

Alfonso Caracciolo di Forino and Generalized Markov Algorithms

SIMONE MARTINI, DISI, University of Bologna, Bologna, Italy

We explore the contributions of Alfonso Caracciolo di Forino, 1925–1996, to the creation of a school of formal methods in Italy, focusing on the problem of the formal definition of programming languages, including the semantic and pragmatic levels. In particular, we present his search for metalanguages and methods that could be used to give a “declarative definition” of a programming language. This research led him to the introduction of Generalized Markov Algorithms (GMAs), which extend Markov’s Normal Algorithms with metalinguistic variables, computable functions, and conditional applicability. For Caracciolo, a language definition was a GMA that, given a string, either rejects it if it is not a legal program or gives its meaning as a state-transforming function, thus taking care of the syntax (both context-free and context-dependent aspects) and semantics of the language, including non-terminating computations. Caracciolo’s group applied GMAs to several language definitions, to a complete simulation of the formalism used by the IBM Vienna Research Center to define PL/I, and, more generally, to the definition and simulation of dynamical systems. We contextualize these contributions in the Italian computing environment of the late 1950s and 1960s.

CCS Concepts: • **Social and professional topics** → **Historical people; History of computing theory; • Software and its engineering** → *Software notations and tools*;

Additional Key Words and Phrases: History of programming languages, Italy, Markov algorithms, PANON, calcolatrice elettronica pisana, CEP, history of formal methods, simulation languages, operational semantics

ACM Reference Format:

Simone Martini. 2026. Alfonso Caracciolo di Forino and Generalized Markov Algorithms. *Form. Asp. Comput.* 38, 3, Article 23 (July 2026), 21 pages. <https://doi.org/10.1145/3770758>

1 Introduction

Recent contributions [8, 9] have described the early years of theoretical computer science in Italy, highlighting the pioneering work of Corrado Böhm in Rome, the crucial role of Eduardo Caianiello in Naples in establishing cybernetics as a research field, and the creation of a powerful research environment in Pisa, around the design and construction of the **Calcolatrice Elettronica Pisana** (CEP). One of the main designers of the CEP was Alfonso Caracciolo di Forino, responsible

Guest Editor: Cliff Jones

Research partially supported by the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, funded by the European Union-NextGenerationEU: (a) Investment 1.3, Call for tender No. 341 published on 15.3.2022 by the Italian Ministry of University and Research (MUR): Partenariato SERICS (PE00000014); (b) Investment 1.1, Call for tender No. 104 published on 2.2.2022 by the Italian MUR, Project Title “Learning Informatics” – CUP E53D23007720006- Grant Assignment Decree No. 959 adopted on 22/04/2022 by the Italian MUR.

Partially supported also by INdAM GNSAGA..

Author’s Contact Information: Simone Martini, DISI, University of Bologna, Bologna, BO, Italy; e-mail: simone.martini@unibo.it.



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 1433-299X/2026/07-ART23

<https://doi.org/10.1145/3770758>

alongside Elio Fabri for the machine's first logical and architectural design. Caracciolo is, of course, a character present in the papers cited above, and his role in developing the CEP is well documented in historical literature. However, there is no comprehensive scholarly account of his wider scientific endeavours, which originated in the context of the CEP, but had a much broader scope. Even the mere list of his publications is ill represented in current repositories: the ACM digital library, for example, lists only two papers [31, 56] under "Alfonso Caracciolo di Forino" and only one [23] under "A. Caracciolo".¹ The lack of literature describing Caracciolo's activities is interesting in itself, since, as we discuss in the final section of the article, at the peak of his career, he abandoned the research field to which he had contributed and stopped writing scientific papers. It is unfortunate that there has been little scholarly interest in his scientific contributions, as this underrepresents his pivotal role in establishing an Italian community in theoretical computer science, as well as in the formation of the international community centred on the study of "formal language description languages", which led to the establishment of IFIP WG 2.2 (see [6, Ch. 4 and Ch. 7] for one of the few mentions of his role on this.)

Central to Caracciolo's scientific production was, in his own words, the question of the "formal definition of Programming Languages, including the semantical and pragmatical² levels of meaning and interpretation" [57]. To this end, he developed a methodology based on a generalization of Markov normal algorithms, which he and his young collaborators applied to the definition of several programming languages (including a symbol processing language, PANON, and a simulation language, SL/1, of their invention), to the definition of the behavior of machine tools and industrial plants, and to simulation languages, in a broader context where **Generalized Markov Algorithms (GMAs)** were seen as the basis of a general theory of dynamical systems. His use of GMAs for the semantics of programming languages was independent but substantially analogous to the method employed by Adriaan van Wijngaarden and, later, Jaco de Bakker.

This article fills the historiographic gap by describing Caracciolo's work in establishing the first school of formal methods in Italy in the 1960s, creating and applying techniques for formally defining programming languages. We refer, instead, to the literature on the CEP (see below) for Caracciolo's role in its design. The main part of the article introduces and discusses his scientific contributions. Sections 2 and 7 contextualize this research in the Italian computing environment of those years. Section 6 presents brief critical reflections on Caracciolo's scientific approach from a contemporary perspective.

2 The Early Years

At the beginning of the 1950s, several Italian research institutes realized that they should have access to an electronic computing machine. Some of them (such as Politecnico di Milano and INAC-CNR in Rome) decided to buy a machine from the manufacturers at the time. The University of Pisa, at the suggestion of the physicist and Nobel Prize winner Enrico Fermi, decided to build such a machine from scratch, developing all the necessary hardware and software technology in-house, without any real prior know-how. The design and development of the CEP and its prototype (the *Macchina Ridotta*, MR), the pivotal role of the Italian private firm Olivetti in investing in this electronic

¹In the references of the present article, we use "Caracciolo" or "Caracciolo di Forino" as it appears in the printed versions of the original papers.

²For the meaning of "pragmatics," Caracciolo followed Gorn's definition [68, page 337]:

Pragmatics is concerned with the relationship between symbols and their "users" or "interpreters". We will "interpret" the words user and interpreter to have a mechanical sense, i.e. to mean "processor". [...] The word "processor" in our context refers ambiguously to mechanisms, algorithms, programs for a machine, or systems and organizations composed of machines, programs, and algorithmized people.

computer project, and the international research contacts that the CEP generated have already been described in the literature (see, for example, references [15, 66] for national-level accounts and [59, 61] for specific aspects of the CEP project and Olivetti's role). The project officially started in 1955, with the creation of CSCE (Centro di Studi sulle Calcolatrici Elettroniche), organized into two main sections: Engineering (to design and realize the hardware of CEP) and Mathematical Logic (responsible for the logical and architectural design, the instruction set and the development of all the necessary software and programming techniques). Alfonso Caracciolo, a 30-year-old physicist who had graduated from the University of Turin only three years earlier, was recruited as head of the Mathematical Logic Section.

Alfonso Caracciolo di Forino was born in Penne³ in 1925, into a noble Neapolitan family with a long history.⁴ He grew up in Rome; in 1943, he enrolled in Physics at the University of Roma La Sapienza. After a good start, he began to feel unsatisfied. Even taking the war period into account, in July 1950 (physics was a four-year program) he was still struggling with some exams and wrote to Ludovico Geymonat,⁵ the most prominent Italian epistemologist of the time. Alfonso told Geymonat that he was interested in fundamental topics and the use of language in physics, having read and been fascinated by Eddington's "The Philosophy of Physical Science", but that he could not find a guide on these subjects in Rome's albeit excellent physics institute. Geymonat replied by return mail, they began to correspond, and Alfonso found a mentor who, despite their age difference, would become a friend in time. In 1951 Caracciolo wrote to Vittorio Somenzi, another epistemologist and family friend, asking about "computing machines", the first archival mention of the topic that would later become his main scientific interest. In January 1952, he moved to the University of Turin as a student, where Geymonat remained a strong influence. He graduated that same year and experienced the cultural environment of the Centro di Studi Metodologici, which Geymonat and others had founded in 1948 to carry out research at the frontier between philosophy, logic, science, technology, and language.

It was in that context that Caracciolo began to develop his interest in formal logic and languages, publishing two still immature, tentative papers. The first one [19] investigated what kind of constraints on the formal arbitrariness of axioms and syntax are induced by the metatheoretical requirements of completeness, effectiveness, and categoricity. The second [28], presented at the 11th International Congress of Philosophy in Paris, offered some reflections on the use of language in Physics.⁶ For the academic year 1952-1953, he served in Turin as "voluntary assistant professor"⁷ and translated Hans Reichenbach's "Philosophic Foundations of Quantum Mechanics" into Italian⁸

³Penne is a small town in the Abruzzo region, where the Caracciolo family owned a *palazzo*. For the biographical notes we have used material from the archives of the University of Turin, the University and CSCE in Pisa, the LUISS University in Rome, Cliff Jones in Newcastle, Adriaan van Wijngaarden in Amsterdam, Ludovico Geymonat in Milan, the private archive of the Caracciolo family, and the personal recollections from some of his former students: Luigia Carlucci Aiello, Gualtiero Leoni, Vincenzo Manca.

⁴His father was the Prince of Forino and the Duke of Belcastro, titles that, in due course, passed to Alfonso.

⁵Ludovico Geymonat, 1908–1991, was born in Turin, where he graduated in Philosophy in 1930 and in Mathematics in 1932. Assistant professor in Mathematics in Turin, in 1934 he went to Vienna to study the neo-positivist philosophy of the Vienna Circle. In 1940 he joined the clandestine Italian Communist Party. In 1949 he was appointed professor of Theoretical Philosophy in Cagliari, then of History of Philosophy in Pavia from 1952, and finally of Philosophy of Science in Milan from 1958, the first chair of this discipline in Italy. He was responsible for establishing an Italian school of philosophy of science and for the renewed interest in mathematical logic after Peano.

⁶The paper received a nonenthusiastic review by Patrick Suppes in the *J. of Symbolic Logic*, Vol. 19(2), 1954, pp. 149 - 150.

⁷*Assistente volontario*, a non-paid position.

