Average Size of Turner's Translation to Combinator Programs

by

Teruo Hikita

(Tokyo Metropolitan University)

Abstract.

D. A. Turner proposed in 1979 an interesting method of implementation for functional programs by first translating them to combinator expressions and then reducing the graphs they represent. J. R. Kennaway has recently shown that the worst case of the size of this translation is of order n^2 where n is the size of an original program. In this paper we study the average size of the translation and show that the order of the average is bounded by $n^{3/2}$. Results on lower bounds of the average are also shown in the case of programs with one variable. Finally numerical statistics for the average size are exhibited, showing that the size expansion is of reasonable range.

1. Introduction

Turner [6] showed in 1979 that use of combinators can be a neat and efficient method of implementation for functional programs. In his implementation of the applicative language SASL first a user's program is translated, by syntax-desugaring and abstracting variables, into an expression composed of constants, predefined functions and several special kinds of combinators. Then the graph which the translated expression represents is transformed (reduced) to a result value. Among these special combinators some are classical in combinatory logic [2], and the others devised by Turner, the latter playing an essential role in preventing the exponential growth of the size of the translated expression [7]. One of the advantages of this method of implementation is a coherent and efficient treatment of higher order functions. Several implementations both by software and hardware have since then appeared along this line [5].

Still, in general, the sizes of the translated expressions expand, and the analysis of this expansion has fundamental importance since the size directly relates with the time and space of the translation. Kennaway [3] has recently shown that in the worst case the size of the translated expression is of order n^2 where n is the size of an original program. Burton [1] has shown that a certain pre-transformation of original programs can reduce the size expansion to a linear one, provided the programs do not contain global variables.

In this paper we study the average behavior of the size of Turner's translation. In particular we will show that the order of average size of translated expressions is bounded by $n^{3/2}$. We will also give some partial results on the lower bounds of the average size in the case of programs with one variable. Finally we will exhibit the numerical statistics of the average size obtained by computer computation.

2. Turner's translation scheme

We will not explain here the basic facts of λ-calculus and combinatory logic, but the reader is referred to an excellent exposition in [6]. We will, however, show a small example of translation of a function pick which selects and returns the n-th element of a given sequence s.

Example 1.

In Example 1, first the syntax-sugaring of program (1) is removed and a tree-like program (2) is obtained. Here, and throughout this paper, association of application is to the left, which is usual in combinatory logic; thus abc means (ab)c. Next the variable s is eliminated by "abstraction" in the righthand-side expression of (2) using the translation scheme explained later, obtaining program (3). Finally the variable n is abstracted in (3) and program (4) is obtained, which is the object program for graph transformation. In this example the size of the original program (2) is 12, counting the number of constant values, variables and constant functions, while that of the translated program (4) is 18, counting constants and combinators.

In general, an original expression consists of constants and variables composed by the operation of application. The translation proceeds by abstracting each variable successively from the expression, and the final result consists of constants and combinators. Although Turner [6], [7] gives the way of translation in detail for functional programs to combinator expressions, the description there does not seem to be clear enough to be stated without ambiguity. Kennaway [3] thus gives the translation scheme in a more decisive way. We here state the scheme in the following form as Scheme T.

Scheme T.

In this scheme the variable of the current abstraction is denoted by x, and abstractions for subexpressions E, F and G are denoted by E*, F* and G*, respectively. The definitions of the combinators used are as follows:

$K \equiv \lambda xy.x, \qquad I \equiv \lambda x.x,$

$B \equiv \lambda xyz.x(yz), \qquad C \equiv \lambda xyz.xzy, \qquad S \equiv \lambda xyz.xz(yz),$

$B' \equiv \lambda wxyz.wx(yz), \quad C' \equiv \lambda wxyz.w(xz)y, \quad S' \equiv \lambda wxyz.w(xz)(yz).$

Note that all the cases in the scheme are exhaustive and mutually exclusive. The abstraction by a variable proceeds by recursively applying the scheme to an original expression.

