REPORTS ON MATHEMATICAL LOGIC
53 (2018), 19–42
doi: 10.4467/20842589RM.18.002.8835

Łukasz LACHOWSKI

ON THE COMPLEXITY OF THE STANDARD TRANSLATION OF LAMBDA CALCULUS INTO COMBINATORY LOGIC

A b s t r a c t. We investigate the complexity of the standard translation of lambda calculus into combinatory logic. The main result shows that the asymptotic growth rate of the size of a translated term is $\Theta(n^3)$ in worst-case, where n denotes the size of the lambda term.

1. Introduction

Both lambda calculus ([Chu36]) and combinatory logic ([Cur30]) are Turing-complete models of computation. As such, lambda calculus serves as a the-

Received 8 June 2017
This work was partially supported by the grant 2013/11/B/ST6/00975 funded by the Polish National Science Center.
Keywords and phrases: combinatory logic, lambda calculus, complexity analysis, functional programming.
AMS subject classification: 03B40, 68Q25, 68N18.

oretical basis for many functional programming languages, e.g. Haskell and SML. Evaluating expressions in lambda calculus is not easy since semantics of the substitution must be defined carefully to avoid problems with variables binding. One of the motivations for introducing combinatory logic was to eliminate the notion of bound variables from logical formulas and to simplify the definition of substitution. Thereby, compared to lambda calculus, combinatory logic is simpler in terms of computational interpretations.

In the definition of combinatory logic we use only two combinators, namely S and K. Together with the notion of reduction, it is sufficient to define any recursive function in this model. Details regarding the relation of lambda calculus and combinatory logic can be found in [Bar12, page 151]. The question that immediately arises is how to reformulate computations defined in lambda calculus by means of combinatory logic and what the complexity of this process is. The standard translation is one of the basic algorithms providing extensional equivalence of these models. We believe it was first defined in [Sch24]. Unfortunately, we were unable to find a detailed analysis of its worst-case complexity. In the literature other translations have been studied (see [Tur79; BD97; KS87; Nos85; Joy85]). In this work we focus on providing the asymptotic growth rate of the size of a lambda term translated using the standard translation.

We start with some basic definitions concerning both lambda calculus and combinatory logic and define the standard translation. Next, we present an upper bound for the size of combinators that are translations of lambda terms of a fixed size. We do not search for a particular set of worst-case instances of this algorithm, but rather focus on finding some specific set that contains all translations of λ-terms of a given size and for which we are able to calculate the size limit of its elements. In the next section we describe how the translation behaves on a specific class of terms, obtaining as well a lower bound of the worst-case behavior. Our method can be easily adapted to other translations, e.g. the one defined by David Turner in [Tur79].

Definition 1.1 (lambda terms)**.** Assume we have a countable set of variables $\mathcal{V} \coloneqq \{x, y, \dots\}$. The set Λ of λ-terms is defined by the following grammar:

$$\Lambda \coloneqq \mathcal{V} \mid (\lambda\mathcal{V}.\Lambda) \mid (\Lambda\Lambda)$$

Definition 1.2 (combinatory logic terms)**.** Assume we have a countable set of variables $\mathcal{V} \coloneqq \{x, y, \dots\}$. The set $\mathcal{CL}$ of combinatory logic terms, called $\mathcal{CL}$-terms or combinators, is defined by the following grammar:

$$\mathcal{C} \coloneqq \mathcal{V} \mid K \mid S \mid (\mathcal{C}\,\mathcal{C})$$

Notation. (i) The symbol $\equiv$ denotes the syntactic equality of λ-terms or combinators.

(ii) Outermost parentheses are usually omitted.

(iii) Application associates to the left, e.g. $ABCD$ stands for $((AB)C)D$.

(iv) Lower-case letters $x, y, z, \dots$ denote variables.

(v) Abstraction associates to the right, e.g. $\lambda xyz.M$ stands for $\lambda x.(\lambda y.(\lambda z.M))$.

(vi) We denote arbitrary combinators by letters P, Q, R, ... and λ-terms by M, N, O, ...

(vii) Constant I denotes both SKK and $\lambda x.x$, depending on the context in which it is used.

(viii) In the context of lambda calculus, the constant K denotes $\lambda xy.x$.

Before we describe the complexity of the translation we need the notion of size for both λ- and $\mathcal{CL}$-terms.

Definition 1.3. We define notions of size for both λ- and $\mathcal{CL}$-terms.

(i) Size of a lambda term M, denoted by $|M|_\lambda$, is inductively defined as follows:

$$\begin{aligned} |x|_\lambda &= 1 \quad \text{for every variable } x,\\ |MN|_\lambda &= 1 + |M|_\lambda + |N|_\lambda,\\ |\lambda x.M|_\lambda &= 1 + |M|_\lambda. \end{aligned}$$

(ii) Size of a combinatory logic term P, denoted by $|P|_{\mathcal{CL}}$, is inductively defined as follows:

$$\begin{aligned} &|S|_{\mathcal{CL}} = |K|_{\mathcal{CL}} = |x|_{\mathcal{CL}} = 1 \quad \text{for every variable } x,\\ &|PQ|_{\mathcal{CL}} = 1 + |P|_{\mathcal{CL}} + |Q|_{\mathcal{CL}}. \end{aligned}$$

We can interpret $\mathcal{CL}$-terms as labeled binary trees, where inner nodes represent applications and labeled leaves stand for variables or constants (see Figure 1). In this interpretation, the size of a combinator corresponds to the total number of its leaves and internal nodes. Before we introduce the standard translation, we also need to define notions of free variables for both of these models.

Definition 1.4. (i) The set of free variables of a lambda term M, denoted $\mathsf{FV}_\lambda(M)$, is defined inductively as follows:

$$\begin{aligned} \mathsf{FV}_\lambda(x) &= \{x\} \quad \text{for every variable } x, \\ \mathsf{FV}_\lambda(MN) &= \mathsf{FV}_\lambda(M) \cup \mathsf{FV}_\lambda(N), \\ \mathsf{FV}_\lambda(\lambda x.M) &= \mathsf{FV}_\lambda(M) \setminus \{x\}. \end{aligned}$$

(ii) We call a lambda abstraction $\lambda x.M$ binding if $x \in \mathsf{FV}_\lambda(M)$, otherwise it is called non-binding.

(iii) In the case of combinatory logic, the set of free variables of a combinator P, denoted $\mathsf{FV}_{\mathcal{CL}}(P)$, is defined as collection of all of its variables.

(iv) We call a $\mathcal{CL}$-term closed if it contains no free variables. By $\mathcal{CL}^0$ we denote the set of all closed $\mathcal{CL}$-terms.

Both λ- and $\mathcal{CL}$-terms can be equipped with notion of reduction. On the set of λ-terms we define relation called β-reduction and for $\mathcal{CL}$-terms a corresponding notion of w-reduction. For details see [Bar12, pages 50 and 154].

Definition 1.5. On the set of lambda terms we define the reduction relation $\to_\beta$ as the smallest relation such that:

- For all λ-terms M and N we have $(\lambda x.M)N \to_\beta M[x := N]$, where $M[x := N]$ denotes substitution of every occurrence of the variable x in M by the term N. During this process we must avoid capturing free variables of the term N by some lambda abstractions of M. We can avoid this by changing names of bound variables of the term M prior to application of this rule.

- For all λ-terms M, N and O, and for every variable x, if $M \to_\beta N$, then we also have $MO \to_\beta NO$, $OM \to_\beta ON$ and $\lambda x.M \to_\beta \lambda x.N$.

The relation $=_\beta$ is the smallest equivalence relation containing $\to_\beta$.

