriate model on the input to express the structure inherent in it. Kieffer and Yang [3] proposed *grammar compression* that uses a context-free grammar as the model. Grammar compression is a compression scheme that translates an input text into a set of context-free grammar rules, before encoding the rules. In grammar compression, a smaller set of rules gives a better compression ratio. Finding the optimal grammar is an NP-hard problem [5] but several excellent heuristic algorithms have been proposed for its solution [6–11]. Obtaining a smaller grammar is of great importance since most existing algorithms that work on grammar-compressed texts have running time dependent on the grammar sizes (see the excellent surveys e.g., [12,13]). Due to information theory arguments, any compression cannot be better than a binary encoding on all numbers. However, as stated in [1] and the above, higher-order compression can be better for a certain subset of numbers, i.e., for $n = i^k$.

Composition: The rest of this paper is organized as follows. In Section 2, we review tetration, lambda notation, and Church numerals. In Section 3, we define the proposed RTP method and present the translation algorithm using RTP. We also prove the upper bound of the size of the λ -term produced by our algorithm. In Section 4, we discuss application to higher-order compression and present our experimental result. Conclusions are presented in Section 5.

Previously, one page abstract of this work has been published in the proceedings of Data Compression Conference 2016 [14]. This paper is the full version article of [14].

2. Preliminaries

2.1. Tetration and Super-Logarithm

Definition 1 (Tetration). For natural numbers φ and t , the t -th tetration of φ , denoted by ${}^t\varphi$, is recursively defined as follows:

$${}^t\varphi := \begin{cases} 1 & (\text{if } t = 0), \\ \varphi^{({}^{t-1}\varphi)} & (\text{otherwise}). \end{cases}$$

For example, ${}^12 = 2^{02} = 2^1 = 2$, ${}^22 = 2^{12} = 2^2 = 4$, ${}^32 = 2^{22} = 2^4 = 16$, and ${}^42 = 2^{32} = 2^{16} = 65536$. The following corollary is easily induced from Definition 1.

Corollary 1. For natural numbers φ and t , it holds that $\log_{\varphi} {}^t\varphi = {}^{t-1}\varphi$.

Definition 2 (Super-logarithm). Super-logarithm, denoted by slog , is the inverse operation of tetration, defined as $\text{slog}_{\varphi} ({}^t\varphi) = t$ with natural numbers φ and t .

The iterated logarithm of n , denoted by $\log^* n$, is also known as the number of times the logarithm function must be applied to n before it becomes less than or equal to 1. For positive numbers, the super-logarithm is essentially equivalent to the iterated logarithm, i.e., it holds that $\log^* n = \lceil \text{slog}_e n \rceil$ for any $n > 0$. However, note that in the $\mathcal{O}(\cdot)$ -notation the size of super-logarithm depends on its base, that is, $\mathcal{O}(\text{slog}_a n) \neq \mathcal{O}(\text{slog}_b n)$ if $a \neq b$.

2.2. Lambda Terms

Definition 3 (Lambda terms (λ -terms)). Let x be a variable. Then, lambda term (λ -term) M is defined inductively as follows:

$$M ::= x \mid \lambda x.M \mid M_1 M_2.$$

We call λ -terms formed $\lambda x.M$ and $M_1 M_2$ λ -abstraction and functional application, respectively. Let V_1 and V_2 be the sets of the variables occurring in M_1 and M_2 , respectively, then, we identify M_1 with M_2 if there is an isomorphism from V_1 to V_2 . $M[x_1 := M_1, x_2 := M_2, \dots]$ denotes the λ -term such that all occurrences of $x_1, x_2, \dots$ in M are replaced by $M_1, M_2, \dots$, respectively.

We show in parentheses “(” and “)” the precedence of combining λ -terms. However, for simplicity, we omit these parentheses based on standard omission rules, that is, we consider that functional applications are left-associative and prior to λ -abstractions. For example, we consider $M_1 M_2 M_3$ as $(M_1 M_2) M_3$ rather than $M_1 (M_2 M_3)$, and consider $\lambda x.M_1 M_2$ as $\lambda x.(M_1 M_2)$ rather than $(\lambda x.M_1) M_2$.

Definition 4 (β -reduction). Let x be a variable, and M_1 and M_2 be λ -terms. Then, β -reduction is a relation of λ -terms, denoted by using $\longrightarrow_{\beta}$, such that $(\lambda x.M_1) M_2 \longrightarrow_{\beta} M_1[x := M_2]$. We write $\longrightarrow_{\beta}^*$ for the reflexive transitive closure of $\longrightarrow_{\beta}$.

For example, $(\lambda x.M_1 x x) M_2 \longrightarrow_{\beta} M_1 M_2 M_2$ because the right-side of $\longrightarrow_{\beta}$ is $(M_1 x x)[x := M_2]$. Similarly, $(\lambda x.\lambda y.x y y) M_1 M_2 \longrightarrow_{\beta}^* M_1 M_2 M_2$ because $(\lambda x.\lambda y.x y y) M_1 M_2 \longrightarrow_{\beta} (\lambda y.M_1 y y) M_2 \longrightarrow_{\beta} M_1 M_2 M_2$.

We define the sizes of λ -terms as follows. The definition can be found in [1].

Definition 5 (Sizes of λ -terms). Let x be a variable and M be a λ -term. Then, we denote the size of the λ -term by $\#M$, and inductively define the size of each λ -term as follows:

$$\#x = 1, \quad \#(\lambda x.M) = \#M + 1, \quad \#(M_1 M_2) = \#M_1 + \#M_2 + 1.$$

For λ -terms M_1 and M_2 , we say that M_2 is more *compact* than M_1 if $\#M_2 < \#M_1$. *Compaction* of a λ -term M_1 means to find a λ -term M_2 such that $M_1 N \rightarrow_{\beta}^* M'$ and $M_2 N \rightarrow_{\beta}^* M'$ for every N with a λ -term M' , and $\#M_2 < \#M_1$.

In this paper, we assume Word RAM as the computational model. Thus, we ignore the size of names and pointers when we discuss size and space.

2.3. Church Numerals

It is known that natural numbers are represented as *Church numerals* on λ -terms. Also, some λ -terms enabling arithmetic operations on Church numerals are provided.

Definition 6 (Church numerals). For natural number n , its Church numeral, denoted by $\mathcal{C}(n)$, is

$$\mathcal{C}(n) := \lambda f. \lambda x. \overbrace{f (f \cdots (f x) \cdots)}^n.$$

Corollary 2 (Arithmetic operations on Church numerals). Let n_1 and n_2 be natural numbers. Then we obtain three λ -terms $\text{Add}(n_1, n_2)$, $\text{Mul}(n_1, n_2)$, and $\text{Tet}(n_1, n_2)$, such that $\text{Add}(n_1, n_2) \rightarrow_{\beta}^* \mathcal{C}(n_1 + n_2)$, $\text{Mul}(n_1, n_2) \rightarrow_{\beta}^* \mathcal{C}(n_1 \cdot n_2)$, and $\text{Tet}(n_1, n_2) \rightarrow_{\beta}^* \mathcal{C}(n_2^{n_1})$, respectively, as follows:

$$\begin{aligned} \text{Add}(n_1, n_2) &= (\lambda p. \lambda q. \lambda f. \lambda x. p f (q f x)) \mathcal{C}(n_1) \mathcal{C}(n_2), \\ \text{Mul}(n_1, n_2) &= (\lambda p. \lambda q. \lambda f. \lambda x. p (q f x) \mathcal{C}(n_1) \mathcal{C}(n_2)), \\ \text{Tet}(n_1, n_2) &= (\lambda p. \lambda f. \lambda x. \overbrace{p p \cdots p}^{n_2} f x) \mathcal{C}(n_1). \end{aligned}$$

As can be seen in Corollary 2, in each λ -term, one λ -abstraction occurs first, and one or two Church numerals follow it. We call the former λ -abstraction and the following Church numerals *function term* and *argument terms*, respectively. For example, for $\text{Add}(n_1, n_2)$, $(\lambda p. \lambda q. \lambda f. \lambda x. p f (q f x))$ is the function term and the following $\mathcal{C}(n_1)$ and $\mathcal{C}(n_2)$ are the argument terms.

