Automatic Analysis

Paul Zimmermann

This chapter contains the following papers from Philippe Flajolet:

— [31] *A complexity calculus for classes of recursive search programs over tree structures*, with Jean-Marc Steyaert, proceedings of the 22nd Annual Symposium on Foundations of Computer Science, pages 386-393, 1981;

— [35] *Élements d'un calcul de complexité de programmes récursifs d'arbres*, with Jean-Marc Steyaert, invited talk to the *Mathématiques de l'Informatique* AFCET colloquium, Paris, pages 81-91, 1982;

— [67] *A complexity calculus for recursive tree algorithms*, with Jean-Marc Steyaert, Mathematical Systems Theory, volume 19, pages 301-331, 1987. A preliminary conference version appeared in *Journées d'Avignon, Théorie des Langages et Complexité des Algorithmes*, pages 39-88, 1983;

— [79] *Lambda-Upsilon-Omega: An assistant algorithms analyzer*, with Bruno Salvy and Paul Zimmermann, proceedings of AAECC-6 (Applied Algebra, Algebraic Algorithms and Error-Correcting Codes), edited by T. Mora, volume 357 of Lecture Notes in Computer Science, pages 201-212, 1989;

— [80] *Lambda-Upsilon-Omega: The 1989 CookBook*, with Bruno Salvy and Paul Zimmermann, INRIA Research Report 1073, 1989, 116 pages;

— [94] *Automatic average-case analysis of algorithms*, with Bruno Salvy and Paul Zimmermann, Theoretical Computer Science, volume 79, number 1, pages 37-109, 1991.

Related papers: [26], [28], [63], [69], [86], [88], [91], [108], [110], [111], [112], [113], [118], [128], [130], [137], [201].

Two of Philippe Flajolet's main goals in his research were first to design algorithms that would work for very broad classes of inputs, and secondly to implement those algorithms to demonstrate their efficiency. This general approach is reflected by some words that systematically appear in his papers: "general theory", "automatic program analysis", "calculus", "general analysis", "algebraic methods", "class". He applied this framework successfully to several domains: analytic combinatorics (Volume I, Chapter 1), singularity analysis (Volume I, Chapter 2), Gaussian limit laws (Volume II, Chapter 1), dynamical systems (Volume II, Chapter 3), tries and digital

search trees (Volume III, Chapter 3), term trees (Volume IV, Chapter 1), and random generation and simulation (Volume VI, Chapter 3).

This chapter discusses his research about automating the analysis of algorithms. Ph. Flajolet was mostly focusing on the *average-case* analysis of algorithms, which — compared to worst-case analysis — relies on completely different methods. The general idea was already clear in the minds of Flajolet and Steyaert in the first papers [31,35] from 1981 and 1982: *"we give systematic translation rules which map programs into complexity descriptors"*. What makes this method powerful is that it avoids the classical *one-shot* analyses using recurrences — as for example in the work of Wegbreit — and *ad-hoc* solutions, which become quickly very complex. Instead generating functions are used, both to enumerate the possible inputs — counting generating functions — and to measure the complexity of programs — cost generating functions. Also by its nature itself, this general method applies ideally to recursive descent procedures over recursively defined data structures [31], in particular algorithms working on trees [28].

The *complexity calculus* described by Flajolet and his coauthors is not sufficient alone: what makes the approach work magically is the close connection with the *singularity analysis* of generating functions (see Volume I), and the fact that it is not necessary to obtain closed forms of the generating functions, since the asymptotic analysis can be performed — sometimes even more efficiently — directly on the implicit equations. Around 1980, the singularity analysis methods were quite new, with mainly the Darboux-Pólya method, advertized by Meir and Moon in 1978, and used in [26] and [28].

