UNIVERSITA' DEGLI STUDI DI TRENTO - DIPARTIMENTO DI ECONOMIA

A COMPUTABLE ECONOMIST'S PERSPECTIVE ON *COMPUTATIONAL COMPLEXITY*

K. Vela Velupillai

Discussion Paper No. 23, 2007

A Computable Economist's Perspective on *Computational Complexity*

K. Vela Velupillai[*]
Department of Economics
University of Trento
Via Inama, 5
381 00 Trento, Italy

Paper Prepared for:
THE HANDBOOK OF COMPLEXITY RESEARCH
Edited by: *J. Barkley Rosser, Jr.*,
Edward Elgar Publishing Ltd.

September 27, 2007

[*]John McCall, Chico Doria and Stefano Zambelli have been educating me, from different points of view, on many fascinating visions of the thorny concept of complexity. More than a decade and a half of constant, stimulating, discussions with John McCall on Kolmogorov – algorithmic – complexity and theories of probability have been a source of great pleasure and consistent intellectual adventures. Not being particularly competent in the actual programming of theoretically structured computing models, I have had to rely on Stefano Zambelli for instruction and advice on the felicitous link between theory and application that is the hall mark of computational complexity theory. More recently, I have had the pleasure and privilege of being able to pick Chico Doria's fertile brain for exotic ideas about the $P = ?NP$ question. Years ago, now bordering on almost two decades, the decision by Shu-Heng Chen to write his UCLA dissertation on the application of algorithmic and stochastic complexity theories, with me as his main thesis advisor, forced me to try to be at least one-step ahead of his incredible learning abilities. Barkley Rosser's stimulating visit to Trento last year provided much inspiration on many topics related to a broader perspective on complexity. I wish I could blame them, and not have to plead guilty, for the infelicities and errors that remain!

Abstract

A computable economist's view of the world of computational complexity theory is described. This means the model of computation underpinning theories of computational complexity plays a central role. The emergence of computational complexity theories from diverse traditions is emphasised. The unifications that emerged in the modern era was codified by means of the notions of efficiency of computations, non-deterministic computations, completeness, reducibility and verifiability - all three of the latter concepts had their origins on what may be called 'Post's Program of Research for Higher Recursion Theory'. Approximations, computations and constructions are also emphasised. The recent real model of computation as a basis for studying computational complexity in the domain of the reals is also presented and discussed, albeit critically. A brief sceptical section on algorithmic complexity theory is included in an appendix.

1 Prologue

"There are many levels of complexity in problems, and corresponding boundaries between them. Turing computability is an outer boundary, ... any theory that requires more power than that surely is irrelevant to any useful definition of human rationality. A slightly stricter boundary is posed by computational complexity, especially in its common "worst case" form. We cannot expect people (and/or computers) to find exact solutions for large problems in computationally complex domains. This still leaves us far beyond what people and computers actually CAN do. The next boundary... is computational complexity for the "average case" .... .. That begins to bring us closer to the realities of real-world and real-time computation. Finally, we get to the empirical boundary, measured by laboratory experiments on humans and by observation, of the level of complexity that humans actually can handle, with and without their computers, and - perhaps more important – what they actually do to solve problems that lie beyond this strict boundary even though they are within some of the broader limits."

Herbert Simon, Letter to the author, 25 May 2000.

As it happens, Herbert Simon's lucid letter to me, delineating a rough ordering along the complexity scale for problem solving by means of a machine model of computation, was dated 25 May, 2000. The day before, on 24 May, 2000, Arthur Jaffe, as the then President of the *Clay Mathematical Institute* had announced the *Seven Millennium Problems*[1], the solutions for which – in the form of a proof or a counterexample – would earn prizes of a million dollar each. It was also fittingly, at least in the context of this brief paper, announced to commemorate the centennial of Hilbert's famous lecture on '*Mathematical Problems*', [28], given at the meeting of the International Congress of Mathematicians in Paris, in August, 1900. Fittingly, because the most enduring model of computation underpinning almost all the frontier developments in computational complexity theory is the Turing Machine. This device was conceived and

[1]The seven millennium problems, listed alphabetically, are, [29], p.655:

1. The Birch/Swinnerton-Dyer conjecture;
2. The Hodge conjecture;
3. The Navier-Stokes equations has smooth solutions;
4. P is not NP;
5. The Poincaré conjecture;
6. The Quantum Yang-Mills theory exists with a mass gap;
7. The Riemann Hypothesis;

implemented by Alan Turing in the process of answering the *Entsheidungsproblem*, formulated by Hilbert, as a way of answering unambiguously and decisively, Brouwer's sustained criticisms of standard mathematics and its methodological bases, particularly its proof theoretical underpinnings. The earliest formulation, at least by Hilbert, of the *Entsheidungsproblem* can be traced back to the 10th of Hilbert's 23 Problems in that famous Paris Lecture of 1900.

The fourth in the alphabetic list of seven problems is the quintessential problem of computational complexity theory: $P = ?NP$. Computational complexity theory is doubly related to mathematical economics and economic theory[2]: first, as a theory of the *efficiency of computations*, it is best viewed as the economic theory of computations; secondly, having at its central core the paradigmatic combinatorial, intractable, $NP - Complete$, (in)famous *TSP*: the *Travelling Salesperson's Problem*[3]. In the former case, it must first be remembered that the pure theory of computations abstracts away from all kinds of resource constraints; computational complexity theory, the 'applied' theory of computation, is its finessing, taking explicit account of resource constraints, typically time and space constraints. In other words, computational complexity theory is the economic theory of computation.

Computational complexity theory considers problem solving, with *a model of computation* explicitly underpinning it, as *decision problems*[4]. In these two aspects it differs fundamentally from economics: in economics, whether in its mathematical modes or not, there is no explicit model of computation underpinning its optimization problem formulation – which is the second sense in which it differs from economic theory; i.e., in economics the quintessential problem solving framework is one of *constrained optimization*, albeit without ever incorporating a model of computation which underpins the construction, detection or computation of the optimum. Generally, the latter can be transformed in to a form of the former (see below for an explicit and paradigmatic example of converting an *Integer Linear Programming* (*ILP*) problem into a *conjunctive normal form* (*CNF*) *decision problem*).

Anyone who tries to learn a programming language is introduced to it, in the

[2]One of the modern pioneers of computational complexity theory, Richard Karp, perceptively noted, [32], p.464, Italics added:

> "[I] do think there are some very worthwhile and interesting analogies between complexity issues in computer science and in economics. For example, economics traditionally assumes that the agents within an economy have universal computing power and instantaneous knowledge of what's going on throughout the rest of the economy. Computer scientists deny that an algorithm can have infinite computing power. *They'e in fact studying the limitations that have arisen because of computational complexity. So, there's a clear link with economics.*"

[3]For reasons of 'convenience' I may abscond from gender neutrality when referring to the *TSP*.

[4]A *decision problem* asks whether there exists an *algorithm* to *decide* whether a mathematical assertion does or does not have a proof; or a formal problem does or does not have a solution. Thus the definition makes clear the crucial role of an underpinning model of computation; secondly, the answer is in the form of a yes/no response. Of course, there is the third alternative of '*undecidable*', too, but that is a vast issue outside the scope of this paper.

very first lesson or lecture, via instructions on how to write a simple program, such as: Print, 'Hello, World'; or: Print, 'n', where 'n' is an arbitrary, given, integer. But anyone who tries to learn the theory of programming is taught, again in the very first lecture, that *there is no algorithm for finding the shortest program for printing such an integer, 'n'*, and halting! Put another way, this negative answer is for the following absolutely simple 'minimization' problem:

Problem 1 *Find the (or one of the) shortest programs for printing a given integer n, and halting.*

At much more sophisticated level of *abstract computational complexity theory*, there is a kind of analogue of this simple non-maximization result.

Theorem 2 *Blum's Speedup Theorem*[5]
$\forall$ *Complexity measures – i.e., criteria of optimization,* $\exists$ *total recursive functions s.t.,* $\nexists$ *optimal programs – i.e., algorithms – to compute them.*

Whatever the underlying model of computation, my own view is that the best way to introduce the subject of computational complexity theory to economists would be by way of a three stage process. First, an introduction to economics in the computational mode[6]; this entails an underpinning of economic theory in the mathematics of recursion theory or constructive analysis, both of which are intrinsically algorithmic. Second, a discussion of the *intrinsic* complexity of the kind of functions and sets that are in the domain of recursion theory and constructive analysis; i.e., the intricacy of the functions and sets, quite apart from the difficulty of actually computing them. Finally, an analysis of the *computational* complexity of the functions and sets; i.e., the actual difficulty of computing the functions or deciding membership or other properties of the sets. But such a strategy is clearly not feasible within the limited scope of one paper. Therefore, I will have to forego any and all discussions of the abstract approach to computational complexity theory, an area pioneered by Manuel Blum. I will also have to eschew any attempt at introducing an economic theory in its

[5] The Blum Speedup theorem should be presented in conjunction with the equally important Gap and Compression theorems of abstract computational complexity theory, especially when presenting the subject to an economic audience. These are the kinds of results from abstract computational complexity theory that make Simon's behavioural economics, underpinned by *boundedly rational* agents *satisficing* in the face of *decision problems*, against a backdrop of a model of computation, the natural framework for the economics of choice under constraints. Instead, economists are weaned in the mathematics of real analysis and pure existence proofs which are then, at another stage, given computational, cognitive and numerical content, entirely divorced from the original problem formulation.

[6] That which I have called 'computable economics'. This is different from the currently fashionable reference to 'computational economics' where, in general, there is no underlying model of computation. The paradigmatic examples of such computational economic models are, in macroeconomics, the recursive computational equilibrium, *RCE*, model; in microeconomics, the (in-)famous example is, of course, the computable general equilibrium, *CGE*, model. Neither of these models are underpinned by any formal model of computation. The connection between the mathematics of the theory and the mathematics of the computation (or the mathematics of the numerical analysis) is non-existent.

computational mode. Instead, I will have to remain at a fairly superficial level of the concrete, applied part of recursion theory. Regrettably, therefore, I shall not be able to develop the full implications of the above kind of fundamental theorems - the Speedup, the Gap and the Compression theorems - in this paper. The perceptive reader will know how to keep these results at the back of the mind – and if such an elusive person also equips herself with an idea of Simon's Behavioural Economic research program – or to seek them out, when, and if, the issues I will actually discuss lead to questions about orthodox formulations of mathematical economics in general, and choice theory in particular.

The three most important classes of decision problems that almost characterise the subject of computational complexity theory are the P, NP and $NP - Complete$ classes. Concisely, but not quite precisely, they can be described as follows. P defines the class of *computable problems* that are solvable in time bounded by a *polynomial* function of the size of the input; NP is the class of computable problems for which a solution can be *verified* in *polynomial time*; and a computable problem lies in the class called $NP - Complete$ if every problem that is in NP can be *reduced* to it in *polynomial time*.

It is important to observe the following features characterising these three classes (and will, eventually, come to characterise every finer, nuanced, classification, too). Firstly, the classification is of *computable problems*; secondly the ubiquity of '*polynomial*' in the definitions; thirdly the role of the two notions of '*verify*' and '*reduce*'. Finally, conspicuous by its absence is any mention of *how*, in the case of the class NP, *one arrived at a solution* which is *verifiable* in *polynomial time*. Magic, intuition, leap of faith, appeal to oracles, ESP, whatever is allowed in finding solutions; only algorithmic means are allowed in verifying their truth status. One of the greatest 'solution finders' was Srinivasa Ramanujam; one of the great algorithmic problem solvers was George Polya. There was once, in fact, an interesting encounter between the magician and the problem solver (albeit - as befits a magician - many years after his death). It illustrates the real life conundrums of verifying solutions versus devising them, in the first place, [80], p.73; italics added:

> "One day in 1951 while Polya was visiting Oxford, he borrowed from Hardy his copy of Ramanujan's notebooks. A couple of days later, Polya returned them in almost a state of panic explaining that however long he kept them he would have to keep attempting to *verify* the formulae therein and never again would have time to *establish* another original result of his own."

1.1 Preamble

> "The 'P versus NP' question ..... focuses on a simplified view of the goals of (efficient) computations. Specifically, [this focuses] on efficient procedures that always gives the exact answer. In practice, one may be content with efficient procedures that 'typically' give an 'approximate' answer."
>
> Goldreich, [?], p.522; italics in the original.

Economists, flushed with the wide and easy availability of powerful symbol manipulating software packages, such as *Mathematica* and *Matlab*, and the ubiquity of the computer, approximate, compute and evaluate almost indiscriminately. Is there any issue, of any reputable Journal in economics, with a modicum of computations, estimations or simulations, let alone serious mathematical calculations, devoid of arbitrary Taylor series expansions, thoughtless linearizations and gratuitous statements about choosing analytically simple functions ('*w.l.o* generality', as the cliché goes!) that facilitate computations without approximations?

Efficiency of computations and *approximations* are highlighted in the above important questions posed in [6]. They are the two decisive features characterising the theory of computational complexity. Moreover, it must be kept in mind that economic theory is predominantly mathematized in terms of real analysis. Hence, any consideration of the computational complexity of economic processes, economically motivated functional assumptions and economic computations must focus on some kind of real number model of computation. These are the issues I shall try to discuss in this paper.

The paper is divided into three main sections, in addition to this Preamble. Each of the main sections are sub-divided into further sub-sections dealing with more focused topics.

The next section,§2, 'Preliminaries', is sub-divided into two sub-sections. In the first subsection, 'Introduction', some of the terminological and conceptual ideas and hurdles are outlined with examples. In the next sub-section, titled 'Exotica', I illustrate some conceptual issues that figure crucially in computational complexity theory; notions such as easy and fast verification of a proposed result or theorem, the speed of convergence, the nature of the growth of functions and the subtle and nuanced differences between deriving a theorem and verifying its truth value.

Section 3 is a mixture of historical settings for classic problems of computational complexity theory and frontier topics in the subject.

Section 4 is a telegraphic presentation and critical discussion of a model of real computation, recently advocated with tremendous conviction in [4].

In the concluding section 5, there is an attempt to reflect on the future of a more computationally oriented economic theory, where questions of the efficiency of computations and approximations can be posed and discussed more coherently and consistently. Related speculative thoughts on alternative, constructively founded, mathematical economics suggests that Herbert Simon's research program is the only hope for those of us who want to be able to analyse the efficiency of computations .

2 Preliminaries

2.1 Approximating π,Deciding γ,Calculating 2^{64} and all that!

"The kinds of questions we pose are:

(1). How much work (by various types of *computational* or *approximation* measures) is required to *evaluate* n digits of a *given function* or *number*?

(2). How do analytic properties of a function relate to the *efficacy* with which it can be *approximated*?

(3). To what extent are analytically *simple* numbers or functions also easy to *compute*?

(4). To what extent is it easy to *compute analytically simple functions*?

Even partial answers to these questions are likely to be very difficult.

(6). Why is the Taylor series often *the wrong way to compute familiar functions*?

..."

Borwein & Borwein, [6], p. 589; italics added.

'*Given function*' is a phrase that would make sense to most analytically minded people, especially those who are in the habit of interpreting the notion of a *function as a rule* (rather than the Dirichlet-Kuratowski inspired interpretation as a 'graph'). But what is the meaning of a '*given number*', particularly a 'given *real* number'? Take, for example, π. By definition, it is 'the ratio of the circumference to the diameter of a circle'. Given this definition, n digits of it can be evaluated – i.e., $\exists$ an algorithm to evaluate it, say, to $n = 20$ digits: 3.14159265358979323846 .. . Is there any other way *a real number* can be '*given*', except by a rule defining it? Suppose the defining rule for a real number - or, even a function - is *non-constructive*; what can be done to 'evaluate' it? What, indeed, can be done to 'approximate' it? Archimedes inscribed polygons to approximate the circumference of the inscribing circle. Is there, then, a 'fastest' algorithm to inscribe a polygon to a 'given' circle? How *fast* does an *algorithm* to evaluate the circumference of a given circle *converge*?

