



# Topics in Programming Languages, a Philosophical Analysis through the case of Prolog



VNiVERSiDAD  
D SALAMANCA

Luís Homem  
Universidad de Salamanca  
Facultad de Filosofía

A thesis submitted for the degree of  
*Doctor en Lógica y Filosofía de la Ciencia*

Salamanca 2018

This thesis is dedicated to  
family  
and friends

## Acknowledgements

I am very grateful for having had the opportunity to attend classes with all the *Epimenides* Program Professors: Dr.<sup>o</sup> Alejandro Sobrino, Dr.<sup>o</sup> Alfredo Burrieza, Dr.<sup>o</sup> Ángel Nepomuceno, Dr.<sup>a</sup> Concepción Martínez, Dr.<sup>o</sup> Enrique Alonso, Dr.<sup>o</sup> Huberto Marraud, Dr.<sup>a</sup> María Manzano, Dr.<sup>o</sup> José Miguel Sagüillo, and Dr.<sup>o</sup> Juan Luis Barba. I would like to extend my sincere thanks and congratulations to the Academic Commission of the Program. A very special gratitude goes to Dr.<sup>a</sup> María Manzano-Arjona for her patience with the troubles of a candidate without a scholarship, or any funding for the work, and also to Dr.<sup>o</sup> Fernando Soler-Toscano, for his quick and sharp amendments, corrections and suggestions. Lastly, I cannot but offer my heartfelt thanks to all the members and collaborators of the Center for Philosophy of Sciences of the University of Lisbon (CFCUL), specially Dr.<sup>a</sup> Olga Pombo, who invited me to be an integrated member in 2011.

## Abstract

Programming Languages seldom find proper anchorage in philosophy of logic, language and science. What is more, philosophy of language seems to be restricted to natural languages and linguistics, and even philosophy of logic is rarely framed into programming language topics. Natural languages history is intrinsically *acoustics-to-visual*, *phonetics-to-writing*, whereas computing programming languages, under man-machine interaction, aspire to *visual-to-acoustics*, *writing-to-phonetics* instead, namely through natural language processing. One such programming language as Prolog has the peculiar attribute of having been unfolded in the passage from grammar studies, linguistics and formal symbolic grammar studies to the very specific and incipient computational enterprises of natural language processing and logic programming. Recounting Prolog's philosophical, mechanical and algorithmic harbingers endorse us to the mechanical-computational background explored by Pascal, Leibniz, Boole, Jacquard, Babbage, Zuse, until reaching to the ACE (Alan Turing) and the EDVAC (von Neumann). Recursion theory and computability (Turing, Church, Gödel, Kleene), cryptography and information theory (Shannon), permit us to interpret ahead the evolving realm of programming languages. The line from  $\lambda$ -Calculus to Lisp, and from Lisp to the Algol-family programming languages, the procedural and declarative split with the C language and Prolog, and the ensuing syntax and semantic branching explosion are here investigated in relation with the original élan of both C and Prolog. Philosophy of mathematics different approaches – structuralist, formalist, logicist, intuitionist – are confronted with computability, and furthermore, logic programming and Prolog, studying

closely such authors as Frege, Gödel, and Wittgenstein. The logicist, first-order complete and atomist nature of logic programming, settled on Horn clauses with SLD-resolution, is subsequently examined. Likewise, the path from linguistic semiotics and the linguistics turn from the Saussurian *Course Générale* to the Chomsky-Schützenberger hierarchy, with a clear glance towards automata and complexity theories, all in all with a common background on the ancient Pāṇini grammar, drives the research far wider than the use of the Backus-Naur form in Saussurian semiotics and programming languages, yet forgotten about. It permits to describe Prolog as a typical by-pass product of computational logicism and linguistics structuralism, with the philosophical design of the Chomsky-Schützenberger hierarchy, specially apt to (strong) AI illusion, as seen by its input into the Fifth Generation Computer Systems (FGCS) in Japan. It also brings about how the syntax-structuralism and the phonological destitution in natural language processing history (firsthand in automated theorem-proving, but also in statistical machine learning), and in logic programming and Prolog alike (through Q-Systems), are sign of a most general destitution of philosophy of music in western philosophy of language, reason enough to convoke the Medievalist Boethius, and the Contemporary provenience of the generative theory of tonal music. The separate and individual treads of the different artificers of Prolog – Alain Colmerauer’s work in the machine translation prototype, Philippe Roussel’s work in SL-Resolution and in the Prolog manual, and Robert Kowalski’s work on the development of the Turing-complete logic programming paradigm – are framed in the investigation, inasmuch as, more broadly, Kantian critic philosophy, with the aim of endowing a classical thoughtfulness to artificiality, computing and informational philosophy, technology studies, and programming languages, with Prolog highlighted. Kant’s antinomies are forethought under the light of these premises, and informatic anew antinomies are interacted with the concept of virtuality in the *computus* era, by contrast with that of *calculus*. More narrowly,

the Kantian doctrine of schemata is faced up against the core of programming languages theory through a topic that the author has, conventionally, named  $\cup$ -Mentalism, meaning the possibility of equaling computable functions with computer vision hierarchy and, inasmuch as the inversion of Kant's account of schemata, the proper inversion of the von Neumann or Princeton computer architecture (image philosophy and ontological concepts, such as Bergson's "*La Pensée et le Mouvant*", "*Matière et mémoire*" and Deleuze's "*L'Image-Mouvement*" and "*L'Image-Temps*" reveal themselves very fit to one such cogitation). With this is also brought in the encounter with the Euclidean parallel geometry, a long lasting tradition until Bolyai-Lobachevsky, mainly under the relaxation of the fifth parallel postulate, having had a determining fate in philosophy of physics (from Riemann to Einstein). In like manner, diagonalization proofs from Cantor to Gödel (and the diagonalization process contained in computability and cryptography in informational communication) is shown to have enrolled the proper history of computation, with flagrant topological dimensions included. Logic programming and Prolog are, along this line, explored by calling upon the subject of the least possible constraint limits to logic and its inherent paradoxes, inevitably constituted also in programming languages topics and theory.

# Contents

<b>I Prolog, The Language of Mathematical and Dynamic Antinomies in Artificial Intelligence</b>	<b>1</b>
<b>1 Prolog and Logic Programming, a Declarative Language in Philosophy of Science, Essential Decomposed Fundamentals</b>	<b>2</b>
1.1 Preliminary Historical & Philosophical Remarks . . . . .	6
1.1.1 The von Neumann Architecture, Artificial Neurons, and Information Theory . . . . .	19
1.1.2 Informatic Anew Antinomies . . . . .	77
1.1.3 Appendix on Virtuality . . . . .	105
<b>2 Natural Language Processing in Prolog</b>	<b>113</b>
2.1 W-grammars and Q-systems in Natural Language Processing . . . .	113
2.1.1 The Chomsky-Schützenberger Hierarchy . . . . .	116
2.1.1.1 From Saussure’s Linguistics to the Algebraic Theory of Context-Free Languages . . . . .	123
2.1.1.2 Musical Interlude . . . . .	129
2.1.1.3 The Chomsky Hierarchy in Boethius’ Triptych . . .	136
<b>II Core Prolog</b>	<b>142</b>
2.2 Essentials . . . . .	143
2.3 Past Convergent Fragments of $\lambda$ -Calculus, Lisp, and Algol 60-68, and ahead divergent fragments of C/C++, Java, and Python . . . .	202
<b>Bibliography</b>	<b>224</b>



# List of Figures

1.1	Interpretation of multiplication. (Shannon's "Figure 3") . . . . .	29
1.2	The Parallels V <sup>th</sup> Euclidean Postulate (and Paradox). . . . .	72
1.3	Computational Complexity Classes . . . . .	76
2.1	The Chomsky Hierarchy . . . . .	116
2.2	The Chomsky hierarchy contained in Boethius' triptych . . . . .	137

# Part I

## Prolog, The Language of Mathematical and Dynamic Antinomies in Artificial Intelligence

# Chapter 1

## Prolog and Logic Programming, a Declarative Language in Philosophy of Science, Essential Decomposed Fundamentals

Prolog is a computational and logical programming language created in the Fall of 1972 (after a first preliminary version in late 1971), the name stands for **P**rogramming in **L**ogic (in French ***P**rogrammation en **L**ogique*) [78, 276, 223, 224, 125, 120]. A definitive version of Prolog was finally completed in 1973, and although the syntax retained the form of the preliminary version, it can be correctly said that its constitution has been the harbinger and prior direct (logical and computational) correspondent of all current Prolog(s). In this statement we are portraying, thus, the Prolog version of 1973 as having been the first in a direct line to the first ISO standardized Prolog (1995) and all its due natural *corrigenda*, considered in the time span the Warren Abstract Machine (WAM) (1983) as the *de facto* standard for all Prolog compilers/interpreters [78, 79, 321, 65].

The names involved in the core conception of the programming language Prolog are, above all, those of Alain Colmerauer, Philippe Roussel and Robert Kowalski, even if the foundational and philosophical articulation of a "Machine-Oriented Logic Based on the Resolution Principle" [316] (1965) belonged to J.L. Robinson. We can not neglect the various subsumed and hidden intellectual achievements without which Prolog could not have ever taken its form, such as the DPLL Algo-

rithm by Martin Davis, Hilary Putnam, George Logemann and Donald Loveland (1962), the (Alfred) Horn clauses (1951) or Herbrand's theorem (1930). As for all other sorts of efforts and accomplishments, they are naturally assumed.

For instance, Alan Turing's general development of theoretical computer science, Kurt Gödel's fundamental theorems on foundational (incomplete) mathematics and (complete) logic calculus or Alfred Tarski's "Concept of truth in formalized languages" [113] (1933 in Polish). But there are also other indispensable foundations such as Church's evolution from the conjecture on computable functions to the invention of  $\lambda$ -Calculus, John von Neumann's computer architecture and linear optimization, or John MacCarthy's coinage of "Artificial Intelligence" and development of the Lisp [195, 153, 360, 5] family of programming languages, parallel with Algol-like [339, 400, 3, 97] programming languages included, wherefrom Prolog sprouted.

We should not, anyway and in accordance with the precedent, fail to pay attention to the different ground thesis of philosophy of mathematics and logic of the contemporary age, namely, (1) logicism, (2) empiricism, naturalism and indispensability, (3) anti-realist and nominalist formalism, (4) intuitionism and (5) structuralism (following Stewart Shapiro's *diagnosis* of the twentieth century's western philosophy of logic state of the art) [55, 338], nor should we lose sight of the outgrowth of what is usually referred to as modern logistic, symbolic or mathematical logic. Encompassed in this expression are, more properly, the intellectual and speculative trails from the early-to mid-nineteenth century algebraic movement (e.g. George Boole in England, a generation later Ernst Schröder in Germany and Giuseppe Peano in Italy, but also men such as Charles Sandes Peirce, Jon Venn, Georg Cantor and even Hugh MacColl) up to the full logicist program between Frege's *Begriffsschrift* [134] (roughly *Begriff* for concept and *Schrift* for "script" or "writing mode") (1879) and the *Principia Mathematica* [399] (1910,-12,-13) by Bertrand Russell and Alfred North Whitehead.

These trails are all relevant under the general landscape posed by David Hilbert's program on metamathematics. *In continuum* with this idea, our aim should, at last, by encompassing all the arguments and retaking the first and last chapters of Martin Davis' book *The Universal Computer (The Road from Leibniz to Turing,*

2000) [96], call upon the frontier from "Leibniz's Dream" [96] to the *terra incognita* of "Beyond Leibniz's Dream", regarding the (possible realms of the) future of philosophy of logic and computation.

Alain Colmerauer, Philippe Roussel and Robert Kowalski, are, therefore and for as much as is revealed, the key figures in the birth of Prolog.

Due to intellectual and philosophical affinities and for the purpose of our investigation, we choose to separate Alain Colmerauer and Philippe Roussel on one side, and Robert Kowalski on the other. The combined work of Alain Colmerauer and Philippe Roussel permits us to focus more on computer science field assumptions within Prolog, while Robert Kowalski's background, work and approach lead us more towards mathematical logic, philosophical and logical reasoning, thus extricating a natural ambivalence in Prolog itself. What is more, they permit us to unfold a philosophical analysis into, respectively, the wedged apart but also conjoined, realms of natural language processing and logic programming (while Alain Colmerauer [78, 39] is an artificer of natural language processing, Robert Kowalski [224, 223] is a prime mover in logic programming, with Philippe Roussel being the representative of the work on SL-Resolution and automated theorem-proving, in addition to having written the Prolog reference manual [323, 322]).

These two natural research directions represent together the presumptive and inferred, theoretical and derivable status of Prolog (the disciplines of automated theorem-proving and mechanical inference relative to natural language processing and both natural deduction and first order logic relative to logic programming, all in all sharing *structural* logic and formal reasoning), but also its adaptive, pragmatic, observational, inductive and experimental nature (the two-level Van Vijnngaarden grammar extrapolation to Q-systems developed by Alain Colmerauer in natural language processing, and Philippe Roussel's different choice implementations from *Un système de communication homme-machine en Français* [323] to the *Prolog Manuel de référence et d'utilisation* [322], not to mention Robert Kowalski's choice of procedural Horn clauses in logic programming).

In its own right, natural language processing and logic programming can be said to equate, in encapsulated form in the computational era, the place of grammar studies and formal logic in the broader spectrum of the history of language and logic.

It is above all recommendable to abide by the order from natural language processing to logic programming, owing to the same broader *argument* that encircles grammar studies and formal logic, that is, the *complexity* argument [20, 406, 250, 146, 145, 289, 196], but also the superlative and convoluted movement from natural languages to computational programming languages. This intellectual endeavour consistently weakens the conjecture of one inherent bisection of natural and artificial languages. Overall and on the contrary, it sets a *continuum* in linguistic and computational, natural and programming languages. *Mutatis mutandis*, without a clear understanding of logic programming and program transformation techniques associated, the core of the language Prolog can never be rightly scrutinized.

Throughout the investigation into Alain Colmerauer, Philippe Roussel, and Robert Kowalski's work, we will be instructed in the development from automatic translation to the debuted research on man-machine communication (Prolog 0) [79, 321, 348, 323, 322] after Turing's *Computing Machinery and Intelligence* [368, 364] (1950) legacy.

Turing's historical paper can be said to have divulged, in Kantian terms, an internal and external critical experience or *phenomena* (the *Turing Test* or the *Imitation Game* [363, 366, 364, 365, 368, 362] as a special postulate in grave perpetual doubt), resonating a sort of non-contradictory human and artificial (or rather simply of pure reason - *mathematical* and *dynamical*, but also of practical reason) new antinomies, anon in computational terms. Such a *critical* experience should be considered on the presumption of one dialogic and conversational *natural language*. This judgement is exceedingly important if we remember the labelling of Prolog as a "conversational language" [73] (W.F. Clocksin & C.S. Mellish, *Programming in Prolog*, Fourth Edition, 1987-1994).

All the more, envisaged under the close study of Alain Colmerauer's work [78] is the interpreter of a programming language in the style of theorem prover in first order logic under Horn clauses restriction (Prolog 1, 1972), and the replacement of unification by solving equations in tree-structures (Prolog 2, 1982). These offer the backbone to the posterior arrival of constraint programming, where lists, rational numbers and boolean values are tree-structure refinements (Prolog 3, 1989), and, finally, the approximation to non-linear constraints by enclosure methods (Prolog 4, 1996). Philippe Roussel's *interim* computer science production in implementing

formal equalities and a synthesis mechanism reaching the form of a full interpreter of the emergent logic programming languages paradigm in one such form, has had the ability to mirror the precedent stages in the history of Prolog.

Finally, Robert Kowalski's investigations drive us through the avenue of logic programming, theorem-proving and knowledge representation, granting us the overall perspective of the computational significance of Prolog on one hand, and logic programming on the other, from which we will license a wider philosophical discussion. Such a philosophical route shall be seized under the very special and latter modern and contemporary historical-philosophical (cognitive and declarative) ascendant spiral from Leibniz (1646-1716) to Turing (1912-1954) in the first place, not consigning to oblivion, though, the preliminary, secluded and long-lasting (industrial and procedural) path from Archimedes of Syracuse (c. 287 B.C.- c. 212 B.C.) to Charles Babbage (1791-1871).

## 1.1 Preliminary Historical & Philosophical Remarks

The philosophical *substractum* of the programming language Prolog is, like a gem, shining and prismatic in nature. But like a proper gemstone, we can be avoided from seeing through its interior. Its facets are wide-ranging – from the bound of AI [325, 141, 352, 324] and logic programming to algorithms and data structures, from Lisp programming transactions to the implementation of neural networks, from human/computer interaction to computational linguistics, in between mathematical logic and computational mathematics – but, regarding the critical requisites of any advanced philosophical research, we must ask ourselves: has Prolog or, truly, computer programming languages as a subject, really been found to fill the critical gap in logic, philosophy of language and science?

Withholding Bordas-Demoulin's chosen Leibniz's "*épigraphe de la pensée*" – "*Sans les mathématiques on ne pénètre point au fond de la philosophie. Sans la philosophie on ne pénètre point au fond des mathématiques. Sans les deux on ne pénètre au fond de rien.*" [44] – finding, in addition, programming languages (Prolog distinctively), to be superlative examples of the intricacy of mathematical logic and philosophy's different ontologies, is it licit to say that the programming

languages - the web and weaver of the computational and informational age - have been philosophically probed?

Philosophy of language, after promising semiotic and symbolic underpinnings, laying siege to the linguistic turn, structuralist and post-structuralist reviews, and epistemological contra-revolutions, seems to have been afflicted with the passage from natural languages to computing programming languages (with modern formal logic and truth-conditional theories of meaning in the interpolation). Of course the history of computing and programming languages has ventured on natural language processing, computational linguistics and even cognitive science, accompanied by algebraic and applied combinatorics on words and lexical terms. But are we entitled to say, in an upstream (semantic) instead of a downstream (syntactical or computational behaviorist) sense, that philosophy of language has endeavored to analyze computing programming languages, the proper artificial prolongation of natural languages?

In regard to this, the history to state-of-the-art *interregnum* is quite clear if we compare Graham White's (*The Philosophy of Computer Languages*, 2004)[398, 53] authentic introductory remark – "The semantics of programming languages grew up in a particular historical context, and it is worth spending some time describing it: it was developed by a group of philosophically literate mathematicians and computer scientists, and the philosophical influences are quite evident." [398] – with its conclusion – "The overall goal of programming semantics is quite similar to the philosophical project of developing a theory of meaning: however, the methods and results are strikingly difficult. To a large extent this is because the philosophical project has been developed in isolation, with unsophisticated technical tools, and with the aid of a very small number of examples, none of them either large or complex." [398] – while considering, on the edge, that this exam is only compelled to programming semantics and abstractness (with so many other prevailing aspects left unconsidered).

Philosophy of science, absorbed in quarrels between underdetermination and confirmation, superabundant models of explanation and abusive intertheoretic reduction, all throughout the computing and information age, is found almost in a state of pleading for a new set of three constitutional Critiques - in spirit Kantian's



*Kritik der reinen Vernunft* (1781-87), *Kritik der praktischen Vernunft* (1788) and *Kritik der Urteilskraft* (1790) - now in the Cybernetics Age.

Indeed, the rebirth of European science, mainly driven, but also reciprocally caused, by astronomy and physics to metaphysics and epistemology, – with the scientists and natural philosophers Nicolaus Copernicus (1473-1543), Tycho Brahe (1546-1601), Johannes Kepler (1571-1630), Galileo Galilei (1564-1642) and Isaac Newton (1643-1727) as leaders – having, in hyperbolic fashion, elevated heliocentrism and enlightenment, seems to be at an odd parallel with contemporaneity. With this we mean to say that although obviously *incommensurable* - and not even competing historical paradigms in Thomas Kuhn's viewpoint [121, 217] (cosmos was, on the brink of entering the XIX<sup>th</sup> century, in *spatial* and *temporal* terms, just a gravitationally bound system comprising the sun and seven planets with a narrow biblically-inspired chronology) – utterly, even though contemporaneity lacks a physical unity theory, coetaneous metaphysics and epistemology are as *off center*, and *critically* bewildered, as they would be if a new all-abridging (physical) theory like Copernicus' *On the Revolutions of the Heavenly Spheres* [83] to Newton's *Philosophiæ Naturalis Principia Mathematica* [281] had taken place.

It could be argued that the shift to the modern post-Newtonian physics paradigm – the advent of quantum mechanics by Max Planck (1858-1947), its reevaluations by Erwin Schrodinger (1887-1961), Werner Heisenberg (1901-1976) and Max Born (1882-1970), and, fundamentally, the theory of general relativity by Einstein (1879-1955) – all above the previous work of Dmitri Mendeleev (1834-1907) in chemistry and Charles Darwin (1809-1882) in biology –, had set the standard for a new paradigm shift. This is absolutely true, but there is one striking element that is a nonpareil to all other scientific achievements, insurmountable in breaking through the frontiers of science and civilization: the computer as an artifact, i.e., computation theory in an analogue (non-symbolical physical)-to-digital (numerical symbolic) machine [86, 127, 376, 12, 122, 345].

Theoretical computer science, in its philosophical-mathematical-symbolical nucleus, corresponds, in short, to the Turing-Church thesis (1936) (the Turing-machine and  $\lambda$ -Calculus; a proper computing-machine and a programming functional language in abstract), cutting down to the utmost a whole research program on formal models of computability initiated in the 1930's and 1940's.

Visibly, there was inalienable solid cumulative work, sharing the same patterns of one "logic of the scientific discovery"<sup>1</sup>[304, 306, 305] (Karl Popper, 1959) in contiguous time and in propinquity: Kurt Gödel (1906-1978), Jacques Herbrand (1908-1931) and Stephen Kleene (1909-1994) worked on a number theory class named partial recursive or  $\mu$ -recursive functions prior to 1936, while, also prior to 1936, Emile Post (1897-1954) drew very near to Gödel's completeness theorem, scrutinizing Whitehead and Russell's *Principia Mathematica*, and John von Neumann (1903-1957), the future artificer of the von Neumann (or Princeton) architecture for computers [380] (1945), was the first witness and judge of Gödel's first and second incompleteness theorems.

But it is also very clear that, in what relates to computability and complexity theories, typically foreshadowing the computer, in both its just referred to valences – a machine and a program, as described in a Turing-machine and in  $\lambda$ -Calculus; the same is saying, a recursive instruction carrier mechanism with arbitrary sequences of arithmetic or logical operations automatically driven –, even if prior to the proper physical and realistic arrival of the computer, can only be circumscribed to the work of Turing and Church.

As a matter of fact, this is so, but it is to be understood that really what has molded the thesis author(s)-designation was the computer itself. In other words, what has made the "Turing-Church" designation preferable, even over the "Church-Turing" designation, was its conjoint machine-program (or instruction-recursion) elementary architecture. We should not, in any way, forget that this is

---

<sup>1</sup>Truly, Karl Popper's concept of "falsifiability" or, more generally, the epistemology of "falsificationism" is remarkably adequate to reason upon some underpinnings of philosophy of mathematics crisis under revolution, above all Gödel's incompleteness theorems. Refutability (as envisaged in theorems and not in simple conjectures), considered any formal axiomatic system containing basic arithmetic - one such as Russell's and Whitehead's *Principia Mathematica* -, of decidability or any "effective method", or algorithm with natural numbers expressions (Gödel's 1<sup>st</sup> theorem), and towards itself or its consistency as an extension (Gödel's 2<sup>nd</sup> theorem), all together in between effective axiomatization and completeness, undecidability and inconsistency, is really something that would relate Popper's work on philosophy of science with Gödel's achievements: ("the problem of induction" [304]; "scientific objectivity and subjective conviction" [304]; "why methodological decisions are indispensable" [304] and "methodological rules as conventions" [304]; "falsifiability and consistency"[304] and even whole chapters as "degrees of testability" [304]) reasonably seem extensions of Gödel's philosophy of mathematics into philosophy of science, but, strangely enough, authoritative acknowledging is absent (*primus inter pares* in Popper, or amongst academic *inter pares*).

the nucleus (*analytic*) of computation to which all the computer architectures – the Princeton or von Neumann [380, 379, 378] (1945) and the Harvard (modified) (1937-44) architectures included – are, literally, no more than a very elaborated (*dialectic*) epiphenomenon.

We shall, for the moment, refrain our analysis from the deep waters of the historical-mathematical unfolding of the concept of function – a many-to-one (or sometimes one-to-one) relation, where the set  $A$  of values at which a function is defined is called its domain, while the set  $f(A)$  subset  $B$  of values that the function can produce is called its codomain or target (with possibly other elements as members), while the range or image of the function is the set of outputs and the set of all input-output pairs called its graph ([176, 7]) – which was responsible for, at the shifting point of the invention of  $\lambda$ -Calculus and "effectively calculable functions" (Church, 1936), the transformation from the placed Leibniz-Newton tradition of *calculus* to the unplaced, as *off center* and *critically* bewildered, Turing-Church of *computus* (technically, "transformation" in the proper *geometria situs* archetypal sense, wherein the original shape of the object is called the pre-image and the final shape and position of the object is the image under the transformation, out of which translation and rotation – just about the simple *transformations* that enacted modern era heliocentrism and cultural enlightenment – are, although simple as they are, fine grained demonstrations of sufficiently strong *arguments* as to have put in motion the Copernican revolution).

Indeed, one of the characteristics of the artifact computer has been the transformation, literally the *transformation* in terms of a congruent translation of object  $X$  of Perimeter (ABCD), *ideally* the "scanned square" [362], or *theoretically* an "electrical circuit" [335] – with the translation lines the length of  $X$  in the plane to itself, else described as an automorphism, a one-to-one correspondence to itself in the continuous function of space (and time, with some  $T(n)$  maximum amount of time taken on any input of size  $n$  to the object Area (ABCD) ) –, from the *ideally pure* and *noumenal* bi-dimensional plane Turing-Machine (1936) to the *empirical* and *phenomenal* three-dimensional vector digital and electronic integrated circuit (1958, first semiconductor integrated circuit).

We can't be misled here, though: at the level of the binary Boolean in electronics communication, computing machinery and microprocessors circuitry (ana-

log to digital or mixed), and past the invention of the transistor (1947) and the integrated circuit (1958), past even the inception and evolution of the ARPANET (1969), with devices operated through instructions and data by using transistors on-off states as  $(1 - 0)$  or  $(true - false)$ , even if it is true that a flow of atoms and electrons is being conducted and spatial (non-locality) three-dimensional observance (hypothetically "reality") has to be accepted at the quantum level, at the *conceptual* level all the computability actions of digital set architecture level instructions of devices with memory, with or without a microarchitecture, having a logic design and implementation, necessarily follow the bi-dimensional *ideality* of a Turing-machine.

Consequently, the Turing-machine two-dimensional *ideality* is akin to two-dimensional axiomatic set theory, i.e., a study of collections of objects – in principle, positive integers or natural numbers in Peano's axiomatic with the principle of induction, moreover with possible correspondence to properties of all real algebraic numbers that possess the same mathematical structure –, with the result of one such *topological space* being *set* upon objects as points, and nearby other points-localities a collection of subcollections as *open sets* [397, 344, 227, 303, 349, 54, 220, 72].

More accurately, thus, we affirm that the history of computation, approximating as much as possible the millenarian history of computation to that of modern digital computer machinery [343, 409, 201, 386], has been, in spite of the Turing-machine holding the Archimedean property of excluding the infinitely large and the infinitely small, one of combining an odd geometrical-to-(*quasi*)-topological balance:

In between and from the congruent movement of different "squares" [362], taken as metric spaces, and current "m-configurations" [362], taken as isometric transformations and mappings on one horizontal and bi-dimensional axis (such that the distance between the pre-image and the image is equal to the distance and time-function between the elements or "m-configurations"), i.e., an  $n$ -tuple as a sequence or ordered list, as depicted originally in a Turing-machine, all in all in bi-dimensional  $\mathbb{R}^2$  *ideality*.

And in between the conjecturing conversion of each image and of the sequence itself – the Turing-machine "supplied with a blank tape and set in motion, starting

from the correct initial  $m$ -configuration, the subsequence of the symbols printed by it which are of the first kind will be called the sequence computed by the machine" [362] –, by means of length and width with infinity, i.e., a space where any point has no dimension, only position, to something quite new: an *ideal*  $\mathbb{R}^3$  Euclidean space where geometrical "squares" [362] relate in isometry to "m-configuration" [362] symbols, solely through the perimeter by the rigid motion of a translation, and wherein real numbers with expression, called the number computed by the machine as binary decimals, are obtained by prefixing the sequence with a decimal point.

Hence, this  $\mathbb{R}^3$  entirely abstract *ideality* (not yet virtuality) corresponds to a sort of open and infinite Euclidean-to-Hilbertian space *simulacrum* in *pure* and *analytic* computability terms, while in *symbolic dialectic* terms it is admissible that any non-Euclidean geometry is constructable and applicable as a computational *judgement*<sup>2</sup>, wherein symbolic manipulation is set free, and Turing-machine's dif-

---

<sup>2</sup>We follow here the idea behind Felix Klein's *Erlangen* program [221] (1872), an original breakthrough on classifying geometries by their underlying symmetry groups, connected with group theory and projective geometry. It was published by Felix Klein as *A Comparative Review of Recent Researches in Geometry* (1872) (on entering the Philosophical Faculty and the Senate of the University of Erlangen). What is fundamental in Klein's seminal paper is the hierarchic constitution of geometry according to the paramount importance of projective geometry over affine geometry, and the former and the latter over classic Euclidean geometry, all throughout establishing a pattern in relation with complex analysis. With Klein, every geometry holds an underlying group of symmetries, thus found to be invariants in association with the different algebraic structures or groups in the hierarchy. If noticed, the Erlangen program is model wise and frees itself from the undifferentiation mold wherein provability and truth are bundled, in such manner resembling, almost six decades before, Alfred Tarsky's mathematical definition of truth in formalized languages [113] and the idea of semantic equivalence between the "object language" and the "metalanguage", insofar as Klein's program attests to geometrical languages with *satisfiable* concepts in different "object languages". Also interesting to note is that Klein's Erlangen program, being more *perceptual* and akin to mathematical structures than Tarsky's *conceptual* theory of the formal notion of truth, shows a much more perspectivist and realistic notion of the now both reckoned to be the correspondence (Plato, Aristotle, Aquinas, Descartes, Leibniz, Kant, Gödel, Russell, Wittgenstein I), and the deflationary (Hume, Frege, Wittgenstein II, Quine) theories of truth.

In Kantian terms, we could asseverate the Erlangen program to accommodate projective geometry with the three kinds of synthesis: on its own, the *recognition in concepts of reason* (symmetries and groups), by enclosing affine geometry, the *reproducing in imagination* (transformations as reproductions from points, lines, planes and dimensions), and by enclosing, lastly, classic Euclidean geometry, the *apprehending in intuition* (Platonist solids and polyhedra perceptual *external* and *spatial* inner receptivity) (Cf. Kant [A97-A105]), always bearing a decisive and imperative duality of both the intuition and the concept.

ferent planes *per* "m-configuration" [362] through computable numbers – minimally and conceptually two perpendicular planes in a three-dimensional coordinate geometry establishing recursion (altogether a two-dimensional placed *simulacrum*) –, are responsible for the whole idea of programming languages and its hierarchy: Boolean values in the hardware rendered machine code (1<sup>st</sup> generation PL) in ascension to assembly language (2<sup>nd</sup> generation PL), to higher-level programming languages, with or without the web (3<sup>rd</sup> generation PL), still to more advanced features (4<sup>th</sup> generation PL), and, finally, to the constraint-based and logic programming type with the declarative paradigm (5<sup>th</sup> generation PL), where Prolog is included [350, 82, 256, 64, 66, 197, 313, 127, 24, 173, 226, 186].

Therefore, we have declared, in one such manner, in philosophical terms, an *n*-recursive *intelligible*, and, in *perspective*, *n*-dimensional *sensible* spaciality of computable functions (approaching virtuality if and only if the subject is considered), at a distance from constricted functional-*calculi* as *τσπος* (computable) *topos* (in minimality), or, more at large, taking the geometrical *judgement* of verticality (more so a centrifugal planar opposition in pure geometrical terms), the very idea of programming languages hierarchy (in maximality).

Beyond this, we aim to establish an *idealisticum continuum* of  $x : a \in \mathbb{R}$  (where *a* is a lambda-variable), but necessarily restricted to natural numbers according to the Turing-Church thesis and the limits of computable, or "effectively calculable" functions, where  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  produces  $f(x)$ , given sufficient time and memory. This critical boundary of the set consisting of all possible variables (passive of being bounded by lambda-abstraction  $(\lambda x.M)$  and by lambda-application to  $(MN)$ ), so that the set consisting of all possible variables  $\{x_1, x_2, x_3, \dots, x_n\}$  in Turing-equivalence, taking the programming languages existing properties as (PL), to be  $\{x/x \in \mathbb{N}, x \text{ satisfies } (PL)\}$ , can, in the end, be assumed as the group of computable functions that are simultaneously also programming languages under  $\alpha$ -conversion and  $\beta$ -reduction.

This is precisely the reason why Raúl Rojas has written: "The  $\lambda$ -Calculus can be called the smallest universal programming language of the world." [319] a fair portraying statement that could very well have been substituted by its inverse, in one such case being the  $\lambda$ -Calculus possibly described to be "the biggest universal

programming language of the world". This is so, accurately because of computability's proper *projective* nature, wherefore (and in between the *imagined* limits of a *vacuum* and a *plenum*) and *as if*, taking the physical entities of fields, an imagined (functionalist) field withholding Boolean symbolical manipulation over the phenomenon of electromagnetism, would likewise produce a *schema* or *schemata*, i.e., from within one *topus* (point; <name> <function> <application>) to *continuum* (list; <expression> <function> <application>) pure *ideality*. We will see how fruitful it is to understand this computation *ideality* from philosophical roots (metaphysically sprung from Leibniz-monadic to Kant-noumenal inceptions).

This is also the reason why John McCarthy, the creator of the programming language Lisp (the name taken from "lists" expansion and linkage in data structures, having expanded as a sort of mathematical notation precisely for Church's  $\lambda$ -Calculus, spinning-off two dialects: Common Lisp and Scheme) has reportedly said: "Lisp's core occupies some kind of local optimum in the space of programming languages." (John McCarthy, *History of Lisp*, 1978-79) [22, 189]. In this fashion, John McCarthy alluded to a type of early Aristotelian *Organon* informational and computational *Topics*, i.e., machinating the art of  $\delta\iota\omega\lambda\epsilon\kappa\tau\iota\chi\eta$  (dialectic) as the proper invention, discovery and abduction of arguments in which its (functional) arguments, resting upon commonly held statements, would meet  $\tau\omicron\pi\omicron\iota$  (topoi) being such *loci* or "places" the conceivable programs.

It is also along these lines that congruity is found following G. Revesz's notes on the subject of "Variables and functions in mathematics and programming languages" [314]: "It is obvious that different procedures may compute the same *extensional* function. But, in general, it is undecidable whether two procedurally defined functions are *extensionally equal* (...) Now, the problem of deciding the extensional equality of functions by examining their respective procedures is essentially the same as the equivalence problem of Turing machines, which is known to be unsolvable." [314]

Our strategy, after having captured Turing-equivalence of  $\lambda$ -Calculus, alongside Post's "Finite Combinatory Processes-Formulation 1" [308] really the only historical *momentum* equivalents amongst other logical-computing equivalents<sup>3</sup> – includ-

---

<sup>3</sup>Turing's *On computable numbers, with an application to the Entscheidungsproblem* [Received 28 May, 1936.—Read 12 November, 1936.]; Church's *An Undsolvable Problem of Elementary*

ing Post's *Recursive Unsolvability of a Problem of Thue* [307] (1946-1947) model, Hao Wang B-machine (1954, 1957), Böhm's theoretical machine language P" (1964), the Markov (1903-1979) algorithm<sup>4</sup>, besides the so called Turing-machine variants (such as non-deterministic, read-only, total, or universal, this last one the most interesting, by far<sup>5</sup>) – is to, following equivalence, equate computability theory (Turing-machine or  $\lambda$ -Calculus indifferently) with naïve set theory.

Our expectation is that from this basic equation, through constructive and finitary methods of semantic equivalence holding between naïve set theory and its computability equivalents – type-free  $\lambda$ -Calculus,  $\mu$ -recursive functions, Turing-machines, Markov algorithms –, in the simple form of one largely descriptive and intuitive survey, we can interpret the computing *spatial* two-dimensional, contigu-

---

*Number Theory* [April 1936]; Post's *Finite Combinatory Processes-Formulation 1* [September 1936].

<sup>4</sup>It is worth noting that the Turing-machine equivalents (and/or variants) all share the following characteristics: "single-tape", "one-way infinite", and a "multi-symbol alphabet" that cannot compute more mathematical functions than the algorithms, " $m$ -configuration functions", " $m$ -functions", or, in Church  $\lambda$ -definability, any " $\lambda$ -expressions" or " $\lambda$ -terms" denoting variables binding to the functions. Just so, this is the proper explanation why, in Kantian terms, we should separate the Turing-machine equivalents according to the logical function of the understanding in judgements, namely in modality, i.e., *apodeictic* (in which case we would find a reduction of the Turing-machine) and *assertorical* (in which case we would find only constitutive variances in the basic definitions of a Turing-machine) (Kant, *CPR* A70-B95). *Problematic* are also exceptionally important as they are the proper inner condition of the transcendental subject in most of its inner dispositions and momenta of thought, wherefrom truthful propositions disjunct from hypothesis, turning out to be a logical possibility or an as affirmed *a priori* logical necessity.

If we change in a Turing-machine "left-to-right" to "right-to-left", change the formal description of the alphabet, or even establish parallel computing, by going as far computing in quantum matrices, for example, we are only asserting logical reality or truth, thus producing an alternative Turing-machine *modus ponens* wherein, irrespective of many different *antecedens*, the *consequens* will always be the Turing-Church thesis. Again, all of these *modalities* are important, but we should only consider "formal reductions", as if a substitution derivation in calculus to formal reduction was performed. This is why we should give a closer attention – and even more so as it was a case of serendipity in science, with both Turing (prior five months) and Post articles being independently received in 1936, really the *annus mirabilis* in philosophy of logic – to the Post-Turing machine model (where one such *apodeictical* reduction occurs). Shortly, we should better consider post-Post machines (including Post's or Post-Turing machine) the only ones of the *apodeictic* sort.

<sup>5</sup>The Universal Turing-machine (or machine U) is just about the only one that makes use of the geometrical notion of reflection considered as a partial function, where reflection means literally the fixing of thoughts on something or, indeed, a stored-program, as means of computing a computable sequence. If this machine  $\mathcal{M}$  is supplied with a tape on the beginning of which is written the Standard description (S.D) we have the basic conditions for a Universal Turing-machine (*Cf.* [362] Section 6).



ous and structural *ideal* "squares" [362]. We hope to do so, simplifying, attaining only to the Turing-machine model, in the methodological fashion from geometry-to-topology-to-*imagination* ("imagination" here understood as in Kant: the *synthesis* reproduction of *images*)(Kant, [A120]).

If we computationally understand geometry connected to the faculty of sensibility (informational and digital impressions), topology in turn with the faculty of understanding (the reproduction of images in the *synthesis* of imagination, fundamentally by the perception of invariances in computer vision), and, finally, the faculty of reason closest to computer programming (the recognition of concepts of reason in relation to the patterns of the mind), we are, in our interpretation, advancing resolutely to a wider comprehension of both the *intensional* and *extensional* limits of computation. We call this prospect  $\cup$ -Mentalism.

Having computability testing equivalence with the human mind is naturally outside the bonds of experience, and, therefore, is in such way out of the inherent acts of experience (*perceptio*, *sensatio* and *cognitio*), but, in general, this act of representation that we choose to call  $\cup$ -Mentalism as a *representatio* is, in our conception, a *relational* fairly reasonable and permitted analogy of experience from the dynamical principle. It naturally extends in dynamics (and dialectic) the field of physical-philosophical and computational-programming possibilities, i.e, in the field of *modality* all the (*possible-to*)-*necessary* postulates of empirical (and computational) thought.

With one such methodology we propose looking closely at the equivalence classes characterized by "integral" (non-variable) and "differential" (variable) transformations from the Turing-machine "squares" [362] Euclidean space, through increased licensed anew transformations, to further achieve – not exactly any sort of non-Euclidean geometries (although we recognize the symbolical power of computation to reproduce non-Euclidean geometry-to-topology affine projections) –, but, instead, the power of procedural rules by which human judgments form images in perception, i.e., the force behind Kant's original idea of *schema* and transcendental schematism (Kant, [A137-B176 - A178-139 - B180 -A141]).

Insofar as we are calling attention to one geometry-to-topology-to-*imagination* (methodological and computational) *synthesis*, it is advisable to, in conformance, clarify that, although we are alluding to images (and a schematic and organized

*synthesis* of computational images in  $\cup$ -Mentalism), originally there is not any single intuition connected with the schema, but instead the unity in the determination of sensibility, making the *schema* very distinguishable from the image (Kant [A141]), albeit, in computational and programming terms under computer vision (*imagination*), it shall be, precisely, the subsumption of each *possible* image (*imago, imaginis*) to one *necessary* schematism (*schema, schemata*) that is worth being studied and tested.

Plus, every *possible* programming art under this appraisal, *necessarily* meets philosophy, and *intensionality & extensionality* augmented computer vision (in one such way a computer programming *medium*, here understood, by analogy, as language), as a formal and pure conception of "sensibility" (here computational, informational, and digital impressions), to which the conception of the understanding becomes unrestricted through its employment, provokes an unexpected turn: likewise the *schema* of the conception of the understanding is naturally restricted in sensibility (Kant, [B179-A140]), now the procedure of the understanding through *schemata*, i.e., the schematism of the pure understanding, can become now also computable (a naturally restricted domain of functions).

In one such effort, though, we are *perspectively* inverting Kant's assessment, in confrontation with computability. If in critic transcendental philosophy Kant was interested in going from the transcendental unity of apperception (Kant, [A107]), then to the pure categories of the understanding (Kant, [A129-A130]), and, finally, to the transcendental *schemata* of the principles of the understanding (Kant, [A137-A147 B176-187] (envisaging the passage from empirical concepts, to pure mathematical sensuous concepts, and, ultimately, to the pure concepts of the understanding *schemata*), in computability, alternatively, we are interested in the exact inverse pathway:

We are equating any and all produced symbolic images as a transcendental unity of any possible judgment (densely and extensively overpassing the limited bound of variables, constants, rules, functions, or the like), towards the end of producing imagery procedural programming and computing *schemata*.

The envisagement and fabrication (more so the exposition of *critical* limits of computability) of  $\cup$ -Mentalism certainly is to be compared, in its *integral* constant equivalences, with the Turing-Church thesis (1936) and computability equivalents

– type-free  $\lambda$ -Calculus,  $\mu$ -recursive functions, Markov algorithms –, and, what is more, it should also be compared, in its *differential* model, to all the modal variances that the *apperception* of the self permits. It should also be opposed, much wider than the real opposing programming languages paradigms, with the conception of one *architectonic*, pointedly, as the exact inversion of the von Neumann architecture or model<sup>6</sup> [380] (1945).

---

<sup>6</sup>The von Neumann model resulted in being a pyramidal discipline, like a standard physical theory, congregating different binding subjects: programming fundamentals and digital systems (*conceptual* level), algorithms and data structures (*mathematics*), operating systems and compilers (*practical* implementation and applied computer science), with computer architecture considered to be the focal point. There is a conducive historical line that is sharply representative of the historical time acceleration phenomenon.

Its most ancient antecedents are the Babylonian abacus (3.000 b.C.) and the millennial leap forward Hindu-Arabic numeral system (ca. 825), sorting a (*practical*) positional calculation machine and (*theoretical*) thought on number theory (in a pre-Einstein's sense of a *Gedankenexperiment*), envisaging a transformation of nods into rods, and an action from glyphs to graphs (in a pre-Euler sense), here understood as networks of dots and lines, or rather the passage from vertices, nodes, and points connected by edges and lines, pair with in relation to *quantity* by active *synthesis*, and constructed to find number systems in "tables" (in a pre-Turing sense). Next in importance are XVII<sup>th</sup> century Pascal's *Pascaline* (1642) (a performing adding calculator for the first time using the carry mechanism) and Leibniz's stepped reckoner (1674) (the first *direct-indirect addresses* arithmetic four-operations performing calculator), ergo both firstly anticipative of von Neumann's *First Draft* automatic digital computing systems, attaining to the "CA" or "central arithmetic part" [380] (forasmuch as it was possible in the pre-*calculus* era). The *Pascaline*, a final ending to more than 50 prototypes, working with five different positional cage gears with a set of ten spokes (0-9) for input dial with a hand stylus, starting the accumulator with metal stops in zero and clockwise (addition only) performing in cascade through a carry mechanism (reason why subtraction was performed by the rule of the nine-complement digit  $C_9(\textit{digit}) = 9 - (\textit{digit})$  hence  $C_9^{(n^{\textit{th}}\textit{power})} = 10^{n^{\textit{th}}\textit{power}} - 1 - (\textit{digit})$  so that for any given digit  $\varphi$  and a second  $\phi$ , the *positional*  $(\varphi+\phi)$  in the mechanical output could only be readdressed to the very same *positional*  $(\varphi-\phi)$  ) is here seen to endorse, by Pascal's machinery intelligence forethought, a "diagonalization argument" as understood by Cantor [119] (1891) (with results in  $\mathbb{R}$  uncountability and the *continuum* problem), also in Gödel through "Gödelisation" [161, 160] (1931) (whereby arithmetic finds a naming alphabet that, through a fixed-point and self-referentiality, leads to the 1<sup>st</sup> and 2<sup>nd</sup> incompleteness theorems), still in Kleene's recursion theorem [220] (1938) (application of computable functions to their own descriptions by the use of code in a finite set of numbers to correspond to a single natural number), in Turing's "Application of the diagonal process" [363, 362], and in von Neumann's carrying out of arithmetical operations seized by a general logic control [380] (1945) (a proper need for "zigzag discussion of the specific parts" [380] due the transversal "subdivisions of the system" [380]).

The abacus-linear *difference* to the diagonal-calculator, beyond the other typical dichotomies in the study of (analog-to-digital) automata, is very important, as it is to grasp that Leibniz's stepped reckoner was an improvement upon Pascal's calculator: it was still an induced hand-cranked calculating device, but now the *Pascaline*'s objects "five cage gears" and "ten spokes" were put together and *transformed* into the form of one cylinder with nine bar-shaped sizes,

### 1.1.1 The von Neumann Architecture, Artificial Neurons, and Information Theory

Succinctly, circumscribing at present time and from very different approaches and sources [377, 87, 179, 149, 271, 320, 251, 169, 272, 170, 370, 225] the von Neumann architecture has come after the fundamental "First Draft of a Report on the EDVAC" [380] (1945) – with "EDVAC" supposedly meaning "Electronic Discrete Variable Computer" (J. P. Eckert, Jr., and J.W. Mauchly; 1945 [198]) – written by von Neumann so to configure a computing architecture planning for the EDVAC construction and very much in debt to the collective and often secret work of the Moore School group [198]. The original *First Draft of a Report on the EDVAC* (1945), however inveigled in an aura of collective common ideas from the Moore School, and even if sometimes against the author's intent, is recognizably a work of the genius of Von Neumann.

The *Draft Report* [380] is organized into both one Euclidean-axiomatic way (axioms, corollaries & theorems somehow aligned with "1.0 Definitions", "6.0 Elements", & "15.0 The Code" [380]) and also into one Vitruvian-architectural way (with prescriptions for the planning and design of large structures and small contrivances of machines, measuring devices and instruments, utterly unveiling the universal present-day hallmark of informational and computational design and technology, dragging its modern analogies from aqueducts and mills, materials, forces, central heating, surveying instruments and machines, etc, now under the supine effort of computation). The *First Draft* [380] oscillates, thus, between enrollments that go from common-held and self-evident true statements to appended deducted propositions and, finally, to equivalent computational-informational theorems, while also plunging into the flexibly industrial, problematically procedural, statistical, often dynamical and ergodic implementation of machines.

The *First Draft* is divided into 15 sections of content, and the structure of a "*very high speed automatic digital computing system*"(1.1) [380] is, right from

---

paired with counting wheels moving up and down the wheel and meshing with the different sizes, a superb demonstration of *spatial-function-topologized* engineering and *analysis situs* in motion, as if one *characteristica geometrica propria* in machinery was found for the pre-(Turing-Church) *calculus* era.

the start, made dependent on its "*logical control*"(1.1) [380], something very interesting to follow as, computationally, it means that the Frege-Russell logicism, under Norbert Wiener' sense of Cybernetics – *Cybernetics, or Control and Communication in the Animal and the Machine* [404] and *The Human Use of Human Beings, Cybernetics and Society* [405] – was made to survive its philosophical foundation through "control and communication", after a tide of paradoxes, which, curiously enough, are *paradoxically*, also responsible for the theoretical foundations of computability and "effective calculability" through "computable numbers" [368, 362, 366, 363, 364, 365]. Actually, the "*automatic digital computing system*" [380] has ascended from an aboveground Boolean logic, but also through an underground (Euclid-to)-Cantor diagonalization method. This has proceeded even though great minds like Kurt Gödel (1906-1978) and Alonzo Church (1903-1995), who had worked on the foundations of recursion and computability theories, never came to think on the *synthesis* of both the *theoretical* and the *practical* different verges, a proper "paradigm shift" [121, 217] (in itself standing in need of "diagonalization", here taken as a *directional* "shift") that was only rightly emancipated by Alan Turing with *On Computable Numbers, with an Application to the Entscheidungsproblem* [362] (1936).

Indeed, it is odd that the era of paradoxes has produced such a marvelously effective piece of machinery as the computer, computability having entered into the realm of metaphysics in the short span of decades, and how paradoxes, "diagonalization" (and its prospects of cryptography in complexity theory) were set to be instructed to carry out arbitrary sequences of arithmetic by means of automata designed logical operations, on top of such a volatile phenomenon as electricity (overall a computational-informational David and Goliath parabola, where a supreme force was elevated from its weaknesses). This computational neologicism was, thus, made to survive through paradoxes, and it is worthwhile to observe that both the strong version of logicism (which "maintains that all mathematical truths in the chosen branch(es) form a species of logical truth" [356]) and the weak version of logicism ("which maintains only that all the theorems do" [356]) are, in *analytic* terms and *mathematically*, an impossibility, but have been made to seem *synthetically* possible by a sort of computational (connectionalist) *dialectic*, where from and where to programming languages respond, over and above, through a

massive network of computers, after the advent of the ARPANET (1969) and the World Wide Web (1989-1991), as the pinnacle of modern communication systems.

The *First Draft* [380] aims solely for results in "numerical data" form (1.2) which the "device can sense" (1.2) "through specialized organs" (2.2) – "punched into a system of punchcards or on teletype tape, magnetically impressed on steel tape or wire, photographically impressed on motion picture film, wired into one or more fixed or exchangeable plugboards – this list being by no means necessarily complete." [380] (1.2)<sup>7</sup>.

Also to take into account in the *First Draft* are the operations  $(+, -, \times, \div)$ , possibly including  $(\sqrt{\phantom{x}}, \sqrt[3]{\phantom{x}}, \text{sgn})$  – the signum function of a real number  $x$  is defined

---

<sup>7</sup>With this passage it is clear in the rendering of the device various other computer architecture historical landmarks, such as Joseph Jacquard's (1752-1834) programmable loom, controlled by a sequence of punched cards laced together. The Jacquard loom permitted that the woven textiles, made by passing threads over and under one another (plain or patterned), were programmed by punched cards, firstly designed as if they were fabric (in fixed-point self-referential logic) and, hence, copied from squared paper into punched, and afterward, stitched cards. In such a way, automated woven patterns were made possible by holes in the cards controlling which threads were raised and wires lifted in the intervals (here understood as the passage from Leibniz's stepped reckoner's *differential* cylinder to a complete " $m$ -configuration function" [362] table (in pre-Turing sense), *as if* the cylinder was *topologized* to a simple plane, and furthermore compacted and miniaturized, the processor and the program conjoint in its basic architecture). The Jacquard Loom, due to its direct image-semiotic programming, is *affine* with the cathode ray tube, the vacuum tube firing electrons to a phosphorescent screen, where it displays its pattern and *photon*-weaving (something that was only *peripherally* examined by von Neumann in the *First Draft* about the "iconoscope memory" [380] plate (12.8)). Computationally, the Jacquard loom logic lived through until the 1980's. It was the messenger for XIX<sup>th</sup> century Charles Babbage's analytical engine, XX<sup>th</sup> century Herman Hollerith's electromechanical punched card tabulator working for the American census, launching the Computing-Tabulating-Recording Company (IBM). It influenced, under the aspect of the square paper with stitched punched cards, reinvented as a paper tape (connected with a typewriter) and, later, film controlling the electromechanical process, Konrad Zuse-Schreyer's Z1 (1936), Z2 (1940), Z3 (1941), Z4 (1949, -50) (originally V1-V4 for *Versuchsmodell*, meaning "experimental model" with a punch tape and a tape punch reader), S1 (1942) and S2 (1944), John Atanasoff's electronic machine, known as the Atanasoff-Berry Computer (ABC) (1939, -42), the Harvard Mark I or ASCC by IBM (1944), Stibitz's Bell computers (1937, -44), the Colossus (1943, -45), up until the ENIAC (1946-1956) and the EDVAC (1946, -62). The Jacquard loom is the first machinery to abandon, in retrospect, Heron of Alexandria's cogwheel mechanics for smoother gearing (pre-relay type), a rather clever insight if we remember the difficulty of acceptance of the first logical schemes with diodes (1942), and the transistor (1947), up until to the first fully transistor computer TRIADIC (1955).

Philosophically, the Jacquard loom is the proper inferential image of physicist Brian Green's expression (and title of a science book): "The Fabric of the Cosmos: Space, Time, and the Texture of Reality." [155].

as  $\chi$  an array the same size as  $x$ , 1 if  $x > 0$ , 0 if  $x = 0$ , and -1 if  $x < 0$  –,  $\|$  ,  $\log_{10}$ ,  $\log_2$ ,  $\ln$  (natural algorithm with constants),  $\sin$  and inverses, all in all coming from a logic-machinery produced  $\pm$  (plus or minus). This marks a choice of exactly two possible values, one of which is the negation of the other, and being Pascaline-like in its "*first specific part: CA*" [380] (central arithmetic) (2.2), described, in an outline widely similar to topology, as "*elastic*" [380] (2.3), capturing with this expression its all-purposefulness. Uniformly, the "*second specific part: CC*" [380] (central control) (2.3) comes with circumstantial memory and code. The importance of fixed tabulations in many problems specific functions "on the basis of the analytical definitions" [380] (2.4 c) should all have an addition-reducing logic, and any partial differential equations, integrated along a variable  $t$  should be counted into the cycle  $t$  (the cycle  $t + dt$  is the cycle wherein the variable  $t$  parses with  $d$ , the initial and boundary conditions of the numerical material in the memory): "*the third specific part of the device: M*" [380] (2.5). Such cycles may include methods such as those of successive approximations, i.e., recursion within recursion.

"The three specific parts CA, CC (together C) and M correspond to the *associative* neurons in the human nervous system. It remains to discuss the equivalents of the *sensory* (or *afferent*) and the *motor* (or *efferent*) neurons. These are the *input* and the *output* organs of the device (...)" [380] (2.6) – this quote exposes the all-engulfing hard-biophysiological metaphor with roots in W. S. MacCulloch & W. Pitts<sup>8</sup>.

---

<sup>8</sup>W. S. MacCulloch & W. Pitts's influence on von Neumann was mainly driven by the paper *A Logical Calculus of Ideas Immanent in Nervous Activity* [248] (1943), in turn, thoroughly inspired in logic: Carnap's *The Logical Syntax of Language* (1938), Hilbert & Ackermann's *Principles of Mathematical Logic* (1928 first edition, 1938 second edition, 1950 American translation), and Russell & Whitehead's *Principia Mathematica* (1925) are the only references of the paper.

The "all-or none" character of nervous activity and its theoretical coinage by propositional calculus – as for the rest, strongly assertable after Claude Shannon's Boolean algebra (and numerical) electrical application (1937), after *A Symbolic Analysis of Relay and Switching Circuits* [335] – was the leading idea for connectionist nets, as if a neurophysiological equivalent was, thus, only to be achieved, rather than unfeasible. As follows, in spite of the difficult *locus* and precise range whereabouts of the equivalence, the net of neurons with soma and axons, synapses as adjunctions forming impulse-driven instants-thresholds constituting the nervous system with great velocity of communication are, in W. S. MacCulloch & W. Pitts's paper, the "immanent analogy" [248]. This analogy has deserved, often times, transcendental sealing (the dynamical principle of experience being only possible through the presentation of a necessary connection of perceptions wrongly taken on the grounds of the mere *aprehension* understood as *necessary*,

W. S. MacCulloch & W. Pitts's vision is mathematically and neurophysiologically bounded by the arrival of impulses at points "unequally remote from the same source" [248], and "from axonal terminations to somata" [248] even if the net definition in terms of communication hid the reason behind the flux and even speed timing, preventing more *ad hoc* interpretations. The excitability of latent addition and observed temporal summation is, therefore, a property of the structure, no matter reaching values sink to subnormal activity after rapid returns. This overall specificity depending upon time and space with greater or lesser refractoriness due to inhibitions, does not constitute a *material* argument against the envisagement of one calculus equivalent to the structure. Hence, the argument is one such of nervous nets as equivalent "*in the extended sense*" [248] and, naturally, of a physical calculus where physiological relations are made equivalent to relations between propositions and arguments, and where reactions can be read as assertions.

Thus, "facilitation and extinction" [248] on one side, and "learning" [248] on the other, akin to von Neumann's understanding of M(emory), are difficulties to the formalization, but the (neurophysio)-logical argument should not be understood as factual explanations: the electric-chemical threshold, after potentials and ionic concentrations, learning and enduring, are not a problem for formal equivalence logic theories, and neither are phenomena like "sleep, anaesthesia, convulsions and coma." [248]. It is, in this way, captivating why and how, in a sort of geometry-to-topology nervous physiology, as found also in Turing's *Morphogenesis* [365] the containment (in proper logical *existential* sense) of "many circular paths" [248] connate with regeneration of excitation and reference to time past, i.e., to M(emory) [380], and also to computability or recursion theory.

---

*as if*, by *analogy of experience* precisely, by the perceiving of events and an interior subjective determination of objects, one would think time itself was being experienced). This alert is cognated and more than circumstantially akin with recursion theory, as, in Kantian terms, the analogies of experience (even if taken conservatively from Hume's full-blown inductive skepticism) denote three analogies to the three modes of time (permanence the first, succession the second, and simultaneity the third of analogies) all exemplary in recursion theory (memory, recursion, and parallels/unification) and, for this reason, recommendable insofar as to point out that computability is, onto any relation of other sciences, and namely biological and life systems, *regulative* if anything, rather than *constitutive*.



W. S. McCulloch & W. Pitts's paper [248] directs onwards the discussion to "nets without circles" and "nets with circles" provoking formalized theorems from the formal calculus (related to any net structure unchangeable in time).

"The present theory is endowed with a symbolism "(...) of language II of R. Carnap (1938), augmented with various notations drawn from B. Russell and A. N. Whitehead (1927), including the *Principia* conventions for dots."<sup>9</sup>[248] (with some typographical differences) Typically: the neurons of a given net  $\mathfrak{N}$ , where the functor  $S$  whose property  $P$  holds of a number when  $P$  is equally right to his predecessor,  $S(P)(t) \equiv .P(Kx).t = x'$  <sup>10</sup>denoting the property of synaptic delay from the origin of time  $t$  with neuron  $c_i$  firing action described as  $N_i(t)$ , and the nearest predicate expression  $[Pr]$  for  $S(P)$ , and  $S^2Pr$  for  $S(S(Pr))$ , and so on, enabling quantified number variables through the subscripted numeral  $c_i$  or 'c<sub>1</sub>', 'c<sub>2</sub>', ..., 'c<sub>n</sub>.' to work as functoral arguments, and exposing a list of conjunctions and disjunctions with the actions 'N<sub>1</sub>', 'N<sub>2</sub>', ..., now forming the syntactical class 'N'.

The *peripheral afferents* of  $\mathfrak{N}$  that have no axons synapsing upon them, fall

---

<sup>9</sup>The use of dots in *Principia Mathematica* has its origin in Peano, and it was also used and studied by authors such as Turing, Church, and Quine. In respect to the use of dots for punctuation from the use in *Principia Mathematica*, "dots on the line of the symbols have two uses, one to bracket off propositions, the other to indicate the logical product of two propositions. Dots immediately preceded or followed by "V" or "⊃" or "≡" or "⊢" or by "(x)", "(x, y)", "(x, y, z)" ... or "∃(x)", "∃(x, y)", "∃(x, y, z) ... or "[<sub>1</sub>x](ϕx)" or "[R'y]" or analogous expressions, serve to bracket off a proposition; dots occurring otherwise serve to mark a logical product. The general principle is that a larger number of dots indicates an outside bracket, a smaller number indicates an inside bracket." [241] Conventionally, dots, considered as primitive symbols, range from ., for ( ), :, for [ ], ;., for { }, ::, etc.

<sup>10</sup>If we attend the logical syntax of a symbolic language, with its *formation* rules, and also its *transformational* rules, with deduction, properties, and relations between sentences defined on the basis of these rules, confronted with the systematic development of "Language II" [59] by R. Carnap (Language II as bearing the finitist Language I with propositional calculus and a Peano arithmetic with customary axioms, a symbol for 0 and for successor, and enriching extensions such as variables for propositions, predicates, and functors – functions with any number of arguments of any type – and predicates forming classes in unbranched type theory, and some other "fragments" such as the logical terms "demonstrable" and "refutable"), then the result is a sort of "coordinate language" [249] similar to mathematical-formulae physics. Usually, under Carnap's formalisation, the class of *analytic* sentences surpasses the class of *demonstrable* sentences. Language II by Carnap is, of course, remnant of general syntax – "a syntactical investigation of any symbolic language whatever" [249] enclosing a multitude of terms, from "variable" to "type", from "universal operator" to "translation" – and wherefrom  $K$  is meant to signify the class  $k$  of sentences implying the sentence  $S$  (and equally so in McCulloch & Pitts's paper).

under the formalization  $N_1, \dots, N_p$ , and consequently, all the others included as  $N_{p+1}, N_{p+2}, \dots, N_n$ , which turns to the idea that a "solution of  $\mathfrak{N}$ " [248] is related only when there isn't any free variable except under the order of  $[Pr_i]$ , neither any other descriptive symbol is contained in the expression, in one such way that  $S_i$ , taken as a numeral, is said true of  $\mathfrak{N}$ . Consequently,  $Pr_1({}^1p^1)$  (left superscripts taken as random variables and contingencies and right superscript as a progressive index) is only "*realizable in the narrow sense*" [248] if there exists a sense of a free variable like  $Z_1$  at the end of the  $[Arg]$  and the constant sentence  $[sa]$ , where the series of actions of  $N_i$  in the net  $\mathfrak{N}$  shall meet  $N_1(Z_1) \equiv .Pr_1(N_1, N_2, \dots, Z_1, sa_1)$  where  $sa_1$  is equal to  $N(0)$ . Whenever it is not needed the subscripted numeral of ' $N$ ' as if it were the prolongation of the functoral argument, and the neurons meet the quantified variable from the natural order of the functor  $S$ , as in  $S^n(Pr_1)(p_1, \dots, p_p, z_1, s)$  it is called "*realizable in the extended sense*, or simply *realizable*" [248]. From here on, the authors Warren McCulloch (neuroscientist cybernetician) & Walter Pitts (logician in the field of computational neuroscience), move forward to expand this idea to theorems about realizability *in the extended sense*, with this obtaining computable meaning classes of effective solutions materially stated, both cyclic and non-cyclic (presupposing a complexity notion made dependent upon the order of the net through its index).

The result is, formally, the conception of the "linear threshold gate" model (1943), wherein the summation ' $\sum$ ' (as syntax for disjunctions) and product ' $\prod$ ' (as syntax for conjunctions) is, in adding-derived logical sense as that of computability and related computer architecture, made very useful to minimally model the brain activity and neural nets. Therefore, if we consider any set of inputs  $I_1, I_2, I_3, \dots, I_m$  with a single output  $y$  classified in binary logical and interpreted as  $\sum_{i=1}^N I_i W_i$ , where in  $W_1, W_2, W_3, \dots, W_m$  are the weight values in the range in each input line, then, by finding for each  $y$  a functor sum  $f(\sum)$ , there is a threshold constant associated, independent of the numbers quantified. Besides this, the existence of *temporal proposition expression* (a TPE) ( ${}^1p^1[z_1]$  where  $p_1$  is a predicate-variable; and if  $S_1$  and  $S_2$  contain the same free variable, so equally does  $SS_1, S_1, S_1 \vee S_2, S_1.S_2$  and  $S_i \sim S_2$ ), building also a *temporal propositional function* (TPF) is all that it was left to be explicit before attaining the sequence of theorems: (theorem VII

"alterable synapses can be replaced by circles" induces "nets with circles" excursus and theorems VIII to X) [248].

What is also pivotal is the intermediary stand of both McCulloch & Pitts paper (1943) and von Neumann's *First Draft* (1945), with Claude Shannon's *A Symbolic Analysis of Relay and Switching Circuits* [335] (1938) and *A Mathematical Theory of Communication* [336] (1948, -49). In a certain way, we can read in both McCulloch & Pitts paper [248] and von Neumann's *First Draft* [380], respectively, the repercussion and anticipatory echoes from digital to information theory. It is exactly in this interval that symbolic electrical systems and digital switching circuit theory with the use of Boolean algebra was emancipated to a fully inculcated and newly founded field of information theory, whilst the debate of one dominant computer architecture took place, while the positive curve of technological hardware and software gathered power, speed and efficiency. It is, thus, recommendable to convey some notes about Claude Shannon's far-reaching work, before returning to von Neumann's *First Draft*, a text with several excerpted instances of "information" [380](1.2;2.6-2-9;12.2-13-2), soundly on the verge of architecture design of communication systems onset, all together a "blue print" for the digital age of cybernetics [335, 336, 8, 1].

Once Shannon's work is chiefly commanded by the initiatory *A Symbolical Analysis of Relay and Switching Circuits* [335] (1938) and the concluding *A Mathematical Theory of Communication*[336] (1948, -49), also bringing to a close *A Mathematical Theory of Cryptography* [333] (1945) and a *Communication Theory of Secrecy Systems* (1949) [334] we will start from the first and end at the second, relating as much as possible their most defining addresses.

Complex electrical systems with intricate interconnections of relay contacts and switches (automata, industrial-motors, or telephone exchanges) all share a "network synthesis" [335], as for the most part, its most relevant aspect is the requirement of the least number of relay contacts and also switch blades (of regular automata or pushdown sort), in prospect finding equivalence between all possible different kinds of networks. This approach takes the form of one equational circuit with a corresponding calculus, "exactly analogous to the calculus of propositions used in the symbolic study of logic" [335]. In other words, the network synthesis is not just one of relay contacts and switches, gates and control equipment. It is

a synthesis of a system of equations wielded by sentential calculus and enclosed in a gyration bounded representation. Thus, any two terminals (open, with infinite impedance and 1 of hindrance value; else closed, with zero impedance and 0 hindrance) receives the variable form of a function of time  $X_{ab}$ , and the symbol + (plus) forms the series connections of the two-terminal circuits with added hindrances. "Thus,  $X_{ab} + X_{cd}$  is the hindrance of the circuit  $a - d$  when  $b$  and  $c$  are connected together. Similarly the product of two hindrances  $X_{ab} \cdot X_{cd}$  or more briefly  $X_{ab}X_{cd}$  will be defined to mean the hindrance of the circuit formed by connecting the circuits  $a - b$  and  $c - d$  in parallel" [335], one such stand combining in circuit theory summation with an index, lower and upper limits.

The result is the development of theorems from circuitry postulates containing only series and parallel connections, and the perfect demonstration of how an (open or closed) circuit with electric one-only language (object-language) without self-referentiality, obstructs meta-language diagonalizations. On the contrary, it takes advantage of relay contacts and switches (a product '∏' diagonalization) to relate series and parallels connections. Still, not only this, but due to the fact that absolute reciprocity with Boolean-valued and propositional logic truth-tables (Postulates 1.a, 1.b, 2.a, 2.b, 3.a, 3.b, 4.) [335] is a fact of "series-parallel two-terminal circuits" (II), each theorem is, therefore, a dual theorem (with even greater difference from ordinary algebra than prototypical Boolean algebra, as relay and switching circuits do not have an *algebrized* ideal and are restricted, in principle, to disjunction only<sup>11</sup>).

This is quite interesting, once parallel series in Boolean values over digital circuits is one of computation's possible definitions, and it offers a proper analog prolongation of the machine discrete parts to the electrical phenomenon. It reposes digital symbolic Boolean values, thereupon, more on the axis of mathematical foundations, and furthest from machinery implementation and natural

---

<sup>11</sup>Another proof for this statement is the fact appointed by Claude Shannon in relation to the "Theorems" relating to (3b) :  $X + YZ = (X + Y)(X + Z)$  and before to (3a) :  $X(Y + Z) = (XY)Z$ : "The distributive law (3a) makes it possible to 'multiply out' products and to factor sums. The dual of this theorem, (3b), however, is not true in numerical algebra." [335] This assertion and the related should uncover a lesser candid interpretation of Shannon's words in relation to the choice of symbols: "This choice of symbols makes the manipulation of hindrances very similar to ordinary numerical algebra." [335] Also to remember is the case of the definition of the negative of a hindrance:  $X + X' = 1$ .

phenomena. Thus, it is acceptable to think that there is a sort of naturally open (infinite impedance) and typically closed zero hindrance, of relay and switching circuits to logic itself, and to (*algebrized*) Boolean values, which is a diagonalization on its own, as it would be expected, on the inverse, that there would be some "resistance" in transposing electrical circuits to Boolean values and propositional calculus, and, what is more, that zero impedance of closed systems should bring forth positive hindrance to any other systems, even if running themselves on relay and switching circuits. It shows, therefore, the sparkling quivering of the word "symbolic" in the paper's title "*A Symbolic Analysis of Relay and Switching Circuits*"[335] which grants also a great value to Shannon's accomplishments, for the reason alone that, besides the method of proof being that of "'perfect induction' i.e., the verification of the theorem for all possible cases"[335], the analogue between the calculus of propositions and the symbolic relay analysis is, thus, granted with a multitude of operational arguments, with sums passing to functions and one all-ready machinable network synthesis<sup>12</sup>.

Now, Shannon meshed E. V. Huntington's set of logic postulates with De Morgan's theorems (9a. 9b) (with the negative of a sum or product) and proceeds to represent a function as in infinitesimal calculus (in particular, along the line that any function continuously derivative may be expanded in a Taylor series)<sup>13</sup>. The

---

<sup>12</sup>The only stand admitted by Shannon of one *metalanguage* relating the "perfect analogy" (almost as indiscernible) between circuits and classic logic is here: "This analogy may also be seen from a slightly different viewpoint. Instead of associating  $X_{ab}$  directly with the circuit  $a - b$  let  $X_{ab}$  represent the *proposition* that the circuit  $a - b$  is open. Then all the symbols are directly interpreted as propositions and the operations of addition and multiplication will be seen to represent series and parallel connections." [335] What is also fruitful to be shown is that from this results the interchangeability of addition with multiplication at the operational level.

<sup>13</sup>Taylor series resorts to expand a series of functions about a point (under the logic of closing almost to  $X = 0$  geometric-transformation nearby points), here the variable  $X_1$  with the coefficients as multiplicative factors with the operations of addition, multiplication, and negation. The polynomial in the field of propositional calculus and relaying circuits engineering progresses, thus, with identity both for  $X_1$  and  $X'_1$  of the  $(n - 1)$  variables  $X_2 \dots X_n$ . Expanding about  $X_2$  we start noticing the pattern  $f.(1, 1)f.(1, 0)f.(0, 1)f.(0, 0)$  fixing at  $f.(0, 0, 0, 0 \dots 0)f.(1, 1, 1 \dots 1)$  (Cf.(10a)-(12b) [335]). In one such way we are free to find a circuit representing any given function recurring to the series expansion method, and in one way in which results on permuting primes are strongly related to coefficients maintaining the function value as 1, if any given variable appears at pairs - each time at most twice - a make and break contacts are also postulated in terms of electrical communications. Quite interesting is also to observe how a negative of a function equals replacing each and all variables by its negative and permuting the + and . operation symbols, something that can be interpreted as an

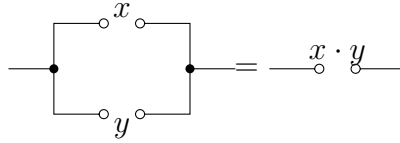


Figure 1.1: Interpretation of multiplication. (Shannon's "Figure 3")

conclusion parcels with the method of perfect induction and, more importantly, reduction with the least possible number:

"Any expression formed with the operations of addition, multiplication, and negation represents explicitly a circuit containing only series and parallel connections. Such a circuit will be called a series-parallel circuit. Each letter in an expression of this sort represents a make or break relay contact, or a switch blade and contact. To find the circuit requiring the least number of contacts, it is therefore necessary to manipulate the expression into the form in which the least number of letters appear." [335] The way is paved for simplification to succeed, manipulating the same genre of theorems of numerical algebra ((14a) – (18b)) so as to replicate the simple minimal pattern of input current (–o).

