Complexity of a *modelling exercise*: a critique of the role of computer simulation in Complex System Science

Fabio Boschetti and David McDonald
CSIRO – CMAR, Australia

DRAFT

Abstract

Shifting the emphasis from a *model* to a modelling *task*, which involves both a computer model and a modeller, we ask what makes a problem complex. We propose that a modelling task can be seen as a set of questions- and-answers, nested at multiple levels. The role of the modeller then lies in posing the questions and choosing the best procedure to answer them, while the role of the model lies in answering the questions, via algorithmic, thus logically simple, procedures. Within this framework, complexity is broadly related to the number of question-answer levels involved in the process and addressing this complexity depends crucially on the ingenuity and creativity of the modeller, which are fundamentally non-algorithmic processes. This may lead to a view of complexity which is no more observer-independent, nor computable, but rather accounts for both historical and cultural development, as well as for the context of the problem at hand.

1 Introduction

A fundamental *concept* underpinning Complex System Science (CSS) is that local interactions between relatively simple components can lead to considerably more complex non-local behaviours. A fundamental *conjecture* CSS attempts to study is that these local interactions are the drivers for processes like self-organisation and emergence and, in turn, are responsible for the immense variety of structures, patterns and phenomena we see in Nature. These two ideas can be found, in slightly different form, in most 'manifestos' of CSS [14]. These manifestos can be seen either as refreshingly humbling or as suspiciously audacious. From one side a CS scientist is willing to abandon the lure of formal mathematical rigor and embrace all potentially useful tools to address problems which classical scientific disciplines have deserted. From the other side, studying processes as different as galaxy clustering and biological evolution under the same framework may be seen as yet another dangerous attempt at a 'theory of everything', this time replacing fields and forces with patterns, structure and organisation.

We subscribe to the humble view. We believe that the tendency of CS scientists to embrace any tool available to address the problem at hand, be this a mathematical equation, a computer simulation, visual pattern recognition or discourse in human language, may be the *only* avenue which can enable CSS to address the fundamental conjecture we described above. With this view, the current lack of a formal theory of complex systems may be seen as a strength rather than merely a limitation.

Formatted: Right: 0.63 cm

The purpose for the present paper is to justify this claim and to emphasise the crucial role that a Complex Systems scientist plays in addressing a complex problem. We do so by highlighting a paradox which may arise if the focus of CSS is limited to computational tools and their implementations in the form of computer models: the very tools which allowed CSS to blossom may prevent it from addressing the phenomena CS scientists aim to study. The paradox can be summarised as follows:

1) the formal analytical tools which proved so successful in physics, work well when they address systems comprised of a very small or a very large number of components, whose behaviour is mostly context independent. Somehow they fall apart in the middle sized, strongly contextual, problems which CSS addresses;
2) only way to circumvent this scale-dependency problem is via numerical simulations;
3) studying how local interactions lead to global behaviour implies, inevitably, large computation loads which cannot be carried out by hand;
4) the advent of cheap and widely available computers was thus essential for allowing scientists to address those problems (with computers came simulation, and with simulation CSS was finally possible);
5) the limits of formal logic, on which computation is based, prevent us from studying self-organisation and emergence (supposedly the drivers for generation of novelty and diversity) in their deepest formulation [15]; and
6) computer simulation, currently essential for CSS, is unable to model the very phenomena which typically distinguish CSS from other disciplinary approaches.

Comment: I am not at all sure about this one. It seems to pre-suppose that self-organisation and emergent behaviour can only arise on or beyond the fringe of mathematical accessibility.

In the rest of the paper we argue that, in practise, only a small fraction of a CSS problem is addressed via computation. Rather, the choice of the problem, its formulation, the selection of the numeral tools to address it, the interpretation of the results and the choice of how to act in response to such results, are all steps which involve the creativity and ingenuity of a CS scientists. It has been argued elsewhere [7,8] that this creative process may not be constrained by the limits of formal logic. Whether this is correct or not, the complexity of a modelling exercise cannot be estimated without accounting for this crucial contribution and we propose some initial steps in this direction. Our target audience consists of modellers, in a multidisciplinary sense; consequently we aim to discuss this subject in a fairly informal manner, trying to avoid technical terms and referring the interested reader to the ample available literature should he/she wish to focus on the topic at more depth.

