Wherefore art thou ... Semantics of Computation?

Furio Honsell

Wherefore art thou ... Semantics of Computation?

Furio Honsell

Università di Udine, Italy
sindaco@comune.udine.it

Abstract. Nearly 60 years have passed since the notion of *semantics* was first used to explain *Programming Languages*. There was quite some divergence of opinions, at the time, in what the *semantics of semantics* was supposed to be. Today, in face of the plethora of different models and logical systems based thereupon, are we in a better position to address this *socratic* question? We analyse philosophical issues revolving around the Foundations of Formal Reasoning, Proof Cultures, Logical Frameworks, the Algebraic/Co-algebraic Duality, and Games. We put forward the thesis that, rather than being a drawback, *plurality* is what makes Semantics useful. In that Semantics of Computation is a "*partita doppia*", a double check of what we think we understand in computing.
Keywords. Semantics, Algebraic/Co-algebraic Semantics, Logical Frameworks, Typed Lambda Calculus, Games

1 Introduction

A mathematician, at one point while giving a talk, said:
- And this trivially holds.
But then, somewhat to himself, he added:
- ... but is this really trivial? ...
He kept silent for nearly a minute, and finally cried out triumphantly
- Yes, it is indeed trivial!

The concept of *Semantics of Computation* or *of Programming Languages*, is nearly 60 years old. The notion started to be used in connection to Programming Languages (PL's) by the ALGOL Committee[1] in the late '50's. Semantics *per se* was weaned in 1964 when the IFIP WG 2.2 *Formal Description of Programming Concepts* was established. This occurred in the aftermath of the momentous IFIP TC2 Working Conference *Formal Language Description Languages for Computer Programming* held in Vienna in September 1964. The proceedings of that Conference [25] are an impressive as well as exhilarating catalogue of the embryos of the most influential ideas of Theoretical Computer Science to come, expressed by the founding fathers themselves!

But what a divergent collection of methodologies appears in [25]. It fully accounts for the fact that the word "Semantics" is a *pluralia tantum* and that the very title of [25], besides using the word "languages", in the plural, has itself an ambivalent reading. It can be parsed both as "Formal (Language Description Languages)", *i.e.* formal metalanguages, or, alternatively as "(Formal Language) Description Languages", *i.e.* metalanguages for formal languages. The "true"

[1] J.Backus, in 1959, used it explicitly in the title of the paper [12].

meaning was probably both, *i.e.* "Formal (Formal Language) Description Languages", but for æsthetical reasons the two adjectives "formal", were coalesced. The 1964 Vienna Conference was a gathering of giants. D. Knuth introduced *attribute grammars*. P. Landin, C. Böhm, and C. Stratchey introduced different methods and metalanguages based on λ-*calculus*. Landin presented *A formal description of Algol 60*, using a precursor of ISWIM, Böhm defined the *The CuCh Machine*, combining (Cu)rry and(Ch)urch's formalisms, and Strachey, in *Towards a Formal Semantics*, envisaged what was to become *denotational semantics*. J. McCarthy gave a formal description of ALGOL, M. Nivat discussed its semantics, and C. Elgot analysed RAM's.

The following decades witnessed breathtaking developments, in the directions outlined in Vienna. (One may still wonder if all ideas in [25] have been fully explored!) Milestones in this triumphal march of Semantics are the papers by Scott, *e.g.* [53–55] where he introduced domains, using topological spaces and advocated category theory; the results by Plotkin, *e.g.* [47, 48, 58], where both were developed to an outstanding level; the seminal work by Milner [42, 45] on calculi for communicating and mobile processes; and the later revolutionary and massive contributions of Girard [28, 29] and Abramsky [4, 6], who explored interaction categories and brought into the picture the crucial metaphor of *games*. Since the 70's the growth of Semantics was strengthened also by a logical strain of research pioneered by Martin-Löf [41], based on the *formulæ-as-types analogy*, whereby programs can be extracted from proofs in *Constructive Type Theory*.

Standing on the shoulders of these giants, today we have full-fledged research fields such as Domain Semantics, Categorical Semantics, and Game Semantics, and excellent systems for extracting certified programs from formal proofs [18]. But frankly, today we are even further than in 1964 from a convergent view of what is the *ultimate* semantics or metalanguage, having given up completely the quest for the *universal programming language*.

The purpose of this paper is not in the least to attempt at providing, 60 years later, a unifying viewpoint or even an exhaustive account of Semantics. But rather, drawing from Philosophy, Logics, Mathematics, and Computing[2] I will present various ideas and problematic issues calling for a general reflection on what do we mean by *Semantics*. Semantics is intended to *explain*. Hence any semanticist, not to sustain immediately an attack by an *ad hominem* argument, should show how it can explain *itself*. Many consolatory narratives are available, and it is easy to be carried away by the sheer logico-mathematical beauty of the constructions. For this reason, in the title, I rephrased Shakespeare's ontological, almost tautological, question to express the need to address the dramatically naïve, but as yet unsettled, issue of *What is the Semantics of Computation?*

2 Some intriguing, as well as distressing, issues

2.1 The Pythagorean Dream and the Original Sin

There is no doubt that the power of digital simulation combined with the elementary simplicity of Universal Computational Models is the Pythagorean Dream

[2] Bearing in mind [44], I speak here of "Computing" rather than "Computer Science".

made true. The Universe is apparently *rational* and *comprehensible*. We are capable of digitally simulating even the most challenging phenomena from, say, weather patterns to the style of impressionist painters. All natural processes appear to be *encodable*, once the appropriate data are gödelized, even using just a single symbol, and then *processed computationally* with a simple PL.

We have countless Universal Computational Models, *e.g.* Turing Machines, Church's λ-calculus, Shönfinkel's and Curry's Combinatory Logic (see [15]), Von Neumann self-reproducing automata, and cellular automata such as Conway's Game of Life. There are even more abstract devices such as the Billiard Balls Computers and other reversible computers which require theoretically no minimal energy for computation, trading off speed for energy cost. What we need is a little, but finite, control, and provided unlimited resources are available, we are up and running *recursive functions*, no matter whether our device is sequential or concurrent, reversible or not, deterministic or stochastic, digital or analogic.

At a closer look, there is a critical philosophical dychotomy among these models which is best expressed by the Dreyfuses in [19]. We can either proceed in the tradition of Socrates, Descartes, Leibniz, Kant, Husserl, Wittgenstein and manipulate *context-free,* purpose-free *symbols*, to achieve formally specifiable tasks, or we can try to develop devices capable of performing tasks which can only be dealt with statistically, such as *use* or *pattern recognition*, and are difficult to grasp formally. This latter approach, which can be traced back to Heidegger and the later Wittgenstein, tries to capture "a socially organized nexus of equipment, purpose, and human roles", common-sense performing, and every day know-how,

But there is a more critical issue involved, which often turns the Pythagorean dream into a nightmare. The very generality, remoteness and *gratuitousness* of these universal Computational Models, is also the original sin of Computing. In effect, the dynamics of the elementary token/symbol manipulation going on in these models is over-idiosyncratic and opaque. Despite *Church's Thesis* these systems are not *semantically insightful*. When we look at how Universal Computational Models operate, we realize that the *emerging phenomena* which we would like to understand are accessible only indirectly. This is the root reason for developing abstract formal methods and formal metalangages, *i.e.* for studying the *formal semantics of Computation*.

2.2 The myth of formal reasoning

The need for correct and dependable software has grown enormously in the post-modern world, because of the many life-critical applications of software such as those required in *fly-by-wire*, or the management of large hazardous industrial plants. "Digital woes" are always lurking in the background. The Pentium Bug, the Millennium Bug, and failures in space industry, *e.g.* the Ariane 501 flight [22], highlight the need for certified software.