2.4. Binary Expression of Natural Numbers on λ -Terms

Instead of Church numerals, we can represent natural numbers on λ -terms based on binary representation as follows [15].

Definition 7 (Binary representation on λ -terms). For natural number n , assume that its binary representation is $B_k B_{k-1} \cdots B_2 B_1$ with $B_i \in \{0, 1\}$ for $1 \leq i \leq k$. Then, the binary expression of n on λ -terms, denoted by $\mathcal{B}(n)$, is

$$\mathcal{B}(n) := b_{B_1} (b_{B_2} \cdots (b_{B_{k-1}} (b_{B_k} (\lambda x. x))) \cdots),$$

where $b_0 = \lambda p. \lambda f. \lambda x. p f (p f x)$ and $b_1 = \lambda p. \lambda f. \lambda x. f (p f (p f x))$.

For example, the binary expression of 57 is 111001, then its binary expression on λ -terms is $b_1 (b_0 (b_0 (b_1 (b_1 (b_1 (\lambda x. x))))))$.

Note that $\mathcal{B}(n) \rightarrow_{\beta}^* \mathcal{C}(n)$ and $\#\mathcal{B}(n) = \Theta(\log n)$ follows Definition 7, while $\#\mathcal{C}(n) = \Theta(n)$.

3. Proposed Method

3.1. Basic Idea

Let n be a natural number. As stated in Section 1, we can reduce $\#\mathcal{C}(n)$ by using decomposition of n into an arithmetic expression. For the example, for $n = 500$, we can decompose it to an arithmetic expression $5 \cdot 10 \cdot 10$ (more precisely, $(5 \cdot 10) \cdot 10$). A corresponding λ -term M of the expression is

obtained as $M = (\lambda p.\lambda q.\lambda f.\lambda x.p (q f) x) ((\lambda p.\lambda q.\lambda f.\lambda x.p (q f) x) \mathcal{C}(5) \mathcal{C}(10)) \mathcal{C}(10)$, and its size is 85, while $\# \mathcal{C}(500) = 1003$. Here, $M \rightarrow_{\beta}^* \mathcal{C}(500)$. Moreover, we can make M more compact by combining two λ -abstractions into a single λ -abstraction, such as $(\lambda p.\lambda q.\lambda f.\lambda x.p (q (q f)) x) \mathcal{C}(5) \mathcal{C}(10)$. This also generates $\mathcal{C}(500)$ through β -reduction similar to M , while its size is just 51.

We perform compaction of $\mathcal{C}(n)$ in the following three steps.

Step 1: Decompose n into a special form of arithmetic expression, called TAE.

Step 2: Translate the arithmetic expression into a λ -term having the form of $M \mathcal{C}(\varphi^*)$.

Step 3: Apply the above Step 1 and 2 for the natural number φ^* recursively.

For Step 1, we introduce TAEs in Section 3.2. For Step 2, we describe a translation method in Section 3.3. There are many ways to achieve decomposition of a natural number into a TAE, and the size of the translated λ -term changes depending on the expression. Then, we consider how we effectively obtain a TAE such that the translated λ -term of the TAE is compact. We propose RTP in Section 3.4, which is a heuristic approach decomposing a natural number into a TAE. Finally, for Step 3, we discuss further recursive compaction of Church numerals in Section 3.5.

3.2. Tetrational Arithmetic Expression (TAE)

We define *tetrational arithmetic expression (TAE)*, denote by E , as a special form of arithmetic expression.

Definition 8 (Tetrational arithmetic expressions (TAEs)). *Let N be an arbitrary natural number. Then, tetrational arithmetic expression (TAE) E is inductively defined as follows:*

$$E ::= N \mid E + E \mid E \cdot E \mid {}^{N_2}N_1.$$

If the evaluated result of a TAE E is a natural number n , we write $E[n]$ for the TAE. Especially, let E_{φ} denote a restricted form of TAE with a natural number φ , defined as

$$E_{\varphi} ::= \varphi \mid E_{\varphi} + E_{\varphi} \mid E_{\varphi} \cdot E_{\varphi} \mid {}^t\varphi,$$

where t is an arbitrary natural number.

For example, $2 \cdot 3 + {}^22$ is a TAE, which is written as $E[10]$ because its evaluated result is equal to 10. Similarly, ${}^22 \cdot 2 + 2$ is also a TAE written as $E_2[10]$.

Informally speaking, reducing the number of natural numbers occurring in TAEs is effective for compaction of corresponding λ -terms, since it enables to reduce the number of the kinds of argument terms in the λ -terms; note that we can transform $(\lambda p.\lambda q.\lambda f.\lambda x.p f (q f x)) \mathcal{C}(n_1) \mathcal{C}(n_1)$ to $(\lambda p.\lambda f.\lambda x.p f (p f x)) \mathcal{C}(n_1)$, for example. From this point of view, it is expected that the corresponding λ -term of $E_{\varphi}[n]$ becomes compact when n is a multiple of φ . Let $r = n \bmod \varphi$ and $\bar{n} = n - r$. Thus, we consider TAE $E_{\varphi}[\bar{n}] + r$ instead of $E_{\varphi}[n]$ itself.

3.3. Translation from TAE to λ -Term

Here, we consider how we translate an TAE $E_{\varphi}[\bar{n}]$ to a corresponding λ -term. A simple method is to translate each arithmetic operation occurring in $E_{\varphi}[\bar{n}]$ by using λ -terms stated in Corollary 2. For example, for $E_2[200] = {}^32 \cdot ({}^22 \cdot 2 + {}^22) + {}^22 \cdot 2$, the translated λ -term is obtained as

$$\text{Add}(\text{Mul}(\text{Tet}(2, 3), \text{Add}(\text{Mul}(\text{Tet}(2, 2), 2), \text{Tet}(2, 2))), \text{Mul}(\text{Tet}(2, 2), 2)).$$

As seen above, the corresponding λ -term seems to become long. However, we can translate the TAE to a compact λ -term.

Definition 9 (Corresponding functional λ -terms (CFLT)). *Let n and φ be natural numbers. We say a λ -term M is a corresponding functional λ -term (CFLT) of TAE $E[n]$ if both of the following conditions hold for M :*

1. M has the form of $(\lambda p.\lambda f.\lambda x.\hat{M}) \mathcal{C}(\varphi)$, where $\hat{M}$ is a λ -term that contains no Church numerals.
2. $M \rightarrow_{\beta}^* \mathcal{C}(n)$.

