PROPERTIES OF CALCULUS IN R-COMPLEXITY

Rares Folea[1] and Emil Slusanschi[2]

This paper presents a series of general properties of the r-Complexity calculus, a complexity measurement for assessing the performance and asymptotic behaviour of real-world algorithms. This research describes characteristics such as reflexivity, transitivity, or symmetry and discusses several conversion rules between different classes of r-Complexity, as well as establishing fundamental arithmetic principles. The work also examines the behaviour of the addition property within this system and compares its characteristics with those frequently used in the traditional Bachmann-Landau notation. Through utilizing these properties, this research seeks to promote the exploration and development of novel applications for r-Complexity, as well as accelerating the adoption rate of calculus in this refined complexity model.

Keywords: r-Complexity, computational complexity, code complexity, properties of complexity models, asymptotic analysis, algorithmic performance

1. Introduction

Algorithms stand at the core of computer science as they are used as primary building blocks and asymptotic notations are being widely accepted as the main method for estimating its performance, by calculating the complexity of the analysed algorithm [5, 6, 9]. When examining and comparing two or more code snippets, calculating the asymptotic complexity is important for a number of reasons: it helps predicting the scalability of the solution, serves as a comparison method between different algorithms and it can help engineers understand the inherent limitations and trade-offs of each solution. Nonetheless, it can help identify bottlenecks in the code, which can then later be optimized by developers. Algorithmic complexity is commonly expressed utilizing the Bachmann-Landau [2, 3] asymptotic notations, encompassing Big-Theta, Big-O, and Big-Omega measurements.

[1]Department of Computer Science and Engineering, Faculty for Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Romania and Doctoral School of Engineering and Applications of Lasers and Accelerators (S.D.I.A.L.A.), e-mail: rares.folea@stud.acs.upb.ro

[2]Department of Computer Science and Engineering, Faculty for Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Romania, e-mail: emil.slusanschi@cs.pub.ro

Asymptotic code complexity is a topic of interest in various fields, including logic, coding theory, and computational systems. In general, the asymptotic complexity mostly aims to map how the algorithm's utilisation of resources (e.g. total CPU time or memory) grows, with respect to increases in the input size, to later estimate how it will perform with larger inputs. Having this metric, asymptotic notations permits the comparison of multiple algorithms' efficiency, as for example an $\Theta(n \cdot log\ n)$ algorithm is more efficient than an $\Theta(n^2)$ algorithm, for sufficiently large input sizes. Over time, many asymptotic notations [4, 5, 6, 7, 8, 9, 10, 11] have been introduced, but the fundamental concept has remained essentially the same: the definition of an asymptotic notation is based on measuring different complexity functions against a given reference function. Also, there have been efforts to try and automate the process of finding the complexity class for a given program [12, 13].

The r-Complexity model [1] is an alternative asymptotic notation to the Bachmann-Landau notations, that offers better complexity feedback for similar programs and provides subtle insights, even for algorithms that are part of the same conventional complexity class. The model aims to address some of the shortfalls of the traditional model, such as providing a solution for making discrepancy between two similar algorithms with two similar complexity functions, $v, w : \mathbb{N} \longrightarrow \mathbb{R}$, for which that there exists g, such that $v(n) \in \Theta(g(n))$ and $w(n) \in \Theta(g(n))$.

In **Section 2**, we present the motivation of our research and illustrate a scenario where two algorithms solving the same problem would have been grouped into the same complexity class in the traditional analysis, despite their vastly different performance in practice. Then, in **Section 3**, we present some formal definitions and associated corollaries for the various r-Complexity classes, including Big r-Theta, Big r-O and Big r-Omega, as well as presenting alternative criteria for admittance in a given set. **Section 4** explores properties like reflexivity, transitivity, symmetry, and projections within the context of r-Complexity. **Section 5** examines the behaviour of r-Complexity classes under addition and finally, in **Section 6**, we summarize the key results of the research.

2. Motivation

The r-Complexity model demonstrates value by analysing dominant constants, revealing significant differences in some algorithms' behaviour, that can substantially impact the total execution time. To help software engineers collaborate and share knowledge more directly, by discussing and comparing algorithms using the r-Complexity asymptotic notation, this paper presents a set of common properties of calculus within the r-Complexity classes, that can be used to obtain a faster pace of operating with complexities.

In the traditional Bachmann-Landau notations, the Big-Theta definition [4, 5, 6] states that a function $f(n)$ belongs to the $\Theta(g(n))$ complexity set

if and only if two positive constants, denoted by c_1 and c_2, exists, as well as a positive integer n_0, such that $f(n)$ is bounded between $g(n)$, multiplied by c_1 (lower bound) and c_2 (upper bound) constants, for any value of n greater than or equal to n_0:

$$\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2 \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^* \; s.t. \; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n), \; \forall n \geq n_0\}$$

To highlight potential situations where the lack of distinction can mislead developers, consider two algorithms, $Alg1$ and $Alg2$, that solve the same problem are defined by different complexity functions: $f_1, f_2 : \mathbb{N} \longrightarrow \mathbb{R}$, with the properties that the functions defer exactly by one constant: $f_1 = x \cdot f_2$, with $x \in \mathbb{R}_+, \; x > 1$.