Definition 1.6. We define the reduction relation $\to_w$ on the set of $\mathcal{CL}$-terms as the smallest relation such that:

- For all combinators P, Q and R we have $KPQ \to_w P$ and $SPQR \to_w PR(QR)$.
- For all combinators P, Q and R, if $P \to_w Q$, then we also have $PR \to_w QR$ and $RP \to_w RQ$.

Similarly to the previous definition, by $=_w$ we denote the equivalence closure of the relation $\to_w$.

After examining semantics of both these definitions, it turns out that lambda abstraction can be simulated by S and K in combinatory logic. We call this process bracket abstraction.

Definition 1.7. For every variable x we define a function $\lambda^* x : \mathcal{CL} \to \mathcal{CL}$, called bracket abstraction, as follows:

$$\begin{aligned}
\lambda^* x.x &= SKK, \\
\lambda^* x.P &= KP \quad \text{if } x \notin \mathsf{FV}_{\mathcal{CL}}(P), \\
\lambda^* x.PQ &= S\,(\lambda^* x.P)\,(\lambda^* x.Q) \quad \text{if } x \in \mathsf{FV}_{\mathcal{CL}}(PQ).
\end{aligned}$$

Fact 1.8. (i) $\mathsf{FV}_{\mathcal{CL}}(\lambda^* x.P) = \mathsf{FV}_{\mathcal{CL}}(P) \setminus \{x\}$

(ii) $(\lambda^* x.P)x =_w P$

(iii) $(\lambda^* x.P)Q =_w P[x := Q]$

Once we have a mechanism capable of simulating variable abstraction, we can use it to obtain a mapping between terms of Λ and $\mathcal{CL}$. In the following sections, we focus on showing what the complexity of this algorithm is.

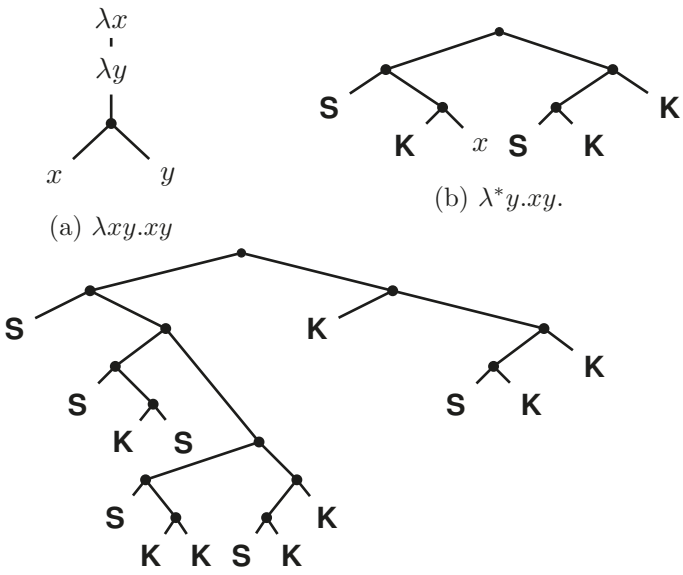
Definition 1.9. The standard translation $[\quad] : \Lambda \to \mathcal{CL}$ is defined inductively as follows:

$$\begin{aligned}
[\, x \,] &= x \quad \text{for every variable } x, \\
[\, MN \,] &= [\, M \,]\,[\, N \,], \\
[\, \lambda x.M \,] &= \lambda^* x.[\, M \,].
\end{aligned}$$

Example 1. Let us translate term $M \equiv \lambda xy.xy$.

$$\begin{aligned}[\lambda xy.xy] =& \lambda^* x.\lambda^* y.xy \\ =& \lambda^* x.S(Kx)(SKK) \\ =& S(\lambda^* x.S(Kx))(K(SKK)) \\ =& S(S(KS)(S(KK)(SKK)))(K(SKK))\end{aligned}$$

We see that the size of the term M is 5 and the size of its translation is equal to 27.



(a) $\lambda xy.xy$

(b) $\lambda^* y.xy$.

(c) $[\lambda xy.xy] = \lambda^* x.\lambda^* y.xy$

Figure 1: Visualization of some consecutive steps of the translation algorithm for the term $\lambda xy.xy$.

Remark 1.10. Two terms which are equal up to renaming of bound variables are translated into the same combinator. Moreover, if two terms differ only by their free variables, then their $\mathcal{CL}$ counterparts have the same size.

We also define a method of translating $\mathcal{CL}$-terms into lambda calculus. Using both these translations, we can prove that every term of lambda

calculus can be defined (up to $=_\beta$ relation) by means of $K \equiv \lambda xy.x$ and $S \equiv \lambda xyz.xz(yz)$.

Definition 1.11. The translation of $\mathcal{CL}$-terms into lambda calculus is defined as follows:

$$
\begin{aligned}
(\ x\ )_\Lambda &= x \quad \text{for every variable } x,\\
(\ K\ )_\Lambda &= \lambda xy.x,\\
(\ S\ )_\Lambda &= \lambda xyz.xz(yz),\\
(\ PQ\ )_\Lambda &= (\ P\ )_\Lambda\,(\ Q\ )_\Lambda.
\end{aligned}
$$

Fact 1.12. (i) *For every λ-term M we have* $(\ [\ M\ ]\ )_\Lambda =_\beta M$.

(ii) *For all $\mathcal{CL}$-terms P and Q, if* $P =_w Q$, *then* $(\ P\ )_\Lambda =_\beta (\ Q\ )_\Lambda$.

Unfortunately, the reverse composition does not preserve equality of $\mathcal{CL}$-terms, which we show in the following example.

Example 2. Composition of $(\ )_\Lambda$ and $[\ ]$ is not an identity of $\mathcal{CL}$-terms (up to $=_w$ relation), e.g. $[\ (\ K\ )_\Lambda\ ] = S(KK)I \neq_w K$. Moreover, equivalence $=_\beta$ of some λ-terms does not imply equivalence $=_w$ of their translations, e.g. $\lambda x.KIx =_\beta \lambda x.I$, but $[\ \lambda x.KIx\ ] = S(K(S(KK)II))I =_w S(K(KI))I \neq_w KI = [\ \lambda x.I\ ]$.

One way to fix this inconvenience is to extend $\rightarrow_\beta$ and $\rightarrow_w$ by the rule of extensionality.

Definition 1.13 (Rule of Extensionality)**.** By symbol $\rightarrow_\square^{\mathsf{ext}}$ we denote extended relation $\rightarrow_\square$ satisfying the following rule (symbol $\square$ is to be replaced with w or β, respectively): if x is not a free variable of M or N and we have $Mx \rightarrow_\square^{\mathsf{ext}} Nx$, then we can infer $M \rightarrow_\square^{\mathsf{ext}} N$. By $=_\square^{\mathsf{ext}}$ we denote the equivalence closure of the relation $\rightarrow_\square^{\mathsf{ext}}$.

With this additional rule we are able to prove that these models are equivalent in the following sense.

Fact 1.14. (i) *The composition of translations* $(\ )_\Lambda$ *and* $[\ ]$ *preserves equivalence* $=_w^{\mathsf{ext}}$, *i.e.* $[\ (\ P\ )_\Lambda\ ] =_w^{\mathsf{ext}} P$.

(ii) *For all λ-terms M and N, if* $M =_\beta^{\mathsf{ext}} N$, *then we can infer that* $[\ M\ ] =_w^{\mathsf{ext}} [\ N\ ]$.