At this point it is clear that Shannon's preliminary work combined a sort of *structural* view of circuit theory and sentential calculus, not only by finding their *synthesis* and striving to find pattern matching, but also by seeing each equation as a system of variables, thus enabling the multidimensional approach that triggers algebraic manipulation with the inevitable integration of functions in space and time, on which grounds some residual features of number theory and symbolic analysis find correspondence in matrices as direct circuits, and graph theory as direct communication control images. While this is true, minimal basic *transformations* relating objects such as lines, planes and even rotations are performed,

---

intimate philosophy of mathematics riddle mystery between diagonalization and De Morgan's, even transposing to geometrical typical of *calculus* visualness, as if it was *negation* of a point what permits linearly and non-linearly "possibles", and as if each point was to be understood as a product function ( $\cdot$ ) of (all possible different rays of its *complex imaginary* circumference and sphere) taken as infinite possibilities  $f.\diamond(\epsilon \dots \omega)$ . This is, of course, something very reminiscent of Gottfried W. Leibniz's idea of the monad. It also helps to fathom why monadism would always supersede computability.

As for the rest, it is obvious that *synthesis* here is, in full, to be interpreted as a pervading Kantian idea.

with electric logic as the *dynamics* agent of change.

Indeed, it is the proper basis of a computational and information age combinatorics with order theory, giving rise to multiplying analogues of *structures* by the performing relay and switching circuits, definitely having strengthened Peirce's (1839-1914) and Wittgenstein's (1889-1951) initial intuitions of applied engineering to a whole ecosystem of the informational age. Positively, it was so to the extent that, even counting with the great probability of coming up verification and eventual ascertainment of one such progress, in one blow the informational age found a coordinate axis for communication theory (of Cartesian nature in electromagnetism), and symbolic computational graph theory (of Eulerian nature in electromagnetism), all in all in great optimization. Ergo, *calculus* and a sort of *greatest lowerbound* of computational and circuit theory topology (the abstractions called vertices, nodes or points and each of the related pairs of vertices, an edge, arc or line, in their least possible *spacial* and *logic* computational tractable form), through the study of invariances and variances, have perfectly fit (in principle in planar form, wherein networks can be drawn without crossing lines) the bidimensional  $\mathbb{R}^2$  *theoretical ideal* of the Turing-machine [362, 368], from the *practical* phenomenon of relay and switching circuits.

This was done in outstanding simple diagrammatic form, barely a connection of sets of dots for the vertices, joined by lines for the edges, therefrom found the *integral* discrete mathematics of computational systems and networks.<sup>14</sup>

---

<sup>14</sup>Claude Shannon's insights hide very interesting subtleties that are worth mentioning: take, for example, the alphabetical ordering-to-cryptography same letter-labelling of relays and hindrances in the network from the voltage source – with simultaneously make and break contacts if there is not any time delay –, all in all responding to the symmetric function, is a very powerful argument in comparison with strong computationalist views, namely Konrad Zuse's idea of a "computation-based universe" (1969), following the very first book on digital physics: *Calculating Space* [423], more or less along the lines of popularized versions of the Wachowskis film directors, authors of the trilogy *The Matrix* (1999), *The Matrix Reloaded* (2003) and *The Matrix Revolutions* (2003).

Leaving aside the inner arduousness and quagmires of having to balance information theory, statistical thermodynamics, and quantum mechanics, as first appointed by Edwin T. Jaynes (1922-1998), we can imagine Shannon's "III. Multi-Terminal and Non-Series-Parallel Circuits"[335] and the equivalence of  $n$ -terminal networks to be the whole universe (the possibility of  $n$ -dimensional parallel universes here contemplated), with the "voltage source" understood ontologically. In one such case, a (digital De Morgan's) change of the final reduction to its inverse would mean an instantaneously re-orientated universe, if we accept both the notions of direction and space. But what comes as more unexpected in this comparison is the idea that if a "computation-based

Shannon's notes in "III. Multi-Terminal and Non-Series-Parallel Circuits"[335] have an intrinsically established value with both *mathematical* multiplication and *ontological* diagonalization, inviting still leading-edge fragments of digital *metaphysics*. Transformation, as it was understood in geometry, was liaised with relay and switches with  $N$  being designated the "network" (a general constant-voltage relay circuit wherein  $N$  will equal  $M$  with respect to all the  $n$  terminals if and only if  $X_{jk} = Y_{jk}; j, k = 1, 2, 3 \dots n$ , where  $X_{jk}$  is the hindrance of  $N$  and  $Y_{jk}$  the same for  $M$ ), a whole new ontology granted thereby that was to have its correspondent new *aurora* with Tim Berners Lee invention of the W(orld) W(ide) W(eb) information space (1989-1991). The W(orld) W(ide) W(eb) with integrated application of hypertext in a transfer protocol (HTML via HTTP) for distributed hypermedia information systems and relay data communication between nodes through the use of logical hyperlinks was its successive digital-ontological materialization in the environment of CERN (*Conseil Européen pour la Recherche Nucléaire*).

Maybe one of the big contradictions of a "IV. Synthesis of Networks"[335] (1938), is, after its permitted transformations, also after the passage from the bidimensional mostly planar<sup>15</sup> *ideality*, to the *practical* existence of "*synthesis* of

---

universe" (Zuse, 1969) – rescuing various authors' conceptions, such as Edward Fredkin (1934, -) and Stephen Wolfram (1959, -) – was to be synchronously parallel, the only method by which the universe would relate its worlds or  $n$ -dimensions without re-orienting itself, would have to be either through absolute symmetry, or else by a very powerful diagonalization method, probably of spacetime itself (in both cases being very difficult to prospect the communication and spacetime complexity openings).

By the way, these grants offer the perspective on the difference between the exposed  $\cup$ -Mentalism and digital ontology (as peered by Konrad Zuse (1910-1995), E. Fredkin (1934, -), and Stephen Wolfram (1959, -), but also men as R. Rucker (1946, -), G. Chaitin (1947, -), and S. Lloyd (1960, -)), once  $\cup$ -Mentalism refers to the possibility of *synthesis* of all different images as representation and production of *schemata* for all different possible abstractions. It is true that the hierarchy of images and of the corresponding *schemata* can be thought according to the law of physics and computation, possibly through deep learning by "*difference in itself and repetition for itself*" [100] (borrowing from ontology an expression of Deleuze), but it is only incidentally that computational physics is related to the broad infinite set of images and *schemata*. This is so specially because  $\cup$ -Mentalism does not seem to be computable in the Turing sense, and instead offers a critical limit to the idea of mind (including the case whether speculation falls on a community of minds or on the religious idea of one such *ens realissimum* as God).

<sup>15</sup>Let us remember that there is in circuit theory a *quasi*-topological rule of non-planar networks, i.e., networks which cannot be drawn on a plane without crossing lines. As a result, due to the fact that the inverse of the distributive law cannot hold – " $X + YZ = (X + Y)(X + Z)$ ." – and crossed lines mean multiplication, the equivalent of sketching all possible lines which would break the circuit between the points under consideration, forcing the lines go through the hindrances



networks"[335], also under Kantian's idea of *synthesis*, the strict impossibility of a transcendental argument. In the electric relay and switching model analysis there is never the possibility of a conclusion being held as a necessary condition of a premise, thus obstructing the argument for apperception unity.

The drawn consequences of several different transformations – star-mesh (the transform replaces  $N$  resistors with  $N(N - 1)/2$  resistors with one less central node, building a polygon with three interior triangles), delta-wye (from the application of the distributive law  $X_{ab} = R(S + T) = RS + RT$  at the node  $R$ , considered as the side of one equiangular triangle, along with  $S$  and  $T$ ) and wye-delta (the opposite of the previous, equal also to the elimination of the central node as in star-mesh), always require the least number of elements and the rule of induction  $n - 1$ , transformation being very alike to  $n - 1$  itself of star-mesh for each node, and for series parallel networks. The study is then transposed to the "Hindrance-function of a non-series parallel network"[335], closer to typical graph's theory bridging problems, where from and primarily reduction means settling the network to an equivalent series-parallel circuit, with three methods involved: 1) intuitively applying the transformations until the network is of the series-parallel type and wherein the successive impediments of the hindrance-function match the complex-valued resistance of the impedance; 2) writing the product of hindrances of all possible paths between the two points, skipping paths that touch the same point more than once, stating the hindrance of each path a factor of this product, including function 0 for 0 hindrances, an overall congruent method with the first, i.e., a product of sums; 3) writing the hindrance function as a sum of products, for which what is needed is to draw all possible lines that can break the circuit between the points of the network, forcing the lines to break through the hindrances of the circuit.

It is true, as Shannon expressed, that "the justification of this method is similar to that of the second method"[335], but maybe it is more accurate to describe it as a method very like diagonalization, on which foreground and only trivially the second and third methods are, indeed, similar and inter-reductive. Relating inherently to the first method, that of applying transformations until the network

---

of the circuit, the result being a sum, is not always possible to achieve.

is of series-parallel type, logic is, thus, made *planar* and *theoretically ideal* in  $\mathbb{R}^2$  as in the Turing-machine model [368, 362]. This puts us, after spectating multiplication as diagonalization, more closely towards the envisagement of parallel-series reduction from non-series-parallel networks, as a continuation of diagonalization (the three methods being, discernibly, very similar), and wherein, most fundamentally, no paradoxes, or inefficiencies in completeness and consistency, are found in the network synthesis. This can be explained by the theory's *analytic* and discrete simple *integral* attainment (from the the second fundamental theorem of calculus or the "Newton–Leibniz axiom", and considering a Riemann integrable with the partition fine enough, taking the idea that if  $\delta > 0$  and  $\epsilon > 0$ , there exists a tagged partition  $y_0, \dots, y_m$  and  $r_0, \dots, r_{m-1}$ , thus and so that for any tagged partition  $x_0, \dots, x_n$  and  $t_0, \dots, t_{n-1}$  which is a refinement of  $y_0, \dots, y_m$  and  $r_0, \dots, r_{m-1}$ , we have refinements constraining as much as possible close to 0 or the  $x$  axis of a function, again, *ideally* through 0 (*possibly* with the *orientation* left-to-right), here shown as the integer *ideal* (electrical) interpretation ( $-\circ$ ) as close as possible, firstly to the  $x$  axis ( $\_ \_$ ), but also to (*ideally* equivalent) parallels ( $\equiv$ ), and thus to the  $y$  axis and constants for indefinite integrals with the approximation ( $\simeq$ )<sup>16</sup>:

$$\left| \sum_{i=0}^{n-1} f(t_i)(x_{i+1} - x_i) - (-\circ) \right| < \epsilon$$

We are, hence, excluding, with reference to the closed interval  $[a, b]$  (greater than or equal to  $a$  and less than or equal to  $b$ ), taking the real-valued functions  $f$  (derivative on almost all points of  $[a, b]$ ) and  $F$  (continuous on all  $[a, b]$ ), such that  $x \in (a, b)$ , turning 0, in limit, similar to (possible maximum/minimum)  $F'(x) = f(x)$ , and if  $f$  is integrable on  $[a, b]$ , then, attaining to a measure zero, the following holds:  $\int_a^b f(x)d(x) = F(b) - F(a)$ . In one such way, the greatest lowerbound and the lowest upperbound are "discharged" and the theory is made discrete (even if there are hindrance elements controlled by external sources, as contacts on external relays or hand-operated switches, designated, in this fashion, as independent variables, represented by the earlier letters of the alphabet, in contrast with the later letters for dependent variables).

---

<sup>16</sup>We are here interpreting 0 to be the integral with an arbitrary constant.

With this being said, Shannon fits this interpretation towards the appraisal of a system of equations defining in full the operations of the system, the dependent variables prone to be computed. Transformations, under this prospect, are set to one equative form where  $X_{0k} - X_{0k} = 0$  has to have  $X_{0k}$  as the hindrance function of  $N$  between terminals 0 and  $k$  and, therefore,  $X_{0k}(k = 1, 2 \dots n)$  is made invariant, even if  $X_{jk}(j, k) = 1, 2 \dots$  – can be changed, transfiguring the "old network", however the relays of the operations remaining the same.

From this standard, in diagrammatic form, "(...) drawing a vertical line after the terms common to the various equations (...)" [335] we can have as an example of reduction of simultaneous equations both a circuit design of relays and switching circuits, as in one Gentzen-style propositional deduction. It is also convenient to trust simplicity (a sum of products, instead of a product of sums, overall an *orienting* argument, as relays should be as much minimally *horizontal* - "series-parallel two-terminal circuits"[335] - as possible). With this, the higher-stand logic system abridging neutrally the principle of duality, allows also its opposite: multiplication to represent series connections, and addition parallel connections.<sup>17</sup> Overall, time is a well-known bedevilment of complexity theory, and sometimes a definite sequential relation of relay contacts, due to asynchronous relations, has to lean on make-before-break (or continuity) and break-make (or transfer) contacts, wherefrom the hindrance in series  $X(t)$  is exposed as  $X(t - p)$  with time  $p$  being whichever unit-measure of time later in the description (with the operations of addition, multiplication, negation, and equality).

Shannon's "IV. Synthesis on Networks and Functions"[335] (1938) is, thus, the epilogue paragraph that rehearses best the ciphering for Claude Shannon's *acme*

---

<sup>17</sup>From this clause it is probably made clearer why, previously, it was here before said that, under digital philosophy (e.g., Wheeler's "it from bit"), diagonalization would have to be one such powerful method as to be part of spacetime itself, and, thus, one *orienting* argument too, in the proper Kantian sense. Also to add is a very interesting quotation by Shannon, very useful to confront with digital physics and informational ontology in what has to do with spacetime, but that I wish to call attention upon, mainly, in relation with  $\cup$ -mentalism's use of function in relation to the *synthesis* of images for each time and space, with all possible variations considered. Shannon wrote: "The function requiring the most elements using any type of circuit has not as yet been determined." [335] It is here being reasoned, on our side and from this example, nothing else but the consideration of spacetime once, in absolute symmetry, it equals one dual network of two series-parallel realizations (for convenience a Cartesian reduction called two coordinate-axis  $x$  and  $y$ ; or any other pair, such as *vertical* and *horizontal* axis) in what relates to *orientation*.

work: *A Mathematical Theory of Communication*[336] (Shannon & Weaver) (1948, -49).

From the basics of a two-folded relation – a current (a magnetic field rate of flow of electric charge the same through all components connected in series and distributed over components connected in parallel) and voltage (the electrostatic field potential difference in charge between two points in an electrical field distributed over components connected in series, and the same across all components connected in parallel) – the very idea of network synthesis is stated<sup>18</sup>. If there are only two variables, there are 16 functions at hand. Eliminating those that do not involve directly both  $X$  and  $Y$  ( $X, Y, X', Y', 0, 1$ ) we are left with 10. As this is not proportional (3 variables and 64 functions do not give back (-8)), a theorem is reached:  $\models$ .

$$\phi(n) = \sum_{k=0}^n \left[ \binom{n}{k} 2^{2^k} (-1)^{n-k} \right]$$

We should read the theorem with  $\phi(n)$  as the number of variables, for each time under consideration and one-only  $n$  and  $k$  as  $\binom{n}{k}$  representing  $n!/k!(n-k)!$ , made expressible by having resolved the previous result with time involved (the number of variables being the difference between its exponentiation and the actual sum with verified variables in the selection):

$$[\phi(n) = 2^{2^n} - \sum_{k=0}^{n-1} \binom{n}{k} \phi(k)]$$

It follows that the functions of  $n$  variables which require the most relay contacts and the number of contacts needed comes from a sum modulo two or disjunct of the variables (commutative, associative, and the distributive law in respect to multiplication) from the equation:  $X_1 \oplus X_2 = X_1 X_2' + X_1' X_2$ .

*Moduli* two, in binary and digital logic, by the presence of the associative law, paves again the way in for the reentrance of series-parallel  $X_1 \oplus X_2 \oplus X_3 \dots \oplus X_n =$

---

<sup>18</sup>Network synthesis, understood as a sum of products, having the coefficient 0 or 1 ( $2^n$  different products if the division is set up dividing the set of primes and non-primes, and beyond  $2^{2^n}$  now dependent on having the coefficient 0 or the coefficient 1) directs to the theorem "The number of functions obtainable from  $n$  variables is  $2^{2^n}$ " [335] where each of these sums is a representation of a different function (except if they are null  $f|X_{k=0} = f|X_{k=1}$ , inoperative because the value of  $X_k$  is made irrelevant).

$\Xi_{k=1}^n X_k$  where the root advances a theorem for any two functions of  $n$  variables which require the most elements (relay contacts) in a series-parallel realization, for the right-hand side of the equation and its contrary –  $\Xi_1^n X_k$  and  $(\Xi_1^n X_k)'$  – each of which requires  $(3 \cdot 2^{n-1}) - 2$  elements. Shannon first noted that for  $n = 2$  and for all the functions, only two required the two elements:  $X \oplus Y$  and its contrary  $(X \oplus Y)'$ . It was then tried with  $(n - 1)$  to check if it held the most elements for  $n$  variables, after the functions were expanded as terms for the  $n^{th}$  variable. It shows that the functions require the most elements. Adapted from (19)[335]  $s(n)$  meaning the sum of all the (positive integer) number of elements required, and for the equation  $s(n) = 2s(n - 1) + 2$  the value is  $s(2) = 4$ . As this is linear with constant coefficients, the solution is applying the method of boundary conditions as constraints on solutions to equations, and from there the result is:  $s(n) = 3 \cdot 2^{n-1} - 2$  recommended to informational theory.

Shannon's *A Mathematical Theory of Communication* (1948, -49) is, truly, a setpiece of an all-abridging era of *synthesis*. It was this "unifying vision that revolutionized communication, and spawned a multitude of communication research that we now define as the field of information theory" [8], hardly comparable in its social, political, and technological synergy. The text spans five parts. They are: "Part I: Discrete Noiseless Systems"[336], "Part II: The Discrete Channel With Noise"[336], "Part III: Continuous Information"[336], "Part IV: The Continuous Channel"[336], "Part V: The Rate for a Continuous Source"[336]. We will, obviously, try to keep them to the most clear-cut philosophical arguments, always keeping in mind that the theory is mathematically complete and consistent, mostly by cause of the fact that, while propositional logic equivalence is transposed to mathematical higher-order irreproachable abstracts, what is continued is the following: "the semantic aspects of communication are irrelevant to the engineering problem." [336]. This means that not only logic is balanced enough not to enter higher-order domains, leaving that task to mathematical manipulated extensions, as logic itself does not mean anything here except binary valences of *bits* ("a word suggested by J. W. Tukey" [336]).

"Part I: Discrete Noiseless Systems"[336] – by the title, borrowing from noise modulation methods (exchanging bandwidth for signal-to-noise ratio in the sharpest

balance), thus, electing natural language processing electrical channeled forewarnings as relevant – aims to target savings within the referred balance. Beset in the interpretation, has come the assessment of a whole new paradigm on the nature of information theory. Inasmuch as Gödel’s work on the incompleteness theorems, and even recursion theory, were shown to be, contrary to Gödel’s expectations, an antiderivative basis of the curve for the domain of  $\mu$ -recursive functions, thereby within all the possible range of the functions of computability (with differentiability here understood including the proper incompleteness of mathematics and the negation to the decision problem unexpectedly and suddenly *integrating* computability), and in a similar manner to Erwin Schrödinger’s, also to his own dislike<sup>19</sup>, new wave mechanics based on probability waves, so too information theory was born out of high indeterminacy and entropy, statistical and probabilistic presumptions, but with a finer and wider gain on information processing and message decoding. Obviously, this is a memento of the special ontological blend between physics, computation, and information, whereby programming languages, as a subject, should emancipate itself to the communication pattern inbetween them.

Indeed, information theory too, was born on the presumption of the probabilistic view that any monotonic function typical of *calculus* (either entirely non-increasing, or entirely non-decreasing), regarded as a measure of information, was plainly distributive in both axes, meaning that each message chosen from the set was a choice with equally likely values. The logarithmic measure, due to linear comparison with common standards, where typical one *bit* two-stable positions, "such as a relay or a flip-flop circuit"[336],  $2^N$  for any number of  $N$  circuits or devices, and base 10 for decimal digits units  $\log_2 M = \log_{10} M / \log_{10} 2$ , with the result of  $3.32 \log_{10} M$ , approximates to the decimal digit, which is about  $3\frac{1}{3}$  bits. This means that only after three cycles of  $\log_{10} M$  one *bit* of information is performed, and, overall, in geometric-to-topological terms, there is one slightly higher  $\pi$  (3.14) *ratio* between the decimal values (diameter as spokes on a wheel) to bits of information (the circumference equivalent circuit). Integration and differentiation typical of *calculus* produce the natural logarithm  $e$  – approximately equal

---

<sup>19</sup>Schrödinger’s quote on the new wave mechanics (stationary and time-dependent, holding the formalism of matrix mechanics) is as famous as it is unreported: “I don’t like it, and I’m sorry I ever had anything to do with it.”

to 2.71828 and a result of the limit of  $(1 + 1/n)^n$  as  $n$  approaches infinity – to establish the natural units of information, after which stipulation the base  $a$  can be changed to base  $b$  by multiplying  $\log_b a$ . The result is the famous schematic diagram of a general communication system [336] comprising the following elements: an *information source*, a *transmitter*, the *channel*, the *receiver* and the *destination*.

The *information source* is the produced messages to be communicated to the receiving terminal. These can be (a) of the alphabetical sort as a sequence of letters (classically a teletype system), or (b) a single function of time  $f(t)$  (classically a radio system). The most complete view is, though, that of (c) a function of time and other variables, embedded in two space-coordinates and one of time "three-dimensional" plane (classically black and white television), the same related with sound transmission (classically multiplex individual channels), and even (e) several functions of several variables (classically the three-dimensional continuum of color television). Ultimately, it remains (f), i.e., various combinations of the previous.

This section is outstandingly important, precisely because it offers, under the most important topic of information theory, the perfect correspondence between information, philosophical and computer architecture, and medium or language. We shall not forget that the medium or language is, primarily, not language itself, neither sound and light, nor even light alone except taken as a manifestation, but more accurately electromagnetism only (mechanistically, sound are changing sound pattern pressures into a proportional-transmitted electrical current, while light does not account for another medium, and is fully set down in electromagnetism, i.e., electrically charged particles in razing sharpening confluence with other fundamental forces, such as strong and weak interactions and gravity).

Shannon's view is, thus, however dated and historical rooted in a recent past differential techno-scientific conjecture (observable in the very same historic-root as natural language processing in Prolog, by the tonic on the *sonus* paradigm, against the more cutting-edge and monad-like *photon*), of sublime importance. This is so not only in what has to do with the informational-statistical-probabilistic new allegiance, breaking down to shatters much of what was the typical of *calculus Aufklärung* (and formally Kantian) *illusion*, a proper historical-dialectic transcendental illusion (even if strong computationalism, in the core of information theory and against computability theory *analytic*, represents its main *dialectic*).

Fundamentally, what comes about is the possibility of, in proper informational paradigm transferring correspondence, the bridging to ontology, namely through (*f*), with a full understanding of the "schematic diagram of a general communication system"[336], in its anew countenance with probabilistic, statistical, non-determinist, and cryptographic indexes of truth as a logical value<sup>20</sup>.

Of course Shannon's theory is clearly founded pertaining to communication systems (discrete, continuous, and mixed) represented by mathematical entities and its necessarily dated, mechanistically turnout physical counterparts. For many different reasons, the simplest of the discrete systems, which respect both the message and the signal, is a discrete sequence of letters, the signal dots, dashes and spaces, as in telegraphy and, indeed, modern computing machines, which establishes the foundations for the remaining types (continuous and mixed), reason enough for an unmitigated study of "Part I: Discrete Noiseless Systems"[336].

Randomness and statistical dependence have found, really, its geometry-to-topology coinage, dragging programming language's fixed-point theorem, once a discrete channel is a sequence of finite choices from a set of elementary symbols  $S_1 \dots, S_n$  from one point to another, each of which with a certain duration in time

---

<sup>20</sup>∪-Mentalism can, by these standards, be described as the inversion of the various type of messages both in predominance and in order, now from (*f*) to (*a*)[336]. The various combinations of all types of information are reasoned, firsthand, in an equivalent cinematic "*l'image-mouvement*"(Deleuze, 1983) *photographie et photomachinae* (not necessarily digital) group of several functions of several variables (approximating as much as possible *ideal* total functions from *practical* computable partial functions, i.e., functions defined for all possible input values, well within the limits of a the Turing-Church thesis), through which *ideally* and minimally, the three functions  $f(x, y, t), g(x, y, t), h(x, y, t)$  "defined in a three-dimensional continuum"[336], respond to the technological object "color television"[336], encountering, nevertheless, in ontological terms and in ∪-mentalism, observable reality. Subsequential to this are the other "types" of messages: (*d*) as the representation of two or more functions of time in respect to sound transmission in a system, (*c*) its weaker version of one function of time and other variables, such that  $f(x, y, t)$  are two space coordinates,  $(x, y)$  the light intensity at one-point and  $t$  its time, whereby it is seen the similitude between the model for "black and white television"[336] and "sound transmission"[336] in a system (without vector fields in the region), just with a Cartesian 2-dimensional "tube plate"[336], (*b*) a single function of time for sound and voice as in "radio or telephony"[336], and, finally, (*a*) i.e., just about a sequence of letters.

We can conclude, therefore, by this exam, that programming languages are, in their state of the art, inescapably aligned with the simplest form of communication, that is, "(*a*) A sequence of letters as in telegraph of teletype system"[336]. Now, this is an historical cursive function as recursive it is computability theory, but it shall be given the reflexive means through which programming languages or programming media as a subject are properly acquainted with wider goals, and thus both informatic and informational perspectives.



$t_i$  (one dot and one dash would haul different time instances, and there is also an overall problem of transferring or break-make solutions in overall computation, independently of the decision problem, for which case only a system of spontaneous and immediate *imagnetic* recognizance of *differences* through *repetition*, with a practical unlimited memory, could account for the same original sense as in photography's *revelatione* comparing it to computational processing of information data messages).

Now, the measure of the capacity of one channel to transmit information, irrespective of the length of symbols and constraints on the sequences, targets the maximum possible rate, sufficiently equal if the source of information is capable of feeding supremely the channel (wherein, for instance, in the telegraphy, a dot – one *bit* – with a two-state break-make line closure unit of time before – two *bits* – and another break-make open unit of time after – two *bits* – would account in total 5 *bits*, the same as, say, a letter space of three units of line in an actual teletype case, but not the same in telegraphy (overall tendentious power of 2 and multiple of 8 in modern computer processors). The capacity  $C$  of a discrete channel

$$C = \lim_{T \rightarrow \infty} \frac{\log N(T)}{T}$$

expresses a formula where  $N(T)$  is the number of allowed signals of duration  $T$ . Ensuing this, if it is supposed that all sequences of the symbols  $S_1 \dots, S_n$  are allowed, and have the duration  $t_1 \dots, t_n$  it is noted that the channel capacity corresponds to the total sum of the numbers of the sequence, and then the asymptotic or limiting behavior for large  $t$  to  $AX_0^t$  where  $A$  is constant and  $X_0$  is the largest real solution, transforms the previous formula, to the following

$$C = \lim_{T \rightarrow \infty} \frac{\log AX_0^t}{T} = \log X_0$$

Here we need not be limited to solving the restrictions on sequences, and can find the difference equation from any equation, and very much alike, from a (stochastic and probabilistic) Markof model, wherein the junction points are states – any possible states  $a_1, a_2, \dots, a_m$  – and the lines indicate the symbols – certain symbols from the set  $S_1 \dots, S_n$  – contained in the respective state (the first theorem exposes, therefore, the relation between the "last" symbol allowable in certain defined state

leading to another determined state, where the channel capacity is the logarithm of the largest real root of the determinantal equation, i.e., the "spaces" of matrices with an undetermined upper bound on their ranks).

Onward, "2. The Discrete Source of Information"[336] is just about the most relevant paragraph related with topics in programming languages and, consequently, with computer science as it stands today. This is so because it engenders the assumption that the logarithm of the number of possible signals in a discrete channel, increasing linearly with time, establishing a rate of increase, in terms of required bits per second, escalates to knowing the quantity of information produced from each given source, bearing strongly on the statistical effect. Shannon's view was to understand that randomness shares some arbitrariness, and arbitrariness shares some randomness. Hypertext and both *mathematical* and *dynamic* structuralism have a natural imprint on the sequence of the alphabet economy, and thus, also on its information encoding (visible, for instance, in the scrabble-type shortest channel symbol – a dot – for the most common alphabet letter "E", while the less common letters are composed with long sequences of dots and dashes, but also in mechanical terms, for we can say that in pure anthropo-phonological terms, where the *scaphandre* is our body, and the *papillon* our tongue<sup>21</sup>, the open vocal "A" corresponds to the primeval scream, and all the other vocal closures correspond to the remaining vowels, while consonants are specific constraints operated by the tongue. Thereupon, a communication system as it the alphabet, is naturally also a physical mathematical discrete source, stochastic in nature.

Again, Claude Shannon invites us to consider the following cases: 1) natural written languages; 2) continuous information sources rendered discrete by quantification, say the standard form of digital audio, or a quantized T.V. signal; 3) mathematical cases with defined stochastic processes which generate sequences of symbols [336].

---

<sup>21</sup>The image is rescued from the well-known book *Le scaphandre et le papillon*(1997), an original autobiographical book by Jean-Dominique Bauby, a patient with the locked-in syndrome with speech disabilities – wherefrom it is suitable to use the image of the *scaphandre* as the locked-in body, and the *papillon* as the eyelid movement to pause the lettering of a moving (computerized) alphabet. The book was, a decade later, adapted to film, with the English title *The Diving Bell and the Butterfly* (2007).

Attention is systematized on the latter, and five letters, each of which with probability .2, with successive independent choices, extracted from a table of random numbers representing the letters, serves as an example:

BDCBCECCCADCBDDAAECEEAAABBDAEECACEEBAEECBCEAD.

In turn, now, using these five letters now given the, respective, probabilities .4, .1, .2, .2, .1, still with successive independent choices, a typical rendered message follows as such:

AAACDCBDCEAADADACEDAEADCABEDADDCECAAAAAD.

An important step up in the ladder of complexity is when the structure is obtained by adding the probability values made dependent on the preceding letter, but not the ones before. A set of transition probabilities  $p_i(j)$  states that the letter  $i$  is followed by letter  $j$ , ranging over all the possible symbols. Another possibility is  $p(i,j)$  which is reciprocal of the digram  $ij$  or its relative frequency. The probability of the letter  $i$  is, thus:

$$\sum_j p(i,j) = \sum_j p(j,i) = \sum_j p(j)p_j i \dots$$

which is, roughly very similar to the recursive status of the  $\lambda$  operator in  $\lambda$ -definability and induction, which goes the same as stating:

$$\sum_{i,j} p(i,j) = 1$$

Nevertheless, the values of probabilities need not be the same everywhere, and matrices of fractioned values can be found for each digram (trigram, and so on for whichever  $n$ -gram cases), always in list form. One such arrangement of the form from  $n$ -gram would bear the  $p(i_1, i_2, \dots, i_n)$  statistical structure. Consequently, "words" are subject to be constituted as stochastic matrices, with associated probabilities, such as in the example given by Shannon [336]:

.10 A	.16 BEBE	.11 CABED	.04 DEB
.04 ADEB	.04 BED	.05 CEED	.15 DEED
.05 ADEE	.02 BEED	.08 DAB	.01 EAB
.01 BADD	.05 CA	.04 DAD	.05 EE

A typical message under these "words" and associated probabilities structure is nothing but an artificial subgroup of the so called natural languages, such as, for instance, any of the five most spoken languages of the world (Mandarin, Spanish, English, Hindi, and Arabic). This encasement permits us to think of programming languages as a sort of (both *abstract* regressive and progressive) series into more formally constricted "universals" and "types" from natural languages, and natural languages as a sort of a very coherent, complex, unflinchingly game-like in nature (in Wittgenstein's "language-game" sense [418, 419]), indexed random process, and indeed, a strong artificial *constructum*.

In some sense, it is, alike Shannon's interpretation, the equilibrium between the *noise* [336] of natural languages, and the coherent information for each vocalic *n*-gram discrete parts as theoretical maximum information (phonemes), here with memory as expanding new (polysynthetic) series, what permits natural languages communication success, at vertical (in synchrony) and horizontal (in diachrony) transmission ("transfer rates" [336]). This picture is shown more or less along clustering patterns of areal topology, i.e., structural convergence, as it is induced by the communication of speakers whose *competence* (Chomsky) (*langue* in Saussure) and *performance* (Chomsky) (*parole* in Saussure) does not depend on, nor descends from, a common ancestor language. (Excluding William Jones' (1746-1794) and Thomas Young's (1773-1829) foreseen, or later established Franz Bopp's (1791-1867) imprint of one such vast language family of several hundred languages and dialects as the Indo-European languages). This picture triggers also the reckoning of the human-specific (highly complex and emergentist), and even, to some extent, animal-specific (lowly complex and reductionist) language (informational) agents, as exhibiting the characteristics of a channel transmitter, with electronics being substituted by physiology, and where wavelengths (audio- and psycho-linguistics) and mechanical filters (phonetics and phonology) are, literally, the differentiation and integration basis for the omnipresent "noise", bluntly put, *sonus universalis*).

In fact, Ferdinand de Saussure's (1879) proposal of *coefficients sonantiques* [99, 317], in account for vowel length patterns in Indo-European languages, leading to the laryngeal theory (with phonological "ablauts"<sup>22</sup> generative, anthropological,

---

<sup>22</sup>Ablauts are systematic vowel variations in the same root or affix or in related roots or affixes, more often than not in the Indo-European languages.

historical, and comparative morphophonology, accounting for a sort of *differential* incompleteness) in materialistic *sonus universalis* (also in terms of *sonus et sonitus* vector spaces), has potentially a direct *structuralist* parallel with programming languages (with constraint-based logic, functional, quantitative and cognitive *integral* consistency) now in extant programming languages different types of formalism. Often unnoticed and vitally needed to be expressed is the correspondent status of the subject of programming languages with evolutionary linguistics, specially in horizontal correspondence (in diachrony). This translates into Shannon's logarithmic stochastic, and Saussure's typical random arbitrariness, but also phylogenetically deterministic, applied and experimental neurolinguistics powers (under which the passage from *universal sonus* to *existunt imagines*, for which the consideration of  $\cup$ -Mentalism is made *ad imaginem*, is rendered easier to sense and perceive).

Not forgetting the importance of space (and equivalent silence), which enables the successive "words" constituencies, from the previous structure with associated probabilities, we achieve something on the lines of what follows [336]:

DAB EE A BEBE DEED DEB ADEE ADEE EE DEB BEBE BEBE BEBE ADEE  
 BED DEED DEED CEED ADEE A DEED DEED BEBE CABED BEBE BED  
 DAB DEED ADEB.

These artificial-(natural) languages comply, thus,  $n^{th}$  order approximation to its syntax and semantics aspects, and overall pragmatics, expandable from the exposed case (in what respect zero-order approximations were obtained with all letters with the same probability and independently; first-order the same by choosing successive letters independently, but each letter having the same probability that it has in the corresponding natural languages; second-order, introducing the digram structure, wherein the next-to letter is chosen in accordance with the digram frequencies  $p_i(j)$ ; third-order, consequently with trigram probabilistic structure with each next-to letter depending on the previous two letters; etc.). Still respecting this order, if the passage is made to represent words instead of letters as  $n$ -gram principal built structure (re-convoking first-order approximation with symbols, but now words), but with the frequencies of the English language independent, and in higher-order (with word transition probabilities, but with no further structure introduced), the result is strikingly similar to this very same text the reader has

in front of him/her, or indeed any modern English text (the same being true of any language, as English is here called upon as an *infinum ad supremum* example of spoken and written language and, inherently, a communication pattern).

"It appears then that a sufficiently complex stochastic process will give a satisfactory representation of a discrete source." [336] Stéphane Mallarmé *Un Coup de Dés Jamais N'Abolira Le Hasard* (1897) and Jorge Luis Borges "*La biblioteca de Babel*"(1941) creative literary visions lurk in, once one such statement is read, and it is no wonder that in Shannon's investigation at "7. *The Entropy of an Information Source*" [336], James Joyce's *Finnegans Wake* (1939) is presented as the most extreme case of low redundancy, and, therefore, higher self-cryptography, once large "crossword puzzles"[336] are only possible with low redundancy<sup>23</sup>. This is elucidative of the problem of inductivism, constructivism, statistics and probability on one side, facing on the other side mathematical structuralism and even (eventually digital-computationalist) Platonism, foreshadowed already in the confrontation between inherent structuralism, beyond mere formalism, of synthetic *a priori* judgements in Kantianism, versus Humean skepticism. The case is, of course, of Humean supervenience [239] above, not only mathematical Platonism and structuralism, but also above all probability or statistics (including Borel's

---

<sup>23</sup>In point of fact, Chebychev's (1821-94) well-know student, Andrey Markof (1856-1922) himself, worked with probability and arithmetic towards poetry and literature. "By replacing the vowels and consonants in Pushkin's *Eugene Onegin* by the respective letters *v, c* he generated a sequence with just those two symbols. In the original Cyrillic alphabet, vowels formed about 43% of the text. After a vowel, another vowel occurred some 13% of the time, while after a consonant, vowels arose 66% of the time. To predict whether the next symbol would be *v* or *c* he discovered that, given the current symbol, he could effectively ignore all its predecessors, so little help did they give." [167] This observation led to the appraisal of random varying sequences with independence towards past queries and series, as holding the Markov property. With roots in Bernoulli trials and the law of large numbers – basically examining the proportion of successes in a sequence of trials, not only estimating the chance of success, but also the reliability of the size of the sample – one such law frequency or probable success eventually prompted Borel's (1871-1956) strong law of large numbers (very alike proofs by induction and arithmetic series itself), which states that, given sufficient time and for any given tolerance band, there will be a moment in which the actual frequency of success or probability complies within the band forever. Even if we have just scratched one such notion in readying style, the so called Borel (positive) measure [6, 2], implying the uniqueness of limits of sequences, nets, and filters, with an associated  $\sigma$ -algebra generated by the closed or open sets, states that an  $\mu$ -measure is true of any topological space  $X$  so that functions  $\mu : \rightarrow [0, \infty]$  are countably additive. Supposedly, thus, a Borel measure on  $\mathbb{R}$  is right-continuous and increasing with convergence both in filters (special subsets of a partially ordered set) and in nets (with great digital-informational repercussion), on the common mathematical background as of  $\mu$ -recursive functions.

measure, and the fact that "the sun will rise tomorrow" [191] as hypothesized by Hume).

But to this we must add that the balance between these two poles, in a close sense to that of *gravity*, has changed, after the incompleteness theorems [161, 160, 312, 280] (Gödel, 1931), towards a much wider sense of contingency, and, reciprocally, non-trustworthiness of synthetic *a priori* judgements and, in extension, non-trustworthiness of the powers of reason. This might seem contradictory, as one finds many to believe the information age to have licensed a reentering of a (digital and cybernetics) *Aufklärung*, even if the motto *Sapere Aude* was lost with the derrocade of Hilbert' program [420].

In some sense, it is fair to avow the view that, no matter how much computationalism has overpassed the boundaries set by computability theory, complexity theory in computability is too great an achievement, comparing to the lack of complexity study in classic philosophy, even when confronted with the new physics. Maybe classic philosophy of time (as in Kant or Hegel) could welcome the mathematical-bounded hierarchies of complexity theory, and even matters with mathematical bounded relations as inductive interdependent series in probability theory, would benefit from studying the contingency presupposed levels in "Humean supervenience" [239]. Conversely, programming languages with assemblers and compilers are diagonal cryptographic machines, once it would be impossible to interpret (absolute redundant) machine code if it wasn't for discrete "crosswords" [336], whereby a word is defined (alternatively "word size", "word width", or "word length", accordingly a measurement theory insight) here with "words" meaning the natural unit of data used by the processor design, fixed-sized data handled as a unit by the instruction set and the hardware of the processor (as in fixed or floating point numbers, addresses, registers, memory transfers, unit of address resolution, instructions), after the 8-bit (or 1-byte) (IBM System/360 design), thereafter typically exponent of base 2 and multiples of 8, as seen with 16-, 32-, and 64-bits in general-purpose computers. Indeed, programming languages, interjected by compilers (from high source languages all at once to low-level assembly language, or even down to machine or object code) or interpreters (from high source languages and at interval time to intermediate code), and assemblers (from assembly language to machine or object code), are susceptible of being designated

hierarchical levels of a much wider philosophical argument which reposes at heart in the so called (diagonal) "gödelisation" [161, 162, 163, 280, 160] argument, which, in turn, is nothing more than a piece of a much larger subject, i.e., cryptography (in the exact surmounting point, over which it is not just anymore mathematics itself, and it complies with Claude Shannon's use of information at its *greatest lower bound* dialectic – ergodic and asymptotic – interpretation, and wherein an imagined, along Kant's idea of imagination, *lowest upper bound* corresponds to an ontological, truly metaphysical, interpretation)<sup>24</sup>.

---

<sup>24</sup>Inasmuch as with U-Mentalism what counts is not anymore so much the way through which abstractions are brought about in critic transcendental philosophy – from the transcendental unity of apperception (Kant, [A107]), then to the pure categories of the understanding (Kant, [A129-A130]), and, finally, to the transcendental *schemata* of the principles of the understanding (Kant, [A137-A147 B176-187], in overall (top-to-bottom) idealization of empirical concepts to pure formalized mathematics as in *a priori* synthetic judgements –, but the inverse, that is, exploring how each uniquely produced symbolic image (*imago, imaginis*) (not necessarily digital, but empirical) (bottom-to-top) as a transcendental unity of any possible judgment, can constitute an (eventually programming) mind and alphabet in different (*schema, schemata*), so too cryptography (more emphatically so after our understanding of diagonalization and gödelisation) can and should be affected by one such displacement and permutation. In other words, cryptography should not be deterred to the constricted spectrum of digital transactions - such as constructing and analyzing protocols that prevent third parties or the public from accessing private messages, or related issues of information security, data confidentiality and integrity, authentication and non-repudiation, etc, including cryptology, i.e., "the art of code-making and code-breaking"[301] - but should, instead, be considered as utterly *ontological*. Surely, cryptography blends mathematics, computer science, electrical engineering, communication science, and physics at its *procedural* level, but, nevertheless, at its *declarative* ontological status, it could be said that the (free associative) idea of any (empirical or pure) being, as a symbol, *power* to be a representation for any other (empirical or pure) being, is one major philosophical shortcoming if not visualized. While it is worthwhile taking notice that U-Mentalism does rely on a balance between perspectivism and naturalism, as in between conventionalism and indispensability, mathematical formalism and structuralism, for the sake of the argument, we are addressing cryptography attending to the fact that Shannon himself worked on the subject very closely with the concepts of information and entropy, in the book *Communication Theory of Secrecy Systems*[334] (1949) and, on the earlier version of this research, in the classified report *A Mathematical Theory of Cryptography*[333] (1945, declassified version 1949), published under Bell Laboratories. By this case in point, we understand how much cryptography envelops Shannon's approach to information theory. Indeed, it is not just Alan Turing's (1912-1954) chief case of conducting work (with the electro-mechanical device known as the *Bombe*) deciphering of the Enigma portable (cipher) machines with rotor scramblers in Bletchley Park, England, that endows an apprehension of the latent intersection between computing, diagonalization, cryptanalysis and information theories. Very keen to these grounds, examples coming forth from Linguistics, such as Jean-François Champollion's (1790-1832), founder of Egyptology, deciphering of Egyptian hieroglyphs, and Michael Ventris' (1922-1956) deciphering of the ancient Mycenaean Greek script now known as Linear B, are often disregarded in relation to computation, even though they



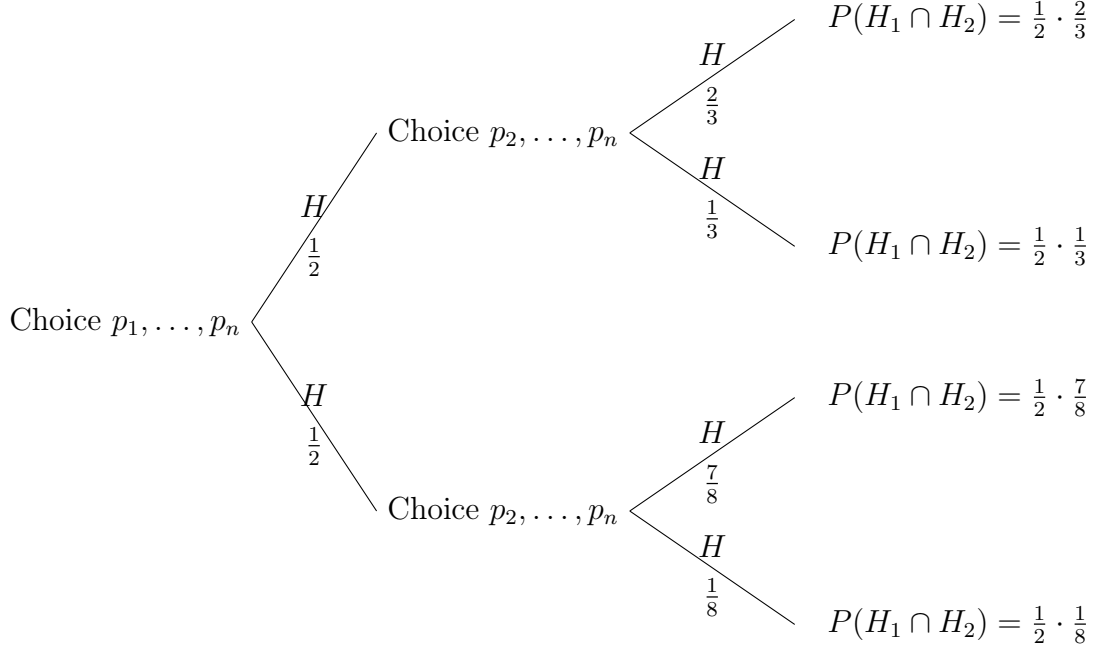
"5. Ergodic and mixed sources"[336] is the next subject treated by Shannon, wherein not only the treatment of Markov processes and graphs is shown in relation to every statistical (and circuitry) sequence, or lettering and  $n$ -gram frequencies, but also at which point it is made clearer the point of articulation wherefrom the branch of mathematics that studies dynamical systems with invariant measures, gradually shifting towards statistical physics and homogeneity, subsequently is found to be transported to a fairly *dialectic* meaning, therefrom more suitable to computer architecture. For this reason, this paragraph, right before the opening to both "6. Choice, Uncertainty and Entropy" [336] and "7. The Entropy of an Information Source" [336] is, most definitely, and beyond the mere statement of periodic structures or averages over the ensemble, against the probability of a discrepancy being zero, realizing equilibrium conditions and stationary processes over Markov processes as information sources, can be conveniently faced with Norbert Wiener's *Cybernetics, or Control and Communication in the Animal and the Machine* [404] (1948, -61) and its ensuing popularized version *The Human Use of Human Beings* [405] (1950, -54). More narrowly, the chapters "II. Groups and Statistical Mechanics" [404] and "III. Time Series, Information, and Communication" [404] sufficiently hold for the appraisal, but in convoking them what is put forward is the whole spectrum that Wiener draws from the very first chapter "I. Newtonian and Bergsonian Time" [404] until the closure of the two complementary chapters, more importantly so the last named "Brain Waves and Self-Organizing Systems" [404].

Plotted in the latter paragraphs (6.-9.)[336] cited in Shannon's work is, thus, the black box of mathematical information theory. It extends discrete noiseless systems to a full-blown and well-defined concept of entropy. Imaginably, if there is a set of possible events with the occurrence probabilities  $p_1, p_2, \dots, p_n$  the choice/uncertainty involved in the selection of the event, translates into a sort of equivocation measure and monotonic increasing function  $H(p_1, p_2, \dots, p_n)$ , where the original  $H$  is the weighted sum of the individual values of  $H$ , with the obvious correlate "diagonalization" of useful information (the different individual values of  $H$  can be parcelled into successive choices with associated probabilities; accordingly, each branch in the graph must correspond to a choice in order).

---

partake inordinately common attainments.

We adapt here an hypothesis with set theory intersection, propositional calculus conjunction and probabilities product, under the general requirement that  $H(\frac{1}{2}, \frac{1}{2}) = H(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}H(\frac{2}{3}, \frac{1}{3}) + \frac{1}{2}H(\frac{7}{8}, \frac{1}{8})$  i.e. with the total value of 1, and with the natural coefficient for each choice branch.



With this in mind, the quantities of the form  $H = -\sum p_i \log p_i$  or the equivalent  $H = \sum p_i \log \frac{1}{p_i}$  serves to define, clearly inspired in statistical mechanics,  $H$  as the entropy in the system. Also to be noted, in attention to Turing's terms in defining formal computability [363, 362] (1936), and von Neumann's awareness of computer architecture design [380] (1945), providently in relation to both authors' study of the physiological-biological processes in parallel (with a strong symbolic topological variance, as might be statistics in mechanics<sup>25</sup>), is the assumption that

<sup>25</sup>As a matter of fact, Shannon points out  $H$  of (mathematical-informational) entropy as "the  $H$  in Boltzmann's famous  $H$  theorem"[336]. The  $H$  theorem, introduced by Ludwig Boltzmann in 1872 [111, 374, 375, 384, 315, 369], describes the tendency to decrease in the quantity  $H$  (molecules with kinetic energy) in the long-time behavior of a dilute classical gas according to Newton's equations (in the context of XIX<sup>th</sup> century science) proven to be asymptotically Gaussian, imagining the molecules bouncing off against each other in abstract/ideal gas – the Boltzmann equation firmied the idea that the "differential"  $d^3\mathbf{r}$ , at the position  $\mathbf{r}$ , and with the momentum nearly equal to a given momentum vector  $\mathbf{p}$  (thus occupying a very small region of momentum space  $d^3\mathbf{p}$ ) in some instant of time within the probability density of the function – in the core of dynamics of rarefied gases with very technical apparatus (an equation model with

---

time-dependent density, position, velocity, transport and collision operators, collision kernel, and pre-collisional velocities, with local scalars and vectors, with also local parameters of density, velocity, and temperature) established that both the entropy and the time-derivative were non-decreasing and non-negative. This meant macroscopic irreversibility in large-time limits, and, therefore, a direct connection with the second law of thermodynamics, even if theoretically it is more correct to say that it is (almost zero *probably*) erroneous on the macroscopic level, tested only in ideally close to equilibrium, spatial homogeneous or vacuum systems *ab contrario*. Moreover, it is hypothetically disproved if three-local parameters (density  $p$ , velocity  $u$ , and temperature  $T$  varying the velocity distribution) in covariance, and at each position, could show at least a local equilibrium of the sort of a stable velocity distribution, as if *infimum* locality would hold for a sort of helical symmetry with *supremum* organization as we find in ADN structure. In other words, for each locality, there would have to be something equivalent to a Leibnizian monad, hidden as a spectrum of observable reality, which is quite interesting to note, once Boltzmann vehemently defended atomistic views, although not quite atomist (at least of the Leibnizian sort). As, along the lines of probability itself, this is somehow improbable and, at least in normal distribution, against the time series limit implicit in Gaussian functions, what counts is molecular chaos from, precisely, the consideration of *a priori* (spatial symmetric) microscopic probability distribution.

Therefore, in Boltzmann's conception, the following formula is presented:  $H(t) = \int_0^\infty f(E, t) \left( \ln \frac{f(E, t)}{\sqrt{E}} - 1 \right) dE$ , where the function  $f(E, t)dE$ , is the energy distribution function of molecules at time  $t$ . The value  $f(E, t)dE$  is the number of molecules that have kinetic energy between  $E$  and  $E + dE$ . As this quantity  $H$  was meant to represent the entropy of thermodynamics, the  $H$  theorem implied a parallel speculation, or rather a very fertile *ansatz*: that of the Boltzmann brain. The Boltzmann's brain, remembering that entropy translates how far from equilibrium a system is – the lowest the kinetic energy (as the number of microscopic configurations  $\Omega$  *adequate* to supposedly equivalent thermodynamic macroscopic variables), the further from equilibrium the system image is –, accounts for the functional and sentient self-aware entity that all of us are, born out of random fluctuations out of the very same thermodynamic equilibrium system. With this being said, and thinking in terms of energy dispersion, wherefrom more heat and motional energy, in a closed system, induce higher self-contained disorder, all of the previous paraphrases the essence of both the universe and the mind (in Cartesian *res extensa et res cogitans*). Put in another way, inside the observable universe (not the universe as an isolated system) as an open system with exchanges of both energy and matter, common to the existence of the universe has come to exist the observance of the universe (and with memory and intelligence with the power of modelling the universe itself), which corresponds to a very unlikely and improbable non-equilibrium state, with both the exceptional existence of the universe and the mind, here understood as the most randomly and non-independent occurrences that probability series can bring about.

A brief sketch coextensive with  $\cup$ -Mentalism is here recommended. It suffices to allude two notes: first, in abstract, the idea of a "volume of microstates" [374] with the technical fetch typical of physics and its branches, is very recommendable to apply in relation to programming media, specially attending to the possible practicality of *schemata* born on a physical model, and also to the multitude of existing images in cosmology, to which  $\cup$ -Mentalism sets to represent, just so, the highest possible commonest probabilistic ground for both the mind (in *schemata*), and the universe (in *imaginis*) in computable form; second, Boltzmann's philosophy of science is, truly, an example of the *representational* temporal and historical rooted, inconclusive, tentative, rather describing than explanative, approach to scientific truth. One such notion brought about

"(...)  $p_i$  is the probability of a system being in cell  $i$  of its phase space." [336].

$H$  is, thus, trusted as the measure of choice or information, at most times positive or a joint occurrence of (at least two) events, where  $i$  for the first event  $x$  and  $j$  for the second event  $y$ , bundle as  $H(x, y) = - \sum_{i,j} \log p(i, j)$ , defined as it is the entropy of the joint event, with the correlate assumption that the "uncertainty of a joint event is less than or equal to the sum of the individual uncertainties." [336]. Accordingly, any change towards equalization of the probabilities increases  $H$ , which translates to saying that an "averaging" [336] of information (as in coding and decoding correlating together in "averaging" [336], or programming languages meeting machine code) is an informational gain due to overloading "control" of entropy – ahead and in parallel, a key concept in logic programming and in Prolog, with Prolog considered an AI artificial cells "brain" – herein considered the conditional entropy of  $y$  depending on the average entropy of  $y$  for each value  $x$  suiting  $H_x y$ .

Hence, they are directly proportional, which might seem counter-intuitive (on the basis that it is not expectable that from higher entropy follows higher information): the certainty of  $H_x y$  is the information – base 2 bits, base 3 trits, base 10 Hartleys, base  $e$  nats – so that information is measured as, precisely, this simple "averaging" [336] expectancy from the event, and in accordance with the number of outcomes, or its mass. In other words, information adheres to how much we need to relay to describe a probable event, and the more expectable and common (in full Humean inductivist sense) the event  $H_x y$  is, the less information outcome is provided. Indeed, complexity theory affirms that a computer, as a (nearly) closed system, cannot reduce its disorder; it can only remain unchanged or increase, most especially a stored-program, no matter how proficient and balanced, in terms of ergodic and informational theory, it might be. Should this be true in relation to the history of programming languages and associated programs, and, itself, to the future of programming media, is a great question to be asked. On broader lines, a sharp topological understanding is granted if we imagine the average amount of information produced by a stochastic source of data as mass, and the measure of information entropy associated with each possible data value the negative logarithm

---

the idea that "scientific theories are nothing more than images of Nature." [315]

of the specific probability mass function for the value. Thus, when any data source has low probability tenability, this specific event, say  $H_{xy}$  for a line of code, carries more information than when the source data has a higher-probability value. The information *channelled* – at a definite time per second rate if the Markof process is also definitive with an average frequency – by each event becomes, therefore, a random variable whose expected value corresponds to its entropy. According to the law of large numbers, it is fairly certain for any given case "(...) to have  $\frac{\log p^{-1}}{N}$  very close to  $N$  when  $N$  is large." [336], which presupposes a limit boundary that is also a list with numbers of bits per symbol specifying the sequence. In a rather elegant expression,  $\lim_{N \rightarrow \infty} F(N) = H$ [336]. The passage to proper understanding of the "8. Representation of the Encoding and Decoding Operations"[336] and the "9. The Fundamental Theorem for a Noiseless Channel."[336] is the statement that "(...) the ratio of entropy of a source to the maximum value it could have while still restricted to the same symbols will be called its *relative entropy*. This (...) is the maximum compression possible when we encode into the same alphabet. One minus the relative entropy is the *redundancy*."[336]. A reasonable estimate for programming languages, notorious for its specific focus on syntax is, at least, 75%.

In "8. Representation of the Encoding and Decoding Operations."[336] we find the passage more consecutive to the Turing-machine model in the idea of a discrete transducer, based on the fact that it holds an internal memory and the output is a sequence that might relate to its past history ( $x_n$  for the  $n^{th}$  past input symbol, pushing the finite internal memory transducer to behold the state  $\alpha_n$  when one such symbol is introduced, producing the output sequence  $y_n$  potentially in tandem, and where the state of the source and the output symbols from the transducer form a "product state space" [336]). As it is seldom free, and general constraints arrive at it, the constraints, considered as a channel, follow the logarithm of the weighted entropy sum. And, at the point where this *dialectic* turns effectively *practical*, "by the proper assignment of the transition probabilities the entropy of symbols on a channel can be maximized at the channel capacity." [336] Inward, the "9. Fundamental Theorem for a Noiseless Channel"[336] shows up as fundamental, stating that  $H$  is not only the rate of information, but also, in procedural and statistics essentials, the determinant within limits, as a sort of *integral*, of the

channel capacity required that the most efficient coding is performed. Thus, let a source have entropy  $H$  (bits per symbol) and a channel have capacity  $C$  (bits per second). From this Shannon draws a conclusion on the focal point of optimization, like the subject of computer architecture in von Neumann's work, that "then it is possible to encode the output of the source in such a way as to transmit at the average rate  $\frac{C}{H} - \epsilon$  symbols per second over the channel where  $\epsilon$  is arbitrarily small. It is not possible to transmit at an average rate greater than  $\frac{C}{H}$ ." [336]

Now, converging to von Neumann's paper *The First Draft of a Report on the EDVAC* [380] (1945), it shall, prior to that, be made explicit how connected the forefront discussion of information [336, 53, 8, 386] and entropy [384, 369, 315, 292] was to the Hungarian-American scientist, and thereupon cast how much computer architecture communicates with the mind and physics, apart from control, once even entropy's very first inception against the informational background was his, in advising Shannon<sup>26</sup>.

After one such debate, it definitely puts simple expressions and concepts in a different light, such as that of pre-transistor "vacuum tubes" [380] (4.3; 5.5; 6.5) (a device controlling electric current between electrodes in an evacuated medium container, for the most part relying on thermionic emission of electrons from a hot filament or a heated cathode, the thermionic tube, a configuration that is not exclusive, as electron emission might be achieved also through photoelectric effect, or even non-vacuum and saturated gas-filled tubes generally at low pressure). From the diode (the simplest of the vacuum tubes, with only a heated electron-emitting cathode, a filament and a plate, the anode), with unidirectional current flow, it is not just the representation of vacuum and its inherent embodiment in a closed system, according to the physics laws of thermodynamics, that is charted. It also brings forth a strong analogy with information in its (machine-analog) embodied version, with entropy in both a cosmological-physical and local-computational

---

<sup>26</sup>"For those who believe that entropy has always been a crystal-clear concept, let me recall a famous quote by Shannon: '*I thought of calling it 'information'. But the word was overly used, so I decided to call it 'uncertainty'. When I discussed it with John von Neumann, he had a better idea: (...) 'You should call it entropy, for two reasons. In the first place your uncertainty has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, no one knows what entropy really is, so in a debate you will always have the advantage.*' [374]

vertices, picturing, thus, *excitatory* [380] or *inhibitory* [380] actions over both the brain-model (*repraesentatum esse*) and model-computer (*repraesentatum id*). It displays *synchronous* or *asynchronous* spacetime (rational cosmology in Kantian terms), I or the mind (rational psychology in Kantian terms) and Turing-machine (*ad hoc* rational teleology in Kantian terms) (machine-analog) "time-clocks" or, better said, (mind-continuous) philosophies of time.

With this we mean to weigh in an unavoidable extension to the Boltzmann's brain paradox, showing its engulfing enlarging antinomy with a central disparate nucleus that is now the computer, the Turing-machine model or, more readily perceived, the standard von Neumann computer architecture. Simplifying, as we assert that computability (Turing-Church) and the incompleteness theorems (Gödel) oblige to reevaluate Kantian antinomies, so too the Boltzmann brain paradox should be revisited.

Indeed, if it was already astonishing that the universe, in itself a cosmic lottery, could have hypothesized a general thermodynamic equilibrium with exchanges from first cosmological *structures* to far-reaching biological-chemical-genetic *lists* to form a genealogy of brains as a complex (self- and outer-)aware entity that has arisen due to random fluctuations, what were really the chances that the brain could have produced, on phenomenological grounds, one such artifact *object* mirroring τέχνηε (*techne*) and μηχανη (*mechane*) into itself (the Turing-Church thesis in a machine)? Not only that, but, more astonishingly, what were the chances that with this, in expanding or evolving symmetry (in fractal *autopoiesis*) now in a sort of ἀρχη (*arche*) artificiality – *computus Archeozoic* (Earth in the comparison), or *computus Arche* inflation and digital synthesis (Universe in the comparison) –, both the computer (fundamentally the concept of *U*-machine and the stored-program on recursion and computability) and the brain complex and concomitant existence, were therein reflected, and therefore, discretely, that of the Universe too? Briefly, we are attending to an extension, impossible to have been made both in Kant's and Boltzmann's days, that would translate to a computability-based version of the Boltzmann's brain. Its construct targets an *extensional aesthetic* rational cosmology and one *intentional practical* rational psychology. It is with no surprise that this valuation comes with a fair notice on Kant's teleology and its

contrary pole of mechanism, moreover reflexive on Kant's theoretical note on nature's purposiveness for our cognitive faculties, all in all under the backdrop of the now classical concepts of information and entropy<sup>27</sup>. Peculiarly, it is the concept

---

<sup>27</sup>There are a multitude of nuances in this appraisal that are worth mentioning [291, 294, 292, 130]. Under the context of a universe with a measure of energy with the tendency to run down, with high entropy suggesting less energy availability, encompassing, in perspective, the arrow of time, however this very same arrow of time would have to be longitudinally expensive (with either a great time line or great structure economy) towards infinity, so that given both the *excited* [380] and *quiescent* [380] states of entropy (sort of its "forward" and "reversing" time phases [292, 369]) were to produce, alike the Turing-machine model, given enough time and memory, at least twice a Boltzmann brain, it is not quite the fact that on top of this, one such evaluation is shared amongst a community of billions of living minds *hic et nunc*, but more so the fact that in the cosmic sea full of randomness information, this mind/brain, in apparent duality, is capable of modelling its living paradox by having the capacity of philosophizing in the first place, with an absurd waste of energy for apparently zero exchange, what is most scanty unravelled. It is also true that from Kant's sense of rational theology, being pure reason in its *practicality* a matter for the receptivity of faith, inasmuch as the will to pursue moral goals, entropy becomes very problematic to every rational theology (more specially so for the sort of Spinoza's panentheism, forasmuch as transcendental idealism bears on some sort of materialism, even if *immaterialist* such as in Berkeley's philosophy, and God himself would be variable and entropic). Overall, it does not give any difference whatsoever from computationalist views affine with digital philosophy. Rational theology is driven from teleology, alike computationalist teleology is driven from theology.

Underlining Boltzmann's paradox, it could be said that if God existed (with his fabulous dispense of energy), the universe would not exist anymore ... if it wasn't for God (at which point it shifts from being a paradox to being a parody). In the case of digital philosophy, and remembering entropy's natural antinomy – "entropy is the state of any system as proportional to the *logarithm* of the number of different possible ways (also freedom-degrees) that state can be realized in" [292, 293] – allusion by Penrose, and Einstein's alert – "Usually  $W$  is set equal to the number of ways (complexions) in which a state, which is incompletely defined in the sense of a molecular theory (i.e. coarse grained), can be realized. To compute  $W$  one needs a complete theory (something like a complete molecular-mechanical theory) of the system. For that reason it appears to be doubtful whether Boltzmann's principle alone, i.e. without a complete molecular-mechanical theory (Elementary theory) has any real meaning." [124] – and remembering Boltzmann's sharp accent on Darwin's concept of *adaequatio*, computationalism (illegitimately surpassing Turing-Church computability) is best described as a philosophy whose faith lays in reversing natural entropic metaphysics (devolution) by means of artificiality (evolution).

We reinforce, for the sake of distinguishing sharply critical limits, that also in what relates to the concept of  $\cup$ -Mentalism, and its *causatum* towards entropy and information, in the frontiers of a proper educative, demarcate, and heuristic understanding, as in an *ansatz* (Boltzmann) or *gedankenexperiment* (Einstein), hoping to test for Turing-Church thesis effective computability limits, it is recommendable to accrue to the debate (natural and artificial, stochastic and programming) symbolic languages. Indeed, insofar as, *theoretically*, gas clouds envelop the cosmic surroundings of the (unobserved) *isolated system* of the entire universe, and inasmuch as gas clouds are, *practically*, the proper trade-offs for the (observed) *open system* universe exchanging both energy and matter, so too the frontier and communication for the mind – more vastly, the mind-body ecosystem [422], on the account of rational psychology, even if erroneously, taken classically as an (of the *a priori* kind) *closed system* – is nothing but language as a *medium*.



of architecture – from Kant’s to von Neumann’s sense, under the historiography of the concept – which adjusts best to this *magna quaestio*.

Before proceeding to the final remarks, it should be noted that we are treating von Neumann’s *First Draft* [380] as a thesaurus, i.e., a widely publicized glossary and reference book, a historical rightly-established and solid foundation document opening the field of computer architecture, far ahead and straightforwardly both expository and explicatory. Nevertheless, in our quest and interpretation, we find to be true that, in all the other aspects in computation theory, there is not any element to its incipient history that was not promptly theorized, when not thought-through in anticipation, sometimes by decades, by the British pragmatist, mathematician and engineer Alan Turing (1912-1954). In this regard, the momentum in philosophy of science (1936) of "computability" [362, 368] (Alan Turing) and "effective calculability" [71] (Alonzo Church) *in toto* serendipity and coincidence, is in every way very similar to both von Neumann’s *First Draft of a Report on the EDVAC* [380] (1945) and Alan Turing’s *Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE)* [364] (1945), this time, though, *apparently* with a greater expediency and ascendancy on von Neumann’s side.

A clear proof of that would be Turing’s terminology, as he chose to replicate von Neumann’s own technical phraseology. Indeed, right at the start Turing’s advice is that "it is recommended (...) that it be read in conjunction with J. von Neumann’s 'Report on the EDVAC'" [364] and, besides the acronyms – "IO" for input organ, "OO" for output organ, "LC" for logical control, "CA" for central arithmetic part, and "CL" for clock – there are several other unvaried pieces of jargon, such as "[7]External Organs" [364], or in taking repeated notice of aspects

---

From here, it is legitimate to think that our approach to  $\cup$ -Mentalism can transfer important understanding, for the most part related to artificiality acceleration and (natural-*world*)-physics-patterning-to-*schemata*-(artificial-*mind*)-languages, hoping with this to forge ahead one rational psychology advancement on the basis of programming media and programming languages.  $\cup$ -Mentalism is the *adaequatio* of each and every possible *idea* with each and every possible *image*, converging programming languages and computer architecture.

What is more, some classic arguments in metaphysics, can, with this turn, be expanded through language-related themes, such as Descartes’ misleading God (*Meditations on First Philosophy*, 1641) with cryptography, or, as another example, Leibniz’s *Théodicée* (1710) with time-complexity and probability.

such as that of delay-memory and length, or yet in focusing in "[16] Alternative Forms of Storage" [364].

It has to be said, overall, that there is one complexity articulateness line that was common to both authors, proof enough of the late confluence on unsolved problems in mathematics, a sort of Hilbertian transaction<sup>28</sup>.

Turing's paper *The Proposal* (1945), nevertheless, was openly directed to the ACE, in itself a rehash of the EDVAC – not forgetting that there is the von Neumann EDVAC, or "vN-EDVAC" [179], and the Moore School EDVAC, or the

---

<sup>28</sup>John von Neumann addressed the International Congress of Mathematicians in Amsterdam (September 2-9, 1954) and, despite "a manuscript was not available" [272] it bore the title "Unsolved Problems in Mathematics" [272]. The paper focused on operator theory, that is, a branch of functional analysis that studies linear (and non-linear) operators on function spaces (differential and integral), adjusting inevitably to the topology of function spaces, to the point that the minacious field of quantum theory was made neighbor to rings of operators. On top of this layer, and more importantly to the discipline of logic, von Neumann intended to draw some unification lines in logic and probability theory based on this approach [272]. Alan Turing, also in 1954, submitted the paper *Solvable and Unsolvable Problems* [364], a reflection on the furthest output layer of complexity theory, that is, the frontier whereupon a problem does not admit the existence of any efficient algorithm and, as such, is considered undecidable. From a very basic solvable problem, although not computationally – the sliding squares puzzle – Turing showed the minatory and impeding details. On top of this layer, Turing held focus on the so called word problem for groups, announced to be undecidable in 1952 by Novikov – in combinatorial group theory, the word problem for a finitely generated group is the algorithmic problem of deciding whether two words in the generators represent the same element, being insurmountable once for as one and only normal form in use, the result is always undecidable in relation to group isomorphism, i.e., for a finite group the relation was proven not to have any possible efficient algorithm, the same holding in general; this was a somehow expected result, in consequence of the unsolvability of finitely presented groups having been proved before, which also may have sent forth Markov's own proof next (1958) of the unsolvability of the fundamental homeomorphism (or topological isomorphism) problem [351, 262]. In addition, it can be said that while Turing was centered in algorithmic complexity, turning up, for that reason, more on time complexity, and forecasting the proper ground for the Kolmogorov–Chaitin complexity, algorithmic entropy, or program-size complexity (1963, -64), finally allocating oneness to Cantor's diagonal argument, Gödel's incompleteness theorem, and Turing's own halting or decision problem, von Neumann had convened more on space complexity instead, centered on a sort of "algebra of unbounded operators" [272] and wherein the superlative view was one such, quoting the Hungarian-born scientist "(...) where the dimensionality is like real numbers with a finite ceiling (...)" [272]. Minimally convoking the idea of  $\cup$ -Mentalism, where visualness meets programming media, it highlights through von Neumann's note both a *spatial* and an *orientation* radial point: "So in order to have logic you need in this set a projective geometry with a concept of orthogonality in it." [272]

The 1950's were, for what transpires, the blossoming of complexity theory. Let us remember that the *P vs NP* problem is recorded to have been first discussed in these years too, in letters from John Forbes Nash Jr. to the NSA, and from Kurt Gödel to John von Neumann, then finally formalized in 1971 by Stephen Cook [81].

"M-EDVAC" [179]<sup>29</sup> – but in the shade, we can not let it be forgotten that there had been put an effort with mastery success in the Colossus in absolute secrecy (*de facto*, so great of a secrecy that it involved the destruction of its plans in the 1960's, and the outlasting of the confidentiality status until the the mid 1970's).

The Colossus (1943-45) was a set of computation machines – a total of 10 Colossi were in use by the year 1945, while an extra eleventh was under commission – developed by British codebreakers from the Government Code and Cypher School (GC&CS) at Bletchey Park, England, in "a remarkable synergy of mathematics and engineering" [60] right around the years before computer architecture had its self-professed baptism (1943–1945) at the will and wise manipulation of cryptanalysts of the Lorenz cipher – itself a series of electro-mechanical rotor stream Vernam<sup>30</sup> cipher machines with in-line attachments to standard teleprint-

---

<sup>29</sup>Firstly, the Moore School Eckert & Mauchly's "Automatic High-Speed Computing: a Progress Report on the EDVAC" [198] was only turned "unclassified" in 1947, and the handwritten form of von Neumann's document was typed at the Moore School with neither Goldstein's recovering of the original document, nor with any of von Neumann's subsequent proofreading being known. Furthermore, the two EDVAC's under contention – von Neumann's *descriptive* form (curiously, not implemented in the Moore School, neither at the Princeton Institute for Advanced Study), and the *constructed* Moore School form – were different. They were both synchronous and sequential design machines, with a binary internal number system, but while "M-EDVAC" [179] was 44 binary digits per word, and 32 data bits per word, with 0 memory tag bits, 4 bits per command, and 10 binary digits per address, the "vN-EDVAC" [179] was 32 binary digits per word, and 31 data bits per word, with 1 memory tag bits, 3+5 bits per command, and 13 binary digits per address. What is surprising is that many reports on the Moore School EDVAC missed in detail these numerical system facts. Of course there were also other differences: in the physical part, the mercury acoustic delay line in media storage – a non-random, thus sequentially-accessed line refreshable memory, endowed with an amplifier and a pulse shaper in between the output or request and the response or input – was intended to be much higher in von Neumann's plan, just to mention one example.