Over the years a multitude of beautiful *formal specification systems*, *meta-languages* and *programming logics* based on *semantics* have been introduced to achieve this goal, *e.g.* [26, 27]. But many of these are often just as impenetrable and idiosyncratic as what they purport to explain. Just think a bout λ-calculus, CCS, π-calculus, Ambient Calculus, Dynamic and Temporal Logics, ...

A significant step forward in formal reasoning on software was made in the late 70's when Milner first started using computers to assist in reasoning rigorously on computers, *i.e. Computer Assisted Formal Reasoning* (CAFR), see [43]. His seminal system LCF originated a vast research field which later converged with that on machine-checking Mathematics, which had been carried out in the AUTOMATH project since the 60's by N. De Bruijn and his group [1] in the Netherlands. Capitalizing on the *propositions-as-types* and *proofs-as-λ-terms* analogies this line of research culminated in the late 80's and 90's with the construction of a number of *interactive proof development environments* such as NuPRL, Coq, LF Elf, Alf. Most of them are based on Constructive Logic and thus permit to synthesize a certified program meeting a given specification, by extracting it from a formal proof that the specification can be fulfilled[3].

Once CAFR was available for a given logic, the real challenge became not to start from scratch a new implementation for each logical system one needed. An *ad hoc* implementation for a specific formal system does not even appear to be worthwhile since there are often several different presentations of the same system. Moreover, starting from scratch requires a daunting effort to build tools for supporting syntactic operations related to binders and substitution, rule application, and all the necessary procedures associated with proof checking, proof construction, automated search, unification.

The crucial idea for solving this problem was to look for a *Logical Framework*, *i.e.* a *universal proof metalanguage* which can specify the features of a wide range of logics, and implement that system once and for all. The tools of the logical framework can than be tailored to the specific idiosyncracies of each object logic. The first *Logical Framework*, called the Edinburgh Logical Framework, LF was introduced by Harper, Honsell, Plotkin in 1987, [32]. In 2007 it received the *Test-of-Time Award* at the ACM/IEEE LICS Symposium.

LF is a dependent simply typed λ-calculus together with a paradigm for reducing all the notions and features of a generic logical system[4] to λ-terms of a suitable, possibly higher-order, type. Then all meta-logical operations, *e.g.* instantiation or application of a rule, are reduced to β-application, all constraints are enforced by static scoping, and all judgements are expressed as types. Thence, providing the *evidence for a judgment/assertion*, the *fulfillment of a task/specification*, the *satisfaction of an expectation*, the *proof of a theorem* amounts to showing that *a suitable type is inhabited by a closed λ-term*. This methodology, whereby a logic is encoded as a signature in LF, is called the *Judgements-as-Types paradigm*. An important point to make here is expressed by the slogan "LF *is normative*". Namely, the LF encoding can be taken also

[3] In Constructive Logic a formal proof of $\forall x.\exists y.P$ contains an algorithm for computing y given x. *E.g.* the proof that primes are infinitely many contains a method for finding a prime larger than any one in a given list. B. Constable, the founder of Nuprl, mentioned this intriguing example: any finite list of integers has a subsequence of consecutive elements with a largest sum. A natural inductive proof of this obvious fact contains a *linear* algorithm for computing the largest sum, but if we do not follow the program-extraction approach we easily slip into a *quadratic* algorithm.

[4] *Viz.* terms, variables, binders, schemata, rules, assumptions, proofs, theorems, etc.

as the definition, actually one of the very few uniform definitions, of *what* is an axiom/rule, *what* is a theorem, *what* is a proof, *what* is a Logic!

The most successful enterprise in computer-assisted proof development is the system Coq [18] developed at INRIA since the late 80's. Coq originated as an implementation of higher order constructive logic. Since then, it has been continuously enhanced and today important and complex mathematical proofs such as the Four-Color Theorem, the Feit Thompson Theorem in Group Theory, Kepler's sphere-packing problem have been formalized and proof checked in Coq, as well as many areas of Mathematics and Computer Science. For this reason Coq has received the ACM SIGPLAN Software 2013 Award.

But is this the final word as far as software certification? We conclude this section with some provocative issues. It must be said, after all, that even if formal program correctness appears not to be used that much in practice, there have *not* been that many software failures, even if Murphy's Law should be in full action given the extremely large number of times that a given software component is utilized. Why then proving formally program correctness, or formal *proof checking* which is decidable albeit *super exponential*? Is it really necessary to go through a formal proof which is so brittle and usually extremely cumbersome? What are formal proofs useful for? A modest answer is that once proofs become *first class citizens*, they can be carried along with the code and hence can act as verifiable certifications, with applications to security. But the utilitarian might still ask: "what does a proof that a program meets its specifications buy you?" Some generous arguments usually put forward are the following. To have a proof is to have a sort of *invariant*, more sophisticated, but similar to types in programming languages, the dimensions check in Physics, the divisibility by 9 test in decimal Arithmetic. A type correctness result expresses precisely that an *invariant* is preserved under program evaluation. Proofs increase our *trust* in the program or statement. Ancient Greeks made very few mistakes indeed, even if some of the proofs in Euclid's $\Sigma\tau o\iota\chi\epsilon\iota\alpha$ have minor errors.

A final provocative question: are all *interesting* mathematical statements *easy*[5] to prove? Are long proofs of universal statements practically useless? In that all instances that one can encounter in practice are true *independently* of the long universal proof?

2.3 Demystifying the Myth of Formal Reasoning

The crucial issue is whether formal proofs convey an intuitive *understanding* of a theorem or an algorithm? Proofs provide *evidence* but not *explanations*, Shopenhauer ([56],15) criticism of Euclid's proof of Pythagoras Theorem goes in this direction. In practice the hypothetico/deductive method has been and still is replaced by *heuristics*, by *conjectures* subject to counterexamples, and even empirical methods. But do proofs convey any information of the heuristics used to conjecture them? Are formal proofs, then, for the human user or for the

[5] Consider for instance *e.g.* Karatsuba's multiplication. Already Babbage knew that: if $x = x_1B^m + x_0$, $y = y_1B^m + y_0$ then $xy = x_1y_1B^{2m} + (x_1y_0 + x_0y_1)B^m + x_0y_0$ but he failed to notice the optimization behind $x_1y_0 + x_0y_1 = (x_1 + x_0)(y_1 + y_0) - x_1y_1 - x_0y_0$.

machine? What difference does it make if we do not understand the proof to *rely on a machine* rather than another form of authority? What difference is there between: to know *that something is true vs* to know *why it is true*?

Following this line of thought a truly universal proof metalanguage should try to accomodate many different "proof cultures" such as *proofs-without-word*[6] which rely on visual intuitions, proofs using *Diagrams* and *Constructions*, which are crucial in Greek Mathematics when numbers are taken as measurable segments, *proofs by analogy* especially physical[7]. Apodyptic fundamentalism should be avoided. The only arguments which are forbidden are those *by authority*. But is it really so? What about *axioms* and *rules*?

Since Euclid first crystallized the concept of *rigorous proof* using the axiomatic/deductive method, philosophers have been questioning the nature of mathematics: is it *analytic* or *synthetic*? And have often opposed *Algebra* **vs.** *Geometry*, or *computation according to rules* **vs.** *deduction from axioms*, or *verification***vs.** *proof checking*? What is the difference between proving a computation correct and demonstrating its verification by directly executing it? Shopenhauer ([56],15) contrasted obscure geometrical proofs to the sharp direct computation of $\frac{(7+9)*8\ -\ 2}{3} = 42$ [56]. A similar issue was taken up also by Poincaré in his discussion of $2 + 2 = 4$ ([49],p.20). Of course there are proofs of *non-existence* and *uniqueness*[8] which require *arguments* and not just *computations*. But once we have learnt that a rewriting rule is correct why are we not allowed to use it freely! The old alternative between proofs, in Euclid's tradition, **vs** demonstra-

[6] Look at Matita Proof Assistant's logo.