2. Complexity of the standard translation

Our main result states that using the standard translation the worst-case asymptotic growth rate of the size of a translated λ-term is $\Theta(n^3)$, where n denotes the size of the λ-term. We introduce a function that computes the upper bound for the translation size as follows:

$$\mathit{SizeT}(n) \coloneqq \sup\{|\ [\ M\ ]\ |_{\mathcal{CL}} : M \in \Lambda,\ |M|_\lambda = n\}.$$

Although the set of λ-terms of a given size is of infinite cardinality, by Remark 1.10 it is sufficient to consider only some specific finite set of terms. Hence, the supremum is always finite and $\mathit{SizeT}(n)$ is a well defined function. By means of a Haskell program ([Lac17]) we can enumerate first few values of this function:

$$1, 5, 11, 17, 27, 37, 47, 65, 79, 101, 123, 141, 179, 205, 235, 285, \ldots$$

Since $\mathit{SizeT}(5) = 27$, we see that the term $\lambda xy.xy$, used in Example 1, is a witness of the worst-case behavior.

First, for any given size n, we focus on finding an upper bound for the value of the function $\mathit{SizeT}(n)$. To this end, we define some specific family of sets of $\mathcal{CL}$-terms and show that for any λ-term, its corresponding combinator belongs to one of those sets. Using their inductive definition we are able to calculate an upper bound for all translated terms. To find a lower bound, we study some specific class of λ-terms for which it is easy to analyze how the algorithm changes their size. As it turns out, the growth rate estimated in the first part is asymptotically tight (up to a constant).

2.1 Upper bound

We need some additional tools to calculate an upper bound of the function $\mathit{SizeT}(n)$ for every value of n. For a collection of all λ-terms of a given size n, we define a superset of the set of all translated forms of these terms for which we are able to calculate the size of the largest term among its elements. More formally, we look for a set, denoted by T^n, fulfilling the following conditions:

$$\begin{aligned} &\{[\ M\ ] : M \in \Lambda,\ |M|_\lambda = n\} \subseteq \mathsf{T}^n, \\ &\sup\{|P|_{\mathcal{CL}} : P \in \mathsf{T}^n\} \leq P(n) \quad \text{where } P(n) \text{ is some polynomial of } n. \end{aligned} \tag{3}$$

Then, instead of searching for an upper bound of the function $SizeT(n)$, we estimate the supremum for the size of all terms of the set T^n, i.e. $SizeT(n) \leq \sup\{|M|_{\mathcal{CL}} : M \in \mathsf{T}^n\}$. The following definition is crucial for proving our result.

Definition 2.1. For all positive integers n, j and k, such that $j \leq n$, we inductively define sets of combinatory logic terms denoted by symbol $\mathsf{C}_{n,\,j,\,k}$ as follows:

(i) for every $n \geq 1$ the constant K and every variable are elements of the set $\mathsf{C}_{n,\,1,\,1}$,

(ii) if a combinator Q is an element of the set $\mathsf{C}_{n,\,j,\,k}$ such that $j < n$ and P is a closed combinator such that $|P|_{\mathcal{CL}} < 2n$, then the combinator PQ is in $\mathsf{C}_{n,\,j+1,\,k}$,

(iii) if combinators P and Q are elements of sets $\mathsf{C}_{n,\,j_1,\,k_1}$ and $\mathsf{C}_{n,\,j_2,\,k_2}$ respectively, then the combinator PQ belongs to the set $\mathsf{C}_{n,\,1,\,k_1+k_2+1}$.

By carefully examining both this definition and of the translation we can notice how they are related. As it will be revealed in the following propositions, the second case of this definition corresponds to terms of the form $S(\lambda^*x.P)$ and KP created by the bracket abstraction and their further development during the translation process. Similarly, the third case of this definition coincides with terms of the form $(S(\lambda^*x.P))(\lambda^*x.Q)$ and $[\,P\,][\,Q\,]$ created by the translation.

It is worth noting that some of the combinators can be derived in more than one way, e.g. term $(SK)K$ belongs both to the set $\mathsf{C}_{2,\,2,\,1}$ as well as to $\mathsf{C}_{2,\,1,\,3}$. We can also find combinators which do not belong to any of these sets, e.g. KS.

Terms of a set $\mathsf{C}_{n,\,j,\,k}$ can be presented as colored trees (see Figures 3 and 5). In this interpretation, each of the parameters n, j and k describes some quantitative property of that colored representation. That is to say, the parameter k determines how many distinct colors we have used and the parameter j allows us to bound the number of nodes with the same color as the root of a tree. Value of the parameter n determines the maximal number of nodes sharing each color. Each case of Definition 2.1

$$\forall n > 0 \quad K \in \mathsf{C}_{n,1,1} \qquad \frac{x \in \mathcal{V}}{\forall n > 0 \quad x \in \mathsf{C}_{n,1,1}}$$

$$\frac{P \in \mathcal{CL}^0 \qquad Q \in \mathsf{C}_{n,j,k} \qquad j < n \qquad |P|_{\mathcal{CL}} < 2n}{PQ \in \mathsf{C}_{n,j+1,k}}$$

$$\frac{P \in \mathsf{C}_{n,j_1,k_1} \qquad Q \in \mathsf{C}_{n,j_2,k_2}}{PQ \in \mathsf{C}_{n,1,k_1+k_2+1}}$$

Figure 2: Definition 2.1 presented by a diagram.

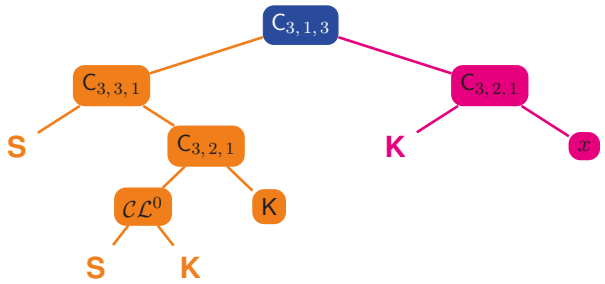


Figure 3: Colored tree representation of a combinator from the set $\mathsf{C}_{3,1,3}$.

describes a method of constructing colored trees representing elements of the set $\mathsf{C}_{n,j,k}$. According to its first case, we can pick up any variable or the constant K as the basis of our tree and paint it by any color. The second case describes how we can extend a colored combinator by attaching a closed term of the same color as its root. Having a tree with its root painted e.g. red, and with the value of the parameter j still smaller than n, we can pick up any closed term Q of size limited by $|Q|_{\mathcal{CL}} < 2n$, paint all of its nodes using red color, and attach it to the original tree by a new red binary node. Afterwards, we increase the value of the parameter j, marking that the number of red nodes has been increased. From this, we see that when the value of j is 1, then there is no other node with the same color as the root and that n limits the number of nodes of each color. The last case of this definition specifies a method for merging two colored representations. Namely, having two colored trees P and Q, which have no color in common, we can combine them using a binary node with a brand new color. This forces us to increase the value of k by 1.

After this procedure, the new root is the only one using that new color, therefore the value of j equals 1. Two such trees can be merged only if they belong to sets parametrized by the same value of n. Thanks to that restriction, the meaning of the parameter n remains valid throughout the construction. Using this representation, the following propositions can be interpreted as a way of finding some bounds for the number of colors and the number of nodes of each color which are, needed to paint a translated λ-term. The main result shows that for a λ-term of size n we can paint its translation using no more than n colors and having no more than $\mathcal{O}(n^2)$ nodes of each of these colors.

For any fixed values of parameters j and k we can easily infer that the sequence $(\mathsf{C}_{n,j,k})_n$ is an ascending sequence of sets.