⁸*I fondamenti filosofici della meccanica quantistica*, Einaudi, Torino, 1954. More than ten years later, he also translated Hermann Weyl's "Philosophy of Mathematics and Natural Sciences": *Filosofia della matematica e delle scienze naturali*, Boringhieri, Torino, 1967.

at Geymonat's invitation. Back in Rome in 1953, he came into contact with Corrado Böhm⁹ and other researchers from INAC-CNR who were in the process of acquiring a Ferranti Mark I.¹⁰ In autumn 1954, Marcello Conversi, a physicist from Pisa who was responsible for organizing and staffing the CSCE, offered Caracciolo a trial period. The official start date for CSCE operations was March 1955. Caracciolo's first task was to write a report [20] on the "modern computing machines". The paper's main sources were an article by Bruno de Finetti in an Olivetti in-house journal [64], two reports on computing machines by Enzo Aparo and Dino Dainelli of INAC-CNR [4, 5], a review of a recent British symposium [65], and a survey by the US Office of Naval Research [12] listing around a hundred machines in use in the USA. Instead of focusing on the different features of the existing machines, the paper described the general logical structure of what we now call the von Neumann architecture, using the term *organ* (as in von Neumann's *First Draft*) for the different units (memory, control, arithmetic unit, I/O devices.) Caracciolo observed that

*mentre nelle questioni di dettaglio le macchine finiscono per essere notevolmente diverse fra loro, nelle linee generali funzionano tutte secondo gli stessi principi. [...] si può ben dire che quasi tutte le macchine sono 'universali' nel senso che ciascuna di esse è in grado di risolvere qualunque problema che possa essere risolto da ogni altra calcolatrice elettronica a cifre.*¹¹

It is a bold observation for a 30-year-old who was just entering the field of computing, which exemplifies the curiosity and depth of thought of the young Alfonso. When asked to review computing machines in order to *build* one, he seized the opportunity to remark on their abstract properties instead of analyzing all the "considerably different" models.

From 1955 to 1960 Caracciolo was mainly concerned with the logical design of the prototype of CEP,¹² the "**Macchina Ridotta**" (MR), which was completed in July 1957. The MR "adopted state of the art solutions that it was not easy to find all together on other machines of that period" [59]. Among these state-of-the-art solutions, we find a fully microprogrammed control. A key element that made microprogramming possible was the MR's small instruction set (only 32 instructions), for which Caracciolo's Mathematical Logic Section was responsible.

At that point, it was clear that CSCE also needed to prepare people to fruitfully *use* the computer. A first technical course for a dozen graduating engineering students was held in 1956 [67]. However, the most important decision was to intervene directly in the education of university students by introducing programming topics into the standard mathematics and physics curricula. At that time, the courses that could be taken for a particular degree were determined by national law and there was no course directly available for "computer programming". For the academic year 1958–1959,

⁹Corrado Böhm, 1923–2017, was born in Milan [14]. Escaping racial persecution, he graduated in Electrical Engineering in 1946 at the University of Lausanne, before becoming a graduate student and research assistant at ETH in Zürich. He had two supervisors: Eduard Stiefel, the director of the Institute for Applied Mathematics and designer of the *Elektronische Rechenmaschine der ETH* (ERMETH), and Paul Bernays, Hilbert's former assistant and close collaborator. Following the submission of his thesis and a brief stint at Olivetti, he was appointed as a permanent researcher at the National Institute for the Applications of Calculus (INAC) of the CNR in Rome in 1953. He remained at the INAC-CNR until he took up a professorship in Modena in 1969. He then moved to Turin in 1970 and finally to Roma La Sapienza in 1974, where he concluded his career as a professor emeritus. A mentor to generations of computer scientists, he produced his best-known scientific results in the 1960s: the theorem with Giuseppe Jacopini showing that any flowchart can be effectively transformed into one containing only conditionals and while loops, and the so-called *separation theorem* for λ -calculus.

¹⁰Assembly of the computer was due to begin in December 1954.

¹¹"While in matters of detail the machines end up being considerably different from each other, in general, they all work according to the same principles. [...] it may well be said that almost all the machines are 'universal' in the sense that each of them is capable of solving any problem that can be solved by any other electronic digit calculator".

¹²The CSCE archives contain many documents, primarily internal reports that are occasionally presented at conferences, e.g., [21, 24–27, 29].

the University of Pisa decided to contract Böhm to teach the course *Calcoli numerici e grafici*,¹³ the then title for what a few years later would be called Numerical Analysis. The lesson plan¹⁴ for this course has been published and discussed [60]. Böhm “naturally” embedded standard numerical topics into a mathematical theory of computation. Not citing numerical analysis subjects, we find: Moore automata,¹⁵ **Turing machines (TMs)** and universal TMs, partial and total computable functions, the existence of uncomputable functions, the logical structure of a digital computer, and techniques for programming such a digital computer. While most other universities around the world focused on the details of the computer architecture they had, Böhm made clear that programming should be understood within a general theory of computation.

We find the same vision also in Caracciolo’s teaching at the time. He gave a first lecture on “Considerazioni generali sulle moderne calcolatrici elettroniche”¹⁶ in January 1955 [3]. Then, the archives of the University of Pisa have records of a course on “Programmazione di una calcolatrice elettronica”¹⁷ in 1959-60 and 1960-61 for a graduate degree in Nuclear Physics.¹⁸ From 1961 to 1969 he was in charge of the course “Cibernetica e teoria dell’informazione”,¹⁹ in 1968 he also started to teach “Teoria ed applicazioni delle macchine calcolatrici”,²⁰ the course finally created to introduce computers and programming to students majoring in science; in 1970 and 1971 he gave “Linguaggi formali e compilatori”,²¹ more centered around formal languages than compilers, though, according to the recollection of some students.²² The archives only have the lesson plans of Cybernetics, for the three years starting from 1961-1962. For the first of these academic years, we may group the presented topics in three sections. Less than a quarter of the course was devoted to cybernetics proper: neural nets à la McCulloch and Pitts, Rosenblatt’s perceptron, shape recognition, feedback, and so on. The second part dealt with automata and computability: finite automata, TMs, universal TMs, undecidable problems, Post systems, Markov algorithms and their equivalence to TMs. This section is similar to Böhm’s, with the notable addition of Markov algorithms, see Section 5. The largest part was devoted to the structure of a digital computing machine (including some details of the CEP), and to programming and formal languages. He introduced elements of ALGOL and FORTRAN, emphasizing the recursive nature of ALGOL’s block structure. He then covered how programming languages can be described: syntax, semantics and metalanguages for formalizing syntax, the Backus-Naur form, context-free languages and some of their properties. In the following years, the cybernetics part was reduced to just a few lectures, and the course became centred on computability, automata, and formal languages.²³

The part on programming languages and Markov algorithms reflected Caracciolo’s research interests. By the end of the 1950s, the CSCE was in desperate need of people proficient in programming techniques, especially systems programming and translators (compilers). They formalized further collaboration with Böhm (who remained a permanent researcher at INAC-CNR, in Rome), hired a young programmer (Florence Ragusa) for two years from the Courant Institute in New York (via Peter Lax), but it was clear that without an active local research community on these very topics

¹³Computing by numerical and graphical means.

¹⁴*Registro delle lezioni*, preserved in the archives of the university.

¹⁵[80] had been published just two years before Böhm’s course was given.

¹⁶General considerations on modern electronic computing machines.

¹⁷Programming an electronic computing machine.

¹⁸Scuola di Specializzazione in Fisica Nucleare

¹⁹Cybernetics and information theory.

²⁰Theory and applications of computing machines.

²¹Formal languages and compilers.

²²Pierpaolo Degano, personal communication.

²³The CSCE archives also contain lecture notes for this course, for 1967-1968, in three parts: Elements of general theory of automata, Elements of theory of algorithms, Elements of formal theory of languages.

they could not get the right momentum. Caracciolo's interest in languages and their foundations, which had been evident since his university days, finally had the opportunity to become his field of research.

3 “Declarative Definitions” of Programming Languages

After the long series of reports on the CEP, Caracciolo published a paper on more general topics in 1961 [22], introducing “selection functions”, proved to generate a Boolean algebra. Selection functions are essentially equivalent to the “conditional expressions” used by McCarthy in the same years [75–77], as Caracciolo observed in a sequel paper [32]. In this second paper, he generalized selection functions to selections from n objects (to model the case construction under discussion for extensions of ALGOL), relating it to n -valued and algebraic logic, also treated in a paper he presented at a summer school the following year [37]. These papers are among the most abstract and mathematically oriented of his production.

More interesting for our story is a series of papers which he presented between 1962 and 1966, mainly at several IFIP conferences, all focusing on the formal description of programming languages. The first of these was in 1962 at the IFIP Symposium on Languages for Processor Construction [30], containing a brief (two pages) account of his current research interests. Astarte [6, Sect. 2.3] presents clearly how during the meetings for the design of ALGOL, a recurring theme was “how to describe the language”, referring to both syntax and semantics. Starting with the informal description in the “Preliminary Report” on IAL [84], passing through Backus' formal description of the same language [10], and arriving at the “Report” on ALGOL 60 [11], we see the coexistence of formal methods (the BNF) and carefully formulated natural language, the former used (in the “Report”) to define the syntax, the latter kept for the semantics. However, natural language was also used to define the context-dependent constraints on the syntax (mandatory and unique declaration of an identifier, compatibility between formal and actual parameters of procedures, etc.). There were, and always have been, different ideas among the members of the ALGOL committee about the level of formality to be used, and about the suitability of formal methods for defining the language. It is in this context that we should read Caracciolo's 1962 paper [30], the purpose of which he described the following year as follows:

Our original idea was to develop a particular kind of Programming Language for a complete formal definition of Programming Languages, including the semantical and pragmatistical levels of meaning and interpretation. [57]

He was looking for a formal metalanguage that “provides adequate means for formalizing the definition of the language [...], from which parsing routines will be derived by an appropriate compiler [...] *without having to specify the translation procedurally*”.²⁴ He mentioned that possible tools for such a metalanguage could be Markov's normal algorithms. It is unusual for a scientist to openly publish research plans that they will stick to for the next 10 years.