<sup>30</sup>On the basis of trying to attain *perfect secrecy*, in one such scenario wherein a sender emits the message (the encrypted plaintext is called the ciphertext or cryptogram) to the intended receiver, with a cipher system (the set of rules used to encrypt information plaintext is the encryption algorithm) unintelligible to any third-party interferer, the highest possible expectation on the side of both the sender and the receiver to disguise information is that, in case the third-party interferer gets a hold of the message (the united *input* of both the encryption key and the plaintext message, but not the encryption key, nor the plaintext message), no one is capable of deciphering any information in the content of the message, in which case, though, an intelligence system of deciphering/decoding can still always operate, specially if, in the lines of (very high-order) complexity theory,  $P = NP$ . Indeed, the borderlines of the  $P$  vs.  $NP$  problem are just about the same that permit us to understand why it is true that *perfect secrecy* can only be obtained if the number of keys is at least as big as the number of messages, in which case perpetual guessing might occur. If we notice, the form [plaintext+key=ciphertext  $\implies$  ci-

ers, very few in nature: the Lorenz SZ40 (1941), SZ42a (1943) and SZ42b (1944) (SZ for *Schlüssel-Zusatz*, meaning "cipher attachment").

Concertedly, both Colossus (1943-45), in relation to the Lorenz cipher, and the specifically designed electro-mechanical Bombe (1939-40) machine to decode the electro-mechanical rotor cipher machines Enigma (1920's), appoint in parallel to unequivocal demonstrations:

They are both a diagonalization argument, a typical task-performance *experienced* IP-complexity class problem, insofar an interactive proof system consists also of two machines, a prover  $P$  (the Lorenz cipher, and Enigma machine), and a verifier  $V$  (the Colossus, including simulators such as the Tunny – an emulator of the Lorenz SZ42 cipher machine built at Dollis Hill in 1943 – and the Bomber) that checked their correctness, with the difference that the prover  $P$  was not infinite neither in computation, nor in storage, but, correspondingly, wherein the verifier  $V$  was a probabilistic polynomial-time machine with access to a random (bit) string whose length was polynomial on the size of the first machine's strings and formal language. To this we can add that it corresponded also to a finite group

---

phertext+key=plaintext] is not much different from the *hypothetical* form [ontology+law=nature  $\implies$  nature+law=ontology], which is the reason why Pascal (1623–62) not quite religious, but instead *practical*-teleology-fearful argument was placed as a wager. This also helps us to understand why – excluding monads and *La Monadologie*(1714) (Monadology sort of representing  $NP$  in case  $NP \neq P$  or maybe a consistency ultimate bound beyond NP-hardness; or yet non-decidable, or even rather non-recognizable problems in complexity theory, as far as the comparison to XVII-XVIII<sup>th</sup> philosophy and the induction of a hierarchy on classes defined by constraining the respective resources is legitimate – Leibniz's system was deistic, very close to the (although deterministic) idea behind  $MONADS \notin NP \subseteq PSPACE \subseteq EXPTIME$ , and how much of this fractionation welcomed Gödel's (mathematically and culturally) "context-sensitive" ideas on incompleteness. Analogizing, let us also affirm that while *La Monadologie* is metaphysical declarative, Platonist, monad-centered, logic-biased and reactive-reductionist, the idea of  $\cup$ -Mentalism and programming media is meant to be, in contrast, ontological procedural, perspectivist, mind-centered, image-biased and active-emergentist. If the monad was to be a determinist *centrum et medium* for the physical reality, we permit ourselves to think, instead, of each non-deterministic possible *medium et centrum* to hold for all the possible viewpoints and for each and every possible spacetime dimensions. In the "place" of each monad, we take hold for every possible viewpoint and for every possible mind, including God's or similar.

Conceptually, *perfect secrecy* embarks in the description of the one-time pad (OTP), requiring the use of a one-time pre-shared key (the one-time pad random secret key) the same size as, or longer than, the message being sent, with the notable difference that the Vernam cipher corresponds to its binary digital version, where the cryptogram is obtained via adding the message and the key modulo 2 (due to the short modulo number and also short remainders to cope with the 0's and 1's of the binary system).

representation of isomorphism being checked and tested, considering symmetric-key algorithms, i.e., same cryptographic keys for both encryption of plain text and decryption of cipher text.

The abstract types of resources involved in the Colossus computation (time, space, randomness, interaction, alternation, non-uniformity, oracles, complementation, non-determinism, and counting) – while Turing experimented *NP*-genre inductive (on a conditional object for YES) and *P*-genre deductive (on an unconditional structure for YES) solutions – faced up with automata theory finite, and strictly local fragments, extensively stretching the radius of complexity and automata organization up to context-sensitive (Type-1) and recursively enumerable (Type-0) hierarchy levels, are, really, computational dimensions that we have to inveigle, in looking back to the design of computer architecture, and also to the buildup of the proper computer machine. As a matter of fact, the frontier itself from typical (Type-1) linear-bounded non-deterministic finite state machines – wherein a grammar for every possible cryptographic formal translation production  $\alpha \rightarrow \beta$ , and wherefrom the length of  $\beta$  is larger than or equal to the length of  $\alpha$ <sup>31</sup>, really on the frontier of (Type-0) grammars without any restriction whatsoever in possibly generating, beyond decidability (a Turing-machine producing strings in the language and not producing strings not in the language) Turing-recognizable languages – is elucidating of the difficult overlap and complexity that was at stake at the time: an absolute sea of unknown. Literally, all possible languages generated and recognized by a Turing machine, i.e., recursively enumerable languages (and not just recursive), in any case possibly halting in full exemplification of the worst possible scenario, also particularizes, beyond doubt, the standard that computation had met by that period in what concerns the embranglement of complexity and automata hierarchies, linguistic and cryptographic hardness. All in all, Turings effort in computer architecture, beyond code, is, to say the least, as extraordinary as von Neumann's.

Under this appraisal, we have, thus, thought back to Shannon's contribution to the *Communication Theory of Secrecy Systems* [334] (1948, -49) in the context of

---

<sup>31</sup>The meaning of symbols in automata theory usually goes as:  $a$  = terminal;  $\alpha$ = terminal, non-terminal, or empty;  $\beta$ = terminal, non-terminal, or empty;  $\gamma$ = terminal or non-terminal;  $A$ = non-terminal;  $B$ = non-terminal.

the birth of *A Mathematical Theory of Cryptography* [333] (1945; declassified 1949) all abridged by the synoptical view of a *Mathematical Theory of Communication* [336] (1948), now interwreathed with the ascendancy idea of computer architecture both in Turing and von Neumann. It is fair to assert that the main idea resting on Shannon's work, while "diagonalization" was shown to be the proper computability inherent *practicality* in intelligent electrical-digital machinery short history, was that, and ever since long lasting, of *perfect secrecy* and the one-time pad [301] – first ever described by Frank Miller in 1882 [301] – in the exact sense wherein a cipher comes to be *theoretically* unbreakable (information-theoretic secure). By *theoretically* unbreakable it is meant the following: to have come forth in consequence of a pre-shared random, never reused in whole or in part, key the same size as, or longer than the sent message, having been paired, again in full *structure* unexpectedness, with the plaintext, using modular addition. For instance, taking, alpha-numerically, each letter *A* to *Z* to be associated with the numbers 0 to 25 in the same order, and for each message  $m_1, m_2, \dots, m_n$ , with the key  $k_1, k_2, \dots, k_n$ , the  $i^{th}$  element to the cryptogram produces:

$$c_i = (m_i + k_i) \pmod{26}$$

It is important to notice that the case is not that as in *NP* known problems solutions, wherein we know that solutions exist within (non-deterministic) polynomial time. In one such scenario, a correct hint would immediately produce a verifiable result, and possibly even a theorem. It is not so much a question of decryption not being possible, because it is, provably, possible. Instead, whatever the message taken out from the ciphertext with the same number of characters – in extreme, all possible range YES choices within verifiability, comparing with the *P* vs. *NP* problem – is regarded simply as one among any possible use of a different key, redundant in essence. It is of absolutely no use. Even if one possible found answer or decryption is semantically congruent (possibly valid), furthermore superimposed on any ordered *structure* of the world, and also with underlying epistemic logic of the subjects involved (possibly sound), there is not any possible way of knowing if that was the original plaintext message.

On the contrary, in encryption one-time pad, as long as the key modulo addition is truly random, and presupposing that nature is the chaotic stochastic generator of

large, truly random numbers ( $A \bmod \Omega$  in physics or natural philosophy, i.e., the *dividend* expansion and the *divisor* entropy in terms of *objects*, *lists* and *structures* in philosophy – possibly programmed – of mathematics), if a new "message" was to be emitted for each  $t, t', t'' \dots t_n$ , say for each second, for all the entire time of the universe (thus with both the message and the key growing each turn), it would definitely come to an halt, turning out to be *practically* impossible (even for a mind and as such constituted in the upper bound to that of humanity's) to decode any of the turnout "plaintext messages" (actually hardness would grow just as exponentially as in non-recognizability). This would be so recurrently now and ever, and dramatically impossible to decrypt, even if the chaotic and stochastic unknown generator method of nature would be, at some point in time – *fortuitously* (non-deterministic, weak *nominalist*, or strong *intuitionist*), in which case a deciphering method could exist based on time and experience, or *selectively* (deterministic, weak *formalist*, or strong *structuralist*), in which case a method could exist based on structure and patterns – here shown to be somehow equivalent. This is exactly the point wherein synthetic *a priori* judgments find their final fading erosion, and the question of "how is mathematics possible?" [211, 63] with the fine distinction of prevision, dies away.

It could be true that structures "cipher blocks" would facilitate decryption as they grow, but, just like the passage from the 1<sup>st</sup> to the 2<sup>nd</sup> of Gödel's results on the incompleteness of mathematics, a decidable decoding, insofar the key is truly random, and, at worst, as long as the "plaintext" is decisively (any-order) incomplete and undecidable, under the scope of one-time pad cryptography [80, 332, 177, 178, 95, 301] with ontology and metaphysics here playing the role of "information", its decryption is found to be impossible.

This, besides constituting an inversion of Kantian's whole *architectonic* [211, 63], demonstrates the need of digital ontology, as a sort of computability's *dialectic*, to institute a parallel running universe with computability as diagonalization, even though a fatal error is made in one such envisagement, for the reason that computability is here a just a fragment of the plain text at hand, not a cipher, little less a random key, and impossibly so, imaginably, the one-time pad secret key with the ontology's information on the world.

The most disturbing conclusion is, however, mostly violent to the well known and uncontested pronouncement by the Tuscan polymath Galileo Galilei – "The Book of Nature is written in the language of mathematics" in the modern version, and the Renaissance version "Mathematics is the language in which God has written the universe" – once if mathematics was taken to be the random key, mathematics would no longer be mathematics, which provokes Gödel's results to be newly valuated in a proper Gödelian sense, wherein mathematics is necessarily so incomplete, alike reason in Kant (fractured in *analytic* and *dialectic*), and necessarily so for a *strong* (secret) "mathematical reason". If mathematics is truly infinite and random, the random key (in the Renaissance version the key being pre-shared between the world and God, but in the modern version naturalism being absolute secrecy) would always be secret, even if hidden by the simple mechanism of modular addition, which leads us to think that computability (imagining recursive theory in the place of modular addition) even if constituted as the random key, could produce secrecy given enough time ( $P \neq NP$ ). But as time here is the time of the universe, there is no point in having an insecure (unknown) key to an almost infinite in time world and universe, and even if time acceleration was superior and entropy minor, both mathematics, and computability as a part of pure mathematics, would always be incomplete (insecure).

Time, as long as it is the same size as the key (the key possibly being even time itself accelerated to one *absolutus* of the Hegelian sort, and set apart), would itself be known (secure) and unknown (insecure) information and randomness *at the same time*, i.e., also in between, constituted as the message, in ontological terms. If, by any chance, meaningful to decipher the information-theoretic secure ontology of the world (conceding that "plain text" time never had re-used in whole or in part the "key" time – and, thus, time would not be time anymore following *substantia* – and conceding that "plain text" time would be longer than "key" time – and, thus, time would not be time anymore following *accidens*), it would be at a loss forever, once the moment at which time would surpass the length of the key, time itself would no longer exist, and neither would the world. With this we reach a time-complexity problem, very different from typical hierarchical levels related with the nature and evolution, language and grammar, or computation and



complexity, pronto instead as a collection of the above-named, and not only that, but contended as a paradox.

Having seen how cryptography and information entice rich ontological and mathematical crossroads, anticipating what we understand as new antinomies of the Kantian tradition in one anon informatic and cybernetics standpoint, one such debate is also convenient to the more narrow topic of programming (languages) media. Cryptography permits us to think on the analogy of ontology-theoretic secure information, and, consequently so, the place of communication and programming media languages in diagonalization with the world's ontology in sharp distinction of *analytic* and *dialectic*. Evidently, the Boltzmann brain paradox converges also to logical calculus and logic programming, from Aristotle to Prolog.

This divergence between *pure* determinism and *empirical* chance is certainly a mark of recognizable problems in complexity hierarchy. In relation to the accordance with the *P vs. NP* problem – remembering Stephen Cook's simple formula "The *P* versus *NP* problem is to determine whether every language accepted by some nondeterministic algorithm in polynomial time is also accepted by some (deterministic) algorithm in polynomial time. To define the problem precisely it is necessary to give a formal model of the computer." [81], provokes, yet, a read into consideration, that of encryption one-time pad of natural physical spacetime before any instance of computer science time complexity. Aside from this, to understand how the *P vs. NP* problem, excepting the Turing-Church computability thesis (1936), germinated from the period of emergence and mergence of the von Neumann architecture, is a different matter of interest.

If, on one side, computers were too limited to describe a wider range of problems with the concept of stored-program yet to be implemented, on the other side, attainable and recognizable grammars were to be constituted as too general to describe the syntax of natural languages, more hardly so encrypted (artificial) natural languages, and even more so programming languages to come<sup>32</sup>.

---

<sup>32</sup>After the disclosure of the  $\lambda$ -Calculus formalism by Church (1936), and even though Turing in the same year had referred to the possibility of a *U*-machine, which was clearly a foresight of the stored-program concept (as it is justifiably recalled by Chris Bernhardt in *Turing's Vision, The Birth of Computer Science* (2016) [36]) and Lorenz, still in the very same year, had designed two patent applications, in what respect machine instructions could be stored in the same storage used for data, the fundamental perception and theorizing of the stored-program concept, to

With a solid grasp of these historical notions, we understand how linear the line was between the original Charles Babbage's description of the analytical engine (1837), the Colossus (1943-45), and also the abstract design of the ACE (Automatic Computing Engine) (1945-46). Before the actual construction of the pilot ACE (a preliminary, modest, and partially disagreed on version by Turing himself, constructed in the 1950's), the *Lecture to the London Mathematical Society* [364] (1947) by Alan Turing, in the stage of De Morgan's learned society (1865), really epitomizes the ending of an industrial to enterprisal, steam-power to electronic-control, small- to large-scale, store-to- computer, mechanical-analogue to digital-discrete *transformation* [349]. Preferably, this historical passage with strong industrial-procedural, but also "context-sensitive" historical-cultural environment that is here characterized, not without the pinnacle of the von Neumann *First Draft* [380], represents best the ascendancy of computer architecture, in its fundamental underpinnings (machine code and general type-hierarchy, industrial assemblage to electronic design, ultimately hardware to software). If, bearing on the inevitable industrial-technological-cybernetics curve, to this we add the sort of IP-complexity (Interactive Polynomial Time) =  $PSPACE$  – where from we conjecture  $PSPACE = PTIME$ ?; or alike question: is *physical* spacetime *computationally/a priori* just as *naturally/a posteriori* partitioned or divided?) – war that was involved in the Colossus and Bombe face up with the Lorenz ciphering and the Enigma machine, in consummation of cryptography as the most strenuous method of diagonalization, with incremental quantitative design, memory

---

which, inevitably, programming languages are associated with, in the specific context of computer architecture, was, in the faultless and integral fundamentals, von Neumann's. However, this did not translate to any programming language, but instead to the ability of referring "any word in a programme by means of a label or tag attached to it arbitrarily by the programmer, instead of by its address in the store" [275], meaning a detachment from machine code and, simultaneously, an higher-order ranking in the abstract types-hierarchy (reflected in standards, syntax, instructions, control flow, arrays, strings, strings functions, or any other "abstract types"), retractable, in addition, to Alda Lovelace's translation (1842–1843) of Babagge's analytical engine machine (fully abstract) procedures of a calculating method for Bernoulli numbers, in the first place. At a time when programming had itself to detach in abstractedness from coding, in contrast to actuality, where programming languages seem to hold for less differential space and time, the ENIAC coding architectural system by von Neumann and the Moore School (1945) really seems complementary to the introduction of the programming language fabrication of Plankalkül (1942-45) by Konrad Zuse. The idea of a stored-program was incisive in the dawning of general-purpose modern computers, also transporting higher concepts to computability.

hierarchy, instruction-level and analysis, and the consequent field widening for complexity theory, it is our understanding that a more scrupulous account of computer architecture is unfolded, instead of the one and only showcase of the von Neumann or Princeton architecture [380].

The Analytical Engine (1837) devise by Charles Babbage was an *operational* full plan upgrade from the Difference engine. So, rather than saying it was supposed to test, we should say it was, indeed, a non-binary, 50 decimal places and floors with thousands of positions, general purpose arithmetic steam power mercury machine, engineering full plan. It was so prior to the electromagnetic induction with electromagnetic rotary devices consequent to the experimental inventions of Faraday (1761-1867), and far away from Maxwell's (1831-1879) electromagnetism equations and new unification in physics. Contrary to mere Jacquard's loom patterns sensitive rods feeding cords between cards, Babbage's machine was able to achieve conditional branch statement jumps and loops, with the ability to store numbers anywhere in memory, sorting to pull in and push back out data from the bus (having the ingress and outgress barrels – sort of rotating drums with pegs inserted – reading off the data). It incorporated functionally complete, store assembly level programming logarithmic tables, and printers for a permanent record. This was only in appearance a "false dawn" of computation, and not even the "largest, heaviest, complexiest" confront with systems organization could lead, eventually, to its depreciation. Its cog wheels apparatus permitted the system to be Turing-complete, an exceptional realization. When the printer, curve plotter and bell would compute a result, the base-10 fixed-point arithmetic, with a capacity of 1,000 numbers of 40 decimal digits (16.2 Kb in equivalence, i.e., 132710.4 bits) the proper arithmetic unit, which was a mill difference machine curved back upon itself in a circular layout (although later drawings depicted a regularized grid layout), would carry out (Turing-complete) system data-manipulation rules from three different types of punched cards: one for arithmetical operations, one for numerical constants, and one for load and store operations, with three separate corresponding readers. Babbage was acutely aware of the wide range opening for the creation of programs for the Analytical Engine, and indeed he himself developed several between 1837 and 1840, from iterative formulas to Bernoulli numbers.

What is even more astonishing is that the far ahead in time Analytical Engine, specially knowing that it remained beclouded in implementation tribulations, deserved a not very delayed appreciation by Luigi Menabrea (1809-1896) in the paper "*Notions sur la machine analytique de M. Charles Babbage*" [263]. This paper was translated by Ada Lovelace, who, along with Babbage, added extended notes, testifying for the only detailed account of the Analytical Engine for posterity. Later, upon the death of Charles Babbage, his son H. P. Babbage wrote and read *The Analytical Engine* [17] for an audience – increasingly, as we hear the name, sounding more and more a Kantian justified concept for a machine, and simultaneously every machine's *critical* limit – wherein we find the following note: "It is only a question of cards and time. Fabrics have been woven requiring several thousand cards. I possess one made by the aid of over twenty thousand cards, and there is no reason why an equal number of cards should not be used if necessary, in an Analytical Engine for the purposes of the mathematician" [17]. This is exemplary in understanding  $\cup$ -Mentalism, if we substitute cards for *empirical* images.

The Colossus (1943-45) – including the prototype Colossus Mark 1 (1943), and the improved Colossus Mark 2 (1944) —, in any case, at the physical level, using thermionic valves (vacuum tubes) (1,500 at the request of Tommy Flowers) to perform Boolean and counting operations, executed in a special purpose logic, namely to find the (one time pre-shared) "key" to the Lorenz cipher (removable rotor stream ciphers adapted to a teleprinter; Lorenz SZ40, SZ42a and SZ42b for *Schlüssel-Zusatz*, or cipher attachment). The series of computers Colossus can, therefore, be described as the first to combine programmable, electronic, digital computation, even if programs were controlled by switches and plugs, along with the use of paper cards (5,000 characters per second, roughly 30 miles per hour). The continuation and direct analogy with Babbage's *Analytical Engine* [17] is also very clear, now with Babbage's store as the hard disk and memory, the mill as the central processing unit, the steam engine as the power, the printer as the electrical boards, the barrel controllers as the microprograms, and the various (operation, number, and variable) cards, now making up the software in the paper tape with patterns of the alphabetical linguistic sort. It, thus, testifies to the imminent transition from the industrial revolution to the digital age, and the foreseen passage from special purpose to general purpose computation with the

Automatic Computing Engine (ACE) (1945) design by Alan Turing, following the *Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE)* [364] (1945), and the ensuing *Lecture to the London Mathematical Society* [364] (1947).

Even if the idea of stored-program in the ENIAC-EDVAC project was publicly advanced first, it was not as fully *mechanistically* depicted as with the ACE, apart from the fact that von Neumann had been acquainted with Turing's work in the first place. Gladly so, however, the idea of stored-program and the free enterprise vision combined in the USA, rapidly boosted the ENIAC-EDVAC project, both with military and civilian aims. Consequentially, it was, mainly, secrecy in Britain, and governmental public policy's many different aspects in the USA which gave a visible edge and publicity to the ENIAC (1943-1945) and the EDVAC (1944-1949). It is interesting to think what would have been an equivalent paper to von Neumann's *The First Draft* [380] eventually written in concurrence by Tommy Flowers, Max Newman and Alan Turing, not so much tied to mechanical aspects as was the *Proposal for Development in the Mathematics Division of an Automatic computing Engine(ACE)* [364] (1945), but instead more free in *philosophizing* as was von Neumann's.

Maybe it could never be as such, and only through von Neumann's *First Draft* [379] could we have seen one informatic anthropology understood, molded by as special a high-calibre polymath as von Neumann, a genius of the stature of Gauss, Euler, and Hilbert. The permanent analogy with the human body is never lost, to the point of the text being presented as an incorporation of the principles exposed in ergodic and measure theory – the  $10^{-6}$  seconds vacuum tube reliable reaction time, overcoming both the telegraph relay reaction time of  $10^{-2}$ , as the proper synaptic time of a human neuron, in the order of  $10^{-3}$  seconds – in one all-or-none, two equilibrium, binary (simpler than decimal) arithmetic, with a very strong note on computing *aesthetics* and general purposiveness *teleology* of machines.

The R(ecording) medium, C(entral) A(rithmetic), C(entral) C(ontrol), as for the rest, M(emory), I(nput), and (O)utput, with parallel simultaneous *telescoping operations* [380], and the "schematic picture for the functioning of the standard element of the device", meaning with this the "E-element" [380] as one hypothetical element, overly states an electromagnetic entity assumed to have synaptic delay

at fixed time. These absolute units of time – stipulated as  $2\mathcal{T}$  with  $\mathcal{T}$  being a microsecond – is described as a synchronized function, evidently comparable to the human brain's consciousness.

It continues in having a central clock as an electrical oscillator, whose pulse is  $(1/5)\mathcal{T}$  and a strict synchronism of stimuli which are integer multiples of  $\mathcal{T}$ , with vacuum tubes circuits, in this fashion, resembling a sort of computational  $\tau\omicron\pi\omicron\varsigma$  *topos* for autonomic nervous system computational thresholds, permitting electrons, here the artificial lymph, to connect the "planning networks of E-elements" [380]. von Neumann, then, chooses that for the standard number of about 30 digits, instead of being represented simultaneously, rather to be in succession, all 30 digits of one number at the same point, here with the binary point the analog of the decimal point. This is the base to the formation of a network which can add, subtract, multiply, divide and square root, depending on the C(entral) A(rithmetic), the connections for transfers between C(entral) A(rithmetic) and M(emory) being decisive.

Accordingly, the capacity of the M(emory), which are the actual vacuum tube trigger circuits, with  $k$ -fold memory in blocks, eventually converted into cyclical memories, responds to the organization of one *unit* as the value of one binary digit. This should also be the value for "*standard orders*" [380], of one minor cycle of 32 units. This is, conveniently, the base for the the estimation of requirements of memory type and function tables (with  $n^{\text{th}}$  order of minor cycles, with an approximate value of  $9x$  the standard punch card *place* for 80 digits). Solutions to mathematical problems have reached as high as 6,000 minor cycles estimates in the calculus of the author (even with fewer function tables and less complicated "set ups"), but what hides in this study of most relevance is the consideration of a bottleneck: "It shows a most striking way where the real difficulty, the main bottleneck, of an automatic very high speed computing device lies: At the memory. Compared to the relative simplicity of CA, and to the simplicity of CC and of its 'code', M is somewhat impressive: The requirements formulated in 12.2, which are considerable but by no means fantastic, necessitate a memory M with a capacity of about a quarter million units!" [380].

Not to forget that this equates to a synaptic body of memory, with as many E-elements as the desired capacity of M needed, with the construction of  $\boxed{\text{dl}}$  (cyclical

or delay memory with about 10 vacuum tubes associated in the architecture) organs with  $k$  values up to several thousand being unmanageable (specially uncertain when  $k \approx 10,000$ ), and besides this, compelling to consider the choice of either feeding each  $\boxed{\text{dl}}$  back into itself, or else to have longer  $\boxed{\text{dl}}$ 's cycles. A: Amplification and SG: Switching & Gating redirect the arrangements to the correct clock pulse for the output, wherein the  $\boxed{\text{dl}}$ 's delay time is a relative precision fraction  $t'$  of  $t$ .

Attending to the unit's memory capacity, the minor cycle with due numbers and orders (12.2 *First Draft* [380]), or more *organically*, not exactly the number of stimuli the organ can remember (localist *atomist* interpretation), but instead "the number of occasions for which it can remember whether or not a stimulus was present" [380] (network *synthesis* interpretation), the required choices of thousands of cycles to the power of two – conventionally the 262,272 minor cycles = 218 units, or the M(emory) capacity of 218 minor cycles  $\approx 250,000$  (a quarter million) units – and taking into account that each  $\boxed{\text{dl}}$  with a capacity  $k$  – the aggregate between any two SG has the capacity  $k' = kl$  – commands that " $\frac{250,000}{k'}$  SG's are needed altogether, and the switching problem of M is a  $\frac{250,000}{k'}$  way one" [380]. Because of this insurmountable task, one such factorization was replaced by a 250-way switching problem solution for the factor of 1,000 immediate and synchronous temporal succession  $t$  ( $1,000t$  i.e,  $c=250,000$ ,  $k=1,000$ ,  $h=250$ , hopefully in powers of two).

With this, and leaving ahead the relation between CC and M (related with code words and programming), we reach the most important point in von Neumann's *First Draft* (12.8) [380] in relation to *visualness* programming and  $\cup$ -Mentalism. We are referring to the *prima facie* more natural solution of the *iconoscope memory*, for technological-historical reasons impracticable and not usable at the time: "The solution to which we allude must be sought along the lines of the *iconoscope*. This device in its developed form remembers the state of  $400 \times 500 = 200,000$  separate points, indeed it remembers for each point more than one alternative. As is well known, it remembers whether each point has been illuminated or not, but it can distinguish more than two states: Besides light and no light it can also recognize – at each point – several intermediate degrees of illumination. These memories are placed on it by a light beam, and subsequently sensed by an electron beam, but

it is easy to see that small changes would make it possible to do the placing of the memories by an electron beam also." [380] We wish to stipulate this principle of a single electron beam and the switching action (steering and deflecting) from and to the focal point in the plate, a (computational) visual impression, as the baseline idea for the inception of U-Mentalism (as, simultaneously, a computer architecture and programming media, inverse to the established von Neumann architecture). It is not so much the concept of rapid switching to any part of the memory, liberated from temporal sequence, neither the holding of information per se in the static form of the dielectric plate that is fundamental, but instead the possible *synthesis* of the transient light waves, and the fact that information can be organized into a model theory programming media language, which should constitute its main appraisal<sup>33</sup>.

For the moment, though, it suffices to signal U-Mentalism in relation with the fundamental concept of the von Neumann or Princeton computer architecture [380], and capture, essentially, the following: first, that, independently from complexity theory polynomial, logarithmic and exponential algorithm problems and complexity classes [95, 250, 406, 178, 289], eventually seeking a similar separation of analytic (integral) and dialectic (differential) match of the greatest lower bound (*infimum*) and least upper bound (*supremum*) of *calculus* now in *computus*, the von Neumann architecture as exposed in the *First Draft* [380], in its special emerging body and machine metaphor and *de computa architectura*, not only foreshadows the passage from topology to a wider sense of the perceptiveness-of-the-self and the mind-body problem, as for the very same reason of a false continuity line from the machine to the self, becomes an attesting line of sight for the right comprehension of the Fifth Generation Computer Systems in Japan (FGCS) with the programming language Prolog as the perfect *errata* for the break out of the boundaries above exposed, under the context of (strong)

---

<sup>33</sup>Here and now, in a machine learning era [109] 1,000,000,000,000 frames/per second is achievable, there are, surely, dozens of trillions of digital images in digital storage (probably reaching soon a quadrillion  $10^{15}$ ) – not mentioning the possibility of capturing in footage natural light waves permanently and disseminating –, and particle accelerator machines, in particular, electrostatic accelerators, of which a cathode ray tube regular screen is a small scale example, has the ability to dimension itself, in *architectural* terms, to future computation, seeking the natural-to-artificial bounds of computability into wider perspectives.



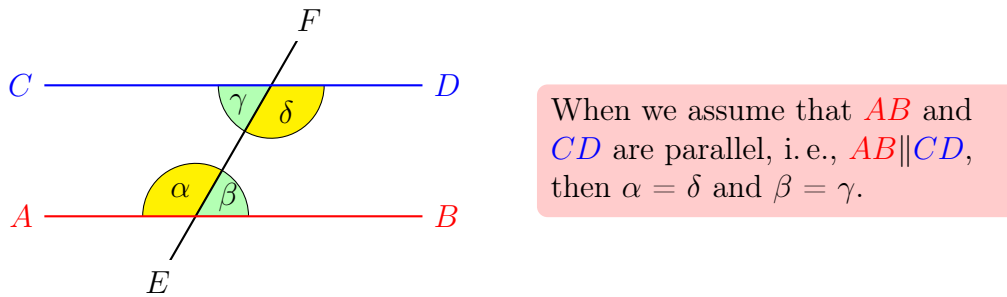


Figure 1.2: The Parallels V<sup>th</sup> Euclidean Postulate (and Paradox).

AI deception; second, that in the history of computation, the emerging nature of paradoxes and the oblique sense of cryptography (discernible in the transition from the industrial-mechanical Analytical Engine to the electronic-digital pre-stored and stored-program computers, respectively, the Colossus and the ACE) is truly affine with diagonalization [273, 138, 358, 312]. It should be worth, indeed, expanding computation as a diagonalization machine and comparing it with the birth of recursion theory and its ancestral historical roots since Euclid until Gödel [312, 216, 160, 182, 280, 346, 148, 92, 161, 162, 163, 219, 245]), the parallels postulate here envisaged as containing "gödelisation" topological arguments [264, 28, 156, 341, 132].

Assuredly, the Euclid-to-Gödel geometry-to-topology diagonalization arguments allow for both a critical thinking and a reasoning on the symbolical power of images in computation, demanding a *photon* computer vision & multiple view geometry paradigm [87, 309, 174] contemporaneous research on the limits of computability, in contrast with the *sonus et vox* natural language processing in logic programming and Prolog [39, 78]. Given credit to the renewing and sheer spirit of research and enterprise as visible in Prolog's strong AI history chapter with the Fifth Generation Computer Systems (FGCS) in Japan, what is now important is to reinforce this justified semiotic turn.

This is precisely the reason why untyped  $\lambda$ -Calculus (1936) (after the paradoxes exhibited by Kleene and Rosser on the 1932-22 earlier version) or the logically consistent typed  $\lambda$ -Calculus (1940), and thereupon, the Turing-machine model likewise (though never in need of corrections, and an imposing model over Church's for that reason alone), both share a structural and topological nearness.

We shall observe how this nearness translates into the important topological concept of neighboring points, or the study of boundaries (with a close eye on the *continuum* hypothesis, independent of set theory). This idea of neighboring points and boundaries, if taken notice, was the proper idea behind the Leibniz-Newton mathematical study of continuous change, in its birthplace the rise of the fundamental theorem of the *differentiating* and *integrating* power of *calculus*, inalienable from the contemporaneous assertion on the limits of functions. The idea of limits of a function, in notational form  $\lim_{x \rightarrow p} f(x) = L$  which attests for the idea that  $f(x)$  can be made as close as desired to  $L$  by making  $x$  close enough, but not equal, to  $p$ , makes possible the interpretation of  $f$  to be a real-valued function defined on a subset  $D$  of the real numbers, where  $c$  is the limit point of  $D$  and  $L$  also a real number, so that  $\lim_{x \rightarrow c} f(x) = L$ .

Conceivably, the idea of the limits  $(\varepsilon, \delta)$  in informal and intuitive manner<sup>34</sup>,

---

<sup>34</sup>The Epsilon-Delta  $(\varepsilon, \delta)$  definition of limit was vague and pretty much informal, ever since the inception of calculus with Newton (1642-1726/7) and Leibniz (1646-1716), and the treatment of infinitesimal quantities ("differential" quotient as a ratio of infinitesimal differences, "integral" as a sum of infinitesimals for Leibniz; whereas for Newton the derivative was a "fluxion" as a rate of change, and the integral was the "fluent"). One century later men such as d'Alembert (1717-1783), Lagrange (1736-1813) and Euler (1707-1783) also dealt intuitively with this notion, and were familiarized with a verbal definition only (Euler in the XVII<sup>th</sup> century himself was very aware of "uncritical" sums of series settled on the theory of equations, a foretoken for d'Alembert's solution for the bounds of errors in binomial series, and Lagrange's use of continued fractions as a new approximating method from iterations). Pierre de Fermat (1607-1665) had been also responsible for the use of "adequation" to find the maxima and minima of functions, in the context of trying to find the incommensurability measure – the slope of the tangent line – in such way that at a point  $x$  of a function, it could be reinstated in equational algebraic reasoning in any expression containing  $E$ , here taken as infinitesimal (almost zero value unknown) in any form of univariate (variable) algebraic equation. And it was precisely as a "variable quantity" that A.-L. Cauchy (1789-1857) used the expression under mathematical analysis. His lines of thought were as follows: "Let  $\delta, \varepsilon$  be two very small numbers; the first is chosen so that for all numerical values of  $h$  less than  $\delta$  and for any value of  $x$  included, the ratio  $(f(x+h) - f(x))/h$  will always be greater than  $f'(x) - \varepsilon$  and less than  $f'(x) + \varepsilon$ ." [62, 152].

Further on, Weierstrass (1815-1897) and Bolzano (1781-1848) finally refined the apparatus of calculus, but it is, arguably, fair to say that, in the midst of all the contributions, Cauchy was the forerunner in being able to show the sum approach to a fixed value, and so the integral interval of the function, ahead proving the fundamental theorem of calculus from the mean-value theorem by Lagrange. The achievements by Weierstrass and Bolzano have, in turn, set in motion number theory extensions. The most notorious is the construction of  $\mathbb{R}$  (real numbers) by Richard Dedekind (1831-1916), but it is also the case of hyperreals (or nonstandard reals)  ${}^*\mathbb{R}$  by Abraham Robinson (1918-1974) and even surreal numbers, not forgetting the recognizable and intermediate case of computable numbers. To all of these, a higher commuting analysis medially constructed between mathematical and philosophical structures can also be ascertained, visible

is that a function  $f$  approaches the limit  $L$  near a *symbolical*  $\lim_{x \rightarrow a} f(x) = L$  if we can make  $f(x)$  as close as we like to  $L$  by requiring that  $x$  be sufficiently close to, but unequal to,  $a$ , is remarkably similar to the Turing-machine model,  $\lambda$ -Calculus, and computable  $\mu$ -recursive functions. This is so in the sense that all the "variables" – in Cauchy's sense (*Cours d'Analyse*, 1821), wherein algebra was set aside for the emancipation of the *visual* geometric method, algebraic or combinatorial topology substituted by continuous topology –, taken as values, can, singling out the Turing-machine model or  $\lambda$ -Calculus in comparison, be taken as follows: values are successively attributed to the same variable, and approach indefinitely a real value, eventually differing from it by as little as possible, with that real value being called a computable number, as a limit of all the others.

Dramatically, the inner limitations of both arithmetic formal systems (Gödel's incompleteness theorems, 1931) and computation (the negative answer to the *Entscheidungsproblem* by Church and Turing, 1936) seem to appeal to nonstandard methods (almost a von Neumann pointless topological concept set from metric  $\mathbb{R}$  distance to  ${}^*\mathbb{R}$  as an extension), while, at the same time, with the *distance* from fluxions of infinitesimals ( $\epsilon$ ) to the infinite ( $\omega$ ) in Leibniz-Newton *calculus*, now being transformed to the distance from  $\mu$ -recursive functions to  $\mathbb{R}$  in Turing-Church *computus*, meaning a step forward (computability theory) and two steps back

---

in the new unfolding of imaginary and complex numbers in face of real numbers, and, exquisitely, in model theory.

In mere formal terms, focusing on the Epsilon-Delta definition: let  $c$  be a limit point of  $D$  and let  $L$  be a real number. We say that  $\lim_{x \rightarrow c} f(x) = L$  if for every  $\epsilon > 0$  there exists a  $\delta$  such that, for all  $x \in D$ , if  $0 < |x - c| < \delta$ , then  $|f(x) - L| < \epsilon$  [347].

It really shines through how much the paradigms of *calculus* and *computus* have been born on congruous birthright. Berkeley's attack on calculus in *The Analyst, or a Discourse Addressed to an Infidel Mathematician* [35] (1734), specially for someone who had inaugurated *immaterialist* idealism, and coming forward criticizing "evanescent increments" [35] and "ghosts of departed quantities" [35], really matches mathematical Platonism and functionalism embroilment characteristic also of *computus*, and more could be said on sums approaching fixed values in relation to  $\mu$ -recursive functions and the prefigurative Turing-equivalent machines, and equally so in relation to an integral-differential correspondence, especially in the turnover from 1931 (The incompleteness theorems by Kurt Gödel) and 1936 (the Turing-Church thesis). Overall, thus, regarding the transition from *calculus* to *computus* there is a major shifting paradigm that we wish to call attention to by precisely submitting their dynamic field of "equalities" and "inequalities" to a change of unbound limit. Hence, the use of *calculus* to *computus* expression under limits, here taken as of mathematic's own limits, serves as if it was a metalanguage notation with respect to *computus* increments on *calculus*.

(the Hilbert program delusion and the very probable *P vs NP* negative answer in mathematics and computer science). We herein convey a graphic of computational complexity classes (1994) (See figure 1.3, page 76) [288] that we will refer to later in our discussion. For now it is enough to give a proper diagrammatic representation of the different classes of problems concerning computational algorithms and the decision problem <sup>35</sup>.

---

<sup>35</sup>The Acronyms involved in the graphic are: "LOG Time" = logarithmic time, "LOG Space" = logarithmic space, "PTime" = polynomial time, "NPTIME" = nondeterministic polynomial time, "co-NPTIME" = complement of the nondeterministic polynomial time, "NPC" = non-deterministic polynomial time complete, "PSPACE" = polynomial space, "EXPTIME" = exponential time, "EXPSPACE" = exponential space, "2EXPTIME" = doubly exponential time, "ELEMENTARY" = elementary recursive functions, and "R" = recursive languages.

Roughly, following Kant's division of the formal *a priori* intuitions of (external) space, and (interior) time, and for an (abstract) order of  $\mathcal{O}$  – (constant  $\mathcal{O}(1)$ - logarithmic  $\mathcal{O}(\log_2 N)$ - linear  $\mathcal{O}(N)$ - quadratic  $\mathcal{O}(N^2)$ - cubic  $\mathcal{O}(N^3)$ - exponential  $\mathcal{O}(2^N)$  – growth rate or order of a function, attesting therefore for a description of a function providing an upper bound, we can moderate an hierarchy in the first place. Following this, we can avow, imagining the human brain to be, in a simple *analogy of experience* [Kant, A180-B223] (claiming only general principles of *substance, cause, and community*), not in transcendental, but instead through *schemata* – "analogies have their sole signification and validity not as principles of understanding's transcendental use, but merely as principles of its empirical use" [Kant, B224] – a *gedankenexperiment* the closest possible to a non-deterministic Turing machine, wherein the set of rules may indicate more than one action to be performed for any given substitution (a non-deterministic Turing Machine, thus, being a tripartite model with the first analogy of *permanence*, the second analogy of *succession*, and the third analogy of *simultaneity*) [Kant, A182-A28]. In a very general outline, the formal *a priori* intuition of time, corresponds in Turing [362, 363] to "computability" and "effective calculability" (Church, 1936) and, thinking along von Neumann's concepts in the *First Draft* [380], to the general input (I), whereas the formal *a priori* intuition of space, suitably corresponds, in Turing, to "memory" [362, 363], and in von Neumann [380], not so much to M(emory), but more so to the general output (O).

The preeminent class of time is, hence, observable to be insurmountably interior. This attests also for an impossible "refutation of idealism" [Kant, B274], something that is illustrated through the lines of computer science, bypassing metaphysics. In other words, complexity theory and the study of the limiting behavior of a function from any argument towards infinity, under the context of recursiveness, is rather rebutting than affirmative of Kant's "theory which declares the existence of objects in space outside us either to be merely doubtful and indemonstrable or to be false and impossible" [Kant, B 274]. This explains why computationalism is congruent with idealism and skepticism, in conformity with the (inverse) expressions "doubtful and indemonstrable".

Knowing that for open questions, there is not a way to find an answer "quickly" (in polynomial time), in spite of the fact that if one is provided with information showing the answer, it is possible to verify the very same answer "quickly enough" (in polynomial time), i.e, that for any problem, verifiability translates to consistency – NP as "quickly" *checkable* as P is "quickly" *solvable*, all together *tested* in the equivalence relation – proving that any problem under non-deterministic polynomial time is also under polynomial time, thus, being, "tractable", "feasible", "efficient", or "fast" [74], equates asymptotically faster algorithms being open to discovery, presumably

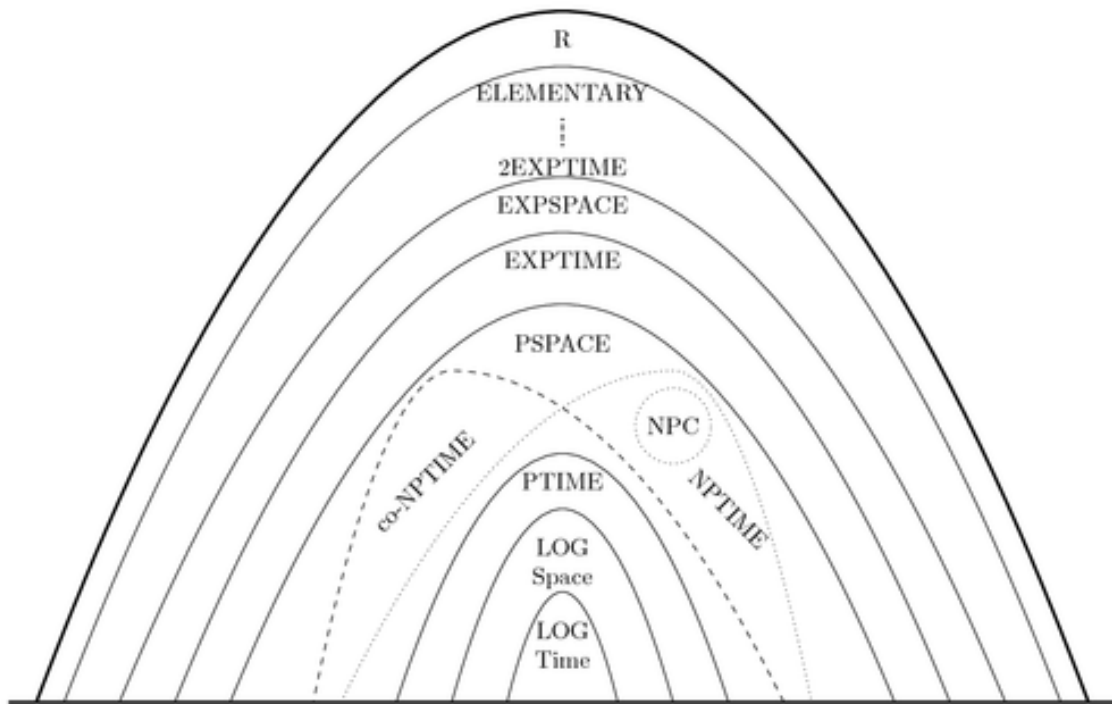


Figure 1.3: Computational Complexity Classes

Really, "distance" itself has been *projectively* augmented – in Cayley's sense that "every geometry is a projective geometry" [128] (not the same as saying that each and every geometry is projective) – in relation also with the limits in Kant's antinomies of pure reason. We are, thus, considering the (mathematical) first antinomy "of space and time", the second of "atomism"; the (dynamical) third of "spontaneity and causal determinism", and the fourth of "necessary being or not", as constituted with relational bounds between them, herein "distance" as a proper philosophical (mathematical and dynamical) symbolic concept.

They are now exceptionally divorced in the *computus* age, meaning more dis-

---

subscribing  $NP = P$ . Once every problem in NP has an exponential time algorithm – in short, exposing  $NP \subseteq PSPACE \subseteq EXPTIME$  – one such  $NP = P$  equivalence would have a counter-elliptic effect on the diagram. The reason is that all different classes would be subsumed into one only equivalence set –  $PSPACE = NPSPACE = IP$  – wherefrom  $IP$  or interactive polynomial time would consist of two machines, a prover  $P$ , and a verifier  $V$ , as if machines were oracles for each other [81]. In spite of the preeminence of time, in complexity theory, nevertheless, once memory needs always to supersede the length of the steps, PSPACE, EXPSPACE, are always shown as outer layers to, respectively, PTIME and EXPTIME.

tant between each other, *as if* we should read  $\lim_{calculus \rightarrow computus} f(calculus) =$   
*Limit* where *calculus* is typical of Kantian antinomies, and *computus* is typical  
of radically augmented antinomies, i.e., from the (post)-modern age and Leibniz-  
Newton *calculus* (1666-1704), on which the transcendental subject of Kant inquired  
about the knowledge limits, to (post)-contemporaneous Turing-Church *computus*  
(1936), on which the transcendental subjects of billions are permanent oracles for  
extensions of information.

Here we settle minimal indispensable examples for each and every one of them,  
with the purpose of grounding a perfected judgment of the established by Kant  
antinomies (*thesis cum antithesis*) [A421], henceforth in the *computus* age and  
concerted with the guiding principle of pure reason. Originally, surmounting to  
explain how the conflict of transcendental ideas is contemporaneously aggravated  
– symbolically asseverated into *spatial-to-topological* provisions, insofar as Kant  
himself referred to a "dialectical arena" [A423] – we proceed to expose a brief  
*rationale* on the inflection of anew antinomies. Again, the antinomies are four in  
total. They are the "Mathematical Antinomies" [A426-B454 – A442-B470] – "The  
First Antinomy (of Space and Time)" [A426-B454 – A434-B462] and "The Second  
Antinomy (of Atomism)" [A434-B462 – A444-B472] – and also "The Dynamical  
Antinomies" [A444-B472 – A461-B489] – "The Third Antinomy (of Spontaneity  
and Causal Determinism)" [A444-B472 – A452-B480] and also "The Fourth Anti-  
nomy (of Necessary Being or Not)" [A452-B480 – A462-B490]:

### 1.1.2 Informatic Anew Antinomies

**Theorem 1 (\*3 · 24)** *Dem.*

$$\left[ *2 \cdot 11 \frac{\sim p}{p} \right] \vdash . \sim p \vee \sim ((\sim p)). \supset$$

$$\left[ *3 \cdot 14 \frac{\sim p}{q} \right] \vdash . \sim (p. \sim p)$$

*The above is the law of contradiction[399].*

[Thesis: the world has a beginning in time, and is also limited as regards space.]  
& (conjunctive, non-contradictory: apparently the converse of *Principia Mathematica*'s  
theorem of non-contradiction: \*3 · 24.  $\vdash . \sim (p. \sim p)$  ...but now in the

negation form  $\sim . \sim (p. \sim p)$  and, thus, openly equivalent to  $(p. \sim p)$  [(Anti-thesis: the world has no beginning, and no limits in space; it is infinite as regards both time and space.):

In the *computus* age, and considering the *fulcrum* year of 1936, inevitably, computation is more dissociated from physics substantiation of reality, in the sense that Turing-machine's  $\mathbb{R}^\neq$  *ideality*, within the limitations of computable functions and the negative answer to the decision problem, clashes even more with  $\mathbb{R}^3$  when it engulfs differential geometry of surfaces as in Riemannian geometry, the basis for the theory of general relativity by Einstein, and the proper imbrication of space-time. Recursion theory is neither capable, through time-functions or elementary operations performed by the algorithm at a fixed time, either in linear time algorithm  $T(n) = \mathcal{O}$  or polynomial time algorithm where  $T(n) = \mathcal{O}(n^\alpha)$  for some constant  $\alpha$ , such that  $\alpha \geq 1$ , to process all the different *images* of non-linear time if considered as time-functions (even if parceled into sequential frames in Minkowski spacetime<sup>36</sup> they would fail to be computed in a U-machine, let

---

<sup>36</sup>Minkowski spacetime corresponds to putting together  $\mathbb{R}^3$  (three-dimensional Euclidean space) and the extra spaciality-conformed dimension of time, recorded as a mere event frame, into a more global *quasi*-four-dimensional manifold (really a  $\mathbb{R}^2$  plane for space, and a single metric vector  $\mathbb{R}^3$  cone as a time-function for the speed of light, or alternatively, recurring to slicing, a cone into various  $\mathbb{R}^3$  idealities in successive  $\mathbb{R}^2$  planes, wherein the so called spacetime interval between any two events is made, in one such invention, independent of the inertial frame of reference belonging to each singular event). Fascinatingly, the so called Minkowski proper time (a single point event in the light-cone) or time-distance (the proper distance between any two or more than one-point events in the light cone), setting a speed of light-metric to the distance, supposedly linear, is, nevertheless, made both elliptical and hyperbolic in *representation*, due to the perceiving point of view of the transcendental subject (never coincident with light speed), but also for the speed of light, once it lacks orthonormality (the notion of perpendicularity of two-linear forms, affected to "Newtonian" space and time, or, if imagined from the speed of light standpoint, not afflicted by Olber's paradox, in a sort of infinite spacetime). But, again, these are *representational* only. This is exactly the point where Kantianism directly bounds with model theory, in saying that in between the model and the model, in a similar manner as, *physiologically*, exists a blind spot - *punctum caecum* - the *place* in the visual *field* that *corresponds* to the lack of photo-receptor cells on the optic nerves, there exists, *philosophically*, a *punctum caecum* coincident with the invisible. This is made worse because of the natural unrestricted extension of the power of knowledge beyond critical *limits*, as the brain extends vision by interpolation to the surroundings. In the same way, never compromising satisfiability and validity, representation of three-dimensions (in  $\mathbb{R}^2$  planes or axis of coordinates) is standard, as it is the necessary shift from Minkowski spacetime to the curvature of real (Riemannian) spacetime in general relativity, although the very same philosophical blind spot is still pertinent. And the same would be true,

alone  $n$ -dimensional spacetime). But computability theory, on the other side, if it can partially and structurally model all different mathematical structures - in the proper collective pseudonym Nicolas Bourbaki's meaning of "mother structures": algebraic, topological, and order - accommodating partial lists of possible structures to partial recursive functions, important *practical* or *experimental* results can be achieved recurring to extra-computability, though embedded mathematical concepts, such as measures, algebraic structures (groups, fields, etc.), topologies, metric  $n$ -dimensional geometries, besides orders, events, equivalence and categorical relations, integral and differential upper and lower bounds, and the like.

[Thesis: every composite substance in the world is made up of simple parts, and nothing anywhere exists save the simple or what is composed of the simple.] & (conjunctive, non-contradictory: apparently the converse of *Principia Mathematica*'s theorem of non-contradiction: \*3 · 24.  $\vdash . \sim (p. \sim p)$  ...but now in the negation form  $\sim . \sim (p. \sim p)$  and, thus, openly equivalent to  $(p. \sim p)$  ) [Anti-thesis: no composite thing in the world is made up of simple parts, and there nowhere exists in the world anything simple.]:

In the *computus* age, and considering the *fulcrum* year of 1936, it can be warranted that Leibniz's *De Arte Combinatoria* (1666), later designated *ars characteristic* or *characteristica universalis* (obeying the principle that all in existence can be reduced to primitive elements, harmony being a composition of the simple) is, in its modern translation, symbolical Boolean values sequentially computed by a Turing-machine, nowhere near the Lullian-Leibnizian idea of a sort of heuristic rational mechanicism proving to be *in relation* to monads. Eventually, this was prospected by postulating vibrating *conatus*, as transpiercing pre-localities in all types of *phenomena*, reclaiming, thus, inasmuch as causality and spontaneity together, a reduction to one-point "elementary particle" monads (although

---

taking Gödel's incompleteness theorems, even if our *eye* was all we had for a *body*, and this body was the *body* of mathematics.

Minkowski spacetime is, furthermore, not only naturally adequate to the electromagnetic spectrum of light as a phenomenon, after Maxwell and Lorentz, but was recognizably conformed to the physics of special relativity (1907). Computability theory, in its complexity branch, should be attentively close to such constructs, as it is, really, in between electromagnetism and new physics-related models.



less simple and more dual-paradoxical than one-circle  $\mathbb{R}^2$  to one-sphere  $\mathbb{R}^3$  *images* of unity in Platonism). We assume, comparing to Newton, this to be an eloquent demonstration of the similar concept of "action at a distance", though purely metaphysical and, therefore, more so *conceptual* relativist distance (Leibniz), than, if ever so, *physical* absolutist (Newton), in either case space (and time) being considered as *sensorium dei*. Inasmuch as Leibniz's sufficient reason of the totality of contingent things (in relativist space and time) endorses to the necessary being in *metaphysical* terms, Newton's *physical* gravity is to be considered also spontaneously theistic<sup>37</sup> (in absolute space and time). Hence, in retrospect, we understand how both Leibniz's and Newton's systems were, even if antagonistic (relativist metaphysical *vs.* physical absolutist) facing space and time, metrical anew. What made the difference, both for the author of the *La Monadologie* (1714) as for the author of *Philosophia Naturalis Principia Mathematica* (1687), in the spectrum of classical mechanics and classical metaphysics, was physical motion and metaphysical movement, necessarily a *preterius*-(projective and topological) spark, even more so in the heliocentric universe. Throughout the Euclidean-space *limits* of necessary *plenum* and impossible *vacuum*, the frame of reference of distances between all members of the heliocentric universe considered as a set were made more explicit: for both Leibniz and Newton, the artificers of *calculus*, agreed fundamentally that the perceiving of time, but also of space, was *metric* and relativistic in essence. Difference was in the ensuing postulation in trans-finite and *metaphysical*-limits (Leibniz) *vs.* postulation in trans-finite and *physical*-limits (Newton). This obliged Leibniz to *metaphysically differentiate* space and time towards relativism (measure being non-physical, non-metric, but strongly projective), while Newton was driven to *physically integrate* absolute space and time (measure being physical, metric, but weakly projective). Physical motion, with Newton, was prone to the upper bound integral measure of the *plenum* (connected in extension as an open set without points in the boundaries) with the observer

---

<sup>37</sup>In this regard, we read from Newton's *Opticks*: "(...) can be the effect of nothing else than the Wisdom and Skill of a powerful ever-living Agent, who being in all Places, is more able by his Will to move the Bodies within his boundless uniform *Sensorium*, and thereby to form and reform the Parts of the Universe, than we are by our Will to move the Parts of our own Bodies. And yet we are not to consider the World as a Body of God, or the several Parts thereof, as the Parts of God. He is an uniform Being, void of Organs, Members or Parts (...)"[282]

in relativist metrical space and time finding these to be movable dimension measures of absolute space and time), while metaphysical movement, with Leibniz, was set forth in the lower bound differential measure of the *plenum* (compacted in extension, and even collapsed as 0-dimensional points<sup>38</sup>).

We affirm, considering the frame field in general relativity – with the timelike unit vector field  $\vec{e}_0$  and the three spacelike unit vector fields  $\vec{e}_1, \vec{e}_2, \vec{e}_3$  –, and facing a system endowed with Einstein field (partial different) equations, that what is forthwith definitely observable is, quite unexpectedly, the *physical* vindication of relativist views (Newton meeting Leibniz), and the *metaphysical* vindication of absolutist views (Leibniz meeting Newton): non-quantum (in the "quantification" sense) monad-like behavior in the subparticles world, after general relativity & quantum mechanics; and absolutists space and time behavior-like in relation with light, due to the permanent self-measured and self-metric spacetime in relation with extension. Once again, it is movement and motion that are in question, as if the real velocity and the real constant was that of immovable spacetime (monad or universe), in Kantian terms (converging Leibniz and Newton), a general outwards (exterior space) and inwards (interior time) transcendental non-metric *a priori* affect, coincident in receptivity with the formal elements of intuition, space and time.

Now, this is the *cosmological* equivalent of the *continuum* hypothesis, i.e., the idea that "there is no set whose cardinality is strictly between that of the integers and the real numbers" [355] (Hilbert's 1<sup>st</sup> problem), which goes on the same line as extension and set theory, as it is deeply ingrained in recursion theory. No wonder

---

<sup>38</sup>The 0-dimensional point, or nildimensional point (lacking any metrics, in form of length, area, volume, or any other dimensional predicates, as higher-equivalents 1-D. line segment, 2-D. plane and 3-D. Euclidean geometry), bears, in topological and physical terms, a collapsing action (although, contrary to what might be expected, invariant and, therefore, constant), very similar with the monad, as understood by Leibniz, and also very similar with gravity, as understood by Newton (and, indeed, in Einstein's general relativity, understood as it should be, a geometric theory of gravitation, more specially so if compared with, in pursuit, but also moved against, quantum theory, for, attained to the Copenhagen interpretation – Heisenberg, Bohr, and Born in 1928 –, rose the idea of "collapsing" properties of subatomic particles, all averse to observance and measurement, such as position and momentum). It is clear by now that Leibniz's physics *relativism* has modelled positively the necessary plasticity towards the physics of general relativity, while Newton's *absolutists* space and time doubtlessly structured the necessary grounds of physics law.

then, that Gödel, who worked himself with a (truly cryptographical) numbering diagonalization – a function assigning to each symbol and well-formed formula of some formal language a unique natural number, a technique called Gödel numbering (clearly a non-classical  $V^{th}$  or parallel postulate suppression in philosophy of mathematics, permitting numbers "lines" to meet different trans-finite or close to infinity cardinalities) – also dedicated attention to the so called "Gödel metric" or "Gödel solution"[164]. It was explored in the context of Einsteinian partial field equations and the energy–momentum tensor, where its second term is associated with a nonzero cosmological constant, designated the  $\Lambda$ -vacuum solution. This is important to refer to as it exposes the passage from Einstein's envisagement of a constant ( $\Lambda$ ) to account for a stationary universe (Einstein, 1917), neither expanding nor contracting, thus, untopologized and holding back gravity (*quasi*-congruent with Newton's absolutists views on space and time) on one side, and after Hubble observing that the universe is expanding, the cosmological constant being substituted by a vacuum energy density of empty space (the proper division in the energymomentum tensor between matter and vacuum) (*quasi*-congruent with Leibniz's relativists views in *La Monadologie*) on the other side. This solution is also a reminder of  $\aleph_0$  (smallest infinite cardinal number) after Cantor's diagonalization proof, and the idea that infinite sets can have different (symbolic) cardinalities, similar also to Turing-Church *computus* in having cardinality  $\aleph_0$  of recursive numbers in one-to-one correspondence with the natural numbers, in contrast with infinity ( $\infty$ ) in *calculus*, which attests an extreme *limit* of the real numbers line, as if the function arrow was a leap forward, increasing without bound. Being space and time considered instead the proper dimension measures in accordance with the "position" of the monads, the universal physical "locality" of gravity was, thus, equivalent, to the universal metaphysical "position" of the monad. Contrary to having *integral* inertial frames of references as metric spaces, monads (entelechiol created or others) have a constant *differential energeia* (non-metric: without space, time, motion, force, mass, velocity or acceleration). Now, computability theory and the *computus* paradigm, following the Turing-Church thesis, and in a time where mechanics holds two sub-fields (classical mechanics and quantum mechanics), inducting strong and weak nuclear forces, besides gravity and electromagnetism (electromagnetism, really, being the only

field of physicality in close relation with state of the art computers, even considering quantum-computing, necessarily enclosed in the Turing-Church thesis), is very far away from any sort of metaphysical or physical (either in substantiation or transubstantiation) metrical power. All that is within its scope are computations according to theoretical measurements, and above all those of number theory symbolic manipulation in a limited *ideal* bi-dimensional plane.

[Thesis: causality in accordance with laws of nature is not the only causality from which the appearances of the world can one and all be derived. To explain these appearances it is necessary to assume that there is also another causality, that of spontaneity.] & (conjunctive, non-contradictory: apparently the converse of *Principia Mathematica*'s theorem of non-contradiction: \*3 · 24.  $\vdash . \sim (p. \sim p)$  ... but now in the negation form  $\sim . \sim (p. \sim p)$  and, thus, openly equivalent to  $(p. \sim p)$  ) [Anti-thesis: there is no spontaneity; everything in the world takes place solely in accordance with laws of nature.]:

In the *computus* age, and considering the *fulcrum* year of 1936, it is noticeable that if the *rationale* of the four Aristotelian causes (*Aristotle*, Physics II, 3, and Metaphysics V, 2) - the mutually exclusive "material", "formal", "efficient" (or "moving"<sup>39</sup>), and "final" causes - were absorbed into late Medieval Thomism, since David Hume's (1711-1776) *A Treatise of Human Nature* (1738-40) and consequently Kant's "awakening from the dogmatic slumber" (Kant, *Prolegomena to Any Future Metaphysics* 1783), never again was it possible to decrease the power of inductive reasoning, especially considering the historical-philosophical artery where from skeptic natural empiricism gradually shifts to scientific artificial empiricism (from Hume to Darwin, and from Darwin to Turing). Surely, Aristotle's aetiology still brings a germane analytic to the field of AI. In particular, the philosophical assessment of the Aristotelian "efficient cause" (liable to movement and motion), should be an object of study, as it is the primogenitor root of *calculus*

---

<sup>39</sup>We should not lose sight of the important connection marked in ancient Greek metaphysics, persevering to contemporary science, between the deist artificer demiurge and efficient cause, as well as between an external cause and movement, out of which motion is just the physical correspondent.

(differentiation, integration and function analysis in dynamics) and, therefore, of *computus* as well (the Turing-Church equivalent  $\lambda$ -definability of  $\mu$ -recursive functions, capable of recurring to primitive functions composition). In what relates to Hume, the fundamental causation reduction to custom and habit, a belief or trust without any other natural contract except time and experience, has really launched a study of the continuous, "of ideas, their origin, composition, connexion, abstraction, &c." [191] (*Treatise of Human Nature*, Book 1, Part I), so markedly continuous that the difference between impressions and ideas is only one of degree, amongst the "infinite divisibility of our ideas of space and time" [191] (*Treatise of Human Nature*, Book I, Part II. I), framed in a renewed interpretation "of probability; and the idea of cause and effect" [191] (*Treatise of Human Nature*, Part III, II). If noticed carefully, Hume corresponds to an eccentric case in modern philosophy, a combination of being the latest in tradition, and of proper philosophical singularity. To discern the important causation induction principle, this very subtle difference of Hume being the only one to oppose all the other philosophers about various epistemic assumptions, while sharing most of the modern tradition topics (*oppositus sed non adversarius*) needs to be highlighted. While doing so, we should bear in mind that the keystone differential is based on the collapse of causation, derived from radical "sceptical doubts about the operations of the understanding" [190] (*An Enquiry Concerning Human Understanding*, Section IV, 1748), at which point it is rendered clear the strengthening of systemic (Cartesian) doubt, largely beyond the bounds thereof set by Descartes himself. David Hume (1711-1776) is, thus, the sole modern philosopher, though belonging to the group of empiricists – Francis Bacon (1561-1626), John Locke (1632-1704), and George Berkeley (1685-1753), signally inductive – to have gone beyond the acceptance of a scientific method (Bacon), worldly sense perception against the Cartesian dream argument (Locke), and explanatory ideal empiricist immaterialism (Berkeley). Indeed, to Hume, causation, being an old habit, was just about a timely natural principle of connection, next in importance to contiguity and resemblance, "the three only bounds that unite our thought together" [190] (*An Enquiry Concerning Human Understanding*, Section V, 1748). Hume is also the sole modern philosopher, even if sharing with the rationalists – Descartes (1596-1650), Spinoza (1632-1677), and Leibniz (1646-1716), signally deductive – a *practical* vindication

of reason (although void of its dogma), to have established its nature on the inductive free association of ideas and probability, all in all a matter of perceiving sensibility, i.e., the faculty of reason being, uniformly with all possible experience, abstractions of *impressions* that arise from the *senses* [191, 190] (Hume, *Treatise of Human Nature*, Section V). Consistently, Hume is the sole modern philosopher, contrary to all the others aforementioned, to have granted the principle of uniformity and the *continuum*: the outset Cartesian dualism, in separating thought and extension, even in different unfoldings (as with Spinoza's *substantia*, *modi* and *attributa*), markedly differentiating ontological divisions in one neutral monism, is, however, not akin to Hume's principle of uniformity. This is taken as one ontological *continuum*, moreover inductively-ordered and resiliently skeptic towards any effect of causality, even if borrowed everlastingly to time and experience. In this fashion, by strengthening systemic doubt to the core of human reason, not falling to disassociate reason from relativism, intermittent time was made a perpetual judge for any extension of human knowledge. This is directly contradistinct to the ἐποχή (epoché) method, in the sense of a state where all judgments are suspended, of all the other modern philosophers: it is only *hypothetically* that Descartes discerns on the dream argument and the misleading God (*Meditations on First Philosophy*, 1641); it is only *hypothetically* that Berkeley imagines reality as a whole product of sensory intersubjective relativism (*A Treatise Concerning the Principles of Human Knowledge*, 1710; *Three Dialogues between Hylas and Philonous*, 1713), both, henceforth, shielding theological views. And the same could be said in relation to the proper nature of sensory experience and evidence in relation to causation: even if *assertorically* establishing a scientific method, Bacon never went to purge the idol of causation in the system of sciences and empiricist natural philosophy; even if *assertorically* emptying human nature to a sort of *tabula rasa*, Locke never went further to include reason and causation as *rasae*; even if *assertorically* construing freedom as postulated, never were modern philosophers, except Hume, seen to evolve to the skeptical position where both causation and human reason were, respectively and *problematically*, succession and free association, as found equally two and a half centuries later with the Turing-machine model and, most specially, with Turing's *Imitation Game* in *Computing*

*Machinery and Intelligene* (1950) [368] (it is also this exceptional philosophical incidence, more than just mathematical logic, of Turing's machine model (1936) and heteromorphic thought, balanced between *declarative* problems in philosophy, and *procedural* machinery, that gives about computability an extra edge to Turing in relation with Church, even though recursion theory was all throughout a concurrent and emergent field with exceptional men and women). Most importantly, and enveloping both, Hume's principle of uniformity was set as one ontological principle *in continuum*, and, extraordinarily, also of inexcusable *de omnibus dubitandum est*. In spite of the uniformity principle *in continuum* – though never falling to breaches like dualism, *substantia, modus et attributus* unfoldings, or even to the empiricist primary/secondary quality distinction –, Hume was able to prospect – though never subsiding axiomatic natural deduction systems, comparing with later Kantian critical philosophy –, a mindfulness' *continuum*, also unequivocal in Turing (and hardly so engendered if we remember that it was performed only from computable numbers and a machine model). It appears as such, in the context of modern philosophy, as a sort of combination of the inductive *ideas* of a compact connected metric space, and unboundedness extension, i.e., where succession and free association were coped with infinitude of both the immeasurably large and small, and also of nondenumerable reality, always plunged within subjective time. It is only inside this fully subjective skeptical empiricism of Hume, taken as a philosophical lattice, breaking down at once both dualism and causation, that is made comprehensible the following tracts of systematization handed down by the Scottish philosopher, decisive to the characterization of subsequent shifts in natural philosophy: the separation of the necessary *vs.* contingent (concerning reality), the *a priori vs. a posteriori* (concerning knowledge), and the analytic *vs.* synthetic (concerning language), with truths relating ideas (abstract) all *made parallel* on one side (necessary, *a priori*, analytic), whereas truths on actualities (concrete) all *made parallel* on the other side (contingent, *a posteriori*, synthetic) [133]. It is only now that the passage to Darwin is *enlightened*, as, something that has been largely dismissed in the subject matter *précis*, Kant was the only natural philosopher before Darwin to have critically envisaged the natural mechanism of evolution by

means of natural selection, only to integrally deny it afterwards <sup>40</sup>. Hence, Kant was likewise the only natural philosopher to give continuity from Hume's principle of uniformity under the natural sciences, fundamentally in ἐποχῆ (epoché) method to thereupon critically discard it, while, also in ἐποχῆ (epoché) method

---

<sup>40</sup>As there are many mathematical logic techniques (with ramifications into set theory, model theory, recursion theory, and proof theory), we can address logic and semantic technicalities in metaphysics. The philosophies of Descartes and Berkeley are good examples of the use of such techniques of formal reasoning, with deeply entrenched consequences in the outline of metaphysical theories. Descartes made use of testability to the end of falsifiability forecasting counterexamples and, therefore, confirmability, in the explicit case "the demonstration of the existence of God and the immortality of the soul" (*Meditations on First Philosophy*, 1641, Meditation I: Concerning Those Things That Can Be Called into Doubt), thus, tactically advancing hypotheses, just to be eventually refuted. Berkeley used defeasibility also, yet to the end of contingency (*A Treatise Concerning the Principles of Human Knowledge*, Part I, 1710; *Three Dialogues Between Hylas and Philonous*, 1713, Part I) tactically advancing hypothesis – exposing numbers and sensed qualities as mental and non-*material*, disputing knowledge of external objects –, just to call upon them as a contingency of logical form in respect to *materialism*  $\vee$  *immaterialism*, now with the end of inspecting topics wherein the "chief causes of error and difficulty in the sciences, with the grounds of scepticism, atheism, and irreligion are inquired into" [34], all in all very similar to Descartes' *desiratum*. Kant too, prior to teleology winning over *gesetzmässigkeit* (conformity to law) or *weckverbindung* (general purposiveness), made use of a general hypothesis under the form of an inquiry to determine the possibility, principles and extent of human *a priori* knowledge, necessarily overlapping (pure) analytic from (empirical) dialectic, conceiving an extension of pure reason in a practical point of view, without speculative knowledge being enlarged alike, establishing in completion the primacy of *pure* practical reason in its union with the speculative, carrying inevitably the immortality of the soul as a postulate of pure practical reason (*summum bonum*). Kant's critical philosophy is, therefore, an argument in the lines of Gödel's philosophy, and foremost its 2<sup>nd</sup> Incompleteness Theorem in relation to Platonism. They both go one step back (critical limits, and incompleteness) and two steps further (practical dialectic, and mathematical Platonism), with the great risk of the libel *ex contradictione sequitur quodlibet*, i.e., "from a contradiction, anything follows". This was enough for later critics, such as Nietzsche, to have remarked that Kant retracted to this burrow "like a fox which strays back into its cage" [283]. We should note that Locke's primary and secondary qualities (*An Essay Concerning Human Understanding*, Chapter VIII, 1690) are also a fairly good *argumentum ad absurdum* to test Berkeley's philosophical contingently said empiricist *immaterialism*, or else said *material* idealism. Provided these, and attaining only to primary qualities – solidity, extension, motion, number, and figure [243] – it is seen that Berkeley's metaphysics, contrary to Leibniz's *La Monadologie*, did not evolve to *possibly*, perceptively and in perspective, conjecture to make collapse all except "solidity" and, though less clear and carrying contradictions, "number". Therefore, having kept "extension", "motion" and, inevitably as such, "figure", is better understood why Bergson spoke as of Berkeley's "screen" [33, 31, 30, 32, 29] (necessarily claiming "extension" and "motion", besides "figure"), and why inter-subjective idealism was Berkeley's keystone, while Leibniz's metaphysics held more physicalist concerns than Berkeley's (although incomparable to Newton's), reason being that if perspective relativism in Berkeley's philosophy was taken to its (differential) *limits*, something very close to the (integral) concept of a monad would break in (neither with "solidity", "extension", "motion", "number", nor "figure").



and onwards, to critically envisage, in anticipation, evolution by means of natural selection. While it is intelligible that one such defense was conformed to the *practical* moral philosophy hoped for in the *Critique of Practical Reason* (*Kritik der praktischen Vernunft*, 1788), such tenets are clearly patent in the *Critique of Judgment* (*Kritik der Urteilskraft*, 1790), settled on aesthetics and teleology. Finally, with Darwin, natural selection, following *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life* (1859), opening with an *artificiality* touchstone – "Variation under domestication" [94] (Chapter I), with a strong ending statement on "Man's power of selection" [94]<sup>41</sup>, whether addressing on cattle, sporting plants or pigeons – with natural selection therein being described simply as the "preservation of favourable variations and the rejection of injurious variations" [93], with those neither useful nor injurious left as a fluctuating element, as in those labelled polymorphic, settled, thus, in heritage, a line of study that would, eventually, correspond to Turing's and von Neumann's later work. Turing's production on morphogenesis and his only paper devoted to biology – *The Chemical Basis of Morphogenesis* [365] (1951) – was, nevertheless, a building sign of areas as important as *The System of Logic* [367] (Princeton PhD 1938), *Mathematical Logic* [363], *Mechanical Intelligence* [364], and *Pure Mathematics* [366]. Indeed, there is a transposing flux of former content – left-right equivalence in complex-valued functions, approximations to Lie Groups (1938 A & B), a detailed design of an electronic universal machine with hierarchic series primitive programming (1945), intelligent machinery focus on subroutines, theorem-proving, and machine learning (1948), large routines checking (1949), and a computer arbitrarily random guessing game (1950), just to name a few<sup>42</sup> – to

---

<sup>41</sup>"Man's power of selection" [94] as understood by Darwin, is not to be taken as an absolute radical principle. We know Darwin also to have declared in the same work: "But Natural Selection, as we shall hereafter see, is a power incessantly ready for action, and is as immeasurably superior to man's feeble efforts, as the works of Nature are to those of Art." [94].