A slightly more exotic example is *Euler's constant*[7], usually denoted by γ. To the best of my knowledge, to this day, we do not even know whether it is *rational or not*, even though the formula defining it has the appearance of being a well defined algorithm[8].

[7] Which is, of course, not the same as its more famous 'cousin', *Euler's number, e*.

[8] γ is defined as the difference between the *harmonic series* and the *natural logarithm*, in the limit:

$$\gamma = \lim_{n\to\infty}\left[\left(\sum_{k=1}^{\infty}\frac{1}{k}\right) - \log n\right] = \int_1^{\infty}\left(\frac{1}{\lfloor x\rfloor} - \frac{1}{x}\right)dx$$

Of course, from a fairly young age most of us would have been taught the *harmonic series*; and even earlier, the *natural log*. An enlightened or subtle teacher could use the definition to teach students, at an early age, that the notion of a '*given number*' may entail eternal perplexities.

Most readers will be aware of the famous example of the request for just one grain of rice on the first square of a chess board and, then, a doubling, sequentially, on each of the remaining 63 squares, by the winner of a game of chess against a mythical King[9]. We all know – depending on the initial condition – that the 64th square would require the King to place 2^{64} grains of rice on that final one! This monstrosity emerges naturally in a 'simple' and 'small' typical combinatorial economic problem. There is no more paradigmatic example of intractability than that quintessential economic combinatorial problem: the *travelling salesman's problem* (*TSP*). Consider the following *TSP*[10] (see Figure 1, below).

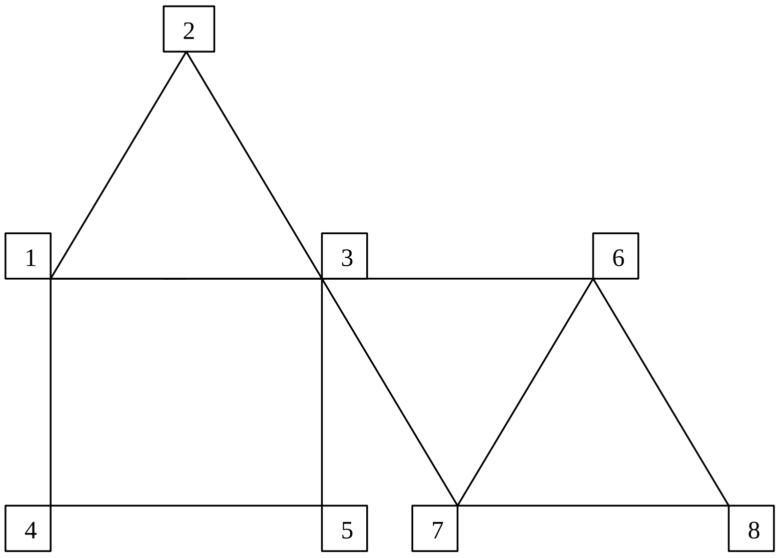

Figure 1: Figure 1: TSP

[9]I was myself taught this fascinating story in childhood as the 'paal payasam' story of a game of chess between a King and Lord Krishna. The King, having inevitably lost the game to a God – although Hindu Gods are not infallible – was requested by Lord Krishna to prepare, and distribute freely, the rice and milk based drink, 'paal payasam'. No doubt the story, in other traditions, is narrated with other metaphors.

[10]This example is taken from [2].

The distance between each of the nodes is 10 miles. Then:

Problem 3 *Can the salesman start at node* 1*and visit each of the nodes in a journey of* 70 *miles?*

This classic TSP problem can be converted to the following equivalent satisfiability problem of the prepositional calculus:

Problem 4 *Assign truth values to* A_t^m *s.t the following prepositional formula is true:*

$$E = J \ \& \ V \ \& \ N \ \& \ A_1^0 \tag{1}$$

Where:

A_t^m: The salesman is at node t, after $10 \times m$ miles;

$J_y^x \doteq \left(A_i^m \rightarrow \left(A_1^{m+1} \vee A_2^{m+1} \vee .. \vee A_{i-1}^{m+1} \vee A_{i-1}^{m+1} \vee ... \vee A_7^{m+1}\right)\right)$; and each A_i^m is true.

Put:

$$J = J_1^0 \& J_2^0 \& .... \& J_8^0 \& J_1^1 \& J_2^1 \& ..... \& J_8^1 \& ...... \& J_8^6$$

Each node has to be visited:

$$A_i^0 \vee ...... \vee A_i^7; \ \forall i = 1, ..., 7;$$

Put:

$$V = \wedge \left(A_i^0 \vee ...... \vee A_i^7\right); \forall i = 1, .., 8;$$

Denote the condition that the salesman can only be at one node at a time by:

$$N_i^j = \left(A_i^j \rightarrow \sim A_1^j\right) \& \left(A_i^j \rightarrow \sim A_2^j\right) ..... \left(A_i^j \rightarrow \sim A_8^j\right)$$

Put:

$$N = \wedge \left(N_i^j\right); \forall i = 1, ..., 8; j = 0, ..., 7;$$

How many possible truth value assignments have to be tried for (1)? There are 64 different values of A_t^m, for which truth assignments of T, F have to be tried: i.e., 2^{64}!

Frankly, almost all the questions of computability, computational complexity and approximation theories can be framed in the context of the above seemingly simple problems. The hallmark of the weird and wonderful world of complexity theory is its feature - which I have tried to illustrate with the above examples - of hiding intricacies and intractabilities behind a façade of simplicities. Lurking behind the simple formulations, assertions and frames are the curses of dimensionality, exponential growth and logical depth.

2.2 Decision Problems as Optimization Problems

"Alternatively, you might *find* a polynomial-time algorithm for an NP-complete problem. If you do, then you have, in effect, found polynomial-time algorithms for *all* NP problems, and the Clay Mathematics Institute will make you a millionaire. But that million will pale beside what you you'll gain by cracking all the cryptographic systems based on the difficulty of NP problems."

Barry Chipra, [**?**], p.87; italics in the original.

Consider the following three-variable Boolean formula:

$$\neg x_3 \wedge (x_1 \vee \neg x_2 \vee x_3) \tag{2}$$

Just as in the case of equations with integer (or rational) values, given a truth assignment $t(x_i) = 1$ or 0 for each of the variables x_i $(i = 1, ..3)$, the above Boolean formula can be evaluated to be true or false, globally. For example the following assignments gives it the value *true*: $t(x_1) = 1; t(x_2) = 1; t(x_3) = 0$. Boolean formulas which can be made true by some truth assignments are said to be *satisfiable.*

Now consider the Boolean formula:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \{\neg x_2\}) \wedge (x_2 \vee \{\neg x_3\}) \wedge (x_3 \vee \{\neg x_1\}) \wedge (\{\neg x_1 \vee \{\neg x_2\} \vee \{\neg x_3\}) \tag{3}$$

Remark 5 *Each subformula within parenthesis is called a clause; The variables and their negations that constitute clauses are called literals; It is 'easy' to 'see' that for the truth value of the above Boolean formula to be true all the subformulas within each of the parenthesis will have to be true. It is equally 'easy' to see that no truth assignments whatsoever can satisfy the formula such that its global value is true. This Boolean formula is unsatisfiable.*

Problem 6 *SAT – The Satisfiability Problem*

Given m clauses, $C_i(i = 1, \ldots, m)$, containing the literals (of) $x_j(j = 1, \ldots, n)$, *determine* if the formula $C_1 \wedge C_2 \wedge \ldots\ldots. \wedge C_m$ is *satisfiable.*

Determine means 'find an (efficient) algorithm'. To date it is not known whether there is an efficient algorithm to solve the satisfiability problem – i.e., to determine the truth value of a Boolean formula. In other words, it is not known whether $SAT \in \mathbf{NP}$. But:

Theorem 7 $SAT \in \mathbf{NP}$

Now to go from here to an optimization framework is a purely mechanical affair. Denoting the union operator as ordinary addition and the negation operator related to arithmetic operators as: $\neg x = (1 - x)$ and noting that it is necessary, for each clause C, there should, at least, be one true literal, we have, for any formula:

$$\sum_{x \in C} x + \sum_{x \in C} (1 - x) \geq 1 \tag{4}$$

With these conventions, the previous Boolean formula becomes the following *integer linear programming* (*ILP*) problem:

$$x_1 + x_2 + x_3 \geq 1 \tag{5}$$

$$x_1 + (1 - x_2) \geq 1 \tag{6}$$

$$x_2 + (1 - x_3) \geq 1 \tag{7}$$

$$x_3 + (1 - x_1) \geq 1 \tag{8}$$

$$(1 - x_1) + (1 - x_2) + (1 - x_3) \geq 1 \tag{9}$$

$$0 \leq x_1, x_2, x_3 \leq 1, \text{ and integer} \tag{10}$$

Definition 8 *A Boolean formula consisting of many clauses connected by conjunction (i.e., $\wedge$) is said to be in Conjunctive Normal Form (CNF).*

Remark 9 *A CNF is satisfiable iff the equivalent ILP has a feasible point.*

Clearly, the above system of equations and inequalities do not, as yet, represent an ILP since there is no 'optimisation'. However, it can be turned into a complete ILP in the ordinary sense by, for example, replacing the first of the above inequalities into:

$$Max\ y,\ s.t:\ x_1 + x_2 + x_3 \geq y \tag{11}$$

Remark 10 *The formula is satisfiable iff the optimal value of y, say $\hat{y}$ exists and satisfies $\hat{y} \geq 1$.*

Finally, we have Cook's famous theorem, rounding off all these connections and bringing into the fold of computational complexity theory, the quintessential combinatorial economic optimization problem:

Theorem 11 *Cook's Theorem*
SAT is **NP** *− Complete*

2.3 Exotica

2.3.1 The Question of 'Fast and Easy *Verifiability*'

In the *Preface* to *Mersenne*'s *Cogitata Physico-Mathematica*, a proposition about *perfect numbers*[11] implies, $\forall p \leq 257, (p \in \mathbb{N}), 2^p - 1$ is *prime only for* $p = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127$ & 257. This proposition, made in 1644, was *falsified* five times between 1883 and 1922, for 61, 67, 89, 107 and 257. The most memorable of the falsifications, at least anecdotally,([91], p.8)[12], was the one by F.N. Cole in a talk given to the American Mathematical Society on 31 October, 1903[11]. Apparently, without uttering a word, he is reputed to have gone to the blackboard and written:

$$2^{67} - 1 = 193707721 \times 761838257287$$

Professor Cole is then supposed to have proceeded 'to perform the long multiplication of the integers on the right hand side to derive the integer on the left.' Given the probable mathematical calibre of the audience, it is not surprising that no questions were asked!

This example highlights six aspects of a key issue in computational complexity theory, remembering that its mathematical foundations lie in computability theory. First of all, the *falsification* of the proposition can be achieved by showing, as Cole did with $p = 67$ (and Kraichik did with $p = 257$), that one of the eleven values of$p$ is *composite*; or, secondly, by showing that some of the other 245 possible values for p are, in fact *prime* (as was done for $p = 61$ by Pervusin, in 1883, and, for $p = 89$ & 167, by Powers, in 1914[13]). Thirdly, by devising a general formula which provides a rigorous criterion, $\forall p > 2$, for verifying primality of $2^p - 1$, as was provided by D.H. Lehmer in 1931. Fourthly, the correctness of Cole's factorization is *easy* to *verify quickly*: easy because long multiplication is an elementary process; quick, since the required multiplication can be speeded up almost indefinitely. Fifth, the 'proof' of Cole's result is almost scandalously short. Finally, how Cole (and Kraïchik) arrived at their results seem to be a non-issue; it may or may not have been *hard*[14]. On the other hand,

[11] A number is said to be *perfect* if it is equal to the sum of all its proper divisors (eg., $6 = 1 + 2 + 3$; $28 = 1 + 2 + 4 + +7 + 14$).

[12] This example has become part of the folklore of dramatic examples in computational complexity theory (cf, for example, [56] and [92], too).

[13] Characteristically illuminating discussions of all these results and formulas can be found in the wonderfully entertaining book by Rouse Ball and Coxeter ([62], ch.1)

[14] Cole is supposed to have claimed that it took him 'three years of Sundays' to arrive at the result ([91], p.8). How long it may have taken Ramanujam to arrive at, for example, the formula for $p(n)$, the number of partitions of n, we do not know; nor do we know whether it was *hard* or *easy* for him to 'derive' them. Very little is known about the thought or other processes used by Ramanujam to arrive at his remarkably complex formulas, except for a poignant remark by his 'mentor' at Cambridge, G.H. Hardy, regarding $p(n)$, [80], p.52, italics added:

> "Ramanujan was the first and up to now, the only, mathematician to discover any [properties of $p(n)$]; and *his theorems were discovered in the first instance, by observation.*"

the Lehmer criterion, built up from the method developed by Lucas, follows the usual mathematical development of a criterion or result as a theorem.

2.3.2 Approximation, Convergence and Divergence

Smorynski refers to Hardy's classic text on *Orders of Infinity* for the following two series and their convergence and divergence behaviours, [74], pp. 360-1. First consider the following convergent series:

$$A : \sum_{n=3}^{\infty} \frac{1}{n \log n \left(\log \log n\right)^2}$$

This series *converges* to 38.43; however it requires $10^{3.14 \times 10^{86}}$ terms before even a two-decimal accuracy can be achieved! What, under such circumstances, is the numerical content and computational meaning of 'convergence'?

Next, consider the series:

$$B : \sum_{n=3}^{\infty} \frac{1}{n \log n \left(\log \log n\right)}$$

This series diverges; however, 'the partial sums exceed 10 only after a googolplex of terms have appeared'! Does it matter, for an applied science, that this series diverges? Years ago, the distinguished Cambridge applied mathematician and polymath, Sir Harold Jeffreys made a pertinent observation regarding divergent series in the context of applied sciences:

> "Enough work has been done on asymptotic and other divergent series to prove that an infinite series may be useful in physical applications *without being convergent* in the sense defined in works on pure mathematics. It is less generally realized that the converse is equally true, that *a series may be convergent and yet be totally useless for the purpose of calculating the value of the function represented.*
>
> ..... In evaluating a function one is limited to expansions that can be actually summed *numerically in the time available*, and it is therefore necessary that the first few terms shall give a good *approximation* to the function. Those terms once obtained, the rest of the series is of no interest to the physicist. ...."
>
> [30], pp.241-2; italics added.

Jeffreys conflates, from the point of view of the applied sciences, the issues of numerical *computation*, *convergence*, *divergence*, *approximation* and the time constraint that should be considered in a numerical evaluation. All of these issues are explicitly and abundantly evident in the behaviour of the two series

Incidentally, it took the PC on which I am writing this document, using *Mathematica Version 5*, 0.016 seconds to factor $\left(2^{67} - 1\right)$!!

above, A & B. Computational complexity theory shares both the vices and virtues here. Being an asymptotic theory, computational complexity theory tends to disregard the kind of common-sense factors mentioned by Jeffreys. But when viewed as the economics of computability theory - i.e., theories of the efficiency of computations - and against the backdrop that all efficiency theorems in economic theory are 'asymptotic theories', this is not necessarily an incongruence. Moreover, the efficiency criteria for computation considered in computational complexity theory is very explicit in a fundamental way about time resources. Finally, there is the issue of approximation. Here, even in computational complexity theory - but, perhaps, not in algorithmic complexity theory[15] - the important distinction between the theory of exact approximation and the approximations intrinsic to numerical analysis is never made explicit, even in the new and innovative theory of computational complexity of real computations developed by Smale and his associates, [4].

2.3.3 Growth of Functions and Orders of Growth

> "If we drop the requirement of computability – well, there is Tibor Rado's Busy Beaver Function. This irrepressible little fellow grows more rapidly than any theoretically computable function – a feat of not particularly great magnitude: ...."
>
> [75]

In economics the implications of exponential expansion is most vividly illustrated in the 'power of compound interest'[16], via neoclassical models of pure growth[17] and in Malthusian population theory. In the theory of computational complexity theory the key wedge between P and NP is driven by the difference between polynomial and exponential growth rates of resources required for computation. In a sense, economists do not seem to realize that the difference between the modesty of 'simple interest' and the 'power of compound interest' is exactly analogous to the mathematical difference between countable and uncountable sets, in a precise sense:

[15] See the remarks and discussions in the appendix.

