2nd International Workshop on

New Worlds of Computation 2011

Orléans, France, 23–24 May, 2011

Laboratoire d'Informatique Fondamentale d'Orléans

Editors
Jérôme Durand-Lose (chair)
Maxime Senot
Mathieu Chapelle

Université d'Orléans

Contents

Timetable

MONDAY 23 MAY

9h30	**Registration & Coffee**	
10h10	*Opening* Jérôme Durand-Lose	p. 3
10h15	*An optical approach to computation* Sama Goliaei	p. 20
11h00	*Beyond Determinism in Measurement-based Quantum Computation* Simon Perdrix	p. 26
11h45	*The legacy of Fourier, Poincaré and Einstein about relative computation* Françoise Chatelin	p. 8
12h30	**Lunch Break (and photo)**	
14h00	*Computing under vagueness* Apostolos Syropoulos	p. 35
15h00	*Recent discussions in cosmological computation* Mike Stannett	p. 33
15h30	*Relativity in mathematical descriptions of automatic computations* Yaroslav D. Sergeyev	p. 32
16h15	**Coffee Break**	
16h30	*Membrane computing — programming & verification* Marian Gheorghe	p. 13
17h00	*Polynomial interpretation of stream programs* Hugo Férée	p. 11
17h30	*About a message system for the tiles of the heptagrid of the hyperbolic plane* Maurice Margenstern	p. 25
18h00	**Discussions**	
20h00	**Banquet at the *Brin de Zinc* restaurant**	

TUESDAY 24 MAY

9h00	*A general framework for computability* Grégory Lafitte	p. 21
9h45	*Toward morphogenetic engineering: biological development as a new model of programmed self-organization* René Doursat	p. 10
10h30	*MGS: a declarative spatial computing programming language* Jean-Louis Giavitto & Antoine Spicher	p. 15
11h00	**Coffee Break**	
11h15	*Solving complex analytic differential equations in polynomial time over unbounded domain* Amaury Pouly	p. 27
11h45	*Equivalence between population protocols and Turing machines* Jonas Lefèvre	p. 23
12h15	**Lunch Break**	
14h00	*A generic and modular signal machine solving satisfiability problems* Maxime Senot	p. 29
14h45	*Points, Distances and Cellular Automata: Geometric and Spatial Algorithmics* Luidnel Maignan	p. 24
15h30	*Two Representations of Music Computed with a Spatial Programming Language* Louis Bigo	p. 6
16h00	**Closing**	

List of participants

Florent BECKER
florent.becker@univ-orleans.fr
LIFO, University of Orléans (France)
http://www.univ-orleans.fr/lifo/Members/Florent.Becker/

Louis BIGO
louis.bigo@ircam.fr
IRCAM, Paris (France)
http://www.lacl.fr/~lbigo/

Olivier BOURNEZ
bournez@lix.polytechnique.fr
LIX, École Polytechnique, Palaiseau (France)
http://www.lix.polytechnique.fr/~bournez/

Julien CERVELLE
julien.cervelle@polytechnique.edu
LACL, University of Créteil (France)
http://www-igm.univ-mlv.fr/~jcervell/

Mathieu CHAPELLE
mathieu.chapelle@univ-orleans.fr
LIFO, University of Orléans (France)
http://www.univ-orleans.fr/lifo/Members/Mathieu.Chapelle/

Françoise CHATELIN
chatelin@cerfacs.fr
CERFACS / Ceremaths, University of Toulouse 1 (France)
http://cerfacs.fr/~chatelin/

René DOURSAT
rene.doursat@polytechnique.edu
ISC-PIF and CREA-Lab, Paris (France)
http://doursat.free.fr/

Denys DUCHIER
denys.duchier@univ-orleans.fr
LIFO, University of Orléans (France)
http://www.univ-orleans.fr/lifo/Members/duchier/

Jérôme DURAND-LOSE
jerome.durand-lose@univ-orleans.fr
LIFO, University of Orléans (France)
http://www.univ-orleans.fr/lifo/Members/Jerome.Durand-Lose/

Christine EISENBEIS
christine.eisenbeis@inria.fr
INRIA Saclay Île-de-France (France)
http://www-rocq.inria.fr/~eisenbei/

Hugo FÉRÉE
hugo.feree@ens-lyon.fr
ENS Lyon (France)
http://perso.ens-lyon.fr/hugo.feree/