2 Formal Logic and Computation

Our discussion relies crucially on the equivalence between the workings of formal systems and computation [5,9,10,12]. Both start from a set of axioms and input data (defined a priori) and a set of inference rules (transformation rules/computer instructions). These generate outputs such as theorems and computational results.

As discussed in [16], in a formal system the truth of a statement is a property which depends only on the set of axioms and inference rules; once these are given, true statements are true for all possible scenarios and consequently do not confer any

Formatted: Right: 0.63 cm

information about the real world. A typical example is the statement 'it's raining', which may or may not be true depending on the weather conditions. Thus, assessing whether the statement is true or not provides information about the real world. A mathematical theorem, instead, is true independently of Nature's vagaries, once the assumptions it depends on (axioms and inference rules) are correct. The fact that Pythagoras' theorem, for example, is useful and matches our perception of reality is due to the clever choices of the axioms of Euclidean geometry; it also helps us to understand Nature better by simplifying geometrical considerations, by putting place holders in our geometrical thinking so we do not always need to refer back to the axioms, and it helps us to communicate this understanding culturally. It does not, however, provide any information which is not already contained in the axioms. The observation which is most relevant to our discussion is that true statements are *transformations* of information, not new information. All theorems of Euclidean geometry could be compressed, with no loss of information, into the basic axioms and inference rules [17].

Comment: I don't think this is right. It basically asserts what Hilbert wanted to prove, and seems to run counter to your argument about the role of the modeller.

Comment: This isn't quite true. If it were, a mathematician would never need to define things. Nothing in ZFC or in the mathematical/logical rules which we apply makes the notion of a chessboard inevitable. Mathematicians add structure, and the basic rules we have adopted mean that consequences from those rules must follow. We cannot compress all of Euclid into the axioms and the transformations: there are also definitions which (often) construct analogues to reality within the mathematical environment. ZFC doesn't inevitably imply the existence of a chessboard, but we *can* prove that it is impossible to place nine real rooks on a real chessboard in such a way that the rooks cannot interfere with each other.

The other aspect to this is that we *do* gain information about the physical world from out theorems. Most of modern nuclear physics is the process of looking for things that the mathematics says ought to be there. Neptune and Pluto were both discovered because the mathematics said they should be. We didn't know the planets were there until the model suggested that they had to be.

Here endeth the rant :-)
\

The PCs on our desks are equivalent to a finite tape Turing Machine (TM), an abstract and general computational device commonly employed in theoretical computer science. Because the working of a TM is equivalent to that of a formal system [9], it follows that all TM's statements are independent of reality. It thus also follows that any computer model we use *transforms* the information contained in the input via the coded algorithm, but does not *generate* information. Clearly, a model's output helps our finite mental capability to see consequences of what we code (which at times we cannot envisage), but its truth and relevance to the real world is limited to the truth and relevance of what the user codes and the input fed to the computation. No actual information about the real world is produced by a simulation. Information is generated solely by the writing of the code, the choice of the input and the comparison of the outputs to what we observe in Nature, which, in turn tells us about the appropriateness of the rules we implement and the input we choose.

3 Forward and Inverse Modelling

Let's assume that we work with a model F of a physical process and we consider the model to be a black box. The process which, given a certain input i, determines an output o, ($o=F[i]$) represents our understanding of cause-effect relations. By changing the input i, we consequently change the output o; that is, we can control the output o, by acting on i, as we satisfy an operative definition of causality [4,18] which allows us to timidly circumvent known philosophical problems. Since F respects our perception of the 'arrow of time' (cause leads to an effect), it is usually called a 'forward' model.

Many engineering and scientific problems however ask question like "what is the cause of this effect?" ("what caused this flood?", "what causes this disease?", "what can stop the next pandemic?"). This implies a hypothetical model I which, given an effect (o) as input, gives a possible cause as an output ($i=I[o]$). Because I reverses the 'arrow of time', it is usually called an inverse model.