[16] Keynes, in speculating on the *Economic Possibilities for our Grandchildren*, ([34], p.361), perceptively noted, (italics added):

> "From [the sixteenth century] until to-day the power of accumulation by compound interest,..., was re-born and renewed its strength. And the *power of compound interest over two hundred years is such as to stagger the imagination.*"

[17] On a neoclassical exponential growth path, at the rate of growth of per capita income at g, with an initial value of income at y_0, we have:

$$y(t) = y_0 e^{gt}$$

Thus, countries like China, India and Brazil, growing at double-digit rates, these days, would double their per capita income in under seven years. How long can this go on and what kind of resources are being used to achieve these kinds of 'growth miracles', ([44]?

"The finite-infinite analogy .... states that polynomial vs. exponential growth corresponds in some sense to *countability vs. uncountability.* The basis for this analogy is that exponential growth and uncountability both frequently arise from *the power set construction.*"

[70], p. 62; italics added.

Put succinctly, for the purposes of computational complexity theory one must be able to understand the sense in which 2^n grows faster than n^2!

In the following definition, formulae and theorems[18], f, g are functions from $\mathbb{N} \rightarrow \mathbb{N}$.

Definition 12 $f(n) = O\left(g\left(n\right)\right)$, *if* $\exists\ c, n_0 \in \Re$, *such that,* $f(n) \leq cg\left(n\right), \forall n \geq n_0$.

Definition 13 *If* $f(n) = O\left(g\left(n\right)\right)$ & *if* $g(n) = O\left(f\left(n\right)\right)$, *then* f & g *have the same rate of growth.*

Proposition 14 *Let* p *be a polynomial of degree* r. *Then* p & n^r *have he same rate of growth.*

Theorem 15 $\forall k > 1, k^n$ *grows faster than any polynomial* p.

3 Computational Complexity Theory: A setting

"Complexity .... is ... a weaselword used to avoid commitment to a definite and clear thought."

pace Fritz Machlup[19]

The world of theories of complexity is many-splendoured. Quite apart from 'orthodox' computational complexity theory, there are the weird and wonderful world of algorithmic complexity theories, *diophantine complexity theory*, *topological complexity theory* – to mention a few of the fields that are especially and explicitly based on a formal model of computation. I am not convinced that algorithmic complexity theory should have been referred to with the sobriquet 'complexity' adorning it in its 'title'. A brief discussion of this view, bordering on an extended comment, is attempted in the appendix to this paper. The background necessary to discuss diophantine complexity – essentially providing an economic context for the *unsolvability of Hilbert's tenth Problem* – will make this already over-long and fractured paper even more so. The best I can do at this point is to refer to the brief remarks in [87], chapter 7. As for topological

[18]There is no need to restrict the range of f, g to $\mathbb{N}$; some prefer to work with the range $\mathbb{R}$.

[19]Fritz Machlup's original remark was about 'structure', [45]:

"Structure ... is .. a weaselword used to avoid commitment to a definite and clear thought."

complexity, even a brief definition and motivating discussion of this concept would require an extended presentation of the model of real computation in [4] (referred to, occasionally, as the *BCSS model*). All I can do here is to give a telegraphic definition of topological complexity in the brief discussion of the *BCSS model*, given in the next main section.

Rosser's interesting outline, [61], of this world gives a much broader and far more illuminating perspective on complexity theory and I refer the interested reader to it for a concise overview. In this section, after what I call a 'potted proto-history' for the setting, *a version of orthodox*[20] computational complexity theory, largely against the familiar backdrop of linear programming, is outlined. As a result of the conundrums arising in this discussion, *a brief sketch of a complexity theory for models of real computation* continues the themes in the next main section.

3.1 Potted Proto-History

"The fact is that every writer creates his own precursors."

Borges, Kafka and His Precursors, in: Labyrinths, p. 201.

On Spring Equinox day, 20th March 1956, Gödel wrote, in his last letter to von Neumann:

> "It is evident that one can easily construct a Turing machine which, for each formula F of the predicate calculus and for every natural number n, will allow one to decide if F has a proof of length n. Let $\Psi(F, n)$ be the number of steps that the machine requires for that and let $\varphi(n) = \max_F \Psi(F, n)$. The question is, how fast does $\varphi(n)$ grow for an optimal machine. *One can show that $\varphi(n) \geq Kn$. If there actually were a machine with $\varphi(n) \sim Kn$ (or even only with $\sim Kn^2$), this would have consequences of the greatest magnitude.* That is to say, it would clearly indicate that, despite the unsolvability of the Entsheidungsproblem, the mental effort by the mathematician is the case of yes-or-no questions could be completely replaced by machines. One would indeed have to simply select an n so large that, if the machine yields no result, there would then also be no reason to think further about the problem."
>
> [23], p.10; italics added.

Essentially, Gödel is conjecturing on the implications of the existence of a polynomial time algorithm for the *satisfiability* problem! As always this 'greatest logician since Aristotle', as von Neumann referred to him, had an uncanny prescience about new directions in applied recursion theory – on the eve of the emergence of the newly codified fields of computational complexity theory and algorithmic complexity theory.

[20]This may well be the only time in my academic life that I find myself a defender of orthodoxy!

Furthermore, the important concepts of *reducibility* and *completenes* were first introduced in Post's seminal paper, [54], and had become, by the time these concepts were fruitfully harnessed in the codifying of computational complexity theory at the frontiers, part of what has come to be called 'Higher recursion Theory', or the study of *recursively enumerable sets and degrees*[21]. Thus, there is, at least with hindsight, a natural transition from classic recursion theory – from the Gödel, Church, Turing, Rosser, Kleene, Post period of 1931-1944 – to the gradual implementation of 'Post's program' on degrees of solvability, say maturing around the period 1965-1980, to the overlapping and intertwined conceptual development of the theoretical framework of modern computational complexity theory.

Alexander Schrijver's monumental 3-volume, encyclopaedic treatise on *Combinatorial Optimization, [68]* will remain the classic on many things, but, surely, above all, on the history of every possible concept of relevance for complexity theory from the point of view of combinatorial optimization. In particular, I can do no better than refer any serious scholar of the history of the ideas that shaped classical computational complexity theory to these three exhaustive and excellent volumes for full details of every possible source, placed, described and explained in impeccable context.

A Whig interpretation of the history of computational complexity theory would mean a retracing of the past from the vantage point of the $P = ?NP$ question; or, more accurately, from the point of view of the *polynomial-time computability* versus NP*-completeness* point of view. From this latter point of view, the modern history of computational complexity theory will have to be told as originating in two traditions: one, the tradition of mathematical logic, particularly recursion theory and proof theory; the other, accepting a model of computation only implicitly, and often even on an *ad hoc* basis, along the special traditions, emerging from, solving special purpose problems: the travelling salesperson's problem, problems in graph theory, linear and integer linear programming problems, network flows and transportation problems, etc. It is the codification of the notion of the *efficiency of computations* that provided the impetus for the merging of the two traditions, almost by accident. The explicit recognition of the model of computation in the second tradition, and the need to go beyond degrees of undecidability in the first tradition, were brought together and fused into the one powerful and fruitful field of computational complexity theory. The four pioneers who achieved this 'unification' of the two traditions – I think, unwittingly, but serendipitously – were Edmonds, Constable, Cook, Levin and Karp, standing, of course, on the shoulders of giants of the past (and the future, even if it may sound paradoxical to be able to stand on the shoulders of giants yet unborn). The former two codified and etched the notion of polynomial-time as the standard of efficiency of computation; the latter three codified the conundrums of non-deterministic computations by defining and characterising the $NP - complete$ class, both theoretically and with imag-

[21] An elegant exposition of which can be found in [76], with a beautiful early treatment in the classic by Hartley Rogers, [58].

inative and exhaustive examples.

It will be a salutary lesson for economists, who will be disabused of optimization in the decision model tradition of computational complexity theory, to learn that *efficiency* in this computational tradition and vision simply means 'good', not the 'best'! The computer scientists and their mathematicians were mindful of that old adage: *the best is the enemy of the good*![22] Edmonds, the almost undisputed pioneer in the codification of polynomial-time computation as the criterion for efficiency introduced the notion and, indeed, the whole subject, as follows (in 1963, [18]; see also [10]):

> "For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an *efficient algorithm.* According to the dictionary, 'efficient' means 'adequate in operation or performance.' This is roughly the meaning I want – ... . *Perhaps a better word is 'good.'*
>
> There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.
>
> ....
>
> For practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite."
>
> [68], p.56; last set of italics in the original.

Three remarks on these perceptive comments by Edmonds may not be out of place. First of all, the notion of efficiency is more akin to Simon's visions of *satisficing processes* than anything from orthodox economics; secondly, by '*algebraic*' Edmonds refers, obviously, to '*polynomials*'; thirdly, it may be useful to recall Sipser's analogy between the polynmial-exponential divide and the countable-uncountable one, whereby the exponential-uncountable nexus comes about due to power-set constructions.

Finally, in line with the aims of being a 'Potted Proto-History', I shall make just one salient remark on Cook's seminal paper, which highlighted the crucial role the satisfiability problem was to play in the development of computational complexity theory. I think the consensus is that the most important point of Cook's paper was the following proposition:

$$P = NP \ iff \ SAT \subset P$$

Cook's contribution achieved a neat synthesis of the two traditions I referred to above; SAT is a quintessential problem of mathematical logic. By using the concept of *reducibility* from computability theory he was able to show that any instance of a problem in NP can be reduced to an appropriate instance of a problem in SAT. Therefore, the instance of the NP problem is solvable if,

[22] Apparently the original Voltaire quote is: 'the perfect is the enemy of the good'.

and only if, the SAT instance is solvable. Finally, he also showed that the reducibility can be achieved in polynomial time[23].

Karp,[31], who felicitously coined the suffix 'complete', built on Cook's framework and, in a sense, codified the subject of computational complexity theory to what it has since become. In particular, his characterisation of $NP-$ *complete* was decisive. Karp's results showed that almost all garden-variety combinatorial optimization problems encountered in economics are – and will forever remain – intractable, unless $P = NP$. Herein lies the significance of its inclusion in the Clay Millennium Seven.

My final comment in this context is the following: these conceptual innovations, mathematical formalizations and computational structures gave rise to exact results on feasible computations – or, efficient computations – precisely in the sense of Herbert Simon's overall research program. In particular, the last mentioned Karp result on the implications of $P \neq NP$ compels the economic theorist and the mathematical economist to resort to 'near-optimal', 'satisfactory', 'satisficing' solutions and processes. That the Simon visions have not been underpinned more formally by the results from computational complexity theory remains an unwritten intellectual chapter in behavioural economics.

3.2 An *LP* Interregnum

Linear programming, *LP*, with the variables and the coefficients defined over $\Re$, had been the classic optimization problem in economics – until recently, when it seems to have been dethroned by *DP*, *dynamic programming*, particularly in macroeconomics – that was also coupled, almost intrinsically, with an eminently practical numerical algorithm to implement it: the simplex method. A couple of intertwined landmark results on the *LP* problem can possibly be given the sobriquet, 'proto-history of computational complexity theory': the Klee-Minty result on the intractability of the simplex method, then the re-absorption of *LP* into the *theoretically tractable* fold by Khachiyan's seminal results on *LP,* where Shor's earlier work on the ellipsoid algorithm was imaginatively exploited, and finally consolidated by Karmarkar's interior point method, which was also (claimed to be) practically tractable. It is important to remember that in all three algorithms the variables are defined on $\Re$, even though the complexity criteria were developed for discrete variable or combinatorial decision problems.

The simplex algorithm for the LP problem with a constant coefficient matrix A of dimension $m \times n$ can theoretically visit the following number of vertices:

$$2\left(\frac{m}{n}\right) = \frac{n!}{m!\,(n-m!)} > \left(\frac{n}{m}\right)^m \tag{12}$$

$$\Longrightarrow \quad \geq 2^m, \ \forall n \geq 2m$$

Thus, there is the possibility of intractability – i.e., computational complexity – is intrinsically embedded within the structure of the simplex algorithm.

[23]All this in a paper that was only 7 pages in length!

It was a recognition of this intrinsic possibility and then an exploitation of it that enabled Klee and Minty to devise their (artificial) problem to show the intractability of the simplex algorithm. They showed intractability in a class of LP problems with $m = n$ equality constraints in $2n$ non-negative variables in which the simplex required $2^n - 1$ iterations; i.e., a 'visit' to every vertex of the problem. How was this achieved? Why was it attempted? The latter question can be answered by saying that *the framework of computational complexity theory had been formulated and, therefore, an analyst knew what had to be sought to show non-tractability.* As for the 'how', the following geometric approach suggests, intuitively, the way intractability was constructed.

1. Recall, first, that the simplex algorithm moves from vertex to vertex; therefore, find a polytope that has an *exponential* number of vertices. For example, a $3 - dim$ cube has $6-$faces (2×3) and $8-$vertices (i.e., 2^3). Thus, a cube with $2d$ faces has 2^d vertices.

2. Next, orient the polytope s.t., the direction of decreasing cost is 'upwards'; i.e., orient the sequence of the exponentially many vertices $\left(2^d\right)$, adjacent to one another, each higher than the previously visited vertex.

3. Then: $\forall d > 4$, $\exists$ a LP with 2^d equations, 3^d variables, and integer *coefficients with absolute values bounded by* 4, s.t., the simplex algorithm can take $2^d - 1$ iterations to find the optimum.

Not much later Khachiyan's seminal result appeared and settled decisively the polynomial time status of LP. Let me first show where exactly the need for precise numerical analysis – i.e., analysis in $\Re$ – enters the ellipsoid method[24].Locating the source of the appeal to precise arithmetic, in $\Re$, is crucial in knowing where and what kind of approximation has to be used in any particular instance of an application of the ellipsoid method[25] so that it is consistent with its underlying model of computation.

Definition 16 *An ellipsoid, $\mathbb{E}$, $Ellip\left[\mathbf{z}, D\right]$, is a set of vectors $\mathbf{x} \in \Re^n, s.t$:*

$$Ellip\left[\mathbf{z}, D\right] := \left\{\mathbf{x} | \left(\mathbf{x} - \mathbf{z}\right)^T D^{-1} \left(\mathbf{x} - \mathbf{z}\right) \leq 1\right\} \tag{13}$$

where:
D : *a positive definite matrix of order* n;
$\mathbf{z}$: *the centre of* $Ellip\left[\mathbf{z}, D\right]$;

Remark 17 *A set $\mathbb{E}$ is an ellipsoid iff $\mathbb{E}$ is an affine transformation of the unit ball,* $\left\{\mathbf{x} | \mathbf{x}^T \mathbf{x} \leq 1\right\}$.

[24] I shall follow Schrijver's characteristically clear presentation here ([67], chapter 13), but [51], chapter 8, is equally felicitous on this issue.

[25] Geometrically, the ellipsoid method is very similar to the classic method of 'binary search', eg., [51], p.171; or, at a popular level, to variations of the '20 questions' game.