These complexities functions will imply that the two algorithms would belong to the same Big-Theta complexity set, because there exists g, with $f_2 \in \Theta(g(n))$, such as:

$$\exists c_1, c_2 \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^* \; s.t. \;\; c_1 \cdot g(n) \leq f_2(n) \leq c_2 \cdot g(n) \;\; , \;\; \forall n \geq n_0$$

holds for $c_1^{'}, c_2^{'} \in \mathbb{R}_+^*, n_0^{'} \in \mathbb{N}^*$, where:

$$\begin{cases} c_1^{'} = x \cdot c_1 \\ c_2^{'} = x \cdot c_2 \\ n_0^{'} = n_0 \end{cases}$$

because $\forall n \geq n_0^{'}$:

$$c_1^{'} \cdot g(n) = x \cdot c_1 \cdot g(n) \leq x \cdot f_2(n) = f_1(n) = x \cdot f_2(n) \leq x \cdot c_2 \cdot g(n) = c_2^{'} \cdot g(n)$$

This implies:

$$c_1^{'} \cdot g(n) \leq f_1(n) \leq c_2^{'} \cdot g(n)$$

and therefore, based to the definition of Big Theta, $f_2 \in \Theta(g(n)) \Rightarrow f_1 \in \Theta(g(n))$.

Similarly, $f_1 \in \Theta(g(n)) \Rightarrow f_2 \in \Theta(g(n))$ for $c_1^{'} = \dfrac{1}{x} \cdot c_1$, $c_2^{'} = \dfrac{1}{x} \cdot c_2$ and $n_0^{'} = n_0$.

Remark that even if $f_1 > f_2$, both complexity functions are part of the same complexity class. This observation implies that two algorithms, whose complexity functions can differ by a constant, even as high as 2024^{2024}, are part of the same complexity class, even if the actual run-time might differ by over five-thousands orders of magnitude. For comparison, only a 30 magnitude order between the two complexity functions $f_1 = 10^{30} \cdot f_2$, signify that if for a given input n, if $Alg2$ ends execution in 1 *attosecond* (10^{-9} part of a nanosecond), then $Alg1$ is expected to end execution in about 3 *millenniums*. Despite of the colossal difference in time, classical complexity model is not perceptive between these subset of algorithms, that have their complexity

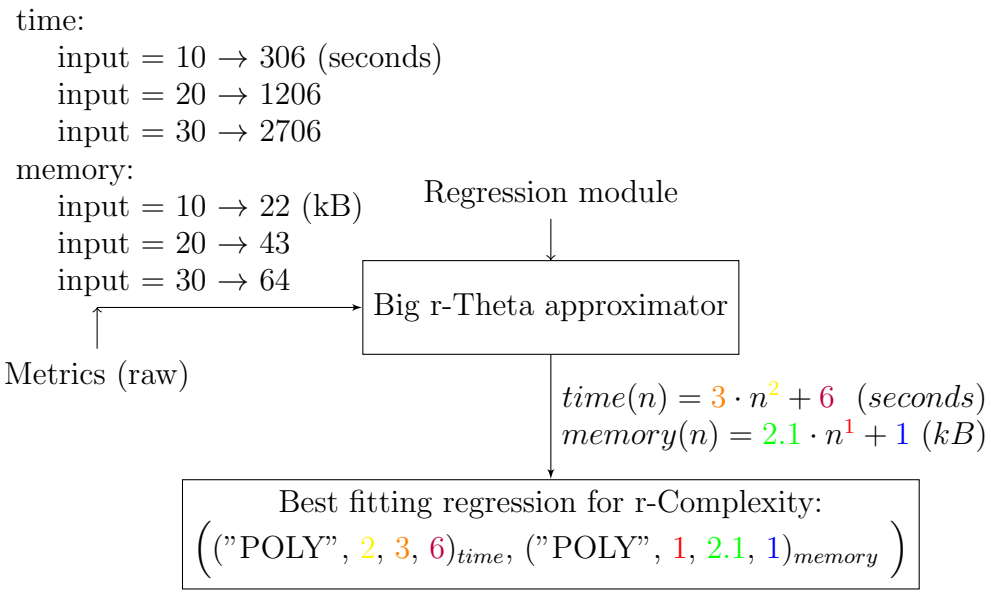

FIGURE 1. An example of the regression fitting process in [14], with a set of two raw metrics: temporarily and space. The goal of the research is to approximate, using simplified functions, the actual Big r-Theta complexity function that models program behaviour across multiple performance metrics, which later can be used to build complexity-based code embeddings. One part of the embedding that analyses the time complexity, POLY 2 3 6, denotes a second degree polynomial, with the coefficient 3 for the squared term of the polynomial and the intercept 6.

functions differing by only some constants. In general, it is easier to evaluate the algorithm's effectiveness and applicability for various scenarios when one is aware of these complexity features.

Some applications of r-Complexity have been presented in [1], which has challenged the common belief that Strassen's algorithm, with its lower asymptotic complexity would always outperform a traditional matrix multiplication for any input size. The result in the paper has shown that for input sizes less than 25 million element, a cache-friendly loop ordering algorithm would finish faster due, despite the increase in complexity. The same paper presents an estimation of the total time using a brute-force algorithm for generating all possible episodes to around 10^{123} seconds, based on the associated r-Complexity function. This has been the core argument for arguing that solving the game of chess perfectly, by brute force algorithms, is currently infeasible, due to the limitations of available computing power. This argument could have not been established by using the traditional complexity models, for finite inputs.