[7] Here is an intriguing list of problems which utilize arguments by analogy:
- Archimedes in his Organon [2] rather than giving a geometrical account of physical phenomena went the opposite way. He utilized physical analogies for addressing mathematical problems and anticipated calculus. His method was to conceive a geometrical figure as composed of thin slices, to which he assigned physical features and which he imagined to hang on a balance scale subject to gravity.
- Given any *point inside a convex polyhedron*, there exists a face of the polyhedron such that the projection of the point onto the plane of that face lies inside the face, for otherwise we would have a *pepetuum mobile*.
- Leibniz suggested to find the point of an acute triangle which has smallest sum of the distances to the vertices of the triangle by hanging equal weights to 3 threads passing through the vertexes, tied all three together in a single knot. The equlibrium point is reached when the potential energy is minimal.
- The towns A and B are separated by a straight river. In what place should we construct a bridge MN, orthogonal to the shores, in order to minimize the length of the road AMNB? Minimal paths are immediate once we think of them as light beams. The bridge corresponds to the refraction caused by a very dense medium.
- Induction *per se* does not appear in Euclid. Even the proof of the so-called Euclid's Algorithm is carried out only up to 3. We conjecture that this is due to the fact that the Greek language, has *dual*, as well as plural endings. Three is already a multitude!

[8] This is nicely expressed by the following puzzle: what is the difference between solving a Sudoku using pencil and eraser by backtracking, rather than using a pen and writing a number only if a stringent argument forces it? The latter solution proves also uniqueness.

tions, in many non-western traditions, can be phrased using the terminology of LF as follows: "proving *inhabitability* of judgements" **vs** "postulating *definitional equality* of types", *i.e. internal* **vs** *external* evidence? A proposal for accomodating different proof cultures in a single LF-like framework following this suggestion appears in [35, 37].

There is a more radical problem, however. *Absolute certainty* is an unattainable goal. In the first place the very specification, or encoding, in the meta-language, might be wrong or imprecise. This is the *adequacy problem* of every formalization. In fact the *formalization process* itself, cannot be *formalized*, and a potential *infinte regress* is triggered here. Moreover the proof engine of the Logical Framework might itself be flawed. The need for a *simple proof engine* in Logical Frameworks to reduce this risk is called *De Brujin Principle*. But we cannot escape the *bootstraping* problem, called *Münchausen trilemma* by Hans Albert. Either we have to face an infinite regress, or a circularity, or we have to rely on some form of reductionism via axiomatics *e.g.* FOL, Naïve Set Theory. And this takes us to the debate of Foundationalism **vs** Anti-foundationalism.

The point of the anecdote *What the tortoise (taught-us) said to Achilles (Ah-kill-ease)* by Lewis Carroll [15], is that at some stage we have to stop making assumptions and "Just do it", turning knowledge into action. To avoid an infinte regress, we have to give up the need for a justification or a proof that the task can be fulfilled, besides the very fact that we can perform it. See [30] for a different more logical analysis of the story.

But what does it mean that we understand a Logic? The skeptical epistemological tradition taken up by Wittgenstein in his argument *against private language*, profoundly discussed in Kripke [40], shows clearly that there is a limit. There is no way by which we can be absolutely certain that we share the same understanding of how to apply a given *rule*. A rule can be applied only *within a proof culture*. From an epistemological viewpoint IQ tests are nonsense, unless what we want to test is whether the examinee is sharing the same prejudice of the examiner. IQ tests cannot show how bright we are, but just how *conformist* we are. This is essentially the old problem of *justifying induction*. As Hume and Wittgenstein showed, we have to resort to *sociology* to explain how rules are understood, or applied. Duhem-Quine's argument, that *there is no testing in isolation*, or Popper's remark that *observations are heavily theory-laden* go precisely in this direction. To exemplify concretely what I mean, consider the following classics in the puzzle genre: which rule gives the following ordering: 4 8 12 2 1 7 6 3 5 11 10 9?[9] What is the next number in the series: 1 1 1 3 1 4 1 0 2 1 3?[10] No finite sequence of data can entail any potentially infinite rule. If the answers given in the foonotes make you say "Aha!", it means that you are belonging to a community which can make sense of that very answer!

A similar issue is: *What counts as an Explanation*? For instance, how do we explain humour? Why do we smile for the joke in the epigraph to this paper,

[9] The alphabetical order of the name of the months.

[10] The series can be read as 1, hence 1 one ; hence 3 one's; hence 4 ones's, 0 two's, and 1 three; so the next counting gives 6 one's.

or listening to the following dialogue between two students: "- Once I thought that correlation implied causation. Then I attended a course in probability. - said the first student. - And did it help you? - asked his friend. - May be! - was the answer he got." Humour arises from a perceived incongruity w.r.t. a rule, a pattern, a structure, a convention. But there is no such thing as a stringent justification for our understanding of a joke. To understand a joke, we have to share something, as is the case of applying a logical rule. Similarly we can enjoy playing games such as Dixit© or the so called *lateral thinking* or *odd-one-out*[11] puzzles, although, strictly speaking, any solution put forward can be challenged.

Explanations are irreducible to complete formalization. They are narratives one can make sense of, within a given culture. In order to understand rules, puzzles, jokes, and explanations we need to belong to a culture. I think that the italian philosopher Antonio Gramsci was among the first to make this point explicit 323-43 (Q1112) [31]: "In acquiring ones conception of the world one always belongs to a particular grouping which is that of all the social elements which share the same mode of thinking and acting. We are all *conformists*[12] of some conformism or other, always *man-in-the-mass* or *collective* man. The question is this: of what historical type is the conformism, the mass humanity to which one belongs?" In recent years Amartya Sen [57] has argued that this *anthropological view* of Gramsci significantly influenced Wittgenstein, through the economist Sraffa, who was well acquainted with Gramsci, and later moved to Cambridge where he met Wittgenstein. Thus indirectly, Gramsci contributed to shift Wittgenstein's viewpoint from the one in the Tractatus to that in the Philosophische Untersuchungen.

Some degree of *conformism*, of *conventionality, gratuitousness*, is necessary to make understanding possible, in any field. But this makes formal reasoning ultimately a little less stringent than naïvely assumed.

3 Semantics as a "*partita doppia*"

Whatever the limitations of Formal Methods, we need *Semantics* to justify logical tools. So we have finally come to the point where we address the problem of *What is a Semantics of a programming language?*

I think that we can all agree on the following:

1. The *Syntax* of a PL is an algebra of *terms* generated by *constructors*.

[11] My favourite example is the following which wittily mocks racism: Who among the following five men is the alien? The first has blue eyes, white skin, medium height, and average build. The second has dark skin and eyes, medium height, and average build. The third has white skin, dark eyes, medium height, and average build. The fourth has short legs, white skin, dark eyes, and average build. Finally the fifth is overwheight, has dark eyes, white skin and medium height. The third man is the alien since he shares 3 characteristics out of 4 with any of the others, while each of the others shares only 2.

[12] Gramsci said: "Conformism means nothing else but sociality, but I prefer to use "conformism" to irritate imbeciles!".