The basic idea of 'binary search' consists of 'halving' a set, a domain, at each iteration of the 'search' process. This is encapsulated for the ellipsoid method in the following proposition:

Proposition 18 *Let* $\mathbf{a}$ *be a row vector of the given* $Ellip\left[\mathbf{z}, D\right]$*, and let* $\mathbb{E}' \supset Ellip\left[\mathbf{z}, D\right] \cap \{\mathbf{x}|\mathbf{ax} \leq \mathbf{az}\}$*, s.t.* $\mathbb{E}'$ *has smallest volume. Then* $\mathbb{E}'$ *is unique and* $\mathbb{E}' = Ellip\left[\mathbf{z}', D'\right]$, *where:*

$$\mathbf{z}' := \mathbf{z} - \frac{1}{n+1} \cdot \frac{D\mathbf{a}^T}{\sqrt{\mathbf{a}D\mathbf{a}^T}} \tag{14}$$

$$D' := \frac{n^2}{n^2-1}\left(D - \frac{2}{n+1} \cdot \frac{D\mathbf{a}^T\mathbf{a}D}{\mathbf{a}D\mathbf{a}^T}\right) \tag{15}$$

and:

$$\frac{vol\ \mathbb{E}'}{vol\ \mathbb{E}} < e^{-\frac{1}{2}\frac{1}{(n+1)}} \tag{16}$$

The point to note is the square root term in (14). No underlying model of computation which is not capable of precise real number computation will be able to implement the ellipsoid algorithm consistently. There are three alternatives to circumvent the problem of precision:

- Appeal to, and utilize, relevant approximation results, as is done in, for example, [67], pp.166-7;
- Build an underlying model of exact real number computation, as claimed to have been done in [4];
- Or, finally, resort to careful and informed *ad hockery*, as is expertly resorted to in [51], §8.7.4;

There is a problem with the first of the alternatives. The second of the two approximation lemmas on which [67] relies on, to circumvent the problem of numerical imprecision, entailed by the inconsistency between the ellipsoid method invoking real number domains and the implicit model of computation relying on rational or integer number domains, is based on a non-constructive argument at a crucial point in the proof (op.cit, bottom, p.167)[26]. A similar infelicity plagues the method of 'informed *ad hockery*'; for example, in [51], p.182, when a claim holding '$\forall x \in \Re$, $|x - \hat{x}| / |x| \leq 2^P$, for a given precision P.' As for relying on an underlying real number model of computation, in a sense the whole research program represented by [4] could be said to have emanated from the conundrums posed by the *LP* problem. My own reservations about this method is given below.

An outline of the Khachiyan solution is as follows. Consider the problem of finding a feasible point satisfying the following system of inequalities:

[26] Moreover, it is not clear to me that the rounding *process* invoked in the first theorem, op.cit, p.166, is itself in P.

$$\langle \mathbf{h}_i, \mathbf{x} \rangle \leq g_i, \; i = 1, ..., p \tag{17}$$

$\mathbf{h}_i \in E^n$, $p \geq 2$, $n \geq 2$.

Khachiyan's algorithm finds a feasible point, or establishes its nonexistence, in a finite number of iterations, polynomial in its input variables.

Denote by L : the length of the binary encoding of the input data; then:

$$L = \sum_{i,j=1}^{n,p} \log_2 (|h_{ij}| = 1) + \sum_{j=1}^{p} \log_2 (|g_j| + 1) + \log_2 np + 2 \tag{18}$$

where: h_{ij} is the jth element of vector $\mathbf{h}_i$. Then, Khachiyan's algorithm is, schematically:

Step 1 : Set $x_0 = 1, H_0 = 2^{2L} I, k = 0$.

Step 2 : Terminate the algorithm with $\mathbf{x}_k$ as the feasible point, If $\mathbf{x}_k$ satisfies:

$$\langle \mathbf{h}_i \mathbf{x}_k \rangle < g_i + 2^{-L}, \forall i = 1, ..., p \tag{19}$$

Step 3 : If $k < 4\left(n+1)^2 L\right)$, then go to *Step* 4; otherwise, terminate the algorithm with the negative, tractable, answer that *no solution exists*.

Step 4 : Choose any inequality for which:

$$\langle \mathbf{h}_i \mathbf{x}_k \rangle \geq g_i + 2^{-L} \tag{20}$$

Now, set:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{(n+1)} \frac{H_k \mathbf{h}_i}{\langle \mathbf{h}_i, H_k \mathbf{h}_i \rangle^{1/2}} \tag{21}$$

and:

$$H_{k+1} = \frac{n^2}{n^2 - 1} \left[H_k - \frac{2}{n+1} \frac{H_k \mathbf{h}_i \mathbf{h}_i^T H_k}{\mathbf{x}_p \mathbf{h}_i, H_k \mathbf{h}_i} \right] \tag{22}$$

Step 5 : Set $k = k + 1$ and go to *Step* 2.

Remark 19 *The Khachiyan algorithm returns a feasible solution or establishes its non-existence in* $4\left(n+1\right)^2 L$ *iterations.*

Remark 20 *With minor modifications the algorithm can be adapted to handle the system above in the presence of an additional system of linear equalities:* $N^T \mathbf{x} = \mathbf{b}$.

Remark 21 *It is clear that the 'villain of the piece' appears in the denominator of the iteration linking* $\mathbf{x}_{k+1}$ *to* $\mathbf{x}_k$ *in equation (21).*

Next, consider the iteration in (21) – (22) as a dynamical system - in this case a map. Suppose using one of the two approximation schemes referred to above the initial conditions are guaranteed to be rational numbers; i.e., computable numbers. Thus, the sequence of numbers generated by the iteration in (21) – (22), can be guaranteed to be computable, i.e., a recursive real number. Does this guarantee that the limit is a recursive real number? I think not and my conjecture is that a proof is relatively easy. I shall leave it as an unproved conjecture for the moment.

Finally, anyone who examines in detail the complete structure of the proof can be certain to find that it is replete with non-constructivites - i.e., appealing to *undecidable disjunctions*. Moreover, the detailed proof given in [67] appeals, at a crucial point, to the proof of Farkas's Lemma; that, in turn, appeals to a theorem which is proved using an algorithm equivalent to the simplex method, whose intractability was our starting point in this saga on the *Interregnum*! In any case, it is best to underpin these kinds of problems on a model of 'real computation', as in [4] (see, in particular, Chapter 6).Fred Richman perceptively observed, years ago, [57], p.125 (italics added):

> "Even those who like algorithms have remarkably little appreciation of the thoroughgoing algorithmic thinking that is required for a constructive proof. ...... I would guess that most realist mathematicians are unable even to recognize when a proof is constructive in the intuitionist's sense.
>
> It is a lot harder than one might think to recognize when a theorem depends on a nonconstructive argument. One reason is that *proofs are rarely self-contained, but depend on other theorems whose proofs depend on still other theorems. These other theorems have often been internalized to such an extent that we are not aware whether or not nonconstructive arguments have been used, or must be used, in their proofs*. Another reason is that the law of excluded middle [LEM] is so ingrained in our thinking that we do not distinguish between different formulations of a theorem that are trivially equivalent given LEM, although one formulation may have a constructive proof and the other not."

What is the meaning – and the point – of proving polynomial time computability of a method which relies on non-constructive proofs, invoking undecidable disjunctions?

4 The Problem of Computational Complexity of Real Numbers and Real Functions

> "The 'computable' numbers may be described briefly as *the real numbers whose expressions as a decimal are calculable by finite means.* Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or *a real* or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account

of the relations of the computable numbers, functions, and so forth to one another. This will include a development of t*he theory of functions of a real variable* expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine."

Alan Turing, [83], p.230; italics added

These fascinating and prescient opening sentences in Turing's classic paper indicate, clearly and unambiguously, that the model of computation that he developed – now felicitously and justly called the *Turing Machine* model of computation – was intimately linked with, and underpinned by, *real numbers* and *real functions*. These links were explored, developed and expanded, in the immediate post-war years, even while orthodox computational complexity theory was in its infancy, in a series of seminal books and papers by Goodstein, Grzegorczyk, Lacombe, Mazur and Specker[27]. Their modern successors are Pour-El and Richards, Friedman and Ker-I-Ko, Weirauch, Aberth[28] and Shepherdson[29]. Both prior to and parallel with these developments, in what is now broadly referred to as recursive or computable analysis, there was the rich vein of algorithmic analysis emanating from the Brouwer-Heyting tradition of intuitionistic constructive analysis; its definitive codification – at least as a constructive, even if not intuitionistic, variant – was achieved with Errett Bishop's classic work of 1967 on the *Foundations of Constructive Analysis* ([3]). There have, thus, been available a variety of supremely elegant, comprehensive and rigorous foundations for models of real number and real function computations for the whole of the period since the Turing Model has been in existence. Computer science – both theoretically and in its applied versions, and even practically – has developed, *pari passu*, with these theories as its foundation. Yet, numerical analysis and numerical analysts have gone about their traditional and noble business, a tradition that can be traced back at least to Newton and Gauss, without paying any attention to these developments. This

[27]The classic references are: [26] [27], [41], [46], and [79]. Grzegorczyk was continuing the Polish tradition, initiated by Banach and Mazur, that had commenced almost simultaneously with Turing's original contribution. In their Forward to the unfortunately relatively little acknowledged classic by Mazur, [46], Rasiowa and Grzegorczyk made the 'melancholy, observation that (italics added):

> "[Mazur's booklet] is based on the lectures [he gave on] 'Computable Analysis' in the academic year 1949-1950 at the Institute of Mathematics of the Polish Academy of Sciences in Warsaw. These lectures present a systematic exposition of the results obtained by S. Banach and S. Mazur in 1936-1939 and contain a detailed examination of recursive *real numbers*, recursive sequences, recursive functionals and recursive *real functions*. The authors never published their results.... because the Second World War made this impossible."

[28]Aberth's work is a development of the Russian Constructivists, but has become a part of the modern theory of real computation based on the Turing Model of Computation. Russian constructivism is clearly and concisely outlined in [7], chapter 3; copious references to the pioneering Russian contributions, almost simultaneous with the above Polish, French and American pioneering works are given in [1].

[29]Here the representative references are: [55], [38], [37], [90] and [66].

remains a paradox, especially since the ubiquity of the digital computer has meant that the numerical analyst's theories have to be implemented on these machines. Numerical analysts have always been concerned with the efficiency of algorithms, for example in terms of approximation criteria for convergence of numerical methods, but the connection between their concerns with efficiency and those of the computer scientist's with computational complexity theory has never had more than a tangential connection with each other. These issues have been raised to a new level of awareness in the work around the *BCSS model* and its origins and development. However, they – the originators of the *BCSS model*, Smale in particular – seem not to have paid sufficient attention to the algorithmic tradition in real analysis; not even to the clearly related tradition of analog computation[30]. Space constraints prevent me from a proper discussion and evaluation of the technical merits and conceptual priorities in the two approaches – the recursive analysis and the *BCSS model*.

I shall, instead, simply outline – in the next subsection – some salient features of the BCSS model and the way orthodox complexity theoretic concepts are defined in it. However, I remain sceptical of the need for considering real number and real function computation from the perspective of the BCSS model; some of my reasons are also sketched out.

4.1 Complexity Theory of Real Computation: Notes on the *BCSS Model*

"In the late 17th century, when Newton and Leibniz were first inventing calculus, part of their theory involved *infinitesimals* – i.e., numbers that are infinitesimal but nonzero. .... .. Indeed, Leibniz and Newton had infinitesimals in mind when they invented calculus; ... ."

Schechter, [64], pp.394-6; italics in the original.

Yet the whole rationale and motivation for the development of the *BCSS model* of real computation seems to be based on ignoring this historical fact! Smale, in particular, has tirelessly propagated the view that the Newtonian 'vision' of the Universe is underpinned by a 'continuous real number model'. I shall return to these issues in the concluding section. For now, let me simply outline just the bare bones *BCSS model* to enable me to define the classes P and NP for real number and real function computations in this framework. Anyone interested in delving into the deeper reaches of the BCSS model can easily do so via the extremely readable and clearly presented monograph by Blum, et.al., ([4]).

Contrary, also, to Turing's original approach, whereby the real numbers are expressed as decimal expansions, in the *BCSS model* they are taken as abstractly and axiomatically given entities:

[30]I have tried to 'resurrect' a consideration of this tradition, again from the point of view of a computable economist, in [88]. I shall have a little to say about this approach below.

> "[C]ontiinuing from a very different point of view, I will devise a notion of computation taking the real numbers as something axiomatically given. So, a real number to me is something *not* given by a decimal expansion; a real number is just like a real number that we use in geometry or analysis, something which is given by its properties and not by its decimal expansion."
>
> [73], p.62; italics added.

One would wonder, then, how properties replete with undecidable disjunctions could be implemented on any kind of computer, if also the axioms are not recursively presented. Moreover, if the starting point is an axiomatic representation of real numbers, why stop there? Why not taken an axiomatic approach to the $P = ?NP$ problem? I shall return to this particular theme in the concluding section.

The *BCSS model* is presented, at least in [4], as a 'mathematical foundation of the laws of computation of numerical analysis' (and the Turing Model is generated as a special case). But the 'main goal' of [4] is 'to extend the [the classical theory of computational complexity] to the real and complex numbers' in a special context:

> "A cornerstone of classical complexity theory is the theory of NP-completeness and the *fundamental* P≠NP? problem.
>
> A main goal of [[4]] is to extend this theory to the real and complex numbers and, in particular, to pose and investigate the fundamental problem within a broader mathematical framework."
>
> [4], p.24

With this aim in focus, the following general framework of computations over a *Ring* (or *Field*) – in contrast to the starting point in natural numbers, according to BCSS, of the traditional[31] 'Turing theory' – is set up.

Definition 22 *Machines over $\mathbb{R}$ (say a commutative ring with unit).*

A machine $\mathbb{M}$ over $\mathbb{R}$ is a finite connected directed graph, containing the following five types of nodes:

1. Input, with underlying input space $\mathbb{I}_{\mathbb{M}}$;
2. Computation, in state space, $\mathbb{S}_{\mathbb{M}}$;
3. Branch[32];
3. Output, with underlying output space, $\mathbb{O}_{\mathbb{M}}$;
4. Shift nodes, $\sum_{\mathbb{M}}$;

Where the input space, $\mathbb{I}_{\mathbb{M}}$, and output space, $\mathbb{O}_{\mathbb{M}}$ are defined in $\mathbb{R}^{\infty}$, which is the disjoint union:

$$\mathbb{R}^{\infty} = \sqcup_{n \geq 0} \Re^n$$

[31] My fidelity to *orthodox* computational complexity theory emanates from my adherence to *orthodoxy* in computability theory!

[32] Topological complexity is defined at this node.

and the state space, $\mathbb{S}_M$, is defined in $\mathbb{R}_\infty$, the bi-infinite direct sum space over $\mathbb{R}$, s.t., typical elements of it, $x \in \mathbb{R}_\infty$, are of the form:

$$x = (......x_{-2}, x_{-1}, x_0 \circ x_1, x_2, ...)$$

and:

$x_i \in \mathbb{R}, \forall i \in \mathbb{Z}$ *(and each such x_i is refereed to as a coordinate of x)* & $x_k = 0$ *for $|k|$ sufficiently large;*

$\circ$: *is a distinguishing marker between x_0 and x_1;*

Remark 23 *The space $\mathbb{R}_\infty$ has natural left and right shift operations:*

$$\sigma_l (x)_i = x_{i+1} \quad \& \quad \sigma_r (x)_i = x_{i-1}$$

Remark 24 *Anyone familiar with the definition of the Turing Machine, say as given in the classic book by Martin Davis, [17] (see also the appendix in [87]), will find it easy to interpret the above definitions in their specific, more general, domains. The mechanism is exactly identical.*

Two 'reminders', particularly to economists, are apposite at this point: the first is to keep in mind that in computational complexity theory of any hue, one is dealing with *decision problems; the second is that* in this field the primary interest is in decidable problems, i.e., problems for which their characteristic functions are computable. The complexity of the former is usually measured by the complexity of the latter.

Definition 25 *A decision problem over $\mathbb{R}$ is a set $\mathbb{S} \subset \mathbb{R}^\infty$*

Definition 26 $\forall x \in \mathbb{R}^\infty$, *its characteristic function is defined as:*

$$\chi_{\mathbb{S}} = \{ \begin{array}{l} 1 \textit{ if } x \in \mathbb{S} \\ 0 \textit{ otherwise} \end{array}$$

Definition 27 *Polynomial functions over $\mathbb{R}$*

Suppose $h : \mathbb{R}^m \to \mathbb{R}$ is a polynomial function of degree d over $\mathbb{R}$.