Observation 2.2. *For all positive integers n, j, k and l we have $\mathsf{C}_{n,j,k} \subset \mathsf{C}_{n+l,j,k}$.*

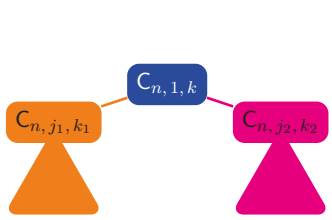
This observation is useful whenever we want to extend a particular color in a combinator, but the value of its parameter n limits this possibility. More precisely, if for an element of a set $\mathsf{C}_{n,j,k}$ we have already attached n closed terms of the same color as its root node, then whenever we want to build a bigger term from it without using a new color, we can simply choose a greater value for the parameter n and then enlarge the term according to Definition 2.1. We can also use it when we need to merge two terms with different values of n. This is possible since at the beginning of the construction of some colored combinator, the first case of the definition allows us to pick n arbitrarily large. The rest of the derivation of that term stays unchanged. Given a combinator P, being a translation of some λ-term, we are interested in minimizing the value of n for which $P \in \mathsf{C}_{n,j,k}$, since this value limits the number of nodes of each color.

The standard translation describes a bottom-up approach for which the most computationally demanding operation is a variable abstraction. The next two propositions show the approximate cost of simulating variable abstraction on an element of a set $\mathsf{C}_{n,j,k}$.

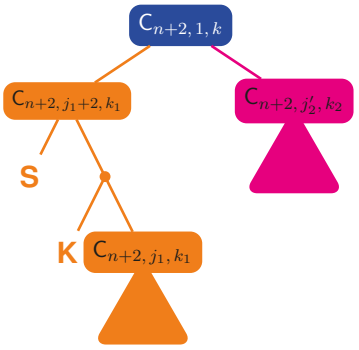
Proposition 2.3. *Given a combinator $P \in \mathsf{C}_{n,j,k}$ and a variable x not occurring in P, the combinator $\lambda^* x.P$ is an element of the set $\mathsf{C}_{n+1,j+1,k}$.*

Proof. If we use the bracket abstraction for the variable x which is not free in P, we simply apply term P to the constant K. In order to do this,

we may need a greater value for the parameter n. By Observation 2.2, we know that $P \in \mathsf{C}_{n,j,k} \subset \mathsf{C}_{n+1,j,k}$. Since $|K|_{\mathcal{CL}} = 1 < 2(n+1)$ for every $n \geq 1$ and value of the parameter j is strictly smaller than $n+1$, we have $\lambda^* x.P = KP \in \mathsf{C}_{n+1,j+1,k}$. □



(a) $P \equiv QR$, where $Q \in \mathsf{C}_{n,j_1,k_1}$ and $R \in \mathsf{C}_{n,j_2,k_2}$. Abstracted variable x occurs only in term R.



(b) $\lambda^* x.MN$

Figure 4: A term before and after applying the bracket abstraction.

Proposition 2.4. *For every combinator $P \in \mathsf{C}_{n,j,k}$ and for every variable x that is free in P, the combinator $\lambda^* x.P$ is an element of the set $\mathsf{C}_{n+2,j',k}$ with $j' = j$ or $j' = j+1$.*

Proof. The proof goes by induction on the derivation of a term P from a set $\mathsf{C}_{n,j,k}$.

(i) If P is in $\mathsf{C}_{n,1,1}$ and a variable x occurs in P, then $P \equiv x$. Using the definition of bracket abstraction and Observation 2.2 we obtain $\lambda^* x.x = (SK)K \in \mathsf{C}_{2,2,1} \subset \mathsf{C}_{n+2,2,1}$ for every n.

(ii) Let $P \in \mathsf{C}_{n,j,k}$ be of the form QR where $R \in \mathsf{C}_{n,j-1,k}$ for which $j-1 < n$ and Q is a closed term such that $|Q|_{\mathcal{CL}} < 2n$. Since x occurs in P and Q is a closed combinator, we derive that x occurs in R. Using the definition of the bracket abstraction, we get $\lambda^* x.QR = S(KQ)\,(\lambda^* x.R)$. We use the induction hypothesis for the term $R \in \mathsf{C}_{n,j-1,k}$ and obtain that $\lambda^* x.R \in \mathsf{C}_{n+2,j',k}$ for some j' such that $j' = j-1$ or $j' = j$. Term Q is closed, therefore $S(KQ)$ is also closed and we have

$$|S(KQ)|_{\mathcal{CL}} = |Q|_{\mathcal{CL}} + 4 < 2(n+2).$$

From this, we see that the combinator $\lambda^*x.P$ is in $\mathsf{C}_{n+2,\,j'+1,\,k}$.

(iii) Let $P \in \mathsf{C}_{n,\,1,\,k}$ be of the form $Q\,R$, where $Q \in \mathsf{C}_{n,\,j_1,\,k_1}$ and $R \in \mathsf{C}_{n,\,j_2,\,k_2}$ are such that $k_1 + k_2 + 1 = k$ (see Figure 4). From the definition of bracket abstraction we have $\lambda^*x.P = S\,(\lambda^*x.Q)\,(\lambda^*x.R)$. If the abstracted variable occurs in Q, then we can use the induction hypothesis to obtain $\lambda^*x.Q \in \mathsf{C}_{n+2,\,j_1',\,k_1}$ such that $j_1' = j_1$ or $j_1' = j_1 + 1$. Otherwise, we use Proposition 2.3 to get $\lambda^*x.Q \in \mathsf{C}_{n+1,\,j_1',\,k_1} \subset \mathsf{C}_{n+2,\,j_1',\,k_1}$ where $j_1' = j_1 + 1$. We notice that in both cases $j_1' < n + 2$ and hence we can still attach the constant S to $\lambda^*x.Q$. By Definition 2.1 we have $S(\lambda^*x.Q) \in \mathsf{C}_{n+2,\,j_1'+1,\,k_1}$. Using similar argument for R we see that $\lambda^*x.R$ is in $\mathsf{C}_{n+2,\,j_2',\,k_2}$ for some $j_2' = j_2$ or $j_2' = j_2 + 1$. By Definition 2.1 we obtain that $\lambda^*x.P = S(\lambda^*x.Q)(\lambda^*x.R)$ is an element of $\mathsf{C}_{n+2,\,1,\,k}$. □

The last two propositions have a simple interpretation using colored trees represantation. The former claims that if we use the bracket abstraction for a variable that does not occur in a given term, then we need to attach a new closed term, namely the constant K, of the same color as the root. The value of the parameter n limits the number of such closed terms, so we might need to increase it too. After this operation the size of a term changes marginally. The latter one, which removes all occurrences of a given variable, has a greater impact on the size. In this case the interpretation is such that for every color we attach maximally two new closed terms of that color and enlarge remaining closed terms of that color by at most 4 new nodes. Using these propositions we can also conclude a method of coloring combinators. Prior to translation we can paint every node of a term by a unique color. Then we assume that during the translation process only the bracket abstraction can create new colored nodes. Whenever it creates a new constant, it attaches it to an already painted term. This new constant, together with the newly created binary node, inherits the color of that term. Representing the color of a term by an upper index attached to a combinator (e.g. P^a), we can redefine bracket abstraction as follows: $\lambda^*x.(P^aQ^b)^c = ((S^a\lambda^*x.P^a)^a(\lambda^*x.Q^b)^b)^c$, $\lambda^*x.M^a = (K^aM^a)^a$, $\lambda^*x.x^a = ((S^aK^a)^aK^a)^a$. This method of coloring corresponds to the way in which we accounted newly created combinators in last two propositions.