2. Terms have a *behaviour* on which various *Observations* can be made.
3. *Semantics* is an *equivalence relation*, *i.e.* a *partition* on Syntax. Terms are *semantically equivalent* if their behaviours yield the *same* observations.
4. Semantical equivalence, can be induced by an *interpretation function* into a domain of *meanings*. Thus Semantics is a *translation* which provides an *invariant* w.r.t the behaviour. The domain of *meanings* is called a *model*.
5. Semantics is *compositional*, or *denotational*, or *extensional*. when it is a *congruence* relation on the syntax constructors. Constructors themselves can then be assigned functional meanings.
6. The equivalence classes under meaning-equivalence can be structured so as to induce a *refinement* calculus.
7. If the behaviour of terms derives from their belonging to a *rewriting system*, λ-calculus say, then the semantics provides an *invariant* under *evaluation*, *i.e. interpretation* as *evaluation*.

Semantics is a plural concept. The more Semantics the better. The most profound semantical theorems amount to showing that *conceptually independent* procedures yield the *same equivalence*. A very satisfactory situation arises when one succeeds in relating, for the same PL, two *dual* semantics. This occurs when one is: *bottom-up, algebraic, observational, denotational, initial, reductionist*, while the other is *top-down, co-algebraic, intentional, operational, behavioral, final, holistic*. Semantics in the former family are usually compositional, while those in the latter are easier to define.

Standard examples are the following. In Propositional Calculus *truth values* are assigned bottom-up, but give the same semantics as a proof system, *e.g. Tableaux*, which is top-down. *Regular expressions* define languages from phrase-structure grammars bottom-up, but correspondingly we have *finite state automata* providing top-down recognizing procedures for the same languages. *Least fixed point* semantics for recursion, which is denotational, is equivalent to *rewrite* semantics for recursion, which is operational. *Behavioural (bisimulation)* co-algebraic equivalences are insightful when proved to be equivalent to *observational* congruences. We look for *Reduction Systems*, closed under *congruence*, for evaluating terms/programs, in order to grasp the operational essence of *transition systems* expressing the behaviour of *communicating processes*.

But even when the distance between behavior and denotations is small, nevertheless Semantics is useful enough to ground formal methods on it. Why? Semantics provides a kind of *partita doppia*, a *simmetry* or rather a *duality*, which enforces some kind of *invariant*, or *check*, entailing a *safety*, or even a *liveness*, property. I use the terminology *partita doppia*, the Renaissance term for *double-entry bookkeping*, because it was introduced precisely as a safety procedure for checking, in two different ways, the erratic behaviour of financial transactions, *i.e.* economic processes.

But how does Semantics induce Formal Methods? Models usually come with some *algebraic/order/metric/topological/categorical* structure of their own. Hence the interpretation function can enforce some *preservation laws* or *proof principles* deriving from that structure. Standard examples in this sense are: *indexed*

reduction for λ-calculus, *i.e.*$(\lambda x.M)^{n+1}N \ \rightarrow_\beta \ M^n[x/N^n]$ for $\bigcup_{n\in\mathbf{N}}[M^n] = [M]$, Least Fixed Point Induction, and Co-induction.

3.1 Initial and Final Semantics

The two alternate families of semantics mentioned previously can be given a very clean categorical account following [10, 39]. The *interpretation* function

$$\mathcal{I} : \textbf{Language} \rightarrow \textbf{Model}$$

can be seen in two different ways: as a morphism in category of *algebras*, thus giving rise to what is called *Initial Semantics*, or in category of *coalgebrs*, thus giving rise to *Final Semantics*. In Initial Semantics, *Languages* and *Models* are viewed as *F-Algebras*, while in Final Semantics they are viewed as *F-Coalgebras*, for suitable functors F. We recall that *F-Algebras* are pairs (A, F) such that $f : F(A) \rightarrow A$, while *F-coalgebras* are pairs such that $f : A \rightarrow F(A)$. In the algebraic case, terms in the Language are construed as an *initial* F-algebra (*i.e.* it can be mapped into all F-algebras), whereby $\mathcal{I}$ is the *initial* mapping. In the coalgebraic case, the behaviours in the model are construed as a *final coalgebra* (*i.e.* all F-coalgebras can be mapped into it), whereby $\mathcal{I}$ is the *final* mapping. The standard example of an initial algebra is the free-algebra of the Syntax of a language given a signature, *e.g.* Natural Numbers $1 + N = N$. While *Streams*[13] $f : Stream \rightarrow Nat \times Stream$, and the behaviours of *processes* $Proc = A \times Proc$ are the standard examples of final colagebras. Initial Semantics is *syntax-oriented* and induces a *congruence relation*, which can be seen as a *least fixed point*, thereby supporting an *induction principle*. Final Semantics is *behaviour oriented*, it induces a *bisimilarity* equivalence which can be viewed as a *greatest fixed point* of suitable *bisimulations*, thereby supporting a *coinduction principle*. The functor F in intial semantics arises from a *reduction system*, while in final semantics it arises from a *transition system*. Initial Semantics are therefore compositional by definition, but it is difficult to find *fully abstract* models, *i.e.* models where the semantical equivalence is as coarse as intended, or *fully complete* models, *i.e.* models all whose points are definable in the language. Final Semantics are not immediately compositional, but yield more easily *fully abstract* and *fully complete* models, albeit these are often uninformative *term models* (apart from the equivalence). There is a precise *duality* between Initial and Final Semantics concepts, see [51]. A promising unifying framework is *Bialgebraic Semantics*, [59]. It yields natural conditions, on the intial algebra defining syntax and on the final coalgebra of behaviours, so that the bisimialrity given by the latter is a congruence w.r.t. the former. Furthermore rules defining *structural operational semantics* can then be understood as *natural transformations*. This approach is not completely satisfactory yet, in that, higher order objects and implicit or recursive definitions complicate considerably the picture.

The interest in co-algebras has grown, in the last decades, due to the fact that *infinite and circular datatypes* and *non-terminatig processes* have become more

[13] The mathematical notion of *sieve*, is immediately programmed on streams.

prominent, if not preminent, in Computer Science. Initial algebras are usually *well-founded, terminating structures* while final coalgebras are *non-well-founded structures*. "Which recursive function does the *Internet* or an *operating system* compute?" are natural questions from the traditional standpoint, but appear clearly ludicrous. Both processes are not algorithms in the standard sense, on the contrary, they are useless unless they do not terminate. Rather than in single input/output objects we are more interested in *streams of interactions*.

But co-algebras have a role to play also in the fields of logic, philosophy, ethics, politics, and art. The *post-modern* cultural milieu is characterized by the emergence of a range of reflexive discourses, circular phenomena and for the constant interplay between theoretic and metatheoretic levels. Co-algebras allow to construe such circular and self-referential situations as *virtuous circles* rather than *vicious circles*. In this respect it is intersting to notice that although gödelization, digitization, and stored-programs computers have considerably contributed to this aspect of post-modern culture, still since the 80's, circular data were not seriously taken as first class citizens in Mathematics itself.

Circularities arise in various disciplines but also in everyday life[14]. An exciting account of how to use coalgebras to model virtuous circularities appears in [14].