Remark. The scheme T in [3] is slightly different from our present scheme T; in [3], (1) is replaced by "E_1 contains variables" and (2) is replaced by "E consists of constants and combinators". The size of the translation by the scheme T of [3] is slightly smaller than that by our scheme T, and the difference is marginal.

In the scheme T the extensionality rule (η-rule)

$\lambda x.Mx = M \qquad$ (x does not occur in M)

is incorporated in several places. This rule is entirely removed in our second scheme U, which will be useful in the analysis of the size as an intermediate step [3]. Again all the cases in the scheme U are both exhaustive and mutually exclusive.

Scheme U.

3. Results by Kennaway

In this section we briefly review the results by Kennaway [3], which will be utilized in the following sections. Throughout the paper we often regard an expression as a binary tree where each interior node stands for application and each leaf node is labelled by a constant, variable or combinator name. Let s(E, x) denote the minimal spanning subtree in the tree representing an expression E such that the subtree shares its root with the tree and it contains all occurrences of a variable x at the leaf nodes. We denote by $|E|$ the size of the expression and by $|s(E, x)|$ the number of interior nodes of the subtree s(E, x). Kennaway has shown the following key lemma for the translation U.

Lemma 3.1. Let V be the set of variables occurring in an expression E. Then the size u(E) of the translated expression of E by abstracting the variables in V from E using the scheme U is

$$u(E) = |E| + \sum_{x \in V} |s(E, x)|.$$

Using this lemma Kennaway has shown the following results. Let $\tilde{E}_{n,k}$ denote the set of expressions of size n with k distinct variables.

Theorem 3.2. For an expression E in $\tilde{E}_{n,k}$ we have

$$u(E) \leq (k+1)n - (k^2-k+2)/2.$$

Let t(E) be the size of the translated expression of an expression E by using the scheme T of Turner.

Theorem 3.3. For the translation scheme T we have

$$t(E) \leq 2n-1 \qquad \text{if } E \in \tilde{E}_{n,1},$$

$$t(E) \leq 3n-3 \qquad \text{if } E \in \tilde{E}_{n,2},$$

$$t(E) \leq (k+1)n - (k^2-k+6)/2 \qquad \text{if } E \in \tilde{E}_{n,k},\ k \geq 3.$$

Let $\tilde{E}_n = \bigcup_{k=1}^{n} \tilde{E}_{n,k}$ be the set of expressions of size n.

Corollary 3.4. We have

$$t(E) \leq (n^2 + 3n - 6)/2$$

for $E \in \tilde{E}_n$, $n \geq 3$.

Remark. The equality in Corollary 3.4 is achieved by the expression

$$x_1(x_2(\ldots(x_{n-1}x_n)\ldots)).$$

Lemma 3.1 tells us that in the case of the scheme U the order of abstractions of the distinct variables in an expression does not affect the final size of the translated expression. However, in the case of the scheme T the order among variables crucially affects the size of the resulting expression, as will be seen in a later example.

4. An upper bound of average size

The average of the sizes of translated expressions is taken over all distinct original expressions of fixed n and k, where n is the size of the expressions and k is the number of distinct variables in the expressions. We assume that each of these distinct expressions occur in equal probability. For example, when $n = 2$ and $k = 1$ we consider three expressions ax, xa and xx, while when $n = 2$ and $k = 2$ there are two: xy and yx. When $n = 3$ and $k = 2$ there are 24 distinct original expressions, which are shown in Example 2 together with their translations by the schemes U and T; there are only two cases when the translations by the two schemes coincide, which are indicated by asterisks. The average sizes under U and T in this case are 6.50 and 3.79, respectively.

Example 2.

For fixed n and k, $1 \leq k \leq n$, there are C_{n-1} distinct rooted binary ordered trees with n leaves where $C_{n-1} = \binom{2n-2}{n-1}/n$ is the (n-1)-th Catalan number. And there are

$$A(n, k) = \sum_{\substack{r_1+\ldots+r_k \leq n \\ r_1,\ldots,r_k \geq 1}} \binom{n}{r_1}\binom{n-r_1}{r_2} \cdots \binom{n-r_1-\ldots-r_{k-1}}{r_k}$$

assignments of k distinct variables in the n leaves of each tree. Thus there are in total $C_{n-1} \times A(n, k)$ distinct original expressions of size n with k distinct variables.