In the next proposition we show that every combinator which is a translation of some λ-term belongs to one of the sets $\mathsf{C}_{n,\,j,\,k}$.

Proposition 2.5. *Let M be a λ-term of size n with b binding and c non-binding lambda abstractions. Then $[\ M\ ]$ is an element of the set $\mathsf{C}_{2b+c+1,\,j,\,n-(b+c)}$, for some j such that $j \leq b+c+1$.*

Proof. The proof goes by induction on the structure of term M from Λ.

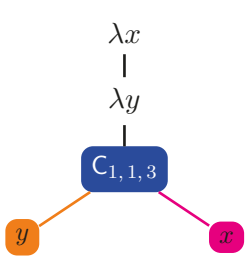
(i) If M is a variable, say x, then we have $|x|_\lambda = 1$ and $[\ x\ ] = x \in \mathsf{C}_{1,\,1,\,1}$.

(ii) Let M be an abstraction $\lambda x.N$ where $|N|_\lambda = n-1$ and N has b binding and c non-binding abstractions. By the definition of the translation, $[\ M\ ]$ equals $\lambda^* x.[\ N\ ]$. Using the induction hypothesis for N, we have $[\ N\ ] \in \mathsf{C}_{2b+c+1,\,j,\,k}$ for $k = n-1-(b+c)$ and some j such that $j \leq b+c+1$. If x does not occur in $[\ N\ ]$, we use Proposition 2.3 and obtain that $\lambda^* x.[\ N\ ]$ belongs to the set $\mathsf{C}_{2b+(c+1)+1,\,j+1,\,k}$. Otherwise, when x occurs in $[\ N\ ]$, by Proposition 2.4 we get $\lambda^* x.[\ N\ ] \in \mathsf{C}_{2(b+1)+c+1,\,j',\,k}$ with $j' = j$ or $j' = j+1$ and so we have $j' \leq (b+1)+c+1$.

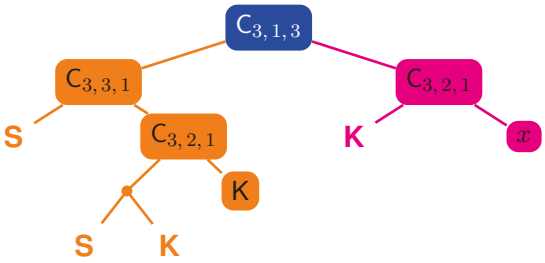
(iii) Let M be an application NO, where $|N|_\lambda = n_1$, $|O|_\lambda = n_2$ and $|NO|_\lambda = n = 1+n_1+n_2$. We assume that term M has b binding and c non-binding lambda abstractions. Since it is an application all of its abstractions are sub-terms of N and O. Let b_1 and c_1 denote respectively the number of binding and non-binding abstractions of N. Similarly, we define b_2 and c_2 for term O. From the definition of the standard translation we have $[\ M\ ] = [\ N\ ][\ O\ ]$. Using the induction hypothesis for both N and O we see that $[\ N\ ] \in \mathsf{C}_{2b_1+c_1+1,\,j_1,\,k_1}$ and $[\ O\ ] \in \mathsf{C}_{2b_2+c_2+1,\,j_2,\,k_2}$ such that $k_1 = n_1-(b_1+c_1)$ and $k_2 = n_2-(b_2+c_2)$. From this we obtain that $k_1+k_2 = n_1+n_2-(b_1+b_2+c_1+c_2) = n-1-(b+c)$. Since both $2b_1+c_1+1$ and $2b_2+c_2+1$ are bounded by $2b+c+1$, we can use Observation 2.2 and obtain that $[\ N\ ][\ O\ ]$ is in $\mathsf{C}_{2b+c+1,\,1,\,n-(b+c)}$. □

Now we know that the translation of a λ-term of size n belongs to the set $\mathsf{C}_{2b+c+1,\,j,\,n-(b+c)}$ for some $j \leq b+c+1$, where b and c denote the number of its binding and non-binding abstractions respectively. Using Observation 2.2, in order to find an upper limit for the size of that translation, we need to find the value of the expression $2b+c+1$ in the definition of $\mathsf{C}_{2b+c+1,\,j,\,n-(b+c)}$ (size n is fixed).

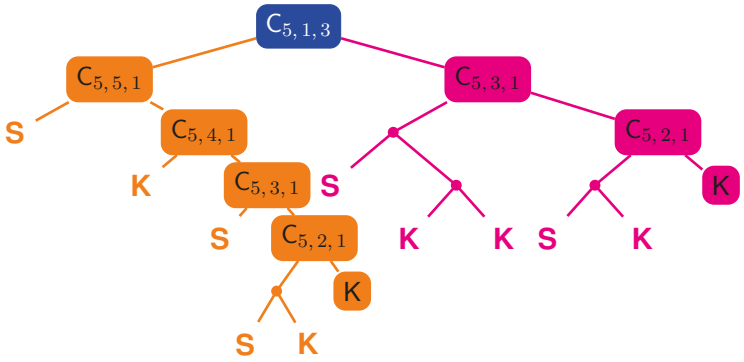
Observation 2.6. *Let M be a λ-term of size $|M|_\lambda = n > 2$ with b binding and c non-binding abstractions. Since each bound variable must*



(a) $\lambda xy.yx$



(b) $\lambda^* y.yx$



(c) $[\ \lambda xy.yx\ ] = \lambda^* x.\lambda^* y.yx$

Figure 5: Step-by-step translation of the term $\lambda xy.yx$ using the colored representation of terms of the set $\mathsf{C}_{n,j,k}$.

occur at least once, the tree representation has at least b leaves and hence at least $b-1$ inner nodes. It also consists of at most $n-1$ non-binding abstractions. Therefore, the following inequalities are satisfied:

$3b - 1 + c \leq n$ *(at least b occurrences of bound variables in total, b binding abstractions and at least $b-1$ inner nodes for each bound variable),*

$c \leq n - 1$.

Proposition 2.7. *For a λ-term M of size n with b binding and c non-binding lambda abstractions, combinator $[\ M\ ]$ is an element of the set $\mathsf{C}_{n+1,j,n-(b+c)}$, for some j such that $j \leq b + c + 1$.*

Proof. Due to Proposition 2.5, we only have to find an upper limit for the expression $2b + c + 1$ used in the definition of $\mathsf{C}_{2b+c+1,j,n-(b+c)}$. From

Observation 2.6, for any λ-term M of size n, we know that $n \geq 3b - 1 + c$ and $n \geq c + 1$. Using it, we can define a linear optimization problem of the following form (in which n is treated as a constant):

$$\begin{aligned} \text{maximize:} \quad & 2b + c + 1 \\ \text{constraints:} \quad & 3b - 1 + c \leq n \\ & c + 1 \leq n \\ & b, c \geq 0 \end{aligned}$$

Therefore, $2b+c+1 \leq n+\frac{4}{3}$. Since we are interested in the integer solution, we get $2b + c + 1 \leq n + 1$. Hence, using Observation 2.2, the translation $[\, M \,]$ belongs to the set $\mathsf{C}_{n+1,\,j,\,n-(b+c)}$ for some $j \leq b + c + 1$. □

Interpreting this proposition using the colored tree representation, it claims that if we translate a term of size n, then the colored version of that translation has at most n different colors, where for each color the number of closed terms of that color of size smaller than $2n$ is limited by $n + 1$. From this, we can conclude that it has at most $2(n+1)n+1$ nodes of each color in total. In the next step we determine an upper bound for the size of all elements of a set $\mathsf{C}_{n,\,j,\,k}$.