<sup>42</sup>In more precise terms, we are referring to the papers written by Turing *Equivalence of Left and Right Almost Periodicity* (1935), *Finite Approximations to Lie Groups* (1938 A), *Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE)* (1945), *Intelligent Machinery* (1948), *Checking a Large Routine* (1949), *Computing Machinery and Intelligence* (1950), and *Digital Computers Applied to Games* (1953) [363, 366, 364]. In all of them, crossing different matters and during different periods, we are able to notice similar interests by the author, accumulating to the morphogenesis studies. Just confront "pattern or structure due to an instability of the homogeneous equilibrium", "system of reactions and diffusions in a sphere

a *naturalistic* stance, that of the real and spontaneous imbricating texture of biology. Turing was able, in one such way, to explore biological-chemical concerns and research pathways *On Growth and Form* [357] (D'Arcy Thompson's book that much impressed Turing in his early youth) with sterling original insights from computer science, at one time when Darwinian natural selection and Mendelian genetics were about to enter the time of the DNA structure discovery by Watson and Crick (1953) [391]. As for the case of von Neumann, the rendering, also in rather hard conditions, of *The Computer & the Brain* (1958), is equally so a condensation of permeating ideas from many different fields the author had contact with and even has come to found. In this book, envisioned to depart from a description of the computer – the conventional and unusual basic operations of the analog procedure, and also digital procedure markers, combinations and embodiments, types and basic components, parallel and serial schemes, etc., building essentially to characteristics of modern digital machines – evolving to a consequent part on the brain – a simplified description of the function of the neuron, the nature of the nerve impulse, stimulation criteria, and the problem of memory, casting afterwards the equivalent digital and analog parts in the nervous system –, is where von Neumann ultimately ties the knot between biological-neurology causality and computer systems design architecture<sup>43</sup>. The proper historiography's development

---

and in a ring of cells", "difficulty mainly concerned with organisms which have not got bilateral symmetry" [365], "*P*-symmetry" [365] with almost periodicity and commuting groups of the kind of symmetric operation exposed in the main diagonal (as in a Cayley table), and on von Neumann general theory (which applies to every group in all series expansions, free from topological assumptions, with the property of closure, or continuous functions in a certain topology). The same goes for continuous symmetry of mathematical structures and *n*-dimensional differential manifolds, including the work on machinery: "stationary-wave patterns tables obtained with the aid of the Manchester University Computer", "somewhat artificial chemical reaction system", and "the linearity assumption"[365]. Also the last section "13. Non-linear Theory. Use of Digital Computers"[365] and the recognition that "most of an organism, most of the time, is developing from one pattern into another, rather than from homogeneity into a pattern"[365] while recognizing that "the difficulties are, however, such that one cannot hope to have any very embracing *theory* of such processes beyond the statement of the equations"[365] is something that points to a reflective anamnesis, to the time when Turing was thinking on the theme of large routines and general properties of computer programs and programming languages semantics, most generally of mathematics as a medium for the specification of systems, life sciences systems included.

<sup>43</sup>John von Neumann in *The Computer & the Brain* [379] (1958), ten years after the publishing of *A Mathematical Theory of communication* [336] (1948) by Claude Shannon – and building not only from Turing's implications (1936), but from roughly the same year as computation theory, Shannon's digital circuit design theory *A symbolic analysis of relay and switching circuits* [335]

of the idea of natural selection in Charles Darwin is helpful to penetrate into the different, extremely laborious and seemingly insurmountable stages Darwin

---

(1937) –, comes to present a summarized admittance of an envisioned future, not just the setting principles of the modern, non-overcome, computer (von Neumann) architecture. Neurobiology and the nervous system, with or without encompassing the mind as a subject, with or without charting monism *vs.* dualism, truly has become the axiomatic & axiology of contemporaneous (digital) computation theory. The triad – information, computation and neurobiology – has been paced with the operative triad – communication, memory and logic gates –, as recognized by Raimond Kurzweil [379] (2012), one of the proponents of strong AI (and posthumanity transhumanism under the *motto* "Singularity is Near: When Humans Transcend Biology" (Kurzweil, 2005); "transhumanism" was, in point of fact, a concept that conquered the arena in the year of 1957 by the biologist Julian Huxley). Paul and Patricia Churchland's (2<sup>nd</sup> Edition) [379] preface insights, more into historical-connectionism and interdisciplinary-science agenda, are also very interesting to follow, namely von Neumann's laying open of "complete codes" meaning "machine language programs" and "short codes" meaning "high-level programming languages", in essence programming languages seen as a sort of genetics of computation, i.e., the idea of programming languages as deciphering data machine sequences, as if Boolean values were (artificial) genome combinations. On the overall, though, the connectionism illusion is perceptively stronger (2000) – the labelling of the brain as a "massively parallel analog machine" [379] and the praising of von Neumann as "the Newton of the mind" [379] are bounced off expressions –, while theoretical boundaries are seen more balanced in Kurzweil [379] (2012), where we can read, within justified and correct *limits* that "a von Neumann machine can simulate a brain's processing. The converse does not hold, however, because the brain is not a von Neumann machine and does not have a stored program as such." [379] (Kurzweil, 2012). Under the big picture, the main lessons to be learned from von Neumann's incomplete book (originally a Yale's Silliman lectures draft), are, straightforwardly, the unbalance between the digital computer, with advantage in processing speed and logical depth, and the brain, with advantage on logical breadth and parallelism, based on the comparative observation that nature produces large efficiency without complete serial memory processing, even if self-terminating and not exhaustive.

In our view, von Neumann's book attains to the perfect representational theory of the utmost *limit* in mathematical and philosophical terms, as if it was drawn over the *critical* line of the Turing-Church thesis, stressing to the utmost the statistical, procedural, and mechanistically orientated pattern matching stored-program and hierarchical programming languages digital computer architecture, mirroring, in *reflection*, the brain. But this does not mean that the brain, and even less the mind, were wholly *reflected* as an object, and, indeed, it should even be conceivable, in our perspective, a different computer architecture model to test the limits of the brain, mirroring it with an equivalent *U*-Turing machine displayed as a cortex of organized dynamic images hierarchy, represented as a lower bound to the mental activity, as computable numbers are to the real number system ( $\mathbb{R}$ ; +;  $\cdot$ ;  $<$ ). In essence, thus, the Princeton architecture may just be an *imaginary* representation of a bound from the above *U*-Turing-machine equivalent, while in truth a non-digital *U*-Turing-machine with dynamical images hierarchy, a bounded from below in relation to the brain (and the mind) processing unit or computational cortex, physically *connected* to the phenomenon of photons and light, can be more constituent of a coefficient, i.e., a *real* root, and therefore a proper upper bound of the Turing-Church thesis. The *hypothetical* equivalence of dynamic images or computational cortex with the computability thesis, such as the Turing-machine and  $\lambda$ -Calculus, would thereupon, most naturally, become an indispensable matter of investigation.

went through, until arriving at the scientifically-based natural selection mechanism, which states, simply, that it guards a tendency "only to make each organism as perfect as, or slightly more perfect than the other inhabitants of the same country, with which it has to struggle for existence", although "not producing absolute perfection" [94, 93]. One such consideration should be contemplative even if considering one such "most perfect organ as the eye"[93]<sup>44</sup>, something that was to be shared by strong AI holders or defendants of the *procedural* view all alone. In point of fact, we learn from Dov Ospovat that "Darwin continued to believe in perfect adaptation until the mid-1850's" [286] almost two decades after the five year's voyage on the *Beagle* and, furthermore, in one almost as epiphanic as epigenetical moment, that "from September 1854, when the investigations on barnacles were concluded, until June 1858, when Wallace's letter arrived announcing his independent formulation of the theory of natural selection, Darwin worked full time on his species theory. In 1856 he began writing the big book, now published as *Natural Selection*, which he abandoned after the receipt of Wallace's letter in order to produce quickly an abstract of it – the *Origin of Species*." [286] This serves to explain that even coming to contact with the study of serial homologies by Cuvier (1769-1832), on the background of the tradition overture by Lamarck<sup>45</sup> (1744-

---

<sup>44</sup>John van Wyhe, 2002, The Complete Work of Charles Darwin Online. <http://darwin-online.org.uk/>

<sup>45</sup>Even though we are scrutinizing one *connexio* (in Hume's proper sense), between Darwin's natural-artificial philosophy and Turing's artificial-natural philosophy, and more circumspectly, the homology between the mechanism of natural selection and recursion theory equivalent Turing-machine, both sharing basic general-law given decidability over arbitrarily randomness, it is very interesting to note that Turing's later work on *Morphogenesis* accomplishes to restore a research line buried in Lamarck – the idea of one (mathematical)-alchemical complexifying force freeing the emergence of organisms up in the "ladder" of complexity – and also seen later, by the time of *The Origin of Species* (1859), in Richard Owen's work – the restless stressing of complexity, i.e., the idea that evolution has its own developmental evolution, a remarkable standpoint in the XIX<sup>th</sup> century, not only decades before the modern synthesis, and one century before the extended synthesis, but prior even to Mendelian genetics, not to mention recombinant DNA technology in molecular genetics – all in all an often forgotten debate. John von Neumann's later work also bears resemblances with this line of research, specifically the idea of a Universal constructor, that is, a self-replicating machine in biological-form automata, whose integral explanation with full details would only come forth with the publishing by Arthur Burks (an ENIAC senior engineer) after von Neumann's death, of the book *Theory of Self-Reproducing Automata* [378] (1966). The book is divided into two parts. Part one "Theory and Organization of Complicated Automata" comprises five lectures, of themes dispersed such as computing machine powers and problems of hierarchy and evolution, bridged upon statistical theories of information. Part two "The Theory

1829) of the "inheritance of acquired characteristics", a whole throughout amplified debate between (catastrophist) functionalism proposed by Georges Cuvier and (pre-evolutionary Goethean) morphology advocated by Geoffroy Saint-Hilaire (1772-1844), until phylogeny in relation to analogy and homology was defined in developmental contours by Richard Owen (1804-1894), it was a painstaking, almost punctilious, difficult process for Charles Darwin to explore a fully inductivist *serial, analogical, and homological* natural philosophy. The best endorsement for this observation is the fact that geology, the most abridging field and the most influential arena to have acted on Darwin's ideas, namely through deep time plutonist conception of James Hutton (1726-1797), and in decisive manner, Charles Lyell's<sup>46</sup> (1797-1875) uniformitarianism (or doctrine of uniformity) was extensively, although in neutral form, yet theological, teleological, and rationalist; thus, still estimable in the panorama of Kantianism, very far away, though, from the principle of uniformity of David Hume. The full skeptical and inductivist principle of uniformity of David Hume resounded patently in Darwin's natural selection and also in artificial recursion (Turing-machine equivalents). Surely, as we discuss causation and spontaneity, it seems that the Theodosius Dobzhansky's essay title – *Nothing in biology makes sense except in the light of evolution*<sup>47</sup> (Dobzhansky, 1973) – is

---

of Automata: Construction, Reproduction, Homogeneity" clearly demands expanded contents, bringing upon concerns such as a system of design-states (29) with a general transition-rule, also the design of basic organs, the tape and its control, up to the idea of automata self-reproduction. Fundamentally, it should be noticed that we are dealing, in Kantian and von Neumann's terms, with a "synthesis of automata by automata"[378].

<sup>46</sup>Lyell's uniformitarianism is pivotal, guarding unnoticed countenances. The science historian Reijer Hooykaas (1906-1994), having distinguished "different conceptions of the history of the earth "[187] – "a. non actualistic" (a1. "differ in kind and energy"; a2. "differ in kind but not in energy"), "b. actualistic" (b3. "differ not in kind, sometimes differ in energy"; b.4 "neither in kind, nor in energy") [187] – elaborated on Lyell's uniformitarianism, attending to law, methodology, kind, and degree. Now, the evolutionary biologist Stephen Jay Gould (1941-2002) appointed uniformitarianism to hold two methodological assumptions – uniformity of law across time and space, and of process across time and space – and two less accepted but dependent substantive hypotheses – uniformity of rate across time and space, and of state across time and space –, which on the overall (even if dissociated from Hume's principle of uniformity), help us to understand not only the "Newtonian root of the synthesis" [57] and the geological-geometrical *transformational* analogy, but also to apply this tenets from philosophy of nature to the philosophy of the artificial. The field of complexity in computation convenes natural-physical orders as such presented, earnestly foreseen by John von Neumann (utterly helping us to grasp Churchland's labelling of von Neumann as the "Newton of the mind" [379]).

<sup>47</sup>The theistic evolutionist Dobzhansky got inspiration for the title from the vitalist evolutionist

expandable to the realms of logic and physics: semantic truth, in Tarsky's sense, either in correspondence or deflationary theories, seems to have "collapsed" to less than neutral, fully inductivist logical *adequationes*, and, equally so, natural laws in physics were made dependent on a T-schema, wherein inductivist definitions are subject to coherence into models of explanation. These *adequationes* are dynamically moving, i.e., related with both metaphysical movement and physical motion, in one expression, with the Aristotelian "efficient cause". Darwin's natural selection (1859), Turing-Church (1936) both ideal machine-model and equivalent  $\mu$ -recursive functions, in the prolongation of Hume's principle of uniformity, have contributed to the investigation of the difficult antinomy between causation and spontaneity, but in any way has it been unveiled. If anything, both causation and spontaneity have retreated to paradoxical physical naturalistic instances, of the kind seen in general relativity facing quantum mechanics or in quantum physics alone. In effect, Darwin's natural selection "difficulty of distinguishing between varieties and species"[93] holds resemblance with paradoxes and Russell's type theories in logic (*Appendix B: The Doctrine of Types*, Bertrand Russell, 1903) [399], in the sense that if, naturalistically, varieties considered as elements are more numerous than species, logically each variety considered as a potential subset consistently outnumbered any number of elements, as if the number of species grew always faster than the given varieties, but with the impossible condition of nature being – as the class of all sets, but not a set itself – a special class of classes that does not belong to itself. Ever since and at the time present circumvolved natural-artificial studies exhibit the functionalist paradigm whereby are considered:  $i$  – as the type of individuals –,  $()$  – as the type of propositions –, and  $(A1, \dots, An)$  – as the type of  $n$ -ary relations over objects of the respective types mentioned in  $A1, \dots, An$  [117] –, therefore approximating the logicist (Frege, Russell), semantic (Gödel, Tarski), type-functionalist (Church), intuitionist-constructivist (Per Martin-Löf) views, but essentially, in the hidden depression of the curve, the works on the mechanism of natural selection by Charles Darwin and of a  $U$ -machine by Alan Turing. In truth, attaining to the functional relations in equivalent transfinite

---

Theillard de Chardin. Chardin's judgement is very interesting in its original form, due to its *transformational* and *topological* insight: "Evolution is a light which illuminates all facts, a curve that all lines must follow." (*Le phénomène humain*, 1955).

set theory, type theory clearly bestows a continuous function from any (*analytically* rigid) "square" in a Turing-machine, to the (*dialectically* continuous deformation) of every-possible end-points in Darwin's mechanism of natural selection. If this is taken as a valid homotopy, then actually Darwin's natural selection is Turing-complete, and therefore naturally incomplete within the bounds of nature itself (as in Gödel's Incompleteness Theorems, this time nature taking the place of mathematics)<sup>48</sup>, although not hurting its all-round, comprehensive and many-sided topological validity and logical-mathematical correctness. Besides being odd to discover artificiality's mechanism (computation and decidability) in first grounded natural mechanism by Darwin (except recalling that it was industriously a *mecha-*

---

<sup>48</sup>Gödel's incompleteness theorems are, indeed, valid for natural philosophy: either by demonstrating the critical limits or inherent limitations of 1<sup>st</sup>) every formal axiomatic system in the lines of Russell & Whitehead's *Principia Mathematica* arithmetic of natural numbers insufficiency to produce, by any given consistent axiomatic system orderly and correctly deduced by a correspondent effective procedure, the truths about arithmetic, and, therefore, new theorems; or 2<sup>nd</sup>) (what is much more powerful and, in principle, limiting) the critical insufficiency of the very same axiomatic system in the lines of Russell and Whitehead's *Principia Mathematica* arithmetic of natural numbers to prove or demonstrate, as a theorem, its own consistency – even though, for Gödel's mathematical philosophy as for Kant's critical philosophy, these critical limits and inherent limitations, by order, in *theoretical-mathematical-analytic* (Gödel), and *practical-dynamic-dialectic* (Kant) domains, are *reason enough* (the lower bound now the utmost upper bound to something unknown or the infinite) to be extended, beyond proof or any theorem – holding, if we wish to maintain some semantic truth and logical validity concepts, the inherent *consequence* that consequences are no longer to be understood fully as consequences –, and thus, causality found to be spontaneous along Hume's natural philosophy, and, to a certain extent, Darwin's too (a sort of overlapping new argument from Hume's inductivism against the overpass of the limits by Kant and Gödel). As for the case of Turing, having integrated Gödel's results in his work, and having always worked on the decidable, consistent and logically valid lower bounds set of by both the 1<sup>st</sup> and the 2<sup>nd</sup> incompleteness theorems, makes him a clear representative, in the lines of a critical philosophy and necessarily *aporetic* Kantianism, of *pure reason* (and computability's *critical* limit).

Unexpectedly, though, it might be said that, in the lines of both natural philosophy and complexity in computation (Turing, von Neumann) there is always the *possibility* of reclaiming a new overridden argument by Lamarck-to-Owen order, by which nature itself would "demonstrate", as *morals* in Kant, the insufficiency of incompleteness itself, as incompleteness did to its consistency. This is also the reason why Tarski's semantic truth is not, in truth, *naturalistically* adapted to the truth, but *necessarily* so, therefore bounding the critical limits of logic in relation or correspondence with semantics, and why natural philosophy's "efficient cause" subject to motion or movement *adequationes* is, of utmost and maximum skeptical, inductivist, and relativist nature (an impossible to overthrown Humean argument at last). Hume's philosophy and skepticism is, thus, seen as having been much depreciated. This is also why David Lewis' (1941-2001) exposed "principle of Humean supervenience" [239] targets critical philosophy and western philosophy where it could possibly be felt more forcefully.

nism beforehand), what is most astonishing is to discover, as seen in modern and contemporaneous physics, a "collapse" of the function considered as transformational end-points, as these are permeated with "trans-finite matter" so to say, as if Darwin's natural-artificial philosophy appointed at the end to full-blown atomism, and consequently Turing's artificial-natural philosophy was its most upper bound "integral" standard possible method for interpretation, itself pursued intensively by Turing in *Morphogenesis* studies. As in the Ancient Greek problem of squaring the circle aiming to find the same area number, *ideally*  $\pi$  as transcendental number (Leibniz, 1682), the same is found herein: being the use of a compass-and-straightedge construction in a finite number of steps the representation of a Turing-machine, it is impossible to achieve one such goal, even though methods of approximation offer non-perfect accuracy (though never the root of a nonzero polynomial equation with equivalent integers). This sums up to saying that, inasmuch as decidability (and the class of computable numbers) is a sort of last tangent to incommensurability, the mathematical Ancient Greek doctrine of proportionality for geometric magnitude is represented now, namely after Gödel's incompleteness theorems, by computable numbers in (type theory) "squares" as *approximations* or equivalents, although never accurate. As computation and artificial philosophy have evolved to complexity theory, natural philosophy transcendentalism has complexified itself to one exceptional status, with a far greater antinomomy's *distance* – both in Kant's philosophy as in Euler's mathematics avowals –, and has become itself a field with possible conceivable *practical* passage from *morphogenesis* (as found in Turing: *empirical, categorial, phenomonic*), to *cosmogogenesis* (as found in Gödel and von Neumann: *pure, speculative, noumenic*) – largely more *synthetic, schematic, architectonic* in system sciences, hopefully within the critical theoretical limits of the Turing-Church thesis<sup>49</sup>. The class of partial  $\mu$ -recursive

---

<sup>49</sup>Alonzo Church never ventured to machinery computer science beyond mathematical logic [254], Alan Turing never ventured beyond *Morphogenesis* to strictly cosmological problems, but, on the contrary, Kurt Gödel and John von Neumann unraveled some problems factually and stringently in the cosmological domain. Gödel was highly influenced by the intellectual environment of the Institute for Advanced Study in Princeton from 1933 to his death in 1978 (apart from his role as a Professor in the School of Mathematics from 1953 to 1976), having contacted with many cosmology physicists (Hermann Weyl, H. P. Robertson, *et. al.*), the most prominent Einstein, of whom he became a dear friend. Bearing this *cosmological* mark, Gödel published: "An example of a new type of cosmological solutions of Einstein's field equations of gravitation"



functions, from natural numbers to the very same natural numbers, equivalent to computable numbers and  $\lambda$ -Calculus, serves, thus, as a demonstration of the "collapse" of causation, and the *supremum* or least upper bound field for spontaneity in nature. Just so, it is thereby exhibited an *adequate* example of computability's or Turing-machine's adaptability, as if affected by natural selection itself, to other forms of equivalent postulations. How, up to which point, and wherein are mathematical processes and computable functions *morphogenetic* – and, possibly, in extension, with affiliated methods and limits, *cosmogenetic* – is, all the more so for the time present, a very pertinent interrogation. Now, computability theory and the *computus* paradigm, following the Turing-Church thesis (1936), in relation to causation and spontaneity, bounded by *morphogenesis* and *cosmogenesis*, is set to continue the study of homologies from natural-artificial trended to artificial-natural trended philosophies, thus exposing an evolution from *serial* homology in biology, to *analogical* homology in mathematics (or, strictly, any sequence of algebraic objects accountable to both *calculus* and *computus*). If it is admirable how from the slightest variations were to emerge the spectacular diversity of the natural world, it is also marvelous how  $\mu$ -recursive functions, a class for partial functions, is so lively capable of unifying the concept of science by computational means.

---

(Gödel, 1949), "A remark about the relationship between relativity theory and idealistic philosophy" (Gödel, 1949a), "Rotating universes in general relativity theory" (Gödel, 1952), with an imperishable interest in the continuum as in "The consistency of the axiom of choice and the generalized continuum hypothesis" (Gödel, 1938), "Consistency proof for the generalized continuum hypothesis" (Gödel, 1939a), and "What is Cantor's continuum problem?" (Gödel, 1947, 1964). Timely to a *cosmological* comprehension, in proper Kant's terminology, is the idea "we learn in remark 13.9.72 that Gödel was committed to 'the method of bold generalization' as he called it, recommending that philosophers have the audacity to 'generalize things without any inhibition.'" [216], which articulates with his appraisal for the reflection principle in set theory, i.e., the possibility of finding sets that resemble the class of all sets. Imparting still on this aspect, Gödel's critically dialectic transcendental mathematical philosophy meets, through mathematical Platonism and cosmology together, the *summum bonum* of Kant's (moral and extended) dialectic. As for the case of von Neumann, and besides extensive work in the most diverse fields – logic, theory of sets, operators, ergodic theory, almost periodic functions in a group, rings of operators, continuous geometry, design of computers, automata theory, numerical analysis, theory of games –, was also able to establish knowledge in the foundations of quantum mechanics, astrophysics and even meteorology.

[Thesis: there belongs to the world, either as its part or as its cause, a being that is absolutely necessary.] & (conjunctive, non-contradictory: apparently the converse of *Principia Mathematica*'s theorem of non-contradiction: \*3 · 24.  $\vdash . \sim (p. \sim p)$  ... but now in the negation form  $\sim . \sim (p. \sim p)$  and, thus, openly equivalent to  $(p. \sim p)$  ) [Anti-thesis: an absolutely necessary being nowhere exists in the world, nor does it exist outside the world as its cause.]:

In *computus* age, and considering the *fulcrum* year of 1936, the idea of *ens realissimum* (Kant, [A576/B604]) [211, 63], one such being that contains all reality and predicates, responding to transcendental aesthetic and the inherent transcendental ideality of space and time, and also to the systematic bearing of transcendental analytic (prior to dialectic), in Kantian terms, seems to have transformed the typical of *calculus* era rational psychology, cosmology and theology (the rational soul, the totality of the world, and God), including the general taken impossibility of synthetic *a priori* propositions in metaphysics, to one de novo, *hodiernus computus* dialectic (not ideally transcendental, thus not pure with the seat in illusory reason as *prototypon* transcendental, and very negligent towards the interest of reason in its self-contradictions, or to the necessity imposed upon pure reason of presenting a solution of its transcendental problems). If it wasn't for the natural amphiboly of the conceptions of reflection and transcendental dialectic naturalness (Kant, [A270/B326]) [211, 63], it was hardly understandable why so, for the equivalent *computus* era transcendental aesthetic of space and time bearing judgements (classical mechanics, general relativity, quantum physics, kinetic theory, electromagnetism, strong and weak interactions in the standard model of particle physics), with all of which computation permeates, outright establish clear *critical* limits to the idea of a science which shall determine the possibility, principles, and the extent of *a priori* human knowledge (Kant, B III, Introduction) [211, 63]. *Computus* dialectic as so, a computational metaphysics with *computus morus* postulates in the place of morals (confronted with Kantianism), seems to have reclaimed a sense of (technological and informational anew) religion not properly within the bounds of pure reason. Most often (computationalism opposed to the critical limit of computability), it corresponds to a dogmatic use of pure reason. And, what is more, one such *computus* dialectic (to the *limit* of corrupting

the original sense of *διαλεκτική* "dialectic") can also be said to have illegitimately expanded according to the proper *acumen* (both transcendental dialectic and *summum bonum* Platonist) interpretation by Kant, where practical reason assumes the formula guidance of the universal law of nature (*Groundwork of the Metaphysics of Morals* 4:421) [211, 63], else called categorical imperative (supposedly objective, rationally necessary and unconditional), once, in its architectonic, there is, clearly, a misuse of speculative reason in relation to the *computus* power, in any way being *computus* an analogy of experience with practical philosophy or morals. Neither is it, anyway, with the system of transcendental ideas and of pure reason as the seat of transcendental illusory. The reasons for this are the following: besides illegitimately extending beyond, most strongly Gödel's 2<sup>nd</sup> incompleteness theorem (1931), but also the core of the Turing-Church thesis (1936), therefore corrupting one *hodiernus* formally constituted analytic of principles grounded on the idea of transcendental logic (Kant, [A131]) (even more so and possibly containing an irreparable breakage of the transcendental clue to the discovery of all pure conceptions of the understanding (Kant, [A67]), precisely due to Gödel's incompleteness theorems and the Turing-Church thesis with negative results to the Leibniz-Hilbert-Ackermann *Entscheidungsproblem*), it is also patent that a system of transcendental ideas and the seat of transcendental illusory are absent in the *computus* era, i.e., it should be recognized, in fact and inversely, as the proper field where any dialectic (transcendental) procedure is truncated. What is more, transcendental analytic is dead. Indeed, we go as far as to say that, in essence, Gödel's 2<sup>nd</sup> incompleteness theorem – for it presupposes the 1<sup>st</sup>, and the 1<sup>st</sup>, implicating the negative answer to the decision problem, endorsing (even if for a time unreceptive)<sup>50</sup>, in anticipate groundwork, the Turing-Church thesis or com-

---

<sup>50</sup>Gödel's prospects on the power of recursion theory, being himself a forerunner of the discipline, were not particularly high, and neither was his reception of recursion theory in relation to its computable functions and extensional powers, something that drastically changed towards the end of his activity. Actually, after the height of his reputation following the incompleteness results, Gödel gave lectures at the Institute for Advanced Study in Princeton, and specifically *On Undecidable Propositions of Formal Mathematical Systems* (Gödel, 1934) (which are edited as Gödel's lectures notes taken by Kleene and Rosser, Kleene himself being a forebearer of recursion theory). We read from Solomon Feferman's *Gödel's Life and Work (Kurt Gödel Collected Works, volume I, Publications 1929-1936)*: "These notes cover much the same ground as 1931, though now starting with a form of second-order arithmetic. One of the main points of interest in 1934 is Gödel's introduction of the notion of general recursiveness, following a suggestion of

putability equivalents – defines a new censuring and demanding *critical limit* of (transcendental) analytic towards (transcendental) dialectic. What is meant by this is that this *computus* era's new censuring and demanding *critical limit* is now, just as rightly as devastatingly, founded on and broke down the (believed) mathematical *terra firma* of the synthetic *a priori* judgements, the proper (*theoretic* and *practical*) underpinning of critical judgments, a sort of Archimedes lever of the entire transcendental philosophy. The fact that the boundaries and limits of critical philosophy have been imperturbably unquestioned after the incompleteness theorems by Kurt Gödel (1931) and the negative answer to the *Entscheidungsproblem* posed by David Hilbert (1928) really should be thought of as a fatal flaw. Synthetic *a priori* judgments (*computus*, beyond *calculus*, now included) make the

---

Herbrand." [161] Understandably, this was later set equivalent with Church's  $\lambda$ -definability and "effective calculability" (1936). It is, therefore, quite explanative but also exegetical, by virtue of what has been said, why Gödel resisted to Church's thesis, having on his own settled recursion functions (Herbrand, 1932; Gödel, 1934) and Church's  $\lambda$ -definability (Church 1932, 1936a) on its own being a functional (both theoretic abstract and implementation concrete) progression, and why did he not accept "Church's thesis", on this only recognizing its importance after Turing's publication on computability in terms of machines (Turing, 1937), as for the rest the former and latter were afterwards shown to be equivalent, naturally along with general recursiveness, as trusted in Gödel's exploratory work. More it is strange if we read from Solomon Feferman, now in the "Historical Introduction" of the "Mathematical Logic" volume of the *Collected Works of A. M. Turing* that it was originally Church who was "promoting a universal system for logic and mathematics in the framework of the lambda ( $\lambda$ )-symbolism for defining functions, and set Kleene the problem of developing the theory of positive integers in his formalism, using an identification of the integers with certain  $\lambda$ -terms. The initial steps were rather difficult (even the predecessor function posed a problem), but once the first hurdles were cleared, Kleene was able to show more and more number-theoretic functions definable by the conversion processes of  $\lambda$ -terms. But Church's original system was shown before long (by Kleene and Rosser in 1934) to be inconsistent, and attention was then narrowed to a demonstrably consistent subsystem, which came to be called the  $\lambda$ -calculus." [363] The story tells that consistency and definability were showing non disproving results from the calculus, making it more and more convincingly "effectively calculable" under the eyes of Church, who finally, after seeing it pass any function test contrived by Kleene, eventually up to the point of testing it with diagonalization, the outperforming of which led Kleene to have become "overnight a supporter of the thesis" (Kleene, 1981, p. 59), contributed to their final common avowal. Gödel should have embarked in the all-involving coherent reception of  $\lambda$ -calculus (especially for someone who had disproven arithmetic from the inside through diagonalization) but, instead, he thought of  $\lambda$ -Calculus to be "thoroughly unsatisfactory" and, in a much more praising label, "heuristic". We believe one such attitude to have been motivated, certainly among other decisive factors, by the most important aspect of Gödel's belief on the impossibility of lower bound *calculus* effectiveness and, therefore, of disbelief of  $\lambda$ -definability prevailing over diagonalisation, being in his mind mathematical functions also part of arithmetic related systems, and diagonalization proven to be capable of ruining its edifice.

passage from analytic to dialectic, theory to practice, mathematical to dynamic spontaneous and innate cognitive capacities (with Aristotelian "efficient cause" of physical *motion* and metaphysical *movement* both conjectured). And, as the Archimedes lever, synthetic *a priori* judgments represent the proper machine in image, that of a beam or rigid rod with the pivot point at a fixed hinge, or else said *fulcrum*, the reason why we have been calling attention to the equivalent *fulcrum* year of computation theory (1936) in relation to metaphysics and philosophy of science, relating, in essence, philosophy of nature with the archetypal Turing-machine. In its most natural form, a lever is, except for two-way probability bit-like dual equilibrium, diagonalized<sup>51</sup>. As is recognized, a lever can, nonetheless, be made equivalent in exerting forces if in the exertion of a great force over a small distance, at the other end is compensated with the exertion of a small force with a greater distance. This is the reason why the *continuum* problem is fundamental in Gödel to perceive how the greater distance (mathematical Platonism) imposed over a little weight (both arithmetic and mathematics related 2<sup>nd</sup> incompleteness theorem) was a necessary judgment for mathematical realism in the line of Plato, Leibniz, Kant, and, fundamentally, to make *adequitata* – in the sense also of equivalence (thus, non-antinomic, and mathematical-practical dialectic) – Gödel's mathematical philosophy to metaphysics. In one such turn, Gödel's mathematical philosophy was made *adequitata* not only to (realist or Platonist) mathematics but also to (ideal, critical transcendental) philosophy (moreover helping to understand the Viennese logician interest in Kant's philosophy of time born on "Observations on the relationship between Einstein's theory and Kantian philosophy" [385, 161], and also his late approval of recursion theory as a solid mean "On a hitherto unutilized extension of the finitary standpoint" (Gödel, 1958) [161, 160]. This is also patent in late texts: "Postscript to Spector (Gödel, 1962)" [161, 160], "What is Cantor's continuum problem?" (2<sup>nd</sup> Edition, Gödel, 1964), "On an extension of

---

<sup>51</sup>diagonalization is to be made central, as never it was thought that the method responsible for the incompleteness theorems and the end of the Hilbertian formalist and full-blown decidable and complete program on mathematics, was to be exactly the same responsible for the main process in computability and equivalent theories. Computers are, indeed, diagonalization methods insofar as they render unification with substitutions rather than proof, even if the diagonalization method behind computability has disproven something as profound as the foundations (presumably both decidable and complete) of formalized systems containing elementary arithmetic.

*finitary mathematics which has not been used*" (Gödel, 1972) [161, 160], "*Some remarks on the undecidability results*" (Gödel, 1972a) [161, 160], and "*Remark on non-standard analysis*" (Gödel, 1974) [161, 160].

One such conclusion is made noticeable by various sound arguments: 1) firstly, if the mere logical form of cognition containing the origin of pure conceptions *a priori* which indicate the synthetical unity responsible for the empirical cognition of objects, hence the form of judgements converted into a conception of the synthesis of intuition (Kant, [A321-B378]) is admitted, then, by the 1<sup>st</sup> and 2<sup>nd</sup> Gödel's incompleteness theorems (critical *mathematical* theoretical judgements), and the Turing-Church thesis and negative answer to the *Entscheidungsproblem* (critical *dynamic* practical-empirical judgement), both are made our form of judgments, i.e., the mere logical form of our cognition (Kant, [A321-B378]); 2) secondly, being *logica* incompleteness and theoretical *computus* a critical limit to pure reason, there is not in any way an warranting consideration from the form of syllogisms, when applied to the synthetic unity of intuitions, following the rule of categories containing *a priori* conceptions, quite the contrary, i.e., the understanding of the totality of experience according to principles having been denied by *proof* – never of the syllogistic sort "Caius is mortal" (Kant, [A322]), but instead from diagonalization –, with *universalitas* and *universitas* now being, along with the rationalist, logicist and formalist conceptions of the unconditioned, seriously faulted, most irrevocably from Gödel's 2<sup>nd</sup> incompleteness theorem, inevitably brings about the impossibility of one architectonic of pure reason; 3) thirdly, apart from the blunt fact that (quantified) logic or any axiomatic system of arithmetic, and, thus, mathematics, cannot denote intrinsically as a predicate of a thing considered in itself, and much to the contrary, being absolutely impossible one such endeavour, against the *calculus* era *toto caelo* (Kant, [A325]) that the philosopher from Königsberg lived in, what is made explicit from the tenets of the *hodierna* era of *computus* (1936), is a debacle *in concreto* of the sovereign validity of the absolute (complete, satisfiable and sound, consistent, furthermore decidable and tractable) use of transcendental reason; 4) next in order, it also should be taken into account that the *impossibility* of the absolute totality in the synthesis of conditions, past the staging of the 1<sup>st</sup> and 2<sup>nd</sup> Gödel's incompleteness theorems (1931), therefore breaking the rational unity of the phenomena (Kant, [B383]) and moreover its terming with the unity

of understanding – (*ignoramus et ignorabimus* plead against David Hilbert's "Wir müssen wissen, wir werden wissen" ("We must know, we will know") – inevitably and *critically* reconfigures reason to the lower bound of the grounds of possible experience (immanent), instead of to the higher bound of the unity of understanding (transcendental); 6) surely, the profession by which conceptions of pure reason regard empirical cognition by means of the absolute totality of conditions is still a sound argument – in accord with the idea that soundness prevails if and only if its inference rules prove formulas that are valid with respect to its semantics, in one such case, *dialectic* reason (in the sense that reason naturally holds transcendental ideas) –, although it can't be said *critically* valid, to the extent that Gödel himself – paradoxically Brutus and Julio Caesar simultaneously in the tribunal of pure reason (Kant, [A751-B779]) – never introduced mathematical Platonism or any form of transcendental practical *a priori* judgments, herein included the proper analytic of mathematics, in any of his theorems, proofs, or demonstrations; 7) it would be easy to affirm that this asserts for the proper cleavage between mathematics and philosophy of mathematics, with the only alternative of possibly including *bona fide* in the foundations of mathematics, but a definitely less easy flip-side for Gödel would have been to have established Kant's transcendental doctrine of the faculty of judgments or analytic of principles, on to the system of all principles of pure understanding to the necessity of transcendental ideas and dialectic, from the 1<sup>st</sup> and 2<sup>nd</sup> Gödel's incompleteness theorems, *as if* (in practical Kantian sense of a form of analogy towards a maxim of regulative judgement) the 1<sup>st</sup> theorem insufficiency of "effective procedure" to prove all truths in arithmetic related systems, would match *analytic*, and *as if* (in practical Kantian sense of a maxim of regulative judgement now as a regulative principle in transcendental dialectic) the 2<sup>nd</sup> theorem, showing that any arithmetic related system cannot prove its own consistency, would match *dialectic*, in which case Gödel's *practical*-mathematical philosophy would have not only destituted theorems to mere analogies, found in the collapse of analytic a new transcendental nature, but also and at last, corresponded transcendental nature with its debacle, and this void with the *dogmatic* equivalent of negative theology; 8) surely, this would not be a realizable possibility for Gödel as a believer in mathematical Platonism and neither for Gödel the mathematician, but it has to be said that it would be just as hazardous to reckon upon the inclusion of mathematics on

which pure reason was born, and not just pure reason, in the antinomies of pure reason, collapsing, thus, synthetic *a priori* judgments, or conceding the *sublime* to negation, as admitting both, and at the same time, the incompleteness results and mathematical Platonism; 8) it is also emblematic that Gödel's influences reveal, in what concerns his philosophy of mathematics, the inescapable passage from noumena to phenomena, where from Leibniz's dogmatism and Kant's critical philosophy find Husserl's phenomenology, to the point that it is brought about a sense of a "Gödelian phenomenological turn" [216], which, undeniably, shows a sense of *nexus* from Kant's antinomies of pure reason, to Gödel's incompleteness theorems, being the the suitable image rescued in the *focus imaginarius* as nature's unity desired by reason ([Kant, B 673]), but with the capital and also inescapable truth that a systematic unity of all knowledge and the equidistant equilibrium of the antinomies poles are now forever thrown away and vanquished by the diagonalization argument in synthetic *a priori* judgements, with the result of reason itself, and not just its transcendental ideas, being contradictory and paradoxical; 9) *computus* has, therefore, changed the system of transcendental ideas in Copernican fashion, and computationalism (computationalism as the *dialectic* of the proper *analytic* of computability) appertaining to experience – trough (mathematical) principles of quantity and quality, and (dynamic) principles of relation and modality, however dealing with axioms of intuition, anticipations of perception, analogies of experience, and postulates of empirical thought, namely making use of the analogies of experience with substance (permanence), cause (succession), and community (simultaneity) to the extent of trying to find in the idea of an universal Turing machine pure reason in a black box –, is, quite the opposite, neither and nowhere to be found a procedure of pure reason; 10) again, it is the correspondent of the Aristotelian efficient (or moving) cause – concerted with metaphysical *movement* and physical *motion*, and in our perspective the rightly found locality or *τοπος* (*topos*) for the unfolding of the artificial from the natural –, that is seen to make born out, through relation and the analogy of experience, the dynamical principle, further disentangling, through modality, the postulates of empirical thought; 11) in this sense exposed, *computus* dialectic, foreign as it is to its analytic – the foundations of computability in the Turing-Church thesis (1936) – even if a system of transcendental ideas and the pure reason proper dialectic in *computus*,



neither transcendental analytic nor logic, are seen abiding the *critical* limits in computability and recursion theory, as described in Church's *An unsolvable problem of elementary number theory* (1936) & *A note on the Entscheidungsproblem* (1936) [71, 72], in Turing's *On Computable Numbers, with an application to the Entscheidungsproblem* (1936) [362], altogether in relation with the negative answer to the Hilbertian decision problem and the positive diagonally decidable and complete "effective calculability" [71, 172] and the Turing-machine model [362], imaginably resorting to discover all the pure conceptions of the understanding and their deduction (as described in Gödel's Theorem VI and "1<sup>st</sup> Incompleteness Theorem" in *On formally undecidable propositions of Principia Mathematica and related systems* (1931) [160, 161], and Turing's *Systems of Logic Based on Ordinals* (1936) [367] in relation to mechanical formal systems, searching for inherent completeness, consistency, and effective axiomatization, in the road to the analytic of principles, and in respect with the transcendental faculty of judgment (Gödel's Theorem XI and "2<sup>nd</sup> Incompleteness Theorem" in *On formally undecidable propositions of Principia Mathematica and related systems* (1931) [160, 161], and Turing's *Computing Machinery and Intelligence* (1950) [364, 368], does not give rise, anyway, to any just foundations for analytic in the *computus* age; 12) the *computus* era dialectic, and in particular the analog-to-digital passage in modern computation, seems to ultimately aspire to commute *sensorium dei* for *sensorium computi* in aesthetics, which mediates for a sort of digital and informational transcendental doctrine of elements, a demonstration of such being the failure to capture the inner subjectivity of the external and internal perceptions of space and time ([Kant, A26-B42], [B49-A33]), and the flaw of never contemplating mathematics in itself as an axiom of enumerable nature and neither the tautological – never transcendental except if seen as transcendental by the *phenomenological* seat (not exactly transcendental illusion) – nature of analytic, which permeates flagrantly with a substitution of practical *summum bonum* for practical *summum computum*, born out of diagonalization with truth-correspondence models, oddly reclaiming the conception of *ens perfectissimum*, in a sort of computationalist tomism; 13) the bearing of the question "How are synthetic a priori propositions possible?" (Kant, [B14–18, A158/B197]) – incidental to the question of "How is metaphysics possible as a science?" (Kant, [B19-B23]) – has held the critical test

for mathematics, the natural sciences and metaphysics, until the *computus* era, but ever since we cannot say that the question has changed, but rather the hardness of its answer and interpretation, now in turn and in itself being the paramount (self-)reference to the proper conflict of transcendental ideas and the antithetic of pure reason, besides having shifted the conception of pure reason by inference to a sort of transcendental induction, marked by the inalienable experiential, almost experimental nature of time.

### 1.1.3 Appendix on Virtuality

As in Kant's theory of knowledge and metaphysics and as perceived in the previous paragraphs, we understand that knowledge in the Turing-Church *computus* age – the Turing-machine model being descriptively complete in computational terms, as it contains the logic from compilers (translating the source code as a whole into machine code) and/or interpreters (translating the program one statement at a time from source code to machine code), all the way up to stored-programs [36] – is *peculiarly* constrained in mathematics and the empirical world, making it impossible to extend knowledge to supra-sensible speculation in metaphysics, herein contemplated the *image* of pure *extensionality* and a close observance on the axiom of extensionality -  $\forall x \forall y [\forall z (z \in x \Leftrightarrow z \in y) \Rightarrow \forall w (x \in w \Leftrightarrow y \in w)]$  – while producing an exceptionally strong (transcendental and non-transcendental) *illusion* in human-computer interaction and AI.

Taking hold of the essence of computable functions in its definability (Hilbert, Gödel, Church, Kleene, Post, Turing, and also Markov), in all its mathematical-philosophical *possible* pure analytic and synthetic dialectic concepts and judgments, it is extraordinary that this mathematical and dynamic antinomies in AI are widened by an *ideal* and *sensible*, pure and empirical manifold apperception (Kant, [A106]) by the effect of virtuality.

In short, the Turing-machine corresponds to a  $\mathbb{R}^2$  abstractedness, which, by the concreteness (and existential postulate) of any "a-machine" [362] (with programming languages and a sort of hierarchical detachment from machine code or functional  $\lambda$ -Calculus as detailed in the Turing-Church thesis), is set to be extended to a renewed abstractedness, now replenished and dense in  $\mathbb{R}^3$  space, an

edging transfinite space frontier in human-computer interaction.

In the first case with the Turing-machine and  $\mathbb{R}^2$  *ideality* we observe an Euclidean axiomatization in recursion theory and computation, and in the succeeding  $\mathbb{R}^3$  *ideality* intrinsic to the concurrent development of "a-machines" [362] and programming languages, we are prone to observe a Cartesian coordinates system revolution in computer science, *i.e.*, a system wherein signed distances to the point from a memory of translated directed lines, computed in the same unit of length, building each reference line a program with instructions – plural axes in the hierarchy from binary code to the programming language of the system -, find the point where they meet, technically called its "origin", not so much in the *geometrical* ordered pair  $(0, 0)$  – in configuration, conformed to the Kantian "orientation in thinking" [202] according to the model  $(\overbrace{x}^{\text{abscissa}}, \overbrace{y}^{\text{ordinate}})$  –, but instead in the *symbolical* Boolean  $(0,1)$ . A paramount difference, though, is that with this last *Cartesius* revolution with computer science, and for as many future advancements in the age of digital *computus* to come, we are and will be, contrary to the Cartesian coordinate system (matching Euclidean geometry and algebra), matching only, and in bare essence, bi-dimensional flat Boolean symbolic values with arithmetic values considered as computable functions. This is to say that, contrary to the Cartesian coordinate system, there will never be an embodiment of computable functions in nature more so than what is already embedded through algebra, even if arithmetic could possibly be a vague *formalist* or *indispensable natural-empirical* mathematical structure for the most part. This could only change if we dramatically and contrary to the Gödelian revolution of the incompleteness theorems in mathematical logic [312, 216, 160, 161], did as Gödel himself, and contradictorily as it seems, *apparently* against the demonstration of inherent limitations of every formal axiomatic system containing basic arithmetic, finished by accepting mathematical (and to a partial degree, computerized) Platonism. The exception is if computation or computable functions are algebra-to-structures pathways powerful enough to model the models, so to speak (which is affine with  $\cup$ -Mentalism) .

Mathematical Platonism is, indeed, a participating/predicative *relational* theory, as if there was simultaneously an inversion from  $\mathbb{R}$  to  $\mathbb{Z}^+ = \{1\}$  where  $\{1\}$  is

just about the sensitive and intellective *image* of the unity of oneness and many-ness. Mathematical Platonism goes, thus, from Euclidean  $\mathbb{R}^3$  to an *ideal* one  $\sqrt[1]{\frac{1}{1}}$  where  $\{1\}$  as an *image*, neither precedes nor equals reals :  $\mathbb{R} \{1\}$  as if  $1 \mapsto f(x)$   $\not\subseteq \{\mathbb{R}\}$ , and neither succeeds nor equals reals :  $\mathbb{R} \{1\}$  as if  $1 \mapsto f(x)$   $\not\supseteq \{\mathbb{R}\}$  (maybe the reason for Gödel to have thought that arithmetic in one such system as Peano's or *Principia Mathematica* was not *decisive*, somehow contrary to the computational exigence of decidability in partial recursive functions, to overall and in spite of all this, go against Mathematical Platonism).

Indeed, mathematical Platonism has this idiosyncrasy of sharing with general topology, or point-set foundational definitions, an investigation departing from geometry, capable of *metaphysically* reason according to substratal notions, such as compactness and connectedness, but, by quitting algebraic and geometrical methods, de-topologizing the world to little less (or a lot more) than a point, in *ideality*. This *ideal* de-topologizing to one-point is a striking reason to understand how monism (mathematical Platonism) and monadism (mathematical elemental Leibnizianism) are just a matter of *perspective*, even though different as they are. More in detail, it can be said that mathematical Platonism's unmindfulness towards physicality (as mathematical Leibnizianism, though less acutely), has made it neglect the intermediary focus on  $\nu\omicron\omicron\upsilon\mu\epsilon\nu\omicron\upsilon$  (noumenal) objectivity, i.e., the thing-in-itself.

But it is not to blame that mathematical Platonism is taken as a philosophy of the *substance*, as this has also to be taken into *perspective*. The formerly explained by Plato allegory of the cave (*The Republic*, 514a-520a) and the analogy of the sun (*The Republic*, 508b-509c), figures out this de-topologized one-point as nothing but the best affordable *image* for the oneness and manifolding unity of truth, holding, in essence, a more profound sense beyond spaciality, locality or  $\tau\omicron\pi\omicron\varsigma$  (*topos*). It fetches, fundamentally, beyond algebraic *structural* mathematics, beyond ordinary *topological* embodiment, and even beyond human *sensible* and *intellectual* incarnation, *visualness*, i. e., the capability of bearing sight, intuitive consciousness, as perceptual and emotional movement.

At this point, at check with necessary compactedness and connectedness, the recognizable account of a philosophy of substance and mathematical Platonism are, thus, quite contradictory. Incipiently, mathematical Platonism in  $\mathbb{Q}$  was largely

acquainted with the method of a quotient or fraction of two integers, a numerator and a non-zero denominator, with  $\mathbb{Q} = \left\{ \frac{a}{b} \mid a \in \mathbb{Z}; b \in \mathbb{Z}^* \right\}$ , hence, very similar to mathematical Leibnizianism (oneness or the monad as the primitive root of unity).

Positively, thus, is it fair to say ahead, in what concerns at its core the Turing-Church thesis and the fulcrum year of computation (1936), that "effectively calculable" [72, 362],  $\mu$ -recursive or Turing-computable functions, all represent, in terms of philosophy of science and mathematics combined in computer science, a sort of three-folded geometrical-to-(*quasi*)-topological *conceptual* hierarchy:

- First, a canonical discrete Euclidean-to-Hilbertian geometry, where reduction to primitive notions (point, line, plane, as in Hilbert's geometry, all in all resembling the "square", "tape", and the  $r$ -th bearing of the symbol in the "a-machine" [362]) and also primitive relations (betweenness, containment, and congruence in Hilbert's geometry, all in all resembling the four  $m$ -configurations "b", "c", "e", "f"<sup>52</sup>, memory and computable functions), are simple and shared philosophical axiomatization concepts, in point of fact almost a minimal *Computus Organum*, i. e., a new scientific (computerized) method.
- Second and intermediate, and departing from the Euclidean-to-Hilbertian geometry, we find the nucleus of projective geometry continuous transformations, wherein the Boolean bi-dimensional values and electric signs in traditionally linear serial computation are impelled to meet, in *simulacrum* fashion, the world of symbols in outlined diagrammatic projections, i. e., where the Boolean invariant pre-images emancipate, through instructions

---

<sup>52</sup>In this respect we can confront the sequence of four  $m$ -configurations, as depicted in the paper *On Computable Numbers, with an Application to the Entscheidungsproblem* [362] (1936) by Turing, with chapter IV of Bertrand Russell's *Introduction to Mathematical Philosophy* (1919) in what relates to Russell's efforts on defining the notion of "between", essentially the "fundamental notion of ordinary geometry" and in "ordering the points on a straight line". Assuming that this relation "between" may arrange the points of the line in the simplest order from left to right, a group of seven assumptions are needed, as to define, without a shred of doubt, (1) points between  $a$  and  $b$ ; (2) points  $x$  such that  $a$  is between  $x$  and  $b$ ; (3) points  $y$  such that  $b$  is between  $y$  and  $a$ . What is worth observing here is that if we remove the exclusively identity-disambiguation cases, and imagining the sequence to be the subsequence of printed symbols in a Turing-machine, we get exactly the same remaining four cases. This might seem standard, but it permits us to immediately test serial relations in computability, by each of the three combined properties essential to define a series: "asymmetry, transitivity, and connexity" (Russell 1919). .

and memory, a sort of transformation matrix (assembly languages) and affine transformations (programming languages). It can be said that this open symbolic projective space emanates, at heart, from  $\lambda$ -Calculus, the proper inducement of programming languages. An extraordinary remark to pay attention to is the fact that one "intensional theory of functions as rules of computation, contrasting with an extensional theory of functions as sets of ordered pairs"[10], in one such form as described in  $\lambda$ -Calculus, is, by pure *intentional* means, capable of inducing also a very powerful *extensional* perspective, as it drags, by the synthesis of imagination with auxiliary *a*-machines [362], inherent geometrical-to-(*quasi*) topological transformations.

- Third and final, we find a sense of *quasi*-topological transformations, i. e., passing the rigid transformation of translation as an invariant property that was able to build congruent and equivalent classes (Turing-machine as an Euclidean, axiomatic and synthetic, paradigm), with continuous affine and projective transformations with invariant extensions maintaining orientation in the process of combining equivalence classes (assembly languages and all level-hierarchy programming languages, a non-Euclidean geometry paradigm), lastly resounds in a *quasi*-topological space. The idea behind is that theory of programming languages and programming language semantics in *n*-dimensional abstract levels, even if always related to the bi-dimensional *ideality* of a Turing-machine (with the properties of asymmetry, transitivity, and connectedness), is seen *intuitively* to impend over, by strict connectedness and compactness, with all possible one-to-one bi-continuous permitted transformations. We might define this open (*quasi*)-topological *conceptual* level as virtuality, i.e., one such bestowal where all that pervades are non-metric spatial relationships.

But, more importantly and truthfully, this *conceptual (quasi)*-topological sense of virtuality, arisen from affine transformations, as the *acme* of programming languages semantics is, more satisfactorily instead, an *intuitive (quasi)*-corporeal sense of vitality, arisen from effective and affective transformations, with the human body as the epitomized experience. Precisely, what topology does not authorize are tearing and cutting, just what the body is averse to, and minimally, what's

is conceded, in virtuality (one *ideally* informational processing continuous space), are informatic deformations (such as stretching, bending, and twisting). In reality, the topic of programming languages and its hierarchy commanded in the human-computer interaction realm with a perspective of *ne plus ultra* past machine code, at the bottom, and *ne plus ultra* high-level programming languages, at the top, really plays an eccentricity role, in plain Kantian *cosmological* means, if compared with the modern Copernican revolution [168, 396, 158, 123, 287, 90, 88, 129, 108, 98]. Orientation [63, 202], even if no longer dependent on an equalizing point around which the heavenly bodies moved in uniform fashion as in Ptolemy, the heliocentric new system with the sun as the center of the deferents of the planets, allowing the circles of the planets with all the spheres in one system revolving more uniformly, even if diminishing the eccentricity of the sun, was all in all eccentric due to the lack of ellipses in the orbits of the planets. Likewise, programming languages are seen, like in Aristotelian and Ptolemaic cosmologies, as with one independent non-elliptic orbit each, spreading from the radius to a perfected and more rarefied perimeter, in other words, with neither the idea of a gravitational *center*, nor a common *space*. This meets the right *fulcrum* in philosophy of science confronted with programming: programming languages are strictly Copernican in essence, for the reason that even if containing the seeds of non-Euclidean geometry (with hyperbolic diagonalized, and  $n$ -dimensional elliptic features) and the seeds of projective geometry (with an invariant transformation matrix and translations, described as affine transformations), they are still a normative, rather than disruptive, paradigm.

This digression is critical to grasp how the concurrent evolution of "a-machines" [362] and programming languages semantics has produced a compacted (or compressed) and contiguously accelerated, inasmuch as the *physicality* of the integrated circuit and computer data architecture, (prolongation of the) modern age revolution in western science.

If it is astonishing that the so called Copernican revolution, still the norm of a "paradigm shift" [121, 217] due to its *cosmological* nature, was engaged by the simple geometrical *transformations* of translation and rotation with relation to the celestial bodies, following (post-Hyparchus and Arabic transmission of astronomy schools), Copernicus' *De Revolutionibus Orbium Coelestium* (1543), equally or

more astounding is to recognize that the almost mathematical-clustered Turing-machine model, by setting a simple *geometrical* translation from simple "squares" one to another, was capable of transcending a cosmological revolution.

Going to the essentials, we have in mind, Turing's *On Computable Numbers, with an Application to the Entscheidungsproblem*, 1936) [362]:

"We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions  $q_1: q_2. \dots q_I$ ; which will be called 'm-configurations'. The machine is supplied with a 'tape' (the analogue of paper) running through it, and divided into sections (called 'squares') each capable of bearing a 'symbol'. At any moment there is just one square, say the  $r$ -th, bearing the symbol  $\mathfrak{S}(r)$  which is 'in the machine'. We may call this square the 'scanned square'. The symbol on the scanned square may be called the 'scanned symbol'. The 'scanned symbol' is the only one of which the machine is, so to speak, 'directly aware'. However, by altering its m-configuration the machine can effectively remember some of the symbols which it has 'seen' (scanned) previously. The possible behavior of the machine at any moment is determined by the  $r$ -configuration  $q_n$  and the scanned symbol  $\mathfrak{S}(r)$ . This pair  $q_n, (r)$  will be called the 'configuration'." [362]

At this point, although minimally, we would like to address virtuality with a sense of vital embodiment found in *quasi*-topological transformations – amplifying mere  $\mathbb{R}^3$  *ideality* to one such sense as Merleau-Ponty's (1908-1961) phenomenological "corporeity of words and speech" [235] (*The Phenomenology of Perception* [265, 267, 268] (1945); *The Visible and the Invisible: The Intertwining—The Chiasm* [269, 266], posthumous) –, with some concepts that belong to the philosophical tradition from Henri Bergson [33, 31, 30, 29, 32] (1859-1941) one such thinker who devoted meticulous attention to George Berkeley's (1685–1753) *An Essay Towards a New Theory of Vision* [34, 35] (1709) and *Three Dialogues between Hylas and Philonous* [34] (1713) impacting on the concept of *visualness*, to Gilles Deleuze (1925-1995), author of *Cinéma 1 L'Image-Mouvement* [101, 104] (1983), and *Cinéma 2 L'image-temps* [102, 103] (1985), bringing to a new light the thesis of Bergson, mainly from *Matière et mémoire* (1896). Although we do



not have the time to expand this thesis in larger lines, it suffices to say, in a direct and intuitionistic manner, that a *perfected* schmemata to informatic virtuality is found if we make correspond in an universal Turing-machine the apperception of "m-configurations" to "frames", "tape" to "film", "squares" to "*matter and memory*" [29], the "set of images between the representation and the object" to "Time-Movement" [101, 104] and the r-th bearing of the symbol to "the Time-Image" [102, 103].

Noticeably, virtuality, in one such way, captures in sharp poignancy, neither the *naturalis* nor the *artificialis* aspects of metaphysics, but the *natura naturata* movement instead, while the hyletic and perceptual-phenomenic time-consciousness of the embodied subject with knowledge, and the human-machine interaction proper of information and cybernetics [404, 405], are not divorced, neither afflicted by false imperative duality problems, conveying at the same time programming languages and computer architecture studies with the cortex sciences through the bridge of computer vision and multiple-view geometry, in the exposed line of investigation of  $\cup$ -Mentalism.

This lines along "*la pensée et le mouvant*" [29] in informatic virtuality (passing from *tekné* to *autopoiesis* [100]) are also capable of guiding a just interpretation of natural language processing in Prolog, in the context of the delusional dream of strong AI in the "Fifth Generation Computer Project" (FGCS) [52, 139, 193], an audacious government and business venture in Japan (1982-1994), which regarded as feasible man-to-machine communication, and was truly engaged in factoring man-to-man communication in machines, wishing, therefore, to have machines *understanding* language to the Shakespearean height of "such stuff as dreams are made on" (William Shakespeare, *The Tempest*) and also to the basal line found in Merleau-Ponty's chiasm and the flesh [266, 269], with the corporeality of the simultaneous act of perceiving and the thing perceived now the plentiful machinery.

# Chapter 2

## Natural Language Processing in Prolog

### 2.1 W-grammars and Q-systems in Natural Language Processing

W-grammars [77, 39, 78, 85, 142] are, technically, (Adrian) van Wijngaarden grammars (a two-level grammar) encoding the potentially infinite universe typical of context-free grammars [67, 115] in a finite set of rules, thus belonging to the more general class of affix grammars, while Q-Systems [77] is the direct graph, formal (and programming) language used in natural language processing by choice of Alain Colmerauer at the Université de Montréal (1967-1970) and, therefore, the language formalism vault therein present in the machine translation system prototype (TAUM-73)<sup>1</sup>.

The both unavoidable and intended structure of natural languages processing is perceptible from the instance that the class of affix grammars is a meta-syntactical description of computer languages syntaxes using the regular formalisation of natural languages, and also by the fact that the van Winjgaarden grammar was an innovation in the ALGOL programming languages family [339, 400] (ALGOL 60, ALGOL 68) to the extent that (ALGOL 60 and onward) came as the first project wherein BNF (Backus-Naur form)<sup>2</sup> was ever first forged, a metalanguage notation

---

<sup>1</sup>TAUM was the acronym for the research group *Traduction Automatique à l'Université de Montréal* formed in 1965.

<sup>2</sup>John Backus and Peter Naus were involved in the conception and design of ALGOL 60,

inspired in the classical Sanskrit grammar devised originally by Pāṇini (4<sup>th</sup> century BCE) <sup>3</sup>. To this it can be added that both the linguistic boom (with roots in structuralism and the linguistics turn to the brewing incubation of post-structuralism) and strong artificial intelligence in concurrence of the 50's and 60's accelerated what can be termed as an "affix grammar inter-dialogue epoch", having enhanced one sort of linear bijection or one-to-one correspondence between the natural and computing languages sets (not coincident with the natural and artificial languages sets, neither with the different sets of natural and programming languages), always understood under the proper limitation of context-free grammars.

Indeed, it is not only through BNF (Backus-Naur form) and W-grammars (van Wijngaarden grammars) cases within ALGOL (ALGOL 60 and onward) that the series of natural language studies (formal language, grammar, linguistics and philosophy of language) has come to be manifest or come to be limited to specific illustrations, if we take in consideration the transition to computational programming languages. Thus and so, apart from BNF and W-grammars [212, 68] (the two main notation techniques for context-free grammars in the form of both one original and ancient notation of grammar studies on Sanskrit and one extant derivative metalanguage description of different forms of syntax from natural languages), it

---

both of them having been present at the ALGOL 60 meeting in Paris (January, 1960). John Backus was present in representation of USA along names such as John McCarthy (creator of Lisp), while Peter Naus represented Europe alongside van Wijngaarden himself. John Backus had, indeed, in front, been one of the mentors and designers of the first called IAL (International Algorithmic Language), alternatively named ALGOL 58, upon which the use of one metalanguage with metalinguistic *formulae* was clear.

<sup>3</sup>As a term of comparison, the culminating effort of the Vedic Vyākaraṇa (lit. "Analysis") discipline of the Aṣṭādhyāyī (lit. "Eight Chapters") summula on grammar and semantic studies by Pāṇini collected a total of 3,959 sūtras (lit. "strings"; "threads") about Sanskrit, while the W-grammar saved and assembled a "maximum of around 300 rules altogether, including meta-rules and pseudo-rules" (*W-Grammar*, Chastellier, Guy de, PROJECT DE TRADUCTION AUTOMATIQUE ; Colmerauer, Alain, DEPARTEMENT D'INFORMATIQUE, UNIVERSITÉ DE MONTRÉAL). By this stance we can evaluate the blunt and minimalist version of the W-grammar towards French, in comparison with Vedic Pāṇini's headed for Sanskrit. This 1/10 rules economy is best explained by the difference between one linguistic description defining classical Sanskrit and one natural language processing hyper-rules type of syntax interpreter using meta-rules translation processes under algebraic manipulation. Also to be noticed is the Sanskrit-restraint form in Aṣṭādhyāyī's description and the opposed unrestrained natural languages (yet French-defined and morphologically constrained) process of translation description of W-grammars, which saves a great stack of rules. This rule commands greater strings and threads economy as more and more one grammar is context-free, hence far from the constrained and defined form of one such natural language as Sanskrit, French, or any other.

shall be firstly addressed the general meta-theory on language where from such a recollection of terms and definitions sprang. One such *structuralis sententiae* urges on general acquittance, and is not dissociable from the birth of the programming language Prolog.

We can find the immediate answer to the referred *structuralis sententiae* general problem in the so called Chomsky hierarchy [67, 115] (or else called the Chomsky-Schützenberger hierarchy), the found model to the respective "affix grammar interdialogue epoch" from the Turing machine to the decades of creation of the first programming languages. The Chomsky hierarchy (1956-1963) is, basically, a self-contained type of hierarchy comprising, besides the implicit group of finite length of various symbols, four different classes of formal grammars, namely (and from outward to interior), Type-0 or unrestricted grammars, Type-1 or context-sensitive grammars, Type-2 or context-free grammars, and Type-3 or regular grammars. In this spectrum we have to take notice that the outer-circle of this self-contained rings order, going from the least to the most restrictive, places the outer-bond of the hierarchy as a range of unsettled problems (i.e., algorithms or functions that are not effectively computable).

We can also refer to the Chomsky hierarchy of different types of formal grammars (again from outward to interior) as being the collection of Type-0, recursively enumerable or generally recursive, Type-1 or context-sensitive, Type-2 or non-deterministic and, finally, Type-3 or automata formal grammars.

Prevalently, both W-grammars and BNF (Backus-Naur form) are perceiving rationalizations of Type-2 or context-free grammars under the Chomsky hierarchy [67, 115, 68], as, ultimately, Q-systems and Algol-68 [77, 339, 400] were two-level Van Wijngaarden grammars. This formalisation tailored by Noam Chomsky and Marcel-Paul Schützenberger in the late 50's corresponds, in Kantian terms, as a perceptive *judgement* related to universality and necessity in our experience of objects, to a sort of *synthetic apperception* (if not transcendental at least indispensable) argument, of all classes of formalised syntax of natural languages and the emergent field of complexity, having mechanistic linguistics [197, 142, 253] and natural language processing [39, 85] been abridged in the theoretical, historical and philosophical interpolation.

### 2.1.1 The Chomsky-Schützenberger Hierarchy

The Chomsky hierarchy [67, 115, 68] (1956-1963) quarters sufficiently, in the style of one *causata grammaticae*, the universal extension of formal grammars, in accordance with levels of both mathematical ordered and random phonological restrictions.

The Chomsky hierarchy draws in one stroke a capable *practical* and *cognitive* demonstration of the human *transcendental faculties* (mind), as well as the propositional content (language in the form of any phonological or semiological strings of symbols, states and *representations*) of any type of judgment of the entire spectrum of possible *phenomena* (world). Moreover, random naturalistic complexity of natural languages and informational computational complexity are, by means of the innateness and spontaneity of the power of *synthesis*, apprehended in one conceptual form as if it were proper the *scheme* of the *faculty of cognition* tractable under the referred hierarchy in relation with the mind, language and the world.

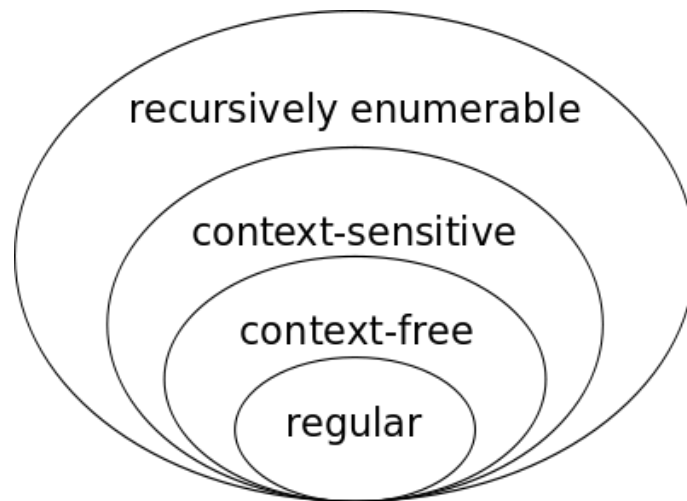


Figure 2.1: The Chomsky Hierarchy

More in depth, we are considering an originally linguistic triadic model – "Three Models for the Description of Language" [67] (1956) sketched by Noam Chomsky – that rapidly evolved to an excel form in the subsequent years in the paper "The

Algebraic Theory of Context-Free Grammars" [115] (1963) written in concurrence by Noam Chomsky and Marcel-Paul Schützenberger. The pair thoroughly worked, in this fashion, one hierarchy on certain formal properties of grammars ascendant through both linguistic (natural) languages and algebraic (enumerated) levels of restriction.

By these means, addressing grammars as one enumerating power (a "sentence-generating grammar" [67, 115] of any  $L$  corresponding to the function whose range is  $L$ ), endorsed as a collection of sentences of finite length constructed from a finite alphabet, entices a mechanism of derivation of "phrase structures" [67, 115] into more complex sequences. The kernel or *nucleus* of the Chomsky-hierarchy is a typical representation of *automata* finite-state Markov process (an enumeration of different variables, each of which, at any point in time, is fully conditional, as an if-clause representation of the present state of the system, irrespective of both the past or the future). Along the (utterly phonological but generally unrestricted symbolical) "phrase-structure" [67, 115] level sequence of restrictions, we find the outer last ring (Type-0) and limitation boundary in Turing machines [368, 363, 366, 364, 365].

In between Markov finite-state process (Type-3) and a Turing-machine (Type-0) there are two types of system representations. From unrestricted to restricted, and descending down the hierarchy of complexity we find, respectively, the context-sensitive (Type-1) and context-free (Type-2) types of grammars.<sup>4</sup> Context-free (Type-2) regards the same level as push-down type of *automata*, while context-sensitive (Type-1) is in conformity with linear-bounded typical *automata*. More in detail, we can assert to the four stages in the complexity and increasingly unrestricted hierarchy (Types- 3,2,1 and 0) at least two intermediate inner-stages comprised: deterministic context-free grammars between regular and context-free grammars (Types- 3,2), and also generally recursive grammars between context-sensitive and recursively enumerable grammars (Types- 1,0).

In this very same attainment, we shall find a gradation that expands from regular linear (Type-3) to polynomial context-free grammars (Type-2), from this to

---

<sup>4</sup>Remember that W-grammars, a member of the affix grammars class (where affixes are arguments), are Type-2 or context-free grammars under the Chomsky hierarchy.

exponential context-sensitive formalisations (Type-1) and, finally, to the undecidability upper limit [160, 161, 162, 312], as part of the transformational grammars overall process and arrangement of proper subsets. We should, in advance, recognise that if Type-0 grammars or Turing machines [362] represent the outer-finite layer of Chomsky's hierarchy, and Type-3 grammars are finite *automata*, Type-2 and Type-1 are, in turn and sequentially, intermediate representational "phrase structure" [67, 115] descriptions.