For a given TAE E , We denote a CFLT of E by $\Lambda(E)$.

For example, for $E_2[200] = {}^3 2 \cdot ({}^2 2 \cdot 2 + {}^2 2) + {}^2 2 \cdot 2$, one of its CFLTs is

$$\Lambda(E_2[200]) = (\lambda p.\lambda f.\lambda x.p p p (\lambda y.p p (\lambda z.p f z) (p p f y)) (p p (\lambda w.p f w) x)) \mathcal{C}(2). \quad (1)$$

The size of the above (1) is 48, which is much smaller than $\#C(200) = 403$.

For TAE E_{φ} , the following lemma holds.

Lemma 1. For any TAE E_{φ} , its CFLT exists.

Proof. We prove the lemma by induction. Considering λ -terms stated in Corollary 2, for $E_{\varphi} = \varphi$ and $E_{\varphi} = {}^t \varphi$, their CFLTs are obtained as

$$\begin{aligned} \Lambda(\varphi) &= (\lambda p.\lambda f.\lambda x.p f x) \mathcal{C}(\varphi) \text{ and} \\ \Lambda({}^t \varphi) &= (\lambda p.\lambda f.\lambda x.\overbrace{p p \cdots p}^t f x) \mathcal{C}(\varphi), \end{aligned}$$

respectively.

Let $E_{\varphi}^{(1)}$ and $E_{\varphi}^{(2)}$ be TAEs. Then, we assume that there are CFLTs of $E_{\varphi}^{(1)}$ and $E_{\varphi}^{(2)}$. Moreover, by Definition 9, we assume that $\Lambda(E_{\varphi}^{(1)}) = (\lambda p.\lambda f.\lambda x.\hat{M}^{(1)}) \mathcal{C}(\varphi)$ and $\Lambda(E_{\varphi}^{(2)}) = (\lambda p.\lambda f.\lambda x.\hat{M}^{(2)}) \mathcal{C}(\varphi)$. Here, CFLTs of $E_{\varphi} + E_{\varphi}$ and $E_{\varphi} \cdot E_{\varphi}$ are obtained as

$$\begin{aligned} \Lambda(E_{\varphi}^{(1)} + E_{\varphi}^{(2)}) &= (\lambda p.\lambda f.\lambda x.(\lambda y.\hat{M}^{(1)}[x := y]) ((\lambda y.\hat{M}^{(2)}[x := y]) x)) \mathcal{C}(\varphi) \text{ and} \\ \Lambda(E_{\varphi}^{(1)} \cdot E_{\varphi}^{(2)}) &= (\lambda p.\lambda f.\lambda x.(\lambda g.\lambda y.\hat{M}^{(1)}[f := g, x := y]) (\lambda y.\hat{M}^{(2)}[x := y]) x) \mathcal{C}(\varphi), \end{aligned}$$

respectively. □

For compaction, next we introduce *simplification* of λ -terms.

Definition 10 (Simplification of λ -terms). Let x be a variable, M_1 and M_2 be λ -terms, and $\rightarrow_{\beta}$ be a special β -reduction from $(\lambda x.M_1) M_2$ to $M_1[x := M_2]$ which defined only if at least one of the following holds:

1. x occurs in M_1 only once (in other words, x is linear in M_1),
2. $\#M_2 = 1$.

Then, simplification is reflexive transitive closure of $\rightarrow_{\beta}$.

By using simplification for $(\lambda x.M_1) M_2$, we can reduce the size of the λ -term since $\#M_1[x := M_2] = \#M_1 + \#M_2 - 1$ (if x occurs in M_1 only once) or $\#M_1[x := M_2] = \#M_1$ (if $\#M_2 = 1$) holds, while $\#((\lambda x.M_1) M_2) = \#M_1 + \#M_2 + 2$.

From the above discussion, we can translate a given TAE E_{φ} into a compact λ -term by applying simplification after translating E_{φ} into the CFLT in the manner of the proof of Lemma 1. We show the translating algorithm in Algorithm 1. The time complexity of Algorithm 1 depends on the total number of the occurrences of addition and multiple in the input TAE.

Algorithm 1 Translation from E_φ to its CFLT.**Input:** E_φ **Output:** $\Lambda(E_\varphi)$

```

1: function TRANSLATE( $E_\varphi$ )
2:   if  $E_\varphi = \varphi$  then
3:     return  $(\lambda p.\lambda f.\lambda x.p\ f\ x)$ 
4:   else if  $E_\varphi = {}^t\varphi$  then
5:     return  $(\lambda p.\lambda f.\lambda x.\overbrace{p\ p\ \cdots\ p}^t\ f\ x)$ 
6:   else if  $E_\varphi = E_\varphi^{(1)} + E_\varphi^{(2)}$  then
7:      $(\lambda p.\lambda f.\lambda x.\hat{M}^{(1)}) := \text{TRANSLATE}(E_\varphi^{(1)})$ 
8:      $(\lambda p.\lambda f.\lambda x.\hat{M}^{(2)}) := \text{TRANSLATE}(E_\varphi^{(2)})$ 
9:     return  $(\lambda p.\lambda f.\lambda x.(\lambda y.\hat{M}^{(1)}[x := y])((\lambda y.\hat{M}^{(2)}[x := y])\ x))$ 
10:  else if  $E_\varphi = E_\varphi^{(1)} \cdot E_\varphi^{(2)}$  then
11:     $(\lambda p.\lambda f.\lambda x.\hat{M}^{(1)}) := \text{TRANSLATE}(E_\varphi^{(1)})$ 
12:     $(\lambda p.\lambda f.\lambda x.\hat{M}^{(2)}) := \text{TRANSLATE}(E_\varphi^{(2)})$ 
13:    return  $(\lambda p.\lambda f.\lambda x.(\lambda g.\lambda y.\hat{M}^{(1)}[f := g, x := y])\ (\lambda y.\hat{M}^{(2)}[x := y])\ x)$ 
14:  end if
15: end function
16:
17:  $M := \text{TRANSLATE}(E_\varphi)$ 
18:  $M^* :=$  the  $\lambda$ -term generated by simplification of  $M$ 
19: return  $(M^* \mathcal{C}(\varphi))$ 

```

3.4. Recursive Tetrational Partitioning (RTP)

We consider how we obtain an effective $E_\varphi[\bar{n}]$ for given $\bar{n}$ and φ in the sense that the CFLT becomes more compact. $E_\varphi[\bar{n}]$ is more effective if it consists of fewer arithmetic operations because the size of the CFLT increases with the number of arithmetic operations occurring in it. For example, for $\bar{n} = 12$ and $\varphi = 3$, $3 \cdot 3 \cdot 3 + 3$ is more effective than $3 + 3 + 3 + 3$. We propose *recursive tetrational partitioning (RTP)*, which is a heuristic method to find such an effective TAE for given $\bar{n}$ and φ .