Unfortunately, inverse models I can be written explicitly for only a very small set of forward models F. This is true not only for closed-form models but also for purely numerical models. As a result, most inverse engineering and scientific problems need

Formatted: Right: 0.63 cm

to be solved by iterative methods, in which sets of inputs *i*, are tested until a reasonable match between *F[i]* and *o* is found. The procedure which allows us to recover *i* from *F* and *o* is called inversion, optimisation, or regression, depending on the discipline, and we refer to it generally as inverse modelling.

4 Modelling a Complex System

In this section we consider a hypothetical Complex System Model (*CSM)* and draw some conclusions about the way it is commonly used. In order to represent a stereotypical complex system model, we imagine that our *CSM* is an agent-based model in which a relatively large set of agents interact with one another by using simple local rules. We also imagine that the purpose of the modelling exercise is to study the global pattern arising from the resulting dynamics, which, as is commonly assumed, results in self-organisation and emergent behaviours.

Following Section 3, *CSM* is a forward model, which takes the local rules as input *i* and generates an emergent behaviour and self-organised dynamics as output *o*, *o=CSM[i]*. Also, because *CSM* is an algorithm, the discussion at Section 2 applies, and we infer that any dynamical behaviour in *o* is implicitly determined by *i* and *CSM*. Indeed, no matter how complex, large, apparently unpredictable and surprising *o* is, its entire information content can be compressed into *i* and *CSM* [5].

In [15] Boschetti and Gray discussed the implications of these simple considerations for the modelling and understanding of emergence and proposed three different classes of emergence, at least one of which cannot be studied by computer modelling. Here we limit ourselves to discussing the modelling process and what we can hope to learn from it when we aim to understand more complex phenomena in Nature.

The discussion in Section 2 seems to rule out the possibility of novelty or creative processes solely via numerical modelling (see also [15,16]). It also points out that, from a purely logical perspective, no difference exists between modelling a complex and a non complex system, as long as the modelling is carried out on an implementation of a Turing Machine, as are all of our computers. The only form of emergence within our reach is thus the milder version of pattern generation [2], according to which we define as emergent those features whose dynamics are not *explicitly* coded in the lower-level rules. As far as we know, the word ‘explicit’ in the previous sentence does not have a formal definition, and consequently we fall into the observation, proposed by Shalizi [1], that emergence becomes a user dependent concept or, even worse, a concept dependent on the expertise, experience and the intelligence of the user.

Consider the standard scenario under which our *CSM* could be used for research or engineering purposes. Supposedly, the user has some expectation of what kind of global behaviour he/she intends to model or which kind of local rules he/she wants to study. We imagine the following steps:

1) the *CSM* is written and some input *i* is chosen;

2) the *CSM* is run and the output *o* (an image, an animation, numerical data, etc.) is obtained;
3) the user analyses the output, often visually and subjectively (sometimes numerically), and expresses a judgement on whether the output matches the expectations (possibly a pattern seen in Nature which he/she wants to reproduce or a result of the local rules which he/she judges to be 'realistic' or 'interesting');
4) if the outcome is not satisfactory, or something unexpected in the output sparks new insights, the user may decide to change the input *i* or the rules in *CSM* and go back to 2.

It is clear that the sequence 1-4 represents an iterative inversion process as described in Section 3, our best bet at addressing an even mildly complicated inverse problem. It is also clear that the only 'logically nontrivial' steps are 1, 3 and 4, in which the user is required to use ingenuity and expertise to design or adjust the code, to judge the output realism or accuracy (by analysing its inner patterns) and to select further input for testing. Step 2, the running of *CSM*, is the 'logically trivial' one, the one in which no information is generated, but only transformed according to pre-defined rules.

It is also clear that the only 'logically nontrivial' information we can recover from the exercise in steps 1-4 is the 'inverse' deduction of the suitable input and the suitable transformation rules in *CSM*, not the generation of the patterns in the output, since they contain no information which is not already contained in *i* and *CSM*.

5 The importance of the modeller

The previous section shows that no formal difference can be found between a complex and a traditional non complex model; they are both algorithms and this imposes clear limitations on the characteristics of the dynamics that our *CSM* model can produce. Does this mean that the exercise of carrying out a task by *running* a complex model and a non complex model are the same? We believe that the outputs will differ considerably and that the difference lies in the role of the user, more than in the algorithm.

