ON SUBRECURSIVENESS IN WEAK COMBINATORY LOGIC

Carlo Batini - Alberto Pettorossi

Centro di Studio dei Sistemi di Controllo e Calcolo Automatici
C.N.R. - Roma

Istituto di Automatica dell'Università di Roma

ABSTRACT

In this paper weak combinatory logic as an algorithmic language is considered and various notions of structural and computational complexity are introduced. Particular attention is devoted to the definitional power of a system of combinators, that is to the concept of "subbase". Some results concerning the relations between specific subbases and their generative power are presented.

SUMMARY

1. INTRODUCTION

The main purpose of this work is to introduce the concepts of computational complexity and subrecursiveness in combinatory logic [1]. Since we are interested in the algorithmic and computational properties of combinatory logic, we think that a fruitful way of approaching these properties is to consider how limiting the power of the calculus (that is limiting the allowed definitions of "programs" or the allowed amount of "resource" used by "programs") entails limitations on the ability of manipulating "data".

Studies on computational complexity have given well established and meaningful results for several abstract machines and languages such as Turing machines, LOOP programs, rewriting systems [2]. Less considered seems to have been the definition of suitable concepts of computational complexity in combinatory logic and λ-calculus. An interesting step in this direction was made by H.R. Strong [3] who defined a measure of depth of computation in a programming language based on Wagner's URS and showed that for each partial recursive function there is an index with uniformly bounded measure of computation.

In order to carry on our investigation on complexity properties of combinatory logic in §2 and §3 we shall examine properties of structural and computational complexity of combinators in weak calculus [4] and in §4 we examine the ability of combinations in generating pure applicative combinations [1]. For reasons that will be made clear later we restricted ourselves to considering only proper combinators [1] or, in general, combinators for which it is possible to define suitable input-output relations.

2. APPROACHES TO COMPUTATIONAL COMPLEXITY IN COMBINATORY LOGIC

The two basic notions of complexity in the literature are structural and computational complexity.

We now introduce the formal definitions of some possible measures of these notions. Structural complexity is inherent to a combinator as a static well formed object in a specific base.

Definition 2.1. - *Length* (SL) of a combinator χ is recursively defined by the following rules:

i) if χ is a basic combinator, $SL(\chi) = 1$;

ii) if $\chi = (\chi_1\chi_2)$, $SL(\chi) = SL(\chi_1) + SL(\chi_2)$.

Definition 2.2. - *Depth of the parenthesis structure* (SD) of a combinatory χ is defined by:

i) if χ is a basic combinator, $SD(\chi) = 0$;

ii) if $\chi = (\chi_1\chi_2)$, $SD(\chi) = 1 + \max\{SD(\chi_1), SD(\chi_2)\}$.

A simple relation between the two measures of structural complexity is the following:

Fact 2.1. - $\lceil \log_2 SL(w) \rceil \leq SD(w) \leq SL(w)-1$ for any combinator w.

Computational complexity measures are related to the reduction of combinators to normal form in the weak calculus.

The measures we may define, depend on the computation rule choosen in the reduction process. Among the rules that guarantee to reach the normal form we will choose the standard (leftmost outermost) rule.

Definition 2.3. - *Number of steps of computation* (CT) of a combinator χ is defined by:

i) $CT(\chi) = t$ if χ reaches a normal form in t steps;

ii) $CT(\chi)$ = undefined otherwise.

Definition 2.4. - *Size of computation* (CS) of a combinator χ is defined by:

$CS(\chi) = \ell$ if $\ell = \max_i \{SL(\chi_i)\}$ where χ_i is a formula achieved during the reduction process.

Definition 2.5. - *Depth of computation* (CD) of a combinator χ is defined by:

$CD(\chi) = d$ if $d = \max_i \{SD(\chi_i)\}$ where χ_i is a formula achieved during the reduction process.

Analogously to what is made [5,6,7] for acceptable Gödel numbering of partial recursive functions we will now give the following definitions for measures of complexity in the weak calculus of combinators:

(1) $| \, |$ is a structural complexity measure if:

i) $| \, |$ is a recursive mapping from the set of combinators to the integers;

ii) $\forall n$ the number of combinators w such that $|w| = n$, is finite.