Then h defines a polynomial function, $\hat{h}$, on $\mathbb{R}_\infty$, of dimension m and degree d :

$$\hat{h} = \mathbb{R}_\infty \to \mathbb{R}$$

where: $\forall x \in \mathbb{R}_\infty, \hat{h} = h (x_1, x_2, ..., x_m)$

Definition 28 *Polynomial time*

$\mathbb{M}$ *over $\mathbb{R}$ works in polynomial time if, $\forall x \in \Re^n \subset \Re^\infty$, and for some fixed $c, q \geq 1$:*

$$cost\ (x) \leq cn^q$$

where, cost (x) : the number of nodes traversed during the computation.

Corresponding to a measure of *bit* over $\mathbb{Z}_2$, there is 'height', $ht_{\mathbb{R}} (x) = 1, \forall x \in \mathbb{R}$.

Definition 29 *$\forall x \in \Re^n \subset \Re^\infty$, length $(x) = n$ & size $(x) = n \cdot ht_{\mathbb{R}}(x)$*
where
$ht_{\mathbb{R}}(x) : \mathbb{R} \to \mathbb{Z}_{0 \cup +}$ & $ht_{\mathbb{R}}(x) = \max ht_{\mathbb{R}}(x_i)$

Definition 30 *$\forall x \in \Re^n \subset \Re^\infty$:*

$$cost_{\mathbb{M}}(x) = T(x) \times ht_{\max}(x)$$

where,
$T(x)$: the halting time of $\mathbb{M}$ on input x.

Definition 31 *Polynomial time Machine, $\mathbb{M}$ over $\mathbb{R}$*
$\mathbb{M}$ over $\mathbb{R}$ is a polynomial time machine on $X \subset \mathbb{R}^\infty$ if $\exists\ c, q \in \mathbb{Z}_+$ s.t:

$$cost_{\mathbb{M}}(x) \leq c\,(size\,(x))^q\,,\ \forall x \in X$$

Definition 32 *$\varphi : X \to Y \subset \mathbb{R}^\infty$ is said to be polynomial time computable – or a $p -$ morphism – over $\mathbb{R}$, if φ is computable by a polynomial time machine on X.*

Finally, for the main relevant definitions, i.e., of the class P over a ring, $\mathbb{R}$:

Definition 33 *The class P*
A decision problem $S \subseteq \mathbb{R}^\infty$ is in class P over $\mathbb{R}$ – i.e., $S \in P_{\mathbb{R}}$ – if χ_S is a $p -$ morphism over $\mathbb{R}$.

Before we proceed to the NP world, the notion or *reducibility* has to be defined.

Definition 34 *$p -$ reduction $p -$ reducible*
A morphism φ is a $p -$ reduction, if :

$$\varphi : \mathbb{R}^\infty \to \mathbb{R}^\infty,\ s.t.,\ \varphi(S) \subseteq S' \quad \& \quad \varphi(\mathbb{R}^\infty - S) \subseteq \mathbb{R}^\infty - S'$$

where:
The decision problem S is, then, said to be $p -$ reducible to the decision problem S'

Definition 35 *The class NP*
A decision problem $S \subseteq \mathbb{R}^\infty$ is in class NP over $\mathbb{R}$ – i.e., $S \in NP_{\mathbb{R}}$ – if $\exists$ $\mathbb{M}$ over R, with (input space), $\mathbb{I}_{\mathbb{M}} = \mathbb{R}^\infty \times \mathbb{R}^\infty$, & $c, q \in \mathbb{Z}_+, s.t$:

1. *if $x \in S$ then $\exists w \in \mathbb{R}^\infty$, s.t., $\Phi_{\mathbb{M}}(x, w) = 1$ & $cost_{\mathbb{M}}(x, w) \leq c\,(size\,(x))^q$;*
2. *if $x \notin S$ then $\nexists w \in \mathbb{R}^\infty$ s.t., $\Phi_{\mathbb{M}}(x, w) = 1$*

The obvious question about *TSP* is immediately answered:

Proposition 36 *TSP is in class $NP_{\mathbb{R}}$*

Definition 37 *$NP - hard$*

A decision problem $\hat{S}$ is $NP_{\mathbb{R}} - hard$ if every $S \in NP_{\mathbb{R}}$ is $p - reducible$ to it.

Finally, we have the important definition of $NP - Completeness$.

Definition 38 *$NP - Completeness$*

A decision problem $\hat{S}$ is said to be $NP - Complete$ over $\mathbb{R}$ if it is $NP_{\mathbb{R}} - hard$ & $\hat{S} \in NP_{\mathbb{R}}$

Remark 39 *By considering the 'special' case of $\mathbb{Z}_2$. the orthodox domain of computational complexity theory based on the Turing Model of computation over binary sequences, it is immediate that Cook's theorem holds.*

Remark 40 *The earlier LP feasibility problem, when specialised over $\mathbb{Q}$ is easily shown, in the above framework, to be in class P*

4.2 Sceptical Notes on the *BCSS Model*

> "What is not so clear is that *continuous processes* (*with differential equations*) may also be regarded as trial and error methods of solution to static equations. The reason why it is not so easy to see is that no human being can make continuous trials infinite in number. This gap in our grasp of the problem has been closed by the perfection of electronic and electro-mechanical computers - sometimes called zeroing servos - which continuously 'feed back' their errors as a basis for new trials until the error has disappeared. *Such a machine is an exact analogue of a continuous dynamic process.* Therefore it seems entirely permissible to regard the motion of an economy as a process of computing answers to the problems posed to it."
>
> [25], pp.1-2; italics added.

What Goodwin is suggesting here is that the economic system can felicitously interpreted using the metaphor and model of the analogue computer - a metaphor that had, with equal finesse, been invoked by Walras, Pareto and Hayek. My sceptical notes on the BCSS model revolve around three perplexities:

(a). What is wrong with the analogue model of computation over the reals and why it was not invoked to provide the mathematical and physical foundation for numerical analysis by the authors of the BCSS model?

(b). What is wrong with the computable and recursive analytic model, with its rich complexity theoretic analysis of classic optimization operators routinely used in economic theory (optimal control, dynamic programming, etc.,), of the perennial paradoxes of the initial value problem on ordinary differential equations and their solution complexities and of much else in a similar vein.

(c). I don't think there is any historical or analytical substance to the Newtonian vision frequently invoked as a backdrop against which to justify the need for a new mathematical foundation for numerical analysis.

(4). Finally, there is an important strand of research that has begun to interpret numerical algorithms as dynamical systems; from this kind of interpretation to a study of undecidability and incompleteness of numerical algorithms is an easy and fascinating frontier research topic within the framework of computable analysis, which owes nothing to - and has no need for - the BCSS kind of modelling framework – even though there are claims to the contrary in [4] regarding such issues[33].

The last named of the perplexities, (d), has, at present, not considered the problem of computational complexity of numerical algorithms via an analysis of equivalent dynamical systems; but, surely, it is only a question of time before that step is taken. There is a cryptic acknowledgement to an important strand in this new tradition, pioneered by da Costa and Doria, in [4], p.35:

> "An undecidability result, related to chaotic dynamics ..., but very different in methodology and spirit is in [da Costa and Doria][34]."

4.2.1 Real Computation with Analogue Computers

One of the recurrent themes in the BCSS modelling approach is the need for a model of computation over the reals so that classic mathematical physics – or applied mathematical – problems like those posed by the need to solve, numerically, ordinary differential equations, defined over $\mathbb{R}$.Consider the following reasonably 'complex' dynamical system, the so-called Rössler System:

$$\frac{dx}{dt} = -\left(y+z\right)$$

$$\frac{dy}{dt} = x + 0.2y$$

$$\frac{dz}{dt} = 0.2 + z\left(x - 5.7\right)$$

Suppose a *General Purpose Analogue Computer* (*GPAS*) is defined in terms of the usual *adders*, *multipliers* and *integrators* as the elementary units, similar to the elementary operations for $\mathbb{M}$ defined above for the *BCSS model* which, in turn, are analogous to the elementary operations defining a Turing Machine or a partial recursive function ($\mu-$recursion, minimalization, etc.,).Then it can be shown that a *GPAS* consisting of 3 adders, 8 multipliers and 3 integrators can simulate the above Rössler System. What kind of functions can be constructed by combining such elementary units to build a *GPAS* to simulate continuous

[33] Primarily in relation to the decidabilty problems of the Mandelbrot and Julia sets, as posed by Penrose, [53].

[34] That in a book published in 1998, the authors of [4] do not even manage to cite the published version of one of the early pioneering results of da Costa and Doria, from 1991 – the early examples are in, [14] and [15] – let alone their later work and the related work of Chris Moore, for example [47], and several others, is, I guess, an indication of the lack of attention paid to these other 'methodologies and spirit'! But these pioneering works are on undecidability of dynamical systems, not the computational complexity of dynamical systems. perhaps that is the reason why not much attention was paid to these works in [4].

functions, defined over $\Re$? A series of results, beginning with Shannon in 1941, [65] and culminating in Rubel's fundamental results on GPAS in 1988, [63], give precise answers to this question. I have summarized the GPAS framework and results pertaining to them in [88]. The main theorem is the following:

Theorem 41 *(Rubel's Theorem): There exists a nontrivial fourth-order, universal, algebraic differential equation*[35] *of the form:*

$$P(y', y'', y''', y'''') = 0 \tag{23}$$

where P is a homogeneous polynomial in four variables with integer coefficients.

The exact meaning of '*universal*' is the following:

Definition 42 *A universal algebraic differential equation P is such that any continuous function $\varphi(x)$ can be approximated to any degree of accuracy by a C^{∞} solution, $y(x)$, of P.In other words:*

$$If\ \varepsilon(x) \text{ is any positive continuous function, } \exists y(x)\ s.t\ \mid y(x) - \varphi(x) \mid < \epsilon(x),\ \forall x \in (-\infty, \infty) \tag{24}$$

There is, then, a ready-made framework in which to study the real functions traditionally used in modelling physical (and, dare I say, economic processors). Why not develop a computational complexity theory for *GPAS* rather than replicate the Turing Model for $\Re$?

4.2.2 Real Computation within the Turing Model of Computation

I have described Chris Moore's construction of a smooth map simulating a Minsky program machine, i.e., a minimal Turing Machine, in [87], pp.47-8. This is an exercise going in the reverse direction to the one pursued in BCSS modelling. But it is directly relevant here, because the computational complexity results for the Turing Model will carry over to the constructed, equivalent, smooth map. In an e-mail message to me, of 20 January, 1995, Chris Moore wrote as follows:

> "I consider the systems [I have constructed] highly contrived, so I'm not sure how relevant [they] really [are] to economics, but enjoy."

[35] An algebraic differential polynomial is an expression of the form :

$$\sum_{i=1}^{n} a_i x^{r_i} y^{q_{0i}} (y')^{q_{1i}} \dots \left(y^{(k_i)}\right)^{q_{k_i i}}$$

where a_i *is a* real *number*, $r_i, q_{0i}, \dots\dots, q_{k_i i}$ *are non* negative integer*s and* y *is a function of* x.

Algebraic differential equations (ADEs) are ODEs of the form:

$$P\left(x, y, y', y'', \dots\dots, y^{(n)}\right) = 0$$

where P is an *algebraic differential polynomial* not identically equal to zero.

My feeling is that the *BCSS model* is also 'highly contrived' and its relevance to economics very dubious[36].

On the more concrete problem of the computational complexity of ODEs, there have been research and impressive results within one or another recursive analysis model for at least 40 years. For example Cleave, [?], p.447, working within Goodstein's recursive analysis, showed, already in 1969, how 'to estimate the computational complexity of the solution of an ordinary differential equation in terms of its position in the Grzegorczyck hierarchy of primitive recursive functions.' The extra finesse here, from a practical, computing point of view, is not only its basis in the Turing model of computation; but also based on first constructivising the classical non-constructive Cauchy-Lipschitz existence theorem. In my own work I have worked with ODEs and a constructivised version of the Peano existence theorem to replicate the kind of analysis and results achieved by Cleave and others, ever since Henrici proved, [?], that some of the classical theorems in ODEs could be constructivised. Even more pertinent to the claims in [4], on building mathematical foundations for numerical methods, there is Miller's work, [?] aimed precisely at providing a recursive function theoretic analysis of numerical methods[37].

[36] Thoughtless appeals to the BCSS model is not unknown in an economic text purporting to provide a framework and an analysis of 'complexity' in *real* economic models underpinned by a model of computation:

> "Computing with real numbers offers some important advantages in the context of scientific computing. ... it is also relevant to applications in economic theory. *Economic models typically use real variables and functions of them.* A model of computing in which the elementary operations are functions of real variables allows the model to be directly applied to standard economic models, without requiring an analysis of approximations in each application."
>
> [49], pp. 1-2; italics added.

[37] There is a '*grudging nod*' to the long and rich tradition of considering computability and complexity of models defined over $\Re$, in a variety of recursive and computable analyses, by Smale, in [73], p.61:

> "Indeed, some work has *now been done* to adapt the Turing machine framework to deal with real numbers........ Thus, the foundations are probably being laid for a theory of computation over the real numbers."

This acknowledgement comes in 1991 after almost continuous (sic!) work by recursive and computable analysts to adapt the Turing model to domains over $\Re$. Weihrauch, himself a notable contributor to this adapting tradition refutes what I can only call a 'preposterous' claim in [4], p.23:

> "A major obstacle to reconciling scientific computation and computer science is the present view of the machine, that is the digital computer. As long as the computer is seen simply as a finite or discrete object, it will be difficult to systematize numerical analysis. We believe that the Turing machine as a foundation for real number algorithms can only obscure concepts."

I can only endorse, wholeheartedly, Weihrauch's entirely justifiable claim that the 'theory presented in his book, [90], p.268, 'refutes their [i.e., the above] statement.'

4.2.3 The Contrived Newtonian Vision

"Isaac Newton made a continuous mathematical model of a discrete universe using differential equations to explain how things in that universe move. ....

Real numbers are crucial to Newton's universe of a continuous universe. Real numbers give the picture of a continuous set of numbers between zero and one.

[72], pp.90-1.

Clearly, the authors of [4] have not done their homework on Newton (and Leibnitz). That these two pioneers of the differential and integral calculus struggled to found their mathematics on the non-standard numbers, rather than the conventional real number continuum, is part of the folklore of the history of analysis – at least since Veronese, Skolem and, more recently, Abraham Robinson and Edward Nelson. Quite apart from the historically unsubstantiable assertion that the Newton's calculus was based on what is now called real analysis – and, thus, the real number continuum – rather than non-standard analysis, there is the even more contentious suggestion, in the general program enunciated and propagated via the *BCSS model*, that continuous time modelling of physical processes in the natural sciences is, somehow, more 'realistic' or accurate. I can do no better than to remind these authors of one of Komogorov's perceptive observations in this regard:

"Until recently, in the mathematical treatment of natural science ..., the prevailing way of modelling real phenomena was by means of mathematical models constructed on the mathematics of the infinite and the continuous. For example, in studying the process of molecular heat conductivity, we imagine a continuous medium in which the temperature is subject to the equation

$$\frac{\partial u}{\partial t} = K\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{25}$$

Mathematicians usually regard the corresponding difference scheme

$$\Delta_t u = K\left(\Delta_{xx} u + \Delta_{yy} u + \Delta_{zz} u\right) \tag{26}$$

Only as arising out of the approximate solution of the 'exact' equation [25]. *But the real process of heat conduction is no more similar to its continuous model expressed by [25] that to the discrete model directly expressed by [26].*"

[40], p.30; italics added.

Not very recently, Richard Feynman ([20], p.467) wondered:

"Can physics be simulated by a universal computer?

Feynman, in his characteristically penetrating way, then asked three obviously pertinent questions to make the above query meaningful:

- What kind of physics are we going to *imitate*?
- What kind of *simulation* do we mean?
- Is there a way of simulating rather than imitating physics?