Definition 11 (Recursive tetrational partitioning (RTP)). Let n and $\varphi \geq 2$ be natural numbers, and $r = n \bmod \varphi$. Then, recursive tetrational partitioning (RTP) is a method of decomposing n into $E_\varphi[n - r] + r$. Let $T_\varphi[n]$ denote a TAE generated by RTP, then RTP recursively decompose n in the manner described as follows:

$$T_\varphi[n] := \begin{cases} n & (\text{if } n \leq \varphi), \\ \begin{aligned} & {}^k\varphi \cdot T_\varphi[p_k - r_k] + (\overbrace{{}^k\varphi + \cdots + {}^k\varphi}^{r_k}) \\ & + \cdots + {}^1\varphi \cdot T_\varphi[p_1 - r_1] + (\overbrace{{}^1\varphi + \cdots + {}^1\varphi}^{r_1}) + r \end{aligned} & (\text{otherwise}), \end{cases}$$

where k is the maximum natural number such that ${}^k\varphi \leq n$, and for $1 \leq i \leq k$, p_i is the natural number such that $0 \leq p_i \cdot {}^i\varphi < {}^{i+1}\varphi$ and $r_i = p_i \bmod \varphi$. Here, if $p_i - r_i = 0$ or 1 , we omit the corresponding terms or coefficients in the generated TAE, respectively.

For example, for $n = 201$ and $\varphi = 2$, $T_\varphi[n] = T_2[201] = {}^3 2 \cdot ({}^2 2 \cdot 2 + {}^2 2) + {}^2 2 \cdot 2 + 1$. In Definition 11, note that each coefficient $(p_i - r_i)$ of ${}^i \varphi$ is a multiple of φ , and then, each $T_\varphi[p_i - r_i]$ generates just $E_\varphi[p_i - r_i]$ without any remainder.

Theorem 1. For given natural numbers n and $\varphi \geq 2$, $T_\varphi[n]$ is uniquely determined.

Proof. We prove the statement by induction. For $n \leq \varphi$, the statement clearly holds. Otherwise, let k be the maximum natural number such that ${}^k \varphi \leq n$, $\bar{n} = n - (n \bmod \varphi)$, and $P = \{p_k, p_{k-1}, \dots, p_1\}$ be a set such that

$$\sum_{i=1}^k p_i \cdot {}^i \varphi = \bar{n}. \quad (2)$$

Then, it is sufficient to prove that P is uniquely determined for $\bar{n}$ since k , $\bar{n}$, and each r_i are uniquely determined from n , φ , and P . By Definition 11, $0 \leq p_i \cdot {}^i \varphi < {}^{i+1} \varphi$ holds for $1 \leq i \leq k$. We assume that p_k is not the maximum natural number such that $p_k \cdot {}^k \varphi \leq \bar{n}$, then

$$\bar{n} - p_k \cdot {}^k \varphi \geq {}^k \varphi + (\bar{n} \bmod {}^k \varphi) \quad (3)$$

holds. On the other hand,

$$\sum_{i=1}^{k-1} p_i \cdot {}^i \varphi < \sum_{i=2}^k {}^i \varphi \leq {}^k \varphi + (\bar{n} \bmod {}^k \varphi) \quad (4)$$

also holds. By (3) and (4), $\sum_{i=1}^k p_i \cdot {}^i \varphi < \bar{n}$ holds. However, this contradicts to (2). Therefore, p_k must be the maximum one. Similar discussion holds for $p_{k-1}, p_{k-2}, \dots, p_2$, that is, for $2 \leq i \leq k-1$, p_i is the maximum natural number such that $p_i \cdot {}^i \varphi \leq \bar{n} \bmod {}^{i+1} \varphi$. By (2), p_1 is unique if $p_k, p_{k-1}, \dots, p_2$ are unique. Hence, P is uniquely determined for $\bar{n}$ and k . $\square$

Theorem 2. For given natural numbers n and $\varphi \geq 2$, there exists a CFLT of $T_\varphi[n]$.

Proof. By Definition 11, $T_\varphi[n]$ has form of $E_\varphi[\bar{n}] + r$, where $r = n \bmod \varphi$ and $\bar{n} = n - r$. By Lemma 1, for $E_\varphi[\bar{n}]$, there is a CFLT $\Lambda(E_\varphi[\bar{n}]) = (\lambda p. \lambda f. \lambda x. \hat{M}) \mathcal{C}(\varphi)$. Then, a CFLT of $T_\varphi[n]$ is obtained as

$$\Lambda(T_\varphi[n]) = (\lambda p. \lambda f. \lambda x. \hat{M}[x := \overbrace{(f(f \cdots (f x) \cdots))}^r]) \mathcal{C}(\varphi).$$

$\square$

For the size of $\Lambda(T_\varphi[n])$, the following two lemmas hold.

Lemma 2. For a given $T_\varphi[n]$ with natural numbers n and $\varphi \geq 2$, let N_a and N_m be the numbers of addition and multiplication occurring in $T_\varphi[n]$, respectively. Also, for tetration ${}^{n_2} n_1$, we call n_2 second argument of tetration, then let N_t be the sum of second arguments of all tetration occurring in $T_\varphi[n]$. Then,

$$\# \Lambda(T_\varphi[n]) \leq 14N_a + 8N_m + 2N_t + 2r + 2\varphi + 6$$

holds with $r = n \bmod \varphi$.

Proof. By Definition 9, we assume that $\Lambda(E_\varphi^{(i)}) = (\lambda p.\lambda f.\lambda x.\hat{M}^{(i)}) \mathcal{C}(\varphi)$. Then, the following holds:

$$\begin{aligned} \#\Lambda({}^t\varphi) &= \#((\lambda p.\lambda f.\lambda x.\overbrace{p \ p \ \cdots \ p}^t f \ x) \mathcal{C}(\varphi)) \\ &= 2t + 7 + \#\mathcal{C}(\varphi), \end{aligned} \quad (5)$$

$$\begin{aligned} \#\Lambda({}^t\varphi \cdot E_\varphi^{(1)}) &= \#((\lambda p.\lambda f.\lambda x.\overbrace{p \ p \ \cdots \ p}^t (\lambda y.\hat{M}^{(1)}[x := y]) \ x) \mathcal{C}(\varphi)) \\ &= \#\hat{M}^{(1)} + 2t + 7 + \#\mathcal{C}(\varphi), \end{aligned} \quad (6)$$

$$\begin{aligned} \#\Lambda(E_\varphi^{(1)} + E_\varphi^{(2)}) &= \#((\lambda p.\lambda f.\lambda x.(\lambda y.\hat{M}^{(1)}[x := y]) ((\lambda y.\hat{M}^{(2)}[x := y]) \ x)) \mathcal{C}(\varphi)) \\ &= \#\hat{M}^{(1)} + \#\hat{M}^{(2)} + 9 + \#\mathcal{C}(\varphi). \end{aligned} \quad (7)$$