An intriguing class of vicious circles is highlighted by *ad hominem* arguments, see [23]. These can be levelled against theories which become inconsistent because in order to be stated need to assume a presupposition which they make a point

[14] - most English *grammars* are written in English;
- ... from the catalogue of a recent exhibition on *analytic painting*: "The result are paintings which are remarkable for the self-referentiality of their language" - ... from the catalogue of a cinema festival "... every citation of a movie is a reflection on cinema itself, targeted to educate the audiences capable of deciphering the *metatheory of cinema*";
- the *mise en abîme* in paintings and stories, the most remarkable example being the *Mousetrap* in Shakespeare's *Hamlet*;
- there is no History without *Historiography*;
- Mythology can be defined as the way a culture narrates itself;
- self-awareness has been recommended since antiquity: *Know thyself*; and it has been taken as the cornerstone of phylosophical systems, *e.g. Cogito ergo sum*;
- the *third man paradox* in Plato's Parmenides undermines the theory of ideas;
- the formal accounts of such notions as *conventions*, *common knowledge*, *intentionality*, *fashion*, and *statistics* usually involve self-reference in an essential way;
- many *epistemic logic* paradoxes arise from self referentiality;
- an ostensible definition of recursion: "recursion": *viz* recursion;
- the power of *equations*, and *implicit definitions*[15] and *fixed point theorems*;
- a *mind* emerging from a *brain* which is an invention of the mind;
- Akbar the Moghul emperor, who championed tolerance, secularism, and reason, already at the end of the XVI$^{\text{th}}$ century, made the point that *even to dispute reason one has to give a reason for that disputation*;
- the circular assumption (X): *if A, B, C, and X are true then Z is true* could stop the infinite regress in Carroll's anecdote in [15];
- *impredicativity*, say of reals, can be taken as a strong point of the theory;
- the set of concepts I mentioned in this list.

to negate. Nothing is absolutely true for a *skeptic*, but then how can he be absolutely in favour of skepticism? Descartes introduced the *methodic doubt*, namely *he could not doubt that he was doubting*, and thus he found something that was beyond any doubt. Why does a *solipsist* bother to rebut who contradicts his theories? An *ultraformalist*, in claiming that only symbols are meaningful, assumes that a semantics exists, which is precisely what he claims not to exist. Type Theories ban any kind of self-reference, hence cannot assign a *type* to the notion of *type*, thus cannot be general theories for Semantics.

Non-wellfounded Sets and Automata The final example of circular object presented above, namely "The set of all things in the list above", clearly a non-wellfounded set, is the seminal example of co-algebras. The *Axiom of Foundation*, which goes back to the early formulations of Set Theory by Von Neumann and Zermelo in the 20's, whereby all sets are *well-founded*, was not really challenged until the early 70's, although *non-wellfounded sets* had been used earlier in *permutation models* and appropriate axioms had been introduced. But the first Anti-Foundation Axiom which asserts that the Universe is a *final* $\mathcal{P}(\ )$*-co-algebra* is *Axiom* X_1 introduced by M.Forti, F.Honsell in 1983, [24]:
Axiom 1 (X_1) *Given* $f : A \to \mathcal{P}(A)$ *there exists a unique function* $g : A \to B$ *such that* $g(x) = \{g(y) | y \in f(x)\}$.

A Universe satisfying X_1 is not only full with what Mirimanoff in 1917 called teratologies, but it is also *strongly extensional*, namely it is a final $\mathcal{P} - coalgebra$. This was expressed in [24] by saying that the equivalence induced on the structure $f : A \to \mathcal{P}(A)$, namely the $\mathcal{P}$-coalgebra $(A, id_{|A})$, by g is the *maximal fixed point* of the operator $(\ )^+ : Equiv_A \to Equiv_A$ defined by $(R)^+ = \{(x, y) \mid (\forall t \in x.\exists s \in y.tRs) \& (\forall t \in y.\exists s \in x.tRs)\}$. Readers might recognize what later was to be called *bisimulation*. A non-wellfounded set is, in effect, a very elementary non-deterministic automaton when *membership* is viewed as *transition*. This idea of characterizing semantical equivalence as a maximal fixed point is at the core of the *Semantics of Concurrency* and the many *observational equivalences* and corresponding *bisimulations*, introduced since the 80's by Milner and the large number of researchers which developed this vast body of formal methods. Final Semantics arose precisely in this context. The origin of *bisimulations* is remarkable since it occurred independently and almost simultaneously around 1980 in 3 fields at least, by Forti & Honsell in Set Theory, by D. Park in the Semantics of Concurrency, and by J. Van Benthem in the Semantics of Modal Logic.

Among the recent investigations on the foundational role of co-algebraic notions I point out the original paper by Yves André *Qu'est-ce que coagir?: pour une philosophie de la coaction*[16] [11], presented at a Seminar held in Paris in 2014 on the mathematical work of Alain Badiou, the outstanding philosopher, playwright, and militant intellectual. André illustrates the insights provided by *dualities* in various areas of Mathematics and discusses *co-actions* philosophically[17]. He makes the intriguing suggestion of how would Mathematics develop

[16] I thank U. Zannier from the Scuola Normale Superiore in Pisa for pointing it out.

[17] This is a very inspiring passage: "En reprenant la métaphore théâtrale, on pourrait rapprocher le contexte des processus où opèrent les notions algébriques du théâtre

if in high school we would learn to represent a function $f : X \to Y$, both with the traditional representation as a *graph*, but also by means of its *co-graph*. The co-graph is a *partition* on the disjoint sum $X \uplus Y$, or equivalently the equivalence relation induced by the pairs $(x, f(x))$. In this respect André suggests to think about the *partita doppia* of Luca Pacioli, that is to the *stream* of the double entries of credits and debits. A double entry is a means to make a relation *symmetric* thus enforcing an *invariance check*. Axiom X_1, above, gives "*une théorie co-active de la circularité vertueuse*" according to André.

The Foundation Axiom plays a role also in the view maintained by Alain Badiou himself, whereby *Ontology is Zermelo-Frænkel-Gödel-Cohen Set Theory*. In his book *L'être et l'événement*, [13], Badiou asserts that the *event* escapes *Ontology* precisely because ontological concepts are *well-founded sets*, *i.e.* sets founded on $\emptyset$, what he calls the "*pure doctrine of the multiple*". This is also the key point to undestand why he purports that "*the empty is the proper name of being*" and hence that the "*one is not*". He claims that Foundation is a "*metaontological thesis of Ontology*". On the other hand *events* belong to themselves, in that e_x, the *matheme* of the event, *i.e.* its mathematical counterpart, is such that $e_x = \{x \mid x \in e_x\}$. Badiou claims that in grasping an *event* we implicitly take into account the event *itself* through its name and also our very reference to that event, in a potentially infinite regress. The co-algebraic understanding of non-wellfounded sets is a possible counterpart to his outstanding 11^{th} Meditation in [13] on the poem of Mallarmé *Un coup de dès*: "... ou se fût l'événement accompli en vue de tout résultat nul ...".

3.2 Denotational Semantics

Denotationl Semantics, the approach to Semantics orinally envisaged by C. Strachey in [25], is today probably the most successful and natural Semantics for PL's. Denotational Semantics uses *algebras*, thus it is an *initial semantics*. Each syntactic component receives an extensional, functional, denotation in a suitable *structure-enriched* domain. It is compositional by definition. It fully exploits higher-order objects which can be proved to exist, even when defined implicitly, since domains support robust fixed point theorems. At the metalevel, domains can be viewed as objects of *structure-enriched* category. Within such category domains themselves can be defined implicitly, and *domain equations* can be solved by *Inverse Limit* (*co-algebraic*) constructions, because functors over domains have both initial algebras and final co-algebras. Many different categories of domains have been discussed in the literature. The original structures were Scott-Continuous Lattices, a particular category of T_0-topological spaces, [53]. Toady, there are many more order-enriched categories based on the *information ordering*, *e.g.* metric-enriched categories, and we have a very deep categorical understanding of the abstract and universal properties that categories need to

classique, où un petit nombre de protagonistes mène l'action dans un champ spatio-temporel circonscrit. Le contexte où opèrent les notions co-algébriques serait analogue, lui à ces opéras où le protagoniste est un peuple, et où de scéne en scéne, les changements d'états sont marqué s d'intégrales de destins."

satisfy to be used satisfactorily as domains, [50]. Each different category provides its own proof principles for establishing program equivalence. Such equivalences support *refinement calculi* and *inferential programming*, [52].