Another area where r-Complexity has proven useful [14] is in building code-embeddings, representative of the program logic, that would later be

used to classify algorithms into different categories, such as greedy, brute force or divide and conquer. The embedding calculus is based on the idea of finding the coefficients of the associated r-Complexity function, estimated by fitting a regression model to the metrics obtained from the profilers during the execution of algorithm, as a function of its input size. For example, a real use-case presented in Figure 1, indicates, via experimental data, that the algorithm's execution time is proportional to the square of the input size and the amount of memory used by the algorithm grows proportionally to the size of the input data. Using r-Complexity models for estimating the Big r-Theta class, we can have insights into what the proportion is, such as insights that the execution time is scaling three times with the square of the input size, or that the memory scales 2.1 times as fast as the input size.

3. **Definition of the r-Complexity sets**

The definition of the r-Complexity sets have initially been introduced in [1]. In this paper, we extend the expressively of these statements by associating corollaries, build upon the knowledge described in the definition, which will highlight important consequences in asymptotic calculus. The validity of the corollaries presented in this section follows directly from the definition of the classes.

Calculus in r-Complexity can be performed either using limits of sequences or limits of functions. Consider any two complexity functions $f, g : \mathbb{N}_+ \longrightarrow \mathbb{R}$. The following notations and terminology will be used to describe the asymptotic behaviour of an algorithm's complexity, as characterized by a function $f : \mathbb{N} \longrightarrow \mathbb{R}$. We define the set of all complexity calculus $\mathcal{F} = \{f : \mathbb{N} \longrightarrow \mathbb{R}\}$.

Definition 3.1. *Big r-Theta has been defined as the set that represents the group of mathematical functions that have the same growth rate as a function g, in the context of asymptotic analysis. More formally, the Big r-Theta r-Complexity class has been defined* [1] *as:*

$$\Theta_r(g(n)) = \{f \in \mathcal{F} \mid \forall c_1, c_2 \in \mathbb{R}_+^* \ s.t. c_1 < r < c_2, \exists n_0 \in \mathbb{N}^* \\ s.t. \ \ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \ , \ \forall n \geq n_0\}$$

Corollary 3.1. *Criteria for a function f to be in the Big r-Theta complexity class defined by a function g is a direct result of the following:*

$$f \in \Theta_r(g(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = r$$

We will use the same definition for the Big r-O, Big r-Omega, Small r-O and Small r-Omega as presented in [1], with the remark that the last two models are defined for completeness reasons only, as they don't provide any novelty opposed to the traditional notations [16, 17]. Similar to Big r-Theta, following the definitions of Big r-O, Big r-Omega, Small r-O and Small r-Omega classes, the following corollaries are direct implications:

Corollary 3.2. *Criteria for admittance of f in Big r-O class defined by g:*

$$f \in \mathcal{O}_r(g(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = l,\ l \in [0, r]$$

Corollary 3.3. *Criteria for admittance of f in Big r-Omega class defined by g:*

$$f \in \Omega_r(g(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = l,\ l \in [r, \infty)$$

The criteria for admittance of f in both Small r-O and Small r-Omega complexity classes are unchanged from the traditional notations.

Corollary 3.4. *Criteria for admittance of f in Small r-O class defined by g:*

$$f \in o_r(g(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Corollary 3.5. *Criteria for admittance of f in Small r-Omega class defined by g:*

$$f \in \omega_r(g(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

4. Common properties in r-Complexity

This chapter will present new implications in terms of *reflexivity, transitivity, symmetry and projections* in r-Complexity, and it will draw the comparison with equivalent properties within the conventional Bachmann–Landau notations, already well analysed and established in the literature [9, 10, 11, 15]. By recognizing reflexivity, symmetry and transpose properties in r-Complexity, we consider that some problems can be solved more efficiently in this model. To establish the results in this section, we will rely on the definitions introduced in Section 3, and the foundational model described in [1].

Theorem 4.1. *Reflexivity in r-Complexity for Big r-Theta notation*

$$f \in \Theta_r\left(\frac{1}{r} \cdot f(n)\right)\ \forall r \neq 0$$

Proof. Based on the definition of $\Theta_r(f(n))$

$$\Theta_r(f(n)) = \{f' \in \mathcal{F} \mid \forall c_1, c_2 \in \mathbb{R}_+^*\ s.t. c_1 < r < c_2, \exists n_0 \in \mathbb{N}^* \\ s.t.\ \ c_1 \cdot f(n) \leq f'(n) \leq c_2 \cdot f(n)\ ,\ \forall n \geq n_0\}$$

Using substitution $f(n) \longleftarrow \frac{1}{r} \cdot f(n)$

$$\Theta_r\left(\frac{1}{r} \cdot f(n)\right) = \{f' \in \mathcal{F} \mid \forall c_1, c_2 \in \mathbb{R}_+^*\ s.t. c_1 < r < c_2, \exists n_0 \in \mathbb{N}^* \\ s.t.\ \frac{1}{r} \cdot\ c_1 \cdot f(n) \leq f'(n) \leq \frac{1}{r} \cdot c_2 \cdot f(n)\ ,\ \forall n \geq n_0\}$$