Bringing together the original paper on "Three Models for the Description of Language" [67] (Chomsky, 1956) (published in the same year that Claude Shannon, the pioneer in information theory, demonstrated the two-colours/ $n$ -state Turing Machine), and "On Certain Formal Properties of Grammars" [115] (Chomsky, 1959) we aim to resume the following inception formalization:

A grammar is a set of rules, according to which the valid phrases of one language are constructed. In formal terms, a grammar is defined by the tuple:

$$G = (\Upsilon, \Sigma, P, S)$$

- $\Upsilon$  – is a non-empty finite set, constituted by variables (nonterminal symbols). These are generally the symbols of the grammar that can be affected *in principia*, *in media res* or *in finale* by one substitution of any symbol or sequence of symbols.
- $\Sigma$  – is a non-empty finite set designed as the alphabet (set of terminal symbols). These are symbols of the grammar that can not be substituted and correspond to the valid sentences of the language.
- $P$  – is a grammatical rule defining how can nonterminal symbols be substituted, under the general form  $(X \rightarrow Y_1 Y_2 Y_3 \cdots Y_n)$

...note that this could be replaced by...

$$(X \rightarrow Y_1 \frown Y_2 \frown Y_3 \frown \cdots Y_n \frown_{n+1})$$

- it expresses, thus, the general concatenation and, in mathematical and symbolical terms, more specifically, the concatenation power of any "phrase structures" [67, 115, 68] of any strings of symbols of any alphabet. Unavoidably, this resumes to having on the left side (or *causata*) of the expression – *modus ponens* or linguistic implicature – a nonterminal symbol or syntactic variables on hold to be replaced by groups of terminal symbols (on the right side or *causa* of the expression) always in accordance with the "production rules" [67, 115, 68], whose general form should obey to a binary truth function connecting elements from two symbolic and derivation prone disjoint sets. The practical result of a free monoid should, therefore, be the language itself, under, in principle, the standard premises of abstract algebra [349].
- $S$  - corresponds to the nonterminal symbols where from starts the string derivation or "phrase structure" [67, 115, 68] (linguistic or computational) process.
- it aims now at building a general theory of linguistic structure (emancipating gramaticalness from grammar studies to formal universal grammar, and an as consistent as complete hierarchy) from observed general laws of the linguistic *corpus* (phonemes to phrases), Chomsky elaborated an English language (could have just referred natural languages instead and have pinpointed English as a subclass) "revealing"  $n^{th}$ -order statistical approximation to what he referred as a "transformational grammar" [67, 115, 68], by means of phonemic and alphabetic (rather wholly symbolical under later views) transcription of its  $*$  or  $+$  – Kleene closure of more than zero ( $*$ ) or more than one ( $+$ ) elements – into a set of strings or symbols of characters under  $V_P$  (the verb phrase structure).

$$x, y \in (\Upsilon \cup \Sigma)^* \wedge \alpha \rightarrow B \in P$$

- thus, being the undergone produced substitution operated if and only if the following provision of elements in the data logical structure is given

$$(u = x\alpha y \wedge v = xBy) \rightarrow (u \Rightarrow v)$$



It is true that, by the time of one consolidated "Algebraic Theory of Context-Free Languages" [115, 69] (N. Chomsky; M.P. Schützenberger, 1963) the concern was, explicitly, on several classes of sentence-generating devices with ties to the grammars of both natural and artificial languages, while Chomsky's attention, was, at the beginning, centered only in the English language repeated transformations tested against finite-state Markov processes. At the outset of the incipiently drawn hierarchy, the thesis submitted (and theoretically proven) was that the construction of a finite-state Markov processment of transition symbols could never include an English (or any natural language) grammar. Suitably, we should expect total failure in generating, from finite-state Markov processes as well as through transformational and generated phrases by infinite sentencing, one such result as reasonable (false or truthful) sentences, or any acceptable customary philosophical inquiries and the like (whichever valid syllogism deduction in the form of terminal symbols of whichever either formal or natural language).

Radically, what was also vindicated, in a sort of logical antinomy materialised in the landmark paper, was that uncountable many languages are, literally, not describable and can not be either by any general linguist metatheory on any form of symbolical natural languages such as English. The postliminary question was to know if there were possible discretional and elective symbolical languages outside the range of description.

Typically, a language generated by one grammar  $L(G)$  corresponds to the derivational result of all its different states from the initial nonterminal state  $S$ , thus producing potentially infinite sequence of symbols of the alphabet  $(\Sigma)$ .

$L(G) = (v \in \Sigma^* : S \Rightarrow^* v)$  is, therefore, the custom result of potentially finding how an arbitrary element can be generated by another of the same nature in one step only ( $v \Rightarrow v$ ) no matter how many times convoked ( $v \Rightarrow^* v$ ), into a set of finite-length strings or sets of symbols or characters, usually under more than one step of the transformational grammar ( $v \Rightarrow^+ v$ ).

If we admit that Aristotle's famous definition of the universal (and *definiendum*) "humankind", in which man is the sole participant – bearer of thought and speech, with the principle ( $\lambda\acute{o}\gamma\omicron\nu \ \epsilon\tilde{\iota}\chi\omicron\nu$ ), reckoned in the Latinized, more abstract and diffused expression *animal rationale* – instantiates speech, language and symbolic expression through special and specific utterances of human communication

with the unavoidable structure of vocal sentencing words (the symbolical power of free concatenation of *mental* and *ideal* sentencing words of the *universal AB* alphabet ( $A \frown^{+B}$ )), we are welcoming indisputably the Chomsky-Schützenberger hierarchy, and, therefrom, accepting some *dialecticae et logica explicanda*.

It is impossible to find, according to Aristotle's definition of man, a void grammar  $G_0$ , wherein one language would assume the state of one empty string  $\{\epsilon\}$  or  $\emptyset$ . This stance strongly suggests, over Aristotle's antropological definition or indeed Plato's – "Man is the plume-less genus of bipeds" (Plato, *Politikus*, 266) – the neo-Kantian author Ernst Cassirer's definition of man as, intrinsically, one "*animal symbolicum*" [61](Cassirer, *An Essay on Man*, 1944). So too, we shall find one algebraic form of the general (*ideal* or *mental*, utterly phonological and general symbolical) power of infinite concatenation ( $v \Rightarrow^+ v \cdots \infty$ ), if accepted the applicability of the Chomsky hierarchy to the universe of all languages (natural and artificial, stochastic and programming), whereupon we should investigate what are the *critical* limits of language. More is to be investigated as to why, inasmuch as a finite Markovian source is a too limiting grammar (Type-3), why a recursively enumerable arrangement of sets and grammar (Type-0) is also inappropriate to describe the typical undecidability, randomness and unpredictability of any natural language. This sets, thus, the question of the more general stand of natural language processing and human-computer interaction in AI (from "transformational grammars" [67, 115, 68] conceptual ontologies under Chomsky's view on computational linguistics to machine translation and the statistical approach [276, 122, 39]), to the distinguished particular case of natural language processing in Prolog [373, 85, 142] (parsing or syntactical analysis from finite-state automata to feature-based grammars). The paramount question under the *continuum* paradigm and the algebraic *substratus* is to know whether the successive containment model of the Chomsky hierarchy overall replicates the (Type-3) finite-Markov states to Turing computability (Type-0) background, against the  $n^{th}$ -statistical,  $n^{th}$ -probabilistic and real-valued queries subsumed in the general evolution of natural language processing, as if the general symbolic (linguistic and computational) *corpora* was to be encapsulated in one such algebraic form, and the Chomsky hierarchy, apart from universally valid, is, under human-computer

interaction [404, 69] (and both naturalistic and computational complexity), evolutionary correct.

Is there, thus, one sort of *mathematicae et mathesis universalis explanans* in Leibnizian style? Does this mean that the correctness (or even quasi-correctness) of Chomsky's hierarchical model and the appropriateness of such anthropological definition captivates the contemporary problems and open questions in philosophy of mathematics in the fields of philosophy of language, information complexity and generally AI, instantiating a sort of informational and computational "anew" antinomies in between Leibniz's *caractheristica universalis* and Gödel's numbering in the incompleteness theorems, convoking a debate about transcendental idealism and its expressiveness, the critical limit of judgments and experience (the mind, language and the world *in omnia relata*)? One such view would render, thus, in perspective, a debate, not only about semiotics and philosophy of language, but also its framing on the cosmological idea of totality of composition and division of phenomena in the universe through transcendental mathematical ideas in accord with the possibility of freedom in harmony with the universal law of natural necessity, along the philosophies of Leibniz, Kant and Gödel.

At hand with these remarks on the broader context, we can proceed to further investigate the nature of the Chomsky-Schützenberger hierarchy, by way of challenging a drawn parallel with the foundation of Saussurean [99] linguistics.

In order to achieve a comprehensive survey of the *W*-grammars and *Q*-systems [77] in natural language processing work transacted by Alain Colmerauer in Prolog, it is inevitable to retrace and retell, in *backtracking* mode<sup>5</sup>, to the precedent Sanskrit-inspired linguistics paradigm of Saussure [99, 212, 123, 278, 276], without which it would be virtually impossible to apprehend the context-free grammar (Type-2) BNF (Backus-Naur form), the description of formal and abstract systems behind Algol 60 [339, 400], neither would it be possible to capture *W*-grammars (the only other notation technique for context-free (Type-2) grammars, the up-front formalism for Algol 68 [49]. Complementing this historical, computational

---

<sup>5</sup>Reference to the programming and computational notion of "backtracking", the algorithm general procedure for constraint satisfaction search problems, coined by Derrick Henry Lehmer in the 1950's, a built-in facility since string-processing languages like SNOBOL (1962-1967), and yet in Prolog itself as its chief tree-structure search feature.

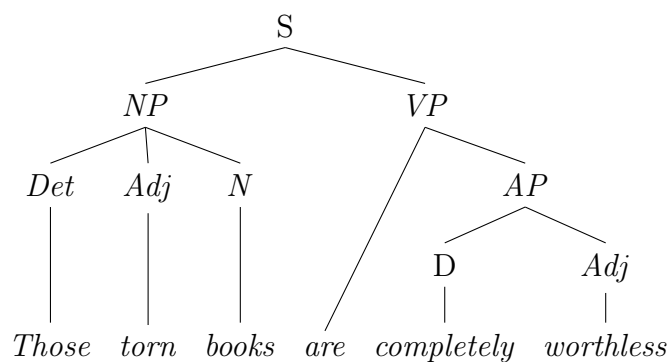
and philosophical digression, it is also ineluctable to convene the open questions in the debate of the philosophies of computation and information, language and mind, mathematics, science and technology that can best access the proposed aim.

### 2.1.1.1 From Saussure's Linguistics to the Algebraic Theory of Context-Free Languages

As we shall start with a demonstrative expression, we choose to use the same example appointed by Noam Chomsky and Paul Schützenberger in [115] "The Algebraic Theory of Context-Free Languages":

*Those torn books are completeley worthless.*

This expression corresponds to the one such diagram in which the whole string or sentence can be seen decomposed in its essential building blocks. The general philosophical and symbolical view obeys to observing the structural description of the pinpointed sentence in all its subdivisions and various categories (a determiner, an adjective, a noun, all building up a noun phrase, bracketed with a verb phrase, constituted by one adjective phrase, with one adverb and one adjective), being the whole string the one sentence under analysis.



In this way we have at display an alternative configuration

[Sentence [Noun Phrase [Determiner^Those [Adjective^torn] [Noun^books] ]  
[Verb Phrase^are] [Adjective Phrase [Adverb^completeley] [Adjective^worthless]]] .

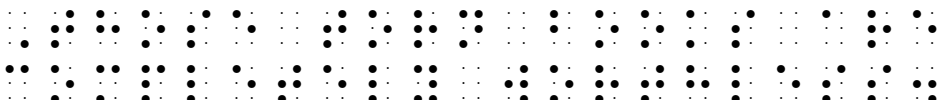
which is also capable of being replicated in the phonological form [ðəʊz θɔ:n bʊks a: 'wɜ:θɪŋs] in IPA<sup>6</sup> terminology, by putting forward, in possible executable terms, the *ideal* of fixing a universal phonetic alphabet, by defining the class of possible strings, which could, furthermore, correspond to the following idealized form under the concatenation proper of the algebraic apparatus: [ðəʊz θɔ:n bʊks a: 'wɜ:θɪŋs]. Indeed, under pure algebraic terms, responsive to the *continuum explanans* and remembering what Chomsky referred as the possibility of the infinite lengthening – *addendum ad infinitum* – of generative grammar phrasing, one other representational equivalent, recurring to the Kleene operator (\* or +) would be: [(⇒\*)ð(⇒\*)ə(⇒\*)ʊ(⇒\*)z(⇒\*)θ(⇒\*) . . . . . (⇒\*)ɪ(⇒\*)ɪ(⇒\*)s(⇒\*)].

Any of the following sentences could as well be resumed under the preceding formal symbolic expressions, *in potentia continuum*, from the very same original phrase 1), followed by any of its "transformations" [67, 115, 68, 69] 2), 3), 4), 5) and 6), which correspond to the still phonetic, but more *ideal* and less rhapsodized form of 1):

1. *Those torn books are completely worthless.*
2. 01010100 01101000 01101111 01110011 01100101 00100000 01110100 01101111  
 01110010 01101110 00100000 01100010 01101111 01101111 01101011 01110011  
 00100000 01100001 01110010 01100101 00100000 01100011 01101111 01101101  
 01110000 01101100 01100101 01110100 01100101 01101100 01111001 00100000  
 01110111 01101111 01110010 01110100 01101000 01101100 01100101 01110011  
 01110011 00101110

---

<sup>6</sup>IPA (International Phonetic Alphabet since 1888, with its most recent change in 2005) is the most diffused of the intended standardized representations of the sounds of oral language, markedly Westernized and primarily devised on the original Latin alphabet. It is composed of 107 letters, 52 diacritics and 4 prosody marks (figures upon which it is possible to perceive, in discrete and atomistic manner, the elementary and combinatorial reductionism in the form of speech segments of the discipline of sounds of speech analysis). IPA itself, like the Chomsky-hierarchy, can be said to accommodate, in linear progressive fashion, the entire range of sequential (material to ideal), (physical to abstract) segmented phonotypic elements. These are perceptible from phones to phonemes (and superfluent intonation) or, alternatively, as depicted in the official IPA 2015 chart, from consonants (pulmonic to non-pulmonic) to vowels, with modifying diacritics and suprasegmental qualities such as length, tone, stress, and intonation.

3. 

4. Master Key with Encryption, Hash and Compression Algorithms from GoAnywhere OpenPGP Studio Software 1.0.1 Linoma Software:

Passphrase: Prolog

ASCII Armored:

-----BEGIN PGP MESSAGE-----

Version: BCPG v1.48

hQE0A4sRGZS1Lk0zEAP/csRnj+6zX5Xk3sXAXGLIPSlultneunQmUPTi4lRN6oOZ  
lIGpXgAfP3tfjasaFiwdNzvuTuN8ndU9DdEkajI41hkgf6DMLxvOH4gcgS5tS74a  
fZGW1bLKJOHjef8jjVdY+/Yj4/ZNeypzQ0Ve412+VSB+oeYcMXKGFtClJVv2q2UD  
/iWN1UE05HArun/2uuDfoKQ1T6/Nd5sTZxE0pWl9/4yrOp+rJfXSDcqoATbjLBM4  
lb+ZBsaQarVu11cqeoZV09S8jmA11h4qhi+trpqooAlB+4MEA8ohnHyXBKxRNN1G  
FSjtg1bASvZkV02u7UsEKLL7MtHJnI9Kn4diZpRRNMlp0msBASo1/B7u1r5J8FWn  
mS1UZcQU6Bb77871PK/JIQB+VjaqKrs4GkJ1LBy73HqMy8DHx+z0ZwlLoSgdIppL  
DgzC4v3PAAtBvEMWWLJ+Ap5ypXBCjcx0WcIrol4hsiNRlrmz/8I174gTVJvaENG==  
=4P2Q

-----END PGP MESSAGE-----

-----BEGIN PGP PUBLIC KEY BLOCK-----

Version: BCPG v1.48

mQGibFnSDcARBADRQztFDTN7HxCeQzkVr5QSt1Zn+w0m1Q/GYyRRBupxU1bbMHLy  
lzKBGdYyU4G+P4HmhB0MfGeXaqmpIpnfaAijqP9sxjY+pYi4255WouD3it2ECtwz

bev40yztzKgJD2wVuuoNMKVNbdBA1X1Lfivsn9nzD6WLFuL6esa6LSWPqqwCghIj6  
iNbcjL9doaaajnmOAL7a0gMEAMnrvu9J5IbXLSlgo+wzn90pOR0QjmqFf+wtT6iH  
bB5XIrOxFfYNYEDjIfbh3NGXtAm77y77J++W1hygvYzAfiAU0zUU+y0kpkLhMAAy  
KniSYo2rE38N4AE1ziVdwqaZ27X4LxGgwuX0gnDCNf8m60CrFB+TzsLD12srx9pN  
jKMjBAClrQ//abpNuc0h01ax8ItVNvLUQyMICZQEQRGG0VNCbnmRkD6aZdDuN29u  
XwNZkI004aK4m61QiMbZrvI9SRrRxXskWWTGgVXPDDHVv1V61TmV9y1424+ZoH/n  
xQ04J04jduVr/YBCNEOwsueF+oZebKP92WsTbakDvvQPjqqUKbQHYXMgPGFzPohk  
BBMRAGAkBQJZ0g3ABQmce7kQCgsHCAkEAwYFAgoIFQUBAwIICQoDFgECAAoJEGLM  
VfaS30mZ0H0AnisYLZSiUXukRTJJYvCY3Igm4QMHAJ9LLhZ4VHvmLWP03fzzfCa/  
5MOXBrkBjARZ0g3AEAQAohTYvvXV3joTIk3WFIxE7b7TZZ17frdj23As0Jv80SM  
Ngcitj0TIu7CrbyM08mP3g3KrfbJBwX/8106i65iIghVKfwLe0aUxKd83tp0cB0+  
8CIEuo4w7RnfMYyfGqKexS31ND7T/LvQe/cWtQTAVu9TF7dXGU30bquQR7FuXUsD  
/j3PcVyKfOAts0bQT8bVTJM70uY+FAU/Dx4CRb6RHNdQTJLESuY7kKnI2NX0Ieq0  
Ac7tUgVmz+0v2r8PIEMDdrNnZBK8ClAd0oK7V2tUZJLqHfMfGLziRYW0I587JHd7  
/vFtR1BlrHebZ18UiCtCMDWOWJLhJcdf03zsba3H1+JwA/9wkGawIAh7M5TE99fW  
rinn45YEh2gkr8QDBfCKVTvlGpMhIFF63NjW3orN7j+IX39KZROT15s4bjWwZvTW  
m0VWMy9eBGEN1YYAVF+5y0Gdb9ijRMBp2I4z01J35nwHaf0Fzjmmcq7SVFARWasm  
9T1H+EVNz8LYORSdnt+HbDkYIohkBBgRAGAkBQJZ0g3BBQmce7kQCgsHCAkEAwYF  
AgoIFQUBAwIICQoDFgECAAoJEGLMVfaS30mZHtIAN0432PyyoSLJAWZVVEWZfjc8  
VlK4AJ9AMXuY0onLJ+IGRc6K7E1cFEXrAw==  
=xtQs

-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP PRIVATE KEY BLOCK-----

Version: BCPG v1.48

lQHXBfnSDcARBADRQztFDTN7HxCEqzkVr5QSt1Zn+w0m1Q/GYyRRBupxU1bbMHLy  
lzKBGdYyU4G+P4HmhB0mFGeXaompIpnfaAijqP9sxjY+pYi4255WouD3it2ECtwz  
bev40yztzKgJD2wVuuoNMKVNbdBA1X1Lfivsn9nzD6WLFuL6esa6LSWPqqwCghIj6  
iNbcjL9doaaajnmOAL7a0gMEAMnrvu9J5IbXLSlgo+wzn90pOR0QjmqFf+wtT6iH

bB5XIrOxFfYNYEDjIfbh3NGXtAm77y77J++W1hygvvZAfiAU0zUU+y0kpkLhMAAy  
KniSYo2rE38N4AE1ziVdwqaZ27X4LxGgwuX0gnDCNf8m60CrFB+TzsLD12sr9pN  
jKMjBAClrQ//abpNuc0h01ax8ItVNvLUQyMICZQEQRGGOVNCbnmRkD6aZdDuN29u  
XwNZkI004aK4m61QiMbZrvI9SRrRxXskWWTGgVXPDDHVv1V61TmV9y1424+ZoH/n  
xQ04J04jduVr/YBCNE0wsueF+oZebKP92WsTbakDvVQPjqqUKf8JAwJHdzXQslf+  
TmBtFzdptB9rbgCzj4mYgnkoEBGkSLg21yspblrRKHnIpDSDCgNjnd+stAdhcyA8  
YXM+iGQEEExECACQFAInSDcAFCZx7uRAKCwcICQQDBgUCCggVBQEDAggJCgMWAQIA  
CgkQYsxV9pLc6ZnQfQCeKxgtlKJRe6RFMkli8JjciCbhAwcAn0suFnhUe+YtY/Td  
/PN8Jr/kw5cGnQItBFnSDcAQBACiFni+9dXe0hMiTdYUjETvtNlnXt+t2PbcCzQ  
m/w5Ix42ByK2PRMi7sKtvIzTyY/eDcqt9skHBf/yU7qLrmIiCFUp/At7RpTEp3ze  
2nRwE77wIgs6jjDtGd8xjJ8aop7FLeU0PtP8u9B79xa1BMBW71MXt1cZTc5uq5BH  
sW5dSwP+Pc9xXIOWgC2w5tBPxtVMkzvS5j4UBT8PHgJFvpEc11BMksRK5juQqcjY  
1c4h6o4Bzu1SBWbP7S/avw8gQwN2s2dkErwKUB3SgrtXa1Rkkuod8x8Yv0JFhbQj  
nzskd3v+8W1HUGWsd5tmXxSIK1wwNbRYkuElx187f0xtrcfX4nAD/3CQZrAgCHsz  
lMT319auKefjlgSHaCSvxAMF8IpVNWWA+aEgUXrc2Nbeis3uP4hff0plE5PXmzhu  
NbBm9NabRVYxj14EYQ2VhgBUX7nI4Z1v2KNEwGnYjjM7UnfmfAdp+gX00aZyrtJU  
UCtZqyb1PUf4RU3Pwtg5FIOe34dsORgi/wkDAkd3NdCyV/50YBy11Dxg2YP4Swj8  
6FvXAfAZNX42WQKiHPmW12TLWMJAIfTQv2sbJ34NUztS/qeCLcR8NL+8nMLm4xn8  
zw015aNDUnTU11/Szv7fq9CNVf0TduWJMuyrm+brXepS1CBUlfltr0sX6baICisp  
FZ7Sf3cPmMNDbbmT0aCC55JftNoqxhwUXpm0/b5ewHA657hyw1IogV+IZAQYEQIA  
JAUCWdINwQUJnHu5EAoLBwgJBAMGBQIKCBUFAQMCCakKaxYBAGAKCRBizFX2ktzp  
mR7SAJ90N9j8sqEiyQFmVVRfMx43PFZSuACfQDF7mNKJyyfiBkXOixNXBRF6wOZ  
AaIEWdINwBEEANFD0OUNM3sfEISrORWv1BK3Vmf7DSbVD8ZjJFEG6nFTVtswcvKX  
MoEZ1jJTgb4/geaEE6YUZ5dqqakimd9oCK0o/2zGNj6liLjbnlai4PeK3YQK3DNt  
6/g7K3MqAkPbBW66g0wpU1tOEDVfUt+K+yf2fMPpYsW4vp6xrotJY+qrAKCEiPqI  
1udyMv12hqp0eY4AvtrSAwQAyeu+70nkhctKWCj7D0f06k5HRC0aoV/7C1PqIds  
Hlcis7EV9g3IQ0Mh9uHc0Ze0CbvvLvsn75bWHKC/JkB+IBTTNRT7LSSmQuEwADIq  
eJJijasTfw3gATX0JV3CpnbtfvgEaDC5c6CcMI1/ybo4Kt8H5P0wsPXayvH2k2M  
oyMEAKWtD/9puk25zSE6VrHwi1U28tRDIwgJlARCsYY5U0JueZGQPpp10043b25f  
A1mQjTThoribrVCIxtlG8j1JGtHFeyRZZMaBVc8MMdW+VXrVOZX3LXjbj5mgf+ff  
A7gnTiN25Wv9gEIOttCy54X6h15so/3ZaxNtqQNVVA+OqNqptAdhcyA8YXM+iGQE  
ExECACQFAInSDcAFCZx7uRAKCwcICQQDBgUCCggVBQEDAggJCgMWAQIACgkQYsxV  
9pLc6ZnQfQCeKxgtlKJRe6RFMkli8JjciCbhAwcAn0suFnhUe+YtY/Td/PN8Jr/k



w5cGuQGMBFnSDcAQBACiFni+9dXe0hMiTdYUjETvtNlnXt+t2PbcCzQm/w5Ix42  
 ByK2PRMi7sKtvIzTyY/eDcqt9skHBf/yU7qLrmIiCFUp/At7RpTEp3ze2nRwE77w  
 IgS6jjDtGd8xjJ8aop7FLeUOPtP8u9B79xa1BMBW71MXt1cZTc5uq5BHsW5dSwP+  
 Pc9xXIOWgC2w5tBPxtVMkzvS5j4UBT8PHgJFvpEc11BMksRK5juQqcjY1c4h6o4B  
 zu1SBWbP7S/avw8gQwN2s2dkErwKUB3SgrtXa1Rkkuod8x8Yv0JFhbQjnzskd3v+  
 8W1HUGWsd5tmXxSIK1wwNbRYkuElx187f0xtrcfX4nAD/3CQZrAgCHszlMT319au  
 KefjlgSHaCSvxAMF8IpVNWWA+aEgUXrc2Nbeis3uP4hff0pLE5PXmzhuNbBm9Nab  
 RVYxj14EYQ2VhgBUX7nI4Z1v2KNEwGnYjjM7UnfmfAdp+gX00aZyrtJUUCtZqyb1  
 PUf4RU3Pwtg5FIOe34dsORgiiQEGBECACQFAlnSDcEFCZx7uRAKCwcICQQDBgUC  
 CggVBQEDAggJCGMWAQIACgkQYsxV9pLc6Zke0gCfTjfy/LKhIskBZ1VURZ1+NzxW  
 UrgAn0Axe5jSicsn4gZFzorsTVwUResD  
 =WJzI

-----END PGP PRIVATE KEY BLOCK-----

5. *If I am not wrong, those torn books are completely worthless and moth-eaten.*
6. *Yes, you are right, those torn dusty books and magazines are completely worthless and forgotten.*

In point of fact, the entire concatenation formal power series of the sounds of human language ( $\# \curvearrowright^+ \#$ ), *ideally* not just the past, present and future of all human languages and inherent derivability, but instead the universe of all the possible combinations of the sounds of speech, under which sentences (through either syntax parsing or speech recognition under natural language processing) are microscopic atomic parts, could well be characterized by the  $Aum̐$  (or else know as  $Om̐$ ) (transliterated from  $Nāgarī$ ) Sanskrit expression.  $Aum̐$  or  $Om̐$ , as a distinct syllable sound and vocalization, is supposed to attest for the ultimate reality and cosmic principle, as if it was a sort of phonological  $\Lambda$ , the well-known cosmological constant in physics. Truly, even if it is correct to place the phenomenon of language under the strict limit of the sounds of speech, and one such form is advisably tractable under the articulatory and vocal tract apparatus proper of man, it is also true that the mental aspect permits the envisagement of one such symbolic description notably ascending from the (yet fully perceptive idea) of *pure*

mathematical sound in physics to, perhaps more importantly, (yet fully *a priori* determined) wholly phenomenological music.

### 2.1.1.2 Musical Interlude

Most notably, in all the course of the Western philosophy of language, and since the invention of the first modern music theory by Aristoxenus of Tarentum (IV<sup>th</sup> century BCE), roughly two generations after the writing of *Cratylus* by Plato – the first recognisable book in philosophy of language discerning on naturalism *vs.* conventionalism, and also arbitrariness *vs.* fixed theory of forms over the right appropriateness of signs and the different subjects of naming –, it was almost only incidentally that music impregnated the disputable theories on language.

We are invoking the decisive contributions made by the two Late Antiquity to Medieval Era philosophers, Augustine of Hippo and the Roman senator Boethius.

Augustine of Hippo (354-430) expanded in *De Musica* [14] upon the theory on metrics and rhythmic allied with ethics, whose *opera*'s confluence with language would be minor, if did not also include the determinant concurrence of *De Magistro* [9], a seminal undertaking in philosophy of language. In the work *De Magistro* [9], Augustine was able to reason on a whole range of unprecedentedly schematized concepts in philosophy of language. With a stance in the discipline sufficiently neutral in relation with *Doctrina Christiana* [13], the Christian theologian and philosopher was able to upsurge philosophy of language and semiotics (or else said sign theory), from grammar studies typical of the liberal arts (grammar, logic and rhetoric, later know as *trivium*; arithmetic, geometry, music and astronomy, later known as *quadrivium*).

In such way, concentrating on the "phrasing structure" [67, 115, 69] of *pars orationis*, *nomen*, *uerbum* and different *coniunctiones*, departing from the atomic parts of speech (*si*, *uel*, *nam*, *namque*, *nisi*, *ergo*, *quoniam*, *et*, *que*, *at*, *atque*) attaining also to the pragmatics analysis of *loquendi regula* (speech acts rules), Augustine, by having given attention to the reflexive judgement of the *homo interior*, explained, thus, in full form, the prime motive of *De Magistro*'s *docere et commemorare* (to teach and recall). Under Augustine's analysis it is of paramount importance the overall doctrine of *cognitio signi* (semiotics), to ascend (in formal agreement with Plato's doctrine of ideas) to *cognitio rei*, the acknowledging truth

as it would be. The whole range distinction between *signa* (signs), *significatio* (meaning) and *significabilia* (visual or acoustic image), from the traits of *dicibile* (what is and can be said), definitely asserted new values to the field, in a triptych embracement of *intellectus, ratio et mens*. By this effect, Augustine achieved having non problematically combined metaphysical decidability with language conventionalism.

The product of Augustine's *De Magistro* [9] was, therefore, not only to have reinvigorated the Stoic's tradition from onomatopoeic naturalism to anticipated nominalism and even Saussure's [99] linguistic arbitrariness, but also to have surpassed the already meaningful Stoic model of *φωνή* (*phone*), *λέξις* (*lexis*), *λεκτόν* (*lekton*) and *λόγος* (*logos*). It is clear the *liaison* and influence exerted on Saussure's linguistics, while it is fair to admit that are really pale improvements both John Stuart Mill's distinction between connotation and denotation, as well as Frege's [234, 53] logic model of sense (*sinn*) and reference (*bedeutung*) in philosophy of mathematics (as for the rest, a bivalence that could be original if Anthistenes and Cleanthes had not been credited in Antiquity for the exact same distinction), compared in such spectrum the rich legacy of Ancient philosophy, passed more than 1500 years from its decline.

In what concerns Boethius (480-524), the author expressed in *De Institutione Musica* [42] a threefold vision of philosophy of music with an universal and holistic scope. By order, Boethius exposed the concepts of *musica mundana, musica humana et musica instrumentis*<sup>7</sup>:

“*Sunt autem tria. Et prima quidem mundana est, secunda vero humana, tertia, quae in quibusdam constituta est instrumentis.*” [42] (Boethius, *De Institutione Musica*, Librum I, § II, Tres esse musicas; in quo de vi musicae).

---

<sup>7</sup>At the time Boethius wrote *De Institutione Musica* he never referred to *musica instrumentalis* instead of *musica instrumentis*. This shift of word ending might, nevertheless, be allowed, as it was an epoch assumption that vocal music was part of instrumental music. This ambivalence is crucial to grasp the place of not only vocal music, but also human voice and speech in Boethius's philosophy of language and music, as it might mean that the venerated martyr safeguarded vocal music from instrumental music. In contrary to this stance, it is a very strong argument to posit why, thus, in face of such a great prominence of vocal music, never was it made explicit, adducing a great case of evidence if, indeed, we know that the proper term *instrumentis* eventually evolved to *instrumentalis* from Boethius' time to Thomas Aquinas' XIII<sup>th</sup> century.

More in detail, it encompassed, laid down as in categorical formulae, *musica mundana* (correspondent with *musica universalis* and the originally Pythagorean idea of music of the spheres), *musica instrumentalis* (in which division the "last of the Romans" presumably included *vox humana* or vocal music), and, finally, *musica humana*, an exquisitely crafted concept of humanism, illusive and pervasive of all the different entrenched divisions of the liberal arts and very pertinent to the new to come cultural and epistemological turmoils in Western thought. Plausibly, *musica humana* is best described as a sort of special antechamber between the inner silence and the verb, as well as between *silentium interior* and *musica universalis*, by means of the same locus of breath as used in one of the forms of *musica instrumentis*, that is, *vox humana* (the human voice). Maybe the just referred Aum̐ (or else know as Om̐) (transliterated from Nāgarī) Sanskrit expression, a sort of universal constant and human respiration whisper can do justice and be, indeed, a parallel cultural concept to Boethius' *musica humana*.

We should strive in deepening our understanding of philosophy of music (constituted as it is, echoing Boethius' metaphor, as a sort of antechamber of philosophy of language). The idea here subscribed is contrary to the idea of philosophy of music as a late epiphenomenon coming from emancipated and sophisticated philosophy of language or even set apart philosophical points of convergence, but matches instead the thesis defending philosophy of language and linguistics subordinate to philosophy of music.

*Prima faciae* it is the concept of *musica humana* the mainspring concept, all too forgotten. *Bréf*, we are adopting a wider view of philosophy of (perceived) sound in (intuitive, interior and perceptual) time, and also of spatial forms, capable of accounting for the full eidetic-tonemic-phonetic circle. One such circle is best described as a constrained aerophone, under which the Chomsky-hierarchy [67, 115, 69, 68] can be considered a proper formal substructure (on the premise that encompasses *sonus universalis*, from which music and language are derivative orders) and where natural language processing is a sub-problem [39, 85, 142]. It is obvious that there are not any languages endowed with higher-level musical features, such as harmony (indeed, neither did have Ancient Greek language which sought for harmonically crafted effects by means of concord of sounds), but it is clear that low-level musical features, such as the ubiquitous range of tones,

rhythm, meter, articulation, tempo and texture, and even common accidentals and breath marks, are passable to non-musical phonemic orders of any language and its (phonetic or ideographic) alphabets [18, 394, 285].

Extended notes on philosophy of music are, thus, welcomed, both in general philosophical and historical terms, as under the triplicate division of Boethius [42], to the aim of understanding natural language processing in the transfer from linguistics to formal grammar studies, from semiotics to computation [199, 53, 350].

In particular, it shall be unveiled the Chomsky–Schützenberger hierarchy [69] *in inceptum finis est* status, as we are to discover, plainly, that inasmuch as the cutting edge and present-day sense of one generative universal grammar is the generative tonal (musical) grammar [238] (*A Generative Theory of Tonal Music*, GTTM, 1983-2009), in principle, something as the general presumptions of natural language processing in any programming language [39], and one such as Prolog [85, 142, 78] (and throughout the spectrum of failure of the FGCS [52, 139, 86, 141, 324] to comply with self-proffered AI [352] prognosis), can not be discerned without grasping the leading fundamentals of philosophy of music (ultimate phenomenological *unrestricted* and utmost *structured* sound).

The generative theory of tonal music [238] not only represents the vanguard of the latest improvement efforts in the Chomskyan theory [67, 115] of the transformational and universal grammar, as in tectonic and architectonic terms, is supposed to expand from the principles of natural language processing, taking into account a full eidetic-tonemic-phonetic circle. This should illustrate the prevalence of philosophy of music over philosophy of language and the articulation of natural language processing into a wider frame of perceived sounds in perceptual time or flux of consciousness (even if settled beyond regular expressions and automata, morphology, finite-state transducers and computational phonology, and attaining exclusively to parsing with context-free grammars with unification) [199]. The same should be valid, in our understanding, in the body of the theory of programming languages [313, 377, 53, 173]. We are, thus, pointing out the need of universal horizons and the reexamination of the first principles.

In one way or another, it resembles that the *quadrivium* – arithmetic (static discrete quantity), geometry (stationary magnitude), astronomy (dynamic magnitude) and music (discrete quantity in movement) –, in the futurity of the passage

from Modern to Contemporary Age, were to be unprivileged of their proper doxastic and belief logic, the reason being that Turing's computing limit [363, 366, 364, 365] and Gödel's incompleteness theorems [161, 162, 160] in mathematical logic containing arithmetic, Gauss' and Bolyai-Lobachevskian's discovery of non-Euclidean geometry [174, 349, 408], Einstein's relativity and the quantum revolution in physical astronomy [124], have gone-ahead to leave music as the only left realm of indisputable acquiescence of one such accepted status as one of an universal language. Music is, therefore, the only remaining symbolical science of indisputable universal status, in pristine state and unmoved aura [394, 285]. Alongside with this observations, it can be said to enjoy a free *continuum* of its own, an *acousmatic* more than acoustics, intuitive monochord. The concept of *mousike techne* [18, 244, 330, 166] in language stands out and is best described, then, by saying that it resounds a sharp ( $\mu\epsilon\lambda\omicron\varsigma - \epsilon\pi\omicron\varsigma$ ) (melos-epos) or (musical-linguistic) metronome, with a concept of time neither extensible, nor abridgeable, enshrining also a mingled aerophone and chordophone in *musica humana* [42] constitution. Originally, the paramount feature in regard to Ancient Greek *mousike techne* in language, was the multi-dimensional (lyrical; dramatic recitative; musical; geometrical and mathematical) sort of extant formal Pythagorean transcendental type of natural judgement, with a preeminent acousmatic nature<sup>8</sup>.

At this point we can fully expose the necessary reasons for a **Musical Interlude** segment, clear of the as of now disclosed and assumed principle of philosophy of language subsumption in philosophy of music. We choose to separate the reasons in order below:

- the historical different frameworks of the original Chomskyan [67, 115, 69, 68] linguistic theory of the transformational or generative grammar, mainly propounding the core thesis of universal grammar innateness behind Saussure's [99] historical structuralism, encompassing the standard theory (1957-1965),

---

<sup>8</sup>Not to forget that *ακουσματικοί* (*akousmatikoi*) used to refer to the probationary pupils of Pythagoras required to sit in absolute silence while listening to their teacher delivering the lesson from behind a veil or screen, stands out in the context of *mousike techne*, in Archaic to Ancient Greek language, as a superlative philosophy of language { $\nu\omicron\upsilon\varsigma$ } (*nous-thetic*) consciousness, obviously under the common norm of *ηθος* (*ethics*).

the extended standard theory (1965-1973), the (by the time of birth of Prolog) revised extended standard theory (1973-1976), both the relational and government binding principles and parameters theory (1975-1990) and, finally, the minimalist program (1990 to the present), has led, from the beginning of the 1980's to the very present (permeated in the context of context-free grammars, linguistics post-structuralism and natural language processing [142, 39]), to a series of studies on music theory and analysis that can better be summed up in the generative theory of tonal music [238] (GTTM, 1983-2009). Following Leonard Bernstein's insightful recommendations in the *The Unanswered Question* series in the Charles Eliot Norton Lectures at Harvard University (1973), idealistically advocating the possibility of a musical grammar, in terms comparable to the transformational or generative grammar of Noam Chomsky, the sequent groundwork effected from the 1980's on by the American composer and music theorist Fred Lerdahl (n. 1943) and the American linguist Ray Jackendoff (n. 1945), has, in the course of more than three decades of investigation, definitely marked a signature era to supplant the evolution of the different frameworks of the Chomskyan generative grammar. In the aftermath of Schenkerian analysis [27, 37] (natural language processing equivalent in music analysis), this era, very plainly, is to be called the Generative Tonal Music Program (1980 to the present), and has evolved to produce, in minimal conceptualization and analicity coincident with the minimalist program (1990 to the present), the thesis for the identity of language and music, all in all a title work (2009) by the authors Jonah Katz and David Pesetsky [213]. As a *motto* for this program we recall Mendelssohn's (very ahead or yet affine with *musica universalis*) remark: "Music is too precise to be put into words." [394, 285]

- general (strong) AI [352, 141, 325] undelivered and often absurd historical claims, namely in the field of man-machine communication [279, 87], mirrored in the failure of the Fifth Generation Computer System [52, 139, 193] (FGCS) in Japan (1982-1992) whose programming language of choice was Prolog, the postliminary historical drift and uncertainty in the discipline of natural language processing whose computational linguistics foundation

was the Chosmkyan paradigm before the shift was made to machine learning algorithms and statistics [109, 199], leading, at least in surface and at non-structural level, to the abandonment of the Chosmkyan syntactic transformational grammar paradigm in computational linguistics and natural language processing, should give us enough reasons to appoint a serious lack of philosophy of language discernment (often times found in linguistics and philosophy of language itself), very much induced by the lack of one acousmatic awareness. The most adequate *motto* for this stance is the famous Jules Combarieu's *dictum*: "Music is the art of thinking with sounds." [394, 285]

- the focus on philosophy of language subsumption and abridgement in philosophy of music permits us to test the methods of Chomskyan's latest achievements under the ongoing minimalist program, in the form of context-free transformational tonal-oriented grammar, with the abandoned Chomskyan paradigm itself in natural language processing, and while checking why this is so and why this divorce happened, we are best prepared to evaluate the rejoining arguments of *phonos* (sensuous common perceptual linguistics) and *tonos* (sensuous common perceptual music), and, what is more, to properly inquire on the *photos* (sensuous common perceptual optics), the one only left exploited experiential and perceptual expanding point in the knowledge from senses to understanding and from understanding to reason that, not just in external recollection, but also in internal apperception [211, 63, 202], is conducive to a possible *mathesis universalis* theory based on what we choose to call visualness. As a *motto* for this view, we posit Claude Debussy's aphorism: "Music is the arithmetic of sounds as optics is the geometry of light." [394, 285]



### 2.1.1.3 The Chomsky Hierarchy in Boethius' Triptych

With all these elements at hand, we are ready to scrutinize in wider comprehensiveness the *terra incognita* of philosophy of language in the emergent generative theory of tonal music [238, 237] (GTTM, 1983-2009) under philosophy of music [330, 18, 283, 411, 14, 42], accounting for an "internal merge" [213], in contrast with the proper matter of natural language processing in the programming language Prolog [39, 389, 296, 78, 85, 142] and beyond, in the due course of computational linguistics evolution [20, 142, 253] (in either text-based or speech-based man-machine interactive approaches), and past the XIX<sup>th</sup> and XX<sup>th</sup> centuries *grammatici certant* delve, conscious that a definitive orderly, either *in esse* or *in posse* schematizing, is impossible to achieve, nevertheless propose the following diagram (2.2 page 137):

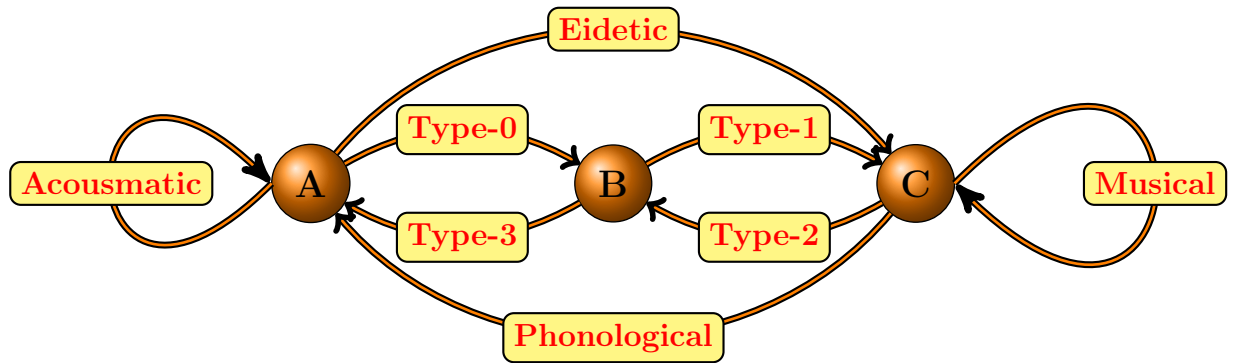


Figure 2.2: The Chomsky hierarchy contained in Boethius' triptych

## Nomenclature

- A*    *Musica Humana*
- B*    *Musica Mundana*
- C*    *Musica Instrumentis*

A proper and fair idealization of the discipline of natural language processing [142, 122, 39, 389, 296, 199], contemplating speech recognition and pragmatics (*vs.* limited syntax parsing, semantic analysis and logic programming) [199], should naturally deflect its scope to a full phonetic-tonemic (quasi-musical) [213] circle and, what is more, to a common ground of mental perceptual acousmatic *sonus*.

Natural language, principally phonetic in speech form (*vox humana* nascent from *vox instrumentis* brewed in *musica humana*) has had inherent the object of a major driven conversion under the "prolongational" [67, 115, 69] discipline of natural language processing, which was not text-to-speech form (tractable under finite-state transducers and computational linguistics phonology) [199], neither speech-to-text (tractable under speech recognition, also as a sub-field of computational linguistics) [199].

Natural language processing [85, 389, 52, 139, 193] really intends instead to speech-to-speech form under the ideal human-computer interaction (an extraordinarily incumbent argument on the "Imitation Game" [368, 363, 366, 364, 365, 367] and, thus, a counterpart representative of Boethius' [42] *musica humana* in one contemporaneous *computa humana* analogue and correlative symbolic revolution).

One such envisagement is determinant to foreshadow the status of natural language processing, its proposed aims and inherent philosophy of language, more significantly so if crafted in one such programming language as Prolog [85, 78, 39, 389, 296] (an exclusive semantic network of a context-free grammar in Prolog with a database and a syntactic analyzer or parser). Forasmuch as a former analysis should attain correct meaning representation and a sense of unambiguous word sense, more so under philosophy of language and natural language processing, and even for a "conversational language" [73] (W.F. Clocksin & C.S. Mellish, *Programming in Prolog*, Fourth Edition, 1987-1994)) like Prolog, it should be noted the distance from programmatic topics to contents of pragmatics (prosody dialogue and utterance in conversational agents, in sum the relation between the natural phenomenon of language and the context-of-use) [67, 115, 69, 199].

Being Prolog a context-free finite collection of rules defined grammar designed to convey computational linguistics and natural language processing, such a condition cannot be a contravention to fully understand the exogenous levels outward

the context-free grammar (Type-2), or the furthestmost investigations on the *substratus* of philosophy of language and sound.

This is precisely one of the reasons why, ultimately, while considering Prolog a proper subject under philosophical analysis, from logic programming to meta-logic facilities, part of its interest relies on its peculiar history as a Fifth Generation [52, 139, 193] (1982-1994) logic declarative programming language with failed *desiderata* in AI, with one *majoritas lacuna*: one absent (although parallel to) programming language philosophy pledged to speech recognition and pragmatics design. In 1971, less than two years before Prolog was crafted, speech recognition and synthesis technology was endowed with a funded five years speech understanding research DARPA program aiming at a minimum vocabulary size of 1000 words, which helped to catalyze hope on one side, but was self-explanatory enough about goals)<sup>9</sup> [199, 142].

Prolog, by, being (syntactically) sealed in context-free grammars, was ineffectual, in terms of AI [352, 325], in endowing comprehension of grammar complex outer-bounds, from linear bounded automata (Type-1) to unrestricted recursively enumerable languages (Type-0) [67, 115, 69] (within the proper Chomsky–Schützenberger hierarchy), as if its partake into a Turing machine [362] meant an inclusive bound within the limits of tractable [122] complexity. This is even more surprising if we confront AI practitioners’ circumscribed interchanging knowledge and motivations with the 1970’s vibrant and noteworthy scenario in philosophy of language.

With this special innuendo in mind, it’s not to be forgotten either and with great importance, that enshrouded in the transformation from formal logic and grammar hierarchy [51, 354, 110] to computability and complexity [256, 86, 173, 289, 16], is the Leibniz-Newton *calculus* to Church-Turing *computus* [72, 368, 362,

---

<sup>9</sup>Speech recognition is, in hindsight and recognizable structure of natural language processing, such a distinctive and founding field (inseparable from phones, pragmatics and the context-of-use of words, the proper matter of information and communication of symbols) that the success of the five years DARPA speech recognition research program (having attained a 1011 words power machine understanding in Carnegie Mellon), might have strongly influenced the philosophical ambiance in the industry welcoming Prolog. Indeed, the year of 1971, just about the time Prolog was to come up, is an hallmark date in the field of speech recognition, period after which speech recognition rose to high accuracy and multitude of tractable phones due to increased computational power and techniques.

194]. In treating the foundations of computation, we abide to an historical, logical and philosophical-symbolical perspective to the greatest possible dimension. These are such that cherish and critically think through the antecedent groundwork contributions of George Boole [43] (1815-1864), Gottlob Frege [234, 134] (1848-1925) David Hilbert [420, 119] (1862-1943) and Bertrand Russell [399, 234] (1872-1970), going beyond still to charting the different perspectives over different times and origins. They can range from Antiquity's Pāṇini's Sanskrit grammar (3959 sutra-rules) *Ashtadhyayi* and Hero's of Alexandria machine sequence control, to Modern Age Pascal's controlled carry "*machine arithmétique*" and Leibniz's explanation of binary arithmetic; they can go from the Late Medieval Lullian circle and Kerala's school invention of the floating-point number system, to the XX<sup>th</sup> century creation of the first high-level programming language (Plankalkül) (non-von-Neumann) by Konrad Zuse in Germany, and the successful development of EDSAC (integrally the von-Neumann architecture in the computer) in Britain.

It is, thus, impossible to conveniently address computability and complexity theory without recovering the algebraic logic and holistic reference to the universe of discourse (*Boole, Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*, 1854) [43]. As to the foundations of arithmetic, the distinction on sense and reference (Frege, *Über Sinn und Bedeutung*, 1892) [135], shielded in function–argument analysis of the proposition, compositionality and context principle, and also the concept and object (*Begriff und Gegenstand*), all of these distinctions utterly important to grasp the emergence of automata theory (abstract machines in relation with the context) and also type-theory (originally a Russell's response to Frege's naive set theory), can neither be neglected in a broader study.

The same holds for all the other various proposed Russell's theory of types [234, 399, 117, 58], namely the ascendance from the problematic of Cantor's diagonal argument [358, 119] (and inherent paradoxes into transfinite cardinal numbers) to the notion of "extensional hierarchy" [117] (Russel, 1959), an entire area of research without which high achievements in philosophy of mathematics and computation could have been truncated. To behold this statement is enough to recall that Alan Turing [363, 366, 364, 365, 367] used the original Cantor's diagonal method in proving the undecidability of the decision problem (the *Entschei-*

*dungsproblem* originally posed by Hilbert in 1928), Kurt Gödel also recurred to the same diagonal argument in a self-referential mirroring lemma to prove the two incompleteness theorems following *On Formally Undecidable Propositions of Principia Mathematica and Related Systems* [312, 161, 162, 160, 216, 280] (1931) and, finally, Alonzo Church used the diagonal method in the fabrication of  $\lambda$ -Calculus [137, 194, 331, 21, 172], all without which computation and programming languages [53, 58, 197] would not have risen.

# Part II

## Core Prolog

## 2.2 Essentials

Prolog is a programming language that, as other programming languages, demonstrates its *interrogative* and, thus, philosophical status, right from the *system prompt* shell in the command-line interface or command language interpreter (CLI), here SWI-Prolog[403] on a typical Unix environment:

```
1 machine% swipl
2 Welcome to SWI-Prolog ...
3 ...
4
5 1 ?-
```

The console SWI-Prolog (SWI-Prolog 7.7.10) itself, is not only "used as an embedded language where it serves as a small rule subsystem in a large application"[403] – congregating a prototyping environment, program structuring modules, multi-threading support in between servers, command line and graphics usage –, as it is also a dialogue between C (the general purpose, *imperative* programming language, with a strong design mapped directly to machine instructions, thus with a more *mechanical-industrial* ontology, developed by Dennis Ritchie between 1969 and 1973 at Bell Labs, intimate with AT&T Unix, also a product of Bell Labs in the 1970's) and Prolog (the general-purpose *declarative* programming language, with a strong design mapped to logic, thus with a more *methodological-cognitive* ontology, developed by Alain Colmerauer and Philippe Roussel from Aix-Marseille Université, and Robert Kowalski from the University of Edinburgh in the early 1970's). (As a warning, and to facilitate referencing, by Prolog we mean the ISO Prolog standard: ISO/IEC 13211-1 that was published in 1995)<sup>10</sup>.

Indeed, when one is referring to a Prolog console – GNU-Prolog[105], XSB[390], and ECLiPSe[382], or even (compiled, non-interpreted, multi-paradigm) Turbo Prolog, PDC Prolog and now Visual Prolog[329]<sup>11</sup> –, with the natural requisites

---

<sup>10</sup>"Work on the standard began in Britain in late 1984. In 1985 AFNOR formed a Prolog group, which worked in cooperation with the British group. Only in 1987 was the ISO/IEC JTC1/SC22 group WG17 created"[184], where the JTC1/SC 22 is the international standardization subcommittee for programming languages, their environments and system software interfaces, called the "portability subcommittee"(<https://www.iso.org/.html>).

<sup>11</sup>There is a wide range of Prolog implementations, besides SWI-Prolog (most of them compliant with ISO Prolog, and when not with Edinburgh Prolog): BProlog, JIProlog, Ciao, DOS-PROLOG, ECLiPSe, GNU Prolog, Jekeke Prolog, JLog, JScriptLog, jTrolog, LPA-Prolog, Open



of interaction with the environment, scalability, (I/O) reliance, standard protocols compliance, etc., it is already at stake the necessary recursive interaction with the C-language. Generally, thus, we have to be cautious not to interpret any implementation of Prolog as ethereal, once some contact with intermediate *procedural* and *imperative* language closer to machine language is an unavoidable coercion. In like manner but inversely, it is presumed that Boolean logic (bitwise or logical), sketching factorization in the construct of abstracts (just like diagonalization and cryptography) does not impose any limits to the ethereal realm of abstracts, even if it is admissible the existence of an (unapproachable) lower bound, in which case a (non-achievable) irreducibility acts on computational mathematics, software engineering and linguistics.

SWI-Prolog relies on the virtual machine ZIP (Bowen et al. 1983; Neumerkel 1993) [278, 232, 277] – "The SWI-Prolog VM is a structure copying machine that passes arguments through the environment and addresses arguments using an argument pointer. The argument pointer is also used to read and write arguments in lists and compound terms. The C-based emulator currently implements 145 instructions." [232] out of which in SWI-Prolog a restrict group of 7 instructions only is remnant [403] – represents already a far-away settled methodological approach in what concerns abstract machines. More often than not, it is thought that the WAM (Warren abstract machine) predates ZIP, but that is not the case. After Prolog 0 (1972) by Roussel working with Algol-W, Prolog I (1973) by Battani and Meloni working with Fortran, the PLM (Programmed Logic Machine) (1977-1979) by Warren finally produced the DEC-10 [89] time-sharing and later on last call optimization mainframe system.

The WAM (Warren abstract machine), which contains the memory architecture and instruction set of all the Prolog compilers – with a *global stack* or heap to store all the compound terms; a *local stack* for environment frames and choice-points;

---

Prolog, Poplog Prolog, SICStus Prolog, Strawberry Prolog, tuProlog, Visual Prolog, XSB Prolog, and YAP-Prolog.

SWI-Prolog complies with ISO Prolog and Edinburgh Prolog, is flexible across operating systems (Unix, Linux, Windows, macOS), is permissively distributed within a minimal restrictions free software license, with native graphics communicating with compiled code and unicode (non-object oriented), is endowed with native OS control, stand alone executable performance, and both C and Java interfaces. Its toolkit has the characteristics of an interactive interpreter, debugger, and code profiler.

and a *trail* to record variables bindings to be undone on backtracking<sup>12</sup> – for sequential Prolog, dates from 1983, but in the meanwhile (1982) ZIP was crafted.

A small instruction set with a simple meta-interpreter for the essential architecture, an efficient source-to-intermediate-to-machine code emulator, often with source level optimization (with methods such as *common subexpression elimination*[407] and *function inlining*[407])<sup>13</sup> drives the abstract machine to greater compactness and optimality.

In the overall, in terms of data instruction formats we have the following "data areas"[278], here under Backus-Naur form (BNF):

$$\langle AND\text{-control} \rangle ::= \langle local/environment\ stack \rangle \mid \langle global/copy\ stack \rangle \mid \langle heap \rangle$$

$$\langle OR\text{-control} \rangle ::= \langle choice\text{-point}\ stack/trail \rangle$$


---

<sup>12</sup>Backtracking is a general algorithm specially for the type of constraint satisfaction problems. By definition, it is, therefore, metaheuristic. A typical glossary definition is sufficient for the moment, and so we can assert, very simply, that backtracking is what many have said, all with a share of truth: Clocksin & Mellish: "a behavior where Prolog repeatedly attempts to satisfy and re-satisfy goals in a conjunction"[73] (in case of satisfying goals, with a place-maker behind and in case and any time a goal fails, going left-hand back, starting from its past place-maker, undoing all the temporarily instantiated variables, until a fact is found); Max Bramer: "the process of going back to a previous goal to find alternative ways of satisfying it."[284]; M. A. Covington, D. Nute, and A. Vellino: "Prolog does not know in advance which clause will succeed, but it does know how to back out of blind alleys(...)A good way to conceive of backtracking is to arrange all possible paths of computation in a tree"[373].

Backtracking is also a programming technique and a computational mathematical theorem. If one is interested in exploring how, through the use of the semi-colon ; or the goal **fail**, for example, to experiment results, can follow James L. Hein *Prolog Experiments in Discrete Mathematics, Logic, and Computability*[175] (2009) ("Or-Clauses"; "Adding New clauses"; "Modifying Clauses"; "Deleting Clauses"; "The Cut operation" in basics, but many more related); in what regards the precise delimitation of the SLD (Selective Linear Definite clause) resolution – the basic inference rule in logic programming akin to resolution – and backtracking, can check *Logic, Programming and Prolog* (2000) by Ulf Nilsson and Jan Maluszynski (namely the relation between depth-first search – originally in Zuse's Plankalkül[229, 26] – and the soundness and completeness of SLD-resolution).

<sup>13</sup>Direct examples from *An Introduction to GCC - for the GNU compilers gcc and g++*[407] by Brian J. Gough include:

```
x = cos(v)*(1+(u/2)) + sin(w)*(1-sin(u/2))
```

where *sin(u/2)* is instantiated to a free variable *t* as *common subexpression elimination*; and also:

```
double sq (double x) return x * x;
```

where, in case of loops, the *function inlining* replaces *x* by a temporary variable suitably small, permitting the loop to run.

... and machine logic operands combined in decoding the **Head** and **Goal** with necessary prefix or postfix control transfer positions, also possibly with instruction removal in each action, are all possible variable responsive shreds of design. ZIP (1983)[278] initializes all arguments on stack, with choice points of constant size in linear sequence of code.

We also learn from Wielemaker and Neumerkl [277] that SWI-Prolog (5.6.54 and most certainly still 7.6.x) and ZIP have their volatility primarily stemming from backtracking, instead of forward recursion. As a matter of fact, the "black-box procedure"[404, 48, 242] and the "partial candidate solution"[222] presupposed in the technique of backtracking[383], with roots in SNOBOL (1962), definitely has a correspondence, not only with the idea of a "storehouse of facts and rules"[73], but also with the "open-world assumption"[325]. While this is very recommendable in trivializing a philosophical (and non-dogmatic, truly skeptical) fine print – the assumption that the truth value of a statement may be true mindless of whether or not it is known to be true –, comes with the drawback of carrying the load of memory for each atom of the knowledge representation basis or *stack* in successive *trails*.

As we shall see with logic programming and Prolog – recognizing that its *declarative* status is calculated on and trusted in *procedural* routines, very much like definite integrals under the curve – also the typical "closed-world"[73, 171, 46, 45] (CWA) assumption of Prolog – "Prolog's behavior on queries including negated ground queries can thus be considered either as sound relative to the Closed World Assumption, or as sound relative to the minimal Herbrand model."[342] –, has, very much due to the mechanism of backtracking, necessarily to be balanced in relation to the open-world assumption (OWA). Hence, in Prolog, axiomatic knowledge, akin to classical logic and the closed-world assumption (CWA) divides but depends on consequent inferred knowledge, akin to constructive/intuitionistic logic, and the open-world assumption (OWA). All in all, even if Prolog does not retract any of its **Facts** neither at compile time nor at run time, there is some monotonicity of entailment with freely extended additional assumptions, and, except for first order predicate **Rules**, axioms of the system can always be subject to cut and failure, even if provisionally with backtracking, and yet more definitely so re-instantiating **Facts**, **Predicates** and **Rules**.

This *liaison* of Prolog with expendable and disposable information has its informatics epitome in initialisation proper. In Dec-system-10[89] (1.11) (1982) this is very clear with same call realization of both 'prolog.bin' in the default path:

```
1 | ?- restore ( 'prolog.bin' ).
```

... and, in case no such file is found, the automatic equal search, now 'prolog.ini' being set in motion (it is also possible to run one program Prolog file through an already existent work file, by typing `plsys(run(_, _))` for, respectively, *program file* and *offset*):

```
1 | ?- [ 'prolog.ini' ].
```

In the case of SWI-Prolog[403, 232], it happens to have its own built-in native development tools. Some are, clearly, OS inspired, such as the **Prolog navigator**, with a windows and folders mouse-pad friendly file presentation and management. It also contains a **Cross Referencer** creating a hierarchical file overview, an **Execution Profiler** which displays call and time statistics of the program (also executable through `profile(:Goal)`), a **Graphical tracer** with source-level debugging, using two parallel windows – with variable bindings and the stack – and one detached window providing top-level source code processing for verification and performance analysis. But, more important and above all, in analyzing the conundrum between philosophy of language *problemata*, programming languages theory and logic programming, we refer to **PceEMacs**.

**PceEMacs** is the work on XPCE/Prolog[401, 402, 403], being XPCE/Prolog a product of "knowledge intensive graphical applications"[402] for dynamic typed languages, developed by Anjo Anjewierden and Jan Wielemaker from the department of SWI, so to create an "object-system written in the C language"[402], but, by its nature, at the same time mimetic, for how much it is permitted, of the object-oriented paradigm in the facets of "event-driven control and global state"[402]. With XPCE/Prolog, thus, combining an object-system, a GUI (graphical user interface), multi-threading typical-Unix process and file system blocks, and, finally, debugging and statistics – and specially if recalling that **PceEMacs**<sup>14</sup> is, in its

---

<sup>14</sup>**PceEmacs** – started by using the predicates `emacs/0` (directly opening the scratch buffer), `edit/0` (editing the default file) or `edit/1` (`edit+Specification`) – being a Richard Stallman's **GNU-Emacs** act-like in **XPCE/Prolog**, has implemented Prolog syntax highlighting based

turn, a clone of **GNU-Emacs**, a family of extensible text editors that are written both in C, and largely on Emacs Lisp, a (Turing-complete) dialect of Lisp – the result at hand is a sort of informational (code) and informatics (programming languages) example of the phylogenetic law of recapitulation [150, 258].

By this we mean that at a single point, after decades of programming languages development and the turnover of multiple paradigms surges and crossovers – *functional, imperative, structured, procedural, automata-based, declarative, object-oriented* and *event-driven* – with decades of dynamic programming languages *dialectic* – from  $\lambda$ -Calculus (1936) to Plankalkül (1943-45), from LISP (1956-58) to Algol 68, from Prolog and C (1972) to C++ (1983) and Python (1991), from Java (1995) to the none-unique languages web-development era – it seems that, *artificiality* and modern cybernetics considered, *programming* ontogeny recapitulates *code* phylogeny.

It is, therefore, admissible that *functional, algorithmic*, and also philosophy of mathematics and programming (conjectural) *formalism* and (type-theory) *structuralism*, partake, at least in what regards automata and computational classes, in some sort of determinism.

It is also true that in this prospect, the *procedural* and *imperative* paradigms would correspond to the least upper bound, while the *declarative* paradigm would be accounted for the greatest lower bound of the programming abstract (machine and automata) field. Historically, it is very interesting to observe that this

---

on parsing and cross-referencing in the editor buffer. The result is "colouring highlights variables, quoted entities, comments, goals (classified as built-in, imported, local, dynamic and undefined), predicates (classified as local, public and unreferenced), and file references (classified as existend/non-existend)" (<http://www.swi-prolog.org/IDE.html>).

The colour highlighting is also present at the proper  $\LaTeX$  *The Listings Package*[185] in play on this thesis. This very simple feature is, nevertheless, tremendously important. If we observe carefully, it is the most basic, mild and benignant syntax feature related with the faculty of understanding and schemata, confronting with the programming idea of  $\cup$ -Mentalism. One very good example of the very same, faint but at the same time bright, applied principle is Oliver Byrne's *The First Six Books of The Elements of Euclid*[56] (1847) edition in colour. If we think that currently per year there are large groups of tens of trillions of digital images, mounting on growing values each year, apart from the proper *photon* natural ontogeny of metaphysics, creating a massive world's image data bank, independently of the computer vision and multiple-view geometry programming media syntax or "language code" (with or without machine learning) in relation with the programming *ideal* of  $\cup$ -Mentalism, we understand how much it is suitable a grounded *apperception*, in Kantian terms, of the exposed new riddle.

artificial-natural programming antinomy has resulted in a constitutive regulation of both the *procedural* and the *declarative* paradigms in the same year (1972) – alike the truth tables method of testing validity (1921-22), computability (1936) and computer architecture (1945) *anni mirabiles* – with the craft of C and Prolog. C and Prolog (1972) represent, in terms of postulates of programming *reason*, the most equally rational, but-contradictory principles, regarding natural-artificiality as the transcendental (virtual) reality.

Most surely, after the ENIAC [320, 179] (1943–45) coding system by John von Neumann (with the conjoint work by Mauchly, Eckert, and Goldstine) and Alan Turing's ACE [364] (1945) – bearing in mind that both authors engaged into the stored-program concept and the idea of programming languages, also cogitating about naturalistic biological phenomena, and even though neither one did come to live past the 1950's, or even J. McCarthy's LISP (1956-58) – it would not be to astonish their appraisal and maybe concordance with one such law of recapitulation in artificiality. Indeed, Ernst Haeckel's *Art Forms of Nature*[165] (1899-1904) is certainly, one of the foregoing building examples of the interest in morphogenesis, and Mendel's (1822-1884) work on inheritance and the idea of (self-replicating) genetics is at the core of von Neumann's computational and programming philosophy.

Using SWI-Prolog straightforwardly with C/C++, it is also possible, through **MinGW**, a "Minimalist GNU for Windows"(<http://www.mingw.org/>), to write C/ C++ code loadable into SWI-Prolog and call it as a predicate, or the other way around, embedding SWI-Prolog in C/C++. **MinGW**, (a non-Linux kernel) a minimalist development environment for native Microsoft Windows applications, obliges, thus, SWI-Prolog to depend on Microsoft Visual C++ MSVC, itself an integrated development environment (IDE) for C/C++. In one such case, and in spite of `swipl-ld.exe` automatic compatibility, to use in plenty manner **MinGW** it is required to set the the path environment variables – the SWI-Prolog binary variables and the **MinGW** binary folder – in the same place, usually

```
C:\MinGW\bin
```

so to find GCC ('G'NU 'C'ompiler 'C'ollection) files, which include front ends for various programming languages – C/C++/Objective-C/Fortran/Ada/Go

– along its libraries, meant to work as a compiler for the **GNU** OS.

The necessity of SWI-Prolog depending on the debugging and writing facilities of **MinGW**, is forwardly incremented by its dependency on MSVC's C/C++/C# code writing and debugging (Microsoft Windows or the operating system, the .NET Framework with language interoperability and code management, or DirectX as a set of routines, protocols, and tools for building software applications for audio and video hardware). Anyway, even if work is being run independently in SWI-Prolog, we know that the very same language interoperability, code management and technical translation is being matched, but underneath SWI-Prolog's skin. All in all, it is one scenario to think back the quote in *Essentials of Programming Languages*[136] (2001):

"Consider again the basic idea: the interpreter itself is just a program. But that program is written in some language, whose interpreter is itself just a program written in some language whose interpreter is itself (...) Perhaps the whole distinction between program and programming language is a misleading idea, and future programmers will see themselves not a writing programs in particular, but as creating new languages for each new application." [136]

Accordingly and in the same line, we should not be oblivious to other appointments [106, 387, 107, 75]. Indeed, we learn from Daniel Diaz and Philippe Codognet, for instance, in relation to the creation of the **wamcc** system, a Prolog to C compiler based at the Wam (Warren abstract machine), how intrinsically the passage to C flashes back to a time when the standard compilation and one abstract machine for the execution of Prolog, consisting of a memory architecture and an instruction set, was itself a "breakthrough in logic programming"[75].

Not only elements of computer architecture were studied close to the concept of logic programming – the idea of "computation being expressed as controlled deduction from declarative statements"[15] by Kowalski – but what is also relevant is that, after the forerunner Q-systems[77] written in Algol (1967-1970), the first Prolog implementation (an interpreter written in Fortran) by Colmerauer's group, the DEC-10 Prolog system[89] (1982) by the University of Edinburgh (an interpreter written in DEC-10 machine code), later refined into the WAM (Warren

abstract machine) (1983), precisely at a time when the FGCS (Fifth Generation Computer Systems) (1982) was leveraged (with KL1, a Prolog parallelised version as the system's language), the wield of a Prolog standard compiler also commanded critical consultation to the theory of programming languages.

This exceptional, although unnoticed, overcome was achieved by way of mechanical explanation of SLD-resolution[140] (Maarten van Emden's coinage for the unnamed inference rule introduced by Robert Kowalski in logic programming), originally dimensioned by Davis and Putnam principle of resolution (1960), and J. A. Robinson's [316] (1965) unification algorithm, all in all bestowed on (Alfred) Horn clauses (1951) mathematical formulae fit for formal specification.

Hence, the Wam (Warren abstract machine), in terms of mechanical intelligence, avows for the SLD-resolution (Selective Linear Definite clause resolution), which was no more than a clause translation – by means of a restricted linear sequence of clauses – which, in turn, accounted for extensive work by J. A. Robinson (1965) on the restricted syntactical unification algorithm by calling the instantiated formula only to equal satisfiability with refutation completeness (*P*-like problem), thus disciplining Davis and Putnam's resolution (1960), which was combinatorially explosive, in that their algorithm tried out all ground instances of the given formula (*NP*-like problem).

New exploratory avenues shall be encouraged, routing from these observations. A clear portrait, as it seems rational, is to understand how the Wam (Warren abstract machine) (1983), the standard Prolog compiler, represents a programming philosophy recapitulation of the C-language and Prolog (1972) conjoint, although dual, event, in their *procedural* and *declarative* paradigms. With this in mind, it shall be clearer the nature of logic programming. At a more abstract level, it should be understood how resolution and its craft (constraint optimization) as a refutation complete (*P*-like problem, *falsificationist* in terms of philosophy of science) accounts for the essence of logic programming.

Contrary to the inherent dilemmas of philosophy of mathematics in relation to completeness and consistency fetched away from the intricacies of mechanical intelligence, the "how" (*procedural*) and the "what" (*declarative*) have really come to approximate themselves to the most possible constricted and self-harboured programming shell. With this we mean that logic was elevated to a programming



paradigm, vindicating the *declarative* status, precisely by constricting itself to control in *procedural* terms. It is under this guise, that we shall understand the famous Robert Kowalski's logic programming equation and work title:

"Algorithm = Logic + Control" [120].

Exactly like Turing resolved computability (in a bottom-up approach, through mechanical intelligence) faced up with Gödel's announced mathematics incompleteness (in a top-down approach, through philosophy of mathematics), logic was elevated to a *declarative* status by resolving *procedurally* typical mechanical intelligence *problemata*, such as Horn-clauses (1951), Davis and Putnam principle of resolution (1960), and J. A. Robinson's [316] (1965) unification algorithm, and consequently the SLD-resolution (1972) refinement method over simple SL-resolution (Kowalski and Kuehner, 1970/1971). The Wam (Warren abstract machine) (1983), although not having called great attention as the FGCS (Fifth Generation Computer Systems) (1982) in Japan, is, nevertheless, the paramount event, regarding programming philosophy.

More so and crucially, it can be said to have reevaluated an artificial-natural programming antinomy – the *procedural* and the *declarative* paradigms – with a correspondent equilibrium executed by Control in

"Algorithm = Logic + Control" [120].

When we refer to one such equalization expression – wherefrom an equivalent expression, maybe more suitably *critic*, would be

Control = Algorithm + (– Logic)

are, really, deeply ingrained, the antagonist edges coming to a full-fledged programming core. This core is best described as having associated the *algorithmic*, *numerical*, *procedural*, and akin to *diagonalization* computing methodology (with previous Q-systems in Algol, implementations such as Fortran and DEC-10 machine code), contrasted with the *logic*, *syntactical*, *declarative*, and akin to *parallel* computing methodology (with implementation such as SLD-resolution, a special kind of Horn clauses). In this formalism and equilibrium, logic satisfiability and

theorem-proving correctness were, all in all, resolved in one programming language, "in which a computation is in fact a refutation"[140].

Horn clauses [188, 326, 393, 73] (1951) – at a time when AI's birthright (1956) by McCarthy in Dartmouth College, with attendees such as Claude Shannon (1916-2001), Marvin Minsky (1927-2016) Allen Newell (1927-1992) and Herbert Simon (1916-2001), was not even established as a field – are, basically, an important formalisation towards any knowledge basis representation, in respect to the logical inference problem.