Let $\bar{u}(n, k)$ and $\bar{t}(n, k)$ denote the average sizes of the translated expressions of all distinct original expressions in $\tilde{E}_{n,k}$ under the schemes U and T, respectively. First we need a definition. The _external path length_ of a tree is defined to be the sum, taken over all leaf nodes, of the lengths of the paths from the root to each leaf node. The following lemma is known (see e.g. [4], Section 2.3.4.5).

Lemma 4.1. The average of external path lengths over all binary trees with n leaf nodes is asymptotically $n\sqrt{\pi n}$.

Proposition 4.2. We have $\bar{u}(n, n) \sim n\sqrt{\pi n}$.

Proof: When k = n original expressions consist entirely of variables and each variable occurs just once. By Lemma 3.1 the size of the translated expression by the scheme U in this case is equal to the external path length of the tree which an original expression represents. Thus the proposition follows from Lemma 4.1.

Theorem 4.3. The order of the average size of the translation U is bounded by $n^{3/2}$.

Proof: This follows immediately from Proposition 4.2 and the obvious fact that

$$\bar{u}(n, k_1) \leq \bar{u}(n, k_2)$$

whenever $k_1 \leq k_2$.

Theorem 4.4. The order of the average size of the translation T is bounded by $n^{3/2}$.

Proof: This follows immediately from Theorem 4.3 and the obvious fact that $\bar{t}(n, k)$ is always less than $\bar{u}(n, k)$.

5. Lower bounds of average size

In this section some partial results on the lower bounds of average sizes will be given; those in the case of original expressions with only one variable.

Proposition 5.1. We have $\bar{u}(n, 1) \geq \frac{3}{2} n$ if $n \geqq 2$.

Proof: There are in total $(2^n-1) \times \binom{2n-2}{n-1}/n$ distinct original expressions in $\tilde{E}_{n,1}$. The following argument is independent of the shape of a tree which represents an expression. Let r be the number of occurrences of the unique variable in the expression, thus $1 \leqq r \leqq n$. When $1 \leqq r \leqq n - 1$, from Lemma 3.1 we can easily see that the size of the translated expression is at least $n + r$. When $r = n$ the size is always $2n - 1$. Thus, averaging the translated sizes over all 2^n-1 expressions which are represented by some tree, we have

$$\begin{aligned}\bar{u}(n, 1) &\geq \frac{1}{2^n-1} \{(n+1)\binom{n}{1} + (n+2)\binom{n}{2} + \dots + (2n-1)\binom{n}{n-1} + (2n-1)\} \\ &= n + \frac{1}{2^n-1}(n \cdot 2^{n-1} - 1) \\ &\geq \frac{3}{2} n.\end{aligned}$$

Lemma 5.2. We have

$$\frac{1}{3}\{\bar{u}(n, 1) - n\} \leqq \bar{t}(n, 1) - n.$$

Proof: Fix a tree with n leaf nodes. Then, by changing the assignment of occurrences of the (unique) variable at the leaf nodes, the tree can represent 2^n-1 distinct expressions with one variable. There are n-1 interior nodes in the tree. We divide these interior nodes into the following four classes according to whether each of its two son nodes is a leaf node (constant/variable) or an interior node (application), and we compare in each class the increases of the size at the node under the translation schemes U and T. In the following a and b denote a constant or a variable, E and F an application, and x the (unique) variable.

class 1. ab. (Both sons are constant/variable)

This class is further divided into four subcases according to whether each of a and b is constant or variable. The increases of the sizes under U and T, and the probability p that each subcase occurs in the class, are summarized as follows.

	$x \neq a, x \neq b$	$x \neq a, x = b$	$x = a, x \neq b$	$x = a, x = b$
U	0	+1	+1	+1
T	0	-1	+1	+1
p	$(2^{n-2}-1)/(2^n-1)$	$2^{n-2}/(2^n-1)$	$2^{n-2}/(2^n-1)$	$2^{n-2}/(2^n-1)$

class 2. Eb. (Only right son is constant/variable)

The situation of subcases is summarized as follows.