Relying on proper Denotational Semantics, we can naturally present the semantics of a PL by *Reified* Denotational Semantics. This amounts to the specification of an implementation, by means of Plotkin's *structured operational semantics*, with the confidence that higher-order and implicitly defined objects such as *closures* (*i.e.* $\rho \vdash \lambda.xM \Rightarrow < \lambda.xM, \rho >$), can be naturally tamed in a well-behaved category of domains. This approach is useful pædagogically and allows for elementary tools, based on bisimulations, for reasoning on PL's.

60 years on, since Denotational Semantics first appeared, denotations as *points* of a domain have lost importance. The objects we are concerned today are *morphisms* in categories. We speak therefore of domains as CPO's and Scott-continuous functions, Metric spaces and Continuous functions, Games and Strategies. The latter being the only one which assigns, as we shall see, a new role to elementary points, namely as *moves*. Today we have excellent categorical conceptual compositional understanding of all features of PL's using the categorical notion of *monad*, [46].

But a natural question arises:"Does Category Theory provide a basic insight as Naïve Set Theory does? The answer is probably yes, but Category Theory has suffered from too intense militancy.

Types as finitary approximations in domains and Stone Duality. Scott-Strachey's idea in Denotational Semantics, whereby computation *per se* is understood using the *information ordering* from Recursion Theory provides outstanding results. Computation is reduced to *iteration of suitable operators in suitable domains* using fixed points. This is what makes ordered models work. But how do we make sense of *program logics*, that is the logical invariants of computation? A very successful line of research views *properties* or *types* as *Scott-open sets*. *Intersection Types*, originally introduced by Dezani and Coppo, and later developed by Ronchi, Honsell, Cardone, Abramsky *et al.*, to capture various properties of programs go precisely in this direction.

Intersection Types are essentially a suitable basis for the open sets of the domain. Observation/properties are represented by the compact points of the domain which define the open sets of all the objects which satisfy that property. The denotation of a term is the supremum of the *observations* that one can perform on it. Compact points are computable but not necessarily expressible in the language and this makes domain models not fully abstract. Abramsky [3] provided a categorical understanding of all this by viewing the emergence of program logics as a generalization of *Stone Duality*[18]. Yet again a *duality*, a virtuous circularity, a "*partita doppia*", lies at the core of Semantics.

A long standing open problem We claimed that Denotational Semantics is the most successful semantics because it abstracts the functional behaviour of

[18] In Stone spaces, *i.e.* compact totally disconnected Hausdorff spaces, a point is the filter of the clopen sets, which are a Boolean algebra, to which it belongs.

PL's features. However the denotational semantics of the basic metalanguage of Denotational Semantics, *i.e.* λ-calculus, was extremely difficult to find because of cardinality reasons. One can say that Domain Theory really took off only when Scott, in [53], gave the first mathematical model, D_∞, for λ-calculus, using his famous inverse limit solution of the domain equation *i.e.* $D \simeq [D \to D]$, in the category of Continuous Lattices and Scott-continuous functions. Such functions were suitable to model computable functions because they gave a mathematical account of the *information ordering*. But solutions of domain equations are not completely satisfactory because, usually, domains have too many points and hence do not provide *fully abstract models*. The *operational/observational* semantics are *coarser* than the *denotational* equivalences which these semantics induce. This drawback has been solved brilliantly in many important cases using Games and Strategies [6]. But there is a deeper question, see [34], which has not been satisfactorily solved yet. Is there a Domain Category which is *complete* w.r.t. the theories of λ-*calculus*? Namely are there equivalences which are necessarily enforced upon us, because of the specific way by which the *information ordering* is modeled mathematically in the category, which are not implied by the mere operational rewriting of λ-calculus? The problem is still open. Hence we cannot claim that we have a full account of the semantics of λ-calculus yet.

4 Games and Strategies

In the early 90's a major paradigm shift appeared in Semantics: the *game metaphor*, which allowed for the "explanation" of *proofs (programs) as strategies* and *computation as interaction*. Games first arose, implicitily, in the seminal paper on the semantics of *Linear Logic* by J-Y.Girard, [29], where the new semantics was called *Geometry of Interaction*. Girard proposal stemmed from a very strong criticism to traditional denotational semantics. It deals only with the input-output, extensional nature of computation and does not provide any account for the *dynamics* of computation. Moreover, when it addresses dynamics this is too low-level, syntax-oriented, and ultimately idiosyncratic. The analogy *games-as-propositions* and *strategies-as-realizers*, on the other hand, provides a denotational account of dynamics. In just a few years, Game semantics was extended by Girard himself, Abramsky, Hyland and many others to cover most logics and most features of PL's. Various *categories of games* were introduced which gave rise to many fully abstract and fully complete models.

Games are a pervasive metaphor indeed, which is useful in many diverse scientific areas and social milieux, precisely because games abstract many features of *interactions*. It comes as no surprise that the metaphor emerged in Theoretical Computer Science when the Internet and mobile communication devices turned modern society into the *society of digital interactions* and shifted the *digital revolution* to the *digital communication revolution*. Throughout this paper we used puzzles to promote an interactive particiaption of the reader. And what are puzzles but games! We used puzzles and the game Dixit© for explainig the unexplainable aspects of what counts as an explanation (see Section 2.3). *Turing's test* for judging whether a behaviour is intelligent is based on the *Imitation Game*,

i.e. on the *interaction* made possible by a game. The *Hypergame Paradox*[19] is a game-theoretic analogue of a classical set-theoretic paradox. Conway, [17], has defined a unifying approach to games, as objects of a wonderful mathematical universe given by the the initial co-algebra of the functor $X = \mathcal{P}(X) \times \mathcal{P}(X)$. Conway games are terminating. In [33] we extend his theory to non-terminating games by studying the *final co-algebra* of that functor.

The crucial ingredient in the game metaphor is the *polarity* between the two players making alternating moves, rather than the idea of *winning* the game[20]. Polarity is yet another kind of "*partita doppia*". In the context of game semantics, the concept of *winning strategy*, which is usually more emphasized than that of *strategy*, amounts to a *liveness property*. And this is precisely the essence of many winning strategies based on the preservation of some invariant. Canonical examples are winning strategies in perfect information *Nim-like* games[21].

From Resumptions to Strategies There is a vast literature on Game Semantics and *Geometry of Interaction*, but the presbyopic view on contemporary research makes it difficult to grasp the true underpinnings. So I shall not discuss the many exciting narratives currently circulating. The risk not to be able to see the forest because of the tree in front of us is serious. I will just mention two insightful papers by Abramsky namely [5, 8], a small result and a conjecture.

In [5], Abramsky suggests a possible way of reconciling the denotations of *processes-as-strategies* with previous semantical concepts based on automata. The bridge is given by the concept of *Resumption*. Resumptions where introduced by Milner in 1973 to model Transducers, that is *history-dependent* automata which process streams of words in an *input language*, Σ_{in}, and produce streams of words in an *output language* Σ_{out}. *Resumptions* can be made into a category whose objects are sets and whose morphisms $Hom(\Sigma_{in}, \Sigma_{out})$ are the final co-algebra of the functor $\mathcal{R}(X) = \Sigma_{in} \rightharpoonup (\Sigma_{out} \times X)$. Composition is defined coinductively by $f \circ g(x) \cong ((\pi_1 \circ f)((\pi_1 \circ g)(x)), (\pi_2 \circ f)((\pi_1 \circ g)(x)) \circ (\pi_2 \circ g)(x))$. Resumptions, can be viewed as a primitive form of strategies in the game $(\Sigma_{in}, \Sigma_{out})$, *i.e.* functions from *opponent's moves*, or environment's moves, or symbols in Σ_{in}, to *player's moves*, or system's moves, or symbols in Σ_{out}. *History free* strategies arise as resumptions r such that $\pi_2(r(x)) = r$. The non-deterministic behaviour of the transducer which is not subsumed already by Σ_{in} can be recovered by using the functor $\mathcal{R}(X) = \Sigma_{in} \rightharpoonup \mathcal{P}(\Sigma_{out} \times X)$.