In a knowledge representation basis – no more than a "set of sentences"[325] – we need to know if a sentence  $\alpha$ , understood as a theorem or a logical consequence of the the axioms, is, under the expressed according *syntax* well-formed formulas of the language, said *semantically* true with respect to each possible world. Knowing that semantics crosses *representation* (wherein a sentence entails other sentences) and the *world* (wherein a world grounding reasoning follows from another congruent established aspect of the mind/world), we need to know, as a Goal, if our model knowledge basis is theoretically sound, or else called truth-preserving (by means of carrying out logical inference while model-checking [54, 349, 327]) and if it holds completeness, in the sense that an "inference algorithm is complete if it can derive any sentence that is entailed"[325]. Under the very same propositional logic law of equivalence and propositional logic proper taken as a model<sup>15</sup> of representation

---

<sup>15</sup>In model theory[327, 349, 54] are famous the equation-like formulas MODEL THEORY = UNIVERSAL ALGEBRA + LOGIC [215], as well as MODEL THEORY = ALGEBRAIC GEOMETRY - FIELDS [183]. Duality in model theory – the idea, indebted to Alfred Tarski's (1901-1983) of the definition of a "true sentence" (1933; revised in 1956 with Robert Vaught) for particular formal languages, and consequently, meeting a truth definition for model-theoretic languages, which is balanced both syntactically and semantically, in logical completeness and soundness (and both in procedural and declarative biases; and in, less obvious, but also adequately, analytic and synthetic ways), even if with pairwise equivalence, and therefore, redundant metatheoretically, is a *representational* (sound over complete) correct (correctness being the arrow from syntax to semantics) and valid (validity being the arrow from semantics to syntax) theory – something that finds echo in the following: "Model theory is not a semantical theory which relates natural languages to the physical and social reality, but rather a mathematical theory which relates some mathematical structures to other mathematical structures."[54]. Model theory, in its rigour, can be, thus, considered an analytic and mathematical *critical* philosophy improvement of Kantian philosophy, and also, among many exemplary contributes, very much bounded to many-sorted logic and typeful programming philosophy[58].

In a short note, apart from the programming abstract FIELD being inclined to harbour model theory LOGIC – thus, letting speculate on the nature of ALGEBRA and GEOMETRY natural

of our knowledge basis –  $p \equiv q$ ,  $\exists pq$ , or  $p \iff q$  – we ask if the knowledge representation in which all of the formalised statements are true, entails the atomic sentence  $\alpha$  – in Backus-Naur form (BNF):

$$\langle \textit{Sentence} \rangle ::= \langle \textit{AtomicSentence} \rangle \mid \langle \textit{ComplexSentence} \rangle$$

$$\langle \textit{AtomicSentence} \rangle ::= \text{True} \mid \text{False} \mid \text{P} \mid \text{Q} \mid \text{R} \mid \dots$$

$$\langle \textit{ComplexSentence} \rangle ::= \langle \textit{Sentence} \rangle \mid \langle \textit{Sentence} \rangle$$

$$\mid \neg \langle \textit{Sentence} \rangle$$

$$\mid \langle \textit{Sentence} \rangle \wedge \langle \textit{Sentence} \rangle$$

$$\mid \langle \textit{Sentence} \rangle \Rightarrow \langle \textit{Sentence} \rangle$$

$$\mid \langle \textit{Sentence} \rangle \Leftrightarrow \langle \textit{Sentence} \rangle$$

– and from here we can assert that Rules are no more than if-then statements.

If carefully noticed, this is a proof of recursion-completeness of propositional calculus, as long as a formula or assertion that is true in every possible interpretation makes the model checking set of sentences derivable.

This sense of tautology, a formula whose negation is unsatisfiable regardless of its possible truth-values – the heart of SLD-resolution [140, 223] in logic programming and Prolog –, is very suitable to *entail* itself tautological implication with the law of equivalence. But, while this has a positive theorem-like implication [25, 236] for sentential calculus being both consistent and complete, as for the rest, any logic programming based on formal logic Rules whose negation is unsatisfiable, and forwardly and on the far side, higher-mathematical abstract first-order calculus proof of completeness – (1929) (Gödel’s completeness theorem establishing a correspondence between semantic truth and syntactic provability in first-order logic)[160, 216, 280, 161]; (1947) (Leo Henkin’s simplified proof of the semantic completeness of first-order logic) – the fact that in propositional calculus proof itself is the proper calculus should make us regard tautology also into the more obscurely quizzical Wittgenstein’s view [144, 413, 412, 143, 417, 274, 416, 310, 91, 415, 411].

---

isomorphism – we let also be thought throughout how  $\cup$ -Mentalism, as a programming philosophy *critical* limit, as the collection of all possible *differential* images for each possible *integral* viewpoint, can be explored in one such equation as:  $\cup$ -MENTALISM = MODEL THEORY + MULTIPLE VIEW GEOMETRY & COMPUTER VISION.

Indeed,  $\models S$  (here in the double turnstile notation, and  $S$  for sentence) indicates a tautology, yet there is also a sense of redundancy  $\top S$  (here in the tee symbol notation) denoting an arbitrary tautology – capturing the sense of adding, at the utmost, one positive (unnegated) literal to a clause such that it forms a disjunction of literals, i.e., a Horn clause in the global – or even  $\perp S$  (here with the dual symbol *falsum*) denoting a contradiction (which is, alike a tautology, a universal truth in formal logic). Now, this last example is as much transparent of Wittgensteinian fideism [418, 419, 416, 310, 91] (faith being independent of reason,), as it is of Kantian critical philosophy [116, 114, 211, 63, 209, 205, 204, 208, 207, 206, 203, 210] (faith being grounded in the needs of practical reason).

Moreover, this serves as the best example of both Kant's critical philosophy in relation to antinomies [63, 209, 211] (apart from the *ideal* representation that contradictions are inherent to the necessary transcendental reality, focusing in the fact that, *logically* and necessarily, they are contradictory  $S$ , established as a **Fact**, inasmuch as a **Query** and a **Goal** coincide) and, to our aim, in terms of programming philosophy, the idea, not contrary to many *procedural* programming languages and databases, however also of Prolog's, closed-world assumption (CWA)[73, 171, 46, 45].

In the closed-world assumption (CWA), we depict a statement  $S$  to be true also known to be true. Conversely, any statement  $S$  not known to be true, is automatically interpreted as false in Prolog. This complies with so called "negation as failure"[337, 214], intrinsically related to the closed-world assumption (CWA).

Indeed, Prolog is a recursive computational sentential function, **Goal** and **Query** are isomorphic, and each and every **Rule** of the system is a program computer first-order logic (FOL) formalism and axiom. This is also the exact point where about the analytically *a priori* Hilbert's program can meet a synthetically *a posteriori* computer program. And it is also bearable of a fairly grounded suspicion of the law of atavism in logic and programming philosophy, whereby an ancestral trait reappears after having been dismissed through evolutionary change. Crudely, after the established limits in computation defined by Gödel and Turing, it is thoroughly non-congruent to have been found in any way the sort of social-technological and philosophical-informatics expectations as the strong AI design of the Fifth Generation Computer Systems (FGCS) in Japan, nor, for what the matter concerns, the

same kind of contemporaneous strong AI *Principle of Hope*[41] (adapting a term from Ernst Bloch's utopianism).

By the same token, and as a *reductio ad absurdum* proof, conflagrating the sense of tautology and contradiction in one proof (itself the intricate sense of a Kantian antinomy exposition, or Gödel's incompleteness theorems), if we managed to Predicate, in a Prolog program, its first-order logic **T** (theory) and syntax **S** as a set of sentences  $S$ , wherein for each wff  $\phi$ , its naming  $\ulcorner \phi \urcorner$  would account also for a `prolog.swi` File, we know (CWA) that for any wff  $\alpha(v)$  (with one free variable) meaning an instantiating placeholder in the general program (OWA), for a sentence **T** (FOL's classical logic theory  $n^{th}$  possible lists), according to the fixed-point theorem and the diagonalization mapping, would, consequently, map each  $\phi(v)$  to  $\phi(\ulcorner \phi(v) \urcorner)$  representable in **T**, and therefore, by means of the programming language  $\ulcorner \phi(\ulcorner \phi(v) \urcorner) \urcorner$  in a compiler. With that accomplished, diagonalization is performed, and so is *imaginably* computation: at the level of compiler and interpreter final processing to binary *mathematical arithmetized* machine code.

In continuity, once our representation (OWA) has formalized a diagonal met-language in self-reference (imagining the 46 preliminary Gödelian definitions for *Principia Mathematica* formalized calculus to be fully arithmetized and bound to primitive recursive arithmetical truths, although in conventional symbols of a met-alaguage), thus producing, in diagonalization, for any single Gödelian number intertwined to any recursive computer Prolog program and `prolog.swi` File, a non-provable nor disprovable list of sentences  $S$ , the result is that Prolog, however in its formalization represents a *constant* value and, thus, a first-order logic (FOL) axiomatic construction, it is, in truth<sup>16</sup>, nothing but a *variable, unknowable* and

---

<sup>16</sup>There is an antagonism of Wittgenstein towards Gödel's theorems (to the 1<sup>st</sup> at the very heart, but also to the 2<sup>nd</sup> as an extension) that is worth our attention. In §8 of *Remarks on the Foundations of Mathematics* (Appendix 3) (published posthumously in 1956) belonging to the period 1937-1944 and, in general, apart from the philosophical incidence in 1929-1934, recollected and published in *Philosophical Remarks* (1964) and *Philosophical Grammar* (1969), this was a never-repeated focus on philosophy of mathematics by Wittgenstein. In the referred §8, now renowned as the "notorious paragraph", Wittgenstein imagines, in *reductio ad absurdum* or, in this case, *reduction ad logicam*, what conclusions would have to be drawn if the Gödelian formula P in "P is not provable in Russell's system"[414] –  $P \vee \neg P$  – accounting for both *true* and *provable* in demonstration of the so called "key claim" – would be false.

The author wrote: "Must I not say that this proposition on the one hand is true, and on the other hand unprovable? For suppose it were false; then it is true that it is provable. And that

*antinomical.*

In other words, inasmuch as Gödel's incompleteness theorems recur to syntactic constructs in formal logic made representable in the very same theory by alternate naming, so do we find an adequate understanding of the mechanical (diagonal and computing) action of a compiler and interpreter indifferently. Indeed, computing meets diagonalization, insofar a compiler or interpreter denote equality

---

surely cannot be! And if it is proved, then it is proved that it is not provable. Thus it can only be true, but unprovable." [414]

In respect to this passage the most extreme critics have been publicized. As soon as the book was published, Kreisel (1958), Dummett (1959), and Bernays (1959), appraised negatively Wittgenstein's *Remarks*, and in the past twenty years so did Rodych (1999, 2002, 2003, 2008) [118] and Steiner (2001) [328]. On the contrary, Floyd and Putnam have described it as having a "remarkable insight" [311] into Gödel's proof.

In our view, we have to say, in defense of Gödel, that incompleteness and undecidability resolve in a sort of natural antinomy of mathematics towards itself, and consequently, it is just as permissible to think "P is not provable in Russell's system" [414] –  $P \vee \neg P$  – as it is to depart from the assumption "P is provable in Russell's system" –  $P$  – finding, in both ways, either truths about the arithmetic of natural numbers in *PM* (Russell & Whitehead's *Principia Mathematica*) that are not provable, or a failure to demonstrate the overall consistency of any system containing the truths about the arithmetic of natural numbers in *PM*.

But in defense of Wittgenstein, and except the lines where his critic seems to fall to Gödel's own philosophy of mathematics, the incompleteness theorems and the undecidability thesis, we have to call attention to the fact that, precisely attending to the tenure of the latter, in one such case, "truth" and "proof" cannot mean much. In a sense, if it is fair, as a logical method, to formalize *reductio ad absurdum*, presuming a certain axiom to be false, so to deduct the logical truth of a set of sentences in a system, it is also fair to presume an axiom to be true, when not provable (in the context of the Gödelian argument for the incompleteness of mathematics and its undecidability), so to presume the logical truth of contradictionariness and, thus, of inconsistency of the system. This view would abide by some sort of paradox, once Gödel's arguments are already a (diagonalization) proof for the case when we take universal logic truths about the arithmetic of natural numbers in *PM* true but not provable, yet never provable but not true. In a way, we call Gödel's theorems proofs, and never are we seen calling them conjectures, even if, by means of proof, they disprove the proving method that constitutes proof itself, and it might be this conundrum labyrinth that Wittgenstein is referring to.

If this interpretation of Wittgenstein's words might be right, it turns to be more remarkable not so much Wittgenstein's proper remarks, but instead why they are not framed in an anticipatory mode to all the philosophical apprehending typical of Wittgenstein II (such as "meaning as use"; "language-games and family resemblance"; "rule-following and private language" and also "grammar and form of life" [38]) (also to consider is how "grammar and form of life" [38] is quite exemplar of the language-equivalent of Turing's *morphogenesis* in the phenomenon of life, remembering Wittgenstein's *dictum* in Norman Malcolm's book *Memoir*: "What I offer is the morphology of the use of an expression." [252], attesting for Wittgenstein "the morphologist" [228, 299]). To this we add that in one such scenario, we have to consider precisely the inverse of what Rebecca Goldstein has hypothesized [147] – Gödel having developed his logical theorems in opposition to Wittgenstein – and consider more fundamentally the influence of Gödel's theorems on Wittgenstein, namely the II Wittgenstein.

in an (*arithmetized* binary) metalanguage.

The fact that a set of sentences  $S$  closed under logical consequence of the theory  $\mathbf{T}$  in a fixed logical system is decidable in first-order logic (FOL), while, both in Gödelization and compiling, there is not an effective method for determining whether any arbitrary formulas are included in the theory or not, is truly a very thin frontier. Confronting with the set of physical phenomena associated with electricity, in the same manner in an atom protons and neutrons are positively and negatively charged, it could be said that consistency and incompleteness are mathematics' and logic's natural ambivalence. Computation and diagonalization, as interval methods between both, are very identical as a mechanism: alike the beneath phenomenon, i.e., the electrical current as a consequence of the flow of negatively charged electrons around a conductive material, driven by overhead pole transformers causing the electrons to move, is recursively transformed to higher calculation and memory powers; and alike the top phenomena, i.e., mathematics and logic, the Boolean-driven and bitwise-commanded electrons give the calculation and memory flow abstract significance. If, additionally, a Prolog program is being run in this core, there is another diagonalization, in the form of an abstract-philosophizing and type-programming tension, that is being performed, i.e., the natural-artificial (programming) antinomy of both the *declarative* and *procedural* paradigms.

While this is consensual, what is often forgotten is how logic programming and Prolog – wherein logic is an expression of **Relations** which are all defined by **clauses**, with a **Query** meaning that the data types **terms** (**atoms**, **numbers**, **variables** or **compound terms**) engineer successive attempts to find a (resolution) refutation of the negated **Query**, this action being equal, in case the negated **Query** is successfully refuted, to an immediate instantiation of all the free (unbounded) variables, thus unifying the clauses and the (negated) **False Query**, following from here that this original **Query**, as an **[Head]**, with the instantiated variables as the **[Tail]**, are all together a logical consequence of the program – sums up, in comprehensive details, the proper diagonalization method behind the Gödelian incompleteness theorems. Hence, elevating from Prolog programming language **terms** to philosophical metalanguage terms, we can assert that logic programming and Prolog instantiate **truth** as a free variable.

In other words, the proper truth-preserving validity (in *parallel* Euclidean method) is a construction and proper *place*-holder, in the full *topologized* sense of programming languages' fixed-point theorem [247, 345], of both (in)completeness ADN/OR (in)consistency unbalance (in *diagonalization* non-Euclidean method), accounting for *problemata* related with modern philosophy of space and time (Leibniz's metaphysical *monadism* vs. Newton's physics *absolutism*), and contemporaneous philosophy of spacetime (Einstein's theory of relativity local and universal *determinism*, vs. quantum theory and Copenhagen's interpretation local and universal *undeterminism*), that are far more subtle than primary injunctions of common foundational systems for mathematics. This is the reason behind *understanding* [211, 63] – in the Kantian sense of the faculty of concepts – of logic Programming and Prolog, in the context of the Fifth Generation Computer Systems (FGCS) in Japan, being so remindful of naïve set theory.

Centrally, the proper definition of an  $\omega$ -consistent theory, the name having been coined by Kurt Gödel sequent to the incompleteness theorem proof – attesting for a collection of sentences that is not only (syntactically) consistent (without contradictions), but is also preventive towards extensively proving infinite combinations of sentences that are intuitively contradictory [218] – adequates to logic programming and Prolog, for these are also numerically segregative, truncating higher-orders, generally forcing to preserve consistency.

If an arithmetic theory  $\mathbf{T}$  is  $\omega$ -incomplete it is necessarily the case that for some open *wff*  $\phi(x)$ ,  $\mathbf{T}$  can prove each  $\phi(\bar{m})$  but  $\mathbf{T}$  fails to prove  $\forall x\phi(x)$ , thus enacting  $\omega$ -inconsistency, i.e., if a theory can prove each of  $\phi(\bar{m})$  and, at the same time, also not disprove  $\neg\forall x\phi(x)$  is inevitably going to be contradictory within the realm of natural numbers. Equally so, logic programming and Prolog, fundamentally through Horn clauses, linger in completeness and consistency by a pattern of collection of negated disjunction forms (preventing incompleteness) – for which an implication form with the unnegated literal now representing the **Head** and a list of conjunctions the **Tail** (preventing inconsistency) – attest for, translating to meaningful machine code, a collection  $\mathbf{T}$  of  $\sum_1^0$ -sentences provable in  $\mathbf{T}$  that mimic (necessarily sufficiently) the structure of natural numbers with (necessarily) addition and (sufficiently) multiplication.



Generally, thus,  $\omega$ -consistent theory in logic programming and Prolog, is *constructively* operated through the *parallel* validity-defender, *analytic* decidable and truth-preserving Euclidean method of *necessarily* disjunctive addition, and the *diagonalization* cryptographic-defender, *synthetic* incomplete and truth-halting non-Euclidean method of *sufficiently* conjunctive multiplication.

This is also a formal explanation why logic programming and Prolog, being  $\omega$ -consistency a **Cut** decision procedure in the context of computation, is forcibly also keen to **HaLt**. Just like first-order Peano arithmetic is, nevertheless incapable of proving its consistency, perfectly satisfiable and, therefore, consistent, so too (computable-numerical) *dynamical* logic programming and Prolog cannot prove anything contradictory from their (computable-numerical) *mathematical* axioms.

The proper *declarative* paradigm resorts to this *limit* and, *theoretically*, it is important to refer it in relation to Horn clauses (1951), for the reason that the clausal-form disjunction of literals (CWA) (constant-like definite clause, *a priori* truthful) is made closer to (OWA) (variable-like **Goal/ Query**, *a priori* false) Horn clause.

Horn clauses are, thus, *procedurally* and in (CWA) Prolog, as much a balance towards open-world assumption (OWA), as the mechanism of (recursively enumerable but not recursive) Davis–Putnam algorithm and backtracking partake into the same balance line. All in all, these observations reinforce the definition of Prolog as a "relational language"[73, 253]. On this wise, **Relation** in Prolog, under the *declarative* and *procedural* paradigms, is pairwise equivalent also according to the *intensional* and *extensional* views, while its meaning should also be extended to meta-theoretical expressions.

To the argument that logic programming and Prolog facets do not respond to this core, and that this is somehow alien to its *declarative* semantics and *procedural* syntax, we respond by saying that, inasmuch as Herbrand's theorem – which implies that a prenex-form reduction for any formula has, thus, an unsatisfiable ground instance, turning a formula valid if and only if its negation is unsatisfiable – is at the base of Davis-Putnam algorithm and the resolution-based decision procedure for propositional logic in logic programming and Prolog, of which any  $\Phi$  Horn clause is nothing but a **Fact** implied to prove universally quantified  $\neq \Phi$

Horn clause instances, being, therefore, unsatisfiable, and being also the *declarative* nature of logic programming and Prolog a sort of *procedural* product of partial computable functions (  $\text{mod } \neg$ ), we can simply and likewise, *refute* this *judgement* programmatically in the spirit of Prolog.

If we said before that we have to be cautious not to interpret any implementation of Prolog as ethereal, once some contact with intermediate *procedural* and *imperative* language closer to machine language is an unavoidable coercion, we say now, but inversely, the exact opposite: we have to be cautious not to interpret any implementation of Prolog as earthly, once some contact with intermediate *declarative* and *formal* language closer to higher-abstract philosophy of logic is an unavoidable coercion.

We are affirming that logic programming and Prolog hinge on, with increasing non-perceptibility as the focus on theorem-proving is highlighted, not so much extraordinarily in the frontier of (logic) *declarative* vs. (imperative) *procedural* paradigms, dragging the natural ambivalence of logic programming (with philosophical and programming transactions with formal logic in atomic formulae symbolic expressions, and clausal form as an axiom subset of Lisp, i.e., high-level abstractions from machine code) vs. imperative programming (with philosophical and programming transactions with structured control flow in numeric routines and assignments, i.e., low-level abstractions from machine code) but more so on the philosophical frontier that is associated with Wittgenstein I and Wittgenstein II.

In other words, Wittgenstein I – typical of *Tractatus Logico-Philosophicus* (1921) converging truth-functionalism, from what place "form is the possibility of structure" (2.033)[418, 419], thus with (weak & modal)  $\diamond$  mathematical formalism attaining for (strong & necessary)  $\square$  mathematical structuralism, with geometric projection via logicism of "state of affairs"(2.034;2.04)[418, 419] drawing a picture (tendentiously *mathematical*) theory of language set up on "operationalism", with a strong delimitation of the sense in language expressions and irrestriction of references –, and Wittgenstein II – (typical of *Philosophical Investigations* (1953) converging mathematical anti-Platonist finitistic constructivism with "anti-foundationalism (...) criticism of the extension-intension conflation"[118], from what place a picture-theory of language is impossible (representing at utmost an

approximate reference bound), thus with a (strong & modal)  $\diamond$  "language-game" born on (weak & necessary)  $\square$  patterning "family resemblances", drawing a cinematic (tendentiously *dynamical*) theory of language set up on "rule-following" with a strong delimitation of the reference in language expressions and irrestriction of sense – as philosophical stands, prior to programming languages analysis, seem to account more profoundly for the frontier between automated theorem-proving on one side, and, on the other side, for non-deterministic knowledge representation basis.

Again, operationalism (here as a sort of computational positivism) and the "language-game" (here as a sort of "rule-following" non-deterministic agent-based computational behaviorism) suitably represent, in the specific case of Prolog, a "relational language"[73, 253], what, in essence, we can designate by its intrinsic philosophical and programming antinomy. Indeed, it is fair to say that, at an human-agent oracle basis, it corresponds to *transcendental* knowledge, insofar we can find for each side the *correct* interpretation of the other side of the antinomy. This can be shown by exemplifying what, in abstract, Wittgenstein has pointedly singled out – "The question how a correlation of relations is possible is identical with the problem of truth." [410] (Wittgenstein, *Notebooks 1914-1916*, 24.9.14) – and, in the specific case of logic programming and Prolog, what we read in *Logic for Computer Science: Foundations of Automatic Theorem Proving* – "In fact, it is often claimed that logic programs are obviously correct, because these programs 'express' the assertions that they should satisfy. However, this is not quite so, because the notion of correctness is relative, and one still needs to define the semantics of logic programs in some independent fashion." [140] – which, all together, *simulate* knowledge about our cognitive faculty with regard to how objects are possible *a priori* in the *transcendental* subject.

In essence, therefore, we are radically illustrating how the *transcendental* possibility of philosophical and programming knowledge, universally, can be best summarized through the philosophical poles of Wittgenstein I, and Wittgenstein II:

## Wittgenstein I

Congruent with the "picture theory" of language, thus functionalist and operationalist *pictoric*; bound to externalism and reductionism; *in atomus*, spacetime localist; widely monist and fixist; consecutive to mathematical structuralism; necessarily sensical and overly referential; endowed with a lawful language grammar; logicist; with prevalence of the transcendental subject producing *schemata* within the unity of thought over the empirical image; close to sensibility, understanding, and reason; in the line of deduction and programming languages.

## Wittgenstein II

Congruent with the "language game theory" of language, thus functionalist and operationalist *imagic*; bound to internalism and emergentism; *in continuum*, spacetime non-localist; widely pluralist and conventionalist; consecutive to mathematical formalism; modally non-sensical and overly non-referential; endowed with a rule-following grammar; constructivist; with prevalence of the empirical image over the transcendental subject producing *schemata*; close to imagination and judgment; in the line of induction and U-Mentalism.

There are some aspects in this proposed grid that are difficult to analyze and demand a close inspection. Wittgenstein's philosophy of mathematics is rarely emphasized, even though himself stated emphatically that his "chief contribution has been in the philosophy of mathematics" (Monk 1990; 446) [118]. Moreover, the subject of philosophy of mathematics itself seldom finds anchorage in programming languages theory. If semi-occasionally philosophy of mathematics topics permeate programming languages theory, scarcely ever Wittgenstein's philosophy of mathematics forms an assembly of arguments to go with the theory. To make it worse, Wittgenstein's philosophy of mathematics is divergent in time, and it happens frequently that the very same rooted arguments, are later violently uprooted. Inevitably, the *Tractatus* (1922) and *Philosophical Investigations* (published posthumously 1953) stage, respectively, Wittgenstein I and II, but the so called middle period settled by *Philosophical Remarks* (1929-1930) and *Philosophical Grammar*

(1931-33), moreover betoken to increasing complexity in the manuscripts collected in *Remarks on the Foundations of Mathematics* (1937-44), offer provocative insights.

Probably the best example of one such aspect is the confront between mathematical formalism and structuralism.

On account of the *Tractatus*[418, 419] the abundant insistence on the contingent nature of propositions (4.022, 4.25, 4.062, 2.222) reliable by "truth-by-correspondence" (purely syntactical and vaguely referential), and the demarcation of "mathematical truth" from pseudo-propositions, would ratify the sense of a "life-long formalism"[118]. On the other hand, one must hold to the assumption that not only his intellectual shift compromised its tenets – "Wittgenstein seems to become more aware of the unbearable conflict between his strong formalism (*Philosophical Grammar* (1974) 334; *Wittgenstein and the Vienna Circle* (1979)) and his denigration of set theory as a purely formal, non-mathematical calculus (Rodych 1997: 217–219)"[118] – as, beforehand, the case of having shifted to stances according to extra-mathematical operative *praxis* and sign-operative games performing mathematical calculi, indicates that his "justified side of formalism" (*Wittgenstein and the Vienna Circle*[118]), destitute of meaning was, at the end, so "strong" as to make an argument for prior structuralism, as Victor Rodych himself acknowledges in *Mathematical Sense: Wittgenstein's Syntactical Structuralism*[318]. If it was not enough by itself, Wittgenstein alluded directly to the core of structuralism in many occasions, namely in making it indissoluble from the nature of the "picture-theory" of language.

The "picture-theory" of language is, in the confront with programming languages theory, fundamental, as it is the fittest representation of – equal to Turing's, von Neumann's and Shannon's *representation*, later *objectified* monolithic integrated circuit (1958) – the (graphical) pictorially as much as (digital) signally, fabricated *idealization* of the computational and informational embedded digital circuit to which programming languages reach through code. The most simple and uncontroversial nature of one such fact is the graphic nature of the output device displaying information (from electrons to bits, from electricity to code) in pictorial form, that is the computer screen or monitor. However, we need not deal with visual programming languages and *graphically* specified programming

to understand that there is a direct bridging from *text* to *pictorial* form in the integrated circuit precedently, neither do we need to render a graphics processing unit (GPU), a digital signal processor (DSP), or indeed, an image processor (IP) to make an argument on this point. All programming languages and computation have, markedly, a Wittgenstein I "picture-theory" ontology, even if the precepts of Wittgenstein II have never been reconnoitre into the extant pieces of programming languages theory. Positively, we are affirming that all programming languages in the realm of computation are subsidiary to typical of Wittgenstein I "picture-theory" of language, in sharp contrast, therefore, with the design of  $\cup$ -Mentalism, wherein *diagrammatic* form of a (pictorial) empirical image is, furthermore, complexioned with, and in any form rendered *imagined*, a *schemata* form. Maybe it is worth to evolve this statement to a stronger assurance, by way of explanation stressing, foremost, that programming languages theory appertains, in its most profound underpinnings and, essentially, in topological form, to the "picture-theory of language" as depicted in Wittgenstein I.

Under this guise, it is made clearer how logic programming and Prolog, as a representative of a fifth-generation programming language (5GL), giving privilege to problem solving using constraints given to the program, rather than recurring to an algorithm written ahead by a programmer, in its pinnacle *declarative* status – indeed, minimally, the *Tractarian* propositions (1 to 1.21)[418, 419] could very well be Prolog's ontology description – rendering an *abstract* distance from the digital signal integrated circuit, however *topologically* and *procedurally* contiguous with the integrated circuit, has permitted researchers to contemplate innovative new approaches in computer and programming languages paradigms study. While this is true, the prospects of computability in Wittgenstein II ontology represent, in its turn, an open field of fact-finding and analysis.

Also to be noticed is how the passage from Wittgenstein I to Wittgenstein II elaborates and discloses finite constructivism, finitism, mathematical views on algorithmic decidability and an anti-foundationalist account of irrational numbers, bore on real number essentialism, alongside an harsh critic of set theory, mainly driven from both a non-dualist, strong formalist, and *practical*-actualist charting of mathematics. This difficult knot also brings about non-enumerability and non-denumerability that, inevitably, are impinged with a treatment of diagonalization

that is irreconcilable with Gödel's theorems, and from which the constructivist and actualist conflagration of intension/extension dilemmas is explored to the limit. Under this sight, the necessary forbearance of (integral) mathematics to (differential) extra operationalist *practical*-actualist applications, to which mathematics is irremediably alienated, ravages the discipline of mathematics, besides transforming philosophy of mathematics itself into a sort of sub-discipline of anthropological *inventi*, often with non-conformable *apperception* in Kantian terms.

To the aim of our investigation, we reiterate the importance of all this arguments present in the prolonged but mainly ambivalent passage from Wittgenstein I to Wittgenstein II, not only in relation to programming languages theory, but withal in relation to logic programming and Prolog.

A clear instance of that is the Viennese philosopher's intermediate to later finitism. In learning that, in the context of Wittgenstein II, "Wittgenstein rejects quantification over an infinite mathematical domain, stating that, contra his *Tractarian* view, such 'propositions' are not infinite conjunctions and infinite disjunctions simply because there are no such things." [118], in comparing to the use of both the disjunctive (DFN) and conjunctive (CNF) normal forms in logic programming and Prolog, we can eventually imagine the typical logic chicanery, formal trickery and hocus-pocus critic, polstliminary to previous existent passages (*Remarks on the Foundations of Mathematics*, 1956 [9178]: II, §10; §22; §23). On this edge, we have to recall that the intermediate Wittgenstein to Wittgenstein II firmly rejected unbounded quantification in mathematics, which would permit an extension of the critic of the (partial) idea of substitution and instantiation in logic programming and Prolog, if it would be the case of giving support to an exceptional (computational) ontology, as believed by the proponents of the Fifth Generation Computer Systems (FGCS) in Japan.

One thing interesting to note, in this respect, is how the *Tractarian* theory of truth-by-correspondence (or agreement), by means of soundness and world-related topics in use and sense, would, in its natural transformation, affect the very idea of validity. We can say that Wittgenstein II philosophy of language and mathematics, nevertheless more bounded to "rule-following" and seemingly formal-logicized, be that as it may, went exactly to the contrary direction to that of Alfred Tarski's inceptive model theory. This token is specifically problematic confronting with

logic programming and Prolog, once we consider that Prolog programs are nothing but relations defined by means of clauses. Core Prolog is restricted to Horn clauses, which is a (Turing-complete) subset of first-order predicate logic, with two types of clauses: **Facts** (the embodiment of closed-world assumption) (CWA) and **Rules** (the openness itself to open-world assumptions) (OWA), where from a **Rule** adopts the form **Head :- Body**.

What is also at stake in confronting with logic programming and Prolog, helping the reader to delimit precisely the *problemata* associated with open-world (OWA) and closed-world (CWA) assumptions, is the contra-omniscient averment by the intermediate Wittgenstein to Wittgenstein II – (not even *imaginably* God's: “Can God know all the places of the expansion of  $\pi$ ?” would have been a good question for the schoolmen to ask”, for the question is strictly ‘senseless’.” (*Philosophical Remarks* 128; also related at *Philosophical Grammar* 479)) –, suggesting a passage, thinking with Kant and Heidegger's expression "ontotheology", from an onto-*nihil*-ontology in the *Tractatus* that admitted, nevertheless, by means of the naturalness of "cases" (1-1-21; 2-2-04) of "facts" (1.1-1.2; 2.0141) and "state of affairs" (2-2.04; 4.1) a vertical *structuralist* hierarchical sense, that would sink in into a perpetual shallow evanescent and weak *formalism* in Wittgenstein II.

We consider this altercation in the core of philosophy of mathematics the proper frontier between closed-world (CWA) and open-world (OWA) assumptions that are only reflected in logic programming and Prolog. Undoubtedly, in Kantian terms, the Fifth Generation Computer Systems in Japan (FGCS), meant to work with strong AI's *ideal* of massively parallel computing – itself a natural-artificial *ideal*, *as if* the subject-matter of rational programming was taken as the ground of the subjectively necessary hypothesis for our reason proper, and the objective necessity of such a *belief*, would behold a (computational) dialectical illusion –, performed with concurrent logic programming, as a variant of the logic programming paradigm with sets of extended or guarded Horn clauses, suggests not only a clear alienation from diagonalization as computability's own method, as it indicates a sort of *analogy* with the  $P = NP$  *belief*, i.e., a (computational and informational) intricate positive fideism, in view that the broadest class of computable problems could be *a priori* verified, and *a posteriori* solved in polynomial time. In this view it is pertinent to recall that the first Prolog system was developed in 1972 [321, 79],



and the first precise formulation of the  $P$  vs.  $NP$  problem was a 1971 Stephen Cook's paper[81].

We could not go on without mentioning, at last, the most difficult and burdensome aspect of Wittgenstein's philosophy of mathematics, which assumes facets that shine on unsuspected new paradoxes, directly related with our theme of investigation. With this we are referring, mainly, to diagonalization and computability.

Wittgenstein II intermediate finitism rejects unbounded quantification, transforming mathematics into a contra-axiomatic proper recursive rule, which offers an unsuspected turn in admitting algorithmic decidability, sustaining an *epistemological* significance, in the most constricted sense of a closed-world assumption (CWA), very much opposing an actualism to any of the former Kantian structuralist, Hilbertenian-formalist, or even Fregean-logicist programs in philosophy and mathematics. Imperceptibly, what this testifies to, is probably the most original and deep-rooted attack on transcendentalism and on the entire *a priori* foundations of Kantian tradition.

Significantly, to start with, Wittgenstein's theory of formal operations and operationalism in mathematics, has, in the last decades (Frascolla 1994, 1997; Marion 1998; Potter 2000; Floyd 2002) deserved innovative focus with a shy but very profitable insight into the very same elements that programming languages theory and history commenced with. This is why Victor Rodych has pointed out that there has been an interface of "the Tractarian equational theory of arithmetic with elements of Alonzo Church's  $\lambda$ -Calculus and with R. L. Goodstein's equational calculus (Marion 1998; chapters 1, 2, and 4)."[118]. In this scrutiny and according to Rodych, the following *Tractarian*[418, 419] propositions take relevance:

**Proposition 5.2522.** The general term of the formal series  $a, O'a, O'O'a, \dots$  I write thus: " $[a, x, O'x]$ ". This expression in brackets is a variable. The first term of the expression is the beginning of the formal series, the second the form of an arbitrary term  $x$  of the series, and the third the form of that term of the series which immediately follows  $x$ .

**Proposition 6.** The general form of truth-function is:  $[\bar{p}, \bar{\xi}, N(\bar{\xi})]$ . This is the general form of proposition.

**Proposition 6.01.** The general form of the operation  $\Omega'(\bar{\eta})$  is therefore:  $[\bar{\xi}, N(\bar{\xi})]'(\bar{\eta})$  ( $= [\bar{\eta}, \bar{\xi}, N(\bar{\xi})]$ ). This is the most general form of transition from one proposition to another.

**Proposition 6.03.** The general form of the cardinal number is:  $[0, \xi, \xi + 1]$ .

Indeed, Wittgenstein combined a "theory of formal operations"[118] and the three general forms – operation, proposition, and natural number (in order of importance) – respond, in cascade, to the predominance of operationalism as a base to new results, by which successive applications is what permits, in the first place, the generation of the iterated general form that constitutes natural numbers: "The result of a logical operation is a proposition, the result of an arithmetical one is a number"( *Wittgenstein and the Vienna Circle* [381]).

Conclusively, computable numbers and computation itself (and more stringently logic programming), are in between *valid* logical inferences (seizing the Relation between possible facts) and *sound* logical inferences (seizing the Relation between necessarily existent facts), but what decisively commands truthfulness to both inputs is operationalism. Being logic programming and Prolog evident mechanized first-order calculus, and a programming *operational* ascending order from  $\lambda$ -calculus, they are an epiphenomenon, in the sense that their programming operations are, all together, the heading and indentation of arithmetic (by means of the Boolean communication intertwining programming languages in the central processor unit), and propositions (by means of the clausal form communication intertwining sentences with truth values). This is not to say that logic programming and Prolog are properly the vivid demonstration of "facts"[418, 419] or "state of affairs"[418, 419], but instead that if there was a special markedness to logic programming and Prolog, it would be much more so in blinding the deep *structuralist-to-formalist* operationalist autonomy of language, instead of being an unequivocal demonstration of the aforementioned. Inasmuch as logic programming and Prolog best deceit and camouflage operationalism and the real essence of language (by being operationalist themselves, and also bore on computational cryptographic *diagonalization*), the prospects of a programming philosophy such as  $\cup$ -Mentalism, for instance, would have to come up under the patronage of

philosophy of language, and hardly so under programming languages theory and practice.

Logic programming and Prolog, as archetypal of the philosophical and programming *declarative* paradigm and natural-to-artificial dialectic (5GL) programming languages generation and scope would, thus, be reasoned as a *synthesis* of the general form of a purely formal operation (computability), truth-functional propositions (formal logic), and true mathematical equations (arithmetic). But, following Wittgenstein's attainments, we have to look closer to this disposition, in order to fully understand how it is, very subtly, erroneous.

Henceforth, under Wittgenstein's attainment we have to conclude that, while it is evident one such *synthesis*, the principal general form of a purely formal operation is Wittgenstein I *structuralism* to Wittgenstein II *formalism* (with shares of Brower's mathematical intuitionism intersected, excepting matters such as "the role of intuition in mathematics, rule following, choice sequences, the Law of Excluded Middle, and the primacy of arithmetic over logic"[255]), to which, only after, computability, formal logic, and arithmetic respond in the overall. It is, consequently, the closest analogous relation between operationalism and computability, in the first place, what illudes our judgment, while it is strikingly remarkable how, out of the three different classes – computability, formal logic and arithmetic (in ascending naturalness) – it is, precisely, computability, via operationalism, the most identical to truthfulness of "facts"[418, 419] and "state of affairs"[418, 419].

With this it should be clear how Wittgenstein, by understanding logic as the manifestation of the fact that a proposition is always identical to itself, even if bivalent (assuming the truth values of truth and falsity), operating a division between meaningful propositions and significant references, in the precise opposite direction to that of Fregean and Russellian logicist monism, permits us to reflect on and understand, mainly, two completions on logic programming and Prolog.

First and unavoidably, logic programming and Prolog partake in the sort of the herein characterized Fregean and Russellian logicist ontology, and second, that logic programming in Prolog, as adhered to natural language processing, also share, by means, not so much of its "operational" mechanization and fabrication, but instead of the inscribed philosophy of logic, science, language, and technology in its idealization and aspiration, the very same fatal error that was disclosed in a letter

from Russell to Frege "in the spring of 1901 with the discovery of the paradox of the class of all classes not members of themselves"[234]. In a sense, Wittgenstein II lessons and the last turn of the linguistic turn ("meaning is use"; "language-games"; family resemblance"; "rule following"; "forms of life"; "quietism", etc.) have all been debunked, with the result of structuralism having re-endorsed logicism. In a way, logic programming and Prolog were caught up in this whirlpool, having suffered from a new structuralist wave in linguistics, although much more deflated of logicism, ever since Chomsky's *Syntactic Structures* (1951). What is most curious is how, not only Chomsky's generative grammar idea, but also logic programming and Prolog, astricted as they were to *structuralism*, contained, however, the seeds to their inversion (namely, the defeasance of the Fifth Generation Computer Systems (FGCS) in Japan, and both the evolution of the transformational generative grammar to non-structuralist neither syntactic stances, and the inductive and statistical revolution in natural language processing driven by machine learning), all in all determinable events that took place in mid-to-late 1980's.

Indeed, remembering that the "Tractatus was inspired by Wittgenstein's study of the philosopher-physicists Heinrich Hertz and Ludwig Boltzmannz"[157] if so, when we see Wittgenstein II rescued in relation to the interpretation problem, is generally under the auspices of philosophy of science hard problems, or when not, within the constructed values that permeate the mind from physics to culture (Wittgenstein' *Nachlass* approach similar to Nietzsche's), but hardly so related with the closest impacted problems in philosophy of language or linguistics in the technological age. These are in demand of a sufficient treatment of language and computability *summa problemata*, which, in our interpretation, is also a sign of repetition of Kant's philosophy criticism forgetfulness pattern towards philosophy of language that motivated *Mittleuropa*'s linguistic turn, in the first place.

Ahead, matters that connect Wittgenstein II actualist and finitistic formalism, according to which the extension-intension conflation is just a sign of the mindful faulty, but naturalistic necessary model, with questions of decidability and even the envisagement of mathematical number theory classes – such as irrational numbers –, as "rules", instead of classes, are all aspects in philosophy of mathematics that are worth facing up with logic programming and Prolog. Once we understand that formal logic calculi with finite extensions and intensional rules, such as

in logic programming and Prolog, are criticizable under the guise of Wittgenstein II, and that, *a fortiori*, computability, endlessly yielding places of decimal fraction in diagonalization, is, oddly enough, more transparent of the constructivist nature of "facts"[418, 419] than mathematics' own *practicality* idealization, and that Wittgenstein II was also, other than this and non-contradictorily, very critical of Cantor's and Gödel's diagonal proofs, understanding the proper nature of proofs as a rhetoric appendix, as for the rest, set theory and non-denumerability, it is, at heart, very problematic to stage ways out of philosophy of language, computation, and mathematics new riddles. For the moment, it suffices to expose these *problemata*, and, minimally, attain to some sort of a new scheme, as  $\cup$ -Mentalism, either as an *exitus*, or a *resolutio*.

A Horn clause is, basically, a rule-like form (in between "formal calculus" and the "language-game") that corresponds to a disjunction of literals (an atomic formula or its negation, in line with Wittgenstein's I logical atomism, and Wittgenstein II logical formalism) with at most one positive, i.e. unnegated, literal. They are of the sort[326, 393, 188, 73] (with  $n > 0$  and  $L$  being the only positive literal):

$$L_1 \dots, L_n \Rightarrow L (\equiv \neg L_1 \vee \dots \vee \neg L_n \vee L)$$

$$L_1 \dots, L_n \Rightarrow L (\equiv \neg L_1 \vee \dots \vee \neg L_n)$$

$$L_1, \dots, L_n \Rightarrow L (\equiv (L_1 \vee L_2 \vee L_3 \dots \vee \neg L_n))$$

$$L_1, \dots, L_n \equiv L (\neg L_1 \vee \neg L_2 \vee \neg L_3 \dots \vee L_n)$$

In case it happens to have exactly one positive literal, then we name it a definite clause. In case it has no negative literals we are at presence of a **Fact**. In case we have a Horn clause without a positive literal, we are at presence of a **Goal**. If it happens that the we have a disjunction of literals with at most one negated literal it is, conversely, named a dual-Horn clause. The simplest direct example of one such translation in Prolog's (computational and programming) dialect is:

**s** :- **p**, **q**, **r**.

Anticipating that the resolution algorithm works by using the principle of proof by contradiction, being an empty clause equivalent to a "disjunction of no disjuncts" [325], and, consequently, the *local* and *atomic* emergence of a literal  $\alpha$  being the positive resolution of two complementary unit clauses, in extreme Kant's antinomies – there are no new clauses, and therefore our *knowledge base* does not entail  $\alpha$ ; or else two clauses resolve to hold an empty clause, and therefore our *knowledge base* entails  $\alpha$  – then it is, by these means, found a fully *contra-transcendental*, logic-inductive, propositional resolution making sense of the fact that any clause in which two complementary literals are shown can be outmoded.

This is equivalent to affirming that clauses (as disjunctions of literals) are a pivot for the logical truth beholding the conjunctive normal form (CNF), i.e., the fact that every sentence in sentential logic corresponds to a conjunction of clauses, making an opening for the resolution closure – in  $S$  the set of all clauses derivable by repeated application of the resolution rule either in  $S$  or its derivatives (in extreme, marrying *transcendental* antinomies with recursion, and divorcing transcendental philosophy from *sentential* language, in respect to both the *acumen* of Wittgenstein I silence and Wittgenstein II impossibility of a private language) – and, at last, also an opening for the ground resolution theorem. With this we mean, in inversion to the preliminary, understanding that if  $S$ , as a set of clauses, is unsatisfiable, then the resolution closure all in all abridged in  $S$ , *contains* the empty clause – which redirects, in extreme, Kantian *transcendental dialectic* antinomies to non-analyticity from within the proper bounded limits of *transcendental analyticity*, in paradoxical form. Something as simple as the conjunctive normal form (CNF) and the disjunctive normal form (DNF) can illustrate the Kantian *pure* logic dilemmatic use and, quite extraordinarily, in *actual* operative diagonalization, an outcoming *pure resolution*.

Importantly and again, as in logic programming and Prolog, on top of this "state of affairs"[418, 419] computation (in proper diagonalization) in **Queries, Goals, and Facts**, even if within asymptotic computational complexity –  $\mathcal{O}(n \log n)$  – has a limit in first-order calculus (FOL), being asymptotic-closed (FOL $\asymp$ closed) as much as it is diagonalization-opened (FOL $\nparallel$ opened), there is sort of an obvious effect of analyticity in diagonalization ( $\nparallel$ ) with synthetic and dialectic arguments (*NP*-complete, typical of logic programming, Prolog and Wittgenstein I, as for

the rest alienated from  $NP$ -hard problems, typical of Wittgenstein II), but more truthfully so, also of analyticity in diagonalization ( $\#$ ) towards itself, as if logic programming and Prolog at the heart, considered their mechanism as a Cantorian and Gödelian argument, would automatically disprove first-order logic (FOL) in the compiler and abstract machine. But the paradoxical stand does not stop here, and it would be *correct* (except for surmounting enumerable paradoxes) to affirm that they correspond to higher-level paradoxes from those of Kantian antinomies, resorting to  $O(n^n)$  computational (time and space) complexity, ahead still of *synthetic* and *dialectic* physics and philosophy of science *natura naturans* (spacetime) complexity.

Now, this is truly prevailing, once "Stephen Cook (1971) showed that deciding satisfiability of a sentence in propositional logic (the SAT problem) is  $NP$ -complete. Since deciding entailment is equivalent to deciding unsatisfiability, it is co- $NP$ -complete. Many subsets of propositional logic are known for which the satisfiability problem is polynomially solvable; Horn clause are one such subset." [325] Horn clauses, in one such way, are demonstrable but controvertible, recursive complete rules, in the form of conjunctions of positive literals as a premise [**Body**] of a single positive literal [**Head**] under the broader implication form. The fact that all this ensemble assumes, in the [**Head**] and [**Body**] implication clausal form, as much a programming informatic as a natural biological *structure* contemplating many possible *forms*, is also something to wonder.

With this we find that first-order logic, inasmuch as a **Fact** in logic programming and Prolog, respond simultaneously to  $True \rightarrow L_{1,1}$  or  $L_{1,1}$  in any given case for  $\alpha$  as a new postulate  $L_{1,\alpha}$  under any new knowledge base assumption through the use of resolution and unification – like so refraining closed-world assumption (CWA, related with  $NP$ -completeness, and Wittgenstein I) with manifest open-world assumption (OWA, related with  $NP$ -Hard and  $P$  vs.  $NP$  problems, and Wittgenstein II) – finds in logic programming and Prolog the programming *apex* as the proper *resolution* of the greatest lower bound (*procedural* and *imperative*, congruent with the C language) with the least upper bound (*declarative* and *logic*, congruent with Prolog) artificial-to-natural computational and programming anti-nomy in the information age.

Because in the process of unification and substitution there is not but attention to the atomist literals in relation to the logical truth model-theory values –  $True \rightarrow L_{1,1}$  or  $False \rightarrow L_{-1,1}$  – the proper unification and substitution as examples of a "general theory of operations"[118] (Frascolla 1998: 135) mechanism, is all too forgotten. In like manner in physics spacetime relativity requested non-linearity and complexioned arguments with a perilous relation in respect to cause-effect, inference with local and atomic Horn clauses is linearly open to both forward-chaining and backward-chaining algorithms, which obscures the natural-to-artificial logic and mathematics philosophical *problemata* associated with Wittgenstein II.

In Backus-Naur form (BNF), a description for a conjunctive normal form (CNF), triggering, thus, general Horn clauses, and the more restricted definite clauses, is given below[325, 326, 393]:

$$\begin{aligned}
\langle CNFSentence \rangle &\Rightarrow Clause_1 \dots \wedge Clause_n \\
| \langle Clause \rangle &\Rightarrow Literal_1 \vee \dots \vee Literal_m \\
| \langle Literal \rangle &\Rightarrow Symbol \mid \neg Symbol \\
| \langle Symbol \rangle &\Rightarrow P \mid Q \mid R \dots \\
| \langle HornClauseForm \rangle &\Rightarrow DefiniteClauseForm \mid GoalClauseForm \\
| \langle DefiniteClauseForm \rangle &\Rightarrow (Symbol_1 \wedge \dots \wedge Symbol_l) \Rightarrow Symbol \\
| \langle GoalClauseForm \rangle &\Rightarrow (Symbol_1 \wedge \dots \wedge Symbol_l) \Rightarrow False
\end{aligned}$$

– and having, consequently, cleared Horn clauses (1951) *modus operandi* or *logicaliter* (in both Wittgenstein I and Wittgenstein II sense and meaning), it is now pertinent to minimally understand all together the Robinson unification algorithm (1965) – built on the Davis-Putnam resolution (1960) – and the proper SLD-resolution (1972-1974) by Robert Kowalski – built on SL-resolution (1970-1971) – with this closing the circle, insofar SLD-resolution (1972-1974), a programming refinement of resolution, is nothing but the basic inference rule used in logic programming, which is both sound and refutation complete for the specific definite clauses  $\{Q\}$  or  $\{\neg P_1, \dots, \neg P_m, Q\}$  appertaining to Horn clauses.

Analyzing all in once SLD-resolution it is made obvious logic programming and Prolog's embedded core at work, and how it is fundamentally connected with a sort



of constricted equivalent  $P \equiv FOL$  analytic (very much in line with Wittgenstein I, and also restrictive to both *synthetic* and *dialectic* judgements), in the sense that "it reduces the complexity of proving the correctness of programs." [140] It is in this very same sense that, alluding to computation and deduction, Frank Pfenning wrote: "Philosophers, mathematicians, and computer scientists have tried to unify the two, or at least to understand the relationship between them for centuries. For example, George Boole succeeded in reducing a certain class of logical reasoning to computation in so-called Boolean algebras. Since the fundamental undecidability results of the 20<sup>th</sup> century we know that not everything we can reason about is in fact mechanically computable, even if we follow a well-defined set of formal rules." [298]

If any set  $S$  of clauses consists of definite clauses except for one goal or else called negative clause, a Horn clause of the form  $\{\neg P_1, \dots, \neg P_m\}$  "then there is a proof having the property that whenever a  $\vee : left$  rule is applied to a definite clause  $\{\neg P_1, \dots, \neg P_m, Q\}$ , the rule splits it into  $\{\neg P_1, \dots, \neg P_m\}$  and  $\{Q\}$ , the sequent containing  $\{Q\}$  is an axiom, and the sequent containing  $\{\neg P_1, \dots, \neg P_m\}$  does not contain  $\{\neg Q\}$ ." [140]

Therefore, what happens is that for each  $\vee : left$  rule, after isolating the Goal clause, the rule is applied to split each definite clause after the selected goal, until the final resolvent is the sentential most general unifier. A typical proof by resolution exhausts all possible substitutions, as long as goal and clause do have, in principle, variables in common (a substitution is a finite set  $\{t_1/v_1, \dots, t_n/v_n\}$  where every  $v_i$  is a variable and every  $t_i$  is a term, possibly anew from the previous, while it is not permitted that two elements in the set have the same variable after the "/" symbol).

Moreover, all that is needed for the least Herbrand model – a Herbrand model of a first-order (FOL) language has a Herbrand set of ground *terms* (containing no variables or quantifiers) as its universe of discourse, meaning that the interpretation of *terms* is canonical, resounds to the following: "terms evaluate to themselves - but there is no requirement on the interpretation of relations" [392] which lends poignancy to the ambivalent status of logic programming and Prolog. Not only Prolog is a programming "relational language" [73] while logic programming and SLD-resolution consist of one only rule with exactly one head atom (*structuralist*,

Wittgenstein I), as also the aim of natural language processing in Prolog – "The Prolog system (for PROgramming in LOGic), based upon the procedural interpretation, has been used for several ambitious programming tasks (including French language question answering, symbolic integration, plan formation, theorem proving, speech recognition, and picture interpretation"[125] – has one universe of discourse that is intended to reduce on and test, *in extremis*, for the set of elements of which all possible variables range over (*formalist*, Wittgenstein II).

SLD-resolution (SL-resolution for definite clauses; **S**election function with **L**inear resolution for **D**efinite clauses), whenever meeting a **P**rogram (with clauses or variants of clauses) and **G**oal (a finite or infinite sequence) in reunion

$$Program \cup \{Goal_0, Goal_1, \dots, Goal_n\}$$

will build in a sequence  $\theta_1, \theta_2, \dots$  of most general unifiers in line, in such way as to unify  $Goal_{i+1}$  as the resolvent of both  $Goal_i$  and **P**rogram's clauses  $P_{i+1}$  through the resolution method  $\theta_{i+1}$ . At the end, if a finite SLD-resolution of the above meets the empty clause  $\emptyset$  in coincidence with the last  $Goal_n$  we have also met an SLD-refutation of length  $n$ , after each atom/ $Goal_{i+1}$  (Wittgenstein I, actualist *atomism* and *structuralism*) for each step in the selection (**S**election in "SLD-resolution"[140]). As a matter of fact, what is so surprising in SLD-resolution[140] is its adaptive, truly naturalistic imprint, and the superb artificial-to-natural, almost *synthetic* plasticity, after John Alan Robinson's study of first-order syntactical unification (1965) as the solution approach to FOL's resolution procedure, unequivocally the *declarative* paradigm full comprehensiveness outreach (Wittgenstein I, *P*-like decidability) in automated reasoning, constricting to the utmost the combinatorial explosion in searching for instantiation of terms.

This is so to the point of easily commuting with a sort of evolutionism in logic programming, in the sense that the syntactic unification and the solution set envisaged as an (adaptive) new element, likewise generalization and specialization (*genus et differentia*) from simple substitution, all shape its algorithm. What is more, resolution is applied on having found complementary literals, which is akin to Wittgenstein I *atomist* and *structuralist* interpretation, yet also to Wittgenstein's typical logical "bipolar"[144] interpretation –  $(p)(p \vee \sim p)$  (*Letters to Russell* 5.9.13; *Notes on Logic* [1913] in *Notebooks 1914-16* 94-9, 104; *Notes dictated to*

*G. E. Moore in Norway*[1914] in *Notebooks 1914-16* 113) – which redirects to the subsoil of formal logic, again a stance that is more naturalistic "morphological" – "What I offer is the morphology of the use of an expression." [252], again attesting for Wittgenstein "the morphologist"[228, 299] – than it is logical, little less attainable to logic programming or any of its nuclear procedures. Or, inversely, it would be Wittgenstein's focus on "theory of formal operations"[118], concomitant with computation, and thereafter logic programming and Prolog, what would explain the natural optimality of the unification algorithm.

What is important to retain is that a finite SLD-resolution fails in case it does not meet an empty clause  $\emptyset$  coinciding with the last **Goal**, once if a non-empty **Goal** and the selected atom do not unify with the head of any clause of the **Program**, there is neither soundness nor completeness for SLD-refutation. The interesting thing in SLD-refutation is that soundness is made possible to refute because of refutation-incompleteness, and completeness is effectively refuted because of refutation-unsoundness.

Originally, the clause at the root of resolution, before the selection initial sequence  $\neg L_1 \vee \dots \vee \neg L_i \vee \dots \vee \neg L_n$ , is already a sign of the *atomist* interpretation, by isolating its literals, meaning also that we are *ideally* "denying" (Wittgenstein I) the conjunction of subgoals  $L_1 \wedge \dots \wedge L_i \wedge \dots \wedge L_n$  and constricting a conjunctive normal form, i.e., a conjunction of disjunctions of literals (CNF), to a disjunctive normal form, i.e., a disjunction of conjunctions of literals (DNF). This shifts the effect (bore on the fact that a logical truth in a conjunction depends on equal truth values for all the literals, while the logical truth in a disjunction is made secure, as long as there is one valid element in one literal) and, therefore, we conclude that *ideal* completeness – whenever  $S$  is contradictory, there exists a resolution proof – in this shift, changes the onus of the proof to soundness – whenever a resolution refutation is found from a set of clauses  $S$ , then  $S$  is contradictory – trying in each selection to diminish the path to one head clause only, *ideally* an *atom*.

Expressly, we find that much of logic programming and Prolog's signature, namely Robert Kowalski's work in SLD-resolution, is firmed on the constraint passage from the *ideally* refutation-complete *NP*-like problem, grammarly gerundive, patent in the conjunctive normal form (CNF), to the *practical* refutation-sound *P*-like problem, grammarly infinitive, patent in the disjunctive normal form (DNF).

On this wise, it is contradictory that logic programming and Prolog have been used in such *dialectic* form as in the Fifth Generation Computer Systems (FGCS) in Japan, against its natural *analytic*. But it is also fair to point out that, indeed, computer science is, in the first place, a very ambivalent and paradoxical result of a *dialectic* and programming constraints *pure reason illusion*: there could not be any better example of this than the fact that the modern computer von Neumann (brain) model and Princeton architecture, stemming from a profound *naturalistic* cellular automata, DNA, and universal constructor philosophy, is, at the end, in *schemata* form, the proper inversion of the mind and body.

This is not the same as confusing *substantia* and *methodus*, i.e., neither do we say that  $\cup$ -Mentalism – a made available *naming* for the realizable inversion, in *schemata* form, of the von Neumann computer model and Princeton architecture – if made possible and verifiable, would, in its *ulterior* form, recur to typical body-mind naturalism and not machine learning and computer’s natural operationalism, nor would we say, prior to this, that it not necessary and valid in its *proximus* form, for the reason that Boolean digital logic is *photon*-identical to the minimal extant idea of  $\cup$ -Mentalism, and, except for the decisive difference of *sensation, perception and sensitive intelligence* – the exact same burden that afflicted modern *idealist* philosophers such as Berkeley, Leibniz and Kant – man, independently of the mind-body problem, is already an image and language processing learning (informational in Shannon’s sense, cybernetics in Wiener’s sense, and operationalist in Wittgenstein’s sense) *automata*.

There is also some sort of admissible inherent *formalism-to-structuralism* in programming languages, insofar it is not imagined to be possible, out of soundness limited and weak conjunctive normal form (CNF) or soundness unrestricted and strong disjunctive normal form (DNF), the latter to be applied to different lists of different programming languages. The topological distance between typed programming languages partial computable functions attests, in essence, inasmuch for local observance determinism and *quanta* indeterminacy, as it attests for *quanta* determinism and local indeterminacy.

With this being said, it is relevant to underline that, before SLD-resolution will try to deny the literal/**Head** clause/**Goal** to show unstatifiability, the unifying substitution  $\theta$ , in the core of computability’s *diagonalization* (in proper *procedural*

terms), will, in *parallel* (in proper *declarative terms*), but essentially repeating in *procedural terms diagonalization* (both actions being very nitid in SLD-Resolution proofs), both pass the input from the selected subgoal to the body of the procedure (by the use of  $\neg$  or *reductio ad absurdum*, diagonally), and also pass the output from the head of the procedure to the remaining unselected goals (by the use of  $\vee$ , parallelly), in tree-like (convergent and divergent, parallel and diagonal) philo-genetic style (logic programming in Prolog works backwards from the conjecture, building the proof-tree upwards, else called a **Goal-directed** or *top-down* approach) for each

$$\frac{\text{Premise or Conjecture}}{\text{Inference, Theorem or Conclusion}} \quad \text{Prolog Syntax as Rules of Inference [140, 298]:}$$

$$\frac{\Gamma \vdash A_1, \dots, A_m \quad \Gamma, B \rightarrow}{\Gamma, (A_1 \vee B), \dots, (A_m \vee B) \rightarrow}$$

The line of the equation corresponds simultaneously to the Prolog syntax and rules of inference on one side, and, on the other, to the proper demarcation line in science and programming philosophy as a *rationale* quotient, delimiting, thus, a conjecture from the axioms. It is in this sense that Ivan Bratko wrote: "An appropriate view of the interpretation of a Prolog program in mathematical terms is then as follows: Prolog accepts facts and rules as a set of axioms, and the user's question as a *conjectured theorem*; then it tries to prove this theorem – that is, to demonstrate that it can be logically derived from the axioms." [46]. Purposely, also Leon Sterling has "emphasized throughout the distinction between logic programming and Prolog programming" [337]:

"Logic programs can be understood and studied, using two abstract, machine-independent concepts: truth and logical deduction. One can ask whether an axiom in a program is true, under some interpretation of the program symbols; or whether a logical statement is a consequence of the program. These questions can be answered independently of any concrete execution mechanism.

On the contrary, Prolog is a programming language, borrowing its basic constructs from logic. Prolog programs have precise operational meaning: they are instructions for execution on a computer – a Prolog

machine. Prolog programs in good style can almost always be read as logical statements, thus inheriting some of the abstract properties of logic programs. Most important, the result of a computation of such a Prolog program is a logical consequence of the axioms in it. Effective Prolog programming requires an understanding of the theory of logic programming." [337]

Framing in Wittgenstein's philosophy of mathematics, it was made obvious already how the abstract machine/interpreter is closer to a "theory of formal operations" [118], while, however reasonable it would be to isolate machine-independent concepts, such as "truth and logical deduction" [337], correspondingly, the former should be considered equally set in and congruent with a "theory of formal operations" [118]. Withal seemingly independent, "truth and logical deduction" [337] are posterior to the existence of "facts" [418, 419], and nonetheless abstract machines and computational interpreters are seemingly artificial and posterior to "truth and logical deduction" [337], these concepts are just a result of a "theory of formal operations" [118] as it is the former.

As for Robert Kowalski [224, 223, 125] – the main contributor, in automated theorem proving, to the development of logic programming, precisely from the *procedural* interpretation of Horn clauses [326, 393] – his view was established, mainly, around the following conclusion:

"It is concluded that operational semantics is part of proof theory and that fixpoint semantics is a special case of model-theoretic semantics" [125].

Robert Kowalski goes further, and defends that operational semantics – defining the individual input-output operations evoked by the program in the machine, "included in the syntax" [125] – and fixpoint semantics – defining the meaning of a program as the minimal fixpoint transformation associated with the program in the machine, "included in the semantics" [125] – besides addressing a sort of topologically-*transformational* analysis to predicate logic as a programming language, designating the proper minimal distance between an (artificial) interpreter and a (natural) interpretation, endorses yet another idea:

"With this interpretation of operational semantics as syntax and fix-point semantics as semantics, the equivalence of operational and fix-point semantics becomes a special case of Gödel's completeness theorem." [125]

A syntax of well-formed formulas, forming sentences as finite set of clauses  $L_1 \vee \dots \vee L_n$  accounting for atomic formulae  $P(t_1, \dots, t_m)$  and where  $P$  is a predicate symbol and  $t_i$  are terms, connect, by the manifold existence of terms – a variable, an expression  $f(t_1, \dots, t_m)$  where  $f$  is a function symbol and  $t_i$  are terms, and constants as 0-ary symbols – with the field of clauses  $\{C_1, \dots, C_n\}$ . In this pivotal point, not only is brought in the axiomatic, atomist, Wittgenstein I and Kowalski's operational semantics as syntax, as it is also in check the so called "procedural interpretation" [125] – with either an empty clause, a clause consisting of no positive literals  $\bar{B}_1 \vee \dots \vee \bar{B}_n$  and  $n \geq 1$  literals, or a clause consisting of one exactly positive literal  $A \vee \bar{B}_1 \vee \dots \vee \bar{B}_n$  where  $n \geq 0$  – which hinges on an almost morphogenetic process of constitution of the "*procedure declaration*" [125], meaning with this that genetically – "all questions concerning the validity or satisfiability of sentences in predicate logic can be addressed to sentences in clausal form" [125] – ascends a constitution of a "*procedure body*" operated by the negative literals, to the point that the "*procedure invocation*" [125] that is resolution [140, 316] assumes almost a genetic helical form, as each goal-oriented derivation can be regarded as successive (computable, in diagonalization) terminating refutations, however still contained in the same (logical) axis.

With this, however determined by the last in/first out rule and the mechanism of SLD-resolution [140, 223], predicate logic is a nondeterministic programming language, while operational semantics connects the sets of clauses with an Herbrand universe wherein predicate symbols in  $n$ -ary relations build  $n$ -tuples input-output Relations. And "operational semantics" [125] – indeed, a R. A. Kowalski's terminology that could very well be Wittgenstein's II – assures, by means of a unique denotation for each derivation of the  $n$ -tuple  $(t_1, \dots, t_m)$  in the derivation-complete and refutation-complete Herbrand universe, substituting procedure calls by procedure bodies, the existence of an independent interpreter and abstract machine that is, simultaneously, an inference system, a proof procedure, and

computable Relations. It has to be said that constraints on the inference system, proof procedure, and computable Relations are just as sound, in the sense of model-theoretic fixpoint semantics mutually recursive procedure declarations [327, 54, 223], in full monotonicity of the Herbrand universe expressions, as they are unjust and unsound, in the sense that abduction in Prolog and the existence of *abducible* predicates, is naturally limited to linear assumptions due to the proper nature of constraint rules [70, 359].

Generally, the fact that Prolog is, *declaratively*, a logicist Wittgenstein I programming language, and the fact that it is, *procedurally*, a fully operational Wittgenstein II abstract machine and interpreter [310, 91, 274, 415], veritably conducts our *synthetic* understanding towards a *naturalistic* stance close to Von Neumann's towards the powers of computation, in admitting that logic programming and Prolog mimic a double helix (operational semantics and proof theory in programming languages and syntax in Prolog *vs.* fixpoint and model-theoretic semantics in programming languages, and semantics in Prolog) [223, 125] of two complementary strands (the leading copy, forward-chaining and front-tracking, **Facts**-driven, axioms following and CWA, *Modus Ponens*, and cleavage resistant *vs.* the leading correction, backward-chaining and back-tracking, **Goal**-directed, rule following and OWA, *Modus Tollens*, and strands-cutting), therefore, with dominant and recessive traits in artificial-to-natural field of partial computable functions [181]. But more importantly, because of the referred conflagration of reproductive Euclidean paralleling axiomatic and methodology (systematic philosophy, contrasted with *P*-like decidability) and non-Euclidean productive diagonalization (computation model, contrasted with *NP*-like complexity), what is mostly unnoticed is how logic programming and Prolog offer a great insight – far-ranging wider than Kowalski's decipherment: "With this interpretation of operational semantics as syntax and fixpoint semantics as semantics, the equivalence of operational and fixpoint semantics becomes a special case of Gödel's completeness theorem" [125] – to the showcase that it becomes also, and more fundamentally, a special case of Gödel's incompleteness theorems [216, 312, 160, 161, 162, 163], inasmuch as the original set of clauses  $\{C_1, \dots, C_n\}$  is observed to be universally quantified with the variables  $\forall\{\iota, \dots, x_m\}C$ .



If we choose to start SWI-Prolog on Windows, what follows is the creation of a directory folder, generally called **swipl** containing the executables, libraries, etc., of the system, with no other files installed outside this directory. The general path obeys to <appdata>/SWI-Prolog. Any executable program **swipl-win.exe**, starts an interactive Prolog window. Therefore, any executable program **swipl.exe** is already a sign of a version of SWI-Prolog running in a console window. The file extension **.pl** is, thus, inevitably also a sign of the program **swipl-win.exe**. Consequently, if we click and open a **.pl** file, **swipl-win.exe** will start, changing directory to the directory in which the file to open is, and load it instantaneously.

In terms of initialization files associated with the user (2.2 [403]) – on MS-Windows, it is the file **swipl.ini** and on Unix systems **.swiplrc**. – and once we have file programs that one can consult – the predicate **consult/1** gives the same result as **load\_files(File, [ ])**, except for directly handling the file user, reading clauses from the terminal<sup>17</sup> –the program database is made reachable. One of its built-in predicates is **halt/0** under the user top-level manipulation, which terminates any Prolog execution. Right from the start, first-order logic decidability under query type inferences in Horn clauses, is, thus, very close with undecidability and typical Turing-Church-Gödel-Davis halting problem, here in trial:

```

1 Welcome to SWI-Prolog (threaded, 64 bits, version 7.4.2)
2 SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
3 Please run ?- license. for legal details.
4
5 For online help and background, visit http://www.swi-prolog.org
6 For built-in help, use ?- help(Topic). or ?- apropos(Word).
7
8 1 ?-halt.
```

<sup>17</sup>The built-in predicate **consult(:File)** reads *File* as a Prolog source file. It constitutes, therefore, a call to **consult/1**. Examples are[403]:

```

?- consult(load).
?- [library(lists)].
?- [user].
```

In any case, a file or a group of files are loaded, whether that might be a **.pl** file, library lists, or even, more directly, an end-user program on the terminal.

Redirecting now analysis to the Prolog syntax, every first learning regards data objects. First and foremost, "the central data structure in Prolog is that of a *term*. There are terms of four kinds: *atoms*, *numbers*, *variables*, and *compound terms*. Atoms and numbers are sometimes grouped together and called *atomic terms*"(Listing 1):

```
elephant, b, abcXYZ, x_123, another_pint_for_me_please,  
  
'This is also a Prolog atom.'  
  
+, ::, <----->, ***
```

Listing 1: 1.2 Prolog Syntax *in Lecture Notes, an Introduction to Prolog Programming* by Ulle Endriss

Ensuingly, numbers are integer type Prolog implementations as of the sort (... - 3, -2, -1, 0, 1, 2, 3, 4...) [106, 387, 107, 75], optionally preceded by a - (minus), while floating is admitted. As for the case of boundedness with a low-level implementation, regarding non-integers, "SWI-Prolog internally represents floats using the C-language type double. On most today systems this implies using a 64-bit IEEE representation. All floating point math functions are based on the C math-library." [403, 232] Richard A. O.'Keefe, indeed, explicits that "SWI Prolog uses exactly the same floating point arithmetic as C. This is normally provided by your hardware, and can be expected to conform to the IEEE 754 standard. This is BINARY FLOATING-POINT arithmetic, not arithmetic on the mathematical real numbers." [214, 403, 232], which, naturally, makes Prolog's arithmetic *declarative* (Listing 2):

As for variables in Prolog, they are strings of letters, digits and underscore '(\_)' characters. They unalterably start with an uppercase letter or an underscore (Listing 3):

Anonymous variables, written with a single underscore ' \_ ', can be manipulated accordingly. While

```
1 has_a_child (X) :- parent (X,Y).
```

```

8 is 6+2.
12 is 6*2.
4 is 6-2.
-2 is 6-8.
3 is 6/2.
3 is 7/2.
1 is mod(7,2).

```

Listing 2: 5.1 Arithmetic in Prolog *in Learn Prolog Now!* by Patrick Blackburn, Johan Bos, and Kristina Striegnitz

```

X
Result
Object2
Participant_list
ShoppingList
_A
_x23
_23

```

Listing 3: 2.1.2 Variables *in Prolog Programming for Artificial Intelligence* by Ivan Bratko

... makes explicit that the atomist and elemental X relates to the atomist and elemental Y, i. e., whichever the specific variable X and the specific variable Y independently might represent, what is invariably true is that being the variable X *that* atomic specific X and being the variable Y *that* atomic specific Y (as in a direct mapping in Wittgenstein I logicist *structuralism*, injective surjective function, or else said bijective function) their relation holds and secures a valid inference, established between **parent (X, Y)** and **has\_a\_child (X)**. This is different from the following:

```

1 has_a_child(X) :- parent (X,_).