First look at paper [31], and its extended journal version [67] (which took 4 years to Ph. Flajolet and J.-M. Steyaert to revise). The relation between combinatorial structures or recursive programs and generating functions takes its roots in the works of Rota, Foata, Schützenberger, Jackson and Goulden. If we look at Figure 1 in [31], the set of 11 rules is already quite impressive, and only little progress has been made since that time! Therefore the comment at the end of Section 2 is quite amusing: *"In this preliminary report, we have only given a kernel system that is sufficient for our purposes but which neither achieves minimality nor the fullest possible generality"*. Ph. Flajolet's "dream" of mechanizing results such as Theorems 1 and 2 of [31] for the average-case analysis of differentiation algorithms was later realized in 1989, see for example Report 5 in [80]. Paper [31] already contains some meta-results such as Theorem 6, which says that if Algorithm B is the iteration of Algorithm A over all subtrees of a tree, then the average-case complexity of B can be explictly deduced from that of A by some simple *translation rules*. The remark about depth-r algorithms following Theorem 6 is quite interesting, when one relates it to Report 7 in [80].

Paper [35] is a French variant of [31], with different examples. As usual in Ph. Flajolet's papers, numerous examples illustrate the general idea: binary trees, tree pathlength, pattern matching in trees, formal differentiation, ... And as in [31], meta-theorems are stated (Theorems 1 and 2), and an important remark is that the general method can a priori forbid average-case complexities like $O(n^\alpha)$ for a non-rational α, or polynomials in $\log n$. One main difference with [31] is that paper [35] speaks about exponential generating functions.

Paper [79] describes the Lambda-Upsilon-Omega system ($\Lambda\Upsilon^{\Omega}$ for short), which completes the project started in [31]: *"We have presented here a preliminary version of a system which still needs to be put into a clean form and reduced to a minimal set of rules"*. $\Lambda\Upsilon^{\Omega}$ was developed by Salvy and Zimmermann, based on the work of Flajolet and Steyaert from 1987 to 1992 [31,35,67]. It consists of an 'Algebraic Analyzer', which translates an algorithm described in some specific language into generating functions, and an 'Analytic Analyzer', which deduces from those generating functions an asymptotic expansion of the average-case complexity of the algorithm, for example:

```
Time for Diff on random inputs of size n is
```

$$8\frac{n(-240+37\sqrt{42})}{(-13+2\sqrt{42})\sqrt{42}}+O(1)$$

Paper [79] describes a very early version $\Lambda\Upsilon^{\Omega}{}_0$ of the system, using the Le_Lisp dialect for the algebraic analysis. The Algebraic Analyzer was later rewritten in the Caml language designed at Inria at the same time, while the Analytic Analyzer was implemented using the Maple computer algebra system. (Ph. Flajolet and his group had strong connections with the developers of Maple, in particular Gaston Gonnet; around 1990 the Algorithms project was distributor of Maple to French universities.) The translation rules implemented in $\Lambda\Upsilon^{\Omega}$ formed a subset of those described in e.g., [31], however, the automatic translation made it possible to analyze large programs, and the analysis was less error prone (apart from bugs in the system of course). One main feature of $\Lambda\Upsilon^{\Omega}$ with respect to previous automatic systems is that, thanks to the generating function approach, dealing with recursive programs was possible, and in fact sometimes easier, since most techniques used by the Analytic Analyzer apply to *implicit equations* too. In fact $\Lambda\Upsilon^{\Omega}$ ideally applies to *recursive descent* programs over *decomposable structures* and a similar approach was used by Greene in his PhD thesis (cf Volume VI, Chapter 3: random generation and simulation). Section 4 of [79] describes the methods used by the Algebraic Analyzer, which were further refined in Salvy's PhD thesis. Note that the Algebraic Analyzer was interesting by itself, since at that time it was able to find asymptotic expansions of several expressions where the Maple `asympt` command was either failing or giving a wrong result.