What is remarkable, is that Resumptions can be made into a *traced symmetric monoidal category* defining a suitable notion of *feedback operator* $\mathtt{Tr}^U_{X,Y} : \mathcal{R}(X \otimes$

[19] The game where player I first chooses a terminating game, then Player II starts playing that game, cannot exist because it is terminating, but if it were terminating then player I could choose this very game.

[20] The following *lateral thinking puzzle* makes the point by showing that all games can be played in the *misère* version. "Two jockeys were tired of competing to see who had the fastest horse. So, one day, they decided to take up the opposite challenge: who has the slowest horse? But the race would never start, until ... someone suggested to ...!" The answer is to "swap horses".

[21] A simple example is *race-to-twenty*: starting from $n \leq 20$ the players add either 1 or 2, in turns, to the current sum. The first to reach 20 is the winner.

$U, Y \otimes U) \to \mathcal{R}(X, Y)$. This makes Resumptions a *category of boxes and wires*, where we can find suitable objects, called by Abramsky *GoI*-situations (GoI for Geometry of Interaction), which yield models of Combinatory Logic.

I will not give details, but just provide a concrete example by Abramsky, [8], where the connection between automata and strategies is impressively put to use yielding a *reversible Universal Model of Computation.* Abramsky, abstracting on earlier work [7], defines a structrue of history-free resumptions, $\mathcal{I}$, as follows:
- T_Σ is the language defined by the signature $\Sigma_0 = \{\epsilon\}$, $\Sigma_1 = \{\mathrm{l},\mathrm{r}\}$, $\Sigma_2 = \{< >\}$; terms of the form $r(x)$ are input moves, terms $l(x)$ are output moves;
- $\mathcal{I}$ is the set of *partial involutions* over T_Σ, *i.e.* the set of all partial injective functions $\{f | f : T_\Sigma \rightharpoonup T_\Sigma\}$ such that $f(u) = v \Leftrightarrow f(v) = u$;
- $\mathcal{I}$ is endowed with the structure of a *Linear Combinatory Algebra* (LCA), *i.e.* the linear decomposition of Combinatory Logic, where
- *Replication* is defined by $!f = \{(< t, u >, < t, v >) | t \in T_\Sigma \ \wedge (u, v) \in F\}$;
- linear application is defined by $LApp(f, g) = f_{rr} \bigcup f_{rl}; g; (f_{ll}; g)^*; f_{lr}$, where $f_{ij} = \{(u, v) | (i(u), j(v)) \in f\}$ for $i, j \in \{r, l\}$.

The LCA of Partial involutions $\mathcal{I}$ illustrates how the notion of application makes use of the trace, or feedback, operator. Abramsky's beautiful metaphor of combinators as *boxes and wires* arises from the fact that the denotations of combinators in $\mathcal{I}$ can be viewed as *copy-cat strategies* which link the polarities (input and output moves) of its arguments in a suitable way. The application of two combinators corresponds to linking the input and output wires of the two combinators and *pulling the strings.*

In [8], Abramsky gives an automata-language account of a subset of $\mathcal{I}$ which he calls *Reversible Patern Matching Automata* which he shows is still a Combinatory Logic, in fact a *reversible* universal model of computation. Reversible Pattern Matching Automata are particularly interesting because they provide a remarkable illustration of the potential that moves can have. Moves, which provide the crucial feature of games which is polarity, are usually the most gratuitous ingredient in Game Semantics. In this context, instead, strategies become *reversible* precisely as operations on moves, and the behaviour on some *tell-tale* moves are enough to discriminate results, without having to compute irreversibly the full combinator which would be the result of an application.

The automatic interpretation of Combinators given by Abramsky illustrates in what sense Geometry of Interaction provides a semantics to the *dynamics* of a PL, *i.e.* Combinators in this case. The idyosincratic metalanguage which we use to describe Combinators, receives a straightforward abstract interpretation into a world of Automata. This interpretation provides a first kind of *intensional* equivalence, not yet all that meaningful. The interpreting automaton, in turn, induces a partial involution on the language of moves T_Σ. This is the *execution invariant* of the automaton, which provides the traditional denotational meaning. This is the sense of Girard's "execution formula" [29].

But this is not the end of the story. The equivalence given by partial involutions has contrasting effects. On one hand it is yet too fine, in that it does not equate all that we would like to equate in interpreting λ-calculus. We would like

to enforce Curry's Axioms to equate all the differences arising from the "coding tricks" in the definition of replication, thus obtatining a λ-algebra. But, on the other hand, it equates many more terms than Combinatory Logic would. Of course, this is precisely the main success, and the original reason for introducing Game Semantics, namely defining fully abstract models. But if game models are to be the *ultimate semantics* they should be flexible enough to model just about any semantics even the pure $=_{\beta\eta}$,as we discussed in Section 3.2.

Whether this is possible is yet under investigation. Some partial negative results have been obtained in [20, 33].

Proposition 1. *Let M be a* non-terminating *term of Combinatory Logic, i.e. a term whose head reduction does not terminate, then $[\![M]\!]_{\mathcal{I}} = \emptyset$, i.e. its interpretation is the* empty strategy*.*

Proof (sketch) Use indexed reduction, where the index is the cardinality of the graph of the partial involution. One can see that the interpretation of any term whose head reduction does not terminate is evenutally empty. □

The above result is rather robust and is generalized in [36]. We conjecture that by adding Curry's Axioms the natural bisimilarity on $\mathcal{I}$ yiels a model for Böhm Trees, and that a suitable variant to the definition of application to partial involutions accomodate lazy functions yielding a model of Levy-Longo Trees. Removing the f_{rr} component from application allows for modeling *strict functions* yielding the first *fully abstract model* for the observational theory of Landin's ISWIM, see [21] for the drawbacks using domains. Whether strategies can help to understand the world of non terminating *lambda*-terms is still wide open!

5 Conclusion

We have illustrated some highlights in the exciting history of 60 years of Semantics and Formal Methods, inevitably from a subjective viewpoint. 60 years on we understand that the irreducible plurality of Semantics rather than being a drawback, or the indication that something is missing, is indeed what makes it useful. In this paper we have raised and discussed, sometimes rhapsodically, various phiosophical issues concerning Formal Reasoning, Proof Cultures, Logical Frameworks, the Algebraic/Coalgebraic Duality, and Games. We put forward the thesis that Semantics of Computation is really a "*partita doppia*", a multiple/double check on what we think we understand about Computing. Dualities are at the core. I hope that this paper might have a socratic effect in calling for more attempts at understanding what we mean by Semantics of Computation, or at least in soliciting alternative narratives of its triumphal march.