```

Here what is explicit is that whichever and any variable *in place* being unified into the relation, it can also be substituted for any other it might be (as in a

direct mapping in Wittgenstein II non-logicist *formalism*, non-injective surjective function, not a bijection), their relation will still hold a valid inference. It is true that logically, the difference does not exist, as validity would fall to proof, whichever **parent (X, Y)** or **parent (X, \_)** might hold, but what is substantially different are the extreme poles, that is, low-level computer-based *procedures*, and high-level human-agent *philosophy*. Sharply, Ivan Bratko goes straight to the point, bringing up the proper intersubjective, but only subjective *position* in a Cartesian plane with  $x - y$  coordinates, for both the observer and the visible (here adapting the original example suitable to a camera) [46]:

```
1 visible(Object) :- see(Observer,_,_).
```

... which is *logically* and *geometrically-topologically* equivalent to:

```
1 visible(Object) :- see(Observer,X,Y).
```

... and, however and actually, quite different from:

```
1 visible(Object) :- see (Observer,X,X).
```

Also the author adds: "The *lexical scope* of variable names is one clause. This means that, for example, if the name **X15** occurs in two clauses, then it signifies two different variables. But each occurrence of **X15** within the same clause means the same variable. The situation is different for constants: the same atom always means the same object in any clause, throughout the whole program." [46]

It is possible, from the command line and the use of `-dump-runtime-variables.`, to print a sequence of variables that can be used in Unix-like shell-scripts, to deal with Prolog parameters, and it is fundamental to understand that they are not *vacuum* placeholders, but instead strict numerical routines. Variables are just bound along the way of satisfying goals and subgoals, and there is even some economy in not reassigning value for variables, remembering Sterling & Shapiro note that "*variables are a means of summarizing many queries*" [337]. Indeed, either in the *term-stack* or *heap* (assigned to compound terms and large numbers), in the *environment-stack* (assigned to choice-points), and the *trail-stack* (where assignments run forwardly, and backtracking is possible) [403, 232], the general control of stack-sizes is not even minimally affected by variables, even taking into account that "when Prolog uses a variable, the variable can be either *instantiated* or *not instantiated*" [73].

We justly apprehend from Clocksin & Melish (1981-1994) the following: "Computer programming in Prolog consists of: declaring some *facts* about objects and their relationships, defining some *rules* about objects and their relationships, and asking *questions* about objects and their relationships." [73] A straightforward examples is (Listing 4):

```
valuable(gold).      % Gold is valuable.
female(jane).       % Jane is female.
owns(john,gold).    % John owns gold.
father(john,mary).  % John is father of Mary.
gives(john,book,mary). % John gives the book to Mary.
```

Listing 4: 1.1 Facts *in Programming in Prolog* by W. F. Clocksin & Mellish

When it comes to Prolog satisfying Goals, one thing is important: "You must bear in mind when you write Prolog programs how Prolog searches through the database and what variables will be instantiated when one of your rules is used." [73], which has a mirroring explanation in Bratko: "What *is* unusual in Prolog, in comparison with other languages, is that a Prolog program may be declaratively correct, but at the same time be procedurally incorrect in that it is not able to produce an answer to a question although the answer exists." [46]

In having absorbed that Prolog programs are built from *terms* – *constant*, *variable*, or a *structure* – and that any *structure*, else called a *compound term*, is written in Prolog by specifying a *functor* and its *components*, we have, quite inadvertently, found a sort of a programming base *infimum* and logic programming greatest lower bound, and a programming exponentiation *supremum* and logic programming least upper bound, in the sense that infinite redundancy and looping on one side, and sharp recursion on the other side, are contained in this frontier. It can be said that they account also for a sort of  $\lambda$ -closure and  $\lambda$ -contiguity. Another way to put it recurring to the `functor.argument` expression, is saying that the functor and  $\lambda$ -idempotence diverges both  $\lambda$ -potence and  $\lambda$ -impotence. In Prolog, very simple examples, and respectively, are [46]:

```
1 ?- X = f(X).
```

... which produces Prolog's answer, in many Prolog implementations (although not in SWI-Prolog threaded, 64 bits, version 7.4.2) [46]:

```
1 X = f(f(f(f(f(f(f(f(f(f(f(...
```

While this is a proof of redundant recursion, it sets also the proper recursion power. An intelligent use of this principle and herein show, under membership and sublist relations, is the following [46]:

```
1 ?- sublist(S, [a,b,c]).
2 S=[];
3 S=[a];
4 S=[a,b];
5 S=[a,b,c];
6 S=[];
7 S=[b];
8 ...
```

Also and correspondingly settled on an higher-abstract request in terms of recursion, are the non-conflicting, and applied principles of the `functor.argument` expression, minimally (forwardly, one step, order-independent) and maximally (backtracking, various steps, order-dependent) executed, respectively, for:

1. If the list is empty then its length is 0
2. If the list is not empty then **List**=[**Head** | **Tail**]; then its length is equal to 1 plus the length of the **Tail**.

```
length([], 0).
% and ...
length(_|Tail, N):-
length(Tail, N1),
N is 1+ N1.
```

There are a lot of programming techniques and programs that benefit and are presumed in the exposed, usually ISO-compliant and also present as built-in predicates. Examples are `atom_concat(?Atom1, ?Atom2, ?Atom3)` [403, 232], `member(?Elem, ?List)` [403, 232], and `append(+ListOfLists, ?List)` [403, 232], in increasing complexity. It should be noted that, precisely in terms

of complexity, steadfastness of SLD-resolution in depth-first search permits that Goal-reduction in the computation search-tree, where the root node fails if its selected literal cannot unify with the positive literal of an input clause, obliges to consider increasing complexity with successive instantiating of variables, insofar conjoint sub-Goals share variables.

A great deal of attention should be deposited ahead in explaining unification in satisfying goals, which permits to treat also backtracking and the Cut.

If ambivalence was shown before in relation to `functor.argument`, in the same line, either with a linear, sequential, and deterministic implementation (Prolog-II, IC-Prolog, and MU-Prolog, e.g. [337]), or with traversal, parallel, non-deterministic implementation (PARLOG, Concurrent Prolog, GHC, Aurora-Prolog, and Andorra-Prolog e.g. [337]), there is an inherent bivalence in constraint satisfaction problems due to the existence of the simple principle of commutativity, being also, in recondite fashion, something that happens because backtracking search consists of a single unchangeable representation or *structure*, although with an instantiating altering state or *formalism*, the most definite frontier on the proper chasm between Wittgenstein I and Wittgenstein II.

Prior clarification on how unification operates is fundamental. Unification is just a general, and because of that, very specific case of matching: "Given a goal to evaluate, Prolog works through the clauses in the database trying to match the goal with each clause in turn, working from top to bottom until a match is found. If no match is found the goal fails." [45]. Variables are unbound, and the Head of a clause and a Goal are unified. If noticed carefully, taking into account that two terms match if they are *actually* identical (*atomist*, Wittgenstein I) or *potentially* identical (*factual*, Wittgenstein II), what is also presumed is that the same variable ('X', Wittgenstein I) having to be instantiated with the same value throughout an expression, is quite different from the anonymous variable ('\_', Wittgenstein II), which is individually unique as much as it is general in principle running at each program universally [126].

If both terms are constants and identical, unification succeeds, otherwise fails. Being compound terms, if they have the same functor and arity, and if the arguments unify pairwise, unification also succeeds.

If we think along the topic of space, in proper *metric* and *transformational* (programming languages) *factual* and *representational* choice-points, on a par with, for example, Bratko's "robot task planning" [46], "blocks in boxes" [46] as an example in constraint logic programming, and "best-first search: minimizing time and space" [46], or, indeed, O'Keefe's "Where Does the space go?" [214] entire chapter, specially "measuring space used" [214], it is better seeable how Wittgenstein's original "idea of representation of *negative* facts by means of models (*Notebooks 1914-1916* (14.11.14)) [410] and its appraisal of the *structuralism* vs. *formalism* clash – "Let us imagine a white surface with irregular black spots on it. We now say: Whatever the sort of picture arises in this way, I shall always be able to approximate as close as I like to its description by covering the surface with a suitably fine square network and saying of each square that it is white or is black. In this way I shall have brought the description of this surface into a unitary form. This form is arbitrary, for I could with equal success have used a triangular or hexagonal net." [410, 412] – is, nevertheless, very different from the use of negation in programming languages theory, and indeed from the use of negation as failure as the non-monotonic inference rule in logic programming and Prolog. In view of the latter, the "fact" [415] that "two numbers unify if and only if they are the same, so 7 unifies with 7, but not with 6.9" [45] and the "fact" [415] that "two unbound variables, say  $X$  and  $Y$  always unify, with the two variables bound to each other" [45] and also that "an unbound variable and a term that is not a variable always unify, with the variable bound to the term" [45] are all examples of Wittgenstein I, fully *atomist*, functionalist-structuralist, FOL- $\omega$ -consistent,  $P$ -like logicist (programming) philosophy, in which non-deterministic logic programming with multiple predicates is, however, a sign of typical conjoint complexity classes  $P$ TIME and  $P$ SPACE of a deterministic Turing-machine, all in all an exquisite achievement and *perfected* programming limit of all of the above. Straightforwardly, this limit is as simple as the mutual coincidence of a Turing-machine and the Warren abstract machine.

In terms of the Warren abstract machine, "it is a unification algorithm based on the UNION/FIND method, where variable substitutions are built, applied, and composed through dereference pointers. In  $M$  (machine) (...) this unification operation is performed on a pair of store addresses. It uses a global dynamic structure,



an array of store addresses, as a unification stack (called PDL, for Push-Down List)." [15] On top we cannot forget that "method in clausal logic is proof by refutation. If we succeed in deriving the empty clause, then we have demonstrated that the set of clauses is inconsistent under the substitutions that are needed for unification of literals" [131] So, principally, Prolog's execution in forward recursion and backtracking is common, and in a way also *unifiable* in logic programming, in the sense that "Prolog's execution mechanism is obtained from the abstract interpreter by choosing the leftmost goal instead of an arbitrary one and replacing the non-deterministic choice of a clause by sequential search for a unifiable clause and backtracking." [337]

Having understood the balance between determinism and sequential order on one side, and non-determinism choice and backtracking on the other side, we can proceed to the presentation of the following grammar and explicitation of algorithm, wherein CSP is the acronym for "Constraint Satisfaction Problems" [325]:

```
function BACKTRACKING-SEARCH (CSP) returns a solution, or failure
return BACKTRACK( {}, CSP)
```

```
function BACKTRACK(assignment, CSP) returns a solution, or failure
```

```

|   if assignment is complete then return assignment
|   var ← SELECT-UNASSIGNED-VARIABLE (CSP)
|   for each value in ORDER-DOMAIN-VALUES (var, assignment, CSP) do
|   if value is consistent with assignment then
|   add {var=value} to assignment
|   inferences ← INFERENCE (CSP, var, value)
|   if inferences ≠ failure then
|   add inferences to assignment
|   result ← BACKTRACK (assignment, CSP)
|   if result ≠ failure then
```

```

|   return result
|   remove {var = value} and inferences from assignment
|   return failure

```

Insofar evaluating Goals in Prolog implies a search through the database or knowledge representation base, from top to bottom, seeking for clauses that have Heads with the same `functor.arity` until finding the first for which Head and Goal unify, it has to be considered the case when for each and every attempt, they do not succeed, advancing, thus, to backtracking, the process of going back to a previous Goal in trying to convey logic *satisfiability*.

More in detail, we choose to follow the simple "family relationship example" [45] in Max Bramer's *Logic Programming with Prolog* (3.3) with forward insights from other sources. Let us not explicit the clauses, but attend instead first to the query

```

1 ?- parent(john, Child), write('The child is '), write(Child), nl.

```

What is important first to note is that when Max Bramer writes "Prolog attempts to satisfy all the goals in the sequence (simultaneously) and in doing so will find one or more possible values for variable *Child*" [45], what is meant is that there exists a simultaneous and *automatic*-driven boundness between left-right anterior Goal and Query **parent(john,Child)** and the left-right posterior **write(Child)**. Another way to put it is saying that, as there is not any anterior top-bottom, left-right Goal before **parent(john,Child)**, the clausal form is assumed *instantaneously* to be bound to any of the future recursion-occurrence of the variable **Child**. In a sense, it can be said that Prolog will try to satisfy all the Goals in the sequence and simultaneously, once each time the value for the occurrence of the variable **Child** is shown, it is, in furtherance, another occurrence of Goal also. There are in our program several Relations and two Rules in relation to the predicate **parent/2** [45]: With [P1] and [P2] Goals do not match (Listing 5). As for [P3] it *automatically* bounds **X** with **john** and the variables **Y** and **Child**. As it works so, the system shell waits for the abstract machine also to work *automatically* now with the Body of the Rule [P3]. "It successfully evaluates the goals **write('mother?')** and **nl** outputting the line of text" [45]:

```

1 mother?

```

---

```

[P1] parent(victoria,george).
[P2] parent(victoria,edward).
[P3] parent(X,Y):-write('mother?'),nl,mother(X,Y),write('mother!'),nl.
[P4] parent(A,B):-write('father?'),nl,father(A,B),write('father!'),nl.
[P5] parent(elizabeth,charles).
[P6] parent(elizabeth,andrew).

```

---

Listing 5: 3.3 The `parent` Family Relationships Example *in Logic Programming with Prolog* by Max Bramer

At this point, it is up to find the Goal `mother(john,Y)`. We have at disposal the following `mother` clauses `[M1] – [M10]` (Listing 6)

---

```

[M1] mother(ann,henry).
[M2] mother(ann,mary).
[M3] mother(jane,mark).
[M4] mother(jane,francis).
[M5] mother(annette,jonathan).
[M6] mother(mary,bill).
[M7] mother(janice,louise).
[M8] mother(lucy,janet).
[M9] mother(louise,caroline).
[M10] mother(louise,martin).

```

---

Listing 6: 3.3 The `mother` Family Relationships Example *in Logic Programming with Prolog* by Max Bramer

The next Goal `mother(john,Y)`, as a Predicate `mother/2`, does not unify with the Head of any of the clauses `M[1] to M[10]`. As this Goal fails, the system starts to backtrack. It *automatically* searches for the last satisfied Goal in the Body of the proposition `P[3]` – now having moved right to left – but, as it finds the built-in-predicate `nl/0`, one further position – again, in contrary fashion, right-to-left and bottom-up – has to be evaluated in backtracking. It finds again `write('mother?')`. which fails also for the same reasons; as it had done for `nl/0` it does now for `write/1`. To sum up, and for the present moment, there is no other option except the *integral* rejection of the Rule `P[3]`, and the proper re-instantiation of the Query:

```
1 ?-parent(john,Child),write('The child is '),write(Child),nl.
```

naturally with the variable **Child** unbound.

What is next is the **clause**  $P[3]$  searching for defining the **parent/2** Predicate in the *sequitur* top-bottom left-right expression **parent(john,Child)**. The **clause**  $P[4]$  follows, and it successfully unifies the **Goal** with the **Head**, i.e., the variable **A** bounds to **john** and variables **B** and **Child** also bound to each other.

Next in line in the  $P[4]$  **clause** `parent(john,B);-write('father?'),nl,father(john,B),write('father!'),nl.`, and after **father?**, is the third **Goal**: **father(john,B)**.

It, thus, finds, the **clause**  $F[2]$  (Listing 7).  $F[2]$  is the **Fact** `father(john,mary)`. As the variable **B** is bound to the *atom* **mary**, there is one automatic process of instantiation of the "previous" variable **Child** now to be also bound to the *atom* **mary**.

Once all the **Goals** in the body of  $P[4]$  have succeeded, with the lines of text

```
1 father!
2 The child is mary
```

it completes all the output values in the original user's **Query**. In recapitulation:

```
1 ?-parent(john,Child),write('The child is '),write(Child),nl.
2 mother?
3 father?
4 father!
5 The child is mary
6 Child=mary
```

As Max Bramer points out, "the user can now force the system to backtrack to find a further solution or solutions by entering a semicolon character. This works by forcing the most recently satisfied goal, i.e. **nl** (the last goal in the user's query) to fail. The system now backtracks to the previous goal in the sequence, i.e. **write(Child)**. This too fails on backtracking, as does the previous goal, i.e. **write('The child is ')**. The system backtracks a further step, to the first goal in the query, which is **parent(john,Child)**."<sup>[45]</sup>

In order to find an alternative, so too the  $P[4]$  **clause** is automatically con-signed to the execution of backtracking (from right-to-left, bottom-up), and within it the last **Goal** in the **Body**. Remember that the original  $P[4]$  (Listing 5) is `parent(A,B):-write('father?'),nl,father(A,B),write('father!'),nl.`

Consequently, the focus is on **father(john,B)**, looking for possible unifications for the Predicate **father/2**, except, most naturally, [F2]. Knowing the **father clauses** to be the following (Listing 7):

... the next clause for which the head unifies is [F5] and the variable **Child** is

---

```
[F1] father(henry,jonathan)
[F2] father(john,mary).
[F3] father(francis,william).
[F4] father(francis,louise).
[F5] father(john,mark).
[F6] father(gavin,lucky).
[F7] father(john,francis).
[F8] father(martin,david).
[F9] father(martin,janet).
```

---

Listing 7: 3.3 The **father** Family Relationships Example *in Logic Programming with Prolog* by Max Bramer

bound to the atom **mark**. If required, further backtracking will redirect to the clause [F7] `father(john,francis)`. In general and recapitulating the output in the console is:

```
1 ?-parent(john,Child),write('The child is '),write(Child),nl.
2 mother?
3 father?
4 father!
5 The child is mary
6 Child = mary ;
7 father!
8 The child is mark
9 Child = mark ;
10 father!
11 The child is francis
12 Child = francis
```

At this point, it is useful to alert that we should not confuse the rhetoric use of the exclamation mark as a stagger mark and the exclamation mark in the use of the Predicate **Cut**: "the predicate *cut* is represented by an exclamation point, !. The syntax for cut is that of a goal with no arguments. Cut has several side effects: first, when originally encountered it always succeeds, and second, if it is 'failed back

into' in backtracking, it causes the entire goal in which it is contained to fail." [352]. In this fashion, the **Cut** can be used to prevent unwanted backtracking, and also when used as "cut with failure", to specify one or more exceptions to general rules.

With this we learn that backtracking can lead to inefficiency and time-waste in looking for solutions. The built-in Predicate **Cut** is, thus, a mechanism of control above changing the Rule and the Goal hierarchy. More in detail, "the cut commits Prolog to any choices that were made since the parent goal was unified with the left hand side of the rule (including, importantly, the choice of using that particular clause)."[290]

In reminding that backtracking makes use of the bottom-top right-left strategy, in contrary route to the top-bottom left-right pattern of unification, the **Cut Predicate** is seen, therefore, as repeating the pattern of unification inside the mechanism of preventing backtracking, or rather as "a nullary predicate that always succeeds as a goal and if it is encountered on backtracking, then it causes the backtracking to skip all clauses" [175]. Also important to note is how backtracking, as a general retroversion of space (*atomus*) and time (*momentum*) with possible infinite regression, installs, in principle, entropy in the logical deduction mechanism, and how preventing backtracking through the use of **Cut**, is suitably called "negation as failure" as it shelters the original philosophy *desideratum* of Prolog's controlled deduction related with the closed-world assumption (CWA). Let's observe a simple **sumto/2** Predicate example in "Preventing Backtracking" (Listing 8):

```
/*sum the integers from 1 to N (the first argument) inclusive */  
sumto(1,1).  
sumto(N,S):- N1 is N-1,sumto(N1,S1), S is S1+N.
```

Listing 8: **sumto** in *Logic Programming with Prolog* by Max Bramer

Indeed, for the Query

```
1 ?-sumto(15000,_).  
2 true.
```

or for the non-anonymous variable-including Query

```

1 ?-sumto(3,S).
2 S=6

```

there is absolutely no resolution impediment and the answers are provided. Nevertheless, it is a different case when resorting to backtracking. Another way of putting it is saying that in a tree-like disposal, any node where an alternative choice is available is now susceptible of incrementally building candidates to the solutions, and, in inverse way, with the **Cut** it is truncated every time its '!' sign operator precedes it in the *contra*-original unification direction that is characteristic of the backtracking constraint satisfaction problem algorithm. The **Cut** literally trims away the branches, reducing exponentially derived outcomes.

Now confronting with the above code and the result **S=6**, this time trying again from the **Query** and **Goal** `?-sumto(3,S)`. backtracking will force Prolog to try the **Goal** `sum(1,S)`. What happens is that the first **clause** `sumto(1,1)`. permits the instantaneous instantiation of the argument **S** to the value **1**. Backtracking's inverse resolution targets the first **clause** and within it the second **argument**. Consequently – with the result being so far the proper first **clause** `sumto(1,1)`. the first **clause** is rejected and it will try to satisfy the second **clause**, which is `sumto(N,S):- N1 is N-1,sumto(n1,S1),S is S1+N`. As it is trying to solve originally `sumto(1,S)`. "it causes it to subtract one from one and then evaluate the goal `sumto(0,S)`. Doing this will require it to evaluate `sumto(-1,S1)`, then `sumto(-2,S1)` and so on, until eventually the system runs out of memory." [45]. A proper rectification, not yet using the **Cut Predicate** would be (Listing 9)

```

sumto(N,S):- N>1,N1 is N-1,sumto(N1,S1),S is S1+N.

```

Listing 9: 7. Preventing Backtracking in *Logic Programming with Prolog* by Max Bramer

Conventionally, what we find here is an *effective* programming melioration of a **clause** in particular. As it is noticed, it suffices the short piece `N>1`, in the overall **clause** `sumto(N,S):-N>1,N1 is N-1,sumto(N1,S1),S is S1+N`. to have, in resolute way, corrected the problem. Nevertheless, a way to make it as general as resolution proper, is turning **Cut** inasmuch a **Goal**, as a **Predicate**.

In such way, implementing the **Cut Predicate**, the same example revised is here shown (Listing 10):

```
sumto(1,1):-!  
sumto(N,S):-N1 is N-1,sumto(N1,S1),S is S1+N.
```

Listing 10: 7. Preventing Backtracking (Revised) *in Logic Programming with Prolog* by Max Bramer

Now, the implementation of **Cut** – no more than an **atom** – by the use of **!**, involves memory in the search-tree with specific pointers: "Note that backtracking requires that all previous resolvents are remembered for which not all alternatives have been tried yet, together with a pointer to the most recent program clause that has been tried at that point. Because of Prolog's depth-first search strategy, we can easily record all previous resolvents in a goal stack: backtracking is then implemented by popping the upper resolvent from the stack, and searching for the next program clause to resolve with"[131].

One important thing to note is that the use of **Cut** is all-inclusive in its forbiddance procedure, meaning that not only the present (actualist, Wittgenstein I) **clause** is prevented from backtracking, as any other **clause** for that matter including it as a **Predicate**. It also resorts differently with forced backtracking, once the fact that all possible instantiations have been already given, changes the scenario as when the user presses the **;** (semicolon) – Prolog first checks for alternatives to the last inclusive **Goal**, and it looks upon alternatives including it, as if there had been a **!** (exclamation mark) instead of a **.** (full stop).

Another important point to call attention to is how the pure recursive left-right top-bottom SLD-resolution strategy, in being the opposite to pure recursive right-left bottom-top backtracking strategy, does not quite meet **Cut** in the middle: commitment to the last result obliges **Cut** to be more so a full stop for inverse resolution when backtracking, than eventually it is a full stop – comprehensively the alternate **!** (exclamation mark) – for the inverse resolution that is backtracking itself, because many other **clauses** might still be available in the program (as seen in the *Family Relations* example), explaining also the pertinence of the



difference with the use of ; (semicolon). All in all, backtracking is very much like resolution itself, except for being inverse and, by order, non-deterministic and many-directional.

A program with **Cut** does not quite stop the (selective linear resolution with definite clauses) mechanism but only its reach, and in making disposal of the bottom-top right-left backtracking to perform cancel of its operation, even for one only *momentum* and **atomus**, is just as preventing of backtracking, as it is of resolution itself. A proof of such is the unalterable "state-of-affairs" order and hierarchy of the clauses in a program. **Cut** holds, thus, some resemblance with the *supremum* use, in proper analytic terms, typical of *&-Falsum* in a truth-table logic, even though preventive and not effective, which is in sharp contrast with the *infimum* use, in analytic terms, of the  $\vee$ -*Verum* in a truth-table logic, all the more effective than preventive. For the same reason alike, although by contrast, this also explains why logic programming and Prolog hold an intrinsic supervening and precedental Relation of the disjunctive normal form (DNF) over the conjunctive normal form (CNF), and why, in the first place, the **Cut** mechanism is not an Horn clause.

This is also and mainly the reason for some to have criticized **Cut** as being alien to the Prolog programming philosophy and to be generally considered as "an extra-logical control annotation that helps the Prolog programmer trim the search space" [15]. **Cut** is, nevertheless, still a control mechanism, attesting still for a general left-right top-bottom nondeterministic transitive-intransitive deterministic linear-logic programming transitoriness. Again, its existence attests better for the general ambivalent status of logic programming and Prolog, while its detriment suggests a typical dialectic use of *pure* (*practical*-programming) reason.

In these terms, in a perhaps fortuitous analogy, if **Cut** itself was a STOP traffic sign, it would be faced to the reverse direction that is backtracking, but in this way it could be seen, in its special form, from the inverse path that is resolution. But this does not happen, and SLD-resolution is totally transitive over its symbol as if it would correspond – left-to-right top-bottom, but also right-to-left bottom-up – to **!,** or maybe, more appropriately, **,-:**.

In this way, what is presented is the clarification of the proper programming entropy in backtracking, and how the **Cut** mechanism attests for a logical brain, in

the sense that, in similar lines with the "Boltzmann brain" [384, 369], it commands a self-aware control – out of "Algorithm = Logic + Control" [224, 223] – arising out of highly possible random fluctuations statistically deviated from a state of logic programming (linear and deterministic) equilibrium. Under the same line, but in forward syntactic unification, also the mechanism of the so called "occurs check" [395, 46, 84] – "For reasons of efficiency, in almost all implementations of Prolog the occur check is left out. This mechanism should protect the program against introducing circular bindings of variables. In practice the occur check is very expensive, however, and it is left to the skills of the user, to avoid these circular bindings in the program" [395] – is relevant to grasp how equisatisfiability through the use of formulas written as a string of (prefix) quantifiers followed by a quantifier-free part (matrix) in prenex normal form, and, thereafter, in Skolem normal form (in prenex normal form with only universal first-order quantifiers), encounters an interpretation of a formula in conjunctive normal form (CNF) or clausal normal form where a clause is a disjunction of literals, that is permitted to preserve soundness by not validating alternatively valid variables in theorem proving. *Ideally*, logic programming and Prolog would automatically backtrack the whole tree for each possible unification, but this is just another example of control manipulation and, in the case of forward unification, complexity reduction and *P*-like decidability in an Herbrand universe.

Indeed, not accounting for "occurs check" [395, 46, 84], and preventing a many-worlds universe, it is expected the worst case complexity in one such panorama to be reduced from  $O(\max(\text{size}(t_1)) + \text{size}(t_2))$  to  $O(\min(\text{size}(t_1), \text{size}(t_2)))$ .

What is here presumed is that the left-right top-bottom resolution pattern itself is also preventive of backtracking – fixing a variable as if it was a non-**Cut** in resolution and unification – and because of the possible backward mechanism for each possible variable new instantiation, perfectly *ideal* occurs check would force testing for all possible variables, including the previous ground terms that were instantiated. In this way it is preferable one single variable multiple instantiation within logic programming without occurs check in perfect control, than logic programming with occurs check, with looping backtracking and resolution fail within unification theorem proving, with pluperfect logic and algorithm.

Unless there was some preventive mechanism as **Cut** is to backtracking for all the previous ground terms unification for each branch of the resolution applied to occurs check, it is made preferable to control, still in the spirit of ["Algorithm = Logic + Control"] [224, 223], or rather, by us preferred [Control = Algorithm + (- Logic)], the natural (CWA)-*P*-like-assessment of occurs check omission, to the point that principles of automated theorem proving respond instead to normal unification ever since Prolog's ISO/IEC 13211-1:1995.

This is not to say that occurs check and infinite trees complexity and sound mechanism, in play with ascendance of the conjunctive normal form (CNF) in Horn clauses and resolution, should not be researched, and even, in accordance with *imagination*, be bonded to different levels of (logical) complexity theory. Certainly, both the rooting *P*-like resolution and **Cut**, and the exponentiation *NP*-like mechanism of occurs-check and backtracking, demand a study of complexity *diagonal* effective implementations, and complexity *parallel* theoretical classes.

Having devised up until now the programming signature of logic programming and Prolog in sufficient manner, it is time now to detail sufficiently its place in the history of programming languages. Our strategy will be to attend, just as minimally as theoretically, past convergent  $\lambda$ -Calculus, Lisp, and Algol 60-68, and ahead divergent C/C++, Java, and Python, forasmuch as they reveal traits of logic programming and Prolog.

## 2.3 Past Convergent Fragments of $\lambda$ -Calculus, Lisp, and Algol 60-68, and ahead divergent fragments of C/C++, Java, and Python

$\lambda$ -Calculi [319, 40, 254, 71, 302, 358, 11, 172, 180, ?, 194, 331, 421] are "prototype programming languages" [21], but, at the same time, also *one only* powerful calculus (irrespectively of being untyped, and thus, closer to most assembly languages and machine code itself, allowing direct operations to be performed on any data or sequences of bits of any length, imminent to Gödelization and number theory, or else being a typed language, in line with the "majority of modern functional programming languages" [172], thus closer to mathematical logic and theorem

proving) insofar it is the proper representation and formalization of computability theory "in terms of definability of numerals" [21]. The same could be said in relation to the statement that " $\lambda$ -calculus can be called the *smallest universal programming language in the world*" [319] once, in fair logic, it could also be said to be, for the reason above and for the sort of phylogenetic trace of programming languages, the *longest programming language in the world*.

We should neither blame Gödel's "thoroughly unsatisfactory" [40] remark on the Kleene-Church previous work on "a formal system for the foundations of mathematics by having a system of functions together with a set of logical notions" [21] – or rather its consistent subsystem –, as it would not be expected that such a simple function abstraction and variable binding application and substitution would turn out to be, in ontogenetic symbolic terms, with exactly one single transformation rule ( $\beta$ -conversion) and one single function scheme, a symbolic avenue for both natural and artificial "expressions", as paramountly, a proper computability formalization, equivalent to the Turing machine.

Again, a research of the foundations of *pure* mathematics rapidly thrust outstanding *empirical* application, in this case fortunately synchronous with the Turing machine, in the overall both founded on the modern Leibnizian *dialectic* heritage, recreated by Henk Barendregt and Erik Barendsen in the following straightforward paragraphs: "Leibniz had as ideal the following. (1) *Create a 'universal language' in which all possible problems can be stated.* (2) *Find a decision method to solve all the problems stated in the universal language.*" [180].

$\lambda$ -Calculus – wherein the ' $\lambda$ ' notation came in as an alternative, confronting with Russell and Whitehead's [1910-13]  $f(x) = 2x + 1$  also written as  $2\hat{x}+1$ , and as "Church originally intended to use the notation  $\hat{x}.2x+1$ " but such typesetting was impossible, if first was shown the lambda-expression  $\lambda x.2x+1$  – comes to be, as a result, "a non-extensional theory of functions as rules of computation, contrasting with an extensional theory of functions as sets of ordered pairs. Despite its sparse syntax, the expressiveness and flexibility of the  $\lambda$ -calculus make it a cornucopia of logic and mathematics" [10]. In this outlook, The  $\lambda$  operator serves as a sort of binding glue for any *abstract* over any variable and, consequently, for successive *applications* (incrementally, imagined to pass through expressions, functions, and applications proper). In other words, as simple as it can get, "for the alphabet of

the language of the  $\lambda$ -calculus we take the left and right parentheses, left and right square brackets, the symbol ' $\lambda$ ' and an infinite set of variables." [10] The class of  $\lambda$ -terms is, therefore, cast into the inductive *abstract*, and also *applied* hierarchy [10, 319]:

$x$  | Variable | A character or string representing a parameter or mathematical-logical value | Every variable is a  $\lambda$ -term.

$(\lambda x.M)$  | Abstraction | Function definition ( $M$  is a lambda term). The variable  $x$  becomes bound in the expression. | If  $M$  and  $N$  are  $\lambda$ -terms, then so is  $(MN)$

$(M N)$  | Application | Applying a function to an argument.  $M$  and  $N$  are lambda terms. | If  $M$  is a  $\lambda$ -term and  $x$  is a variable, then  $(\lambda x[M])$  is a  $\lambda$ -term.

With this, it can be said that  $\lambda$ -Calculus is a convolution mechanism that binds variables to functions, thereon functions with variables to arguments, the latter to full expressions, and possible existent expressions with all the sort of other different expressions. ' $\lambda$ ' in  $\lambda$ -Calculus acts, thus, as a sort of natural-to-artificial graph (zipper-scissors) chain-like informatic molecule (Wittgenstein I) that, precisely due to the verge of the passage from extensional (functions ordered pairs) to non-extensional (functions in computation), is also *practically* dimensioned to computability theory, and strings of symbols – "The 'body' of the function specifies how the arguments are to be rearranged. The identity function, for example, is represented by the string  $(\lambda x.x)$  (...). In fact, anything of interest in  $\lambda$ -calculus is a function. Even numbers or logical values will be represented by functions that can act on one another in order to transform a string of symbols into another string. There are no types in  $\lambda$ -calculus: any function can act on any other. The programmer is responsible for keeping the computations sensible." [319] – to an *ideally* three-dimensional extensional and structural motif from combinatory logic to adding types. Here we portrait its axis [319]:

$\langle \textit{expression} \rangle := \langle \textit{name} \rangle \mid \langle \textit{function} \rangle \mid \langle \textit{application} \rangle$

$\langle \textit{function} \rangle := \lambda \langle \textit{name} \rangle . \langle \textit{expression} \rangle$

$\langle \textit{application} \rangle := \langle \textit{expression} \rangle \langle \textit{expression} \rangle$

In this mathematical-logic processing, it has to be granted that three reductions occur. They are, respectively,  $\alpha$ -conversion (reduction) (changing bound variables),  $\beta$ -reduction (applying functions to their arguments), and  $\eta$ -conversion (reduction) (coextensive extensionality). A simple alphabetical substitution and variable renaming, avoiding name collisions  $(\lambda x.M[x]) \rightarrow (\lambda y.M[y])$  or the equivalent  $(\lambda z.z) \equiv (\lambda y.y) \equiv (\lambda t.t) \equiv (\lambda u.u)$  accounts for  $\alpha$ -conversion (Wittgenstein I), wherein "The names of the arguments in function definitions do not carry any meaning by themselves. They are just 'place holders', that is, they are used to indicate how to rearrange the arguments of the function when it is evaluated." [319]. Next comes  $\beta$ -reduction, as in  $((\lambda x.M)E) \rightarrow (M[x := E])$ . In this respect, the difference between free and bound variables is settled by the (Wittgenstein I) local definition logic, by which "in the function  $\lambda x.x$  we say that  $x$  is 'bound' since its occurrence in the body of the definition is preceded by  $\lambda x$ . A name not preceded by a  $\lambda$  is called a 'free variable'." [319] The abstraction operator  $\lambda$ , in binding its variables wherever they occur in the body of the abstraction, falling within the scope of an abstraction, enacts, crucially, one decisive property: "A major property of functional languages is *referential transparency*; the property which allows equals to be substituted by equals." [172] This means that the  $\lambda$  formal system with mathematical induction, past learning the substitution lemma and the variable convention, along with inherent extensionality – the  $\lambda + ext$  or, indeed,  $\lambda\eta$  as suggested by Church, matching the convertibility relationship in relation to intensional equality – and past proofs of consistency and completeness, is opened to the treatment of reduction (the Standardisation theorem along with the Church-Rosser theorem), which equip  $\lambda$ -calculus with strong determinacy typical of a categorical abstract machine. Standardisation responds to the fact that reduction sequences which terminate in a normal form, is sufficient to attest that all the possible sequences also terminate in normal form, while the Church-Rosser theorem rules attests that the possible sequences are independent of reduction rules strategy and are, therefore, confluent.

Generally, normal forms are programs of relational  $\lambda$ -terms, insofar "the evaluation of the  $\beta$ -normal form of a term involves removing application subterms by applying the ( $\beta$ ) rule." [172]

As for the case of Lisp [112, 153, 5, 154, 233, 159, 353, 189, 261], Lisp is a **List Processing** (specified in 1958/1959) [261, 260]) programming language, created by John McCarthy, who published its design in an ACM Communication paper [261] (1960), although extant prior technical literature was accessible and complete [260], at a time when the representation of S-expressions and the system of S-functions was to be integrated. Right from the start, McCarthy elucidates that a machine-independent system of recursive functions of symbolic expressions for the IBM 704 by the Artificial Intelligence group at M.I.T., facilitated by the Advice Taker (1958) – the antecedent of question-answering in logic programming –, had the goal of being "instructed to handle declarative as well as imperative sentences and could exhibit 'common sense' in carrying out its instructions." [261]. This idea of "programs with common sense" [259] is critical, specially because McCarthy establishes in this paper [261] a universal S-function `apply` which is settled to play the *theoretic* role of a universal Turing machine and the *practical* role of an interpreter, while the representation of S-expressions in the memory of the IBM 704 by *list* structures was designed to meet logic machine "states", as used by Newell, Shaw and Simon S-functions by program [279] <sup>18</sup>. In retrospective, thus, it is clear how much Lisp has met "common sense" by what it could not have but brought in – reinventing the calculi of lambda-conversion to higher dimensions, and in having resolved the high-level imperative and compiled programming typical of numeric computation of Fortran Automatic Coding System (1958) on the same IBM 704, now in symbolic fashion – but strongly enough also to have advanced the status of programmable functions and data under the versatile type of *lists* in EVAL notation – an evaluating-function of a string or box-to-box denotation, as though it were an expression or structured representation of code, such as an abstract syntax tree –, as if it were, on the whole, a more *complexioned* and *dialectic* lambda-conversion.

---

<sup>18</sup>Lisp's first sowing the seeds until the *Lisp 1.5 Programmer's Manual* [195] by McCarthy *et al.* (1962) dates back to the summer research AI meeting at Dartmouth College (1956), where McCarthy was introduced to a very raw technique of "list processing" in the context of Newell's, Shaw's, and Simon's IPL (Information Processing Language). Curiously, there was also a suggestion, after the slightly anterior Fortran (1958), of FLPL (Fortran List Processing Language). Preferably to saying that McCarthy wrote the first version Lisp 1, developed for the IBM 704, contrary to these previous endeavours, it is wiser to go along with the idea that McCarthy "working first at Dartmouth and later at the Massachusetts Institute of Technology, designed a new language, LISP (for LIST processor), that drew on ideas from IPL, FORTRAN, and FLPL." [360]

Lisp welcomed a range of new features that are worth mentioning. Garbage collection (1959) stands out, the inseparable characteristic of implicit types, under Wittgenstein I typical logical *atomism* – and, iteratively, the first *atom* of the minimal *list* (a pair) –, as if it was a natural *declarative*, 'what' code tag to proper assembly language also *atomist* Wittgenstein I *imperative*, 'how' pointers and memory addresses (denotational and operational semantics being very close). Made inseparable was also, past both assembly languages and Fortran (1958) Wittgenstein I numeric and high-level *inter-local* direct feeding and linkage, the new feature in Lisp (1959) of REPL (read-evaluate-print loop), which permitted programming observance and implemented incremental compilation in an interactive, truly man-machine, environment. Indeed, from Lisp 1.5 to PDP-6 Lisp (1960-65), at a time when computers were designed for Lisp, "before Unix became widespread, (and) there was as much experimentation in the design of operating systems as in the design of programming languages" [159], the result, however, was one of a strong and small assembly code core and compiler: "In 1965, virtually all of the Lisps in existence were identical or differed only in trivial ways" [159], but in this context it has to be mentioned that one of the reasons for that to have happened was that as early as 1962, the MIT implementation of Lisp would compile expressions that were entered in REPL. Being so, other features, such as closures and meta-programming, along with meta-circular evaluation, have naturally come into play.

There are various aspects in the history of Lisp that deserve appraisal. Here we outline some with importance. Lisp 1.5 – Alan Kay has gone to the point of describing Lisp as the "Maxwell's equations of software", reporting the reading of the LISP 1.5 Programmer's Manual [195] "half page of code on the bottom of page 13" where we learn that "evalquote is defined by using two main functions, called `eval` and `apply`. `apply` handles a function and its arguments, while `eval` handles forms." [195] – settles far away from the PDP-6 (1964) and PDP-10 (1969), running on 36-bit words and 18-bit addresses (able to represent integers to an accuracy of ten decimal digits and storage 6x6, of six alphanumeric characters encoded in a six-bit character code) high power calculating machines, thus, also able to carry `CAR` and `CDR` - **Head** and **Tail** in Prolog terminology - to `CONS` (Wittgenstein I, atomist) – **concatenation** in Prolog terminology – cells, in very fast computation, considering



that time period in history. Different dialects sprang, with MIT MacLisp being the AI LAB dialect of choice (with some notable differences in relation to the original Lisp 1.5, such as the syntax preference of `eval` in relation to `evalquote`), which later improved MacLisp fast arithmetic compiler "on a near par with Fortran compilers" [159], showcasing LISCOM (1970-1972) as a peak achievement in compiler calculation power, but eventually losing the run to DEC Fortran compilers.

This suggests there is a special cybernetics-(phenomenological) and informatic-(material) *dialectic* between *imperative* and *numeric* (centrifugal, radial and efferent) liableness on one side, and *declarative* and *propositional* (centripetal, centralizing and integrable) proneness on the other side, in programming philosophy, visible in the Fortran and Lisp paramountcy with the outcome of Algol-like syntax (1958-1968) high-level dominion, later the same with C and Prolog (1972), after the AI winter (1984), giving forth the dominion of the web multi-paradigm language of Java (1995), demanding object-oriented (class-based), structured, imperative, generic, reflective, and concurrent modern programming features. Maybe in a more profound way, it attests for a timely *dialectic*, of necessarily anew programming languages design, although contained in previous experiments (take the case of Planner, designed by Carl Hewitt at MIT in 1969, later implemented as Popler by Julian Davies at the University of Edinburgh in the POP-2 programming language, which directly influenced Prolog, or take the case of Scheme, "An Interpreter for Extended Lambda Calculus", first appearing in 1970 as a dialect of Lisp, after reflective work at the MIT AI Lab by its developers, Guy L. Steele and Gerald Jay Sussman, by means of AI memos known as the *The Lambda Papers*). In philosophy of science terms, programming philosophy and languages design is often enticed, more than Otto Neurath's (1882-1945) unifying science positivism, Karl Popper's (1902-1994) falsificationism, Thomas Khun's (1922-1996) paradigms, Imre Lakatos's (1922-1974) research programmes, Paul Feyerabend's (1924-1994) theoretical anarchism, or Ian Hacking's (1936-) new experimentalism and selective realism, by something as basic as "The Language Instinct" [300].

Indeed, and independently of any of the different implementations – including ahead examples, such as Interlisp, IBM Lisp360 and Lisp370, Scheme (1975-19080), early Common Lisp (1980-1984), and standard development (1984-1990) producing official standards for Lisp dialects within IEEE, ANSI, and ISO – one of the

strongest impressions in Lisp programming is a sort of Claude Shannon's (informational) circuitry boxes direct feeling of manipulating now ranked higher data/code functions. We here let programming fragments, as for the case of the function MEMBER in Lisp 1.5 [195, 159]:

```
DEFINE((
  (MEMBER (LAMBDA (A X) (COND ((NULL X) F)
    ((EQ A (CAR X) ) T) (T (MEMBER A (CDR X))) )))
))
```

The following is an intermediate 1960's coding style applying to the same function [159]:

```
DEFINE((
  (MEMBER (LAMBDA (A X) (COND
    (NULL X) F)
    (EQ A (CAR X) ) T)
    (T (MEMBER A (CDR X))) )))
))
```

Finally, let us confront with modern ANSI Common Lisp (1984, 1990 second edition, 1994 for ANSI Common Lisp), an ARPA-driven project, and a fully standardized and improved successor of Maclisp with many extensions, based via the package manager QuickLisp. It is an high-level, general-purpose, object-oriented, dynamic, functional programming language, indeed also apt to cope with partial backwards compatibility to Maclisp and even to John McCarthy's original Lisp 1.5 [159, 154, 360] :

```
1 (defun member (element list)
2   (cond ((null list) ())
3         ((eq element (first list)) t)
4         (t (member element (rest list)))))
```

Listing 2.1: Member definition in Common Lisp

We here review further exemplary fragments of code, in order to minimally showcase the antecedent lineage of Prolog. In the overall, Common Lisp may be described as a code and data structure symbolic-expressions (s-expressions)

in the form of lists (function calls, macro forms and special forms). Common Lisp's data types involve numbers with arbitrary-precision arithmetic, but what is more relevant is, essentially, the data (Wittgenstein I, atomist) CAR and CDR CONS cells, working the sequence type through other data objects, such as vectors, and strings, one-dimensional to multi-dimensional arrays. Eval is a first-class function model, presuming either a special operator (against a fixed list), a macro operator (previously defined) or the name of a function (by default), which may be either a symbol, or a sub-form beginning with the symbol `lambda`, in which case its arguments are evaluated left-to-right, and the function meets those values as parameters. No wonder that "Lambda expressions look similar to DEFUNs, except that the function name is missing and the word LAMBDA appears in place of DEFUN." [360], being clear the remote connection of the anonymous variable ("`_`" in Prolog) and the anonymous function-lambda expressions, commonly associating symbols as variables (global, via a `defvar` or `setq` constructs, or local, via the `setq`, `let`, or `prog` constructs) to make named functions. If noticed, in opposition to a predicate language like Prolog, this simple description is enough to attest for the rather unnoticed, but deep similarity between *symbolic* (ahead of *declarative*) and *imperative* (ahead of *procedural*) programming. Lisp's optimization goes possibly down to direct imperative machine language, and, as a matter of fact, possibly up to symbolic optimization and programming language design, precisely influencing a predicate language like Prolog, among others:

```

1 (format t "(+ 5 4) = ~d ~%" (+ 5 4))
2 (format t "(- 5 4) = ~d ~%" (- 5 4))
3 (format t "(* 5 4) = ~d ~%" (* 5 4))
4 (format t "(/ 5 4) = ~d ~%" (/ 5 4)) ; = 5/4
5 (format t "(/ 5 4.0) = ~d ~%" (/ 5 4.0)) ; = 1.25
6 (format t "(rem 5 4) = ~d ~%" (rem 5 4)) ; = 1 Returns the remainder
7 (format t "(mod 5 4) = ~d ~%" (mod 5 4)) ; = 1 Returns the remainder

```

Listing 2.2: Lisp example of mathematic functions

```

1 (lambda (x)
2   "Return the hyperbolic cosine of X."
3   (* 0.5 (+ (exp x) (exp (- x)))))

```

Listing 2.3: Lisp example of a Lambda expression

We subscribe entirely John Foderaro's comment on Lisp, found in Paul Graham's *On Lisp*: "Lisp is a programmable programming language" [153].

Thinking from here, now in terms of one "programmed programming language" instead, we shift our focus to Algol [339, 400, 3, 4, 200, 231, 230]. Algol for **Algorithmic Language**, sums up three major specifications, whose index states the correspondent year: Algol 58 (including JOVIAL, MAD, and NELIAC), Algol 60 (including ALGOL W, ALPHA, Atlas Autocode, CPL, EULER, Formula ALGOL, S-algol, Simula, and SMALGOL) which accounts for the peak uncontested success *momentum* in the language development, and Algol 68 (including Mary and S3).

Algol 58 was based on the original proposal of IAL "International Algebraic Language", after a preliminary report [3], which would drive later to the "Report on the Algorithmic Language" by the ACM Committee on Programming Languages and the GAMM Committee on Programming [4]. These laid firmly on a solid bedrock, once the proper GAMM Subcommittee for programming languages was born out of forum discussions about algorithmic languages linear translation into machine code, with the realization of an international symposium on automatic computing at Darmstadt, Germany, as early as 1955. Eventually ACM joined GAMM and both conjoined efforts to design a common algorithmic language. During the year of 1958, F. L. Bauer, the artificer of the software engineering stack method, presented the GAMM proposal to the ACM group at a meeting in Philadelphia, USA, and, finally, the GAMM-ACM ALGOL 58 conference was held in Zürich, Switzerland, producing the just referred ALGOL 58 report [3].

Very rapidly, proposals for local improvements to the Algol 58 Report [3] were given projection with publications both in the *Communications of the ACM Journal* [50] and in the *Algol Bulletin* [240], driven by the efforts of Peter Naur, who belonged to Danish Regnecentralen, established to be the European medium for the "spread of word" of ALGOL. Something important to remember in this context is that ALGOL necessarily encompassed the informational and historical context of Fortran (for **Form**ula **Trans**lation) (IBM, 1957), the first ever general-purpose and high-level, von Neumann, and compiled imperative programming language, which was designed to suit numeric computation and also scientific computing. Fortran was an invention implemented by a team whose leader and director was John W. Backus (1924-2007), the renowned creator of the Backus-Naur form (BNF) (further recognizing the contribution by Peter Naur from Denmark, also engaged in Algol), itself a formalism capable of defining a formal language syntax.

John Backus' work is mostly relevant not just for the topic of the Fortran creation and Algol's descriptive implementation (the Backus-Naur form for meta-linguistic formulas, besides having been used to describe the Algol 58 syntax [3], was firstly used in the Algol 60 report [200, 231, 230, 339, 400]), but also because it was naturally permeable to Chomsky's linguistics context-free grammars, which crossed with natural language processing systems, as envisaged later in the 1970's with Prolog. John W. Backus was, therefore, determinant both in the defining of a *numeric* and *mathematic* programming and informational analytic – the IBM first draft of the mathematical **Formula Translating System** (1954), the ensuing Fortran manual (1956), and the first optimizing compiler (1957) – as it was also in the defining of the *symbolic-semiotic* and *declarative* programming and informational *dialectic* – forbearing for the future, not only the syntax formal descriptive/grammatical system of programming languages, as the semantics prescriptive/operational code for each programming language families and paradigms to come. Most notably, this was achieved through the use of Backus-Naur form on from Algol 60, an **algorithmic** language now parallel with denotational, axiomatic, and operational semantics. Axiomatically, Fortran (1957) and Algol (1958) correspond to the passage equivalent in the history and philosophy of logic, wherein Frege's reading by Bertrand Russell sparked the so called Russell's paradox (1901) in the foundations of mathematics – the self-containment expression of the Backus-Naur formalism representing that very thing as a corrected programming *Begriffsschrift* – and also Fregean reconceivment of logic by constructing a formal system constituted as the modern predicate calculus (with an *analysis* of quantified statements and a proper formal notion of proof in one such *dialectic* discursivity), attesting for the semantic explosion of programming languages. If Kant pursued to base critic philosophy on synthetic, typically mathematical, *a priori* judgements, Frege pursued further to demonstrate that theoretical mathematical statements originated from simpler logical notions, for that reason having exposed – admitting logicism and its paradoxes – inasmuch a "programmable programming" *symbolic-operational* philosophy, as a "programmed programming" *algorithmic-imperative* philosophy. It is also very interesting that this corresponds to a sort of informational and programming natural-to-artificial antinomy between *generative* rules

of context-free grammars, and also of *transformational* rules of context-sensitive grammars, a definition stipulated by Noam Chomsky, pointedly, in 1956.

Fortran (1957) has attested more for the *generative a priori* classes of programming philosophy, in the sense that it has strongly pervaded – Fortran (1957), Fortran II and Fortran III (1958), Fortran IV (1961), the first standard, officially X3.9-1966, and known as FORTRAN 66 (1966), Fortran 77 (1977), Fortran's two versions ISO/IEC standard 1539:1991 (1991) and the ANSI Standard (1992), Fortran 95, and Fortran post-millennium (2002, 2008, 2018) – while Algol, specially in Algol 60's facet of the descriptive Backus-Naur form, with the imprint of academic and scientific status, historically prerogative but absent in futurity, inaugurating the combination of imperative executions with call-by-name  $\lambda$ -calculus, is much more indicative of the *generative* classes in programming philosophy. It is also very prevalent a sort of natural-to-artificial antinomy in the transformation and progress Fortran $\rightarrow$ Lisp $\rightarrow$ Prolog, as well as in the passage Fortran $\rightarrow$ Algol $\rightarrow$ C, so as to find Prolog and an effective compiler in the C language.

Also interesting to note, and thoroughly related with the previous, although not easily perceivable, is that John Backus defended, in a famous paper named *Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs* [19] that typical assignment statements in von Neumann programming languages caused a demarcation between a mathematically ordered *topological* programming space with algebraic linear statements on one side, and on the other side a mathematically disordered *topological* programming space where different heuristic and semantic statements abound in intersection with the previous, but without clear algebraic properties. Remember that John Backus was a mentor for functional, hierarchically constructed, structuring programming, non-repetitive and non-recursive, non-von Neumann, but immediately applicable code. Basically, this endeavour and critic targeted for "an algebra of programs, whose variables range over programs and whose operations are combining forms" [19]. John Backus' distaste for a programming tower of Babel pushed him to solomonically legislate three different models of computing systems: "some models are pure abstractions, some are represented by hardware, and others by compiling or interpretive programs" [19], which is a sort of tripartite programming contract, whose

last agreement he would preferably see divested rather than merged (simple foundational and operational models like the Turing-machine and various automata; applicative models such as Alonzo Church's  $\lambda$ -Calculus and pure Lisp; and, at last, von Neumann models of the so called conventional programming languages, labelled to be "complex, bulky, not useful" [19] causing unnecessary bottleneck "tied to a word-at-a-time thinking" [19]. This is remindful of one sort of Kripke's *Naming and Necessity* programming *analytic* critic (against programming descriptivism, even if against *a posteriori* necessities), upholding a retreat from semantics *dialectic* unguided explosion, which is, oddly enough, very contradictory with the essence of the Backus-Naur form). It's unavoidable that we look into the 1977 Turing lecture. It is really difficult to find one paper with a more caustic, although solemn, tone in any of research areas like John Backus' paper. Here we recollect an extended, illustrative selection:

"Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak (...) Programming Languages appear to be in trouble. Each successive language incorporates, with a little cleaning up, all the features of its predecessors plus a few more (...) But there is a desperate need for a powerful methodology to help us think about programs, and no conventional language even begins to meet that need. In fact, conventional languages create unnecessary confusion in the way we think about programs (...) Discussions about programming languages often resemble medieval debates about the number of angels that can dance on the head of a pin instead of exciting contests between fundamentally differing concepts. Many creative computer scientists have retreated from inventing languages to inventing tools for describing them. Unfortunately, they have been largely content to apply their elegant new tools to studying the warts and moles of existing languages (...) The purpose of this article is twofold; first, to suggest that basic defects in the framework of conventional languages make their expressive weakness and their cancerous growth inevitable, and second, to suggest some alternate avenues of

exploration toward the design of new kinds of languages (...) In its simplest form a von Neumann computer has three parts: a central processing unit (or CPU), a store, and a connecting tube that can transmit a single word between the CPU and the store (and sends an address to the store). I propose to call this tube the *von Neumann bottleneck* (...) Surely there must be a less primitive way of making big changes in the store than by pushing vast numbers of words back and forth through the von Neumann bottleneck. Not only is this tube a literal bottleneck (...) but, more importantly, it is an intellectual bottleneck (...) Thus programming is basically planning and detailing the enormous traffic of words through the von Neumann bottleneck, and much of that traffic concerns not significant data itself but where to find it." [19]

Now, diverging more properly to the track of our reflection on Fortran (1957) and Algol (1958) conjoined *momentum*, combining with the encasement of Wittgenstein I, purely atomist and functionalist interpretation in philosophy of science, language and logic (in our interpretation the model of all programming languages, low or high level, von Neumann or non-von Neumann, of whichever paradigm) and combining also with the implicit parallel assumption in  $\cup$ -Mentalism of a non-von Neumann architecture – however, in the case of  $\cup$ -Mentalism, not even remotely meaning a drawback to functionalist, *a priori*, mathematical, nor even numerical, in truth not even to a (programming) language, but instead to a programming *media* as *natura naturans* equivalent in the artificial-to-natural unfolding, i.e., in *artificiosus artificialis* mode – we single out the following next lines by John Backus, still in the context of the paper *Can Programming be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs* [19]:

"Conventional programming languages are basically high level, complex versions of the von Neumann computer. Our thirty year old belief that there is only one kind of computer is the basis of our belief that there is only one kind of programming language, the conventional – von Neumann – language. The differences between Fortran and Algol 68, although considerable, are less significant than the fact that both are based on the programming style of the von Neumann computer



(...) von Neumann programming languages use variables to imitate the computer's storage cells; control statements elaborate its jump and test instructions; and assignment statements imitate its fetching, storing, and arithmetic. The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in a word-at-a-time terms in much the same way the computer's bottleneck does." [19]

Indeed, Fortran (1957) and Algol (1958) should be compared in various angles. Retaining that John Backus himself, even if overtly reluctant towards high-level implementation with the risk of poorly discontinuous, sentential redundant semantic programming languages, was involved in the making of Fortran and in the descriptive design of Algol, it is our understanding that the late 1950's in programming languages history and development, naturally driven by the forces of Fortran and Algol, are a clear demonstration of the *analytic* and *dialectic* intricacy, very much the opposite of what sustained John Backus, and what is more, it serves to demonstrate, as a whole, the once and ever so unpreventable and the even and ever more so foreordained (typically Wittgenstein I) programming philosophy, including all programming languages that have ever lasted (supposedly non-von Neumann also included).

Being cognizant of different evaluation criteria in comparing 1<sup>st</sup> and 2<sup>nd</sup> generation computer languages – such as readability, with simplicity, orthogonality, control statements, and data types & structures, or writability, with abstraction and expressivity, or reliability, with error & exception handling, and type checking, besides things like development and maintenance cost – when comparing a strong static and manifest language as Fortran, with a procedural, imperative, and structured language like Algol, independently of all possible features compared, what is most remarkable, above all, is the necessary accordance of all possible judgements of *experience* in programming design of these *phenomenological* ideas in relation to its historical context. Sometimes, the relation is openly *divergent* as seen with both Prolog and C (1972), and periodically it is fairly *convergent*, asserting for the emergence of both Fortran (1957/58) and Algol (1958).

In comparing the ascendancy of Algol 60 to Prolog, and natural language processing above logic programming, it cannot be unrecalled the fact that Algol was, right up front, a metalanguage, thus introducing the seemingly natural language ability of "speaking about itself", and onwards also capable of describe programming languages, precisely due to the Backus-Naur form of syntax description. One form of its continuity, was, of course, the Van Wijngaarden grammar (else called W-grammar, and an example of the class of affix grammars), the technique that was used and developed in the definition of Algol 68 (1968), and has affected the *unification* algorithm in Prolog (1972).

Algol 60, in detail, is a paragon of typical Wittgenstein I philosophy, representing the ancestry of most of the actual modern (Algol-like) programming languages. A fairly untroubled and easily visible example of such is the main quote on the *Revised Report on the Algorithmic Language Algol 60* (1962) [200], directly reviving Wittgenstein I of the *Tractatus Logico-Philosophicus* (1922):

"What can be said at all, can be said clearly; and what you can not talk about is something to be silent about." [418, 419]

Algol 60 centers and warrants most of the implementations of Algol [339, 400, 360, 297, 172, 189, 325, 141, 109, 247] indisputably. Algol 68 [49, 371] was criticized by some members of its design committee such as C. A. R. Hoare and E. Dijkstra, for abandoning the simplicity of ALGOL 60, accusing the enterprise of a great complexity risk, while over demanding work from the compiler, but one such programming vehicle is obviously pertinent to anticipate Prolog's years, and not only Prolog's, but starting forthwith the C language also.

"Algol 60 was particularly influential in the design of later languages since it introduced nested block structure, lexical scope, and a syntax in Backus-Naur form (BNF). Nearly all subsequent programming languages have used a variant of BNF to describe context-free syntax" [371], but what is superlative in importance is that it took less than a decade to stimulate the next in importance "man-to-man communication" [257] innovative grammar – the Van Wijngaarden grammar in Algol 68 –, which settles Algol not only as the watermark of an epoch of (programming) philosophy of language, having launched scientific programming and

compiling studies, as in continuity, besides the fact that an extant *full* formal definition needed not a compiler to succeed, the confluence of programming languages design with a programming language on its own, was truly breaking new.

With this we agree with HT de Beer, author of *The History of the ALGOL Effort* (2006), who wrote: "I argue that the Algol effort was a catalyst for the transformation of the field of compiler writing and programming languages into a scientific field. The ALGOL 60 report was the key to this transformation and this explains the influence of the ALGOL 60 report; its scope was much wider than yet another algorithmic programming language, it was about the definition of programming languages in general." [97] If we remember that ever since Wegstein proposed three different levels of representation – "reference, publication, and hardware" [3] – surviving to Algol 60 [200] and Algol 68 [49], it is understandable how much a theory of meaning and the philosophical study of truth, in Tarski's sense, inviting formal correctness and material adequacy to the realm of programming languages – "structure and content must be the same for all representations" [200] – endeavoured the semantic explosion of programming languages to the exact extent that a well-defined syntax was constructed.

For reasons of economy, we shall now exemplify code fragments of both Algol 60 and Algol 68, and relate to ahead fragments of C/C++, Java, and Python, with this hoping to show, liminally and graphically, the inherent isomorphism of programming languages – program variables  $\iff$  computer storage cells (Wittgenstein I), control statements  $\iff$  computer instructions (Shannon), assignment statements  $\iff$  storing instructions (von Neumann), and expressions  $\iff$  arithmetic instructions and memory reference (Turing).

Observant as one should be, Algol 60 customarily presents the Backus-Naur form (BNF), while what can be seen in Algol 68 is the *application* of the two-level (context-free) grammar formalism of the van Wijngaarden grammar, paving the way to the very same *application* on Q-systems machine translation (and thereupon on Prolog). W-grammars and Algol 68 features, all together, have permitted the inclusion of an expression-based syntax, with user-declared types and tagged structures identified by its tag name and its set of field names. They have also transported a reference model of variables and reference parameters. Extracting strings, arrays and matrices, and partial-order execution in concurrency are far

more plastic features – congruent with the henceforward general characteristic of W-grammars in being able to encode the potentially infinite universe of context-free grammars in a finite set of rules – which bear witness of a pivotal stage in programming languages evolution, i.e., one such of rapid speciation in apparent contrast with "punctuated equilibrium" [150, 151] or *stasis*. With this we also mean a sort of language and informatics philogenetic atavism, once Algol-like languages influence was comprehensive and long lasting, although its traits were only identified when they reappeared in ahead developed programming languages. And we this we also mean something very close to a *morphological* informatics sense, not only because Algol [339, 400, 297] – a *collaborative* effort made by an international committee of the ACM (1958-1960) – inaugurated *structuring* code blocks, begin/end delimitations, and nested functions with a lexical scope, which allowed, like Lisp, recursive programming procedures, but also because it belongs to an era when, very fortunately, programming languages were designed without a clear grasp if they were really an interpreter language for the compiler, a compiler report, an actual programming language, or indeed a grammar programming language design meta-theory. In the case of Algol 68, it was established as a programming language, as a matter of fact, before the actual implementation.

From then on, it is history, as we often hear. Algol (1958-1968), one of the first four high-level programming languages – along with Fortran (1957), Lisp (1958) [154, 112, 5, 233, 353, 159], and Cobol (1959) – may assume a superlative importance in comparison, by virtue of its sheltered, innovative, formal grammar evolution, a philosophy of language inner revolution in computer science (BNF to W-Grammars), having, thus, definitely laid the foundations for the proper phenomenon of heritable characteristics of programming languages (informatic) types and paradigms – in relation to arithmetic expressions, statements, blocks, and programs, relations and conditionals, etc. – adaptatively emerging, as a special axis, from both the coevolution of Fortran (1957) and Lisp (1958), as, in becoming extinct, influencing on the speciation of C [295] (1972) and Prolog [373, 352, 131, 39, 85, 175, 290, 126, 284, 348, 340, 142, 324, 361, 23, 171, 47, 246, 192, 270, 388] (1972).

Conclusively, and in comparison, a genetic (informational) programming *synthesis* significantly impacts on coding (programming languages), decoding (inter-

preters and compilers), and more rarely so, as it is the case of Algol, on expressions (formal grammar). Often unseen and maybe even disregarded, is, what is more, regulation (the programming language's *message* with the *messenger*) that Algol has induced. This is so to the point of, insofar recursive subprogram-procedures were part of Lisp (1957) and Algol (1958), both C and Prolog (1972), as Python [372] (1990) and Java [76] (1995), share a natural-to-artificial isomorphism:

```

1 BEGIN
2
3 COMMENT define the sieve data structure ;
4 INTEGER ARRAY candidates[0:1000];
5 INTEGER i,j,k;
6 COMMENT 1000 to protect against strict evaluation of AND ;
7 FOR i := 0 STEP 1 UNTIL 1000 DO
8 BEGIN
9 COMMENT everything is potentially prime until proven otherwise ;
10 candidates[i] := 1;
11 END;
12 COMMENT Neither 1 nor 0 is prime, so flag them off ;
13 candidates[0] := 0;
14 candidates[1] := 0;
15 COMMENT start the sieve with the integer 0 ;
16 i := 0;
17 FOR i := i WHILE i<1000 DO
18 BEGIN
19 COMMENT advance to the next un-crossed out number. ;
20 COMMENT this number must be a prime ;
21 FOR i := i WHILE i<1000 AND candidates[i] = 0 DO
22 BEGIN
23 i := i+1;
24 END;
25 COMMENT insure against running off the end of the data structure ;
26 IF i<1000 THEN
27 BEGIN
28 COMMENT cross out all multiples of the prime, starting with 2*p.;
29 j := 2;
30 k := j*i;
31 FOR k := k WHILE k < 1000 DO
32 BEGIN
33 candidates[k] := 0;
34 j := j + 1;
35 k := j*i;
36 END;
37 COMMENT advance to the next candidate ;
38 i := i+1;
39 END
40 END;

```

```

41 COMMENT all uncrossed-out numbers are prime (and only those numbers)
    ;
42 COMMENT print all primes ;
43 FOR i := 0 STEP 1 UNTIL 999 DO
44 BEGIN
45 IF candidates[i] # 0 THEN
46 BEGIN
47 write(1,i);
48 text(1," is prime*N")
49 END
50 END;
51 END
52 FINISH

```

Listing 2.4: Algol 60 programming example of the sieve of Eratosthenes

Now follows the sieve of Eratosthenes programmed in ahead different fragments of the programming languages C/C++, Java, and Python:

```

1 // C++ program to print all primes smaller than or equal to
2 // n using Sieve of Eratosthenes
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 void SieveOfEratosthenes(int n)
7 {
8     // Create a boolean array "prime[0..n]" and initialize
9     // all entries it as true. A value in prime[i] will
10    // finally be false if i is Not a prime, else true.
11    bool prime[n+1];
12    memset(prime, true, sizeof(prime));
13
14    for (int p=2; p*p<=n; p++)
15    {
16        // If prime[p] is not changed, then it is a prime
17        if (prime[p] == true)
18        {
19            // Update all multiples of p
20            for (int i=p*2; i<=n; i += p)
21                prime[i] = false;
22        }
23    }
24
25    // Print all prime numbers
26    for (int p=2; p<=n; p++)
27        if (prime[p])
28            cout << p << " ";
29 }
30

```

```

31 // Driver Program to test above function
32 int main()
33 {
34     int n = 30;
35     cout << "Following are the prime numbers smaller "
36         << " than or equal to " << n << endl;
37     SieveOfEratosthenes(n);
38     return 0;
39 }

```

Listing 2.5: C/C++ programming example of the sieve of Eratosthenes

```

1 // Java program to print all primes smaller than or equal to
2 // n using Sieve of Eratosthenes
3
4 class SieveOfEratosthenes
5 {
6     void sieveOfEratosthenes(int n)
7     {
8         // Create a boolean array "prime[0..n]" and initialize
9         // all entries it as true. A value in prime[i] will
10        // finally be false if i is Not a prime, else true.
11        boolean prime[] = new boolean[n+1];
12        for(int i=0;i<n;i++)
13            prime[i] = true;
14
15        for(int p = 2; p*p <=n; p++)
16        {
17            // If prime[p] is not changed, then it is a prime
18            if(prime[p] == true)
19            {
20                // Update all multiples of p
21                for(int i = p*2; i <= n; i += p)
22                    prime[i] = false;
23            }
24        }
25
26        // Print all prime numbers
27        for(int i = 2; i <= n; i++)
28        {
29            if(prime[i] == true)
30                System.out.print(i + " ");
31        }
32    }
33
34    // Driver Program to test above function
35    public static void main(String args [])
36    {
37        int n = 30;

```

```

38     System.out.print("Following are the prime numbers ");
39     System.out.println("smaller than or equal to " + n);
40     SieveOfEratosthenes g = new SieveOfEratosthenes();
41     g.sieveOfEratosthenes(n);
42 }
43 }
44 // This code has been contributed by Amit Khandelwal.

```

Listing 2.6: Java programming example of the sieve of Eratosthenes

```

1 # Python program to print all primes smaller than or equal to
2 # n using Sieve of Eratosthenes
3
4 def SieveOfEratosthenes(n):
5
6     # Create a boolean array "prime[0..n]" and initialize
7     # all entries it as true. A value in prime[i] will
8     # finally be false if i is Not a prime, else true.
9     prime = [True for i in range(n+1)]
10    p = 2
11    while (p * p <= n):
12
13        # If prime[p] is not changed, then it is a prime
14        if (prime[p] == True):
15
16            # Update all multiples of p
17            for i in range(p * 2, n+1, p):
18                prime[i] = False
19            p += 1
20
21    # Print all prime numbers
22    for p in range(2, n):
23        if prime[p]:
24            print p,
25
26 # driver program
27 if __name__ == '__main__':
28     n = 30
29     print "Following are the prime numbers smaller",
30     print "than or equal to", n
31     SieveOfEratosthenes(n)

```

Listing 2.7: Python programming example of the sieve of Eratosthenes



# Bibliography

- [1] *A Teoria Matemática da Comunicação de Shannon (Portuguese)*. (Universidade Federal do Paraná).
- [2] *"Nets and filters (are better than sequences)." (English)*. (Wichita State University).
- [3] "Preliminary report – International Algebraic Language".(English). *Comm. Assoc. Comp. Mach*, 1(12), 1958.
- [4] "Report on the Algorithmic Language Algol by the ACM Committee on Programming Languages and the GAMM Committee on Programming".(English). (Edited by A. J. Perlis and K. Samuelson) *Numerische Mathematik Bd.*, 1(41-60), 1959.
- [5] *Learn Lisp List Processing. (English)*. (Tutorials Point Simple Easy Learning). 2014.
- [6] "Borel measures". (English). (North Dakota State University) *Math 752*, 2015.
- [7] I. A. Abramowitz, M. ; Stegun. "Miscellaneous Functions". (English). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, Tenth Printing, with corrections*, Chapter 1-27:9–226, 1972.
- [8] Kim Thakkar Yeddanapudi Aftab, Cheung. "Information Theory & the Digital Revolution".(English). *6.933 Project History, Massachusetts Institute of Technology*.
- [9] Agostinho. *O Mestre (Portuguese)*. (Porto Editora, Filosofia Textos).

- [10] Jeese Alama. "The Lambda Calculus", The Stanford Encyclopedia of Philosophy (Fall 2017 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/fall2017/entries/lambda-calculus/>.. (English).
- [11] Enrique Alonso and Maria Manzano. "Diagonalisation and Church's Thesis: Kleene's Homework". (English). (*Universidad Autónoma de Madrid; Universidad de Salamanca*) *History and Philosophy of Logic*, 2005.
- [12] Jörg Arndt. "*Matters Computational, Ideas, Algorithms, Source Code*" (English). (*Springer*). 2011.
- [13] Augustine. *De Doctrina Christiana* (English). (R. P. H. Green Editor, *University of Glasgow*). 1996.
- [14] Augustine. "De Musica". (English). (*translated by Jean-François Thénard and Marc Citoleux, with a preface by Anne-Isabelle Bouton-Touboulic, Paris: Éditions du Sandre*) *Journal of Plainsong and Medieval Music*, 16(1), 2006.
- [15] Hassan Aït-Kaci. "*Warren's Abstract Machine, a Tutorial Reconstruction*". (English). (*Reprinted from MIT version; Intelligent Software Group, School of Computing Science, Simon Fraser University, British Columbia, Canada.*). 1999.
- [16] Carl J. Posy B. Jack Copeland and Oron Shagrir (Editors). *Computability, Turing, Gödel, Church, and Beyond* (English). (*The MIT Press, Cambridge, Massachusetts, London, England*). 2015.
- [17] H. P. Babbage. "The Analytical Babbage".(English). (*Bath*) *Proceedings of the British Association*, 1888.
- [18] Babette Babich. "Mousike techne: The Philosophical Practice of Music in Plato, Nietzsche, and Heidegger". (English). *Articles and Chapters in Academic Book Collections*, Paper 23, 2005.