(2) C is a computational complexity measure for a combinator w if C is a partial recursive mapping from the set of combinators to the integers and:

i) w has normal form implies that $C(w)$ is defined;

ii) $C(w)$ is defined implies that "w has normal form" is decidable;

iii) $C(w) = n$ is decidable.

It is not difficult to verify that, if the cardinality of the base is finite, SL and SD are structural complexity measures, while CT, CS and CD are computational complexity measures (*).

As we have already remarked, in general, the way the properties of complexity measures are studied is to consider how limitations on the measures result in limitations on the power of computational systems. On the other hand, for the above listed measures this does not seem to be the way of achieving interesting general results. It is certainly possible to simulate tape and time bounded Turing machines, restricted rewriting systems, primitive recursive computations, etc. in combinatory logic, and to define in this way the corresponding classes of combinators, but this approach does not give classifications of combinators well matched to the computational peculiarities of combinatory logic. These peculiarities are essentially:

i) the ability of "packing" data in such a way that "unpacking" is impossible from outside [9] and that data can be accessed in other than a sequential way;

ii) the "rightward" mechanism of operating of the calculus that makes it more similar to a tag-machine or to a non-erasing Turing machine than to other classical computing systems;

iii) the impossibility in a non typed calculus of an "a priori" distinction between programs and data and, inside a program, between primitives and constructs.

The last point is particularly interesting because the variability of the argument (or at least of the size of the argument) is at the base of the classification of the complexity of programs and functions (typically characterized by asyntotic behaviour).

3. COMPLEXITY MEASURES IN SUBBASES

Taking into account the type of properties of combinatory logic listed at the end of §2, we think that a promising way of studying the com-

(*) The first property of computational complexity measures is satisfied by the norm introduced in [8] in the definition of NURS.

putational complexity of combinators is to use the concept of subbase and to analyze the computational power of various subbases. This somehow corresponds to the limitation of definitions in the formalism of recursive functions, which allows the generation of interesting subsets of partial recursive functions, such as the class of primitive recursive functions, the class of elementary functions and the classes of Grzegorczyk [10].

Definition 3.1. - A *subbase* $\mathcal{B}$ is a non-empty (possibly infinite) class of combinators $\mathcal{B} = \{\Phi_1,\dots,\Phi_n\}$.

In general we will refer to finite subbases of independent combinators.

Definition 3.2. - The *applicative closure* of the subbase $\mathcal{B}$ (denoted $\mathcal{B}^+$) is the class of all finite (applicative) combinations of Φ_i' s. We wish, in the future, to refer to the subclass of $\mathcal{B}^+$ whose elements are in normal form and proper. We will indicate this class by $\mathcal{B}^+_{np}$.

We now summarize a few examples of basic results holding for particular subbases.

Theorem 3.1.-For any combinator w in $\{B,C,K\}^+$ the structural and computational complexity satisfy the following limitations:

i) $CT(w) \leq SL(w) - 1$

ii) $CS(w) = SL(w)$

iii) $CD(w) \leq SL(w) - 1$

iv) $CD(w) \leq \left\lfloor 2^{SD(w)-1} + \frac{SD(w)-1}{2} \right\rfloor$

Proof. (i),(ii) - From a theorem of Curry [1] no combinator exists in {B,C,K} with duplicative effect. So, at every reduction step the length of the formula decreases at least by 1.

(iii). In a formula, whose length is n, appear n-1 applications.

(iv). Since SD(w) is fixed, the maximum number of basic combinators w is $2^{SD(w)}$.

In every reduction step the basic combinators may increase the depth of a formula by at most 1. Therefore, if n is the number of basic combinators used to achieve the maximum depth, we have

$CD(w) = SD(w) + n$, where n is the maximum

integer such that: $SD(w)+n \leq 2^{SD(w)} - n - 1$. (The last inequality is the (iii) written for the deepest formula achieved). Hence point (iv) follows.

Q.E.D.