Before providing fundamental, but tentative, answers to the above queries, he adds a penetrating caveat (ibid, p.468; italics in original):

> "I want to talk about the possibility that there is to be an *exact* simulation, that the computer will do *exactly* the same as nature."

Feynman's answer to part of the first question was that the kind of physics we should simulate are 'quantum mechanical phenomena', because (ibid, p. 486):

> "...I'm not happy with all the analyses that go with just the classical theory, because nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy."

But he was careful to point out, also, that there was a crucial mathematical difference between 'quantizing' and 'discretizing' (ibid, p. 488; italics added):

> "*Discretizing is the right word*. Quantizing is a different kind of mathematics. If we talk about *discretizing* ... of course I pointed out that we're going to *have to change the laws of physics*. Because the laws of physics as written now have, in the classical limit, a continuous variable everywhere ... ."

He was not the only giant in the natural sciences who wondered thus: Einstein, Schrödinger, Hamming, Toffoli, Fredkin and most recently, Penrose, too, have had speculative thoughts along similar lines. Einstein, in perhaps his last published work, seems to suggest that a future physics may well be in terms of the discrete:

> "One can give good reasons why reality cannot be represented as a continuous field. ...."
>
> [19], p.166

Roger Penrose, in his recently published, massive, vision of *The Road to Reality*, was even more explicit:

[W]e may still ask *whether the real-number system is really 'correct' for the description of physical reality at its deepest level.* When quantum mechanical ideas were beginning to be introduced early in the 20th century, there was the feeling that perhaps we were now beginning to witness *a discrete or granular nature to the physical world* at its smallest scales.... Accordingly, various physicists attempted to build up an alternative picture of the world in which discrete processes governed all action at the tiniest levels. ....

In the late 1950s, I myself tried this sort of thing, coming up with a scheme that I referred to as the theory of 'spin networks', in which the discrete nature of quantum-mechanical *spin* is taken as the fundamental building block for a *combinatorial* (i.e. a discrete rather than real-number-based) approach to physics."

[53], pp.61-2; italics in the second paragraph as in original.

These speculations on the granular structure of 'reality' at some deep level arose out of purely theoretical developments in the subject, but in continuous interaction with the epistemology of measurement in well-designed and sound experimental environments. Where these reflections by Feynman, Einstein and Penrose leave the loose epistemology and wild methodological claims in [4], I am not at sure.

5

6 Lessons from the Past; Visions for the Future

[Takeuti says]: *I am now very much interested in proving $P < NP$.*

[Kreisel says]: *I am interested in the subject only if $P = NP$*

pace Gaisi Takeuti (Kreisel and I in: Kreiseliana: About and Around Georg Kreisel, edited by Piergiorgio Odifreddi, A.K.Peters, 1996)

To this I may add[38]:

[da Costa and Doria say]: $S + [P = NP]$ is a consistent theory

where, S : A 'reasonably strong consistent axiomatic system that includes arithmetic and has a recursively enumerable set of theorems.'

Where does all this leave a computable economist and his perspectives on computational complexity theory. da Costa and Doria summarise metamathematical approaches to the $P =?NP$ question, even suggesting that S could well be ZFC. I, as a computable economist, shudder to think how a non-computable

[38] I am relying on [16] for these 'exotic' remarks

economist, non-competent in metamathematics, invoking this kind of exotic result to claim that TSP is tractable, in theory – as they do with equilibria and efficiency postulates, with no hope ever of deciding or computing any such thing.

Any analysis of the computational complexity of any theory requires that the investigated theory should be underpinned by a model of computation. As it is in economic theory today, not a single respectable formalization – whether orthodox or not – is underpinned by a formal or informal model of computation. *Ad hoc*, *ex post*, approximations of uncomputable, non-constructive economic theories, indiscriminate approximations using expansions and contractions without any understanding of the computable structure of a model is routinely resorted to, and far reaching policy conclusions are inferred. Non-constructive contraction mappings, uncomputable local equilibria, undecidable efficiency postulates – these are the staple features of analytical economics. The advent of the digital computer, and its ubiquity, has lent a veneer of respectability to computation and, hence, there seems to be licence to talk about the computational – and other - complexities of such essentially non-computable models.

Is there a way out of this morass and confusion?

My own view, from the point of view of computable economics, is uncompromising: one has to formulate theories with a model of computation underpinning them. There is no point in proving the existence of a solution to the IVP of an ODE, without also knowing how to constructivise[39] the solution so that one can investigate how easy or hard it is to determine it.

The lessons from the past are depressing. I have, myself, been bamboozled by the dazzling beauty a mathematical format that appears to be meaningfully numerical. This happened to me in the particular case of the ellipsoid algorithm. When I was taught it, and then came to apply it, in particular, in policy oriented contexts, I was not sufficiently alert to the fact that there was serious question of consistent approximation involved (see the discussion above, in the section on *LP*. I do not know of any advanced textbook in mathematical economics that deals with the theory of exact approximation. Neither is there any meaningful teaching of constructive mathematics, even of a rudimentary kind. Thus, no able young graduate student is ever trained to think algorithmically; instead, the substitute is indiscriminate computation and blind approximations.

The way I have structured the paper, and the idiosyncratic – almost psychedelic – way the discussion proceeded, was with the intention of providing a series of vignettes of, and brief glimpses into, a world of quantitative adventures where approximations, computations and constructions rule absolutely. Moreover, the opening quote from the letter Simon wrote to me, was given the prominence it deserves, because my conviction is that the only kind of economic theory amenable to a consistent analysis in terms of computational complexity

[39]I have used 'constructivise' very often in this paper. I think it is safe to assume that I really mean 'algorithmise' whether with a Turing model of computation as the backdrop or not. Constructive mathematics refuses to abide by the Church-Turing thesis, but all its propositions are algorithmic. I have, very gradually, come round to the view that the constructivist's more general vision, which subsumes the recursion theorist's algorithmic world, is superior from many points of view.

theory is the Simon version of behavioural economics. In particular, satisficing is a natural concept within the framework of decision problems (in the sense in which that phrase is defined in metamathematics, recursion theory and model theory); bounded rationality, too, is best understood in a theory that is underpinned by a model of computation, as it was the case with any thing Simon modelled.

The new behavioural economics is devoid of an underlying model of computation; so are game theory and experimental economics. That the latter is not underpinned by a model of computation is, for me, a sad indictment of the puzzling hold that orthodox ways of thinking about mathematics can strangle the richness of a fresh subject. It is also puzzling that the subject that defines itself as that which 'studies efficient allocation of scarce resources', and that which has made much of having brought into this fold the study of information and its husbanding, has not done so with respect to computation, computational resources and computational structures.

In his thoughtful 'Foreward' to Chaitin's *magnum opus*, Jacob Schwartz made the perceptive observation that, [69], p.vii:

> "This quantitative theory of description and computation, or Computational Complexity Theory as it has come to be known, studies the various kinds of resources required to describe and execute a computational process. Its most striking conclusion is that there exist computations and classes of computations having innocent-seeming definitions but nevertheless requiring inordinate quantities of some computational resource."

But there are possible worlds and visions that may make them also realizable. It is possible that the eternal charm of 'innocent-seeming definitions' and simple assumptions leading to unfathomable depths of computations and unimaginably difficult approximations may lure the sinners away from facile computations and faulty theorising.

In my visions for the future I envisage a new generation of graduate students willing to learn the mathematics of the computer, the philosophy of computations and the engineering of numerical analysis. They are being taught by an older generation innocent of the metamathematical way of thinking about computations. They are also being inundated with the axiomatic way of reasoning – which I, not having been brought up on the much vaunted Greek mathematical tradition, have never had to unlearn. I think the most important vision for the future is to find a way to teach economics from a consistently and uncompromisingly algorithmic mode.

Yiannis Moschovakis, one of the outstanding frontier scholars of mathematical logic – in its recursion theoretic, set theoretic and model theoretic modes – concluded his highly stimulating essay on the provocative question of '*What Is an Algorithm*' with the following observation, [48], p.935 (italics added):

> "I would guess that many results in the analysis of algorithms are, in fact, *discovered by staring* at recursive equations (or *informal pro-*

> *cedures* which can be expressed by systems of recursive equations), and then proofs are re-written and grounded on some specific model of computation for lack of a rigorous way *to explain an 'informal' argument.*"

Herbert Simon was the ultimate – even, perhaps, in the sense of being the 'last', 'final' – maestro of proceeding to theorise about decision processes in economics in this way. In my visions of the future of and for economics, there will be re-incarnations of Simons who may make the subject intrinsically algorithmic so that computational complexity can become a routine part of the subject that '*efficiently* allocates scarce resources' – where the notion of 'efficiency' is that of the 'good' and the 'practical'. I end with visions, also, of *Keynesian dentists* dominating the subject with their eminent practical sense of the possible.

A Algorithmic Complexity Theory

"There is also *a technical sense of* '*complexity*' in *logic*, variously known as Kolmogorov complexity, Solomonoff complexity, Chaitin complexity, ..., algorithmic complexity, information-theoretic complexity, and program-size complexity. The most common designation is 'Kolmogorov complexity' .... this is probably a manifestation of the principle of 'Them that's got shall get,' since Kolmogorov is the most famous of these mathematicians[40]."

[21]; p. 137; italics added.

This biblical reason for naming these variations on the theme of complexity may well be plausible from some points of view. However, it appears that Kolmogorov himself suggested that what is usually referred to as 'Kolmogorov complexity' should be called 'Kolmogorov entropy':

"[W]e call 'Kolmogorov entropy' what is usually called 'Kolmogorov complexity.' **This is a Moscow tradition suggested by Kolmogorov himself**. By this tradition the term 'complexity' relates to *any* mode of description and 'entropy' is the complexity related to an *optimal* mode (i.e., to a mode that, roughly speaking, gives the *shortest* description)."

[84], p. 271; italics in the original; bold emphasis, added.

Current orthodoxy of the field of algorithmic complexity theory, linking all of the different variations of the themes mentioned above, is elegantly and comprehensively discussed, explained and described in the almost encyclopaedic treatise by Li and Vitanyi. The 'Russian Tradition', referred to above, is also amply discussed in works by Kolmogorov's students and co-authors (for example, [84], [85]). I have myself had a stab at a concise outline of the field, from the point of view of randomness and induction (cf., [87], chapter 5), to which I may refer the interested reader[41] for a potted survey of the field. There is no point in rehashing easily available visions and comprehensively presented interpretations[42]. In this brief appendix I want to suggest that algorithmic complexity

[40]In their comprehensive and admirable text on this subject, Li and Vitanyi first gave the reason for subsuming all these different variations on one theme by the name 'Kolmogorov Complexity', [42], p.84:

"Associating Kolmogorov's name with [algorithmic] complexity may also be an example of the 'Matthew Effect' first noted in the Gospel according to Matthew, 25:29-30.. ."

[41]John Kelly's famous 'elusive creature'!

[42]As in the case of computational complexity theory, there are encyclopaedic treatises, like those by Schrijver, [68], Papadimitriou, [52], Garey & Johnson, [22], and many others, that tell the story of the field, both from the point of view of the history of the forming ideas and the exciting work at the frontiers. I am neither summarizing these excellent treatises nor writing a survey. My aim here has been to try to give the various stories a slightly different perspective from the point of view of a computable economist. This entails a perspective from a generalized algorithmic vision - not just a recursion theoretic perspective.

theory, even by any other name, has no intrinsic connection with computational complexity theory.

The orthodox story, in an ultra-brief – almost scandalous – nutshell, is that the origins of the field of algorithmic complexity theory lie in the work of Kolmogorov, Chaitin and Solomonoff[43], in their approaches to, respectively, the quantity of information in finite objects, program-size descriptions of the information content of a finite object and induction. For an economist the most interesting approach is that by Solomonoff, whose starting point, in fact, was the *Treatise on Probability, [33]* by Keynes. These original aims developed into, and linked with the earlier research on, the von Mises attempt to define a frequency theory approach to probability, randomness[44] and Bayesian estimation. All this is part of the folklore of the subject, easily gleaned from any of the indicated references, above.

In all three traditions – i.e., the Kolmogorov, Chaitin and Solomonoff – the intentions were to measure the amount of information necessary to *describe* a given, finite, binary sequence (or string). A little more precisely, the idea is as follows: given a string x, its algorithmic complexity is defined to be the shortest string y from which a Universal Turing Machine[45] can 'produce' the given x.On the other hand, in computational complexity theory – particularly as a result of adherence to 'Post's Program' – attention is *not* focused on *individual finite strings.* Instead the fundamental questions are about the computational difficulty – i.e., complexity – of recognising *sets.* Thus, the problem is about *deciding* whether a given finite string belongs to a particular set or not. From the main part of the paper it will be evident that in computational complexity theory one tries to associate a function $\tau_{\wp}$:$\mathbb{N} \to \mathbb{N}$, to a recursive set, $\wp$ such that $\wp$ is accepted by those Turing Machines, say Θ, that run in *time* $\Omega\left(\tau_{\wp}\left(n\right)\right).$ Therefore, one way to link algorithmic complexity theory with computational complexity theory will be to define a notion of the former that is *time-bounded* and is able to capture aspects of the complexity of the set $\wp$. In other words, it is necessary to add a time-bounded complexity component, as is routine in computational complexity theory, to the standard measure of algorithmic complexity. If this is done, the complexity of the finite string, x, will now be defined

[43]I have, in my various related other writings on this subject, also tried to give due credit to the earlier and contemporary independent work of Lars Löfgren and Hilary Putnam. The representative references for Chaitin, Kolmogorov and Solomonoff are, respectively, [8], [39], [77] and [78].

[44]One direct link with the contents of the main part of this paper, i.e., with computational complexity theory, was stated succinctly by Compagner, [12], p.700 (italics added):

> "..[T]he mathematical description of random sequences in terms of complexity, which in algorithmic theory leads to the identification of randomness with *polynomial-time unpredictabilty.*"

Once algorithmic complexity theory is viewed as the basis for a definition of finite random sequences, then it is inevitable that the emphasis will be on prediction rather than computation. Thus, the link with orthodox computational complexity theory is not as firm as the inclusion of the sobriquet 'complexity' in the title may suggest.

[45]In the Solomonoff tradition the corresponding 'universality' resides in the concept of a 'universal distribution'.

by the minimum of the sum of the description length and a measure of the time required to produce that x, from the given description.

Thus, it is clear that algorithmic complexity theory is as separate a field from computational complexity theory as information theory or physics – or even economics – are. The main difference is, of course, that algorithmic complexity theory is, *ab initio*, underpinned by the Turing model of computation. Hence, it is natural to define time-constrained generation of the descriptive complexity of members of sets. For example, as a computable economist, I construct economic theories underpinned by Turing's model of computation. Given a computable economic theory, there will be naturally definable time-bounded measures to describe the theory and, hence, immediate considerations of computational complexity of such descriptions. This is the way the complexity of solutions to ODE is studied, in the references given in the main part of the paper. First, the ODE is constructified; then the computational complexity of the constructified solution is evaluated. Again, the difference is that description is intrinsically algorithmized, in algorithmic complexity theory.

In conclusion, the following two remarks add, I hope, further substance to my stance that the sobriquet 'complexity' in algorithmic complexity theory is somewhat unfortunate.

Firstly, I would like to add another point so as to dispel popular misconceptions about correlating or juxtaposing *complexity* with *incompleteness*[46]. Many an unwary reader of Chaitin's important works - and his specific program-size approach to algorithmic information theory, the incarnation of Kolmogorov complexity in Chaitin's independent work - has had a tendency to claim that incompleteness, undecidability or uncomputability propositions are only valid in so-called '*sufficiently complex*' mathematical systems. *A fortiori*, that intuitively *simple* computable systems are not computationally complex. This is simply false. Very simple formal mathematical systems are capable of generating incompleteness and undecidability propositions; just as intuitively very simple computable systems are capable of encompassing incredibly complex computational complexities, as some of the above examples have shown. Conversely, there are evidently complex systems that are provably complete and decidable; and, similarly, there exist seemingly complex functions that are capable of being computed, even primitive recursively. As one obvious and famous example illustrating *incompleteness* and *essential undecidability* in an *intuitively simple, finitely axiomatizable, simply consistent*[47] theory, one can take Robinson's

[46]Or, *simplicity* with *completeness*. I am, of course, referring to *incompleteness* in the strict metamathematical sense.