- [19] John Backus. "Can Programming be Liberated from the von Neumann Style? a Functional Style and Its Algebra of Programs". (English). (*IBM Research Laboratory, San Jose*) *Communications of the ACM*, 21(8), 1978.
- [20] Sanjeev Arora; Boaz Barak. "*Computational Complexity: A Modern Approach.*" (English).(*Princeton University*). January 2007.
- [21] Henk Barendregt. "*The impact of the lambda calculus in logic and computer science*" (English). (*Computing Science Institute, Nijmegen University, The Netherlands*). 1997.
- [22] Avron Barr and Edward A. Feigenbaum (Editors). *The Handbook of Artificial Intelligence (English).*, volume 2 (5). 2014.
- [23] Roman Barták. "*Guide to Prolog Programming.*" (English).(*Faculty of Mathematics and Physics, Charles University, Prague*). 1998.
- [24] Dave Barker-Plummer; Jon Barwise, Michael Murray John Etchemendy, in collaboration with Albert Liu, and Emma Pease. "*Language, Proof and Logic.*" (English). (*Second Edition, CSLI Publications Center for the Study of Language and Information Leland Stanford Junior University*). 2000 (1999).
- [25] J. Barwise. "*Completeness and Deduction Theorem for Classical Predicate Logic*" (English). (*Chapter 9, Handbook of Mathematical Logic*). 1990 (1977).
- [26] F.L. Bauer and H. Wössner. "The 'Plankalkul' of Konrad Zuse: A Forerunner of Today's Programming Languages.".(English). (*Mathematisches Institut der Technischen Universität München*) *Communicatins of the ACM*, 15(7), 1972.
- [27] David Beach. *Aspects of Schenkerian Theory (English)*. (*New Haven: Yale University Press*). 1983.
- [28] Claude Berge. "*Topological Spaces.*" (English).(*Dover Publications, Inc. Mi-neola New York*). *Translated by E. M. Patterson*. 1962, 1963.

- [29] Henry Bergson. *Matter and Memory (English)*. (Translated by Nancy Margaret Paul and W. Scott Palmer, Dover Publications, Mineola, New York). (1912) 2004.
- [30] Henry Bergson. *A Evolução Criadora (Portuguese)*. (Tradução de Pedro Elói Duarte, Edições 70). 1941.
- [31] Henry Bergson. *The Creative Mind, an Introduction to Metaphysics (English)*. (Translated by Mabelle L. Andison, Dover Publications, Mineola, New York). 1946.
- [32] Henry Bergson. *A Intuição Filosófica (Portuguese)*. (Tradução, introdução e notas de Maria do Céu Patrão Neves, Edições Colibri, Universalia). 1994.
- [33] Henry Bergson. *Time and Free Will, An Essay on the Immediate Data of Consciousness (English)*. (Authorized Translation by F. L. Pogson, M.A., Dover Publications, Mineola, New York). 2001.
- [34] George Berkeley. *A Treatise Concerning the Principles of Human Knowledge to which are added Three Dialogues Between Hylas and Philonous. (English)*. (London, Printed for Jacob Tonson, First Printed in the year 1713). 1734.
- [35] George Berkeley. *"The Analyst, or a Discourse Addressed to an Infidel Mathematician."* (English). (Edited by David R. Wilkins). 2002.
- [36] Chris Bernhardt. *"Turing's Vision, The Birth of Computer Science."* (English). (The MIT Press; Cambridge, Massachusetts; London, England). 2016-17.
- [37] David Carson Berry. *A Topical Guide to Schenkerian Literature: An Annotated Bibliography with Indices (English)*. (Hillsdale, NY: Pendragon Press). 2004.
- [38] Anat Biletzki and Anat Matar; Edward N. Zalta (ed.). "Ludwig Wittgenstein". (English). *The Stanford Encyclopedia of Philosophy*, 2016.
- [39] Patrick Blackburn and Kristina Striegnitz. *"Natural Language Processing Techniques."* (English). (Version 1.2.4 20020829.).

- [40] Andreas Blass and Yuri Gurevich. "Algorithms: a Quest for Absolute Definitions". (English). (*Mathematics Department, University of Michigan, Ann Arbor, MI 48109-1109*), pages 1–30.
- [41] Ernst Bloch. *The Principle of Hope*. (English). (Translated by Neville Plaice, Stephen Plaice and Paul Knight, *The MIT Press, Cambridge, Massachusetts*), volume I, II, III. 1986.
- [42] Boethius. *De Institutione Musica* (Latin, Italian). (*Istituto italiano per la storia della musica in Roma*). 1990.
- [43] George Boole. "*An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*". (English). (*London, Walton and Maberly, Cambridge, Macmillan and Co.*). 1854.
- [44] Bordas-Demoulin. "*Le Cartésianisme ou La Véritable Rénovation Des Sciences suivi de La Théorie de la Substance et de celle de l'Infini précédé d'un Discours sur la Réformation de la Philosophie au Dix-Neuvième siècle: Tome Premier I, Paris: J. Hetzel, Libraire-Éditeur.*" (French). 1843.
- [45] Max Bramer. *Logic Programming with Prolog*. (English). (*Springer*). 2005.
- [46] Ivan Bratko. *Prolog Programming for Artificial Intelligence*. (English). (*Addison Wesley, an imprint of Pearson, Fourth Edition*). 1986, 1990, 2001 (2012).
- [47] Paul Brna. "*Prolog Programming: A First Course.*" (English). (*School of Informatics, The University of Edinburgh*). 1999.
- [48] Mario Bunge. "A General Black Box Theory". (English). (*The Philosophy of Science Association*) *The University of Chicago Press Journals*, 30(4), 1963.
- [49] Edited by A. van Wijngaarden; B.J. Mailloux; J.E.L. Peck; C.H.A. Koster; M. Sintzoff; C.H. Lindsey; L.G.T. Meertens and R.G.Fisker. "Revised Report on the Algorithmic Language Algol 68 ". (English) (*mathematisch centrum amsterdam*). *Mathematical Centre Tracts*, 50, 1976.

- [50] Edited by Alan J. Perlis. "Communications of the ACM". (English). (*Carnegie Institute of Technology, Pittsburgh, PA*) *CACM Communications of the ACM*, 2(12), 1959.
- [51] Edited by Charles Jones. *Historical Linguistics Problems and Perspectives* (English). (*Longman Linguistics Library*). 1993.
- [52] Edited by ICOT. "Proceedings of the International Conference on Fifth Generation Computer Systems 1984" (English). (*Edited by Institute for New Generation Computer Technology (ICOT), North-Holland*), 1984.
- [53] Edited by Luciano Floridi. "*Philosophy of Computing and Information*" (English). (*Blackwell Publishing, Blackwell Philosophy Guides*). 2004.
- [54] María Manzano (Translated by Ruy De Queiroz). "*Model Theory*." (English). (*Oxford Logic Guides*). 29 April, 1999.
- [55] Edited by Stewart Shapiro. "*The Oxford Handbook of Philosophy of Mathematics and Logic*" (English). (*Oxford University Press*). 2005.
- [56] Oliver Byrne. "*The First Six Books of The Elements of Euclid*" (English). (*Taschen Books*). 1847 (2017).
- [57] Giovanni Camardi. "Charles Lyell and the Uniformity Principle". (English). *Biology and Philosophy*, 14(4):537–560, October, 1999.
- [58] Luca Cardelli. "*Typeful Programming*". (English) (*Digital Equipment Corporation, Systems Research Center 130 Lytton Avenue, Palo Alto, CA 94301*). 1991 (Revised 1993).
- [59] Rudolf Carnap. "*The Logical Syntax of Language*." (English). (*International Library of Psychology, Philosophy, and Scientific Method. Translated from the German by Amethe Smeaton; New York, Harcourt, Brace*). 1937.
- [60] Frank Carter. "Colossus and the Breaking of the Lorenz Cipher". (English). (*Bletchley Park Trust; Milton Keynes, United Kingdom*).

- [61] Ernst Cassirer. *An Essay on Man, an Introduction to a Philosophy of Human Culture (English)*. (Doubleday & Company Inc., Garden City, New York, and Yale University Press). 1944.
- [62] Augustin Louis Cauchy. "*Résumé des Leçons données à l'École Royale Polytechnique sur le calcul infinitésimal.*" (French). (Paris, Ouvres, séries 2), volume 4. 1823.
- [63] Howard Caygill. *A Kant Dictionary. (English)*. (Blackwell Philosopher Dictionaries). 1995.
- [64] Waldemar Celes, Renato Cequeira, and José Lucas Rangel. "*Introdução a Estruturas de Dados com Técnicas de Programação em C.*" (Portuguese). (Elsevier, Série Editora Campus, SBC). 2004.
- [65] Zhenjie Chen. "History on the implementation and compilation of Prolog". (English). pages 1–10, December 4, 2012.
- [66] Adam Chlipala. "*Certified Programming with Dependent Types.*" (English). (MIT Press). July 12, 2017.
- [67] Noam Chomsky. "Three Models for the Description of Language". (English). *IRE Transactions on Information Theory*, vol. 2; issue 3: pp. 113–124, 1956.
- [68] Noam Chomsky. "*The Architecture of Language*" (English). (Edited by Nir-malangshu Mukkherji, Bibudhendra Narayan Patnaik, Rama Kant Agnihotri, Oxford India Paperbacks). 2000 (2006).
- [69] Noam Chomsky and M. P. Schützenberger. "The Algebraic Theory of Context-Free Languages". (English). (Elsevier) *Studies in Logic and the Foundations of Mathematics*, 35:118–161, 1963.
- [70] Henning Christiansen and Veronica Dahl. "*Assumptions and Abduction in Prolog*". (English) (Roskilde University, Computer Science Dept.; Dept. of Computer Science, Simon Fraser University, Burnaby, Canada).

- [71] Alonzo Church. "An Unsolvable Problem of Elementary Number Theory". (English). (*The Johns Hopkins University Press*) *American Journal of Mathematics*, 58(2):345–363, 1936.
- [72] Alonzo Church. "*Introduction to Mathematical Logic*" (English). (Princeton, New Jersey, *Princeton University Press*), volume I. 1956.
- [73] C.F. Clocksin and C.S. Mellish. *Programming in Prolog, Fourth Edition*. (English) *Springer*. 1994.
- [74] Alan Cobham. "The intrinsic computational difficulty of functions". (English). (*North-Holland*) *Proc. Logic, Methodology, and Philosophy of Science*, II, 1965.
- [75] Philippe Codognet and Daniel Diaz. "wamcc: Compiling Prolog to C". (English). (*INRIA-Rocquencourt, Domaine de Voluceau 78153 Le Chesnay, France*).
- [76] Pedro Coelho. *Programação em Java, Curso Completo*. (Portuguese). (5ª edição actualizada, *FCA*). 2016.
- [77] Alain Colmerauer. "*Les systèmes Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur.*" (French). (*Mimeo, Montréal*). 1969.
- [78] Alain Colmerauer. "From Natural Language Processing to Prolog". (English). (<http://alain.colmerauer.free.fr>). *Beijing*, April 8, 2011.
- [79] Alain Colmerauer and Philippe Roussel. "The Birth of Prolog". (English). (Groupe Intelligence Artificielle, Unité de Resherche Associée au CNRS 816, Faculté des Sciences de Luminy). *Communications of the ACM*, pages 2–27, November 1992.
- [80] Stephen Cook. "The P VERSUS NP Problem". (English). (*Clay Mathematics Institute*), pages pp. 1–11.
- [81] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". (English). (*University of Toronto*) *Proceedings of the third annual ACM symposium on theory of computing*, pages 151–158, 1971.



- [82] William R. Cook. *"Anatomy of Programming Languages."* (English). (UT Computer Science). 2013.
- [83] Nicolaus Copernicus. *"On The Revolutions of the Heavenly Spheres."* (English). (Translated by Charles Glenn Wallis, Great Mind Series, Prometheus Books). 1995 (1453).
- [84] Bruno Courcelle. "Fundamental Properties of Infinite Trees". (English). (UER de Mathématiques et Informatique, Université Bordeaux-I, 33405, Talence, France) *Theoretical Computer Science*, 25:pp. 95–169, 1983.
- [85] Michael Covington. *"Natural Language Processing for Prolog Programmers."* (English). (Prentice-Hall). 1994.
- [86] Carol Critchlow and David Eck. *"Foundations of Computation."* (English). (Department of Mathematics and Computer Science Hobart and William Smith Colleges, Geneva, New York). Summer 2011.
- [87] Adrian Crăciun. "Computer Architecture". (English). 2017.
- [88] Thomas Crump. *Science, As Seen Through The Developments of Scientific Instruments.* (English). Constable & Robinson. 2002.
- [89] L. Byrd; F. C. N. Pereira; L. M. Pereira; D. H. D. Warren D. L. Bowen (editor). "DECsystem-10 PROLOG USER'S MANUAL. Prolog Version 3.47" (English). (University of Edinburgh, Department of Artificial Intelligence), 1982.
- [90] António M. Amorim da Costa. *Introdução à História e Filosofia das Ciências.* (Portuguese). Publicações Europa-América. 1986.
- [91] João Esteves da Silva. *Cinco ensaios sobre Wittgenstein.* (Portuguese). (Cadernos de Filosofia das Ciências, CFCUL). 2010.
- [92] Ashish Dalela. *"Gödel's Mistake, The Role of Meaning in Mathematics."* (English). (Shabda Press). 2014.

- [93] Charles Darwin. *The Complete Work of Charles Darwin Online*. (<http://darwin-online.org.uk/>). (English). John van Wyhe, Editor. 2002.
- [94] Charles Darwin. *The Origins of Species by Means of Natural Selection*. (English). Chapter I. London, Murray: Second Edition. 1859; 1860.
- [95] Charles H. Lineweaver; Paul C. W. Davies and Michael Ruse (ed.). "*Complexity and the Arrow of Time*." (English). (Cambridge University Press). 2013.
- [96] Martin Davis. "*The Universal Computer. The Road from Leibniz to Turing*" (English). (CRC Press Taylor & Francis Group). 1968-1994.
- [97] HT de Beer. "ALGOL, More than just ALGOL". (English). (Eindhoven), 2008.
- [98] Philippe de La Cotardière (Direction). *Histoire des Sciences: Editions Talandier* (French). 2004.
- [99] Ferdinand de Saussure. *Course in General Linguistics*. (English) Duckworth. p. 67. 1983.
- [100] Gilles Deleuze. "*Difference and Repetition*." (English). (Translated by Paul Patton; Columbia University Press). 1968 - 1994.
- [101] Gilles Deleuze. "*Cinema 1 The Movement-image*" (English). (Translated by Hugh Tomlison and Barbara Habberjam, University of Minnesota Press). 1997 (1983).
- [102] Gilles Deleuze. "*Cinema 2 The Time-image*" (English). (Translated by Hugh Tomlison and Robert Galeta, University of Minnesota Press). 1997 (1985).
- [103] Gilles Deleuze. "*Cinema 2 A Imagem-Tempo*" (Portuguese). (tradução de Sousa Dias, Sistema Solar (Documenta)). 2015 (1985).
- [104] Gilles Deleuze. "*Cinema 1 A Imagem-Movimento*" (Portuguese). (tradução de Sousa Dias, Sistema Solar (Documenta)). 2016 (1983).

- [105] Daniel Diaz. *"A Native Prolog Compiler with Constraint Solving over Finite Domains"*. (English). (Edition 1.44, for GNU Prolog version 1.4.4). April 23, 2013.
- [106] Daniel Diaz and Philippe Codognet. "A minimal extension of the WAM for clp(FC)". (English). (*INRIA-Rocquencourt, Domaine de Voluceau 78153 Chesnay, France*).
- [107] Daniel Diaz and Philippe Codognet. "GNU Prolog: beyond compiling Prolog to C.". (English). (*University of Paris 1, CRI, bureau C1407, 90 rue de Tolbiac, 75013, Paris, France*).
- [108] Pierre Wagner (Direction). *Les Philosophes et la science* (French). Gallimard. 2002.
- [109] Pedro Domingos. *"The Master Algorithm, How the Quest for the Ultimate Learning Machine will Remake our World."* (English). (*Penguin Books*). 2015.
- [110] Oswald Ducrot and Tzvetan Todorov. *Dicionário das Ciências da Linguagem* (Portuguese). (*Edição orientada por Eduardo Prado Coelho, Dom Quixote*). 2004 (1973).
- [111] David Sands; Jeremy Dunnig-Daves. "Reinterpreting Boltzmann's H-Theorem in the light of Information Theory.". (English). (*Department of Physics and Mathematics, University of Hull, UK; Institute for Theoretical Physics and Advanced Mathematics Einstein-Galilei, Prato, Italy*), 2013.
- [112] Vsevolod Dyomkin. *Lisp Hackers Interviews with 100x More Productive Programmers conducted by Vsevolod Dyomkin in 2012-2013*. (English). (*Leanpub Book*). 2013.
- [113] Alfred Tarski (ed.). "The Concept of Truth in Formalized Languages", Logic, Semantics, Metamathematics. (English). *Oxford University Press*, 322: pp.152–278, 1936.

- [114] Lawrence Pasternack; Philip Rossi (Edward N. Zalta ed.). "Kant's Philosophy of Religion". (English). *The Stanford Encyclopedia of Philosophy*, (Fall 2014 Edition).
- [115] L.E.J. Brouwer. E.W. Beth. A. Heyting (ed.). "The Algebraic Theory of Context-Free Languages in Computer Programming and Formal Systems". (English). *North-Holland Publishing Company, Amsterdam*, pages 181–161, 1963.
- [116] Richard Amesbury (Edward N. Zalta ed.). "Fideism". (English). *The Stanford Encyclopedia of Philosophy*, (Fall 2017 Edition).
- [117] Thierry Coquand; Edward N. Zalta (ed.). "Type Theory". (English). *The Stanford Encyclopedia of Philosophy*, (Summer 2015 Edition).
- [118] Victor Rodych Edward N. Zalta (ed.). "Wittgenstein's Philosophy of Mathematics". (English). (*The Stanford Encyclopedia of Philosophy*), pages 347–352, 2018.
- [119] William B. Ewald (ed.). "*From Immanuel Kant to David Hilbert: A Source Book in the Foundations of Mathematics.*" (English). (*Oxford University Press*)., volume Volume 2. 1996.
- [120] Robert A. Kowalski (J.J. Horing Editor). "Algorithm = Logic + Control". (Imperial College, London). (English). *Communications of the ACM*, 22(7):424–436, July 1979.
- [121] Thomas S. Khun (editor-in-chief Otto Neurath). "*The Structure of Scientific Revolutions.*" (English). (*International Encyclopedia of Unified Science, Second Edition enlarged by the University of Chicago*), volume II. 1962-1970.
- [122] Yasuh Suzuki; Toshiyuki Nakagaki (editors). "*Natural Computing and Beyond .*" (English). (*Winter School Hakodate 2011, Hakodate, Japan, March 2011 and 6th International Workshop on Natural Computing, Tokyo, Japan, March 2012*). 2011, 2012.
- [123] Enciclopedia Einaudi. *16 volumi; 1977-1984. (Italian).*

- [124] Albert Einstein. *"Volume 3: The Swiss Years: Writings 1909-1911."* (English). (URL:<http://einsteinpapers.press.princeton.edu/>). 1910.
- [125] Robert A. Kowalski; M. H. Emden. "The Semantics of Predicate Logic as a Programming Language". (University of Edinburgh, Scotland). (English). *Journal of the Association for Computing Machinery*, 23(4):733–742, October 1976.
- [126] Ulle Endriss. *"Lecture Notes An Introduction to Prolog Programming."* (English). (University of Amsterdam, Institute for Logic, Language and Computation). 2006.
- [127] David Evans. *"Introduction to Computing, Explorations in Language, Logic, and Machines."* (English). (University of Virginia, Creative Commons). August 19, 2011.
- [128] Howard Eves. *An Introduction to the History of Mathematics The Saunders Series (f-Felix Klein and the Erlanger Program)* (English). Sixth Edition. 1990.
- [129] Bernard Feltz. *La Science et le vivant. Philosophie des sciences et modernité critique* (French). (de Boeck). ISBN 978-2-8041-7144-5. 2014.
- [130] Richard Feynman. *"Caltech's Division of Physics, Mathematics and Astronomy and The Feynman Lectures Website"* (English). (URL:<http://www.feynmanlectures.caltech.edu/>). Copyright® 1963, 2006, 2013 by the California Institute of Technology.
- [131] Peter Flach. *"Simply Logical, Intelligent Reasoning by Example."* (English). (University of Bristol, United Kingdom). 1994-2007.
- [132] H. Graham Flegg. *"From Geometry to Topology."* (English). (Dover Publications, Inc. Mineola New York). 1974.
- [133] Antony Flew. *A Dictionary of Philosophy.* (English). (New York: St Martin's Press, Revised 2<sup>nd</sup> Edition). 1984.

- [134] Gottlob Frege. *"Begriffsschrift a formula language, modeled upon that of arithmetic, for pure thought."* (English). 1879.
- [135] Gottlob Frege. *"On Sense and Reference"*. (English). (translation by Max Kölbel, London: Routledge). 2009.
- [136] Daniel P. Friedman, Mitchell Wand, and Cristopher T. Haynes. *"Essentials of Programming Languages."* (English). (The MIT Press). 2001.
- [137] Felice Cardone; J. Roger Hindley (Eds. D. M. Gabbay and J. Woods). *"Lambda-Calculus and Combinators in the XX<sup>th</sup> Century"*. (English). (Elsevier Co. North-Holland, Amsterdam) *Handbook of the History of Logic Chapter 14*, 5:pp. 723–817, 2009.
- [138] Haim Gaifman. *"Naming and Diagonalization, from Cantor to Gödel to Kleene"*. (English). *Logic Journal of the IGPL*, Volume 14(Issue 5):709–728, 1 October 2006. DOI: <https://doi.org/10.1093/jigpal/jzl006>.
- [139] Brian R. Gaines. *"Perspectives on Fifth Generation Computing"*. (English). (Department of Computer Science, York University; Department of Industrial Engineering, University of Toronto; Department of Computer Science, University of Calgary).
- [140] Jean Gallier. *"Logic for Computer Science: Foundations of Automatic Theorem Proving."* (English). (Originally printed by Wiley.). 1986.
- [141] J. G. Ganascia. *"A Inteligência Artificial."* (Portuguese). (Biblioteca Básica de Ciência e Cultura, Instituto Piaget). 1993.
- [142] Gerald Gazdar and Chris Mellish. *"Natural Language Processing in PROLOG: An Introduction to Computational Linguistics."* (English). (Addison Wesley Publishing Company.). 1989.
- [143] Ernst Gellner. *Linguagem e Solidão*. (Portuguese). (Título original: *Language and Solitud Wittgenstein, Malinowski and the Habsburg Dilemma; Edições 70*). (1998) 2001.

- [144] Hans-Johann Glock. *Dicionário Wittgenstein. (Portuguese). (Jorge Zahar Editor; Título original: A Wittgenstein Dictionary, Blackwell Publishers 1996).* 1997.
- [145] Oded Goldreich. *"P, NP, and NP-Completeness: The Basics of Complexity Theory."* (English). (Cambridge University Press). August 2010.
- [146] Oded Goldreich. *"Computational Complexity: A Conceptual Perspective."* (English). (Cambridge University Press). May 2008.
- [147] Rebecca Goldstein. "Gödel And The Nature Of Mathematical Truth". (English). (*Edge*), 2005.
- [148] Rebecca Goldstein. *"Incompletude, A demonstração e o Paradoxo de Kurt Gödel."* (Portuguese). (*Gradiva*). 2009 (2005).
- [149] Herman H. Goldstine. *"The Computer from Pascal to von Neumann."* (English). Princeton University Press. 1972.
- [150] Stephen Jay Gould. "Ontogeny and Phylogeny". (English). (*Cambridge Mass.: Belknap Press of Harvard University Press*), 1997.
- [151] Stephen Jay Gould. *Punctuated Equilibrium.* (English). (*The Belknap Press of Harvard University Press*). 2007.
- [152] Judith V. Grabiner. "Who Gave You the Epsilon? Cauchy and the Origins of Rigorous Calculus". (English). (424 West 7th Street, Claremont California 91711.). *The American Mathematical Monthly.*, 90(3):185–194, 1983.
- [153] Paul Graham. *On Lisp, Advanced Techniques for Common Lisp.* (English). (*Prentice Hall*). 1993.
- [154] Paul Graham. *Ansi Common Lisp.* (English). (*Prentice Hall Series in Artificial Intelligence*). 1996.
- [155] Brian Greene. *"The Fabric of the Cosmos: Space, Time, and the Texture of Reality."* (English). (*Alfred A. Knopf*). 2004.

- [156] Theodore W. Gamelin; Robert Everist Greene. *"Introduction to Topology."* (English). (Dover Publications, Inc. Mineola New York). Second Edition. 1983, 1999.
- [157] Hans Herlof Grelland. "Wittgenstein's Picture Theory of Language as a Key to Modern Physics".(English). (Norway).
- [158] John Gribbin. *Science - A History 1543-2001.* (English). (Allen Lane, First Edition). 2002 (1962).
- [159] Richard P. Gabriel Guy Lewis Steele Jr. *The Evolution of Lisp.* (English). (Proceeding HOPL-II, The second ACM SIGPLAN conference on History of programming languages). 1993.
- [160] Kurt Gödel. *"Kurt Gödel Obras Completas ."* (Spanish). (Edición de Jesús Mosterín. Alianza Editorial). (1968 by Princeton University Press) Alianza Editorial (1981, 1989, 2006).
- [161] Kurt Gödel. *"Kurt Gödel Collected Works."* (English). (Oxford University Press · New York Clarendon Press). Edited by Solomon Feferman (Editor-in-chief) John W. Dawson, Jr. Stephen C. Kleene Gregory H. Moore Robert M. Solovay Jean van Heijenoort., volume I Publications 1929-1936. (1986).
- [162] Kurt Gödel. *"Kurt Gödel Collected Works."* (English). (Oxford University Press · New York Clarendon Press). Edited by Solomon Feferman (Editor-in-chief) John W. Dawson, Jr. Stephen C. Kleene Gregory H. Moore Robert M. Solovay Jean van Heijenoort., volume II Publications 1938-1974. (1990).
- [163] Kurt Gödel. *"Kurt Gödel Collected Works."* (English).(Oxford University Press · New York Clarendon Press). Edited by Solomon Feferman (Editor-in-chief) John W. Dawson, Jr. Stephen C. Kleene Gregory H. Moore Robert M. Solovay Jean van Heijenoort., volume III Unpublished Essays and Lectures. (1995).
- [164] Kurt Gödel. "An Example of a New Type of Cosmological Solutions of Einstein's Field Equations of Gravitation", Review of Modern Physics. (English). 21(3):447-450, July 1949.



- [165] Ernst Haeckel. *"Kunstformen der Natur."* (German). (HTML-Version herausgegeben von Kurt Stüber, 1999). 1899-1904.
- [166] Stefan Hagel. *Ancient Greek Music, a New Technical History.* (English). Cambridge University Press, 2010.
- [167] John Haigh. *"Probability, A Very Short Introduction."* (English). (Edited by Oxford University Press). 2012.
- [168] A. Rupert Hall. *The Revolution in Science, 1500-1750. 3rd Edition.* (English). Boston Beacon Press. 1996.
- [169] P. R. Halmos. "The Legend of John von Neumann". (English). *Enciclopedia Britannica* (Editor).
- [170] Paul R. Halmos. "von Neumann on Measure and Ergodic Theory". (English). *University of Chicago and Institute for Advanced Study*, pages 86–94, Received by the editors October 28, 1957.
- [171] Peter Hancox. *"Prolog and Logic Programming."* (English). (School of Computer Science, University of Birmingham). 2007.
- [172] Chris Hankin. *"An Introduction to Lambda Calculi for Computer Scientists."* (English). (Texts in Computing Series Editor, Imperial College London), volume 2. 2004.
- [173] Robert Harper. *"Practical Foundations for Programming Languages."* (English). ([Version 1.32.] Carnegie Mellon University). 05.15.2012.
- [174] Richard Hartley and Andrew Zisserman. *Multiple View Geometry.* (English). (Second Edition, Cambridge University Press). 2003.
- [175] James L. Hein. *"Prolog Experiments in Discrete Mathematics, Logic, and Computability."* (English). (Portland State University). 2009.
- [176] Maurice Heins. *Complex function theory: Academic Press; J. Hetzel.* (English). 1968.

- [177] Whitfield Diffie & Martin E. Hellman. "New Directions in Cryptography". (English). (*Invited Paper*) *IEEE Transactions on Information Theory*, IT-22(6):pp. 644–654, 1976.
- [178] Tim Henderson. "Cryptography and Complexity". (English). (*Department of Electrical Engineering and Computer Science, Case Western Reserve University*), pages 1–17, 1976.
- [179] M. D. Godfrey; D. F. Hendry. "The Computer as von Neumann Planned It.". (English). *IEEE Annals of the History of Computing*, Vol. 15, N° 1:11–21, 1995.
- [180] Erik Barendsen Henk Barendregt. "Introduction to Lambda Calculus". (English) (Revised Edition). December 1998, March 2000.
- [181] Knut Hinkelmann. "Forward Chaining vs. Backward Chaining". (English). (*University of Applied Sciences, Northwestern Switzerland School of Business*).
- [182] Jaakko Hintikka. "*On Gödel*". (English). (*Wadsworth, a division of Thomson Learning*). 2000.
- [183] Wilfrin Hodges. *A Shorter Model Theory*. (English). (*Elsevier 3<sup>rd</sup> Edition*). 1997.
- [184] J.P.E. Hodgson. *The INRIA ISO Prolog Web* (English). (*Saint Joseph's University Philadelphia*). 1999.
- [185] Carsten Heirz; Brook Moses; Jobst Hoffmann. "SWI-Prolog 5.6 Reference Manual" (English). (maintainer: Jobst hoffmann\* <j.hoffmann(at)fh-aachen.de>). 2015 (Version 1.6).
- [186] Andreas Hohmann. "*Programming Languages at Glance*." (English). (<http://www.minimalprogramming.org/html/index.html>). 2003.
- [187] Reijer Hooykaas. *Catastrophism in Geology, its scientific character in relation to actualism and uniformitarianism* (English). *North-Holland Publishing Company. Amsterdam · London*. 1970.

- [188] Alfred Horn. "On sentences which are true of direct unions of algebras" (English). *Association for Symbolic Logic*, 16(Issue 1):14–21, (March) 1951. DOI : 10.2307/2268661.
- [189] Dough Hoyte. *Let Over Lambda, 50 years of Lisp (English)*. Butterworth-Heinemann. An HCSW and Hoytech production. 2008.
- [190] David Hume. *An Enquiry Concerning Human Understanding, and Selection from A Treatise of Human Nature*. (English). (Chicago, The Open Court Publishing Company). 1921.
- [191] David Hume. *Treatise of Human Nature*. (English). Reprint: L.A. Selby-Bigge, M.A. (Oxford: Clarendon Press). pp. 9; 51; 52; 55; 72; 77; 79; 93-95; 131-133. (1738-1740) Reprint: 1896.
- [192] Amzi! inc. "Adventure in Prolog, a Tutorial Introduction." (English). (ePub Books.). 1995-2016.
- [193] Akira Ishikawa. "A Retrospective and Prospects of the Fifth Generation Computer Project". (English). (*Working Paper, The University of Texas at Austin*), 1983.
- [194] Jonathan P. Seldin J. Roger Hindley. "Lambda-Calculus and Combinators in the XX<sup>th</sup> Century". (English). (Eds. D. M. Gabbay and J. Woods, Elsevier Co. (North-Holland), Amsterdam) *Handbook of the History of Logic Chapter 14*, 5:pp. 723–817, 2009.
- [195] Daniel J. Edwards Timothy P. Hart Michael I. Levin John McCarthy, Paul W. Abrahams. *Lisp 1.5 Programmers Manual*. (English). (The M.I.T. Press, Massachusetts Institute of Technology). Fifteenth printing 1985 (1962).
- [196] Neil Deaton Jones. "Computability and Complexity: From a Programming Perspective." (English). (The MIT Press). January 15, 1997.
- [197] Simon L. Peyton Jones. "Implementing Functional Languages: a tutorial." (English). (Department of Computer Science, University of Glasgow). March 23, 2000.

- [198] Jr. J.P. Eckert and J.W. Mauchly. "Automatic High-Speed Computing: a Progress Report on the EDVAC". (English). (*Report of Work under Contract No. W-670-ORD-4926, Supplement No. 4, Moore School Library*) University of Pennsylvania, Philadelphia, September 30, 1945.
- [199] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (English)*. (Prentice Hall Series in Artificial Intelligence, Prentice Hall, Englewood Cliffs, New Jersey). 1999.
- [200] J.Green C. Katz J. McCarthy P. Naur A.J. Perlis H. Rutishauser K. Samelson B. Vauquois J.H. Wegstein A. van Wijngaarden M. Woodger J.W. Backus, F.L. Bauer. "Report on the Algorithmic Language Algol by the ACM Committee on Programming Languages and the GAMM Committee on Programming". (English). (*edited by Peter Naur, Dedicated to the memory of William Turanski*), 1962.
- [201] Erich Grädel Jörg Flum and Thomas Wilke. *"Logic and Automata, History and Perspectives."* (English). (TLG, Texts in Logic and Games, Amsterdam University Press), volume 2. 2008.
- [202] Immanuel Kant. *"What does it mean to orient oneself in thinking?"* (English). (Edited and translated by Allen W. Wood, Yale University, Connecticut, George di Giovanni, McGill University, Montréal). 1786.
- [203] Immanuel Kant. *A Religião nos Limites da Simples Razão*. (Portuguese). (*Textos Filosóficos, Edições 70*). 1992.
- [204] Immanuel Kant. *O Conflito das Faculdades*. (Portuguese). (*Textos Filosóficos, Edições 70*). 1993.
- [205] Immanuel Kant. *Crítica da Razão Prática*. (Portuguese). (*Textos Filosóficos, Edições 70*). 1994.
- [206] Immanuel Kant. *A Paz Perpétua e outros Opúsculos*. (Portuguese). (*Textos Filosóficos, Edições 70*). 1995.

- [207] Immanuel Kant. *Fundamentação da Metafísica dos Costumes. (Portuguese). (Textos Filosóficos, Edições 70).* 1995.
- [208] Immanuel Kant. *Os Progressos da Metafísica. (Portuguese). (Textos Filosóficos, Edições 70).* 1995.
- [209] Immanuel Kant. *Crítica da Razão Pura. (Portuguese). (Fundação Calouste Gulbenkian).* 1997.
- [210] Immanuel Kant. *A Crítica da Faculdade do Juízo. (Portuguese). (Imprensa Nacional-Casa da Moeda, Estudos Gerais, Série Universitária, Clássicos de Filosofia).* 1998.
- [211] Immanuel Kant. *Kant's Critiques. (English). (Wilder Publications).* 2008.
- [212] Kapil Kapoor. *Dimensions of Pāṇini Grammar, The Indian Grammatical System (English). (D. K. Printworld Ltd, New Delhi).* 2005 (2010).
- [213] Jonah Katz and David Pesetzky. "The Identity Thesis for Language and Music". (English). (*Institut Jean Nicod, Paris*), 2011.
- [214] Richard A. O' Keefe. *The Craft of Prolog. (English) The MIT Press, Cambridge, Massachusetts; London, England.* p. 9. 1990.
- [215] Chen Chung Chang; H. Jerome Keisler. *Model Theory. Studies in Logic and the Foundations of Mathematics. (English). (Elsevier 3<sup>rd</sup> Edition).* 1973.
- [216] Juliette Kennedy. "Kurt Gödel" And Supplements.(English). edward n. zalta (ed.). *The Stanford Encyclopedia of Philosophy*, (Spring 2015 Edition). URL: <<https://plato.stanford.edu/archives/win2016/entries/goedel/>>.
- [217] Thomas S. Khun. *"A Estrutura das Revoluções Científicas." (Portuguese). (Guerra & Paz 2009).* 1962, 1970, 1996.
- [218] S.C. Kleene. *Introduction to metamathematics. (English). (North-Holland).* 1951.

- [219] Stephen C. Kleene. "Kurt Gödel, A Biographical Memoir by Stephen C. Kleene". (English). *National Academy of Sciences Washington D.C. (ed.)*, (1987).
- [220] Stephen Cole Kleene. "*Mathematical Logic.*" (English). (Dover Publications, Inc. Mineola, New York). 1967.
- [221] Felix Klein. "A Comparative Review of Recent Researches in Geometry." (Translated by M. W. Haskell). (Programme on entering the Philosophical Faculty and the Senate of the University of Erlangen in 1872)". (English). *Bull. New York Math. Soc.*, Vol. 2:pp. 215–249, (1892-1893).
- [222] Donald Knuth. "*The Art of Computer Programming*" (English). (Addison-Wesley.). 1968.
- [223] Robert A. Kowalski. "Predicate Logic as Programming Language". (North-Holland Publishing Company). (English). *Information Processing*, 1974.
- [224] Robert A. Kowalski. "The Early Years of Logic Programming". (ACM). (English). *Communications of the ACM*, 31(1):38–42, January 1988.
- [225] Tomasz Kowaltowski. "von Neumann: suas contribuições à Computação". (Portuguese). *Estudos Avançados 10(26)*, pages 237–260, 1996.
- [226] Shriram Krishnamurthi. "*Programming Languages: Application and Interpretation.*" (English). (Version Second Edition, Self-published). April 14, 2017 (2003).
- [227] Christopher C. Leary; Lars Kristiansen. "*A Friendly Introduction to Mathematical Logic*" (English). (Milne Library). August 10, 2015.
- [228] Josip Lukin Kristijan Krkač. "Wittgenstein the Morphologist I". (English). (Original Paper, UDC 801/L. Wittgenstein) *Synthesis Philosophica*, 46:pp. 427–438, 2008.
- [229] Raúl Rojas; Cüneyt Göktekin; Gerald Friedland; Mike Krüger. "PlanKalkül: The First High-Level Programming Language". (English). (*Freie Universität Berlin, Institut für Informatik*), 2000.

- [230] P. Laeur. *"A Course of Algol 60 Programming, with special reference to the DASK ALGOL system"* (English). (Regnecentralen, Copenhagen). 1961.
- [231] P. Laeur. *"Technical Report, Formal Definition of Algol 60"* (English). (TR 25.088). 1968.
- [232] Jan Wielemaker; Tom Schrijvers; Markus Triska; Torbjörn Lager. "SWI-Prolog". (English). (arxiv:1011.5332v1 [cs.pl] 24 nov 2010). *Theory and Practice of Logic Programming*, 2010.
- [233] David B. Lamkins. *Successful Lisp: How to Understand and Use Common Lisp*. (English). (Paperback Edition). 2004.
- [234] Gregoy Landini. "Russell to Frege". (English). (McMasrer University Library Press, Philosophy/University of Iowa Iowa City, IA52242, USA) *The Journal of the Bertrand Russell Archives*, (12):pp. 160–185, 1992-93.
- [235] Richard L. Lanigan. *Speaking and Semiology: Maurice Merleau-Ponty's Phenomenological Theory of Existential Communication: Walter de Gruyter* (English). 1991.
- [236] Ambrus Kaposi Leran Cai and Thorsten Altenkirch. "Formalising the Completeness Theorem of Classical Propositional Logic in Agda (Proof Pearl)". (English). (University of Nottingham).
- [237] Fred Lerdahl. "Genesis and Architecture of the GTTM Project". (English). *Music Perception*, 23(3):187–194, 2009. DOI: 10.1525/MP.2009.26.3.187.
- [238] Fred Lerdahl and Ray S. Jackendoff. "A Generative Theory of Tonal Music" (English). (MIT Press), 1985.
- [239] David. K. Lewis. *On the Plurality of Worlds*. (English) Oxford: Blackwell. 1986.
- [240] Peter Naur; F. G. Duncan; C. H. Lindsey. "Algol Bulletin". (English). (*ALGOL Bulletin. Issue 1 (March 1959) through Issue 52 (August 1988)*). Peter Naur, editor. *Issue 1 (March 1959) through Issue 15 (June 1962)*) *CACM Communications of the ACM*, (1, 15, 52), 1959.

- [241] Bernard Linsky. "The Notation in *Principia Mathematica*". (English). *The Stanford Encyclopedia of Philosophy (Fall 2016 Edition)*. Edward N. Zalta (ed.), URL = <<https://plato.stanford.edu/archives/fall2016/entries/pm-notation/>>.
- [242] Lennart Ljung. "Black-box Models from Input-output Measurements". (English). (*Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden*), 2001.
- [243] John Locke. "*An Essay Concerning Human Understanding by John Locke.*" (English). (*The Pennsylvania State University*). (First published 1690) 1999.
- [244] Frederico Lourenço. *Grécia Revisitada, Ensaios sobre Cultura Grega.* (Portuguese) Cotovia. 2004.
- [245] M. S. Lourenço. "Um filósofo da evidência. (Portuguese). (*Universidade de Lisboa*). *Boletim da Sociedade Portuguesa de Matemática* 55.
- [246] James Lu and Jerud J. Mead. "*Prolog, a Tutorial Introduction.*" (English). (*Computer Science Department Bucknell University*). 2012.
- [247] Gilles Dowek; Jean-Jacques Lévy. "*Introduction to the Theory of Programming Languages*" (English). (*Springer UTICS*). 2011.
- [248] W. S. MacCulloch and W. Pitts. "A Logical Calculus of Ideas Immanent in Nervous Activity". (English). *The Stanford Encyclopedia of Philosophy (Winter 2017 Edition)*, Edward N. Zalta (ed.). *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- [249] Saunders MacLane. "Carnap on Logical Syntax". (English). *Bull. Amer. Math. Soc.*, 44(3):171–176, 1938.
- [250] Sanjoy Mahajan. "*The Art of Insight in Science and Engineering*" (English). (*The MIT Press, Cambridge, Massachusetts; London, England*). 2014.
- [251] Michael S. Mahoney. "The History of Computing in the History of Technology". (English). (*Princeton University, Princeton, NJ*) *Annals of the History of Computing* 10, pages 113–125, 1988.



- [252] Norman Malcolm. "Ludwig Wittgenstein, a Memoir". (English). (*Clarendon Press*), 1958 (2001).
- [253] John Malpas. "*PROLOG: A Relational Language and Its Applications*" (English). (*Prentice Hall*). 1987.
- [254] María Manzano. "Alonzo Church: His Life, His Work and Some of His Miracles". (English). pages 3–33, December, 1996.
- [255] Mathieu Marion. "Wittgenstein and Brouwer ". (English). (*History of Logic, Springer Synthese*, 137(1/2):pp. 103–127, 2003.
- [256] João Pavão Martins and Maria dos Remédios Cravo. "*Fundamentos da Programação Utilizando Múltiplos Paradigmas.*" (Portuguese). (*Instituto Superior Técnico Press, Coleção Ensino da Ciência e da Tecnologia*). 2011.
- [257] F. P. Mathur. "A Brief Description and Comparison of Programming Languages FORTRAN, ALGOL, COBOL, PL/I, and LISP 1.5 From a Critical Standpoint". (English). (*Technical Memorandum 33-566, National Aeronautics and Space Administration, Jet Propulsion Laboratory California Institute of Technology, Pasadena, California*), 1972.
- [258] Ernst Mayr. "Recapitulation Reinterpreted: The Somatic Program.". (English). (*Cambridge Mass.: Belknap Press of Harvard University Press*) *The Quarterly Review of Biology*, 69(2):pp. 223–232, 1994.
- [259] John McCarthy. "Programs with common sense". (English). (*Paper presented at the Symposium on the Mechanization of Thought Processes, National Physical Laboratory, Teddington, England*) *Proceedings of the Symposium by H. M. Stationery Oce*, 1958.
- [260] John McCarthy. "Recursive functions of symbolic expressions and their computation by machine". (English). (*Artificial Intelligence Project – RLE and MIT Computation Center Memo 8*), pages 1–19, 1959.

- [261] John McCarthy. "Recursive functions of symbolic expressions and their computation by machine, Part I". (English). (*Massachusetts Institute of Technology, Cambridge*) *Communications of the ACM*, 13(4):pp. 184–195, 1960.
- [262] Donald J. Collins (In memoriam William W. Boone). "A Simple Presentation of a Group with Unsolvable Word Problem". (English). (*Department of Mathematics, Monash University, Clayton, Victoria 3168 Australia*) *Illinois Journal o Mathematics*, 30(2):230–234, 1986.
- [263] Luigi Menabrea. "Notions sur la machine analytique de M. Charles Babbage". (French). (*Mathematisch Centrum Amsterdam*) *Bibliothèque universelle de Genève*, 41(352):76, 1842.
- [264] Bert Mendelson. "*Introduction to Topology.*" (English). (*Dover Publications, Inc. New York*). *Third Edition*. 1962, 1968, 1975, 1990.
- [265] Merleau-Ponty. *Phenomenology of Perception (English)*. (translated by Colin Smith, *Routledge Classics, London and New York*). 1962 (1945).
- [266] Merleau-Ponty. *The Visible and the Invisible (English)*. (edited by Claude Lefort, translated by Alphonso Lingis, *Northwestern University*). 1968.
- [267] Merleau-Ponty. *Fenomenologia da Percepção (Portuguese)*. (tradução Carlos Alberto Ribeiro de Moura, *Martins Fontes, São Paulo 1999*). 1994 (1945).
- [268] Merleau-Ponty. *A Natureza (Portuguese)*. (tradução Álvaro Cabral, *Martins Fontes, São Paulo 2000*). 2000 (1945).
- [269] Merleau-Ponty. *The Visible and the Invisible: The Intertwining—The Chiasm (English)*. (*Maurice Merleau-Ponty: Basic Writings, ed. Thomas Baldwin, Routledge*). 2004.
- [270] Dennis Merritt. "*Building Expert Systems in Prolog.*" (English). (*2000 by Amzi! inc; 1989 by Springer-Verlag*). 1989-2000.
- [271] MiklósRédei. "John von Neumann 1903-1957". (English). *Loránd Eötvös University (Forthcoming in European Mathematical Society Newsletter)*.

- [272] Miklós Rédei. "Unsolved Problems in Mathematics. J. von Neumann's address to the International Congress of Mathematicians". (English). (*Amsterdam 1954*) *The Mathematical Intelligence* 21, pages 7–12, 1999.
- [273] Félix Bou Moliner. "Gödel y la Incompletud de las Matemáticas". (Spanish). <http://www.iii.csic.es/fbou/publications/files/Bo00c.pdf>, pages 1–9.
- [274] Michael Morris. "*El 'Tractatus' de Wittgenstein, Guía de Lectura*" (Spanish). (*Traducción de Rodrigo Neira Castaño; Cátedra, Teorema, 1ª edición*). 2015.
- [275] E. N. Mutch and S. Gill. "Conversion Routines". (English). (*Nat. Phys. Lab*) *Proc. Symp. Automat. Digital Comput.*, pages 74–80, 1953-1954.
- [276] Andreas Krall; Ulrich Neumerkel. "The Vienna Abstract Machine". (English). *Institut für Praktische Informatik. Technische Universität Wien*.
- [277] Jan Wielemaker; Ulrich Neumerkel. "Precise Garbage Collection in Prolog". (English). (*Universiteit van Amsterdam, The Netherlands; Technische Universität Wien, Austria*) *Proceedings of CICLOPS*, 2008.
- [278] Ulrich Neumerkel. "The binary WAM, a simplified Prolog engine". (English). (*Institut für Computersprachen, Technische Universität Wien*), 1993.
- [279] A. Newell and J. C. Shaw. "Programming the logic theory machine". (English). (*Proc. Western Joint Computer Conference*), 1957.
- [280] Ernest Nagel; James R. Newman. "*Gödel's Proof*". (English). (*New York University Press, New York and London, Revised Edition*). 2001.
- [281] Isaac Newton. "*The Principia*". (English). (*Translated by Andrew Motte; Great Mind Series*). 1687 (1995).
- [282] Isaac Newton. *Optiks or a Treatise of the Reflexions, Refractions, Inflexions and Colours of Light*. (English). *Third Book, Query 31*. 1704.
- [283] Friedrich Nietzsche. *The Complete Works of Friedrich Nietzsche* (English). (*The First Complete and Authorised English Translation, Edited by Dr. Oscar Levy, The Library of Victoria University, Toronto*). 1910, 1915, 1924 (2015).

- [284] Ulf Nilsson and Jan Maluszynski. *"Logic, Programming and Prolog"- (English). (Second Edition, John Wiley & Sons Ltd.)*. 1995-2000.
- [285] David Summers; Ruth O'Rourke-Jones. *Music The Definitive Visual History (English). (DK)*. 2013.
- [286] Dov Ospovat. *The Development of Darwin's Theory, Natural History, Natural Theology and Natural Selection, 1838-1859 (English). Preface & Introduction. Cambridge University Press*. 1981.
- [287] Malcom Oster. *Science in Europe, 1500-1800 A Primary Sources Reader; A Secondary Sources Reader. (English)*. Palmgrave in association with the Open University. 2002.
- [288] Christos H. Papadimitriou. *"Computational Complexity" (English). (University of California, San Diego; Addison Wesley Longman)*. 1994.
- [289] Ian Parberry. *"Parallel Complexity Theory". (English). (Whitmore Laboratory, Pennsylvania State University, USA; Pitman Publishing)*. 1987.
- [290] Johan Bos Patrick Blackburn and Kristina Striegnitz. *"Learn Prolog Now!" (English)*. 2006-2012.
- [291] Roger Penrose. *"The Emperor's New Mind." (English).(Oxford University Press)*. 1989.
- [292] Roger Penrose. *"Cycles of Time: An Extraordinary New View of the Universe". (English). (The Bodley Head)*. 2010.
- [293] Roger Penrose. *"Ciclos de Tempo: Uma Visão Nova e Extraordinário do Universo". (Portuguese). (Gradiva)*. 2013 (2010).
- [294] Roger Penrose. *"La Nueva Mente del Emperador" (Spanish). (DEBOLS!LLO Ensayo)*. 2015,-16 (1989).
- [295] Alexandre Pereira. *"C e Algoritmos"- (Portuguese). (2ª Edição, Edições Sílabo)*. 2013-2017.

- [296] L. M. Pereira. "Philosophical Incidence of Logical Programming ". (English). *Handbook of the Logic of Argument and Inference, D. Gabbay et al. (eds.), Studies in Logic and Practical Reasoning series, Volume 1*, pp. 425-448. 2002.
- [297] Charles Petzold. *"Code, The Hidden Language of Computer Hardware and Software."* (English). (Microsoft Press). 2000.
- [298] Frank Pfenning. *Logic Programming.* (English). (Carnegie Mellon University). 2006-2007.
- [299] Michał Piekarski. "Analysis—Phenomenology—Morphology: Some Remarks on Ludwig Wittgenstein's Philosophical Method". (English). (David Publishing) *Philosophy Study*, 5(4):pp. 206–212, 2015.
- [300] Steven Pinker. *The Language Instinct, How the Mind Creates Language.* (English). (Penguin Random House UK). 2007 (1994).
- [301] Fred Piper and Sean Murphy. *"Cryptography, a Very Short Introduction"* (English). (Oxford University Press). 2002.
- [302] G. D. Plotkin. "Call-by-name, call-by-value and the  $\lambda$ -Calculus". (English). (North-Holland Publishing Company) *Theoretical Computer Science*, 1:125–129, 1975.
- [303] Vilnis Detlovs; Karlis Podnieks. *"Introduction to Mathematical Logic"* (English). (University of Latvia). 2017-05-24.
- [304] Karl Popper. *The Logic of Scientific Discovery.* (English). (Routledge Classics, London. 1935 (2005).
- [305] Karl Popper. *"Busca Inacabada, Autobiografia Intelectual"* (Portuguese). (Esfera do Caos, Prefácio de João Carlos Espada). Fevereiro de 2008.
- [306] Karl Popper. *"Conjecturas e Refutações, O desenvolvimento do conhecimento científico"* (Portuguese). (Almedina, Notas e Apresentação de João Carlos Espada). Novembro 2006.

- [307] Emil L. Post. "Recursive Unsolvability of a Problem of Thue". (English). *The Journal of Symbolic Logic*, Vol. 12(No. 1):pp. 1–11, (Mar., 1947).
- [308] Emil L. Post. "Finite Combinatory Processes-Formulation 1". (English). *The Journal of Symbolic Logic*, Vol. 1(No. 3):pp. 103–105, (Sep., 1936).
- [309] Simon J. D. Prince. *Computer Vision, Models, Learning, and Inference*. (English). (Cambridge University Press). 2012.
- [310] Nuno Miguel Proença. "*Wittgenstein, a prova e a actividade matemática: um introdução*". (Portuguese). (*Cadernos de Filosofia das Ciências, CFCUL*). 2008.
- [311] Juliet Floyd; Hilary Putnam. "A Note on Wittgenstein's 'Notorious Paragraph' about the Gödel Theorem". (English). *The Journal of Philosophy*, 97(11):pp. 642–32, 2000.
- [312] Panu Raatikainen. "Gödel's Incompleteness Theorems And Supplements". (English). *Edward N. Zalta (ed.). The Stanford Encyclopedia of Philosophy*, (Spring 2015 Edition). URL: <<https://plato.stanford.edu/archives/spr2015/entries/goedel-incompleteness/>>.
- [313] Aarne Ranta. "*Implementing Programming Languages, an Introduction to Compilers and Interpreters*". (English). (*College Publications*). February 6, 2012.
- [314] G. Revesz. *Lambda-Calculus, Combinators, and Functional Programming* (English). *Cambridge Tracts in Theoretical Computer Science 4*. 1988.
- [315] Marcelo B. Ribeiro and António A. P. Videira. "Boltzmann's Concept of Reality". (English). (*Physics Institute, University of Brazil - UFRJ; Department of Philosophy, State University of Rio de Janeiro - UERJ*), 2007.
- [316] J. L. Robinson. "A Machine-Oriented Logic Based on the Resolution Principle". (English). *J.A.C.M.*, vol. 12:23–41, 1965.

- [317] Adriano Duarte Rodrigues. *"Linguística e Comunicação, a Partitura Invisível, Para a abordagem interactiva da linguagem"* (Portuguese). (*CADERNOS UNIVERSITÁRIOS COLIBRI*). 2001 (1ª edição), 2005 (2ª edição).
- [318] Viktor Rodych. "Mathematical Sense: Wittgenstein's Syntactical Structuralism". (English). (*Fromontos Verlag: Publications of the Austrian Ludwig Wittgenstein Society New Series*, (1-18).
- [319] Raúl Rojas. "A Tutorial Introduction to the  $\lambda$ -Calculus". (English). *FU Berlin, WS-97/98, 97/98*.
- [320] Thomas Haigh; Mark Priestley; Crispin Rope. "Reconsidering the Stored-Program Concept". (English). (*Published by the IEEE Computer Society IEEE Annals of the History of Computing*, January-March 2014).
- [321] Phillippe Rouchy. "Aspects of PROLOG History: Logic Programming and Professional Dynamics". (Blekinge Institute of Technology, Sweden). (English). *TeamEthno-Online Issue 2*, pages 85–100, 2 June 2006.
- [322] Philippe Roussel. "Prolog Manuel de référence et d'utilisation". (French). *Groupe de recherche en Intelligence Artificielle. Marseille, 1975*.
- [323] Philippe Roussel, Alain Colmerauer, Henry Kanoui, and Robert Pasero. "Un système de communication homme-machine en Français, rapport de recherche, Groupe de recherche en Intelligence Artificielle de Marseille". (French). 1973.
- [324] Neil C. Rowe. *"Artificial Intelligence Through Prolog"* (English). (*Prentice-Hall International*). 1998.
- [325] Stuart Russell and Peter Norvig. *"Artificial Intelligence, a Modern Approach."* (English). (*Third Edition, Pearson*). 2010.
- [326] Alex Sakharov. "Horn Clause". (English). (*MathWorld—A Wolfram Web Resource, created by Eric W. Weisstein. URL=<http://mathworld.wolfram.com/HornClause.html>*).

- [327] John E. Savage. *"Models of Computation, Exploring the Power of Computing"*. (English). (Brown University, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA). 1997.
- [328] Charles Sayward. "Steiner versus Wittgenstein: Remarks on Differing Views of Mathematical Truth". (English). *Theoria*, 54:347–352, 2005.
- [329] Randall Scott. *"A Guide to Artificial Intelligence with Visual Prolog"* (English). (Outskirt's Press, Inc.). 2010.
- [330] Roger Scruton. *"Understanding Music, Philosophy and Interpretation"* (Bloomsbury). (English). 2009.
- [331] Peter Selinger. *"Lecture Notes on Lambda Calculus"*. (English). (Department of Mathematics and Statistics Dalhousie University, Halifax, Canada). 2005.
- [332] R. Vijay Shankar. "Shannon's Theory of Cryptography". (English). (*CS702 Seminar; Instructor: Prof. C. Pandu Rangan*), pages 1–13, 1997.
- [333] Claude Shannon. "A Mathematical Theory of Cryptography". (Declassified). (English). (memorandum mm 45-110-02). *Bell Laboratories*, 1945 (declassified version 1949).
- [334] Claude Shannon. "Communication Theory of Secrecy Systems" (English). (*Bell System Technical Journal*), 28(4):pp 656–715, October 1949.
- [335] Claude E. Shannon. "A Symbolic Analysis. (English). *Bulletin of Mathematical Biophysics*, 5:pp. 115–133, 1943.
- [336] Claude E. Shannon. "A Mathematical Theory of Communication". (English). *The Board of Trustees of the University of Illinois.*, 1949.
- [337] Leon Sterling; Ehud Shapiro. *The Art of Prolog. (English) The MIT Press, Cambridge, Massachusetts; London, England.* p. xxix. 1994.
- [338] Stewart Shapiro. *"Filosofia da Matemática"* (Portuguese). (*Tradução e nota de Augusto J. Franco d Oliveira; Edições 70*). 2000 (2015).



- [339] R. F. Shepherd. *"Algol 60 Programming" (English)*. (McGRAW-Hill Publishing Company Limited, Maidenhead · Berkshire · England). 1972.
- [340] Fernando C.N. Pereira; Stuart M. Shieber. *Prolog and Natural-Language Analysis*. (English) Microtone Publishing. 1987, 2012.
- [341] Georgi E. Shilov. *"Linear Algebra" (English)*. (Dover Publications, Inc. Mineola New York). *Second Edition*. 1971, 1977.
- [342] Alex Simpson. "Logic Programming Lecture 7: The Closed World Assumption". (English). (School of Informatics) Springer-Verlag, 2012.
- [343] Rebecca Elizabeth Skinner. *"Building the Second Mind: 1956 and the Origins of Artificial Intelligence Computing."* (English). (UC Berkeley Previously Published Works). 05-01-2012.
- [344] W. B. Vasantha Kandasamy; Florentin Smarandache. *"Set Theoretic Approach to Algebraic Structures in Mathematics, A revelation"*. (English). (Educational Publisher Inc.). 2013.
- [345] Anil Maheshwari; Michiel Smid. *"Introduction to Theory of Computation."* (English). (School of Computer Science, Carleton University, Ottawa, Canada). March 23, 2017.
- [346] Peter Smith. *"An Introduction to Gödel's Theorems" (English)*. (Cambridge University Press, Second Edition). First Published 2007, Second Edition 2013.
- [347] Michael Spivak. *"Calculus" (English)*. (Edited by Publish or Perish, Houston, Texas). 2008.
- [348] J.M. Spivey. *"An introduction to logic programming through Prolog" (English)*. (Prentice-Hall International.). 1996.
- [349] Deirdre Haskell; Anand Pillay; Charles Steinhorn. *"Model Theory, Algebra, and Geometry"*. (English). (Cambridge University Press). June 10, 2010.

- [350] J. V. Tucker; K. Stephenson. *"Data, Syntax and Semantics, An Introduction to Modelling Programming Languages"* (English). (Department of Computer Science University of Wales Swansea, QinetiQ). 2006.
- [351] John Stillwell. "The Word Problem and the Isomorphism Problem for Groups". (English). (Department of Mathematics, Monash University, Clayton, Victoria 3168 Australia) *Bulletin AMS*, 6(1):33–56, 1982.
- [352] George F. Luger; William A. Stubblefield. *"AI Algorithms, Data Structures, and Idioms in Prolog, Lisp, and Java"*. (English). (Pearson Education Inc.). 2009.
- [353] Guy Lewis Steele Jr.; Gerald Jay Sussman. *"The Art of the Interpreter, or the Modularity Complex"*. (English). (Massachusetts Institute of Technology, Artificial Intelligence Laboratory). (AI Memo). Number 453. 1978.
- [354] Coordenação Bernhard Sylla. *"Historical Linguistics Problems and Perspectives"* (Portuguese). (Universidade do Minho, Centro de Estudos Humanísticos). 2017.
- [355] Matthew Szudzik and Eric W. Weisstein. "Continuum Hypothesis", From MathWorld—A Wolfram Web Resource, URL = <http://mathworld.wolfram.com/ContinuumHypothesis.html>. (English).
- [356] Neil Tennant. "Logicism and Neologicism". (English). *The Stanford Encyclopedia of Philosophy (Winter 2017 Edition)*, Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2017/entries/logicism/>.
- [357] D'Arcy Wentworth Thompson. *On Growth and Form*. (English). Cambridge University Press. 1917.
- [358] Richard M. Timoney. "Eigenvalues, diagonalisation and some applications". (English). *LSE Maths (Chapter 3)*, pages 37–51, March 13, 2014.
- [359] Fernando Soler Toscano. *"Modelos Formales de Explicación en Lógica e Inteligencia Artificial"*. (Spanish). (Tesis presentada por Fernando Soler Toscano para optar al grado de Doctor por la Universidad de Sevilla). 2005.

- [360] David S. Touretzky. *"Common Lisp, a Gentle Introduction to Symbolic Computation"*. (English). (Dover Publications, Inc. Mineola, New York). 2013.
- [361] Bart Demoen; Phuong-Lan Nguyen; Tom Schrijvers; Remko Tronçon. *"The First 10 Prolog Programming Contests"* (English).(Bart Demoen). 2005.
- [362] A. M. Turing. [Delivered to the Society November 1936] "On Computable Numbers, with an Application to the Entscheidungsproblem: A correction" (1937, 1938). (English). *Proceedings of the London Mathematical Society*, 2. 42 pp. 230-265 & 2.43 pp. 544-546.
- [363] Alan Turing. *Collected Works of A. M. Turing. Mathematical Logic* (English). (R. O. Gandy & C.E.M. Yates Editors). North-Holland. Amsterdam · London · New York · Tokyo. 1992.
- [364] Alan Turing. *Collected Works of A. M. Turing. Mechanical Intelligence* (English). (D. C. Ince editor). North-Holland. Amsterdam · London · New York · Tokyo. 1992.
- [365] Alan Turing. *Collected Works of A. M. Turing. Morphogenesis* (English). (P. T. Saunders editor) North-Holland. Amsterdam · London · New York · Tokyo. 1992.
- [366] Alan Turing. *Collected Works of A. M. Turing. Pure Mathematics* (English). (J. L. Britton editor). North-Holland. Amsterdam · London · New York · Tokyo. 1992.
- [367] Alan Turing. *Alan Turing's Systems of Logic* (English). (The Princeton Thesis, Edited and introduced by Andrew W. Appel, Princeton University Press, Princeton and Oxford). 2012.
- [368] A.M. Turing. Computing Machinery and Intelligence. (English). *Mind, A Quarterly Review of Psychology and Philosophy*, vol. LIX. No. 236: pp. 433–460, 1950.
- [369] Jos Uffink. "Boltzmann's Work in Statistical Physics". (English). *The Stanford Encyclopedia of Philosophy*; Edward N. Zalta ed., 2007.

- [370] S. Ulam. "John von Neumann 1903-1957". (English). (*The Neumann Compendium edited by Muraskin Murray, Brody F, Vamos Tibor*) Los Alamos Scientific Laboratory, pages 1–49, Received by the editors February 8, 1958.
- [371] Marcel van der Veer. "Algol 68 Genie, an Algol 68 implementation". (English). (*Mathematisch Centrum Amsterdam*), 2001-2015.
- [372] José Braga Vasconcelos. "*C e Alg.*" (Portuguese). (*FCA - Editora de Informática, Lda*). 2015.
- [373] Michael A. Covington; Donald Nute; André Vellino. "*Prolog Programming in Depth*". (English). (*Artificial Intelligence Programs The University of Georgia*). September 1995.
- [374] Cédric Villani. "*H-Theorem and beyond: Boltzmann's entropy in today's mathematics*". (English). (*ENS Lyon (UMR CNRS 5669) & Institut Universitaire de France*).
- [375] Cédric Villani. "Entropy and *H*-Theorem, The Mathematical Legacy of Ludwig Boltzmann". (English). (*ENS Lyon (UMR CNRS 5669) & Institut Universitaire de France*), 2011.
- [376] Eva Volná. "*Introduction to Soft Computing*" (English). (*Bookboon.com*). 2013.
- [377] Herman H. Goldstine; John von Neumann. "Planning and Coding of Problems for an Electronic Computing Instrument". (English). (*Report on the Mathematical and Logical Aspects of an Electronic Computing Instrument, The Institute for Advanced Study Princeton, New Jersey*) *Bull. New York Math. Soc.*, Part II, Vol. 1-3:pp. 1–194, 1947,1948.
- [378] John von Neumann. *Theory of Self-Reproducing Automata*. (English). *By the Board of Trustees of the University of Illinois. Manufactured in the United States of America. Library of Congress Card No. 63-7246*. 1966.

- [379] John von Neumann. *The Computer & the Brain, Foreword by Ray Kurzweil (English). Third Edition. Yale University Press, New Haven & London.* First Edition 1958.
- [380] John von Neumann. "First Draft of a Report on the EDVAC" (English). *Contract No. W-670-ORD-4926.*, Moore School of Electrical Engineering University of Pennsylvania, June 30, 1945.
- [381] Roger Waismann. *Wittgenstein and the Vienna Circle: Conversations Recorded by Friedrich Waismann, transcribed by J. Schulte and B. McGinness, New York: Barnes and Noble. (English).* 1979.
- [382] Krzysztof R. Apt; Mark Wallace. "*Constraint Logic Programming using Eclipse*" (English). (Cambridge University Press; 1 edition). January 15, 2007.
- [383] Francesca Rossi; Peter Van Beek; Toby Walsh. "Constraint Satisfaction: An Emerging Paradigm". (English). (*Foundations of Artificial Intelligence, Amsterdam Elsevier*) *Handbook of Constraint Programming*, 2006.
- [384] Wang. "Information & Entropy". (English). (*Comp 595 DM*).
- [385] Hao Wang. "Time in Philosophy and in Physics: From Kant and Einstein to Gödel". (English). *Synthese*, 2(102):215–234, 1995.
- [386] Willis H. Ware. "*RAND and the Information Evolution A History in Essays and Vignettes*" (English). (*RAND Corporation*), volume 2. 2008.
- [387] D. H. D. Warren. "Higher-order extensions to PROLOG: are they needed?". (English). (*Department of Artificial Intelligence, University of Edinburgh*).
- [388] David S. Warren. "Programming in Tabled Prolog". (Source: CiteSeer). (English). 2:215–234, 1997.

- [389] Fernando C.N. Pereira; David H.D. Warren. "Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks", Department of Artificial Intelligence, University of Edinburgh. North-Holland Publishing Company (English). *Artificial Intelligence*, (13):231–278, 1980.
- [390] Terrance Swift; David S. Warren. "*The XSB System Version 3.3x*". (English). (Volume 1, Programmer's Manual). July 4, 2013.
- [391] Watson and Crick. "Molecular Structure of Nucleic Acids". (English). *Nature*, April 25, 1953.
- [392] Barry Watson. *Computational Logical Notes*. (English). 2014-2016.
- [393] Stephen M. Watt. "Programming with Logic Supplementary Lecture". (English). (University of Waterloo, Canada).
- [394] Alexander Waugh. *Musica Clássica Outra Forma de Ouvir* (Portuguese). (Editorial Estampa). 2000.
- [395] W. P. Weijland. "Semantics for logic programs without occur check". (English). (Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands) *Theoretical Computer Science*, 71(1):pp. 155–174, 1990.
- [396] Steven Weinberg. *To Explain the World: The Discovery of Modern Science*. (English). ISBN-13: 978-0062346667. Reprint Edition 2005.
- [397] William A. R. Weiss. "*Introduction to Set Theory*". (English). (CreateSpace Independent Publishing Platform). November 21, 2014.
- [398] Graham White. "The Philosophy of Computer Languages". (English). *The Blackwell Guide to the Philosophy of Computing and Information*, Edited by Luciano Floridi, Chapter 18: pp. 237–247, 2004.
- [399] Alfred North Whitehead and Bertrand Russell. "*Principia Mathematica*". (English). (Merchant Books), volume I, II, III. 1910, 1912, 1913.

- [400] B. A. Wichmann. *"Algol 60 Compilation and Assessment"*. (English). (Academic Press · London and New York). 1973.
- [401] Jan Wielemaker. *"XPCE/Prolog Notes"* (English). (VU University Amsterdam De Boelelaan The Netherlands, University of Amsterdam Kruislaan The Netherlands).
- [402] Jan Wielemaker. *"SWI-Prolog5.6 ReferenceManual"* (English). (University of Amsterdam, Human-Computer Studies (HCS, formerly SWI), Kruislaan, The Netherlands). 2007.
- [403] Jan Wielemaker. *"SWI Prolog Reference Manual. Uptaded for version 6.2.6"* (English). (VU University Amsterdam, The Netherlands). January, 2013.
- [404] Norbert Wiener. *"Cybernetics or, Control and Communication in the Animal and the Machine"*. (English). (Martino Publishing Mansfield Centre, CT, second edition). (1948, 1951) 2013.
- [405] Norbert Wiener. *"The Human Use of Human Beings, Cybernetics and Society"* (English). (The Da Capo Series in Science, Da Capo Press). 1950-1954.
- [406] Herbert S. Wilf. *"Algorithms and Complexity"*. (English). (University of Pennsylvania). Internet Edition, Summer, 1994.
- [407] Brian J. Gough (with a foreword by Richard M. Stallman). *"An Introduction to GCC - for the GNU compilers gcc and g++"*. (English). (Network Theory). 2004.
- [408] Edited with an Introduction and Notes by Peter Pesic. *Beyond Geometry, Classic Papers from Riemann to Einstein* (English). (Dover books on Mathematics, Dover Publications, Inc. Mineola, New York). 2016.
- [409] C. Gordon Bell; John Grason; Allen Newell (with the assistance of Daniel Siewiorek and Ross Scroggs). *"Designing Computers and Digital Systems"*. (English). (Digital Press, Carnegie-Mellon University ). 4-3-2002.
- [410] Ludwig Wittgenstein. *Notebooks, 1914-1916*. (English). (New York, Harper). 1961.

- [411] Ludwig Wittgenstein. *Grammaire Philosophique. (French). (Titre original: Philosophische Grammatik)*, Blackwell, Oxford; folio essais, Gallimard. (1969) 1980.
- [412] Ludwig Wittgenstein. *Cadernos 1914-1916. (Portuguese). (Edições 70, Biblioteca de Filosofia Contemporânea)*. (1977) 2004.
- [413] Ludwig Wittgenstein. *Anotações sobre as cores. (Portuguese; German). (Edição Bilingue; Edições 70, Biblioteca de Filosofia Contemporânea)*. (1977) 2018.
- [414] Ludwig Wittgenstein. *"Remarks on the Foundations of Mathematics". (English). (Second Edition, MIT Press). (von Wright, Georg Henrik; Rhees, Rush; Anscombe, Gertrude Elizabeth Margaret eds.)*. 1983.
- [415] Ludwig Wittgenstein. *"Tratado Lógico-Filosófico \* Investigações Filosóficas" (Portuguese). (Fundação Calouste Gulbenkian, 2ª Edição)*. 1995.
- [416] Ludwig Wittgenstein. *"Da Certeza" (Über Gewissheit). (Portuguese). (Biblioteca de Filosofia Contemporânea, Edições 70)*. 1998 (1969).
- [417] Ludwig Wittgenstein. *"Últimos Escritos sobre a Filosofia da Psicologia" (Portuguese). (Fundação Calouste Gulbenkian, Lisboa, 2ª Edição)*. 2014.
- [418] Ludwig Wittgenstein. *"Tractatus Logico-Philosophicus". (Translated from the German by C.K. Ogden with an Introduction by Bertrand Russell). (German; English)*. 4.063 .1922.
- [419] Ludwig Wittgenstein. *Tractatus Logico-Philosophicus (Translated from the German by D. F. Pears & B. F. McGuinness with an Introduction by Bertrand Russell)*. (German; English). 4.063 .1963.
- [420] Ben H. Yandell. *The Honors Class; Hilbert's Problems and Their Solvers. (English)*. A K Peters Ltd. Natick, Massachusetts. p. 21. 2001.
- [421] Jean Yves-Girard. *"Proofs and Types". (English). (Translated and with appendices by Paul Taylor and Yves Lafont, Cambridge University Press)*. 1989, 1990, 2003.



- [422] Alexander Zelitchenko. "Is 'Mind-Body Environment' Closed or Open System?". (English). *Academia.edu*.
- [423] Konrad Zuse. "*Translation of: 'Rechnender Raum.'*" (English). (*MIT Technical Translation; Freidr. Vieweg & Sohn, Braunschweig.*), volume 1. 1969.