By choosing $c_1' = \frac{1}{r} \cdot c_1, c_2' = \frac{1}{r} \cdot c_2$

$$\Theta_r\left(\frac{1}{r} \cdot f(n)\right) = \{f' \in \mathcal{F} \mid \forall c_1', c_2' \in \mathbb{R}_+^* \ s.t. c_1' < 1 < c_2', \exists n_0 \in \mathbb{N}^* \\ s.t. \ \ c_1' \cdot f(n) \leq f'(n) \leq c_2' \cdot f(n) \ , \ \forall n \geq n_0\}$$

For any $x \in \mathbb{R}^*$, $\forall c_1, c_2 \ \ s.t. c_1 \leq 1 \leq c_2$, we have $x \cdot c_1 \leq x \leq x \cdot c_2$. Thus, if $f : \mathbb{N} \longrightarrow \mathbb{R}$, $\forall c_1, c_2 \ \ s.t. c_1 \leq 1 \leq c_2$, we have $f(n) \cdot c_1 \leq f(n) \leq f(n) \cdot c_2 \ \ \forall n \in \mathbb{N}^*$ or $f(n) \cdot c_1 \geq f(n) \geq f(n) \cdot c_2 \ \ \forall n \in \mathbb{N}^*$

Therefore:

$$\forall c_1', c_2' \in \mathbb{R}_+^* \ s.t. c_1' < 1 < c_2', \exists n_0 = 1 s.t. \ c_1' \cdot f(n) \leq f(n) \leq c_2' \cdot f(n) \ , \ \forall n \geq n_0 = 1 \Rightarrow$$

$$f \in \Theta_r\left(\frac{1}{r} \cdot f(n)\right) \ \ \forall r \neq 0$$

■

Theorem 4.2. *Reflexivity in r-Complexity for Big r-O notation*

$$f \in \mathcal{O}_r\left(x \cdot f(n)\right) \ \ \forall x \geq \frac{1}{r}$$

Proof. Based on the definition of $\mathcal{O}_r(f(n))$

$$\mathcal{O}_r(f(n)) = \{f' \in \mathcal{F} \mid \forall c \in \mathbb{R}_+^* \ s.t. \ r < c, \exists n_0 \in \mathbb{N}^* \ s.t. \ f'(n) \leq c \cdot f(n), \ \forall n \geq n_0\}$$

Using substitution $f(n) \longleftarrow x \cdot f(n), \forall x \geq \frac{1}{r}, \forall r \neq 0$

$$\mathcal{O}_r(x \cdot f(n)) = \{f' \in \mathcal{F} \mid \forall c \in \mathbb{R}_+^* \ s.t. \ r < c, \exists n_0 \in \mathbb{N}^* \ s.t. \ f'(n) \leq c \cdot x \cdot f(n), \ \forall n \geq n_0\}$$

If $r < c$ and $x \geq \frac{1}{r}$, then $c \cdot x \geq 1, \forall r, c, x \ \ r \neq 0$ Thus, if $f : \mathbb{N} \longrightarrow \mathbb{R}$, we have $f(n) \leq 1 \cdot f(n) \leq c \cdot x \cdot f(n) \ \ \forall n \in \mathbb{N}^*$.

Therefore:

$$\forall x \geq \frac{1}{r}, \ \forall c \in \mathbb{R}_+^* \ s.t. \ r < c, \exists n_0 = 1 s.t. \ f(n) \leq c \cdot x \cdot f(n) \ \forall n \in \mathbb{N}^*, \ \forall n \geq n_0 = 1 \Rightarrow$$

$$f \in \mathcal{O}_r\left(x \cdot f(n)\right) \ \ \forall x \geq \frac{1}{r}$$

■

Theorem 4.3. *Reflexivity in r-Complexity for Big r-Omega notation*

$$f \in \Omega_r\left(x \cdot f(n)\right) \ \ \forall x \leq \frac{1}{r}$$

Proof. Based on the definition of $\Omega_r(f(n))$

$$\Omega_r(f(n)) = \{f' \in \mathcal{F} \mid \forall c \in \mathbb{R}_+^* \ s.t. \ c < r, \exists n_0 \in \mathbb{N}^* \ s.t. \ f'(n) \geq c \cdot f(n), \ \forall n \geq n_0\}$$

Using substitution $f(n) \longleftarrow x \cdot f(n), \forall x \leq \frac{1}{r}, \forall r \neq 0$

$$\Omega_r(x \cdot f(n)) = \{f' \in \mathcal{F} \mid \forall c \in \mathbb{R}_+^* \ s.t. \ c < r, \exists n_0 \in \mathbb{N}^* \ s.t. \ f'(n) \geq x \cdot c \cdot f(n), \ \forall n \geq n_0\}$$

If $r > c$ and $x \leq \frac{1}{r}$, then $c \cdot x \leq 1, \forall r, c, x \;\; r \neq 0$

Therefore:

$\forall x \leq \frac{1}{r}$, $\forall c \in \mathbb{R}_+^* \; s.t. \; c < r, \exists n_0 = 1 s.t. \;\; f(n) \geq c \cdot x \cdot f(n) \;\; \forall n \in \mathbb{N}^*, \; \forall n \geq n_0 = 1 \Rightarrow$

$$f \in \Omega_r \left(x \cdot f(n) \right) \;\; \forall x \leq \frac{1}{r}$$

■

Remark 4.1. *Reflexivity does not hold in r-Complexity for small r-O and small r-Omega notation, as this set is the same as the classical sets defined in Bachmann–Landau notations, and reflexivity does not hold for Small Omega class.*

$$f \notin o_r(f(n))$$
$$f \notin \omega_r(f(n))$$

Theorem 4.4. *Transitivity in r-Complexity - Big r-Theta notation:*

$$f \in \Theta_r(g(n)), g \in \Theta_{r'}(h(n)) \Rightarrow f \in \Theta_{r \cdot r'}(h(n))$$

Proof. $f \in \Theta_r(g(n)) \Rightarrow \forall c_1, c_2 \in \mathbb{R}_+^* \; s.t. c_1 < r < c_2, \exists n_0 \in \mathbb{N}^*$
$s.t. \;\; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \;, \;\; \forall n \geq n_0$

$g \in \Theta_{r'}(h(n)) \Rightarrow \forall c_1, c_2 \in \mathbb{R}_+^* \; s.t. c_1 < r' < c_2, \exists n_0' \in \mathbb{N}^*$
$s.t. \;\; c_1 \cdot h(n) \leq g(n) \leq c_2 \cdot h(n) \;, \;\; \forall n \geq n_0'$

Therefore:

$\forall c_1, c_2, c_1', c_2' \in \mathbb{R}_+^* \; s.t. c_1 < r < c_2, c_1' < r' < c_2', \exists n_0'' = max(n_0, n_0') \in \mathbb{N}^* \; s.t.$

$c_1' \cdot c_1 \cdot h(n) \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \leq c_2' \cdot c_2 \cdot h(n) \;, \;\; \forall n \geq n_0''$

For $c_1'' = c_1 \cdot c_1', c_2'' = c_2 \cdot c_2'$, we have: $c_1'' < r \cdot r' < c_2''$.

As a consequence:

$$\forall c_1'', c_2'' \in \mathbb{R}_+^* \; s.t. c_1'' < r \cdot r' < c_2'' \;\; \exists n_0'' = max(n_0, n_0') \in \mathbb{N}^* s.t.$$
$$c_1'' \cdot h(n) \leq f(n) \leq c_2'' \cdot h(n) \;, \;\; \forall n \geq n_0'' \Rightarrow f \in \Theta_{r \cdot r'}(h(n))$$

■

The remaining transitivity properties also hold within r-Complexity Calculus, and their proofs follow a similar structure to the one presented above:

Theorem 4.5. *Transitivity in r-Complexity - Big r-O notation:*
$f \in \mathcal{O}_r(g(n)), g \in \mathcal{O}_r(h(n)) \Rightarrow f \in \mathcal{O}_{r \cdot r'}(h(n))$

Theorem 4.6. *Transitivity in r-Complexity - Big r-Omega notation:*
$f \in \Omega_r(g(n)), g \in \Omega_r(h(n)) \Rightarrow f \in \Omega_{r \cdot r'}(h(n))$

Theorem 4.7. *Transitivity in r-Complexity - Small r-O notation:*
$f \in o_r(g(n)), g \in o_{r'}(h(n)) \Rightarrow f \in o_{r \cdot r'}(h(n))$

Theorem 4.8. *Transitivity in r-Complexity - Small r-Omega notation*
$f \in \omega_r(g(n)), g \in \omega_{r'}(h(n)) \Rightarrow f \in \omega_{r \cdot r'}(h(n))$

Symmetry is a fundamental algebraic property that plays a crucial role in various mathematical contexts, as it simplifies problem solving in many cases, and it also provides means to perform a more elegant flow of calculations.

Theorem 4.9. *Symmetry in r-Complexity:* $f \in \Theta_r(g(n)) \Rightarrow g \in \Theta_{\frac{1}{r}}(f(n))$

Proof. $f \in \Theta_r(g(n)) \Rightarrow \forall c_1, c_2 \in \mathbb{R}_+^* \ s.t. c_1 < r < c_2, \exists n_0 \in \mathbb{N}^*$
$s.t. \ \ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \ , \ \forall n \geq n_0$
Using the substitution $c_1' = \frac{c_1}{r}, c_2' = \frac{c_2}{r} \Rightarrow \forall c_1', c_2' \in \mathbb{R}_+^* \ s.t. c_1 < 1 < c_2, \exists n_0 \in \mathbb{N}^*$
$s.t. \ \ c_1' \cdot g(n) \leq \frac{1}{r} \cdot f(n) \leq c_2' \cdot g(n) \ , \ \forall n \geq n_0$
The previous inequality can be re-written as:

$$\begin{cases} g(n) \leq \frac{1}{c_1'} \cdot \frac{1}{r} \cdot f(n) \\ g(n) \geq \frac{1}{c_2'} \cdot \frac{1}{r} \cdot f(n) \end{cases}$$