The information theory and computational mechanics literature provides the clear impression of a widely-held belief that a model *represents* our understanding of a problem [2,3]. If an agent can break down its understanding of a problem as a set of rules, which it can apply to infer and predict a system's behaviour, then the agent has the most compact and efficient description of the problem at hand [1]. The rules, obviously, result in an algorithm, thus the model.

This is surely true for simple problems. A few decades ago pupils were taught how to calculate a square root by hand thus carrying out (hopefully consciously) the required algorithm. Today, we press a button on a calculator. There are no reasons to believe that turning the square root into a complete black box routine has diminished the mathematical ability of today's pupils. We can take a slightly more complicated problem, like a Fourier Transform or a Kolmogorov-Smirnov test [19]. Most practitioners can use those tools as black boxes too. It is only when something in the output looks somehow suspicious (for example, the Gibbs effect in the signal, or

unexpected correlation in the data) that the user may be led to ponder what actually happens in the black box and wonder whether the tool is appropriate for the problem. Should he/she need to address this, working out each step on the black box routine is simple enough.

Climbing the complexity ladder we may ask whether the above considerations apply also to *very* complex models, like ecological, geophysical or biological models. Does the knowledge still reside in the code? We believe the answer is negative. In our opinion the difference lies in the requirement of the model to simulate processes which are, in their original natural form, parallel, self-referential (as a result of feedback loops) and interactive. Since the sequential and 'closed' world of a Turing Machine prevents complex natural processes to be correctly modelled, the attempt to mimic those behaviours in an algorithm makes the problem code complex enough to make it impossible to remove the modeller from the exercise.

The end result is that, while in the square root and Fourier Transform algorithms we can differentiate the role of the input (a variable) from that of the algorithmic rules (fixed and 'correct'), this distinction becomes blurred in the case of a Complex model. To simplify the issue to the bare minimum, consider a user whose outcome of a Fourier transform (FFT) does not match the expected outcome. He/she can confidently rely that the mistake is in the chosen input (we assume here that the code is correct of course). We can do so because the abstract computation carried out by the FFT is 'right' within the formal system that defines it. Conversely, a user who finds an unexpected feature in the outcome of a complex code, will find it difficult to discriminate to what extent the input or the model are 'wrong'. In this case, the model is no more the implementation of a formally defined computation, but it mimicks of a poorly understood process. What should the user modify on order to generate the expected output? The input, the code or both? There is a sense to which this dilemma maps the unclear relation between input and computation in biological systems as described in [11].

Should we decide that the rules need to be changed, it is then essential to have a very clear understanding of the complex network of causal-effect relations in the complex model: that is, how the effect of a rule modification results in a cascade of effects, possibly several steps down the computation track. The understanding of this effect requires a painstaking exercise of debugging the model line by line, which can take months to achieve but whose necessity no expert modeller would deny. This suggests that modelling a complex system involves the specific role of the modeller in addition to the iterative inverse process we described in Section 3.

6 What makes a *modelling exercise* 'complex'?

Here we shift the question of complexity from the *model* to the *modelling exercise*, which involves a model and a modeller who uses the model to answer specific questions.[1]

[1] We avoid the question of the generation of the initial model. We suppose we start with a model and, at most, we may need to modify the model to improve its ability to simulate the process of interest. The problem of model generation is addressed in [2].

Formatted: Right: 0.63 cm

We propose casting a modelling exercise into a framework of asking and answering questions, in which questions are asked and answered at different levels. At a high level, it is clear that any modelling exercise aims to answer a broad question ("what factors lead to a market crash?", "what decision will improve the resilience of this ecosystem?", "what is the square root of 10?", "what is the optimal scheduling for this process?"). Some problems, which we define as 'simple', can be solved within required numerical precision[2] algorithmically in a single call of a numerical routine ("what is the square root of 10?"). Solving this problem does not involve an iterative inverse problem and the role of the modeller lies merely in posing the question.