That he intended the “parsing routines” to deal with the context-dependent aspects of the language as well was made clear in the 1963 paper [31] in the Communications of ACM, in which, observing that context-dependent features could be seen as “metalinguistic conventions of local validity,” he suggested that the legal expressions in a certain context form “a *locally defined* context-free language”. As an example, the paper gave grammars in BNF for <expression> and <statement> in ALGOL, divided in two parts: the first (*general* productions) contained the same rules of the Report; the second (*local* productions) were “derived from the locally valid set of declarations and label attachments”. The paper gave neither a formalism nor a general way to derive such local

²⁴From [30]; the last sentence is taken from Caracciolo's reply in the discussion after the presentation of the paper. The italics are mine.

productions from the language definition and a specific program. However, it concluded by saying that preliminary results toward the formulation of a formal theory have already been obtained. It is not difficult to link these preliminary results to his parallel, ongoing research on GMAs.

These ideas are extended and formalized in a broader context in the paper presented at the 1964 IFIP Working Conference on **Formal Language Description Languages (FLDL)** [38]. The paper introduced the notions of *formal linguistic systems* and *formal grammatical systems*, which are hierarchical systems for the definition of formal languages, of which the examples given in the previous paper were instances. For the present paper, the discussion that followed the presentation and was recorded in the proceedings (more than five pages) is more important than the content of Caracciolo's paper. First, in response to — and in partial agreement with — Saul Gorn and Seymour Ginsburg, Caracciolo clarified that the paper aimed to formally justify the need to include “some semantic features” in the definition of a language, and that “the most important part of language definition is the semantics”. Second, Peter Naur provoked Caracciolo:

NAUR: I was puzzled to hear your difficulty of handling the block structure [...]. In any compiler this is done simply, mechanically, by means of a suitable push-down list. What is wrong with this?

After some exchanges, Caracciolo showed his hand (the italics are mine):

CARACCILO: The problem is to find a metalanguage for doing that in a *declarative way, not in an operational way*²⁵ [...] What I want to do is to define grammars, where possible, in such a way that a recognizing algorithm can be automatically derived.

Finally, in replying to Edsger W. Dijkstra, Caracciolo stressed again the previous two points:

CARACCILO: This is the central problem in automatic programming. I can define a syntactic analyzer for ALGOL and also an interpreter for ALGOL by writing an algorithm. Now, I think the point of great interest is always to get a different definition, a sort of declarative definition, and to get automatically from that an algorithm, as you have for context-free languages.

Extrapolating from his answers, we see that he envisaged some sort of grammatical definition for “the whole language, syntax, semantics, and pragmatics” (page 48), in which it was convenient or necessary for the “grammar” to refer to “a certain set of objects by a certain set of properties” (page 51). He admitted that some of these properties would necessarily be given by a procedure; others may be given in a more “declarative” way. But the important point is to use “a metalanguage in which to embed all the restrictions in such a way that one can automatically find a recognizer for” the original language (page 51). We discuss in Section 5 how these requirements are met by GMAs.

In a 1965 paper [36], of a more epistemological character, instead of “declarative” he stated that what is needed are linguistic systems “powerful enough for defining programming languages in a *natural way*”.²⁶ The paper is interesting for several reasons. First, Caracciolo insisted

²⁵In later years the adjective “declarative” would be used mainly in connection with *denotational* semantics, whereas Caracciolo's approach would certainly be placed in the *operational* camp. The distinction between the two approaches had not been made in 1964. It should be clear that he was referring to a description “at the same level of abstraction” of the constructs to be defined, without too many algorithmic, unnecessary details. One of the earliest discussions on the distinction between “operational” and non-operational was in Dana Scott's [87], where the *mathematical* meaning of a program (that is, a *function*) was contrasted with its *operational* meaning which “will generally provide a trace of the whole history of the computation” and “will involve an explicit finitary choice of representations of data”. All of the semantic descriptions presented in the FLDL papers (e.g., those by McCarty, Böhm, Landin, and Strachey) had a clear operational character (Caracciolo would have called them “interpretational”), expressing the meaning of a program as a series of syntactic transformations on a representation of the state.

²⁶Italics as in the original.

that programming languages are “languages in the fullest sense of the word, i.e., essentially communication tools”. As such, they are used to describe *jobs*, that is “anything one (man or machine) has to do”. He discussed several criteria for classifying jobs, dividing them into (i) *concrete* (such as “setting a given physical system in a prescribed state”) or *abstract* (when they abstract from any material support or real-time consideration and therefore they are essentially mathematical, or logical, entities); (ii) *ending* or *unending* (e.g., “a numerically controlled continuous process or an abstract simulation of it”); (iii) *sequential* or *parallel concurrent interacting* (whose prototype is “the functioning of a complex dynamical system with interacting parts”), further to be distinguished in *synchronous* and *asynchronous*. This broad view of what a programming language is about was matched by his continuing interest in languages for machine tools or chemical plants, for simulation, and, more generally, for the description of dynamical systems, which we discuss later. Second, he discussed the *interpretational approach* to programming language semantics, citing as one of the “most interesting set of contributions in this field” the papers presented at the FLDL conference by Böhm, Christopher Strachey and Peter Landin, noting, however, that “they are essentially limited to the definition of ending timeless computational jobs”. He contrasted their contributions with those of Adriaan van Wijngaarden [90, 91], emphasizing, in particular, his notion of “value at time t , which seems to be a really basic concept in programming theory”.

Similar ideas were emphasized again in a paper on “theoretical pragmatics” [23]. After reformulating the meaning of “pragmatics” in linguistics as that “branch which deals with the relation between languages and actions or processes”, he followed Gorn’s [68] in identifying this relation as that between the programmer and the “job executor”, where the concept of a job does indeed deal with actions and processes. He stressed that a programming language and its “job executor” are thus two sides of the same coin, demonstrating once again the importance of semantics (and its description) in the definition of a programming language. He then focussed on the formalization of “high level machine tool programming languages”, which consider actions in the physical world.

4 International Networking

With the availability of the “Macchina Ridotta” in the second half of 1957 and the realisation of the “full” CEP, formally inaugurated in November 1961, the daily workload of running the Logical Section of the CSCE eased. This allowed Caracciolo to devote some time to international research networking.

Through the Associazione Italiana di Calcolo Automatico²⁷ (AICA), he was appointed Italian representative to the IFIP technical committee on Programming Languages, TC-2. The archives record his active participation to all the annual meetings until 1970.²⁸

We already mentioned his participation to FLDL in 1964, in Vienna. Astarte [6, Ch. 4 and Ch. 7] provides an in-depth discussion of the role and impact of this conference on the research on programming language semantics, mentioning the role Caracciolo had in structuring that community around a new IFIP working group. Caracciolo had offered to organize the conference in Pisa, but the bid by Heinz Zemanek, from IBM Vienna and chair of TC-2, won. Attendance was by invitation only, “to recognized experts in one or more of the various disciplines of linguistics, logic, mathematics, philosophy, and programming whose frontiers converge around the subject of the meeting” [89, Introduction]. There were 51 people from 12 nations. To consolidate that research

²⁷Italian Association for Automatic Computing.

²⁸The minutes are preserved in the Andrei Ershov online archive, <https://ershov.iis.nsk.su/en/archive>, under IFIP/TC-2, starting with the third meeting, in Oslo, September 1963, where Caracciolo is recorded as one of the attending members (there is no note on the initial date of his term of office). He remained a member until 1976 or 1977, when the Italian representative became Maurelio Boari. The last meeting he attended was in 1970, in Eindhoven. After that he was recorded as “absent without notice”.

community Caracciolo and Thomas Steel, in the aftermath of FLDL, prepared a proposal to TC-2 for the creation of a special interest group on the topics of the conference. The proposal was approved at TC-2's sixth meeting, in 1965, and accepted by the IFIP General Assembly,²⁹ thus creating WG 2.2 on "Formal description of programming languages," whose first official meeting was chaired by Steel and hosted by Caracciolo in Sardinia, in September 1967. He remained a member of WG 2.2 until the early 1970s. In 1973, when the WG needed to resume its activities after a few years of unclear scope and poorly attended meetings, Steel circulated a letter with a list of nine papers "reflecting the fundamentals of the main areas of concern" on programming languages, including Caracciolo's [35].

In June 1964, Eduardo Caianiello invited him as a speaker to the Sixth International Spring School of Physics, in Ravello, on *Automata Theory* [18], one of the "milestone events" [9] for Italian theoretical computer science³⁰ where Caracciolo gave two presentations, on selection functions [37] and on GMAs [35].

In 1965, he organized a NATO summer school on programming languages in Pisa with some of the stars of the field.³¹ In Giorgio Ausiello's recollection [8], "the lecturers were some of the most prominent computer scientists in the world". Caracciolo's lectures presented the special purpose language he was developing for programming machine tools [33].

In September 1966 Caracciolo organized and chaired a working conference on symbol manipulation languages [13] in Pisa, under the IFIP TC-2 sponsorship, where he presented the paper [42] on the symbol processing language PANON we discussed in Section 5. In June 1968, although not a member of WG 2.1,³² he hosted a meeting of that WG in Tirrenia, near Pisa, where van Wijngaarden's controversial proposal leading to ALGOL 68 was discussed for the first time. At the end of Section 5 we mention some more conferences he organized or chaired on simulation languages.

By the mid-sixties, Caracciolo was a respected member of the international programming language research community. The major scientific contribution he developed during these years was the formal method based on Markov algorithms, which he believed could satisfy at least some of the desiderata expressed in the papers we discussed in Section 3.

5 Generalized Markov Algorithms

In 1947, Andrei Andreevich Markov Jr. proved the recursive undecidability of the Thue problem on words using a notion of formal computability that he had introduced and had proven to be equivalent to TMs. He termed this concept "normal algorithms" (usually referred to as "Markov algorithms", or "Markov normal algorithms", in Western literature; we use "MAs" for this notion.) A series of more general results, together with a general theory of algorithms, was published as a 1954 monograph [74], translated into English in 1961.

5.1 Technical Notions

This section provides a precise yet accessible introduction to MAs and their key generalizations, as presented by Caracciolo. The subsequent section recounts the socio-historical events surrounding

²⁹It was adopted at the 10th General Assembly in Nice in September 1965, although the decision is only recorded in the minutes of the 11th General Assembly as an addendum to the minutes of the previous meeting (IFIP online archive).

³⁰There were seven Italian speakers out of 26: V. Amar and G. Putzolu from Olivetti, M. Borillo and P. Camion from EURATOM Center in Ispra, C. Böhm and W. Gross from INAC-CNR in Rome, and A. Caracciolo.