Theorem 3.2. - In the subbase $\{B\}$ the expotential growth of the depth is achievable. In fact:

i) $\forall w \varepsilon \{B\}^+$ such that $SD(w) \leq 2$ then $CD(w)=SD(w)$;

ii) $\forall n>2 \ \exists \ w_n \varepsilon \{B\}^+$ such that:

$$SD(w_n) = n$$
$$CD(w_n) = N + \sum_{i=0}^{n+1} b_i \quad \text{where } b_o=b_1=0 \text{ and } b_{i+2}=b_{i+1}+b_i+1$$

Proof. The point (i) is immediate.
For the point (ii) it is not difficul to see the class of combinators w_n, whose structure is recursively defined as follows:

$$w_n = d_{n-1}w_{n-1} \qquad \text{where: } w_o = d_o$$
$$d_n = B\ d_{n-2}d_{n-1}$$
$$d_o = B$$
$$d_1 = B\,B\,,$$

has the property that all its B's with 3 arguments, in the reduction process, increase by 1 level the initial depth of the w_n. We also have $w_n = d_{n-1}(d_{n-2}(\ldots(d_od_o)\ldots))$, and $n = SD(w_n)$.

Let b_i be the number of B's with 3 arguments in d_i. We can see $b_o = b_1 = 0$ and $b_i = b_{i-1}+b_{i-2}+1$.

Therefore: $CD(w) = n + \sum_{i=0}^{n-1} b_i$ Q.E.D.

Remark 3.1.-In the subbase $\{B,C\}$ the limit of Theorem 3.2. can be improved.

For example, if $q = B^2(CBB)B\ d_4d_5w_6$,

we have $CD(q) = SD(q) + \sum_{i=0}^{SD(q)-1} b_i+1$.

As far as combinators without normal form are concerned, we may define SL(w)(n) and SD(w)(n) to be the length and the depth of the combinator w at the n-th reduction step and we may show some properties of these functions of n for the subbase {W} and {B,W}. In particular, we will show that, in the subbase {W}, SL(w)(n) is somehow linear, while an exponential growth is possible in the subbase {B,W}.

We will first introduce the following:

Definition 3.3. - The *number* $n_r(w)$ *of right-applied objects of* $w \epsilon \mathcal{B}^+$, is defined by: $n_r(w) = k+1$, where $w=(\ldots((b\chi_1)\chi_2)\ldots\chi_k)$, $b\epsilon\mathcal{B}$ and $\chi_i \epsilon \mathcal{B}^+$, $1 \leq i \leq k$.

Remark 3.2-The decomposition of w in right-applied objects is unique.

Definition 3.4. - The *subwords* of a combinator $w\epsilon\mathcal{B}^+$ are recursively defined as follows:

(i) if w=(ab) where $a,b\epsilon\mathcal{B}$ then a and b are subwords of w;

(ii) if w=(AB) where $A,B\epsilon\mathcal{B}^+$ then A, B and the subwords of A and B are subwords of w.

It is now necessary to prove the following lemma.

Lemma 3.1. - Let w be a combinator of $\{W\}^+$. Let $\bar{w}$ be the leftmost subword of w such that $n_r(\bar{w})=3$. We will call $\bar{w}$ the leading subword of w.

(i) If $\bar{w}$ exists:
- w has not normal form;
- $\forall n$ SL(w)(n) = SL($\bar{w}$)(n)+c, where c is a constant.

(ii) If $\bar{w}$ does not exist, w is in normal form.

Proof. Point (ii) is immediate. Point (i) is proved by the following facts:

(i) In any reduction step of a word $w \epsilon W^+$, we have: $n_r(w)(n) \leq n_r(w)(n+1)$.(*)

(ii) If $w=w_1w_2$ and all W in w_1 have one argument, then in the reduction process the W's in w_1 do not change the number of their arguments.

(iii) If $w=w_1w_2$ where $n_r(w_1) \geq 3$, then the reduction process which follows, does not take into account w_2.

Q.E.D.

(*) We denote by w(n) the combinator derived by w after n reductions. Obviously w(0)=w.

For the base $\{W\}$ we have the following result:

Theorem 3.3. - For any $w \varepsilon \{W\}^+$ without normal form $\exists \bar{n}$ finite such that, if $\Delta(n)=SL(w)(n+1)-SL(w)(n)$, we have:

(i) $\forall n<\bar{n} : \Delta(n) \geq 0$ depending on n;

(ii) $\forall n \geq \bar{n} : \Delta(n)=\bar{\Delta} \geq 0$.