Other, more complicated and possibly complex, problems cannot be solved in closed form, and require a numerical iterative search of a parameter space via the use of a forward model. This search involves an iteration of questions and answers and each run of a forward model effectively answers the question "is the scheduling arising from this input parameter set good?". Unlike the previous problem ("$\sqrt{10}$?"), this problem involves questions at two levels: the level of the global problem ("what is the optimal scheduling?"), and the lower level of individual question arising from testing a number of options in the parameter space ("is this scheduling good?"). Also, unlike the previous example, the role of the modeller here is two-fold. First, he/she has to pose the 'global' question. Second, he/she must pose the lower level questions in the iterative search. Traditionally, this is done by the modeller employing his/her judgment to evaluate which configuration in the parameter space should be tested next. Alternatively, it can be done automatically, by using a numerical optimisation routine [20], in which case the role of the user is to choose the suitable optimisation routine and tune it for the problem at hand (which can be seen as an inverse problem itself).

In the case of truly complex systems, we face inverse problems for which the quality of the outcome cannot be judged numerically [21]. Artistic problems fall in this category [22], but so do many scientific and engineering problems [23]. An example of such questions may be "does this agent-based model generate an emergence pattern?" or "does it display self-organisation?". Here, in addition to the questions mentioned in the previous paragraph, the user needs to judge (visually, for example) the global patterns arising from the forward model; expertise, experience, creativity and imagination are required to judge and evaluate patterns, since novelty and surprise are possible outcomes of this process.

Finally, we reach the top of the ladder with complex system problems like "what factors results in a market crash?" and "what decision will improve the resilience of this ecosystem?". In these problems, the modeller is required to carry out another crucial judgment in answering questions like "is the numerical model I am using appropriate for this task?", "does it include all important processes which affect my results?". If the answer to these questions is negative, the modeller needs to intervene by modifying or augmenting the model itself. It should be clear than neither of these tasks (judging the appropriateness of the model and improving it) can be carried out

[2] This requirement is essential, or we must admit that any problem leading to an irrational number would be unsolvable numerically.

Formatted: Right: 0.63 cm

by the model itself with current technologies [15] and both involve a creative intervention from the modeller.

In the above, answering a question is an algorithmic, that is, a mechanical process. The challenge lies in posing the question and, crucially, in being creative in judging what question should be asked and what is the best way to answer it. This challenge must be addressed by a human by using processes which are not necessarily algorithmic.

7 Modelling and problem knowledge

The above discussion is more than a philosophical exploration of the role of modelling in complex system science. It also has practical implications. In 'non-complex' problems, like the 'square root' or the 'Fourier transform', knowledge can be stored *in* the code and as such is easily transferable. In complex problems, the code is not the knowledge: It is a tool which the modeller employs in order to acquire knowledge. This gives rise to a number of practical questions.

1) How can this knowledge be transferred? Can the scientists/engineer who learnt and wrote the code pass the knowledge to the practitioner by simply delivering the code?
2) If not, what is the best way to communicate the knowledge and provide ways for future users of the model to acquire this knowledge quickly?
3) It is increasingly common that very large models are build by generations of visiting scientists, each contributing or improving specific modules. Does the problem-specific knowledge reside in any single user after this process? To what extent does a code built jointly represent a repository of commonly-built knowledge?
4) Finally, when complex models are employed to define and evaluate specific policies and management strategies, how will the rationale of the resulting decisions be communicated to the public?

These questions suggest a need for some novel approaches, whether formal or informal, that enable researchers and practitioners to describe and account for the overall modelling exercise, which includes the building of a model, its use, its improvement and the overall set of activities required for provision of information about a problem. The complexity of a modelling task necessarily needs to account for this overall process.

8 Discussion

In dealing with complexity, we have proposed switching the focus from the analysis of a model to the analysis of a modelling exercise and we have argued that most of the complexity lies in the task the modeller has to carry out, rather than merely in the computation involved. There are two main drawbacks to our analysis; first, we are not proposing a computable measure of such complexity and, second, our definition is not crisp; that is, it does not sharply differentiate a "complex" from a "non complex" problem. We discuss both drawbacks here.