By Definition 11, $T_\varphi[n]$ has the form of $E_\varphi + r$, where E_φ is TAE inductively defined as $E_\varphi ::= {}^t\varphi \mid {}^t\varphi \cdot E_\varphi \mid E_\varphi + E_\varphi$ with natural number t . Therefore, by Equations (5), (7), and (8),

$$\begin{aligned} \#\Lambda(E_\varphi) &\leq 6N_a + 4N_m + 2N_t + 3 + 4 \sum_i \hat{M}^{(i)} + \#\mathcal{C}(\varphi) \\ &= 6N_a + 4N_m + 2N_t + 3 + 4(2N_a + N_m) + \#\mathcal{C}(\varphi) \\ &= 14N_a + 8N_m + 2N_t + 3 + \#\mathcal{C}(\varphi). \end{aligned} \quad (8)$$

holds. Note that, in (5), (7), and (8), the constant size 3 arise from $\lambda p.M$, $\lambda f.M$, and functional application of λ -abstraction and Church numeral $\mathcal{C}(\varphi)$, which do not appear in higher level λ -term as $\hat{M}^{(i)}$. Also note that $\overbrace{p \ p \ \cdots \ p}^t$ occurring in (6) is the λ -term generated by simplification. Then,

$$\begin{aligned} \#\Lambda(T_\varphi[n]) &\leq 14N_a + 8N_m + 2N_t + 2r + 3 + \#\mathcal{C}(\varphi) \\ &= 14N_a + 8N_m + 2N_t + 2r + 2\varphi + 6 \end{aligned}$$

follows (9). $\square$

Lemma 3. For a given natural number $n > 8$, there is a natural number φ such that $2 \leq \varphi \leq \lfloor \sqrt{n} \rfloor$ and $\#\Lambda(T_\varphi[n]) < \#\mathcal{C}(n)$.

Proof. Let $\varphi = \lfloor \sqrt{n} \rfloor$ and $r = n \bmod \varphi$. By Definition 11, $T_\varphi[n]$ has the form of $\varphi \cdot \varphi + r$. Then, since $r < \varphi$,

$$\begin{aligned} \#\Lambda(T_\varphi[n]) &= \#\Lambda(\varphi \cdot \varphi + r) \\ &= \#((\lambda p.\lambda f.\lambda x.p \ (p \ f) \ (\overbrace{f \ (f \ \cdots \ (f \ x) \ \cdots \))}^r)) \mathcal{C}(\varphi)) \\ &= 2r + 2\varphi + 14 \\ &< 4\varphi + 14 \\ &\leq 4\sqrt{n} + 14 \end{aligned} \quad (10)$$

holds. By (10), the statement holds for $n > 15$ because $\#\mathcal{C}(n) = 2n + 3$. For $8 < n \leq 15$, the statement holds when $\varphi = 3$. Table 1 shows n , $\#\mathcal{C}(n)$, and $\#\Lambda(T_\varphi[n])$ in such cases. $\square$

Table 1. n , $\#C(n)$, and $\#\Lambda(T_\varphi[n])$ for $8 < n \leq 15$ and $\varphi = 3$.

n	$\#C(n)$	$\#\Lambda(T_\varphi[n])$	n	$\#C(n)$	$\#\Lambda(T_\varphi[n])$
9	21	20	13	29	26
10	23	22	14	31	28
11	25	24	15	33	28
12	27	24			

3.5. Further Compaction

Let $\Lambda(T_\varphi[n]) = M C(\varphi)$. Hereinafter, we call M and $C(\varphi)$ function term and argument term, respectively, similarly to the λ -terms stated in Corollary 2. Moreover, for a given natural number n , φ^* denotes a natural number such that $\varphi^* \geq 2$ and $\#\Lambda(T_{\varphi^*}[n])$ is smaller than any other $\#\Lambda(T_\varphi[n])$, where $\varphi \neq \varphi^*$.

Lemma 3 implies that there is room for more compaction of $\Lambda(T_{\varphi^*}[n])$ if $\varphi^* > 8$, since there is a natural number $\varphi_1^* \geq 2$ such that $\#\Lambda(T_{\varphi_1^*}[\varphi^*]) < \#C(\varphi^*)$. This holds recursively, that is, we can obtain a compact λ -term $\mathcal{L}(n)$ such that $\mathcal{L}(n) \rightarrow_\beta^* C(n)$ and $\#\mathcal{L}(n) \leq \#\Lambda(T_{\varphi^*}[n])$ as

$$\mathcal{L}(n) := \begin{cases} \Lambda(T_{\varphi^*}[n]) & (\text{if } \varphi^* \leq 8), \\ M \mathcal{L}(\varphi^*) & (\text{if } \varphi^* > 8), \end{cases}$$

where M is the function term of $\Lambda(T_{\varphi^*}[n])$. We show an algorithm generating $\mathcal{L}(n)$ for a given n in Algorithm 2. In line 9, we use Algorithm 1 to obtain $\Lambda(E_\varphi)$. Assume that $\mathcal{O}(\alpha)$ is the time complexity of Algorithm 1, then as shown in Algorithm 2, $\mathcal{O}(\alpha\sqrt{n})$ time is required to obtain φ^* once.

For the size of $\mathcal{L}(n)$, the following theorem holds.

Theorem 3. For a given natural number n if there are φ and t such that ${}^t\varphi = n$, then $\#\mathcal{L}(n) = \mathcal{O}(\text{slog}_\varphi n)$.

Proof. By Definition 11 and Lemma 3,

$$\begin{aligned} \#\mathcal{L}(n) &\leq \#\Lambda(T_\varphi[n]) \\ &= \#((\lambda p.\lambda f.\lambda x.\overbrace{p p \cdots p}^t f x) C(\varphi)) \\ &\leq \#((\lambda p.\lambda f.\lambda x.\overbrace{p p \cdots p}^t f x) C(8)) \\ &= 2t + 26. \end{aligned}$$

holds. Since $t = \text{slog}_\varphi n$, the statement holds. $\square$

As seen in the above Theorem 3, by using proposed method, we can obtain $\mathcal{L}(n)$ which is a tetrational compact expression of $C(n)$ for some n . This is properly smaller than $\mathcal{B}(n)$ which we introduced in Definition 7. However, Theorem 3 refers to only special cases of n . For general n , the following theorem holds.

Algorithm 2 Generation of $\mathcal{L}(n)$ for a given natural number n .

Input: n **Output:** $\mathcal{L}(n)$

```

1: function GENERATE( $n$ )
2:   if  $n \leq 8$  then
3:     return  $\mathcal{C}(n)$ 
4:   else
5:      $\varphi := 2, \varphi^* := \varphi, \text{minsize} := \infty, M := \text{NULL}$ 
6:     while  $\varphi \leq \lfloor \sqrt{n} \rfloor$  do
7:        $r := n \bmod \varphi$ 
8:        $E_\varphi + r := T_\varphi[n]$ 
9:        $((\lambda p. \lambda f. \lambda x. \hat{M}) \mathcal{C}(\varphi)) := \Lambda(E_\varphi)$ 
10:       $\Lambda(T_\varphi[n]) := ((\lambda p. \lambda f. \lambda x. \hat{M}[x := \overbrace{(f(f \cdots (f x) \cdots))^r}]) \mathcal{C}(\varphi))$ 
11:      if  $\#\Lambda(T_\varphi[n]) < \text{minsize}$  then
12:         $\varphi^* \leftarrow \varphi$ 
13:         $\text{minsize} \leftarrow \#\Lambda(T_\varphi[n])$ 
14:         $M \leftarrow$  the function term of  $\Lambda(T_\varphi[n])$ 
15:      end if
16:       $\varphi \leftarrow \varphi + 1$ 
17:    end while
18:     $M_{arg} := \text{GENERATE}(\varphi^*)$ 
19:    return  $(M M_{arg})$ 
20:  end if
21: end function
22:
23: return GENERATE( $n$ )

```

Theorem 4. For a given natural number n , $\#\mathcal{L}(n) = \mathcal{O}((\text{slog}_2 n)^{\frac{\log n}{\log \log n}})$.