Mohammad Hadi FOROUGHMAND ARAABI
foroughmand@gmail.com
Tarbiat Modares University, Teheran (Iran)
http://bioinformatics.ut.ac.ir/index.php

Marian GHEORGHE
m.gheorghe@dcs.shef.ac.uk
University of Sheffield (UK)
http://staffwww.dcs.shef.ac.uk/people/M.Gheorghe/

Jean-Louis GIAVITTO
jean-louis.giavitto@ircam.fr
IRCAM, Paris (France)
http://repmus.ircam.fr/giavitto/

Fabien GIVORS
fabien.givors@lif.univ-mrs.fr
LIF, Marseille (France)
http://www.lif.univ-mrs.fr/spip.php?article435

Sama GOLIAEI
goliaei@gmail.com
Tarbiat Modares University, Teheran (Iran)
http://sites.google.com/site/goliaei/

Emmanuel HAINRY
hainry@loria.fr
LORIA / University Henri Poincaré, Nancy (France)
http://www.loria.fr/~hainry/

Gaétan HAINS
gaetan.hains@gmail.com
LACL, University of Créteil (France)
http://lacl.u-pec.fr/hains/

Emmanuel JEANDEL
emmanuel.jeandel@lif.univ-mrs.fr
LIF, Marseille (France)
http://www.lif.univ-mrs.fr/~ejeandel/

Grégory LAFITTE
gregory.lafitte@lif.univ-mrs.fr
LIF, Marseille (France)
http://www.lif.univ-mrs.fr/~lafitte/

Jonas LEFÈVRE
jlefevre@lix.polytechnique.fr
LIX, École Polytechnique, Palaiseau (France)
https://www.lix.polytechnique.fr/member/view/Jonas.Lefevre

Mathieu LIEDLOFF
mathieu.liedloff@univ-orleans.fr
LIFO, University of Orléans (France)
http://www.univ-orleans.fr/lifo/Members/Mathieu.Liedloff/

Luidnel MAIGNAN
luidnel.maignan@inria.fr
INRIA Saclay Île-de-France (France)
http://www-roc.inria.fr/~maignan/

Maurice MARGENSTERN
margens@univ-metz.fr
LITA, Université Paul Verlaine, Metz (France)
http://www.lita.univ-metz.fr/~margens/

Nicolas OLLINGER
nicolas.ollinger@lif.univ-mrs.fr
LIF, Marseille (France)
http://www.lif.univ-mrs.fr/~nollinge/

Simon PERDRIX
simon.perdrix@imag.fr
IMAG, Grenoble (France)
http://membres-lig.imag.fr/perdrix/

Amaury POULY
amaury.pouly@ens-lyon.fr
ENS Lyon (France)
http://perso.ens-lyon.fr/amaury.pouly/

Victor POUPET
victor.poupet@lif.univ-mrs.fr
LIF, Marseille (France)
http://www.lif.univ-mrs.fr/~vpoupet/

Maxime SENOT
maxime.senot@univ-orleans.fr
LIFO, University of Orléans (France)
http://www.univ-orleans.fr/lifo/Members/Maxime.Senot/

Yaroslav D. SERGEYEV
yaro@si.deis.unical.it
DEIS, Università della Calabria, Rende (Italy)
http://wwwinfo.deis.unical.it/~yaro/

Alexander SHEN
alexander.shen@lif.univ-mrs.fr
LIF, Marseille (France)
http://www.lif.univ-mrs.fr/spip.php?article200

Antoine SPICHER
antoine.spicher@u-pec.fr
LACL, University of Créteil (France)
http://lacl.univ-paris12.fr/spicher/

Mike STANNETT
m.stannett@dcs.shef.ac.uk
University of Sheffield (UK)
http://www.dcs.shef.ac.uk/~mps/

Apostolos SYROPOULOS
asyropoulos@gmail.com
Greek Molecular Computing Group, Xanthi (Greece)
http://obelix.ee.duth.gr/~apostolo/

Guillaume THEYSSIER
guillaume.theyssier@univ-savoie.fr
LAMA, University of Savoie, Le Bourget-du-Lac (France)
http://www.lama.univ-savoie.fr/~theyssier/

Ioan TODINCA
ioan.todinca@univ-orleans.fr
LIFO, University of Orléans (France)
http://www.univ-orleans.fr/lifo/Members/todinca/