Formatted: Right: 0.63 cm

Not all complexity measures available in the literature are computable. A classic example is Kolmogorov-Chaitin complexity measure [5] which is provably uncomputable [5], and for which we can find approximations from above at best and for which at best an approximation from above can be achieved. Other established measures, like the statistical complexity from the Computational Mechanics literature [24], are computable, but only in limited cases, and they require data to satisfy strict requirements for the computation to be carried out [6].

It can, of course, be argued that some sort of approximate measure could be devised. As some measures of complexity for example, are based on simple counting of the number of input parameters, the number of conflicting constraints or the dimensionality of the state space, we could try to count the number of question-answer sets or, possibly better, the number of question-answer levels involved. In this case, the "$\sqrt{10}$?" question would be level 1, the 'scheduling' problem would be level 2 and the 'ecosystem' problem level 3, with the subjective/qualitative inverse problem somewhere in between 2 and 3. At this stage, we do not feel this solution fully satisfactory and we endeavour to further address the problem in our future work.

Similar considerations apply to the crisp discrimination between complex and non-complex modelling exercises. Current information-theoretical measures of complexity provide a continuous transition between 'simple' and 'complex' models, not a crisp boundary. Admittedly, in our approach the problem is even more complicated and the distinction much fuzzier. Let's take the 'simple' "$\sqrt{10}$?" problem. In our discussion we assumed this can be solved by a simple routine call. This is correct only provided we do not concern ourselves of what the routine does. In reality, a square root is itself calculated with an iterative procedure, albeit one which can be coded fully algorithmically, given a limited required resolution [13]. Someone developed the algorithm[3], using his/her own creativity and ingenuity. So is it really correct to assign to it a 'level 1' complexity? A similar consideration applies to the scheduling problem discussed above; from a purely computational perspective a code may contain both a forward complex model and a numerical inverse routine, thereby performing an iterative search in a way that may be transparent to the user; an originally complex modelling task can be turned into a 'simple' black-box operation, if we do not concern ourselves with what the code actually does. As a further example, Kaltwasser et al. [25] attempted to reproduce algorithmically the subjective evaluation of problems for which a numerical cost function could not be easily designed; in principle, this would also turn the iterative, subjective problem in Section 6 into a 'level 1', 'simple' problem.

This observation is important and cannot be dismissed lightly. It suggests a new direction worth attention, in which complexity becomes a multi-criteria concept, which includes both the role of the modeller, the intrinsic complexity of the computation as well as historical and cultural developments. In particular, we suggest that a proper evaluation of complexity should account for at least four factors:

1) the role of the modeller as discussed in this work;

[3] Actually several algorithms to calculate a square root exist, see for example [13].

Formatted: Right: 0.63 cm

2) the computational complexity of the model, as in traditional information-theoretical literature [5,24];
3) the *amount* of computation carried out by both the model and the modeller, along the lines of what is called logical depth in the computational literature [26]; and
4) the creativity and ingenuity which lead to the development of the algorithm, which is now stored in the model and represents the actual knowledge which can be passed from one generation of modellers to the next, in the form of black-box tools.

The final point is surely the hardest to address and for which to devise a measure. It is nevertheless very important because it gives a criterion for which a once 'complex' problem becomes 'simple' as a result of cultural development. Traditionally, the scientific method tends to look with distrust at subjective measures, especially those which depend on cultural development. As discussed in [27] our inability to model and fully understand the concept of complexity may lie in the inability of current scientific method to address contextuality, as expressed most obviously in our struggle to model biological and ecological processes. Whether accounting directly for such contextuality may suggest a way forward is also something we will endeavour to analyse in our future work.

9 Conclusions

Most real-world problems involving computation need to be solved via an inverse process, involving iterative trial-and-error runs of a forward model. This is not different from how most Complex System Science problems are addressed, in which a modeller tests several input parameters to tune a model in order to generate a desired global outcome. Disregarding this observation, thereby focussing only on the computer model, leads to dismissing the role of the modeller in addressing a complex problem. We suggest that the complexity of a modelling task is related to the level and amount of intervention, in terms of non algorithmic ingenuity and creativity, which is a required by the modeller, and that the content of the model represents the amount of complex knowledge which is transmitted within the scientific community.