	$x \notin E, x \neq b$	$x \notin E, x = b$	$x \in E, x \neq b$	$x \in E, x = b$
U	0	+1	+1	+1
T	0	-1	+1	+1
p	$< (2^{n-2}-1)/(2^n-1)$	$< 2^{n-2}/(2^n-1)$	$> 2^{n-2}/(2^n-1)$	$> 2^{n-2}/(2^n-1)$

<u>class</u> 3. aF. (Only left son is constant/variable)

The increase of size at the node is 0 when $x \neq a$ and $x \notin F$, and +1 otherwise. The same under U and T.

<u>class</u> 4. EF. (Both sons are applications)

The situation is similar to class 3.

Now, summing up the increases of sizes over interior nodes for the schemes U and T and averaging over 2^n-1 expressions which are represented by the fixed tree, we obtain the lemma.

Proposition 5.3. We have $\bar{t}(n, 1) \geq \frac{7}{6} n$ if $n \geq 2$.

Proof: This follows immediately from Proposition 5.1 and Lemma 5.2.

The lower bounds of Propositions 5.1 and 5.3 are not very tight, as will be seen in the numerical statistics given in the next section.

6. Numerical statistics

In this section we exhibit the results of numerical statistics obtained by computer computation. Table 1 shows the average size of the translated expressions under the scheme U in the range $1 \leqq n \leqq 6$, and Table 2 is the corresponding one under the scheme T.

Table 1.

Table 2.

Figures 1 and 2 are drawn from Tables 1 and 2, respectively.

Figure 1.

Figure 2.

These tables and figures indicate the following facts.

1) When k is fixed, the average size is almost a linear function of n, both under the schemes U and T.

2) The coefficients of these linear functions, which specify the size expansion of the translation, are of reasonable size under the scheme T: the coefficients are ~ 1.7 when $k = 1$ and ~ 2.2 when $k = 2$.

3) When n is fixed, the average size is "less than" a linear function of k.

4) Comparing the average sizes under the schemes U and T, we can see that the extensionality rule plays an essential role in reducing the translation size, at least when n is small.

7. Concluding remarks

In this paper an initial study has been performed on the average size of the translation of functional programs to Turner combinators from both theoretical and empirical viewpoints. Some remarks drawn from our results are in order:

i) While the order of the translation size is n^2 in the worst case, it is at most $n^{3/2}$ in average. Of course the definition of average is subject to discussion, but this result is of some value in indicating the overall behavior of the translation size.

ii) When the number k of distinct variables appearing in a program is small, which seems common in real situations, the average size is almost a linear function of n and its coefficient is of reasonable size: ~ 1.7 when $k = 1$ and ~ 2.2 when $k = 2$.

These conclusions assure us that the method of implementation by combinators can be effective, at least concerning time and space of the translation phase.

References

[1] F. W. Burton : A linear space translation of functional programs to Turner combinators, Inform. Process. Lett., 14 (1982), 201-204.

[2] H. B. Curry, R. Feys and W. Craig : Introduction to Combinatory Logic, Vol. 1, North-Holland, Amsterdam, 1958.

[3] J. R. Kennaway : The complexity of a translation of λ-calculus to combinators, School of Computing Studies and Accountancy, University of East Anglia, Norwich, 1982.

[4] D. E. Knuth : The Art of Computer Programming, Vol. 1, Fundamental Algorithms, second ed., Addison-Wesley, Reading, Mass., 1973.

[5] S. L. Peyton Jones : An investigation of the relative efficiencies of combinators and lambda expressions, Conf. Rec. of the 1982 ACM Symp. on LISP and Functional Programming, pp. 150-158.

[6] D. A. Turner : A new implementation technique for applicative languages, Softw. Pract. Exper., 9 (1979), 31-49.

[7] D. A. Turner : Another algorithm for bracket abstraction, J. Symbolic Logic, 44 (1979), 267-270.