Proof. Given a w $\{W\}^+$ without normal form by lemma 3.1 does exist $\bar{w}$ (de fined as in lemma 3.1) and $SL(w)=SL(\bar{w})+c$.

Moreover the structure of $\bar{w}$ may be one of the following:

(1) $W\,W\,w_o$

(2) $W(Ww_k)w_o$

(3) $W\ \bar{\bar{w}}\ w_o$ where $w_o, w_k, \bar{\bar{w}} \varepsilon \{W\}^+$ and $n_r(\bar{\bar{w}}) \geq 3$.

The complete case analysis of (1), (2) and (3) by applying the induction principle on the combinators' length, proves the theorem [12]. Q.E.D.

For the subbase $\{B,W\}$ we have:

Theorem 3.4. - There exists w in $\{W,B\}^+$, such that SL(w)(n) grows exponentially with n.

Proof. In a constant number of reduction steps (6)
$W(W_{(2)}B_{(1)})(W(W_{(2)}B_{(1)}))p$, where $p \varepsilon \{W,B\}^+$, reaches
$W(W_{(2)}B_{(1)})(W(W_{(2)}B_{(1)}))p'$ where $p' \varepsilon \{W,B\}^+$ and $SL(p')=2 \cdot SL(p)$. Q.E.D.

4. SUBBASES AND DEFINITIONAL COMPLEXITY

A particularly interesting type of results concerning subbases are related to their generative power:

Definition 4.1. - Let $\mathcal{B}$ be a subbase; let V be an infinite ordered set of variables $\{x_1, x_2, \ldots, x_n, \ldots\}$; let V^+ be the set of all finite applicative combinations of variables.

We say that $L(\mathcal{B})$ is the *language generated by* $\mathcal{B}$ if $L(\mathcal{B})$ is the smal lest subset of V^+ satisfying the property that, given any $w \varepsilon \mathcal{B}^+_{np}$, then if n is the order of w, there is $X \varepsilon L(\mathcal{B})$ such that

$$wx_1 \ldots x_n \text{ reduces to } X^{(*)}.$$

We prove first the following lemma.

Lemma 4.1. If ξ is a proper combinator whose order is 2, then $\{\xi\}^+_{np}$ is the set: $\{\xi, \xi\xi, \xi(\xi\xi), \ldots, \xi(\xi(\ldots(\xi\xi)\ldots)), \ldots\}$.

Proof. If we suppose in $\{\xi\}^+_{np}$ a ψ exists with $n>2$ right-applied objects, then ψ is not in normal formal.

Q.E.D.

The following fact can easily be verified:

Fact 4.1.

(i) $L(\{K\}) = \{x_n | n \geq 1\}$;

(ii) $L(\{W\}) = \{x_1x_2x_2\} \cup \{x_1^k | k \geq 3\}$; (**)

(iii) $L(\{B,I\})$ = the language of *complete ordered applications* that is the language generated by the production $S \to (SS)$ and by substitution from left to right of all occurrences of S in a sentential form by the variables $x_1, x_2, \ldots, x_n, \ldots$ in this order;

(iv) $L(\{B\}) = L(\{B,I\}) - \{\bar{X} x_n |$ if $n=1$ $\bar{X}$ is the empty word, if $n>1$ $\bar{X}$ is a word of the language of complete ordered applications with the variables $x_1, \ldots, x_{n-1}\}$;

(v) $L(\{C\}) = \{x_1x_3x_2\} \cup \{x_2x_3x_1\}$.

Proof of Fact 4.1.(i). From lemma 4.1. we know the structure of the elements of $\{K\}^+_{np}$: for each $n \geq 1$ there exists only one combinator $\xi_n \in \{K\}^+_{np}$ such that $SL(\xi_n) = n$, and $\xi_n = K\xi_{n-1}$.
For $\xi_1 = K$ the corresponding X is x_1. If the fact is valid $\forall n' < n$ then

$$\xi_n x_1 \ldots x_{n+1} = K\xi_{n-1} x_1 \ldots x_{n+1} \geq \xi_{n-1} x_2 \ldots x_{n+1} \geq x_n$$

where the last reduction is guaranteed by the induction hypothesis.