Proposition 2.8. *The size of a combinator $P \in \mathsf{C}_{n,\,j,\,k}$, i.e. $|P|_{\mathcal{CL}}$, is bounded from above by $2n((n-1)(k-1)+j-1)+k$.*

Proof. The proof goes by induction on the derivation of combinator P from the set $\mathsf{C}_{n,\,j,\,k}$.

(i) If $P \in \mathsf{C}_{n,\,1,\,1}$, then $P \equiv x$ or $P \equiv K$ and so $|P|_{\mathcal{CL}} = 1$ for every $n \geq 1$.

(ii) If $P \equiv QR \in \mathsf{C}_{n,\,j,\,k}$ where R is from $\mathsf{C}_{n,\,j-1,\,k}$, then Q is a closed term such that $|Q|_{\mathcal{CL}} < 2n$, and so $|Q|_{\mathcal{CL}} \leq 2n - 1$. We use the induction hypothesis for R to obtain:

$$\begin{aligned} |P|_{\mathcal{CL}} &= 1 + |Q|_{\mathcal{CL}} + |R|_{\mathcal{CL}} \\ &\leq 1 + (2n-1) + 2n((n-1)(k-1)+j-2)+k \\ &= 2n((n-1)(k-1)+j-1)+k. \end{aligned}$$

(iii) If $P \equiv Q\,R \in \mathsf{C}_{n,\,1,\,k}$, then Q is in $\mathsf{C}_{n,\,j_1,\,k_1}$ and R is in $\mathsf{C}_{n,\,j_2,\,k_2}$ such

that $k = k_1 + k_2 + 1$ and $j_1, j_2 \leq n$. Using the induction hypothesis for terms Q and R we obtain:

$$\begin{aligned}|P|_{\mathcal{CL}} &= 1 + |Q|_{\mathcal{CL}} + |R|_{\mathcal{CL}} \\ &\leq 1 + 2n((n-1)(k_1-1) + j_1 - 1) + k_1 \\ &\quad + 2n((n-1)(k_2-1) + j_2 - 1) + k_2 \\ &\leq 2n(n-1)(k-1) + k. \end{aligned}$$ □

By Proposition 2.7 we know that for every λ-term M of size n with b binding and c non-binding abstractions, its translation belongs to the set $\mathsf{C}_{n+1,\,j,\,n-(b+c)}$, for some j such that $j \leq b + c + 1$. Since $j \leq b + c + 1$, we have $n - (b + c) \leq n - j + 1$. From this, we are able to define the set T^n described earlier in (3):

$$\mathsf{T}^n \coloneqq \bigcup \{\mathsf{C}_{n+1,\,j,\,k} : \; j \leq n, \; k \leq n - j + 1\}. \tag{4}$$

We use this definition to calculate an upper bound on the size growth after translation.

Theorem 2.9. *Let M be a λ-term of size $n \geq 1$. The size of its translation into combinatory logic using the standard translation is bounded from above by $2n^3 - n$.*

Proof. Let b and c denote respectively the number of binding and non-binding abstractions of a term M. Using the definition (4) of the set T^n we see that $[\ M\ ] \in \mathsf{C}_{n+1,\,j,\,k}$ where $j \leq n$ and $k \leq n - j + 1$. From Proposition 2.8 we obtain:

$$\begin{aligned}|[\ M\ ]|_{\mathcal{CL}} &\leq 2(n+1)(n(k-1) + j - 1) + k \\ &\leq 2(n+1)(n(n-j) + j - 1) + n - j + 1. \end{aligned}$$

For any fixed value of the parameter n, the last part of this inequality is a strictly decreasing function in j. Therefore, it attains its maximum for $j = 1$. This gives an upper bound of $|[\ M\ ]|_{\mathcal{CL}}$ equal to $2n^3 - n$. □

2.2 Lower bound

In this section we study a particular class of λ-terms in order to obtain a lower bound of the function $\mathit{SizeT}(n)$. These are namely terms of the

form $\lambda x_k x_{k-1} \dots x_1 \,.\, x_1 \dots x_{k-1} x_k$. We show that for such terms, being of size $n = 3k-1$, the standard translation outputs combinators of size exactly $\frac{2}{81}n^3 + \frac{8}{27}n^2 + \frac{80}{27}n - \frac{187}{81}$. We also consider terms for other possible values of the size parameter n, namely $n = 3k$ and $n = 3k+1$. For that part, we simply construct terms which have similar shape as in the former case and show that their size is also of order $\Omega(n^3)$. Using it, we conclude that $SizeT(n)$ is of order $\Omega(n^3)$.

Definition 2.10. For all positive integers i, j and k we define inductively the following combinators:

$W_k = x_1 x_2 \dots x_k,$

$A_{i,k} = \lambda^* x_i. \dots \lambda^* x_1 . W_k,$

$S_{1,k} = S\ (\lambda^* x_1 . W_k),$
$S_{i,k} = S\ (\lambda^* x_i . S_{i-1,k}) \quad \text{for } 1 < i \leq k,$

$N_1 = S,$
$N_j = S\ (K\ N_{j-1}) \quad \text{for } j > 1,$

$P_{j,j,k} = N_i\ A_{i,k} \quad \text{for } 1 \leq j \leq k,$
$P_{i,j,k} = N_i\ P_{i+1,j,k} \quad \text{for } 1 \leq i < j \leq k.$

Our goal now is to characterize combinators $A_{k,k}$ using the above definitions. First we decompose it into some smaller parts and show that every term $A_{k,k}$ contains a sub-term described by the definition of $S_{k,k}$. Then we focus on describing it by means of the terms $P_{1,k,k}$. Due to its recursive definition we are able to compute its exact size. Now we present a series of propositions which lead us to determine the size of the term $A_{k,k}$.

Observation 2.11. *For every i, j and k such that $1 \leq i \leq j \leq k$ we have $\mathsf{FV}_{\mathcal{CL}}(S_{j,k}) = \mathsf{FV}_{\mathcal{CL}}(P_{i,j,k}) = \mathsf{FV}_{\mathcal{CL}}(W_k) \backslash \{x_1, \dots, x_j\} = \{x_{j+1}, \dots, x_k\}$.*

Proposition 2.12.

(i) *For every $k \geq 2$ and $i < k$ we have $A_{i,k} = S_{i,k-1}(\lambda^* x_i. \dots \lambda^* x_1 . x_k)$.*

(ii) *For every $k \geq 2$ we have $A_{k,k} = S(K\ (S_{k-1,k-1}))(\lambda^* x_k. \dots \lambda^* x_1 . x_k)$.*

Proof.