References

[1] R. P. Nederpelt, J. H. Geuvers, R. C. de Vrijer. Selected Papers on Automath. Vol. **133** Studies Logic, Elsevier, Amsterdam (1984)

[2] Archimede. *Metodo. Nel laboratorio di un genio.* Bollati Boringhieri (2013)

[3] S. Abramsky. Domain Theory in Logical Form. *Ann. Pure Appl. Logic* **51**(1) (1991)

[4] S. Abramsky, R. Jagadeesan. New foundations for the geometry of interaction. *Information and Computation* **111**(1), pp. 53-119 (1994)

[5] S. Abramsky. Retracing some paths in Process Algebra. *Proceedings of the Conference CONCUR'96*, Springer Berlin Heidelberg (1996)
[6] S. Abramsky, R. Jagadeesan, P. Malacaria. Full Abstraction for PCF. *Information and Computation* **163**(2), pp. 409-470 (2000)
[7] S. Abramsky, M. Lenisa. Linear realizability and full completeness for typed lambda-calculi. *Ann. Pure Appl. Logic* **134**(2-3)(2005)
[8] S. Abramsky. A Structural Approach to Reversible Computation. *Theoretical Computer Science* **347**(3) (2005)
[9] P. Aczel. *Non-Well-Founded Sets.* CSLI LN **14**, Stanford (1988)
[10] P. Aczel. Final Universes of Processes. *MFPS*, Springer LNCS **802** (1993)
[11] Yves André. *Qu'est-ce que coagir?: pour une philosophie de la coaction.* Notes d'exposé au Colloque Objet/Relation, ENS, Paris (2014)
[12] J. W. Backus. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. *Proc. Int. Conf. on Inf. Processing*, UNESCO, pp.125-132. Typewritten preprint (1959)
[13] Alain Badiou. *L'être et l'événement.* (1988)
[14] J. Barwise, L. S. Moss. Vicious Circles: On the Mathematics of Non-Wellfounded Phenomena. *CSLI LN* **60**, Stanford (1996)
[15] L. Carroll. What the Tortoise Said to Achilles. *Mind*, **4**, pp.278-280 (1895)
[16] F. Cardone, R. Hindley. Lambda-Calculus and Combinators in the 20th Century. *Logic from Russell to Church*, pp.723-817 (2009)
[17] J. Conway. *On Numbers and Games.*, A. K. Peters (2000)
[18] G.Huet, T.Coquand, C. Phlin-Mohring, H.Herbelin, *et al. The Coq Proof Assistant.*, https://coq.inria.fr.
[19] H. L. Dreyfus, S. E. Dreyfus. Making a Mind vs Modeling a Brain, Artificial Intelligence Back at a Branchpoint. *Daedalus Art. Int.*, **117**(1) (1990)
[20] P. Di Gianantonio, G.Franco, F.Honsell. Game Semantics for Untyped $\lambda\beta\eta$-Calculus. *TLCA '99*, SLNCS **1581** (1999)
[21] L. Egidi, F. Honsell, S. Ronchi. Operational, denotational and logical descriptions: a case study. *Fundam. Inform.* **16**(1), pp.149-169 (1992)
[22] J. L. Lions. Rapport de la Commission d'enqute Ariane 501 Echec du vol Ariane 501(1997). http://www.capcomespace.net/dossiers/espace_europeen/ ariane/ariane5/AR501/AR501_rapport_denquete.htm.
[23] F. B. Fitch. Self-reference in Philosophy. *MInd* **55** (1946)
[24] M. Forti, F. Honsell. Set Theory with Free Construction Principles. *Ann. Scuola Norm. Sup. Pisa* **10**, pp.493-522 (1983)
[25] T. B. Steel. ed. *Formal language description languages for computer programming. Proc. IFIP TC 2 Work. Conf. on Formal Language Description Languages*, Vienna, Sept.15-18, 1964, North Holland, (1966)
[26] Fabio Gadducci. Graph rewriting for the π-calculus. *MSCS*, **17**(3) (2007)
[27] F. Gadducci, U. Montanari. Comparing logics for rewriting: rewriting logic, action calculi and tile logic. *Theor. Comput. Sci.* **285**(2)(2002)
[28] J.-Y. Girard. Linear logic. *Theor. Comp. Sci.* **50**, pp.1-102 (1987)
[29] J.-Y. Girard. Geometry of interaction I: interpretation of system F. *Logic Colloquium '88*, North-Holland (1989)
[30] J.-Y. Girard. The Blind Spot: lectures on logic. *Eur. Math. Soc.* (2011)
[31] A. Gramsci. *Prison Notebooks.* Columbia U. P. (1992)
[32] R. Harper, F. Honsell, G. Plotkin. A Framework for Defining Logics. *Journal of the ACM* **40**(1), pp.143-184 (1993)
[33] F. Honsell, M. Lenisa. Conway games, algebraically and coalgebraically. *Logical Methods in Computer Science* **7**(3) (2011)

[34] F. Honsell, G. D. Plotkin. On the completeness of order-theoretic models of the lambda-calculus. *Inf. Comput.* **207**(5), pp.583-594 (2009)
[35] F.Honsell. 25 years of formal proof cultures: some problems, some philosophy. *Proceeding of LFMTP '13* ACM SIGPLAN Workshop, Boston (2013)
[36] F. Honsell, M. Lenisa. Theories of Automatic Combinators. (draft)
[37] F. Honsell, L. Liquori, P. Maksimović, I. Scagnetto. LLFP: A Logical Framework for Modeling External Evidence using Monads. *LMCS* (2014) (to appear)
[38] B. Jacobs, J. J. M. M. Rutten. An introduction to (co)-algebras and (co)-induction. *Cambridge Tracts in TCS* **52**, CUP (2011)
[39] M. Lenisa. *Themes in Final Semantics*. PhD Th. CS Pisa, TD**6** (1998)
[40] S. Kripke. *Wittgenstein on Rules and Private Language*. Blackwell (1982)
[41] P. Martin-Löf. *Intuitionistic type theory*. Napoli, Bibliopolis (1984)
[42] R. Milner. A Calculus of Communicating Systems. SLNCS **92** (1980)
[43] R. Milner. The use of machines to assit in rigorous proof. *Phil.Trans.R.Soc.Lond.*, A **312** (1984)
[44] R. Milner. Is Computing an Experimental Science? *LFCS Inaugural Lecture* (1986)
[45] R. Milner, J. Parrow, D. Walker. A calculus of mobile processes. *IC*, **100**(1) (1992)
[46] E. Moggi. Notions of Computation and Monads. *Inf. Comp.* **93**(1) (1991)
[47] G. D. Plotkin. Call-by-Name, Call-by-Value and the λ-Calculus. *TCS* **1**(2) (1975)
[48] G. Plotkin. LCF Considered as a Programming Language. *TCS* **5**(3) (1977)
[49] H. Poincaré. *La Science et l'Hypothèse.* Flammarion, Paris (1902)
[50] J. Rutten, D. Turi. On the foundations of final co-algebra semantics: non-well-founded sets, partial orders, metric spaces. *MSCS*, **8** (1998)
[51] J. Rutten. Universal co-algebra: a theory of systems. *TCS* **249** (1) (2000)
[52] W. Scherlis, D. Scott. Semantically based programming tools. *SLNCS* **185** (1985)
[53] D. Scott. Continuous Lattices. in F. W. Lawvere ed., *Dalhousie Conference on Toposes, Algebraic Geometry and Logic*, *SLNM* **274** (1972)
[54] D. Scott. Relating Theories of the Lambda Calculus. *To H. B. Curry Essays in Combinagtory Logic, Lambda Calculus and formalism*, Academic Press (1980)
[55] D. Scott. Domains for Denotational Semantics. *ICALP*, SLNCS **140** (1982)
[56] A. Shopenhauer. The World as Will and Representation. (1844)
[57] A. Sen. Sraffa, Wittgenstein, and Gramsci. *J.Economic Lit.*, **41**(4) (2003)
[58] M. Smyth, G. Plotkin. The Category-Theoretic Solution of Recursive Domain Equations. *SIAM J. Comput.* **11**(4) (1982)
[59] D.Turi, G.Plotkin. Towards a Mathematical Operational Semantics. *LICS 97*, IEEE (1997)