Using notation $c_1'' = \frac{1}{c_2'}, c_2'' = \frac{1}{c_1'}$ the inequality becomes:
$\forall c_1,'' c_2'' \in \mathbb{R}_+^* \ s.t. c_1'' < r < c_2'', \exists n_0 \in \mathbb{N}^*$

$$c_1'' \cdot \frac{1}{r} \cdot f(n) \leq g(n) \leq c_2' \cdot \frac{1}{r} \cdot f(n) \ \ \forall n \geq n_0$$

Thus, Based on the definition of $\Theta_r(f(n))$, $f \in \Theta_r(g(n)) \Rightarrow g \in \Theta_{\frac{1}{r}}(f(n))$. ■

Theorem 4.10. *Transpose symmetry in r-Complexity:* $f \in \mathcal{O}_r(g(n)) \Leftrightarrow g \in \Omega_{\frac{1}{r}}(f(n))$

Proof. Using the definition of $\mathcal{O}_r(f(n))$ and $f \in \mathcal{O}_r(g(n))$ ($\mathcal{O}_r(g(n)) \supseteq \{f\}$):

$$\forall c \in \mathbb{R}_+^* \ s.t. \ r < c, \exists n_0 \in \mathbb{N}^* \ s.t. \ g(n) \leq c \cdot f(n), \ \forall n \geq n_0$$

We can rewrite the previous relation as following:

$$\forall c \in \mathbb{R}_+^* \ s.t. \ r < c, \exists n_0 \in \mathbb{N}^* \ s.t. \ \frac{1}{c} \cdot g(n) \leq f(n), \ \forall n \geq n_0$$

Substituting the constant $c' = \frac{1}{c}$:

$$\forall c' \in \mathbb{R}_+^* \ s.t. \ c' < \frac{1}{r}, \exists n_0 \in \mathbb{N}^* \ s.t. \ f(n) \geq c' \cdot g(n), \ \forall n \geq n_0$$

Therefore:

$$\Omega_{\frac{1}{r}}(f(n)) \supseteq \{g\}$$

■

Remark 4.2. *The Transpose symmetry hold for Small r-o and Small r-Omega, as these two sets are equal with the classical sets defined in Bachmann–Landau notations.*

Theorem 4.11. *Projection property in r-Complexity:*
$f \in \Theta_r(g(n)) \Leftrightarrow f \in \mathcal{O}_r(g(n))$ *and* $f \in \Omega_r(g(n))$

Proof. The proof is straightforward, using the definitions of Big r-Theta, Big r-O and Big r-Omega.
" $\Rightarrow$ " $f \in \Theta_r(g(n)) \Rightarrow f \in \mathcal{O}_r(g(n)), f \in \Omega_r(g(n))$ Based on the definition of $\Theta_r(g(n))$, $f \in \Theta_r(g(n)) \Rightarrow$

$$\forall c_1, c_2 \in \mathbb{R}_+^* \; s.t. c_1 < r < c_2, \exists n_0 \in \mathbb{N}^* s.t. \;\; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \;,\; \forall n \geq n_0$$

By splitting the inequality:

$$\begin{cases} \forall c_1 \in \mathbb{R}_+^* \; s.t. c_1 < r, \exists n_0 \in \mathbb{N}^* \; s.t. \;\; c_1 \cdot g(n) \leq f(n), \; \forall n \geq n_0 \\ \forall c_2 \in \mathbb{R}_+^* \; s.t. r < c_2, \exists n_0 \in \mathbb{N}^* \; s.t. \; f(n) \leq c_2 \cdot g(n) \;,\; \forall n \geq n_0 \end{cases}$$

Therefore:

$$\begin{cases} f \in \mathcal{O}_r(g(n)) \\ f \in \Omega_r(g(n)) \end{cases}$$

" $\Leftarrow$ " $f \in \mathcal{O}_r(g(n)), f \in \Omega_r(g(n)) \Rightarrow f \in \Theta_r(g(n))$ Using the definitions of Big r-O and Big r-Omega:

$$\begin{cases} \forall c_1 \in \mathbb{R}_+^* \; s.t. c_1 < r, \exists n_0 \in \mathbb{N}^* \; s.t. \;\; c_1 \cdot g(n) \leq f(n), \; \forall n \geq n_0 \\ \forall c_2 \in \mathbb{R}_+^* \; s.t. r < c_2, \exists n_0' \in \mathbb{N}^* \; s.t. \; f(n) \leq c_2 \cdot g(n) \;,\; \forall n \geq n_0' \end{cases}$$

By choosing $n_0'' = max(n_0, n_0')$:

$$\begin{cases} \forall c_1 \in \mathbb{R}_+^* \; s.t. c_1 < r, \Rightarrow \;\; c_1 \cdot g(n) \leq f(n), \; \forall n \geq n_0'' \\ \forall c_2 \in \mathbb{R}_+^* \; s.t. r < c_2, \Rightarrow \; f(n) \leq c_2 \cdot g(n) \;,\; \forall n \geq n_0'' \end{cases}$$

By merging the inequalities, we have for $\forall n \geq n_0''$:

$$\forall c_1, c_2 \in \mathbb{R}_+^* \; s.t. c_1 < r < c_2, \exists n_0'' = max(n_0, n_0') \; s.t. \;\; c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

Therefore $f \in \Theta_r(g(n))$. ∎

5. Addition properties

Analogous to the calculus in Bachmann-Landau notations (including Big-O arithmetic), valuable insights can be gained by examining the behaviour of r-Complexity classes under addition. For each of the presented theorem, we present a corollary, using a *relax notation*, where consider that by any r-Complexity class notation, we denote an arbitrary function part of the set. This would allow us to write equations such as the one presented below.