³¹Besides Caracciolo himself (Special programming languages), speakers were S. Ginsburg (Theory of context-free languages), P. Landin (λ -calculus and its applications), P. Naur (The systematic design of effective compilers), A. van Wijngaarten (Formal definition of syntax and semantics of programming languages) (IEI-CNR archive, Pisa). Among the attendees, G. Ausiello, J. de Bakker, A. Grasselli, F. Luccio.

³²IFIP WG 2.1 on *Algorithmic Languages and Calculi* was formed in 1962 for the support and maintenance of ALGOL.

these formal concepts and their application to the description of programming languages and dynamical systems.

A *Markov normal algorithm* is a simple string rewriting system defined by a finite *alphabet* A and an ordered finite set of *productions* of the form $w \rightarrow v$ or $w \rightarrow \cdot v$, with $v, w \in A^*$. Given a string $s \in A^*$, a given MA is applied to s by finding in s the leftmost occurrence of a LHS of a production (productions are looked for in the order in which they are given) and replacing that occurrence with the RHS of the production. If the applied production is of the second form ($w \rightarrow \cdot v$), the process *terminates* and the rewritten string is the *result*, otherwise we keep applying the MA to the rewritten string. The numerical functions computable (under a suitable natural definition) by MAs coincide with the functions computable by TMs.

The simple notion of MA is too rigid (or, we might say, too low-level) to be applied to the “practical” description of a programming language, where it is conceptually clearer and simpler to deal with “grammatical categories”, and more generally “abstract syntax”, rather than with concrete strings of symbols. Over the years, Caracciolo introduced several extensions to MAs, collectively termed GMAs.

In *S-Generalized MAs* [57], productions may contain meta-variables standing for decidable sets of strings over the basic alphabet A . One first fixes a set of *names* for decidable languages (say, the names X and Y correspond to the languages A_X and A_Y over A , respectively.) Then, in a production one may use *indexed* names to stand for any string of the corresponding set. If the same name is used more times (on the same side, or on both sides) *with the same index*, then it denotes the same string. For example [57], the “problem of doubling the first occurrence of a word X_1 belonging to a certain class A_X ” is described as

$$X_1 \rightarrow X_1 X_1$$

and “the problem of interchanging the first two words X_1 and X_2 belonging to a certain class A_X ” as

$$X_1 \Omega_1 X_2 \rightarrow X_2 \Omega_1 X_1$$

where Ω is the name for the set of all finite words, A^* .

CF-Generalized MAs [57] (where “C” and “F” stand for “conditional” and “function,” respectively) are S-Generalized MAs where the RHS of a production may contain an arbitrary computable function on strings, and the applicability of a production may be restricted by a computable predicate. For example, given the names X, Y , and Z with the corresponding sets of strings A_X, A_Y , and A_Z , one may write

$$X_1 Z_1 Y_1 X_2 Y_2 X_1 \rightarrow \varphi(Z_1, X_2, X_1) X_2 \varphi(Z_1, X_1, X_1),$$

provided $\pi(X_1, Y_2)$

where φ is a string-valued function and π is a Boolean-valued function. The production will fire on a string s if in s we find a leftmost string $x_1 z_1 y_1 x_2 y_2 x_1$ with $x_1, x_2 \in A_X$, $y_1, y_2 \in A_Y$, and $z_1 \in A_Z$ (and so far it is as in S-Generalized MAs), and $\pi(x_1, y_2)$ is true. The production will rewrite s , substituting the identified LHS with $\varphi(z_1, x_2, x_1) x_2 \varphi(z_1, x_1, x_1)$.

Probabilistic MAs (PMAs) [34] are CF-Generalized MAs where a production can have k right-hand sides, and when it fires, each RHS is chosen with $1/k$ probability (or with an assigned probability)—PMAs thus compute probability distributions.

Parallel Concurrent GMAs [39] are PMAs where one probabilistic production could be simultaneously applied to multiple occurrences of the LHS. Since these occurrences can be overlapping, some notions of incompatibility have to be developed.

Several other variants were sometimes introduced to address specific modeling issues, usually in an informal manner. We discuss some of these in the following section.

5.2 GMAs in Use

GMAs were introduced first in a technical note [57] of CSCE³³ (but certainly known to some researchers of the FLDL circle, see [63].) The paper also informally suggested further generalizations. In one, instead of specifying at definition time the order in which productions are searched for during the rewriting process, this order is defined *within* the transformation rules. In another, “higher-level types” of productions are allowed, so that one can simply express “a canonical occurrence of a word of” a certain class *C*, “occurring in a canonical occurrence of a word of” class *D*, and so on. Some more extensions and possible variations were discussed in a later paper [42].

Similar techniques for handling the semantics of programming languages were used, in those same years, by van Wijngaarden, who, however, did not explicitly refer to Markov in his papers discussing the design and definition of “generalized ALGOL” [90] and a possible formal definition of ALGOL 60 [91].³⁴ The ideas in those papers formed the basis of Jaco de Bakker’s monograph [62] which contained an almost complete definition of ALGOL 60 with more than 800 productions. The van Wijngaarden-de Bakker and Caracciolo approaches, although essentially analogous, differed in the way in which the applicability of a generalized production was determined [50]. For Caracciolo, an external, unspecified method was used to decide if a certain string belonged to one of the decidable languages represented by their indexed names. For van Wijngaarden and de Bakker, this checking was “internalized” in the form of additional productions. In de Bakker’s description, moreover, the list of productions dynamically grew during the rewriting process—“the execution of a particular ALGOL 60 program will lead to the extension of the list of language-defining rules with rules which reflect the meaning of this specific program” [63].

It should be evident how the features of GMAs met the demands Caracciolo expressed in his FLDL paper [38]. The metalinguistic variables of S-Generalized MAs allowed a grammatical description at a “natural level of abstraction”, without the unnecessary details that come with a concrete representation of strings, focusing instead on the properties that these strings have. The fact that it was left unspecified how the membership of a specific string in a syntactic category had to be checked made the approach flexible—sometimes it was convenient to give completely external algorithms, other times, following van Wijngaarden-de Bakker, more grammatical rules were added, as exemplified in [31]. On the other hand, to deal with more complex situations, CF-Generalized MAs allowed for generic programs to be “embedded” into a description. Finally, GMAs were, by their very nature, executable definitions, and thus automatically provided *recognizers and interpreters* for the language they described. This “double nature” of GMAs representations (from one perspective they define a language or system; from another they implement that language or simulate that system) is more apparent and important in their applications to general dynamical systems, which we discuss below.

One of the first applications of GMAs was in the definition of a family of *symbol processing languages*, which emerged as a spin-off from the research on languages for defining (and manipulating, as we have seen) general-purpose programming languages. The introduction of these languages, grouped under the name PANON [42, 56, 57, 88, 92], was motivated by the problem of the formal reduction of mathematical formulas—monomials, polynomials, expansion of powers and products, formal derivatives, and so on. In addition, PANON-2 dealt also with algebraic expressions with numerical coefficients, where some symbolic transformations may be applied only if certain numerical conditions on the coefficients were satisfied. PANON’s relation to the definition of the

³³A later, probably more available reference [35], gave an example of the versatility and precision that can be achieved with the approach—the GMA definition of a “conventional binary computer with 4096 memory location of 18 bits each, an instruction counter of 12 bits, an instruction register of 18 bits, and an accumulator of 18 bits”.

³⁴See [7] for an evaluation of these two van Wijngaarden’s papers from an epistemological perspective.

semantics of programming languages was explained [57] by the close relationships to “Syntax Directed Compilers” [69], because the elementary transformations of PANON were “structural transformations based on a syntactical analysis”.

In the organization of his research group, Caracciolo kept the definition of the formal tools, the general methodology and, above all, the epistemological reflections to himself, leaving the application of these tools and methodology to his collaborators, mainly students writing their master’s theses.³⁵ The formal semantic definitions were rarely published in international journals, and sometimes they were available only as technical reports in Italian.³⁶ They applied the GMA-based methodology to a variety of languages [50]: the definition of ALGOL 60 [53, 71], of SIMULA [53, 55], of PL/I [49], of a part of COBOL [72], and of the memory allocation of FORTRAN [1]. Special attention was paid to context-dependent constraints and the correct handling of dynamic memory allocation for nested blocks, a delicate issue dear to Caracciolo’s school since the early days [30].

It is important to recall what it meant “to give a definition of a language” in these papers. For ALGOL and SIMULA, for example, GMAs were given [53] that, when applied to a string, behaved in one of the following ways.

- If the string was a correct terminating program, then the algorithm generated an image of the value assumed by all variables declared in the outermost block when the program terminated.
- If the string was a correct non-terminating program, the algorithm generated an infinite sequence of transformations, as specified by the program.
- If the string did not represent a correct program in the given language, the algorithm terminated, indicating the presence of an error.
- If some undefined condition arose during the processing of the program, the algorithm stopped in a special state.

To obtain this result, the initial string (i.e., the text that should be a *program* in the given language, enclosed within the two extra symbols α and Ω) is preprocessed to standardize the text, adding information for its handling (e.g., in ALGOL each block is annotated with its nesting depth, any identifier is associated to the block nesting depth in which it is declared). After this initial setup (and if no errors occur, e.g. no identifier is left without a block in which it is declared, no identifier is declared twice in the same block, etc.), the beginning of the annotated program is marked with the “activation symbol” $\diamond$ and the execution phase begins, during which the activation symbol is moved through the program by the productions specifying the various syntactic constructs. Without giving too many details, let us just say that whenever a block is activated, a *local string* is created for handling the local declarations of variables of that block (names and, when assigned, their values). At the exit of the block, the local string is deleted. Procedure calls are handled by copying the body of the procedure, the names of the formal parameters, and its actual parameters, then marking all local identifiers with the information recording this specific call (this is necessary for handling recursive procedures, thus distinguishing local variables with the same name but introduced by calls at different recursive depth). Actual parameters are then substituted for each occurrence of the corresponding formal one (remember that ALGOL 60 adopts passing “by name” by default—the preprocessing phase had taken care of the possible use of the same name in different scopes, thus avoiding name capture) and $\diamond$ is moved to the beginning of the procedure body. When $\diamond$ reaches Ω (the end of the program string), some cleanup is applied, and the process terminates.

³⁵*Tesi di laurea*, in the Italian system.