Example 1.   Translation through repeated abstractions.

```
(1)     def pick n s  =  if n = 1 then hd s else pick (n - 1) (tl s)

(2)     def pick n s  =  cond (eq n 1) (hd s) (pick (minus n 1) (tl s))

(3)     def pick n    =  S (B (cond (eq n 1)) hd) (B (pick (minus n 1)) tl)

(4)     def pick      =  S' S (C' B (B cond (C eq 1)) hd)

                              (C' B (B pick (C minus 1)) tl)
```

Example 2. Translations by the schemes U and T (n = 3 and k = 2).

	U	T	
x(ya)	B'CI(CIa)	B'CI(CIa)	*
x(ay)	B'CI(BaI)	B'CIa	
x(yy)	B'CI(SII)	B'CI(SII)	*
y(xa)	C'BI(CIa)	CB(CIa)	
a(xy)	B'Ba(B'CII)	B'Ba(CI)	
y(xy)	S'BI(B'CII)	SB(CI)	
y(ax)	C'BI(BaI)	CBa	
a(yx)	B'Ba(C'BII)	Ba	
y(yx)	S'BI(C'BII)	SBI	
x(xy)	B'SI(B'CII)	B'SI(CI)	
x(yx)	B'SI(C'BII)	SI	
y(xx)	C'BI(SII)	CB(SII)	
xya	C'C(B'CII)a	C'C(CI)a	
xay	B'C(CIa)I	C(CIa)	
xyy	S'C(B'CII)I	S'C(CI)I	
yxa	C'C(C'BII)a	CCa	
axy	B'C(BaI)I	Ca	
yxy	S'C(C'BII)I	SCI	
yax	C'B(CIa)I	CIa	
ayx	C'B(BaI)I	a	
yyx	C'B(SII)I	SII	
xxy	B'C(SII)I	C(SII)	
xyx	C'S(B'CII)I	C'S(CI)I	
yxx	C'S(C'BII)I	CSI	

Translation scheme T.

1.	a	constant, combinator, variable $\neq$ x	Ka
		variable = x	I
2.	E	E is an application, $x \notin E$	KE
3.	EF	E is not an application, or $E = E_1E_2$ and E_1 contains constants or variables(1)	
		$x \notin E, \quad x = F$	E
		$x \notin E, \quad x \in F, \quad x \neq F$	BEF*
		$x \in E, \quad x \notin F$	CE*F
		$x \in E, \quad x \in F$	SE*F*
4.	(EF)G	E consists of combinators(2)	
		$x = F, \quad x \notin G$	CEG
		$x = F, \quad x \in G$	SEG*
		$x \notin F, \quad x = G$	EF
		$x \notin F, \quad x \in G, \quad x \neq G$	B'EFG*
		$x \in F, \quad x \neq F, \quad x \notin G$	C'EF*G
		$x \in F, \quad x \neq F, \quad x \in G$	S'EF*G*

Translation scheme U.

1.	a	constant, combinator, variable $\neq$ x	Ka
		variable = x	I
2.	E	E is an application, x $\notin$ E	KE
3.	EF	E is not an application, or E = E_1E_2 and E_1 contains constants or variables	
		x $\notin$ E, x ε F	BEF*
		x ε E, x $\notin$ F	CE*F
		x ε E, x ε F	SE*F*
4.	(EF)G	E consists of combinators	
		x $\notin$ F, x ε G	B'EFG*
		x ε F, x $\notin$ G	C'EF*G
		x ε F, x ε G	S'EF*G*

n	1	2	3	4	5	6
k						
1	1.00	3.00	4.86	6.71	8.57	10.44
2		4.00	6.50	8.96	11.42	13.91
3			8.00	10.96	13.90	16.87
4				12.80	16.15	19.53
5					18.24	21.98
6						24.30

Table 1. Average translation size by the scheme U for programs of size n with k variables.

k \ n	1	2	3	4	5	6
1	1.00	2.33	4.14	5.83	7.50	9.17
2		1.50	3.79	6.17	8.38	10.54
3			3.00	6.17	8.96	11.61
4				5.96	9.39	12.51
5					9.70	13.31
6						14.02

Table 2. Average translation size by the scheme T for programs of size n with k variables.