The Lambda-Upsilon-Omega 1989 CookBook [80] was written a few months after the initial description [79] of the $\Lambda\Upsilon^{\Omega}$ system. It consists of a collection of 18 "automatic reports" in three domains (regular languages and tree automata, context-free languages and symbolic manipulation algorithms, combinatorial problems). Each report starts with a specification of the problem to be analyzed, followed by the source program in the Algorithm Description Language (ADL) used by $\Lambda\Upsilon^{\Omega}$, the algebraic analysis, the solution of the generating function equations, and finally the asymptotic analysis. Except the description of the problem and the source ADL program, each report was automatically generated (with some parts slightly edited to make the CookBook easier to read). Some of the reports solve non-trivial problems from the literature, for example improvements to binary exponentiation algorithms (Report 2), problems leading to mutually recursive functions (Reports 3 and 8). $\Lambda\Upsilon^{\Omega}$ was often used to solve questions asked on the `sci.math` newsgroup (see Reports 4 and 18),

and in return those questions suggested several extensions of the system. It should be noted that at that time (around 1989) neither the Encyclopedia of Integer Sequences from Sloane, nor the GFUN or COMBSTRUCT programs did exist; such tools would have made some analyses much easier. Report 6 analyzes the 3rd order derivative of tree expressions, and is an example of an analysis which would be too tedious to perform by hand. Ph. Flajolet knew how to find amusing examples that would draw the attention of the reader: the random trains (Report 12) are such an example, which uses most of the combinatorial constructions for labelled objects. Not only this example is amusing, but the asymptotic analysis to find the average position of the first empty wagon is non-trivial, since one has to deal with a singularity defined as the root of an implicit equation. Report 14 shows that the system can also compute the variance of a cost distribution, or moments of any order; in that report, some "fake" procedures were written by hand to mimic the variance, but this process could be automated too. Some problems dealt with in [79] are still open: for example it is still unknown if the number d_n of permutations that can be sorted by a double-ended queue — or *deque* for short — admits an algebraic generating function (sequence A182216 of the OEIS). Albert, Atkinson and Linton give in 2010 (Annals of Combinatorics) a lower bound of 7.890 and an upper bound of 8.352 for the growth rate of d_n, and mention the possibility that the actual growth rate would be 8, and would be the same as two stacks in parallel (which is similar to a deque with a frontier in the middle, thus can sort fewer permutations).

Paper [94] is the most complete one in this series of papers about the automatic average-case analysis of algorithms, and describes the state-of-the-art in 1991. Similarly to the CookBook [79], it presents a collection of *Automatic Theorems* that were proven with $\Lambda\Upsilon\Omega$. This paper gives more details about the labelled and unlabelled universes, leading respectively to exponential and ordinary generating functions. It also clearly defines what is a *well defined* specification (with no ε-transition in the language of regular automata), and how one can transform such a well-defined specification into a *reduced* specification, i.e., with atomic contructions only. The authors also give upper bounds for computing the couting sequences up to size n. The second part of the paper explains in some [1] detail the analytic theory which enables $\Lambda\Upsilon\Omega$ to obtain the asymptotic expansion of the generating functions computed by the Algebraic Analyzer. Several algorithms are given, with many examples illustrating them. For more details about the Analytic Analyzer, the reader should have a look at Volume I, Chapter 1 (analytic combinatorics) and Volume I, Chapter 2 (singularity analysis).

The work of Philippe Flajolet in automatic (average-case) analysis is strongly related to his work in random generation, in particular the one on the *recursive method* (see Volume VI, Chapter 3). Indeed, as already noticed by Greene, when one is able to count *decomposable structures*, one is also able to generate them at random.

As pointed out above, the $\Lambda\Upsilon\Omega$ system could not only get the asymptotic average-case cost of a program, but also its variance, and moments of any order. In most cases,

1. As usual, Ph. Flajolet likes joking: footnote 14 of that 73-page long paper says: *"Periodicity issues complicate the picture without significantly changing it, so that we do not discuss them in depth in the present paper which is only a short introduction to the subject."*

those moments enable one to guess the full asymptotic distribution of the program cost. One of Philippe Flajolet's dreams was to be able to obtain directly that complete distribution, instead of only a few moments (see Volume II, Chapter 1: Gaussian limit laws). Surely his research contribution will help this dream become true!