Theoretical Computer Science 528 (2014) 85–100

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# A Hitchhiker's Guide to descriptional complexity through analytic combinatorics[☆]

Sabine Broda, António Machiavelo, Nelma Moreira[*], Rogério Reis[*]

CMUP, Faculdade de Ciências da Universidade do Porto, Rua do Campo Alegre, 4169-007 Porto, Portugal

ARTICLE INFO

*Article history:*
Received 23 August 2012
Received in revised form 12 December 2013
Accepted 10 February 2014
Communicated by M. Ito

*Keywords:*
Descriptional complexity
Average-case complexity
Analytic combinatorics
Regular expressions
Finite automata

ABSTRACT

Nowadays, increasing attention is being given to the study of the descriptional complexity in the average case. Although the underlying theory for such a study seems intimidating, one can obtain interesting results in this area without too much effort. In this gentle introduction we take the reader on a journey through the basic analytical tools of that theory, giving some illustrative examples using regular expressions. Additionally, new asymptotic average-case results for several $\varepsilon$-NFA constructions are presented, in a unified framework. It turns out that, asymptotically, and in the average case, the complexity gap between the several constructions is significantly larger than in the worst case. Furthermore, one of the $\varepsilon$-NFA constructions approaches the corresponding $\varepsilon$-free NFA construction, asymptotically and on average.

## 1. Introduction

Descriptional complexity studies the measures of complexity of languages and operations. Usually, the descriptional complexity of an object is the size of its shortest description, which can then be considered in the study of the worst or the average size of a set of objects, or of the result of an operation on those objects. For instance, when converting a regular expression to a non-deterministic finite automaton (NFA), the size of the resulting automaton can be measure in terms of the number of its states, or using its number of transitions. Depending on the chosen method used to do this conversion, the resulting size can vary dramatically and it is of interest to have estimations of both, worst and average cases.

For each measure, it is important to know the size of the smallest representation for a given language, as well as how the size varies when several such representations are combined or transformed. These studies are motivated by the need to have good estimates of the amount of resources required to manipulate those representations. This is crucial in new applied areas where automata and other models of computation are used, for instance, for pattern matching in bio-informatics or network security, or for model checking or security certificates in formal verification systems. In general, having succinct objects will improve our control of software, which may become shorter, more efficient and easier to certify. Recently, the descriptional complexity of formal languages has been extensively researched; see [15,24,21,34,23,9,22,35,8].

Most studies of descriptional complexity as well as of computational complexity (or analysis of algorithms) consider worst-case analyses, for which well-established methods are known [1,26–28]. However, a worst-case behavior seldom occurs and a worst-case upper bound can be of little use in practical applications. Think about the time a given trip can

[☆] This work was partially funded by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT under projects PEst-C/MAT/UI0144/2011 and CANTE-PTDC/EIA-CCO/101904/2008.

[*] Corresponding author.
*E-mail addresses:* sbb@dcc.fc.up.pt (S. Broda), ajmachia@fc.up.pt (A. Machiavelo), nam@dcc.fc.up.pt (N. Moreira), rvr@dcc.fc.up.pt (R. Reis).

http://dx.doi.org/10.1016/j.tcs.2014.02.013
0304-3975/© 2014 Elsevier B.V. All rights reserved.

take if there is the chance of death. Furthermore, the best performing algorithms are not necessarily the ones with the best worst-case complexity. A classical example is the Quicksort algorithm. These facts motivate the study of complexity analysis in the average-case, where input data is assumed to follow a given probability distribution.

Sedgewick and Flajolet [31] is a standard reference for the analysis of algorithms from the average-case point of view. One of the most important concepts used in this context is the notion of generating function. Given a set of objects $\mathsf{C}$ (*combinatorial class*) on which a non-negative integer function (size) $|\cdot|$ is defined, and such that for each $n \geqslant 0$, the number of objects of size $n$, denoted by $c_n$, is finite, the *generating function* $C(z)$ of $\mathsf{C}$ is the formal power series

$$C(z) = \sum_{c \in \mathsf{C}} z^{|c|} = \sum_{n=0}^{\infty} c_n z^n.$$

We let $[z^n]C(z)$ denote the coefficient of $z^n$, $c_n$.