Q.E.D.

Proof of Fact 4.1 (ii). Like proof of Fact 4.1. (i). Q.E.D.

Proof of Fact 4.1. (iii) and Fact 4.1. (iv). The language $L(\{B,I\})$ is contained in the language of complete ordered applications [1].

(*) Notice that we consider for example $a_1a_2a_3$ and $((a_1a_2)a_3)$ to be the same word.
(**) Notice that we consider x_i^k and $\underbrace{x_i x_i \ldots x_i}_{k}$ to be the same word.

Viceversa the language of complete ordered applications is contained in $L(\{B,I\})$, because, if X is a word of the language of complete ordered applications, and:

- $X = x_1 x_2 \ldots x_n$, where $n \geq 1$, then $B^{n-1}I$ corresponds to it;
- $X = \bar{X}\ x_j \ldots x_n$ and $\bar{w} \epsilon L(\{B\})$ corresponds to $\bar{X}$, then $B^n I \bar{w}$ corresponds to it.

The last case is the one in which $X = x_1 X_1 X_2 \ldots X_k$ where at least X_k is a combination of 2 or more variables x_i, and possibly k=1. In this case we will now prove inductively that a combinator of $\{B\}^+$ corresponds to X. One parenthesis can be removed for instance from X eliminating the one surrounding X_k by $B_{(k-1)}$.
Let us assume we succeded, in the expansion procedure, to remove p parentheses, obtaining a combination of the form:

$\xi x_1 Y_1 Y_2 \ldots Y_\ell$ where at least one Y_j is a combination of 2 or more variables, and ξ is a combination of B's.

Now we may remove one more parenthesis as the one surrounding Y_j, by the deferred combinator $B_{(j)}$.

Q.E.D.

Proof of Fact 4.1. (v). We prove first that: $\forall \xi \epsilon \{C\}^+_{np}$ the order of ξ is ≤ 3.

In every reduction step of $\xi x_1 x_2 \ldots x_n$ where $\xi \epsilon \{C\}^+_{np}$, there are at least 3 arguments between the first basic combinator in ξ and x_4.

Infact there will always be x_1, x_2 and x_3, because:

- on the first reduction step there are x_1, x_2 and x_3;
- if there are x_1, x_2 and x_3 on the i-th reduction step, then they are also on the (i+1)-th step, because the order of C is 3, C has not compositive effect and no x_i, where $1 \leq i \leq 3$, may be on the left of the leftmost C in a formula (ξ is proper).

Therefore in order to prove the fact 4.1.(v), we can consider only the combinations of x_1, x_2 and x_3. For $x_1x_2x_3$, $x_2x_1x_3$, $x_3x_1x_2$ and $x_3x_2x_1$ after the first expansion step the last variable, that cannot be reused, is not x_3, as it should be.

Instead for $x_1x_3x_2$ and $x_2x_3x_1$ the last variable is x_3, and they are actually computed by C and CC, respectively.

Q.E.D.

For specific subbases the completeness (meta)algorithms (such as those given in [11] for the base {S,K} and in [1] for the base {B,C,W,K}) become more interesting. In fact while in a complete base those metaalgorithms cannot always give the shortest combinator corresponding(*) to a given combination of variables, this may be accompli shed in the case of a subbase which is complete only with respect to a subset of combinations. For the base {B} we have the following results:

Theorem 4.1. - For any X in $L(\{B\})$ we may obtain the shortest w in $\{B\}^+$ corresponding to X.

Sketch of the proof. We will prove that, in the construction of the com binator w that corresponds to X, a sufficient rule in order to obtain the shortest w consists in the elimination of the rightmost parenthesis that can be eliminated at every expansion step by using one B. Infact, at a given expansion step, we have in general an object ξ that can be de rived from S by the following productions:

$$S \to (SS_V) \mid (S_B S) \mid S_B \mid S_V \; ; \qquad S_B \to (S_B S_B) \mid B$$
$$S_V \to (S_V S_V) \mid x_1 \mid \ldots \mid x_n \quad ,$$

in which all variables x_i appear on the left side of x_j, where $j>i$.