Figure 1. Average translation size by the scheme U for programs of size n with k variables.

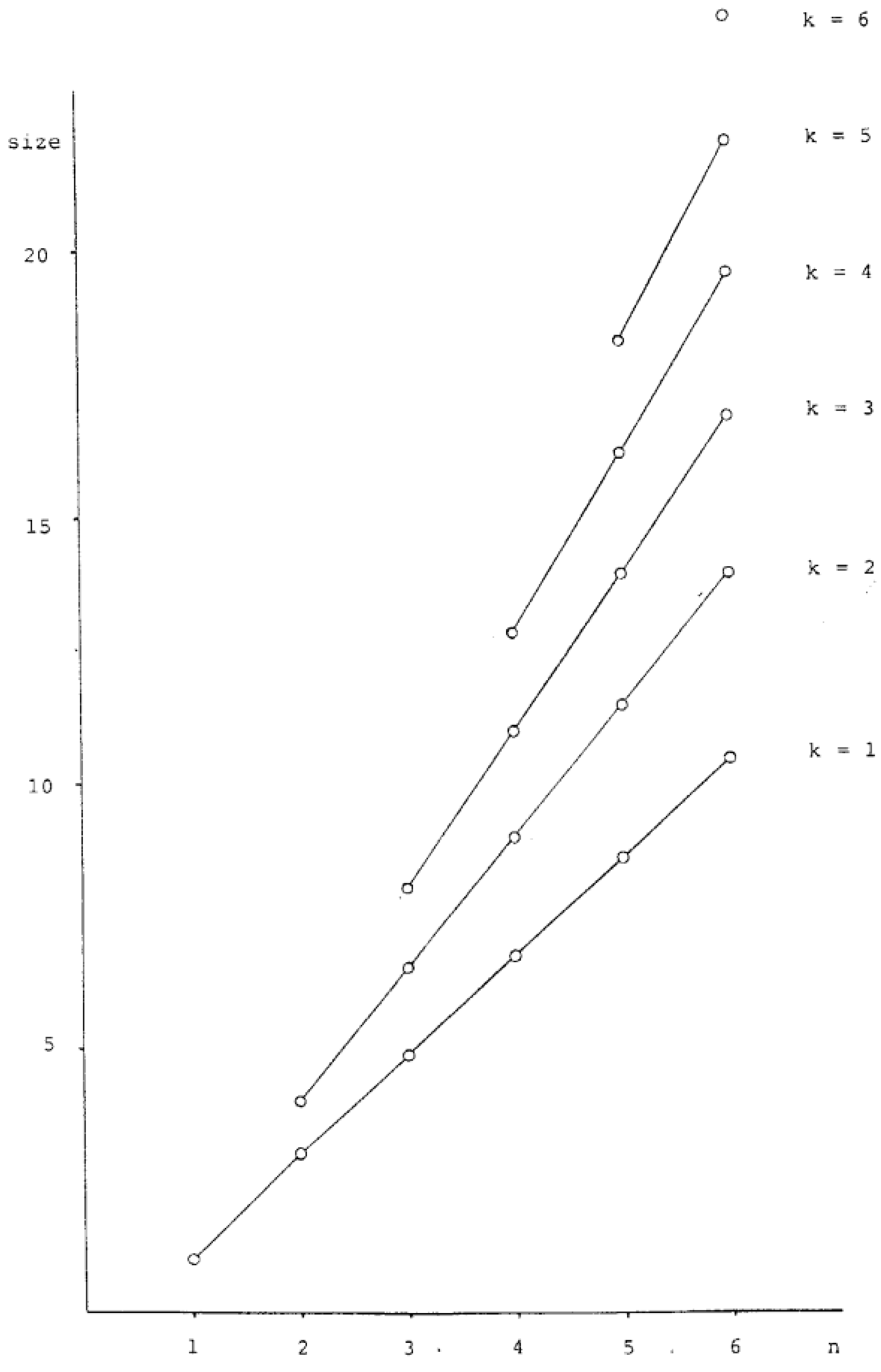
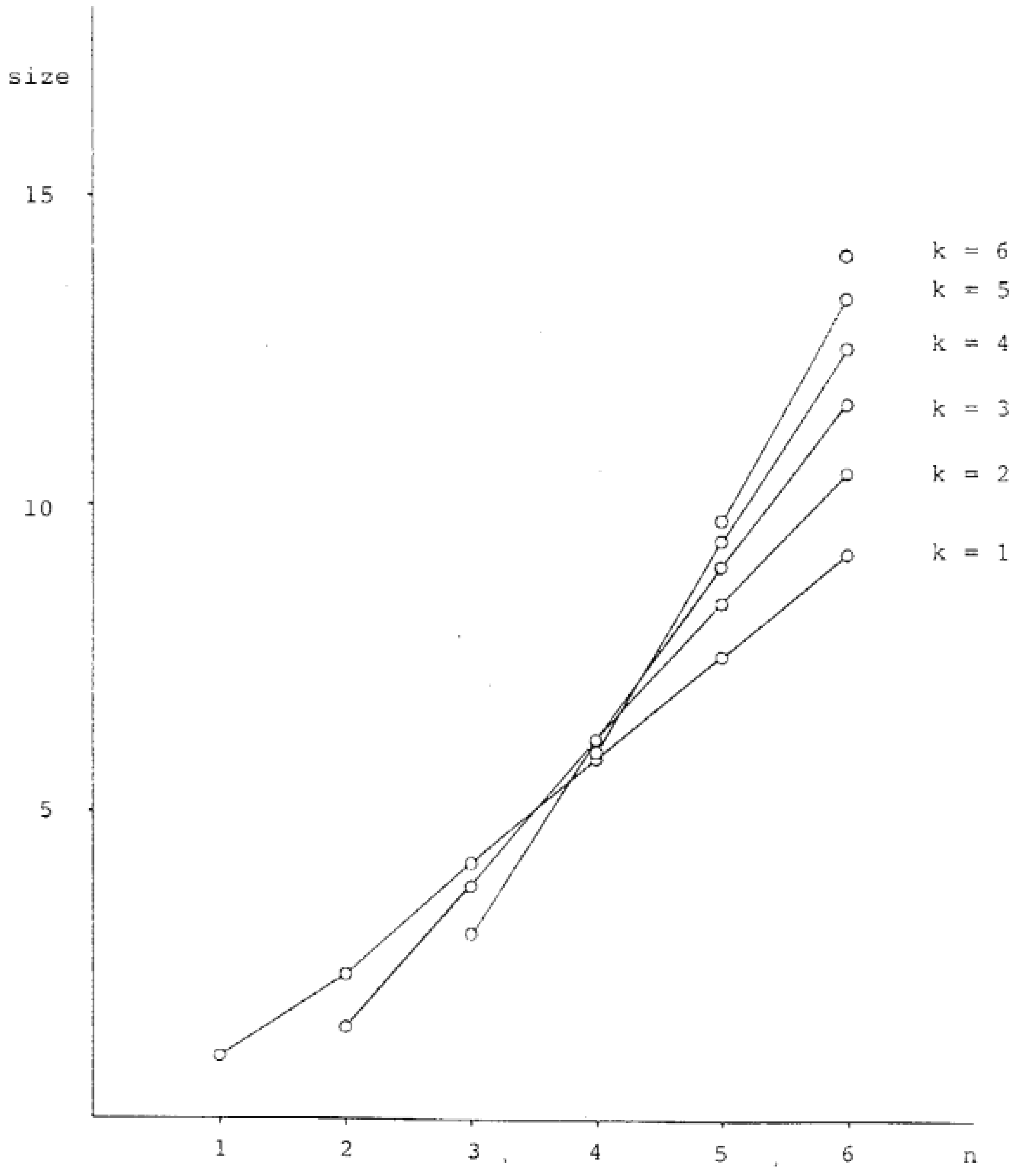


Figure 2. Average translation size by the scheme T for programs of size n with k variables.


size
n
k = 6
k = 5
k = 4
k = 3
k = 2
k = 1