The symbolic method (Flajolet and Sedgewick [14]) is a framework that allows the construction of a combinatorial class $\mathsf{C}$ in terms of simpler ones, $\mathsf{B}_1, \dots, \mathsf{B}_n$, by means of specific operations, and such that the generating function $C(z)$ of $\mathsf{C}$ is a combination of the generating functions $B_i(z)$ of $\mathsf{B}_i$, for $1 \leqslant i \leqslant n$. Such generating functions can be used, when considered as analytic functions over real or complex numbers, to obtain the exact, or, more often, the asymptotic behavior of the coefficients. Multivariate generating functions can be used to simultaneously analyze different measures for a combinatorial class. The framework of analytic combinatorics [14] provides a powerful tool for asymptotic average-case analysis, by relating the enumeration of combinatorial objects to the algebraic and complex analytic properties of generating functions.

Among the formal languages, the regular languages are fundamental structures in computer science. Regular languages do not have a so-called "perfect representation model" for which every desired manipulation, e.g., membership, minimality, equivalence, reverse, boolean operations, etc., is optimal. For instance, although regular expressions (REs) are a particularly powerful notation for regular language representation, even membership testing is a comparably expensive operation. For deterministic finite automata (DFA), the membership testing is optimal, but DFAs simulate REs, or non-deterministic finite automata (NFA), with exponential cost. These non-complexity-preserving simulations by equivalent combinatorial models have been extensively studied, for the worst-case, in the literature [20,21,23]. Nicaud [30] presented an average case study of the size of the Glushkov automata, proving that, on average, the number of transitions is linear in the size of the expression. This analysis was carried out using the framework of analytic combinatorics. Following the same approach, Broda et al. [7,6] proved that the size of the partial derivative automaton is, on average, half the size of the Glushkov automaton. Here we present the main mathematical tools necessary for such an analysis. We then illustrate the analytic combinatorial method by deriving, in a unified framework, new asymptotic average-case complexities of several conversions between regular expressions and equivalent $\varepsilon$-automata. These results are interesting as they show that asymptotically, and in the average case, the complexity gap between the several constructions is significantly larger than in the worst case. Furthermore, one of the $\varepsilon$-NFA constructions approaches, again asymptotically and on average, the corresponding $\varepsilon$-free NFA construction.

In Section 2 we show, as a simple illustrative example, how one can use a generating function to obtain the exact number of words of a given size represented by a specific regular expression. In Section 3 we present the relevant notions and results from complex analysis required in what follows. The symbolic method is then summarized in Section 4. In Section 5 we use that method to compute the generating function corresponding to the regular expressions given by a particular grammar. Multivariate cost analysis is addressed in Section 6, where it is applied to the example used throughout the article. Finally, in the last section, we present new results on the average-case complexity of several $\varepsilon$-NFA constructions.

## 2. Counting with generating functions

Suppose we want to know exactly how many words of length $n$ are represented by the regular expression $(\mathtt{ab} + \mathtt{c} + \mathtt{d})^\star$.

It is easy to count this number of words for small values of $n$. For $n = 0$ there is just one word: $\varepsilon$. For $n = 1$ there are two possibilities: $\mathtt{c}$ and $\mathtt{d}$. Let us denote the number of words of length $n$ by $w_n$. Thus we already know that $w_0 = 1$ and $w_1 = 2$. The value of $w_n$ can be easily obtained from the values of $w_{n-1}$ and $w_{n-2}$. Each word of length $n$ either ends with a character $\mathtt{c}$ or $\mathtt{d}$ (Fig. 1), in which case it results of the concatenation of a word of length $n-1$ with that character, or it ends with a character $\mathtt{b}$ in which case it must be the result of the concatenation of a word of length $n-2$ with the word $\mathtt{ab}$. Thus each word of length $n-1$ gives rise to two different words of length $n$ and each word of length $n-2$ produces, concatenated with $\mathtt{ab}$, just one word of length $n$ that cannot be the concatenation of a word of length $n-1$ and a single character. Hence we have the following recurrence relation (for $n \geqslant 2$):

$$w_n = 2w_{n-1} + w_{n-2}. \tag{1}$$

One way to obtain a closed formula for $w_n$ from such a relation is through the use of the ring of formal power series. Multiplying (1) by $z^n$ and "adding all" the resulting equalities, we get

$$\sum_{n \geqslant 2} w_n z^n = 2 \sum_{n \geqslant 2} w_{n-1} z^n + \sum_{n \geqslant 2} w_{n-2} z^n. \tag{2}$$