³⁶See, for example, the long list of “unpublished” in the references in de Bakker’s survey [63].

As a simple example of the style of the definition, the following are some of the productions for the evaluation of the conditional statement in ALGOL [53, page 44]:

$$\diamond \text{ if } (\langle \text{boolean expression} \rangle) \rightarrow \text{if } (\langle \text{boolean expression} \rangle) (\text{if } \langle \text{boolean expression} \rangle \triangleright)$$

When the activation symbol reaches an **if**, the focus of the evaluation changes (via the $\langle \dots \rangle$ and $\langle \dots \triangleright$ parentheses) to the boolean expression. When this evaluation (via the productions for expressions) terminates with either **true** or **false**, one of the following productions will fire, removing the big $\langle \dots \rangle$ parentheses introduced by the production above, and moving the activation symbol in front of either the then-branch, or the else-branch, or the statement following the “**if**” if no else was present:

$$\begin{array}{ll} \langle \text{if true} \rangle \text{ then} & \rightarrow \text{then } \diamond \\ \langle \text{if false} \rangle \text{ then } \langle \text{uncond stmt} \rangle \text{ else} & \rightarrow \text{then } \langle \text{uncond stmt} \rangle \text{ else } \diamond \\ \langle \text{if false} \rangle \text{ then } \langle \text{uncond stmt} \rangle \langle \text{stmt end} \rangle & \rightarrow \text{then } \langle \text{uncond stmt} \rangle \diamond \langle \text{stmt end} \rangle \end{array}$$

The reader will have no difficulty in seeing how the use of syntactic categories in the productions helps to give an “understandable reading”³⁷ of them.

The most general result of this line of research was obtained by Luigia Carlucci, who in her Master’s thesis showed how GMAs modeled the behavior of the abstract machine introduced by the IBM Vienna Laboratory for the formal definition of PL/I.³⁸ To model the structures used in the **Vienna Definition Language (VDL)**, she introduced two further extensions of the GMA method: GMAs were nondeterministic and handled tree-structured objects, generating a linear representation of them. Via these results (published in a later paper [58]), definitions given by using VDL were translated into GMA ones.

We should not forget, however, that programming languages were to be understood for Caracciolo in their most general sense, as “communication tools” for the description of a job—“anything one (man or machine) has to do” [36]—a notion which “includes, as a special case, the concept of computation or algorithm, but also encompasses the description of concrete processes or operations, such as those that can be performed by machine tools, in the broadest sense of the term” [43]. It is on these broader subjects that he mostly applied GMAs during the second half of the decade, especially taking into account languages for the description of concrete, unending, and parallel jobs, following the categorization of his earlier paper [36]. He proposed the use of GMAs as the basis of a general theory of dynamical systems [39], suggesting that they could provide the core of *simulation languages*. To this goal, he had already introduced PMAs [34], that could be used to model parallel concurrent processes, introducing also both some sort of synchronization rule and a notion of incompatibility between actions.

These ideas formed the core of a series of contributions on two related topics [33]—“special languages” for the description (and thus the control, given the double nature of GMAs) of machine tools and chemical plants, and “simulation languages”. In the former, machine tools were seen as automata specified by GMAs. In [43, 45, 47], a certain algorithm for a cutter machine tool was given as a series of productions, specifying how the position and the movement of the tool changed over time. The “CF-” aspects of GMAs played a crucial role. Conditional application took care, among other aspects, of “prohibitions”, e.g. the interference between the non-cutting parts of the tool and the object to be modified. Arbitrary functions present in the RHS expressed here the result of the cutting action and the modification of the object, hence the essential action of the tool on the object.

³⁷We may even say “structured reading,” but it is clear that GMA productions, lacking the inductive nature of *rules*, are still far from the simplicity and elegance of *structured operational semantics* [85]. See also Section 6.

³⁸Carlucci was invited to present her results as an observer to the second meeting of IFIP WG 2.2 in Copenhagen in July 1968, where Böhm, Dahl, de Bakker, Gorn, Landin, Strachey, and van Wijngaarden, among other members, were present.

By using these primitive functions, a single rewriting corresponded to an elementary transition of the form

$$r(P, P', W) = W'$$

where W' is the shape resulting from a cutting action starting from the previous shape W , and P and P' are the state before and after the transition, respectively. By iterating the rewriting of the GMA, one obtained a global transformation

$$r^*(P_o, P^*, W_o) = W_f \quad (1)$$

which gave the final configuration W_f as a function of the initial state and configuration P_o and W_o , respectively, and of the program P^* . Now, in a familiar move, Caracciolo again took a “declarative” point of view and looked at (1) as an *equation in the unknown* P^* . To solve the equation, one first had to solve a mathematical optimization problem (which is of little relevance here) for finding the best way of obtaining the final shape of the object, given the initial one—this gave as a result P^* , formulated in mathematical terms. But then there was the linguistic problem of defining a “special programming language” for the actual, executable description of W_o , W_f , and especially P^* .

As in the definition of a programming language, once the program P^* was obtained and placed on the tape governing the machine tool, the appropriate GMA defining that machine marked the beginning of P^* with an activation symbol and concatenated the program with a second string representing the complete machine state. At this point the rewrite consumed the program, moving the activation symbol to the right and modifying the state of the machine. Calls to the primitive functions were interpreted as actual modifications of the object on which the machine operated. If the process terminated without error, the string was cleaned up and, in the physical world, the object was in the intended final shape.

In other contributions on the same research line [16, 41, 48, 51, 52] we rarely find the explicit presence of GMAs; this may be because some of the papers were intended to attract the attention of people coming from the application domains. Most of them presented at a very high level (flow-charts) how a certain machine tool (e.g., a milling cutter) could be modeled and guided, or how certain specific non-stationary processes in a chemical plant could be simulated, by using mathematical equations and reduction rules, where we recognise without difficulty the productions of GMAs. In a paper [70] by a chemist working under Caracciolo’s guidance, a high-level language for the description of chemical plants was given.

Probabilistic GMAs and Parallel Concurrent GMAs formed the theoretical basis for Caracciolo’s group’s interest in simulation languages [39, 54, 78, 79], especially for discrete event simulation of dynamical systems. In this context, their main interest—and the originality of their approach—laid in the treatment of “simultaneous” events, where the relation between (in)compatibility and commutability played an important role. They defined the language *SL/1* which was “in many ways derived from *SOL* and *SIMULA*” [54], although “an essential difference with respect [to them was] constituted by the method of analysis of simultaneous events, [...] the most critic [*sic*] point of simulation processes” [78], which in *SL/1* was strictly probabilistic and modeled after the treatment of scheduling, simultaneity, and incompatibility of events covered in the paper on PMAs.

Simulation languages were, for Caracciolo, the most comprehensive application of programming languages to the description of “jobs” because they dealt with systems that affect the physical world and not just the limited formal domain of standard computation. That his interest was not marginal and ephemeral is evidenced by the role he played in organizing the meetings at which these papers were presented. In May 1966, Caracciolo hosted in Pisa a workshop on “Linguaggi di simulazione” organized by the Italian Association for Automatic Computing, attended among others by John Buxton, who gave a general lecture on simulation languages and then presented his *CSL*, and by Ole-Johan Dahl who gave a detailed presentation of *SIMULA 1*. It is there that Giorgio

Molnar, one of Caracciolo's students, introduced *SL/1*. The proceedings recorded lively discussions, with all the participants contributing. Then, Caracciolo was on the organizing committee of the IFIP Working Conference on Simulation Programming Languages held in Oslo in May 1967 [17], where, among others, Dahl and Kristen Nygaard presented their ongoing work on classes and subclasses toward *SIMULA 67*, Steel described the work on the standardization of simulation languages, and Caracciolo discussed his methodology for the simulation of chemical plants [41]. Finally, in 1970, he chaired the program committee of the first IFIP/PROLAMAT Conference on Numerical Control Programming Languages, in Rome, where he presented one of his papers on machine tools [45].

6 Some Reflections, from a Contemporary Perspective

Following the research goals stated in the 1962 paper [30], the GMA definition of a programming language took into account all aspects of the language: context-free syntax, context-dependent constraints, and semantics as a state transformation. Any string (over the given alphabet) is manipulated by the defining GMA, either rejecting it if it is not a legal program, or giving its meaning as a state-transforming function. This should be contrasted with the approach that will become mainstream in both denotational and (structural) operational semantics, where it is *assumed* that you have a correct program (usually already expressed as an abstract syntax tree satisfying all contextual constraints) before you start semantically interpreting it. Another point worth noting is that GMAs also gave semantics to non-terminating programs, an important aspect for Caracciolo who observed that the notion of “value at time t ” (even if t is not final) is a “really basic concept” [36]. Finally, a GMA for a language was both a definition and an interpreter for that language. It is for the presence of these aspects altogether that Caracciolo insisted that GMAs give a true, complete declarative definition of a programming language.

Caracciolo's group never fully developed his remarks that one of the purposes of a semantic definition, and of GMA definitions in particular, is to derive tools for analyzing, executing, and translating programs. Nevertheless, this idea remained fruitful for years to come, particularly in the context of more denotational approaches. One of the first and most accomplished treatments is Peter Mosses' work, which provided a system that “produces a compiler for a programming language from a formal description consisting of an unambiguous context-free grammar and mathematical semantics” [81].

As we hinted at in footnote 37, Caracciolo was one step away from providing an operational description in the style of Gordon Plotkin's SOS [85]. The purpose of using grammatical categories in GMAs was to group rewriting rules according to different categories, and, more importantly, to allow manipulation of abstract structures rather than their string representations, enabling computation at a higher level of abstraction, in a manner similar to SOS. Although transition systems are more general, they are not substantially different from the productions of a Markov algorithm. It is the inductive nature of the *rules*—the most important innovation of SOS over the “usual tedious syntactic definition” [86]—that is absent from Caracciolo's approach. Perhaps someone younger, who was less beholden to tradition, was needed for this.

Finally, for today's reader it is both puzzling and interesting that Caracciolo never referred to artificial intelligence in his writing, despite being fully aware of the contributions of John McCarthy to symbolic manipulation and of Marvin Minsky's work on robotic arms.³⁹ Yet, GMAs and PANON (his programming language for text manipulation) were the forerunners of the rule-based languages that would be at the heart of symbolic AI in the 1980s. For PANON, an explicit inspiration [56, 57] came from Newell, Shaw and Simon's **General Problem Solver** [82] (GPS), since for both PANON and GPS “structural transformation rules” are central. Caracciolo's students recall his insistence on

³⁹Luigia Carlucci Aiello, personal communication.

their learning IPL, the implementation language of GPS, around 1967. Even more suggestive is his scientific production on machine tools, which reveals his interest in the field of robotics.