We need the following lemma to prove Theorem 4.

Lemma 4. For a given natural number n , let M_{φ^*} and M_2 be function terms of $\Lambda(T_{\varphi^*}[n])$ and $\Lambda(T_2[n])$, respectively. Then, $\#M_{\varphi^*} \leq \#M_2$ holds.

Proof. By Definition 5, $\#\Lambda(T_{\varphi^*}[n]) = \#M_{\varphi^*} + \#\mathcal{C}(\varphi^*) + 1$ and $\#\Lambda(T_2[n]) = \#M_2 + \#\mathcal{C}(2) + 1$ holds. Therefore, by definition of φ^* , the following holds:

$$\begin{aligned}
 \#\Lambda(T_{\varphi^*}[n]) \leq \#\Lambda(T_2[n]) &\iff \#M_{\varphi^*} + \#\mathcal{C}(\varphi^*) + 1 \leq \#M_2 + \#\mathcal{C}(2) + 1 \\
 &\iff \#M_{\varphi^*} - \#M_2 \leq \#\mathcal{C}(2) - \#\mathcal{C}(\varphi^*) \\
 &\iff \#M_{\varphi^*} - \#M_2 \leq 0.
 \end{aligned}$$

The statement follows the above. Note that, by Definition 11, $T_\varphi[n]$ is only defined for $\varphi \geq 2$. □

Now we can prove Theorem 4.

Proof of Theorem 4. By the definition of $\mathcal{L}(n)$, we assume that $\mathcal{L}(n) = M_{\varphi^*} \mathcal{L}(\varphi^*)$, where M_{φ^*} is the function term of $\Lambda(T_{\varphi^*}[n])$. By Definition 5, the following holds:

$$\#\mathcal{L}(n) = \#M_{\varphi^*} + \#\mathcal{L}(\varphi^*) + 1. \quad (11)$$

First, we consider the size of M_{φ^*} . For a given natural number n , we inductively define a TAE $\hat{T}_2(n)$ as follows:

$$\hat{T}_2(n) := \begin{cases} n & (\text{if } n \leq 2), \\ k2 \cdot \hat{T}_2(p_k - r_k) + k2 + \dots + 12 \cdot \hat{T}_2(p_1 - r_1) + 12 + 2 & (\text{otherwise}). \end{cases}$$

where k is the maximum natural number such that $k2 \leq n$, and for $1 \leq i \leq k$, each p_i is the natural number such that $0 \leq p_i \cdot i2 < i+12$ and $r_i = p_i \bmod 2$. If $p_i = 0$ or 1 , we omit the corresponding terms or coefficients in the TAE, similarly to $T_2[n]$. While $\hat{T}_2(n)$ is not $E_2[n]$ (since there is a case that the evaluated result of $\hat{T}_2(n)$ is not equal to n), $\hat{T}_2(n)$ is clearly a TAE E_{φ} . Therefore, by Lemma 1, a CFLT $\Lambda(\hat{T}_2(n))$ exists. By Lemma 2, the upper bound of $\#\Lambda(\hat{T}_2(n))$ depends on the number of the arithmetic operations occurring in $\hat{T}_2(n)$. Let N_a and N_m be the numbers of addition and multiplication occurring in $T_2[n]$, respectively. Also, for tetration ${}^{n_2}n_1$, we call n_2 second argument of tetration, then let N_t be the sum of second arguments of all tetration occurring in $T_2[n]$. Let us denote $\hat{N}_a$, $\hat{N}_m$, and $\hat{N}_t$ similarly for $\hat{T}_2(n)$. Here, $N_a \leq \hat{N}_a$, $N_m \leq \hat{N}_m$, and $N_t \leq \hat{N}_t$ hold, since in $T_2[n]$, no terms increase compared to $\hat{T}_2(n)$ while the terms $i2$ following $i2 \cdot \hat{T}_2(n)$ is reduced if $r_i = 0$. Then,

$$\begin{aligned} \#\Lambda(T_2[n]) &\leq \max(\#\Lambda(\hat{T}_2(n))) \\ &\leq 14\hat{N}_a + 8\hat{N}_m + 2\hat{N}_t + 11. \end{aligned}$$

follows Lemma 2. Note that $(n \bmod 2) \leq 1$. Let $\hat{N}(n) = 14\hat{N}_a + 8\hat{N}_m + 2\hat{N}_t + 11$, then by the above,

$$\begin{aligned} \#\Lambda(T_2[n]) &\leq \sum_{i=1}^k (14 \cdot 2 + 8 \cdot 1 + 2 \cdot 2i + \hat{N}(p_i - r_i) + 11) \\ &\leq 2k^2 + 13k + k\hat{N}(p_k - r_k). \end{aligned} \quad (12)$$

holds. We assume that ρ is the depth of the recursion in $\hat{T}_2(n)$. Then, $k\hat{N}(p_k - r_k) = \mathcal{O}(k^\rho)$ holds. Therefore, by (12),

$$\#\Lambda(T_2[n]) = \mathcal{O}(k^\rho). \quad (13)$$

holds. Here, $(k2)^\rho \leq n \iff \rho \log k2 \leq \log n$ holds. Thus, by Corollary 1,

$$\rho \leq \frac{\log n}{\log k2} < \frac{\log n}{\log^{(\log_2 n)-1} 2} = \frac{\log n}{\log \log n}$$

holds. By Definition 2, $k \leq \log_2 n$. Then,

$$\#\Lambda(T_2[n]) = \mathcal{O}((\log_2 n)^{\frac{\log n}{\log \log n}}) \quad (14)$$

follows (13). By Lemma 4, $\#M_{\varphi^*}$ is bounded by $\Lambda(T_2[n])$ since the function term of $\Lambda(T_2[n])$ is smaller than $\Lambda(T_2[n])$ itself. Then, by (14), we obtain

$$\#M_{\varphi^*} = \mathcal{O}((\log_2 n)^{\frac{\log n}{\log \log n}}). \quad (15)$$