We can now eliminate:

(a) 1 or more parentheses surrounding components like $(S_V S_V)$;

(b) 1 parenthesis surrounding a component like $(S_B S)$, where S generates at least one variable;

(c) 1 parenthesis surrounding a component like (SS_V), where S generates at least one B.

Case (a). Let us assume we reached in the expansion step the following object:

$$\xi_1 = \xi' \underset{(1)}{(t_1 t_2)} \ldots \underset{(k)}{(t_k t_{k+1})} x_\ell \ldots x_n \quad , \text{ where every } t_i$$

is either a variable or a combination of the variables $x_p, \ldots, x_{\ell-1}$, and ξ' is a combination of B's and variables $x_1, \ldots, x_{p-1}$.

(*) In the sense of [1] pag. 160.

Let us consider the class S of all the strategies that remove all explicit parentheses in ξ_1 between x_p and x_n and reach an object like: $\xi_2=\xi'' \ x_p\ldots x_{\ell-1}x_{\ell}\ldots x_n$, where ξ'' is a combination of B's and the variables $x_1,\ldots,x_{p-1}$.

It is possible to prove the following assertion:

For every strategy $s\varepsilon S$ that eliminates first the (1) parenthesis, there exists a strategy $\bar{s}\varepsilon S$ that eliminates first the (k) parenthesis taking a not greater number of steps than s.

Since the proof is rather long [12], we give the following example where p=1 and n=6.

STRATEGY s

no.of steps	achieved formula	removed parenthesis	introduced parenthesis (as side effect)
	$x_1 \underset{(1)}{(}x_2 \underset{(2)}{(}x_3x_4)) \underset{(3)}{(}x_5x_6)$		
1		(1)	-
	$Bx_1x_2 \underset{(2)}{(}x_3x_4) \underset{(3)}{(}x_5x_6)$		
2		(3)	$(\alpha),(\beta),(\gamma)$
	$B\underset{(\alpha)}{(}\underset{(\beta)}{(}\underset{(\gamma)}{(}Bx_1)x_2) \underset{(2)}{(}x_3x_4))x_5x_6$		
3		(α)	-
	$BB\underset{(\beta)}{(}\underset{(\gamma)}{(}Bx_1)x_2) \underset{(2)}{(}x_3x_4)x_5x_6$		
4		(2)	(δ)
	$B\underset{(\delta)}{(}BB\underset{(\beta)}{(}\underset{(\gamma)}{(}Bx_1)x_2))x_3x_4x_5x_6$		
5		(δ)	-
	$BB\underset{(\delta)}{(}BB)\underset{(\beta)}{(}\underset{(\gamma)}{(}Bx_1)x_2)x_3x_4x_5x_6$		
6		(β)	-
	$B(BB\underset{(\delta)}{(}BB))\underset{(\gamma)}{(}Bx_1)x_2x_3x_4x_5x_6$		
7		(γ)	-
	$B(B(BB(BB)))Bx_1x_2x_3x_4x_5x_6$		

STRATEGY $\bar{s}$

no.of steps	achieved formula	removed parenthesis	introduced parenthesis (as side effect)
	$x_1 \underset{(1)}{(} x_2 \underset{(2)}{(} x_3x_4)) \underset{(3)}{(} x_5x_6)$		
1		(3)	(ε)
	$B \underset{(\varepsilon)}{(} x_1 \underset{(1)}{(} x_2 \underset{(2)}{(} x_3x_4)))x_5x_6$		
2		(ε)	-
	$BBx_1 \underset{(1)}{(} x_2 \underset{(2)}{(} x_3x_4))x_5x_6$		
3		(1)	(η)
	$B \underset{(\eta)}{(} BBx_1)x_2 \underset{(2)}{(} x_3x_4)x_5x_6$		
4		(2)	(μ),(σ)
	$B \underset{(\mu)}{(} \underset{(\sigma)}{(} B \underset{(\eta)}{(} BBx_1))x_2)x_3x_4x_5x_6$		
5		(μ)	-
	$BB \underset{(\sigma)}{(} B \underset{(\eta)}{(} BBx_1))x_2x_3x_4x_5x_6$		
6		(σ)	-
	$B(BB)B \underset{(\eta)}{(} BBx_1)x_2x_3x_4x_5x_6$		
7		(η)	-
	$B(B(BB)B)(BB)x_1x_2x_3x_4x_5x_6$		

w_s=B(B(BB(BB)))B; $w_{\bar{s}}$=B(B(BB)B)(BB); SL(w_s)=7; SL($w_{\bar{s}}$)=7 .