(i) The proof is by induction on parameters i and k.

$$\begin{aligned} A_{1,2} &= \lambda^* x_1.x_1x_2 = S(\lambda^* x_1.x_1)(\lambda^* x_1.x_2) \\ &= S_{1,1}(\lambda^* x_1.x_2), \end{aligned}$$

$$\begin{aligned} A_{1,k} &= \lambda^* x_1.x_1x_2\ldots x_k = S\ (\lambda^* x_1.x_1x_2\ldots x_{k-1})(\lambda^* x_1.x_k) \\ &= S_{1,k-1}(\lambda^* x_1.x_k), \end{aligned}$$

$$\begin{aligned} A_{i,k} &= \lambda^* x_i.\ldots.\lambda^* x_1.x_1x_2\ldots x_k \\ &= \lambda^* x_i.(S_{i-1,k-1})(\lambda^* x_{i-1}.\ldots.\lambda^* x_1.x_k) && \text{(induction hypothesis)} \\ &= S\ (\lambda^* x_i.S_{i-1,k-1})(\lambda^* x_i.\ldots.\lambda^* x_1.x_k) && \text{(Observation 2.11)} \\ &= S_{i,k-1}(\lambda^* x_i.\ldots.\lambda^* x_1.x_k). \end{aligned}$$

(ii) This part follows from the previous point and the fact that term $S_{k-1,k-1}$ has no free variables:

$$\begin{aligned} A_{k,k} &= \lambda^* x_k.\ldots.\lambda^* x_1.x_1x_2\ldots x_k \\ &= \lambda^* x_k.S_{k-1,k-1}(\lambda^* x_{k-1}.\ldots.\lambda^* x_1.x_k) && \text{(Proposition 2.12(i))} \\ &= S(K\ (S_{k-1,k-1}))(\lambda^* x_k.\ldots.\lambda^* x_1.x_k) && \text{(Observation 2.11).} \end{aligned}$$ □

Proposition 2.13. (i) *For every i, j and k such that $1 \leq i \leq j < k$ we have $\lambda^* x_{j+1}.P_{i,j,k} = P_{i+1,j+1,k}$.*

(ii) *For every j and k such that $1 \leq j \leq k$ we have $S_{j,k} = P_{1,j,k}$.*

Proof.

(i) The proof is by induction on the parameter i. From Observation 2.11 and the assumption that $j < k$ we have $x_{j+1} \in \mathsf{FV}_{\mathcal{CL}}(P_{i,j,k})$. Following the definitions of $P_{i,j,k}$ and bracket abstraction we obtain:

$$\begin{aligned} \lambda^* x_{j+1}.P_{j,j,k} &= \lambda^* x_{j+1}.N_j\ A_{j,k} \\ &= S(KN_j)\ (\lambda^* x_{j+1}.A_{j,k}) && \text{(Observation 2.11)} \\ &= N_{j+1}\ A_{j+1,k} \\ &= P_{j+1,j+1,k}, \end{aligned}$$

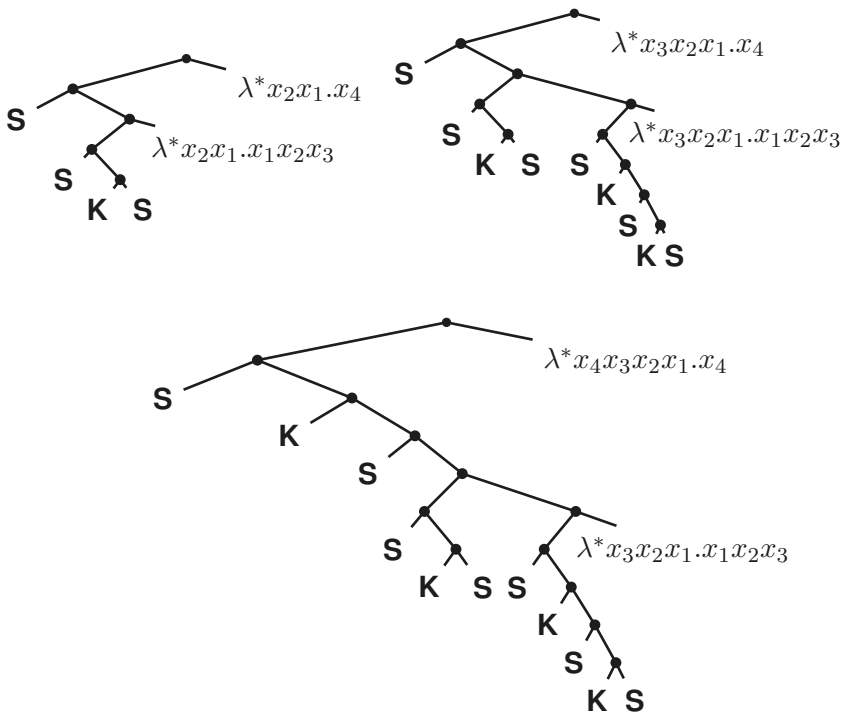


Figure 6: Three consecutive steps of the translation of the term $\lambda x_4 x_3 x_2 x_1 . x_1 x_2 x_3 x_4$.

$$\begin{aligned}
\lambda^* x_{j+1}.P_{i,j,k} &= \lambda^* x_{j+1}.N_i\ P_{i+1,j,k} \\
&= S(KN_i)\ (\lambda^* x_{j+1}.P_{i+1,j,k}) && \text{(Observation 2.11)} \\
&= N_{i+1} P_{i+2,j+1,k} && \text{(induction hypothesis)} \\
&= P_{i+1,j+1,k}.
\end{aligned}$$

(ii) The proof is by induction on the parameter j.

$$S_{1,k} = S\ (\lambda^* x_1.W_k) = N_1\ A_{1,k} = P_{1,1,k},$$

$$\begin{aligned}
S_{j,k} &= S\ (\lambda^* x_j.S_{j-1,k}) \\
&= S\ (\lambda^* x_j.P_{1,j-1,k}) && \text{(induction hypothesis)} \\
&= N_1\ P_{2,j,k} && \text{(Proposition 2.13)} \\
&= P_{1,j,k}.
\end{aligned}$$

□

Proposition 2.14. *For every $k \geq 1$ the size of $P_{1,k,k}$ equals $|P_{1,k,k}|_{\mathcal{CL}} = \sum_{i=1}^{k}(4i-2) + |A_{k,k}|_{\mathcal{CL}}$.*

Proof. By the inductive definition of $P_{1,k,k}$ we get:

$$|P_{i,k,k}|_{\mathcal{CL}} = |N_i\ P_{i+1,k,k}|_{\mathcal{CL}} = 1 + |N_i|_{\mathcal{CL}} + |P_{i+1,k,k}|_{\mathcal{CL}},$$
$$|P_{k,k,k}|_{\mathcal{CL}} = |N_k\ A_{k,k}|_{\mathcal{CL}} = 1 + |N_k|_{\mathcal{CL}} + |A_{k,k}|_{\mathcal{CL}}.$$

Since $|N_i|_{\mathcal{CL}} = 4i-3$ we have:

$$|P_{1,k,k}|_{\mathcal{CL}} = k + \sum_{i=1}^{k}|N_i|_{\mathcal{CL}} + |A_{k,k}|_{\mathcal{CL}} = k + \sum_{i=1}^{k}(4i-3) + |A_{k,k}|_{\mathcal{CL}}$$
$$= \sum_{i=1}^{k}(4i-2) + |A_{k,k}|_{\mathcal{CL}}. \qquad \square$$

Proposition 2.15. *For every $k \geq 1$ the size of $\lambda^* x_k. \ldots \lambda^* x_1.x_k$ is equal to $6k-1$.*

Proof. First, we notice that $\lambda^* x_{k-1}. \ldots \lambda^* x_1.x_k = \overbrace{K(K(\ldots(K}^{k-1 \text{ times}} x_k)\ldots))$
From this we have $M := \lambda^* x_k. \ldots \lambda^* x_1.x_k = \lambda^* x_k.K(K(\ldots(Kx_k)\ldots))$.
Using the definition of bracket abstraction we obtain
$M = \underbrace{S(KK)(S(KK}_{k-1 \text{ times}}(\ldots(S(KK)(SKK))\ldots))$.
Therefore, $|\lambda^* x_k. \ldots \lambda^* x_1.x_k|_{\mathcal{CL}} = 6(k-1) + |SKK|_{\mathcal{CL}} = 6k-1$. $\square$

Proposition 2.16. *For every $k > 0$ the size of $A_{k,k}$ equals $\frac{2}{3}k^3 + 2k^2 + \frac{22}{3}k - 5$.*