7 Changing Interests

By the end of the 1960s, Alfonso Caracciolo was at the peak of his scientific career, playing a prominent international role in the organization and politics of computer science. He was a member of the Italian CNR's Informatics Commission and tried to create a European Software Institute within the European Community with the major European computer manufacturers of the time. In 1968, in one of the meetings for this initiative, he suggested to Maurice Nivat and Marcel-Paul Schützenberger the creation of a European entity dealing with the theoretical aspects of Computer Science [83], which happened in 1972, upon the initiative of Nivat, with a meeting in Brussels that Caracciolo chaired [9]. He was appointed to represent Italy (1972–1975) in the European Committee for Research and Development (CERD-Comitato Europeo Ricerca e Sviluppo), and he chaired the Special Group for “Computer Science Education” established by the Commission of European Communities inside the Scientific and Technical Research Committee (CREST).

In 1972 he accepted a professorship in “Teoria e tecnica della programmazione”⁴⁰ at the University of L'Aquila. He moved his family to Rome, abandoning research on formal methods while retaining some interest in the social implications of computer science, which he had cultivated during his time in Pisa [40]. His last publication in the field of formal methods was a review on programming languages [44], followed by a brief paper on “special languages” [46] in 1970. He stopped attending IFIP TC-2 and WG 2.2 meetings. He began a collaboration with the private LUISS University⁴¹ in Rome, where he was finally appointed professor of “Teoria e sistemi della programmazione”⁴² in 1976. He remained with LUISS until his death in 1996.

One might wonder why he decided to leave Pisa and abandon his interest in formal methods, and why his research team there also disbanded.⁴³ Academic politics in Pisa undoubtedly played the most important role.⁴⁴ In 1969, when the Italian Parliament established the first Italian degree in Computer Science in Pisa, the University had to organize and staff the newly created program. If the area of “systems” was covered by Giovan Battista Gerace—the head of the Engineering Section of CSCE—and his group, the same was not the case for the area of “programming languages and software”. Caracciolo was not the typical academic, the one always active in creating opportunities for his students and collaborators, even taking on local organizational responsibilities. Moreover, the Prince of Forino and Duke of Belcastro never got on well with Gerace, who became a member of the Central Committee of the Italian Communist Party in 1979. The university and CNR thus recruited several young researchers from the Politecnico di Milano for junior positions.⁴⁵ Over time, these researchers went on to establish *other* strong schools of formal methods and, more

⁴⁰Programming theory and techniques.

⁴¹LUISS—Libera Università Internazionale degli Studi Sociali—was born in 1976 from the transformation of an earlier private higher education institution—the Università di Studi Sociali *Pro Deo*—after a group of Italian entrepreneurs had taken over the property from the Catholic *Pro Deo* in 1974.

⁴²Programming theory and systems.

⁴³One of the very few papers on GMAs published after Caracciolo's departure introduced another variant of Markov Normal Algorithms [2], applying it to “formally define the input mechanism for formatted records in FORTRAN”. Some of Caracciolo's former students later studied how MAs could be used to define classes of formal languages [73], analogously to what was done for Chomsky grammars or automata (e.g., regular grammars characterize regular languages). It is interesting to observe that the paper cited de Bakker's survey [63] but did not mention Caracciolo or his papers.

⁴⁴Although we cannot speculate on his personal or family motivations, his wife's work as a jewellery designer and their desire to raise their daughters in a big city may have influenced his decision to leave.

⁴⁵Among the others, Mario Aiello, Antonio Albano, Antonio Grasselli, Giorgio Levi, Fabrizio Luccio, Alberto Martelli, Ugo Montanari, Carlo Montanero, Franco Sirovich.

generally, theoretical computer science. To secure their careers, Caracciolo's collaborators and former students took on teaching roles in the new degree, changed their research interests, and some of them moved abroad. At the new Institute of Computer Science, the influence of Gerace and Grasselli (the most senior of those recruited from Milan) was unchallenged, even for a scientist of Caracciolo's international standing. In 1972, the university finally established three full professorships in Computer Science. Gerace and Grasselli were appointed to the positions of "Sistemi per l'elaborazione dell'informazione",⁴⁶ while Franco Preparata⁴⁷ was chosen over Caracciolo for the position of "Teoria ed applicazioni delle macchine calcolatrici".

All this led to Caracciolo's dissatisfaction with that research environment and his decision to leave Pisa and end his scientific production on formal methods. The pioneering application of GMAs to the definition of programming languages and the simulation of dynamical systems came to an end.

Acknowledgments

I would like to thank Giorgio Ausiello, Luigia Carlucci Aiello, Pierpaolo Degano, and Fabio Gadducci for our discussions and their helpful feedback; Cliff Jones for searching his archive for me; Troy Astarte for providing material from the van Wijngaarden archive; Daniele Ronco for searching the archives of the University of Pisa and scanning all those pages; and Joanne Allison for improving my English. Finally, I would like to thank Isabella Caracciolo di Forino, who has sadly passed away in 2024, for sharing documents from her family archive and answering my many questions about her father.

References

- [1] Gianni Aguzzi and Renzo Pinzani. 1968. *On a Formalization of a Storage Allocation Mechanism for Fortran*. Technical Report CSCE-NI-70-1968 (seconda serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remotedoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1968-B4-006&langver=en>
- [2] Gianni Aguzzi, Renzo Pinzani, and Renzo Sprugnoli. 1972. An algorithmic approach to the semantics of programming languages. In *Automata, Languages and Programming, Colloquium, Paris, France, July 3-7, 1972*. Maurice Nivat (Ed.), North-Holland, Amsterdam, 147–166.
- [3] Patrizia Andronico and Claudia Raviolo. 2013. Il CNR dopo la CEP: La scuola di specializzazione. In *La CEP prima della CEP*. G. Cignoni and F. Gadducci (Eds.), Pisa University Press (Quaderni della Fondazione Galileo Galilei), Pisa, 45–46. Retrieved from <https://www.pisauniversitypress.it/scheda-libro/autori-vari/la-cep-prima-della-cep-9788867413034-155663.html>
- [4] Enzo Aparo and Dino Dainelli. 1953. *Esperienze sull'impiego di alcune calcolatrici elettroniche*. Technical Report 387. Istituto per le Applicazioni del Calcolo; - CNR.
- [5] Enzo Aparo and Dino Dainelli. 1954. *Considerazioni sulle piu recenti machine calcolatrici elettroniche ed il loro impiego*. Technical Report 409. Istituto per le Applicazioni del Calcolo; - CNR.
- [6] Troy Kaighin Astarte. 2019. *Formalising Meaning: A History of Programming Language Semantics*. Ph.D. Dissertation. School of Computing, Newcastle University. Retrieved from http://homepages.cs.ncl.ac.uk/troy.astarte/res/pdf/TK_Astarte_Formalising_Meaning_2019.pdf
- [7] Troy Kaighin Astarte. 2025. Conceptualising programming language semantics. *Philosophies* 10, 4 (2025), 22. DOI: <https://doi.org/10.3390/philosophies10040090>
- [8] Giorgio Ausiello. 2018. *The Making of a New Science. A Personal Journey Through the Early Years of Theoretical Computer Science*. Springer, Cham. DOI: <https://doi.org/10.1007/978-3-319-62680-2>

⁴⁶Information processing systems. Gerace remained at the University of Pisa until his death, in 1987. Grasselli first took a research leave to Grenoble from December 1972 to November 1973; he then moved to the Politecnico di Milano in November 1974.

⁴⁷Franco P. Preparata was born in 1935 and graduated in Electrical Engineering from the University of Rome in 1959. From 1970, he was a full professor of Computer Science at the University of Illinois at Urbana. He remained at the University of Pisa for three academic years before taking research leave in November 1975 to return to Urbana. He resigned from his position at Pisa in January 1976.