Case (b). In this case we must compare all strategies that start with the elimination of the parenthesis of (S_BS) with those eliminating first a parenthesis of (S_VS_V), if they exist.

We can prove the same assertion of case (a) in the same way.

Case (c). As case (b).

Therefore, if we have more than 2 parentheses to be eliminated at every expansion step, we can repeat this argument for all of those; it will come out that a sufficient choice for obtaining the shortest w is to eliminate the rightmost parenthesis.

Q.E.D.

As a consequence of theorem 4.1, we may notice that, if $wx_1 \dots x_n \geq X$ and $w'x_1 \dots x_n \geq X'$, where $X, X' \varepsilon L(\{B\})$ and $w, w' \varepsilon \{B\}^+$:

(i) if X' has a lower number of parentheses to be eliminated(*) than X, then $SL(w') < SL(w)$;

(ii) if X' is obtained from X by moving on the left one couple of parentheses of X to be eliminated, then $SL(w') < SL(w)$.

We can also establish the following:

Theorem 4.2. - For any X in $L(\{B\})$ such that $SL(X)=n$ (**) we have that if $w \varepsilon \{B\}^+$ corresponds to X, then $SL(w)=0(n)$.

Proof. The structure of X such that $X \varepsilon L(\{B\})$ and $SL(X)=n$, in which there is the minimum number (1) of parentheses to be eliminated, is of the form:

$$\bar{X}_n = x_1 \dots x_{n-2}(x_{n-1}x_n) .$$

On the other hand, the structure of X such that $X \varepsilon L(\{B\})$ and $SL(X)=n$, in which there is the maximum number of parentheses to be eliminated and these are in the rightmost position, is of the form:

$$\bar{\bar{X}}_n = x_1(x_2(\dots(x_{n-1}x_n)\dots)) .$$

It can be easily verified that: if $\bar{w}_n x_1 \dots x_n \geq \bar{X}_n$, then $\bar{w}_{n+1} = B\bar{w}_n$;

if $\bar{\bar{w}}_n x_1 \dots x_n \geq \bar{\bar{X}}_n$, then $\bar{\bar{w}}_{n+1} = B(B\bar{\bar{w}}_n)B$;

$\bar{w}_3 = \bar{\bar{w}}_3 = B$.

Q.E.D.

(*) We suppose all parentheses to be removed are explicited.

(**) The definitions of structural complexity obviously can be extended to pure combinations .

5. BIBLIOGRAPHY

[1] Curry,H.B., R. Feys, *Combinatory logic*. vol. 1, North Holland(1974)

[2] Ausiello, G., *Computational complexity - Main results and a commentary*, Séminaires IRIA (1972).

[3] Strong,H.R., *Depth-bounded computation*, J.C.S.S., Vol. 4, n. 1 (1970).

[4] Curry, H.B., J.R. Hindley, J.P. Seldin, *Combinatory logic*, vol. 2, North Holland (1972).

[5] Blum, M., *On the size of machines*, Information and Control 2 (1967).

[6] Blum, M., *A machine independent theory of the complexity of recursive functions*, J.A.C.M. 14 (April 1967).

[7] Ausiello, G., *Abstract computational complexity and cycling computations*, J.C.S.S. Vol. 5 (1971).

[8] Barendregt, H., *Normed uniformly reflexive structures*, (in these proceedings).

[9] Böhm, C., W. Gross, *Introduction to the CUCH*, Automata Theory (ed. Caianiello) N.Y. (1966).

[10] Grzegorczyk, A., *Some classes of recursive functions*, Rozprawy Matematyczne (1953).

[11] Rosenbloom, P., *The elements of mathematical logic*, Dover (1950).

[12] Batini, C., A. Pettorossi, *Some properties of subbases in weak combinatory logic*, Rapporto dell'Istituto di Automatica - Università di Roma (1975).