[47]See [36], p.287, footnote 216 and [35], p.470, Theorem 53. The part played by *simple consistency* and the analogy with Rosser's result of the essential undecidability of $\mathbb{N}$, [60], is also discussed in the relevant parts of [36]. Furthermore, despite some unfortunate misprints and unclarity, Franzén's fine exposition of the use and abuse of *Gödel's Incompleteness* theorems has a good discussion of the way the *Rosser sentence* (rather than the more famous *Gödel sentence*) is used in proving – by reference to [81] – undecidability in *Robinson's Arithmetic* ([21], pp.158-9).

Arithmetic,[59], as shown in some of the classic books of metamathematics[48], eg., [35], [36], pp. 280-1, [5], p.215,ff.

The issues and points raised in the previous paragraph are, in my opinion, relevant as a preface to any discussion of algorithmic complexity, especially because, as indicated in Franzén's perceptive observation in the above opening quote, this whole area is really about '*a technical sense of* '*complexity*' *in logic*'[49]. I believe it adds substance to the 'Russian tradition' of not using the word 'complexity' to refer to this vast and fascinating field.

Secondly, If the distant modern origins of computational complexity theory, via computability theory, can be found in Hilbert's Tenth Problem, then modern exact approximation theory is even more directly linked to Hilbert's Thirteenth Problem[50]. Hilbert's aim in formulating the 13th Problem – based on problems

[48]I cite this example also because Robinson's Arithmetic is sufficient to represent every recursive function. It figures in the very first, 'Introductory', pages of Odifreddi's comprehensive, yet pedagogical, textbooks of *classical recursion theory, [50],* §I.1, p.23. There are many equivalent ways of setting out the axioms of Robinson's Arithmetic (see, for example, the discussion in [50]); the following is the definition in the exceptionally clear presentation of [5], p.215, ff (the prime denotes the successor operation):

1. $x = \mathbf{0} \vee \exists y, s.t., x = y'$
2. $\mathbf{0} \neq x'$
3. $x' = y' \rightarrow x = y$
4. $x + \mathbf{0} = x$
5. $x + y' = (x + y)'$
6. $x \cdot \mathbf{0} = \mathbf{0}$
7. $x \cdot y' = (x \cdot y) + x$
8. $x < y \leftrightarrow \exists z, (z' + x = y)$

[49]However, I believe Chaitin's reference to his own pioneering work as 'algorithmic information theory', is a much better encapsulation of the contents of the field and the intentions of the pioneers. Indeed, the natural precursor is Shannon, rather than von Mises, but Whig history is a messy affair and straightening out historical threads is a difficult task, even in a contemporary field.

[50]If one reads the main content of the 13th Problem by replacing the word 'nomography' with 'algorithm', then the connection with the subject matter of this paper becomes fairly clear, [28], p.424:

> "[I]t is probable that the root of the equation of the seventh degree is a function of its coefficients which does not belong to the class of functions capable of nomographic construction, i.e., that it cannot be constructed by a finite number of insertions of functions of two arguments. In order to prove this, the proof would be necessary *that the equation of the seventh degree* $f^7 + xf^3 + yf^2 + zf + 1 = 0$ *is not solvable with the help of any continuous functions of only two arguments.* I may be allowed to add that I have satisfied myself by a rigorous process that there exist analytical functions of three arguments x, y, z which cannot be obtained by a finite chain of only two arguments."

Kolmogorov and Arnold refuted Hilbert's conjecture by constructing representations of continuous functions of several variables by the superposition of functions of one variable and sums of functions.

of nomography[51] - was to characterise functions in terms of their *complexity* in a natural way: find those defining characteristics of a function, such that, the given function can be built up from simpler functions and simple operations, rather like – once again – the way partial recursive functions are built up from a collection of base functions and elementary operations, like $\mu - recursion$, minimalisation, etc.. Hilbert's honed intuition suggested the formulation of the 13th Problem; it was – like the 10th Problem – solved 'negatively', by Kolmogorov and Arnold. The point to be emphasised here is, however, not the direct and obvious connection with computational complexity theory[52]; but, to strengthen my thesis that the 'Russian Tradition' of referring to Kolmogorov entropy is entirely justified even from the point of view of his work on approximation theory. Essentially, Kolmogorov introduced the concept of '$\varepsilon-$entropy' of a metric space[53] 'to evaluate the order of increase of the volume of the [nomographic] table for an *increase in the accuracy* of [nomographic] tabulation.' In other words, in his work on approximation theory, preceding his work on algorithmic complexity theory by only a few years, Kolmogorov defined the 'size' of a finite body – actually a subset of a Banach space – in terms of its 'metric entropy'; on the other hand, in his work on algorithmic complexity theory, he defined the information content in a finite string in terms on 'entropy' (in the Shannon tradition), too.

Finally, I don't even think the notion of randomness of finite strings needs to be based on algorithmic complexity theory. I believe the more constructive notion of lawless sequences, first enunciated by Brouwer, provides quite an adequate basis for defining randomness of finite sequences[54]. However, this is quite separate from the use of algorithmic complexity theory to provide rigorous foundations for the von Mises notion of 'Kollektives' and thereby make it possible to remove the circularities that plagued early definitions of the frequency theory of probability.

[51] In the opening lines of the section stating the 13th Problem, Hilbert gives an intuitive idea of 'nomography', [28], p.424:

> "Nomography deals with the problem: to solve equations by means of drawings of families of curves depending on an arbitrary parameter."

Those of us who indulge in drawing *vector fields* might see the similarities!

[52] A beautiful discussion of approximation theory from this point of view – albeit implicitly – is given in an unfortunately little reference work by Vitushkin, [89]; a more technical and comprehensive survey of Kolmogorov's work on approximation theory is in [82].

[53] See [89], p. xiii and [43], chapters 9 & 10. 'Order of increase', 'increase in the accuracy', 'most favourable system of approximation', 'rapidity of convergence' are some of the phrases used in Kolmogorov approximation theory. These are the considerations that make approximation theoretical considerations naturally algorithmic and, therefore, also amenable to computational complexity analysis.

[54] This is a distinctly undigested conjecture; but see [86] for rigorous suggestions along these lines.

References

[1] Aberth, Oliver, (2001), **Computable Calculus**, Academic Press, San diego & London.

[2] Austin, A. Keith, (1983), *An Elementary Approach to NP-Completeness*, **American Mathematical Monthly**, Vol.90, #6, June-July, pp.398–9.

[3] Bishop, Errett A, (1967), **Foundations of Constructive Analysis**, McGraw-Hill Book Company, New York.

[4] Blum, Lenore, Felipe Cucker, Michael Shub and Steve Smale (1998), **Complexity and Real Computation**, Springer Verlag, New York.

[5] Boolos, George. S, John P. Burgess and Richard C. Jeffrey, (2002), **Computability and Logic**, *Fourth Edition*, Cambridge University Press, Cambridge.

[6] Borwein, J. M & P. B. Borwein, (1988), *On the Complexity of Familiar Functions and Numbers*, **SIAM Review**, Vol.30, #4, December, pp.589-601.

[7] Bridges, Douglas & Fred Richman, (1987), **Varieties of Constructive Mathematics**, Cambridge University Press, cambridge.

[8] Chaitin, Gregory J, (1966), *On the Length of Programs for Computing Fintite Binary Sequences*, Journal of the Association of Computing Machinery, Vol.13, pp.549-69.

[9] Cipra, Barry, (2002), **What's Happening in the Mathematical Sciences**, **Vol.5**, edited by Paul Zorn, American Mathematical Society, Providence, Rhode Island.

[10] Cobham, A, (1965), *The Intrinsic Computational Difficulty of Functions*, pp.24-30, in: **Logic, Methodology and Philosophy of Science - Proceedings of the 1964 International Congress**, edited by Y.Bar-Hillel, North-Holland, Amsterdam.

[11] Cole, F. N, (1903), *On the Factoring of Large Numbers*, **Bulletin of the American Mathematical Society**, Vol.10, #3, December, pp.134-7.

[12] Compagner, Aaldert, (1964), *Definitions of Randomness*, **American Journal of Physics**, Vol. 59, # 8. August, pp.700-705

[13] Cook, S. A, (1971), *The Complexity of Theorem Proving Procedures*, pp. 151-58,**Proceedings of the 3rd ACM Symposium on the Theory of Computing**, ACM Publications.

[14] da Costa, Newton C A & Francisco A. Doria (1991), *Undecidability and Incompleteness in Classical Mechanics*, **International Journal of Theoretical Physics**, Vol. 30, #8, August, pp.1041-73.

[15] da Costa, Newton C A & Francisco A. Doria (1991), *Classical Physics and Penrose's Thesis*, **Foundations of Physics Letters**, Vol. 4, #4, August, pp.363-73.

[16] da Costa, Newton C. A & Francisco A. Doria & E. Bir (2007), *On the Metamathematics of the P vs. NP Question*, **Applied Mathematics and Computation**, Vol.189, pp.1223-40.

[17] Davis, Martin (1958), **Computability and Unsolvability**, McGraw-Hill, New York.

[18] Edmonds, J, (1965), *Paths, Trees and Flowers*, **Canadian Journal of Mathematics**, Vol.17, pp.449-67.

[19] Einstein, Albert, (1955), '*Relativistic Theory of the Non-Symmetric Field*', in: **The Meaning of Relativity** (5th Edition), Appendix II, pp. 133-166, Princeton University Press, Princeton, New Jersey.

[20] Feynman, Richard P, (1982), '*Simulating Physics with Computers*', **International Journal of Theoretical Physics**, Vol. 21, Nos. 6/7, pp. 467-488.

[21] Franzén, Torkel, (2005), **Gödel's Theorem: An Incomplete Guide to its Use and Abuse**, A K Peters, Wellesley, Massachusetts.

[22] Garey, Michael R & David S. Johnson, (1979), **Computers and Intractability: A Guide to the Theory of NP-Completeness**, W.H. Freeman and Company, New York.

[23] Gödel, Kurt, (1956), *Letter to von Neumann*, 20 March, 1956, translated and reprinted in: Micheal Sipser, (1992), *The History and Status of the P versus NP Question*, **Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing,** Victoria, British Columbia, pp.603-618.

[24] Goldreich, Oded, (2001), *Computational Complexity*, in: **Mathematics Unlimited – 2001 and Beyond**, edited by Björn Engquist and Wilfried Schmid, Springer-Verlag, Berlin and Heidelberg.

[25] Goodwin, Richard. M, (1951), *Iteration, Automatic Computers, and Economic Dynamics*, **Metroeconomica**, Vol. III, Fasc. 1, April, pp.1-7.

[26] Goodstein, R. L, (1961), **Recursive Analysis**, North-Holland Publishing Co., Amdterdam.

[27] Grzegorczyk, Andrezej, (1957), *On the Definitions of Computable real Continuous Functions*, **Fundamenta Mathematicae**, Vol. 44, pp.61-71.

[28] Hilbert, David, (1900), *Mathematical Problems*, **Bulletin of the American Mathematical Society**, Vol. 8, July, 1902, pp.437-79; translated from the original German: *Mathematische Probleme* by Dr Mary Winston Newson.

[29] Jaffe, Arthur. M, (2006), *The Millennium Grand Challenge in Mathematics*, **Notices of the American Matheamtical Society**, Vol.53,#6, June/July, pp.651-60.

[30] Jeffreys, Harold, (1926), *On the Relation to Physics of the Notion of Convergence of Series*, **Philosophical Magazine**, Ser.7, Vol.2, #7, July, pp.241-44.

[31] Karp, R. M., (1972), *Reducibility Among Combinatorial Problems*, pp.85-104, in: **Complexity of Computer Computations**, edited by R.E. Miller & J.W. Thatcher, Plenum Press, New York.

[32] Karp, Richard, (1987), *Complexity and Parallel Processing: An Interview with Richard Karp*, a *Turing Award Interview* with Karen A. Frenkel, in: **ACM Turing Award Lectures - The First Twenty years, 1966-1985**, ACM Press/ Addison-Wesley Publishing Company, Reading, Massachusetts.

[33] Keynes, John Maynard, 1921), **A Treatise on Probability**, Macmillan and Company Limited, London.

[34] Keynes, John Maynard, (1930), *Economic Possibilities for our Grandchildren*, reprinted in: **Essays in Persuasion** by John Maynard Keynes, The Norton Library, W.W. Norton & Company, Inc., New York.

[35] Kleene, Stephen. C, (1952), **Introduction to Metamathematics**, D. Van Nostrand Company, Inc., Princeton, New Jersey.

[36] Kleene, Stephen. C, (1967), **Mathematical Logic**, John Wiley & Sons, Inc., New York.

[37] Ko, Ker-I, (1991), **Complexity Theory of Real Functions**, Birkhäuser, Boston and Basel.

[38] Ko, Ker-I & Harvey Friedman, (1982), *Computational Complexity of Real Functions*, **Theoretical Computer Science**, Vol. 20, pp.323-352.

[39] Kolmogorov, Andrei N, (1965), *Three Approaches to the Definition of the Concept of the 'Amount of Information'*, **Problems of Information Transmission**, Vol.1, #1, pp.1-7.

[40] Kolmogorov, Andrei N, (1983), *Combinatorial Foundations of Information Theory and the Calculus of Probabilities*, **Russian Mathematical Surveys**, Vol.38, #4, pp.29-40.

[41] Lacombe, D, (1955), *Extension de la notion de fonctions récursive aux fonctions d'une ou plusieurs variables réelles, I, II, III,* ***C.R.Acad., Paris***, Vol.240, pp. 2478-80, Vol.241, pp.13-14, pp. 151-3.

[42] Li, Ming and Paul Vitanyi, (1993), **An Introduction to Kolmogorov Complexity and its Applications**, Springer-Verlag, Berlin and Heidelberg.

[43] Lorentz, G.G, (1986), **Approximation of Functions**, AMS Chelsea Publishing, Providence, Rhode Island.

[44] Lucas, Robert. E, Jr., (1993), *Making a Miracle*, **Econometrica**, Vol.61, #2, March, pp.251-72.

[45] Machlup, Fritz, (1958), *Structure and Structural Change: Weaselwords and Jargon*, **Zeitschrift für Nationalökonomie**, Vol.18, pp.280-98.

[46] Mazur, S, (1963), **Computable Analysis**, Rozprawy Matematyczne, XXXIII, edited by: Andrzej Grzegorczyk and Helena Rasiowa, PWN Publishers, Warszawa.

[47] Moore, Christopher, (1990), *Unpredictability and Undecidability in Dynamical Systems,* **Physical Review Letters**, Vol.64, #20, 14 May 1990, pp.2354-7.

[48] Moschovakis, Yiannis, (2001), *What is an Algorithm?*, pp.919-36, in: **Mathematics Unlimited – 2001 and Beyond**, edited by Björn Engquist and Wilfried Schmid, Springer-verlag, Berlin and Heidelberg.

[49] Mount, Kenneth. R & Stanley Reiter (2002), **Computation and Complexity in Economic Organization**, Cambridge University Press, Cambridge.

[50] Odifreddi, Piergiorgio, (1989), **Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers**, North-Holland, Amsterdam and New York.

[51] Papadimitriou, Christos. H and Kenneth Steiglitz, (1982), **Combinatorial Optimization: Algorithms and Complexity**, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[52] Papadimitriou, Christos. H, (1994), **Computational Complexity**, Addison-Wesley Publishing Company, Reading, Massachusetts.

[53] Penrose, Roger, (2004), **The Road to Reality: A Comprehensive Guide to the Laws of the Universe**, Jonathan Cape, London.