Second, we consider the size of $\mathcal{L}(\varphi^*)$. If $\varphi^* \leq 8$, it is constant since $\mathcal{L}(\varphi^*) = \mathcal{C}(\varphi^*)$ and $\#\mathcal{C}(\varphi^*) \leq \mathcal{C}(8) = 19$. Otherwise, $\#\mathcal{L}(\varphi^*) \leq \Lambda(T_{\varphi_1^*}[\varphi^*])$ holds with a natural number φ_1^* . By the definition of $\mathcal{L}(n)$, it recursively holds until $\varphi_i^* \leq 8$. Thus, by (11) and (15),

$$\#\mathcal{L}(\varphi^*) = \mathcal{O}((\log_2 \varphi^*)^{\frac{\log \varphi^*}{\log \log \varphi^*}}) + \mathcal{O}((\log_2 \varphi_1^*)^{\frac{\log \varphi_1^*}{\log \log \varphi_1^*}}) + \dots + \mathcal{O}((\log_2 \varphi_m^*)^{\frac{\log \varphi_m^*}{\log \log \varphi_m^*}}) + m + C \quad (16)$$

holds, where C is a constant and m is a natural number such that $\varphi_m^* \leq 8$. Here, by Lemma 2, $\varphi^* \leq \lfloor \sqrt{n} \rfloor$, $\varphi_1^* \leq \lfloor \sqrt{\varphi^*} \rfloor$, $\varphi_2^* \leq \lfloor \sqrt{\varphi_1^*} \rfloor, \dots, \varphi_m^* \leq \lfloor \sqrt{\varphi_{m-1}^*} \rfloor$ holds, then the recursion depth m is $\mathcal{O}(\log n)$. Hence,

$$\#\mathcal{L}(\varphi^*) = \mathcal{O}((\log_2 \varphi^*)^{\frac{\log \varphi^*}{\log \log \varphi^*}}) + \mathcal{O}(\log n) + C \quad (17)$$

follows (16). Note that $\sum_{i=1}^{\infty} F(x_i) < 2F(x_1)$ holds for function $F(x)$ and natural number x such that $F(x_i) \geq 0$ and $x_{i+1} \leq \sqrt{x_i}$ for $1 \leq i$.

As a result, by (12), (15), and (17), we obtain

$$\begin{aligned} \#\mathcal{L}(n) &= \mathcal{O}((\log_2 n)^{\frac{\log n}{\log \log n}}) + \mathcal{O}((\log_2 \varphi^*)^{\frac{\log \varphi^*}{\log \log \varphi^*}}) + \mathcal{O}(\log n) + C \\ &= \mathcal{O}((\log_2 n)^{\frac{\log n}{\log \log n}}) \end{aligned}$$

and the statement holds. $\square$

4. Application to Higher-Order Compression and Comparative Experiments

We implemented our method stated in Section 3, and conducted an experiment compared to the binary expression on λ -terms stated in Definition 7 and λ -terms generated by using the method stated in [2] (we call the method YKS).

Let $\mathcal{B}(n)$ denote the binary expression of n on λ -terms. As seen in Section 3, our proposed method generates a compact λ -term $\mathcal{L}(n)$ for a given $\mathcal{C}(n)$. While $\mathcal{C}(n)$ has the size of $\Theta(n)$, the size of the λ -term $\mathcal{L}(n)$ achieves $\mathcal{O}(\log_2 n)$ for some n , as shown in Theorem 3. However, as shown in Theorem 4, the size of $\mathcal{L}(n)$ becomes $\mathcal{O}((\log_2 n)^{\log n / \log \log n})$ for general n , which is greater than that of $\mathcal{B}(n)$, $\Theta(\log n)$. On the other hand, YKS is a method of higher-order compression, which generates an extended λ -term for a given text. Here, a *text* is an element of Σ^* , where Σ is an *alphabet*. Let $L_{\text{YKS}}(n)$ be the generated extended λ -term by using YKS for a given text $a^n c$ with $a, c \in \Sigma$. Then, we can find that the size of $L_{\text{YKS}}(n)$ achieves $\mathcal{O}(\log_2 n)$ for some n , similarly to $\mathcal{L}(n)$.

We define *extended λ -terms* following to [2]. We show that the application of $\mathcal{B}(n)$ and $\mathcal{L}(n)$ to extended λ -terms is directly possible.

Definition 12 (Extended λ -terms and their sizes). *Let x be a variable and $a \in \Sigma$. Then, extended λ -term $\tilde{M}$ is defined inductively as follows:*

$$\tilde{M} ::= a \mid x \mid \lambda x. \tilde{M} \mid \tilde{M}_1 \tilde{M}_2.$$

The size of $\tilde{M}$, denoted by $\#\tilde{M}$, is inductively defined as follows:

$$\#a = 1, \quad \#x = 1, \quad \#(\lambda x. \tilde{M}) = \#\tilde{M} + 1, \quad \#(\tilde{M}_1 \tilde{M}_2) = \#\tilde{M}_1 + \#\tilde{M}_2 + 1.$$

Similar to normal λ -terms, we define β -reduction, variable replacement notation $\tilde{M}_1[x := \tilde{M}_2]$, and simplification on extended λ -terms. Clearly, we can regard any normal λ -terms as extended λ -terms.

In higher-order compression, a given text $a^n c$ is regarded as an extended λ -term $\tilde{S}(n) = (\overbrace{a(a \cdots (a c) \cdots)})^n$, then we generate a compact extended λ -term $\tilde{M}$ such that $\tilde{M} \rightarrow_{\beta}^* \tilde{S}(n)$.

From $\mathcal{B}(n)$ and $\mathcal{L}(n)$, we can obtain extended λ -terms $\tilde{\mathcal{B}}(n)$ and $\tilde{\mathcal{L}}(n)$ such that $\tilde{\mathcal{B}}(n) \rightarrow_{\beta}^* \tilde{\mathcal{S}}(n)$ and $\tilde{\mathcal{L}}(n) \rightarrow_{\beta}^* \tilde{\mathcal{S}}(n)$, respectively, as follows. By Definition 7 and the definition of $\mathcal{L}(n)$, we assume that $\mathcal{B}(n) = (\lambda p. \lambda f. \lambda x. \hat{M}_{\mathcal{B}}) M'_{\mathcal{B}}$ and $\mathcal{L}(n) = (\lambda p. \lambda f. \lambda x. \hat{M}_{\mathcal{L}}) M'_{\mathcal{L}}$. Then, $\tilde{\mathcal{B}}(n)$ and $\tilde{\mathcal{L}}(n)$ are obtained as

$$\tilde{\mathcal{B}}(n) = (\lambda p. \hat{M}_{\mathcal{B}}[f := a, x := c]) M'_{\mathcal{B}} \quad \text{and} \quad \tilde{\mathcal{L}}(n) = (\lambda p. \hat{M}_{\mathcal{L}}[f := a, x := c]) M'_{\mathcal{L}},$$

respectively. Note that $\mathcal{B}(n) a c \rightarrow_{\beta}^* \tilde{\mathcal{S}}(n)$ and $\mathcal{L}(n) a c \rightarrow_{\beta}^* \tilde{\mathcal{S}}(n)$ holds, since $\mathcal{B}(n) \rightarrow_{\beta}^* \mathcal{C}(n)$, $\mathcal{L}(n) \rightarrow_{\beta}^* \mathcal{C}(n)$, and $\mathcal{C}(n) a c \rightarrow_{\beta}^* \tilde{\mathcal{S}}(n)$. Here, $\tilde{\mathcal{B}}(n)$ and $\tilde{\mathcal{L}}(n)$ are the results of simplification for $\mathcal{B}(n) a c$ and $\mathcal{L}(n) a c$, respectively.

Figures 1 and 2 show the experimental results. In both of them, “Binary”, “YKS”, and “Proposed” denote the methods using the binary expression on λ -terms, YKS, and ours, respectively. The horizontal axis shows the repetition number n of given text $a^n c$. Then, we compare the sizes of the generated extended λ -terms by using these three methods, for the given $a^n c$.

Figure 1. Term size for integer n .

Figure 2. Average ratio to binary expression.