For analysing the addition properties in Big r-Theta, the following relations hold for any correctly defined functions $f, g, f', g', h : \mathbb{N} \longrightarrow \mathbb{R}$, where

$$h(n) = f'(n) + g'(n)$$

$\forall n \in \mathbb{N}$, where f', g' are two arbitrary functions such that $f' \in \Theta_r(f), g' \in \Theta_q(g)$ and $r, q \in \mathbb{R}_+$:

Theorem 5.1. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow h \in \Theta_q(g)$.

Proof. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow \lim_{n\to\infty} \frac{f'(n)}{g'(n)} = 0 \Rightarrow \lim_{n\to\infty} \frac{g'(n) + f'(n)}{g'(n)} = 1$ and using the result from asymptotic analysis section, we have $g'(n) + f'(n) = h(n) \in \Theta_1(g')$.
Using reflexivity property, $h(n) \in \Theta_q(g)$. ∎

Corollary 5.1. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, *the following relation holds:*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow \Theta_r(f) + \Theta_q(g) = \Theta_q(g)$$

Theorem 5.2. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow h \in \Theta_r(f)$.

Proof. Using the last result, by performing the swap:$f \leftarrow g, g \leftarrow f$ and considering the commutativity of addition, we obtain the proof for this statement. ∎

Corollary 5.2. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, *the following relation holds:*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow \Theta_r(f) + \Theta_q(g) = \Theta_r(f)$$

Theorem 5.3. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = t,\ t \in \mathbb{R}_+ \Rightarrow h \in \Theta_r\left(f + \frac{r}{q} \cdot g\right)$.

Proof. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = t \Rightarrow \lim_{n\to\infty} \frac{f'(n)}{g'(n)} = t \cdot \frac{r}{q} = t' \Rightarrow \lim_{n\to\infty} \frac{g'(n) + f'(n)}{g'(n)} = t' + 1$ and using the result from asymptotic analysis section, we have $g'(n) + f'(n) = h(n) \in \Theta_{t'+1}(g')$.
Using reflexivity property, $h(n) \in \Theta_{t'+1}(q \cdot g)$.
Using the conversion technique, we have $h(n) \in \Theta_r(\frac{1}{r} \cdot t \cdot \frac{r}{q} + 1) \cdot q \cdot g)$.
By swapping back t, asymptotically, we can establish:

$$h(n) \in \Theta_r\left(\frac{1}{r} \cdot (\frac{f(n)}{g(n)} \cdot \frac{r}{q} + 1) \cdot q \cdot g\right)$$

Therefore $h \in \Theta_r\left(f + \frac{r}{q} \cdot g\right)$. ∎

Corollary 5.3. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = t,\ t \in \mathbb{R}_+$, *the following relation holds:*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = t,\ t \in \mathbb{R}_+ \Rightarrow \Theta_r(f) + \Theta_q(g) = \Theta_r(f + \frac{r}{q} \cdot g)$$

For addition in Big r-O, the following relations hold for any correctly defined functions $f, g, f', g', h : \mathbb{N} \longrightarrow \mathbb{R}$, where $h(n) = f'(n) + g'(n)\ \forall n \in \mathbb{N}$, where f', g' are two arbitrary functions such that $f' \in \mathcal{O}_r(f), g' \in \mathcal{O}_q(g)$ and $r, q \in \mathbb{R}_+$:

Theorem 5.4. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow h \in \mathcal{O}_q(g)$.

Corollary 5.4.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow \mathcal{O}_r(f) + \mathcal{O}_q(g) = \mathcal{O}_q(g)$$

Theorem 5.5.

Corollary 5.5.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow \mathcal{O}_r(f) + \mathcal{O}_q(g) = \mathcal{O}_r(f)$$

Theorem 5.6. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = t,\ t \in \mathbb{R}_+ \Rightarrow h \in \mathcal{O}_r\left(f + \frac{r}{q} \cdot g\right)$.

Corollary 5.6.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = t,\ t \in \mathbb{R}_+ \Rightarrow \mathcal{O}_r(f) + \mathcal{O}_q(g) = \mathcal{O}_r(f + \frac{r}{q} \cdot g)$$

The proofs for the latest results is similar with the proof for addition in Big Theta, by using conversion presented for Big r-O class.

For addition in Big r-Omega, the following relations hold for any correctly defined functions $f, g, f', g', h : \mathbb{N} \longrightarrow \mathbb{R}$, where $h(n) = f'(n) + g'(n)\ \forall n \in \mathbb{N}$, where f', g' are two arbitrary functions such that $f' \in \Omega_r(f), g' \in \Omega_q(g)$ and $r, q \in \mathbb{R}_+$:

Theorem 5.7. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow h \in \Omega_q(g)$.

Corollary 5.7.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow \Omega_r(f) + \Omega_q(g) = \Omega_q(g)$$

Theorem 5.8. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow h \in \Omega_r(f)$.