[54] Post, Emil, (1944), *Recursively Enumerable Sets of Positive Integers and their Decision Problems*, **Bulletin of the American Mathematical Society**, Vol. 50, pp.284-316.

[55] Pour-El, Marian. B & Jonathan I. Richards, (1989), **Computability in Analysis and Physics**, Springer-Verlag, Berlin Heidelberg & New York.

[56] Pratt,Vaughan. R, (1975), *Every Prime has a Succinct Certificate*, **SIAM Journal of Computing**, Vol.4, pp.214-220.

[57] Richman, Fred (1990), *Intuitionism as Generalization*, **Philosophia Mathematica**, Vol. 14, pp. 124-8.

[58] Rogers, Hartley, Jr., (1967): **Theory of Recursive Functions and Effective Computability**, McGraw-Hill, New York.

[59] Robinson, Raphael. M, (1952), *An Essentially Undecidable Axiom System*, pp.729-30, in Vol.1 of: **Proceedings of the International Congress of Mathematicians**, Cambridge, Massachusetts, 1950; American Mathematical Society, Providence, Rhode Island.

[60] Rosser, J. Barkley (1936), *Extensions of Some Theorems of Gödel and Church*, **Journal of Symbolic Logic**, Vol. 1, # 3, September, pp.87-91.

[61] Rosser, J. Barkley, Jr., (2007),*Computational and Dynamic Complexity in Economics,* downloadable at: cob.jmu.edu/rosserjb/ DYNAMIC%20 AND%20COMPUTATIONAL%20COMPLEXITY%20IN%20ECONOMICS.doc, pp.21.

[62] Rouse Ball, W. W & H.S.M. Coxeter, (1987), **Mathematical Recreations and Essays**, Thirteenth Edition, Dover Publications, Inc., New York.

[63] Rubel, Leo. A (1988), *Some Mathematical Limitations of the General-Purpose Analog Computer*, **Advances in Applied Mathematics**, Vol. 9, pp.22-34.

[64] Schechter, Eric, (1997), **Handbook of Analysis and Its Foundations**, Academic Press, San Diego.

[65] Shannon, Claude E, (1941), *Mathematical Theory of the Differential Analyzer,* Journal of Mathematics and Physics, Vol. XX, No. 4, December, pp. 337-354.

[66] Shepherdson, J.C, (1976), *On the Definition of Computable Functions of a Real Variable*, **Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik**, Vol.22, pp.391-402.

[67] Schrijver, Alexander, (1986), **Theory of linear and Integer Programming**, John Wiley & Sons, Chichester and new York.

[68] Schrijver, Alexander, (2003), **Combinatorial Optimization: Polyhedra and Efficiency: Volumes A, B, C**, Springer-Verlag, Berlin & Heidelberg.

[69] Schwartz, Jacob T, (1987), *Foreward*, to: **Algorithmic Information Theory** by Gregory J. Chaitin, Cambridge University Press, Cambridge.

[70] Sipser, Michael, (1983), *Borel Sets and Circuit Complexity*, **Proceedings of the 15th ACM Symposium on the Theory of Computing**, Association for Computing Machinery.

[71] Sipser, Micheal, (1992), *The History and Status of the P versus NP Question*, **Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing,** Victoria, British Columbia, pp.603-618.

[72] Smale, Stephen, (1988), *The Newtonian Contribution to Our Understanding of the Computer*, pp. 90-5, in, **Newton's Dream**, edited by Marcia Sweet Stayer, Published for *Queen's Quarerly*, by McGill-Queen's University Press,

[73] Smale, Stephen, (1991), *Theory of Computation*, pp. 60-9, in: **Mathematical Research Today and Tomorrow: Viewpoints of Seven Fields Medalists!**, edited by C.Casacuberta and M. Castellet, Springer-Verlag, Berlin and Heidelberg.

[74] Smorynski, Craig, (1985), *"Big" News from Archimedes to Friedman*, pp.353-66, in: **Harvey Friedman's Research on the Foundations of Mathematics**, edited by L.A.Harrington, M.D. Morley, A. Ščedrov and S.G.Simpson, North-Holland, Amsterdam & New York.

[75] Smorynski, Craig, (1985), *Some Rapidly Growing Functions*, pp.367-80, in: **Harvey Friedman's Research on the Foundations of Mathematics**, edited by L.A.Harrington, M.D. Morley, A. Ščedrov and S.G.Simpson, North-Holland, Amsterdam & New York.

[76] Soare, Robert. I, (1987), **Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets**, Springer-Verlag, Berlin and Heidelberg.

[77] Solomonoff, Ray. J, (1964a), *A Formal Theory of Inductive Inference: Part I*, **Information and Control**, Vol.7, pp.1-22.

[78] Solomonoff, Ray. J, (1964b), *A Formal Theory of Inductive Inference: Part II*, **Information and Control**, Vol.7, pp.224-54.

[79] Specker, Ernst, (1949), *Nicht konstruktiv beweisbare Sätze der Analysis*, **The Journal of Symbolic Logic**, Vol. 14, #3, pp.145-158.

[80] Srinivasa Rao, K, (1998), **Srinivasa Ramanujam: A Mathematical Genius**, EastWest Books (Madras) Pvt. Ltd., Chennai & Bangalore.

[81] Smullyan, Raymond, (1992), **Gödel's Incompleteness Theorems**, Oxford University Press, Oxford.

[82] Tikhomirov, V. M, (1989), *A.N.Kolmogorov and Approximation Theory*, **Russian Mathematical Surveys**, Vol.44, # 1, pp.101-152.

[83] Turing, Alan, M, (1936-7), *On Computable Numbers, with an Application to the Entscheidungsproblem*, **Proceedings of the Lodon Mathematical Society**, Vol. 42, pp.230-65, *Correction*, Vol. 43, pp.544-6.

[84] Uspensky, Vladimir. A & A. Shen, (1996), *Relations between Varieties of Kolmogorov Complexities*, **Mathematical Systems Theory**, Vol.29, pp.271-92.

[85] Uspensky, Vladimir. A, (1992), *Complexity and Entropy: An Introduction to the Theory of Kolmogorov Complexity*, in: **Kolmogorov Complexity and Computational Complexity**, pp.85-102, edited by Osamu Watanabe, Springer-Verlag, Berlin and Heidelberg.

[86] van Lambalgen, Michiel, (1992), *Independence, Randomness and the Axiom of Choice*, **Journal of Symbolic Logic**, Vol.57, # 4, December, pp.1274-1304.

[87] Velupillai, Kumaraswamy, (2000), **Computable Economics**, Oxford University Press, Oxford.

[88] Velupillai, K Vela, (2004), *Economic Dynamics and Computation – resurrecting the Icarus Tradition*, **Metroeconomica**, Vol.55, # 2&3, May/September, pp.239-64.

[89] Vitushkin, A. G, (1961), **Theory of the Transmission and Processing of Information**, translated from the Russian by: Ruth Feinstein, Pergamon Press, Oxford and London.

[90] Weihrauch, Klaus (2000), **Computable Analysis: An Introduction**, Springer-Verlag,Berlin & Heidelberg.

[91] Wigderson, Avi, (2006), *$\mathcal{P}$, $\mathcal{NP}$ and Mathematics – A Computational Complexity Perspective*, downloaded from: WWW.MATH.IAS.EDU /~AVI/PUBLICATIONS/MYPAPERS/W06/W06.PDF, PP.48.

[92] Wilf, Herbert. S, (1986), **Algorithms and Complexity**, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

2002.6 *L'economia senza banditore di Axel Leijonhufvud: le 'forze oscure del tempo e dell'ignoranza' e la complessità del coordinamento,* by Elisabetta De Antoni.

2002.7 *Why is Trade between the European Union and the Transition Economies Vertical?,* by Hubert Gabrisch and Maria Luigia Segnana.

2003.1 *The service paradox and endogenous economic gorwth,* by Maurizio Pugno.

2003.2 *Mappe di probabilità di sito archeologico: un passo avanti,* di Giuseppe Espa, Roberto Benedetti, Anna De Meo e Salvatore Espa.
(*Probability maps of archaeological site location: one step beyond,* by Giuseppe Espa, Roberto Benedetti, Anna De Meo and Salvatore Espa).

2003.3 *The Long Swings in Economic Understianding,* by Axel Leijonhufvud.

2003.4 *Dinamica strutturale e occupazione nei servizi,* di Giulia Felice.

2003.5 *The Desirable Organizational Structure for Evolutionary Firms in Static Landscapes,* by Nicolás Garrido.

2003.6 *The Financial Markets and Wealth Effects on Consumption* An Experimental Analysis, by Matteo Ploner.

2003.7 *Essays on Computable Economics, Methodology and the Philosophy of Science,* by Kumaraswamy Velupillai.

2003.8 *Economics and the Complexity Vision: Chimerical Partners or Elysian Adventurers?*, by Kumaraswamy Velupillai.

2003.9 *Contratto d'area cooperativo contro il rischio sistemico di produzione in agricoltura,* di Luciano Pilati e Vasco Boatto.

2003.10 *Il contratto della docenza universitaria. Un problema multi-tasking*, di Roberto Tamborini.

2004.1 *Razionalità e motivazioni affettive: nuove idee dalla neurobiologia e psichiatria per la teoria economica?* di Maurizio Pugno.
(*Rationality and affective motivations: new ideas from neurobiology and psychiatry for economic theory?* by Maurizio Pugno.

2004.2 *The economic consequences of Mr. G. W. Bush's foreign policy. Can th US afford it?* by Roberto Tamborini

2004.3 *Fighting Poverty as a Worldwide Goal* by Rubens Ricupero

2004.4 *Commodity Prices and Debt Sustainability* by Christopher L. Gilbert and Alexandra Tabova

2004.5 *A Primer on the Tools and Concepts of Computable Economics* by K. Vela Velupillai

2004.6 *The Unreasonable Ineffectiveness of Mathematics in Economics* by Vela K. Velupillai

2004.7 *Hicksian Visions and Vignettes on (Non-Linear) Trade Cycle Theories* by Vela K. Velupillai

2004.8 *Trade, inequality and pro-poor growth: Two perspectives, one message?* By Gabriella Berloffa and Maria Luigia Segnana

2004.9 *Worker involvement in entrepreneurial nonprofit organizations. Toward a new assessment of workers? Perceived satisfaction and fairness* by Carlo Borzaga and Ermanno Tortia.

2004.10 *A Social Contract Account for CSR as Extended Model of Corporate Governance (Part I): Rational Bargaining and Justification* by Lorenzo Sacconi

2004.11 *A Social Contract Account for CSR as Extended Model of Corporate Governance (Part II): Compliance, Reputation and Reciprocity* by Lorenzo Sacconi

2004.12 *A Fuzzy Logic and Default Reasoning Model of Social Norm and Equilibrium Selection in Games under Unforeseen Contingencies* by Lorenzo Sacconi and Stefano Moretti

2004.13 *The Constitution of the Not-For-Profit Organisation: Reciprocal Conformity to Morality* by Gianluca Grimalda and Lorenzo Sacconi

2005.1 *The happiness paradox: a formal explanation from psycho-economics* by Maurizio Pugno

2005.2 *Euro Bonds: in Search of Financial Spillovers* by Stefano Schiavo

2005.3 *On Maximum Likelihood Estimation of Operational Loss Distributions* by Marco Bee

2005.4 *An enclave-led model growth: the structural problem of informality persistence in Latin America* by Mario Cimoli, Annalisa Primi and Maurizio Pugno

2005.5 *A tree-based approach to forming strata in multipurpose business surveys,* Roberto Benedetti, Giuseppe Espa and Giovanni Lafratta.

2005.6 *Price Discovery in the Aluminium Market* by Isabel Figuerola-Ferretti and Christopher L. Gilbert.

2005.7 *How is Futures Trading Affected by the Move to a Computerized Trading System? Lessons from the LIFFE FTSE 100 Contract* by Christopher L. Gilbert and Herbert A. Rijken.

2005.8 *Can We Link Concessional Debt Service to Commodity Prices?* By Christopher L. Gilbert and Alexandra Tabova

2005.9 *On the feasibility and desirability of GDP-indexed concessional lending* by Alexandra Tabova.

2005.10 *Un modello finanziario di breve periodo per il settore statale italiano: l'analisi relativa al contesto pre-unione monetaria* by Bernardo Maggi e Giuseppe Espa.

2005.11 *Why does money matter? A structural analysis of monetary policy, credit and aggregate supply effects in Italy*, Giuliana Passamani and Roberto Tamborini.

2005.12 *Conformity and Reciprocity in the "Exclusion Game": an Experimental Investigation* by Lorenzo Sacconi and Marco Faillo.

2005.13 *The Foundations of Computable General Equilibrium Theory*, by K. Vela Velupillai.

2005.14 *The Impossibility of an Effective Theory of Policy in a Complex Economy*, by K. Vela Velupillai.

2005.15 *Morishima's Nonlinear Model of the Cycle: Simplifications and Generalizations,* by K. Vela Velupillai.

2005.16 Using and Producing *Ideas* in Computable Endogenous Growth, by K. Vela Velupillai.

2005.17 *From Planning to Mature: on the Determinants of Open Source Take Off* by Stefano Comino, Fabio M. Manenti and Maria Laura Parisi.

2005.18 *Capabilities, the self, and well-being: a research in psycho-economics,* by Maurizio Pugno.

2005.19 Fiscal and monetary policy, unfortunate events, and the SGP arithmetics. Evidence from a *growth-gap* model, by Edoardo Gaffeo, Giuliana Passamani and Roberto Tamborini

2005.20 *Semiparametric Evidence on the Long-Run Effects of Inflation on Growth,* by Andrea Vaona and Stefano Schiavo.

2006.1 *On the role of public policies supporting Free/Open Source Software. An European perspective,* by Stefano Comino, Fabio M. Manenti and Alessandro Rossi.

2006.2 *Back to Wicksell? In search of the foundations of practical monetary policy,* by Roberto Tamborini

2006.3 *The uses of the past,* by Axel Leijonhufvud

2006.4 *Worker Satisfaction and Perceived Fairness: Result of a Survey in Public, and Non-profit Organizations,* by Ermanno Tortia

2006.5 *Value Chain Analysis and Market Power in Commodity Processing with Application to the Cocoa and Coffee Sectors,* by Christopher L. Gilbert

2006.6 *Macroeconomic Fluctuations and the Firms' Rate of Growth Distribution: Evidence from UK and US Quoted Companies,* by Emiliano Santoro

2006.7 *Heterogeneity and Learning in Inflation Expectation Formation: An Empirical Assessment,* by Damjan Pfajfar and Emiliano Santoro

2006.8 Good *Law & Economics* needs suitable microeconomic models: the case against the application of standard agency models: the case against the application of standard agency models to the professions, by Lorenzo Sacconi

2006.9 *Monetary policy through the "credit-cost channel". Italy and Germany,* by Giuliana Passamani and Roberto Tamborini

2007.1 *The Asymptotic Loss Distribution in a Fat-Tailed Factor Model of Portfolio Credit Risk,* by Marco Bee

2007.2 *Sraffa?s Mathematical Economics – A Constructive Interpretation,* by Kumaraswamy Velupillai

2007.3 *Variations on the Theme of Conning in Mathematical Economics,* by Kumaraswamy Velupillai

2007.4 *Norm Compliance: the Contribution of Behavioral Economics Models,* by Marco Faillo and Lorenzo Sacconi

2007.5 *A class of spatial econometric methods in the empirical analysis of clusters of firms in the space,* by Giuseppe Arbia, Giuseppe Espa e Danny Quah.

2007.6 *Rescuing the LM (and the money market) in a modern Macro course,* by Roberto Tamborini.

2007.7 *Family, Partnerships, and Network: Reflections on the Strategies of the Salvadori Firm of Trento,* by Cinzia Lorandini.

2007.8 *I Verleger serici trentino-tirolesi nei rapporti tra Nord e Sud: un approccio prosopografico,* by Cinzia Lorandini.

PUBBLICAZIONE REGISTRATA PRESSO IL TRIBUNALE DI TRENTO