In Figure 1, the vertical axis shows the sizes of $\tilde{\mathcal{B}}(n)$, $L_{\text{YKS}}(n)$, and $\tilde{\mathcal{L}}(n)$. The inequality $\#\tilde{\mathcal{L}}(n) \leq \#\tilde{\mathcal{B}}(n)$ holds in 5187 out of 10,000 cases within the range $[1, 10000]$. The average ratio $\#\tilde{\mathcal{L}}(n)/\#\tilde{\mathcal{B}}(n)$ for the range is

approximately 0.9962. Similarly, the inequality $\#\tilde{\mathcal{L}}(n) \leq \#L_{\text{YKS}}(n)$ holds in 5959 cases, and the average ratio $\#\tilde{\mathcal{L}}(n)/\#L_{\text{YKS}}(n)$ is approximately 0.9321.

In Figure 2, the vertical axis shows the ratio (the average size of cumulative sum of extended λ -terms from 1 to n)/ $\#\tilde{\mathcal{B}}(n)$. That is, Figure 2 shows how $\#L_{\text{YKS}}(n)$ and $\#\tilde{\mathcal{L}}(n)$ increase compared to $\#\tilde{\mathcal{B}}(n)$ on average. As can be seen, $\#\tilde{\mathcal{L}}(n)$ tends to be greater than $\#\tilde{\mathcal{B}}(n)$ when n is greater than 10,000. We consider that it is consistent with the theoretical upper bound $\mathcal{O}((\log_2 n)^{\log n / \log \log n})$, stated in Theorem 4.

5. Conclusions

In this paper, we addressed the problem of compaction of Church numerals. For given natural number n , by using proposed RTP, we decompose n into an arithmetic expression, which enables to obtain a compact λ -term leading to the Church numeral of n . We proved that the size of the obtained λ -term becomes $\mathcal{O}((\log_2 n)^{\log n / \log \log n})$. Moreover, we experimentally confirmed that the λ -terms produced by our method tend to be smaller than binary expressions on λ -terms on average, when the given number is less than approximately 10,000. We have not proved the lower bound of the size in the worst case; it is our future work.

The compaction of Church numerals can be applied to higher-order compression, that uses extended λ -terms as the data model. In the procedure of higher-order compression, a repetitive part in the input can be represented as a λ -term with $\mathcal{C}(n)$. Thus, efficient compaction of $\mathcal{C}(n)$ will help for improving the compression performance of higher-order compression.

In data compression, data models are finally encoded in bit sequences. For bit encoding of λ -terms, Tromp [16] proposed a method for untyped λ -terms. Very recently, Takeda et al. [17] proposed an efficient encoding scheme for simply-typed λ -terms. Finding an efficient bit encoding for our method is also our future work.

Author Contributions: Conceptualization, I.F. and T.K.; Software, I.F.; Writing—original draft, I.F. and T.K.; Writing—review & editing, I.F. and T.K.

Funding: This work was supported by JSPS KAKENHI Grant Number JP15K00002, JP18K11149, JP18K19771 and JST CREST Grant Number JPMJCR1402, Japan.

Acknowledgments: The authors would like to thank Ayumi Shinohara and his colleagues for providing the source code for higher-order compression.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kobayashi, N.; Matsuda, K.; Shinohara, A.; Yaguchi, K. Functional programs as compressed data. *High. Order Symb. Comput.* **2012**, *25*, 39–84. [\[CrossRef\]](#)
2. Yaguchi, K.; Kobayashi, N.; Shinohara, A. Efficient algorithm and coding for higher-order compression. In Proceedings of the 2014 Data Compression Conference (DCC2014), Snowbird, UT, USA, 26–28 March 2014; p. 434.
3. Kieffer, J.C.; Yang, E.H. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory* **2000**, *46*, 737–754. [\[CrossRef\]](#)
4. Cover, T.M.; Thomas, J.A. *Elements of Information Theory* (Wiley Series in Telecommunications and Signal Processing); Wiley-Interscience: New York, NY, USA, 2006.
5. Charikar, M.; Lehman, E.; Liu, D.; Panigrahy, R.; Prabhakaran, M.; Sahai, A.; Shelat, A. The smallest grammar problem. *IEEE Trans. Inf. Theory* **2005**, *51*, 2554–2576. [\[CrossRef\]](#)
6. Larsson, N.J.; Moffat, A. Off-line dictionary-based compression. *Proc. IEEE* **2000**, *88*, 1722–1732. [\[CrossRef\]](#)
7. Maruyama, S.; Tanaka, Y.; Sakamoto, H.; Takeda, M. Context-sensitive grammar transform: Compression and pattern matching. In Proceedings of the String Processing and Information Retrieval (SPIRE2008), Melbourne, Australia, 10–12 November 2008; pp. 27–38.
8. Masaki, T.; Kida, T. Online grammar transformation based on re-pair algorithm. In Proceedings of the Data Compression Conference (DCC2016), Snowbird, UT, USA, 30 March–1 April 2016; pp. 349–358.

9. Nevill-Manning, C.G.; Witten, I.H.; Maulsky, D.L. Compression by induction of hierarchical grammars. In Proceedings of the Data Compression Conference (DCC'94), Snowbird, UT, USA, 29–31 March 1994; pp. 244–253.
10. Takabatake, Y.; Sakamoto, H. A space-optimal grammar compression. In Proceedings of the 25th Annual European Symposium on Algorithms (ESA 2017), Vienna, Austria, 4–6 September 2017; Leibniz International Proceedings in Informatics (LIPIcs); Volume 87, pp. 67:1–67:15. [\[CrossRef\]](#)
11. Ohno, T.; Goto, K.; Takabatake, Y.; I, T.; Sakamoto, H. LZ-ABT: A practical algorithm for α -balanced grammar compression. In Proceedings of the 29th International Workshop on Combinatorial Algorithms (IWOCA 2018), Singapore, 16–19 July 2018; Springer: Singapore, 2018; pp. 323–335.
12. Rytter, W. Grammar compression, LZ-encodings, and string algorithms with implicit input. In *Automata, Languages, and Programming (ICALP2004)*; Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 15–27.
13. Lohrey, M. Algorithmics on SLP-compressed strings: A survey. *Groups Complex. Cryptol.* **2012**, *4*, 241–299. [\[CrossRef\]](#)
14. Furuya, I.; Kida, T. Compaction of church numerals for higher-order compression. In Proceedings of the Data Compression Conference (DCC2018), Snowbird, Utah, USA, 27–30 March 2018; p. 408.
15. Mogensen, T.A. An investigation of compact and efficient number representations in the pure lambda calculus. In *Revised Papers from the 4th International Andrei Ershov Memorial Conference on Perspectives of System Informatics (PSI '02): Akademgorodok, Novosibirsk, Russia*; Springer-Verlag: London, UK, 2001; pp. 205–213.
16. Tromp, J. Binary lambda calculus and combinatory logic. In *Kolmogorov Complexity and Applications*; Hutter, M., Merkle, W., Vitanyi, P.M., Eds.; Number 06051 in Dagstuhl Seminar Proceedings; 2006; Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.
17. Takeda, K.; Kobayashi, N.; Yaguchi, K.; Shinohara, A. Compact bit encoding schemes for simply-typed lambda-terms. In Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, Nara, Japan, 18–24 September 2016; Volume 51, pp. 146–157.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).