Corollary 5.8.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow \Omega_r(f) + \Omega_q(g) = \Omega_r(f)$$

Theorem 5.9. *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = t,\ t \in \mathbb{R}_+ \Rightarrow h \in \Omega_r\left(f + \frac{r}{q} \cdot g\right)$.

Corollary 5.9.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = t,\ t \in \mathbb{R}_+ \Rightarrow \Omega_r(f) + \Omega_q(g) = \Omega_r(f + \frac{r}{q} \cdot g)$$

The proofs for the latest results is similar with the proof for addition in Big Theta, using conversion presented for Big r-Omega class.

The additions in Small r-O and Small r-Omega have the same properties as described in Bachmann-Landau notations.

6. Conclusions

The r-Complexity [1] model proposes a different approach to measuring the algorithm complexity, that seeks to provide a more refined awareness of algorithm performance, by not only performing asymptotic analysis, but also providing relevant estimations for finite inputs.

Compared to traditional Bachmann-Landau notations, r-Complexity can distinguish between algorithms that would otherwise fall under the same complexity class, providing more granular insights for developers in deciding the optimal solution to use. Moreover, r-Complexity classes have a flexible nature, as functions defining classes can be easily converted, given different values of r, via the simple technique of changeover developed in [1]. For this reasons, the r-Complexity framework has the potential to enhance the standard ways engineers and developers assess and optimize their algorithms. The presented applications in Section 2 demonstrated that r-Complexity provides a valuable framework for analysing algorithm performance beyond asymptotic complexity, enabling more accurate predictions(as observed on matrix multiplication) and deeper understanding of program behaviour (as exemplified by their applicability on building code embeddings). One of the key insights provided by this complexity model is that Strassen's algorithm, despite its theoretical asymptotic advantage, is slower than a traditional matrix multiplication, for practical input sizes.

The main contributions of this paper consists in enumerating and proofing some of the properties of calculus within r-Complexity classes, such as reflexivity, transitivity, symmetry, and projections. Additionally, the study investigated the behaviour of r-Complexity classes under addition and conversion rules between r-Complexity classes and the more traditional Bachmann-Landau complexity classes. These findings aim to help users of r-Complexity

to speed up their calculus, and potentially support the wider adoption of this method of performing algorithm analysis and trade-off comparisons.

REFERENCES

[1] *Folea, Rares and Slusanschi, Emil-Ioan*, A new metric for evaluating the performance and complexity of computer programs: A new approach to the traditional ways of measuring the complexity of algorithms and estimating running times (2021), 2021 23rd International Conference on Control Systems and Computer Science (CSCS), 157–164.

[2] *Bachmann, Paul*, Analytische Zahlentheorie [Analytic Number Theory] (1894), Vol. 2. Leipzig: Teubner p. 402-403.

[3] *Landau, Edmund*, Handbuch der Lehre von der Verteilung der Primzahlen [Handbook on the theory of the distribution of the primes] (1909), Leipzig: B. G. Teubner. p. 883.

[4] *Knuth, Donald E.* "Big omicron and big omega and big theta." ACM Sigact News 8.2 (1976): 18-24.

[5] *Giumale, C. A.* Introduction to the Analysis of Algorithms: Theory and Application. (2004) Polirom, Bucharest, Romania.

[6] *Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.*. Introduction to algorithms (2022). MIT press.

[7] *Wilf, Herbert S.* Algorithms and complexity. (2002) AK Peters/CRC Press.

[8] *Bovet, Daniel Pierre, Pierluigi Crescenzi, and D. Bovet.* Introduction to the Theory of Complexity. (1994) Vol. 7. London: Prentice Hall.

[9] *Mogoş, Andrei-Horia, Bianca Mogoş, and Adina Magda Florea.*"A new asymptotic notation: Weak Theta (2015). Mathematical Problems in Engineering.

[10] *Affeldt, Reynald, Cyril Cohen, and Damien Rouhling.* "Formalization techniques for asymptotic reasoning in classical analysis." Journal of Formalized Reasoning (2018).

[11] *Mala, Firdous Ahmad, and Rouf Ali.* "The Big-O of Mathematics and Computer Science." Appl. Math. Comput 6.1 (2022): 1-3.

[12] *Phalke, Swapnil, Yogita Vaidya, and Shilpa Metkar.* ""Big-O time complexity analysis of algorithm." 2022 international conference on signal and information processing (IConSIP). IEEE, 2022.

[13] *Vaz, Rayner, et al..* "Automated big-o analysis of algorithms." 2017 international conference on nascent technologies in engineering (ICNTE). IEEE, 2017.

[14] *Folea, R., Iacob, R., Slusanschi, E., Rebedea, T.*, Complexity-Based Code Embeddings (2023). In: Nguyen, N.T., et al. Computational Collective Intelligence. ICCCI 2023. Lecture Notes in Computer Science(), vol 14162. Springer, Cham.

[15] *Goloveshkin, V. A., and M. V. Uljanov.* "A measure of functions' asymptotic growth and the complexity classification of computer algorithms." Journal of Emerging Trends in Computing and Information Sciences 3.11 (2012).

[16] *Papadimitriou, Christos H..* "Computational complexity." In Encyclopedia of computer science, pp. 260-265. 2003.

[17] *Wegener, Ingo.* Complexity theory: exploring the limits of efficient algorithms. Springer Science & Business Media, 2005.