- [9] Giorgio Ausiello and Giorgio Gambosi. 2021. Theoretical computer science in Italy: The early years. *IEEE Ann. Hist. Comput.* 43, 3 (2021), 44–56. DOI: <https://doi.org/10.1109/MAHC.2021.3099878>
- [10] John W. Backus. 1959. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference. In *Information Processing, Proceedings of the 1st International Conference on Information Processing*. UNESCO, Paris, 125–131. Retrieved from https://www.softwarepreservation.org/projects/ALGOL/paper/Backus-Syntax_and_Semantics_of_Proposed_IAL.pdf
- [11] John W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, et al. 1960. Report on the algorithmic language ALGOL 60. *Commun. ACM* 3, 5 (1960), 299–314. DOI: <https://doi.org/10.1145/367236.367262>
- [12] N. M. Blachman. 1953. *A Survey of Automatic Digital Computers*. Technical Report. Office of Naval Research, Department of the Navy, Washington, DC, USA. Retrieved from <https://archive.computerhistory.org/resources/access/text/2024/07/102686252-05-01-acc.pdf>
- [13] Daniel G. Bobrow (Ed.). 1968. *Symbol Manipulation Languages and Techniques*. North-Holland, Amsterdam. Proceedings of the IFIP working conference on symbol manipulation languages, Pisa 1966.
- [14] Ariela Böhm, Emanuele Böhm, Michele Böhm, Mariangiola Dezani-Ciancaglini, Francesca Manfredini, and Nora Perugia Böhm. 2020. Corrado Böhm: The lambda-adventure. In *Contemporary Logic and Computing*. Adrian Rezus (Ed.), College Publications, London, 399–414. Retrieved from <http://www.di.unito.it/~dezani/papers/Bohm20.pdf>
- [15] Corrado Bonfanti. 2012. Information technology in Italy: The origins and the early years (1954 - 1965). In *Reflections on the History of Computing: Preserving Memories and Sharing Stories*. Arthur Tatnall (Ed.), Springer, Berlin, 320–347. DOI: https://doi.org/10.1007/978-3-642-33899-1_19
- [16] Alessandro Brambilla, Giuliano Nardini, Alfonso Caracciolo di Forino, Romano Celati, and Juliusz Kardasz. 1968. Ricerche sul linguaggio per la simulazione di impianti chimici. *Calcolo* 5, 1 (1968), 237–248. DOI: <https://doi.org/10.1007/BF02576069>
- [17] John N. Buxton (Ed.). 1968. *Simulation Programming Languages. Proceedings of the IFIP Working Conference on Simulation Programming Languages (Oslo, May 1967)*. North-Holland, Amsterdam.
- [18] Eduardo R. Caianiello (Ed.). 1966. *Automata Theory*. Academic Press, New York, London. Papers of the Sixth International Spring School of Physics, Ravello, June 1964.
- [19] Alfonso Caracciolo. 1952. Sulla arbitrarietà delle logica dei sistemi formali. In *Atti del Congresso di Studi Metodologici*. Taylor, Torino, 148–154.
- [20] Alfonso Caracciolo. 1954. *Rapporto sulle moderne calcolatrici elettroniche*. Technical Report NI-28-1954 (prima serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1954-B4-001&langver=en>
- [21] Alfonso Caracciolo. 1958. *The Computer of the Centro Studi Calcolatrici Elettroniche of the University of Pisa*. Technical Report CSCE-P-2-1958 (prima serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1958-A2-001&langver=en>
- [22] Alfonso Caracciolo. 1961. Sulla definizione delle funzioni di selezione. In *Atti Del Convegno Nazionale di Logica*. Levrotto & Bella, Torino, 127–136. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1961-B2-002&langver=en>
- [23] Alfonso Caracciolo. 1966. Some preliminary remarks on theoretical pragmatics. *Commun. ACM* 9, 3 (Mar 1966), 226–227. DOI: <https://doi.org/10.1145/365230.365273>
- [24] Alfonso Caracciolo, Giuseppe Cecchini, Elio Fabri, and Sergio Sibani. 1956. *Progetto dettagliato di una prima calcolatrice elettronica (macchina ridotta)*. Technical Report CSCE-NI-26-1956 (prima serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1956-B4-008&langver=en>
- [25] Alfonso Caracciolo and Elio Fabri. 1955. *Caratteristiche fondamentali del progetto di massima logico-strutturale della calcolatrice elettronica*. Technical Report CSCE-NI-1-1955 (prima serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1955-B4-001&langver=en>
- [26] Alfonso Caracciolo and Elio Fabri. 1955. *Presentazione e discussione della lista e del codice delle istruzioni*. Technical Report CSCE-NI-2-1956 (prima serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1956-B4-001&langver=en>
- [27] Alfonso Caracciolo and Elio Fabri. 1957. *Complementi e variazioni al progetto logico dettagliato della macchina ridotta*. Technical Report CSCE-NI-36-1957 (prima serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1957-B4-003&langver=en>
- [28] Alfonso Caracciolo di Forino. 1953. Sur la construction du langage de la physique. In *Actes du XIème Congrès International de Philosophie – Philosophie et méthodologie des Sciences de la Nature*, Vol. VI. North-Holland, Amsterdam, 113–118.
- [29] Alfonso Caracciolo di Forino. 1958. *A Complete System of Instruction Modification which Simplifies Programming*. Technical Report CSCE-NI-40-1958 (prima serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1958-B4-003&langver=en>

- [30] Alfonso Caracciolo di Forino. 1963. On a research project in the field of languages for process construction. In *Proceedings of the IFIP Symposium on Languages for Processor Construction, Munich 1962*. North Holland, Amsterdam, 514–515. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1963-A2-001&langver=en>
- [31] Alfonso Caracciolo di Forino. 1963. Some remarks on the syntax of symbolic programming languages. *Commun. ACM* 6, 8 (Aug 1963), 456–460. DOI : <https://doi.org/10.1145/366707.367584>
- [32] Alfonso Caracciolo di Forino. 1964. *N*-ary selection functions and formal selective systems. Part 1. *Calcolo* 1, 1 (1964), 49–82. DOI : <https://doi.org/10.1007/BF02576888>
- [33] Alfonso Caracciolo di Forino. 1965. Linguaggi programmativi speciali. *Calcolo* 2, Supplemento n. 2 (1965), 69–89. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1965-B0-003&langver=en>
- [34] Alfonso Caracciolo di Forino. 1966. *An Elementary Constructive Theory of Discrete Probabilistic Processes and Parallel Concurrent Processes (Presented at "Wiener Memory Day", Genova)*. Technical Report CSCE-P-62-1966 (seconda serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1966-IS-002&langver=en>
- [35] Alfonso Caracciolo di Forino. 1966. Generalized Markov algorithms and automata. In *Automata Theory*. Eduardo Caianiello (Ed.), Academic Press, New York and London, 115–130. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1966-A0-002&langver=en>
- [36] Alfonso Caracciolo di Forino. 1966. Linguistic problems in programming theory. In *Information Processing 1965. Proceedings of IFIP Congress 65*. W.A. Kalenich (Ed.), Spartan Books, Washington, DC, 223 – 228. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1966-A2-002&langver=en>
- [37] Alfonso Caracciolo di Forino. 1966. *m*-valued logics and *m*-ary selection functions. In *Automata Theory*. Eduardo Caianiello (Ed.), Academic Press, New York and London, 107–114. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1964-B4-003&langver=en>
- [38] Alfonso Caracciolo di Forino. 1966. On the concept of formal linguistic systems. In *Formal Language Description Languages for Computer Programming*. T.B. Jr. Steel (Ed.), North-Holland, Amsterdam, 37–51. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1966-A1-001&langver=en>
- [39] Alfonso Caracciolo di Forino. 1966. Some theoretical considerations about simulation languages. *Calcolo* 3, Suppl. 1 (1966), 245–265. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1966-B0-002&langver=en>
- [40] Alfonso Caracciolo di Forino. 1968. La formazione del personale di fronte allo sviluppo degli elaboratori elettronici. *Calcolo* 5, Suppl. 1 (1968), 69–73. DOI : <https://doi.org/10.1007/BF02576052>
- [41] Alfonso Caracciolo di Forino. 1968. On a research project on a high level simulation language for chemical plants. In *Simulation Programming Languages*. J. N. Buxton (Ed.), North-Holland, Amsterdam, 446–450.
- [42] Alfonso Caracciolo di Forino. 1968. String processing languages and generalized Markov algorithms. In *Symbol Manipulation Languages and Techniques*. D.G. Bobrow (Ed.), North-Holland, Amsterdam, 191–206. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1966-IS-001&langver=en>
- [43] Alfonso Caracciolo di Forino. 1968. Sulla programmazione delle macchine utensili. *Calcolo* 5, Suppl. 1 (1968), 283–295. DOI : <https://doi.org/10.1007/BF02576073>
- [44] Alfonso Caracciolo di Forino. 1969. Programming languages. In *Advances in Information Systems Science: Volume 1*. Julius T. Tou (Ed.), Springer US, Boston, MA, 59–116. DOI : https://doi.org/10.1007/978-1-4615-9050-7_2
- [45] Alfonso Caracciolo di Forino. 1970. The formal definition of machine tool languages. In *Numerical Control Programming Languages - Proceeding of the 1st Intl. IFIP/PROLAMAT Conference, Rome*. W.H.P. Leslie (Ed.), North-Holland, Amsterdam, 315–331.
- [46] Alfonso Caracciolo di Forino. 1970. I linguaggi speciali di programmazione come strumenti avanzati di pensiero. In *Linguaggi Nella Società e Nella Tecnica*. Edizioni di Comunità, Milano, 401–402.
- [47] Alfonso Caracciolo di Forino and Antonio Camera. 1968. *On a Formal Definition of Direct Machine Tool Languages*. Technical Report 1. IEI-CNR, Pisa.
- [48] Alfonso Caracciolo di Forino, Antonio Camera, and Stefano Trumphy. 1970. *Languages for Automatic Programming Tools*. Technical Report B70-23. IEI-CNR, Pisa.
- [49] Alfonso Caracciolo di Forino and Luigia Carlucci. 1969. *On an Algorithmic Interpretation of the Formal Definition of PL/1*. Technical Report B69-4. IEI-CNR, Pisa.
- [50] Alfonso Caracciolo di Forino, Luigia Carlucci, Gualtiero Leoni, Rina Rebaudo, Gianni Aguzzi, and Roberto Pinzani. 1968. Sulla definizione formale dei linguaggi programmativi. *Calcolo* 5, Suppl. 1 (1968), 296–303. DOI : <https://doi.org/10.1007/BF02576074>
- [51] Alfonso Caracciolo di Forino, Romano Celati, and Giuliano Nardini. 1967. *Sulla ricerca di un linguaggio programmatico per la simulazione degli impianti chimici*. Technical Report CSCE-P-122-1967 (seconda serie). CSCE, Pisa.

- Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1967-IS-005&langver=en> Presented at the AICA working group on “Linguaggi Programmatici”, 30 November 1967.
- [52] Alfonso Caracciolo di Forino, Romano Celati, Giuliano Nardini, and P. Pratali. 1968. Contributo alla ricerca di un linguaggio di simulazione per impianti chimici. *Quaderni Dell'Ingegnere Chimico Italiano* 4 (Maggio 1968), 81–87.
 - [53] Alfonso Caracciolo di Forino, Gualtiero Leoni, and Rina Rebaudo. 1969. *On the Formal Description of ALGOL/60 and SIMULA/1 Semantics by Means of a Generalized Markov Algorithm*. Technical Report B69-10. IEI-CNR, Pisa.
 - [54] Alfonso Caracciolo di Forino and Giorgio Molnar. 1969. *Sistemi dinamici e linguaggi di simulazione*. Technical Report CSCE-P-63-1969 (seconda serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1969-IS-001&langver=en> Presented at the AICA conference on “Metodi di simulazione”, Venice.
 - [55] Alfonso Caracciolo di Forino and Rina Rebaudo. 1968. On a Formalization of SIMULA by Means of a Generalized Markov Algorithm. Technical Report. IEI-CNR, Pisa.
 - [56] Alfonso Caracciolo di Forino, Luigino Spanedda, and Nicola Wolkenstein. 1966. PANON-1B: A programming language for symbol manipulation. In *Proceedings of the 1st ACM Symposium on Symbolic and Algebraic Manipulation (SYMSAC '66)*. ACM, New York, NY, USA, 0601–0615. DOI: <https://doi.org/10.1145/800005.807956> Also in *Calcolo*, vol 3(2), 245–265.
 - [57] Alfonso Caracciolo di Forino and Nicola Wolkenstein. 1963. *On a Class of Programming Languages for Symbol Manipulation Based on Extended Markov Algorithms*. Technical Report CSCE-P-21-1963 (seconda serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1963-IS-004&langver=en>
 - [58] Luigia Carlucci. 1971. A conditional functional tree-handling generalized Markov algorithm which interprets a nondeterministic abstract machine. *Calcolo* 8, 3 (1971), 161–183. DOI: <https://doi.org/10.1007/BF02575513>
 - [59] Giovanni A. Cignoni and Fabio Gadducci. 2012. Rediscovering the very first Italian digital computer. In *Proceedings of the 2012 3rd IEEE HISTory of ELECTro-technology CONference (HISTELCON)*. IEEE, Pavia, 1–6. DOI: <https://doi.org/10.1109/HISTELCON.2012.6487563>
 - [60] Giovanni A. Cignoni and Fabio Gadducci. 2013. A syllabus for the fifties. teaching computer science on the first Italian computers. In *HaPoC 2013 : Proceedings of the 2nd International Conference on the History and Philosophy of Computing*. ENS, Paris, 27–29. Retrieved from http://hmr.di.unipi.it/Documenti/HMR_2013s_GCFG-HaPoC-ab.pdf
 - [61] Giovanni A. Cignoni and Fabio Gadducci. 2020. Pisa, 1954-1961: Assessing key stages of a seminal italian project. *IEEE Annals of the History of Computing* 42, 2 (2020), 6–19. DOI: <https://doi.org/10.1109/MAHC.2020.2978162>
 - [62] Jaco W. de Bakker. 1967. *Formal Definition of Programming Languages : With an Application to the Definition of Algol 60*. Number 16 in Mathematical Centre tracts. Amsterdam : Mathematisch centrum, Amsterdam. Retrieved from <https://ir.cwi.nl/pub/13018/>
 - [63] Jaco W. de Bakker. 1969. Semantics of programming languages. In *Advances in Information Systems Science: Volume 2*. Julius T. Tou (Ed.), Springer US, Boston, MA, 173–227. DOI: https://doi.org/10.1007/978-1-4899-5841-9_3
 - [64] Bruno de Finetti. 1952. Macchine che pensano (e che fanno pensare). *Tecnica ed Organizzazione* Anno III, 3-4 (1952), 2–34. Retrieved from https://www.butterfly.eu/islandora/object/librib:43084/datastream/PDF/content/librib_43084.pdf
 - [65] Bruno de Finetti, Giulio Rodinò, and Norman Kitz. 1953. Symposium on automatic digital computation. *La Ricerca Scientifica* 23, 7 (1953), 14.
 - [66] Giuseppe De Marco, Giovanni Mainetto, Serena Pisani, and Pasquale Savino. 1999. The early computers of Italy. *IEEE Ann. Hist. Comput.* 21, 4 (1999), 28–36. DOI: <https://doi.org/10.1109/85.801530>
 - [67] Elio Fabri. 1958. *Appunti delle lezioni di “Introduzione alla programmazione di una calcolatrice elettronica”*. Technical Report 35. CSCE. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1957-B4-004&langver=en>
 - [68] Saul Gorn. 1961. Some basic terminology connected with mechanical languages and their processors: A tentative base terminology presented to ASA x3.4 as a proposal for subsequent inclusion in a glossary. *Commun. ACM* 4, 8 (1961), 336–339. DOI: <https://doi.org/10.1145/366678.366682>
 - [69] Edgar T. Irons. 1961. A syntax directed compiler for ALGOL 60. *Commun. ACM* 4, 1 (Jan. 1961), 51–55. DOI: <https://doi.org/10.1145/366062.366083>
 - [70] Juliusz Kardasz. 1967. *On Language for Dynamic Description of Continuous Systems*. Technical Report CSCE-NI-67-1967 (seconda serie). CSCE, Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1967-B4-025&langver=en>
 - [71] Gualtiero Leoni. 1967. *Una completa definizione del linguaggio programmatico ALGOL-60 per mezzo di un algoritmo generalizzato di markov*. Master’s thesis. Università di Pisa. Retrieved from <http://pumax.isti.cnr.it/remote/linkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1967-B4-012&langver=en>
 - [72] Gualtiero Leoni. 1968. *On Formal Definition of COBOL Semantics*. Technical Report. IEI-CNR, Pisa.
 - [73] Gualtiero Leoni and Renzo Sprugoli. 1977. Some relations between Markov algorithms and formal languages. *CALCOLO* 14, 3 (1977), 261–284. DOI: <https://doi.org/10.1007/BF02576816>

- [74] Andrei Andreevich Markov Jr. 1954. *The Theory of Algorithms* (in Russian). Trudy Mat. Inst. Steklov., Vol. 42. Acad. Sci. USSR, Moscow–Leningrad. English translation 1961, published for the National Science Foundation and the Department of Commerce, by the Israeli Program for Scientific Translations.
- [75] John McCarthy. 1960. Recursive functions of symbolic expressions and their computation by machine, part I. *Commun. ACM* 3, 4 (1960), 184–195. DOI: <https://doi.org/10.1145/367177.367199>
- [76] John McCarthy. 1963. A basis for a mathematical theory of computation. In *Computer Programming and Formal Systems*. P. Braffort and D. Hirschberg (Eds.), Studies in Logic and the Foundations of Mathematics, Vol. 35, Elsevier, Amsterdam, 33 – 70. Retrieved from <https://www-formal.stanford.edu/jmc/basis1.pdf> A preliminary version presented at the Western Joint IRE-AIEE-ACM 1961 Computer Conference, pp. 225–238. ACM, New York, NY, USA (1961).
- [77] John McCarthy. 1963. Towards a mathematical science of computation. In *IFIP Congress 1962*. C.M. Poppolewell (Ed.), North Holland, Amsterdam, 21–28. Retrieved from <https://www-formal.stanford.edu/jmc/towards.pdf>
- [78] Giorgio Molnar. 1966. Simulation language SL/I. *Calcolo* 3, Suppl. 1 (1966), 71–94. Retrieved from <http://pumax.isti.cnr.it/remotelinkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1967-B0-001&langver=en>
- [79] Giorgio Molnar. 1968. Self optimizing simulation. In *Simulation Programming Languages*. J. N. Buxton (Ed.), North Holland, Amsterdam, 258–265. Retrieved from <http://pumax.isti.cnr.it/remotelinkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1968-A1-001&langver=en>
- [80] Edward F. Moore. 1956. Gedanken-experiments on sequential machines. In *Automata Studies. (AM-34), Volume 34*. C. E. Shannon and J. McCarthy (Eds.), Princeton University Press, Princeton, 129–154. DOI: <https://doi.org/10.1515/9781400882618-006>
- [81] Peter Mosses. 1975. *Mathematical Semantics and Compiler Generation*. Ph.D. Dissertation. University of Oxford. Retrieved from <https://ora.ox.ac.uk/objects/uuid:b590173b-0a86-40c0-8e75-6a6fc5035c43>
- [82] Allen Newell, J. Clifford Shaw, and Herbert A. Simon. 1959. Report on a general problem-solving program. In *Proceedings of the 1st International Conference on Information Processing, 15-20 June 1959*. UNESCO, Paris, 256–264. Retrieved from https://iiif.library.cmu.edu/file/Newell_box00005_fld00314_doc0003/Newell_box00005_fld00314_doc0003.pdf
- [83] Maurice Nivat. 2015. The true story of TCS. *Theor. Comput. Sci.* 591 (2015), 1–2. DOI: <https://doi.org/10.1016/j.tcs.2015.05.022>
- [84] Alan J. Perlis and Klaus Samelson. 1958. Preliminary report: International algebraic language. *Commun. ACM* 1, 12 (1958), 8–22. DOI: <https://doi.org/10.1145/377924.594925>
- [85] Gordon D. Plotkin. 1981. *A Structural Approach to Operational Semantics*. Technical Report DAIMI FN-19. Computer Science Department, Aarhus University. Retrieved from https://homepages.inf.ed.ac.uk/gdp/publications/sos_jlap.pdf
- [86] Gordon D. Plotkin. 2004. The origins of structural operational semantics. *J. Log. Algebr. Program.* 60-61 (2004), 3–15. DOI: <https://doi.org/10.1016/j.jlap.2004.03.009>
- [87] Dana Scott. 1970. *Outline of a Mathematical Theory of Computation*. Technical Report PRG-2. Programming Research Group, University of Oxford. Retrieved from <https://www.cs.ox.ac.uk/files/3222/PRG02.pdf> Also in the Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems, pp 169–76.
- [88] Luigino Spanedda. 1968. Synrec: L’analizzatore sintattico per l’interprete PANON-1B. *Calcolo* 5, 2 (1968), 1132–1146. DOI: <https://doi.org/10.1007/BF02576623>
- [89] Thomas B. Jr. Steel (Ed.). 1966. *Formal Language Description Languages for Computer Programming*. North-Holland, Amsterdam. Proceedings of the IFIP Conference, Vienna, September 15-18, 1964.
- [90] Adriaan van Wijngaarden. 1963. Generalized ALGOL. *Annu. Rev. Autom. Program.* 3 (1963), 17–26. DOI: [https://doi.org/10.1016/S0066-4138\(63\)80002-6](https://doi.org/10.1016/S0066-4138(63)80002-6)
- [91] Adriaan van Wijngaarden. 1966. Recursive definition of syntax and semantics. In *Formal Language Description Languages for Computer Programming*. T.B. Jr. Steel (Ed.), North-Holland, Amsterdam, 13–24. Retrieved from <https://ir.cwi.nl/pub/9205>
- [92] Nicola Wolkenstein. 1967. Some remarks on the implementation of PANON-1B. *ACM SICSAM Bull.* 7 (1967), 40–45. Retrieved from <http://pumax.isti.cnr.it/remotelinkdoc.php?authority=cnr.isti&collection=cnr.csce&idcode=1967-B4-009&langver=it>

Received 28 May 2025; revised 12 September 2025; accepted 30 September 2025