References

[1] Shalizi, C, Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata (2001), PhD Thesis, (http://www.cscs.umich.edu/~crshalizi/thesis/).
[2] Crutchfield, J. P. (1994) The Calculi of Emergence: Computation, Dynamics, and Induction. *Physica D* **75**: 11-54.
[3] Crutchfield, J., "Is Anything Ever New? Considering Emergence", in Complexity: Metaphors, Models, and Reality, G. Cowan, D. Pines, and D. Melzner, editors, SFI Series in the Sciences of Complexity XIX, Addison-Wesley, Redwood City (1994) 479-497.
[4] Pattee, H.H. (1997), "Causation, Control, and the Evolution of Complexity". Downward Causation, P.B. Anderson, P.V Christiansen, C. Emmeche, and N.O. Finnemann. In Press.
[5] Chaitin, G., 1997, The limits of mathematics : a course on information theory & limits of formal reasoning, Springer, New York.
[6] Shalizi, C. R. and Shalizi, K. L. (2004). Blind Construction of Optimal Nonlinear Recursive Predictors for Discrete Sequences. In "Uncertainty in Artificial Intelligence: Proceedings of the

Formatted: Right: 0.63 cm

Twentieth Conference", Arlington, Virginia, Max Chickering and Joseph Halpern, AUAI Press. p. 504-511

[7] Penrose, R., 1994, Shadows of the mind: a search for the missing science of consciousness, Oxford University Press, Oxford; New York.

[8] Aronson S., NP-complete Problems and Physical Reality, CM SIGACT News, March 2005 http://arxiv.org/abs/quant-ph/0502072

[9] Turing, MA 1936. On Computable Numbers, with an Application to the Entscheidungsproblem, Proceedings of the London Mathematical Society 42:230-265

[10] Ord, T., 'Hypercomputation: computing more than the turing machine', CoRR math.LO/0209332 (2002).

[11] Milner, R., 1993, Communications of the ACM archive, Volume 36 , Issue 1, 78 – 89.

[12] Penrose, R., 1989, The emperor's new mind : concerning computers, minds, and the laws of physics, Vintage: London, Melbourne

[13] Weisstein, Eric W. "Square Root." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/SquareRoot.html

[14] http://www.dar.csiro.au/css/whatis.shtm

[15] Boschetti & Gray , 2006, A Turing test for Emergence, in preparation.

[16] Boschetti & Gray , 2006, Emergence, Novelty and Computability, in preparation.

[17] Chaitin, Gregory J. 1974, Information-theoretic limitations of formal systems. Journal of the ACM, vol. 21, pp. 403-424

[18] Pearl, J., 2000, Causality: models, reasoning and inference, Cambridge, Mass., MIT Press, 2000, 384 pages.

[19] Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (2002) Numerical Recipes in Fortran 77: The Art of Scientific Computing, Cambridge University Press, Cambridge.

[20] Tarantola, A., 1987. Inverse Problem Theory. Elsevier, Amsterdam, 613pp..

[21] Takagi, H., 2001, Interactive evolutionary computation: Fusion of the capacities of EC: optimization and human evaluation: Proceedings of the IEEE, v. 89 (9), p. 1275–1296.

[22] Baluja, S., Pomerleau, D., and Jochem, T., 1994, Towards Automated Artificial Evolution for Computer-generated Images, Connection Science, 6, pp 325-354.

[23] F. Boschetti and L. Moresi, 2001, "Interactive Inversion in Geosciences", Geophysics, 64, 1226-1235.

[24] Crutchfield, J. P. and Young, K. (1989). Inferring Statistical Complexity. Physical Review Letters 63: 105-108.

[25] Kaltwasser P., Boschetti F., and Hornby P., 2004, Measure of similarity between geological sections accounting for subjective criteria, Computer & Geosciences, Vol 31/1 pp 29-34 .

[26] Bennett,CH; 1988, Logical Depth and Physical Complexity, in The Universal Turing Machine, A Half-Century Survey, Eds.Herken,R; , Oxford University Press, Oxford, pages 227-257

[27] K. Kitto, (2005), Gauging ALife: Emerging Complex Systems, in Recent Advances in Artificial Life. Advances in Natural Computation vol. 3, pp117-130, Eds. H.A. Abbas, et al. Singapore: World Scientific.