Proof. By Proposition 2.12

$$A_{k,k} = S\,(K\,(\,S_{k-1,k-1}\,)\,)\,(\,\lambda^* x_k. \ldots \lambda^* x_1.x_k\,).$$

Then, to calculate the size of $S_{k-1,k-1}$ we use Propositions 2.13 and 2.14 and obtain:

$$|S_{k-1,k-1}|_{\mathcal{CL}} = |P_{1,k-1,k-1}|_{\mathcal{CL}} = \sum_{i=1}^{k-1}(4i-2) + |A_{k-1,k-1}|_{\mathcal{CL}}.$$

Together with Proposition 2.15, which establishes the size of a combinator of the form $\lambda^* x_k \ldots x_1.x_k$, we get the following equation:

$$\begin{aligned}|A_{k,k}|_{\mathcal{CL}} &= |S(K\ (S_{k-1,k-1}))(\lambda^* x_k.\ldots\lambda^* x_1.x_k)|_{\mathcal{CL}} \\ &= 5 + |\lambda^* x_k.\ldots\lambda^* x_1.x_k|_{\mathcal{CL}} + |S_{k-1,k-1}|_{\mathcal{CL}} \\ &= 5 + 6k - 1 + \sum_{i=1}^{k-1}(4i-2) + |A_{k-1,k-1}|_{\mathcal{CL}} \\ &= 6 + 4\sum_{i=1}^{k} i + |A_{k-1,k-1}|_{\mathcal{CL}}.\end{aligned}$$

In order to obtain the closed form of this recurrence, we easily compute that its solution is $6k - 5 + 4\sum_{i=1}^{k}\sum_{j=1}^{i} j$. Therefore,

$$\begin{aligned}|A_{k,k}|_{\mathcal{CL}} &= 6k - 5 + 4\sum_{i=1}^{k}\sum_{j=1}^{i} j \\ &= \frac{2}{3}k^3 + 2k^2 + \frac{22}{3}k - 5.\end{aligned}$$

□

Proposition 2.17. *The standard translation algorithm maps terms of the form $\lambda x_k \ldots x_2 x_1\,.\,x_1 x_2 \ldots x_k$, which are of size $n = 3k-1$, into combinators of size $\frac{2}{81}n^3 + \frac{8}{27}n^2 + \frac{80}{27}n - \frac{187}{81}$.*

Proof. It follows from Proposition 2.16 by substituting $\frac{n+1}{3}$ for k in the expression $\frac{2}{3}k^3 + 2k^2 + \frac{22}{3}k - 5$. □

Using this proposition we can deduce what is the lower bound for the worst-case space complexity.

Theorem 2.18. *The worst-case space complexity of the standard translation is of order $\Omega(n^3)$.*

Proof. In previous propositions we focused on the case where the size parameter n is of the form $n = 3k-1$ for every possible value of k. We also need to consider other possible values of n. When n is equal to $3k$, we may consider terms of the form $A_{k+1,k}$. By the definition of the translation we obtain that $[\ A_{k+1,k}\ ] = K\,[\ A_{k,k}\ ]$. Using Proposition 2.17 we can calculate its exact size, which is $\frac{2}{81}n^3 + \frac{2}{9}n^2 + \frac{22}{9}n - 3$. Similarly, for the case when n equals $3k+1$ we consider terms $A_{k+2,k}$. After simple calculations, we

Size	Terms scheme	Expected size
$n = 3k - 1$	$\lambda x_1 \dots x_k \,.\, x_1 \dots x_k$	$\frac{2}{81}n^3 + \frac{14}{27}n^2 + \frac{74}{27}n - \frac{223}{81}$
$n = 3k$	$\lambda x_1 \dots x_k x_{k+1} \,.\, x_1 \dots x_k$	$\frac{2}{81}n^3 + \frac{4}{9}n^2 + \frac{28}{9}n - 3$
$n = 3k + 1$	$\lambda x_1 \dots x_k \,.\, x_1 \dots x_k x_1$	$\frac{2}{81}n^3 + \frac{16}{27}n^2 + \frac{68}{27}n - \frac{659}{81}$

Table 1: Predictions of the worst-case terms for the standard translation algorithm.

obtain that the size of its translation is $\frac{2}{81}n^3 + \frac{4}{27}n^2 + \frac{56}{27}n - \frac{263}{81}$. These, together with Proposition 2.17, allows us to conclude that the worst-case space complexity of the standard translation is of order $\Omega(n^3)$. □

3. Conclusions and future work

From Theorems 2.9 and 2.18 we see that the worst-case size of a term produced by the standard translation is of order $\Theta(n^3)$:

$$\frac{2}{81}n^3 + \frac{4}{27}n^2 + \frac{56}{27}n - \frac{263}{81} \leq \mathit{SizeT}(n) \leq 2n^3 - n.$$

There is still room for improvement since these bounds are not tight.

We used a Haskell program which computes exact values of the function $\mathit{SizeT}(n)$ to conduct several experiments. Based on the results, we state a conjecture concerning the shape of the worst-case terms. If we optimistically assume that the size of these terms after translation can be expressed by some polynomial, then we can simply interpolate the growth rate for each of them. These predictions are listed in Table 1. We believe that our method of analysis of the upper bound can be easily adapted to investigate other translation algorithms as well, e.g. the translation using Turner's combinators used in [Tur79]. We would also like to provide a lower bound for the space complexity of all possible translations. Then, we can investigate whether it is possible to design an algorithm that is asymptotically optimal. The analysis of its space complexity can be used to determine an upper limit of the number of λ-terms of a given size, which is still an open problem.

Acknowledgements

We would like to thank Marek Zaionc, Katarzyna Grygiel and Agnieszka Łupińska for inspiration and many fruitful comments and suggestions.

References

[1] H.P. Barendregt, The Lambda Calculus: Its Syntax and Semantics, College Publications, London, 2012.

[2] D.A. Turner, Another Algorithm for Bracket Abstraction, *The Journal of Symbolic Logic* **44**:2 (1979), 729–740.

[3] S. Broda and L. Damas, Compact bracket abstraction in combinatory logic, *The Journal of Symbolic Logic* **62**:3 (1997), 729–740.

[4] A. Church, An Unsolvable Problem of Elementary Number Theory, *American Journal of Mathematics* **58**:2 (1936), 345–363.

[5] H.B. Curry, Grundlagen der Kombinatorischen Logik, *American Journal of Mathematics* **3** (1930), 509–536.

[6] M.S. Joy, On the efficient implementation of combinators as an object code for functional programs, PhD Thesis University of East Anglia (1985).

[7] R. Kennaway and M.R. Sleep, Variable Abstraction in $O(n \log n)$ Space, *Information Processing Letters* **24**:5 (1987), 343–349.

[8] Ł. Lachowski, l2cl, A computer program which searches for worst-case instances of the standard translation. Available at: https://bitbucket.org/zbyszko/l2cl, 2017.

[9] K. Noshita, Translation of Turner Combinators in $O(n \log n)$ Space, *Information Processing Letters* **20**:2 (1985), 71–74.

[10] M. Schönfinkel, Über die Bausteine der mathematischen Logik, *Mathematische Annalen* **92**:3 (1924), 305–316.

[11] D.A. Turner, Another Algorithm for Bracket Abstraction, *The Journal of Symbolic Logic* **44**:2 (1979), 267–270.

Department of Theoretical Computer Science
Faculty of Mathematics and Computer Science
Jagiellonian University
Łojasiewicza 6, 30-348 Kraków, Poland

lukasz.lachowski@tcs.uj.edu.pl