Pierre VALARCHER
valarcher@univ-paris12.fr
LACL, University of Créteil (France)
http://lacl.univ-paris12.fr/valarcher

Pascal VANIER
pascal.vanier@lif.univ-mrs.fr
LIF, Marseille (France)
http://www.lif.univ-mrs.fr/spip.php?article437

Jean-Baptiste YUNÈS
jean-baptiste.yunes@liafa.jussieu.fr
LIAFA, University Paris 7 (France)
http://www.liafa.jussieu.fr/~yunes/

Preface

Jérôme Durand-Lose

LIFO, University of Orléans, France
jerome.durand-lose@univ-orleans.fr

These are the proceedings of the second international workshop *New Worlds of Computation*. The first workshop, which was held in 2009, was also hosted by the university of Orléans, France. It was a one day event and was attended by twenty people. This second workshop was a two day event (May 23–24, 2011) and twice as many people attended it.

This workshop addressed computation in the broadest understanding and away from classical computation mainlands. This includes, but is not limited to:

- analog computation,
- continuous computation,
- hybrid systems,
- computation on infinite structures (ordinal, linear orders),
- hypercomputation,
- infinite time computation,
- non-Euclidean spaces,
- non-standard approaches,
- optical computation,
- abstract geometrical computation,
- spatial computing,
- fuzzy computation,
- cellular automata,
- collision based, quantum, DNA, membrane computing...

It aims at gathering, but is not restricted to, people interested in the conferences *Unconventional Computation* (UC) *Computability in Europe* (CiE) *Machines, Computations and Universality* (MCU) and the *Hypercomputation Research Network* (HyperNet).

Aways from mainstreams, most of the results are somehow singular. We want to offer the people in the field an opportunity to identify one another, exchange knowledge and let emerge common perspectives and problematics. We want this workshop series to foster discussions and collaborations.

After the first workshop, we made a special issue of the *International Journal of Unconventional Computing* (actually under press). There should also be a special issue devoted to the present edition of NWC.

We would like to thank the Laboratoire d'Informatique Fondamentale d'Orléans (LIFO, joint computer science research laboratory of the Université d'Orléans and ENSI de Bourges) for its support.

I would like to thank cheerfully Mathieu Chapelle and Maxime Senot who managed all the organization.

J. Durand-lose

The physical Church-Turing thesis and the principles of quantum theory

Pablo Arrighi[1,2], Gilles Dowek[3]

[1]*École normale supérieure de Lyon, LIP, 46 allée d'Italie, 69008 Lyon, France*
[2]*Université de Grenoble, LIG, 220 rue de la chimie, 38400 Saint-Martin-d'Hères, France*
[3]*INRIA, Paris, France*
parrighi@imag.fr, gilles.dowek@inria.fr

Abstract

Notoriously, quantum computation shatters complexity theory, but is innocuous to computability theory [5]. Yet several works have shown how quantum theory as it stands could breach the physical Church-Turing thesis [10, 11]. We draw a clear line as to when this is the case, in a way that is inspired by Gandy [7]. Gandy formulates postulates about physics, such as homogeneity of space and time, bounded density and velocity of information — and proves that the physical Church-Turing thesis is a consequence of these postulates. We provide a quantum version of the theorem. Thus this approach exhibits a formal non-trivial interplay between theoretical physics symmetries and computability assumptions.

The Church-Turing thesis. The physical Church-Turing thesis states that any function that can be computed by a physical system can be computed by a Turing Machine. There are many mathematical functions that cannot be computed on a Turing Machine (the halting function $h : \mathbb{N} \to \{0,1\}$ that decides whether the i^{th} Turing Machine halts, the function that decides whether a multivariate polynomial has integer solutions, etc.). Therefore, the physical Church-Turing thesis is a strong statement of belief about the limits of both physics and computation.

The quantum computing challenge. The shift from classical to quantum computers challenges the notion of complexity: some functions can be computed faster on a quantum computer than on a classical one. But, as noticed by Deutsch [5], it does not challenge the physical Church-Turing thesis itself: a quantum computer can always be (very inefficiently) simulated by pen and paper, through matrix multiplications. Therefore, what they compute can be computed classically.

The quantum theoretical challenge. Yet several researchers [8, 10, 11] have pointed out that Quantum theory does not forbid, in principle, that some evolutions would break the physical Church-Turing thesis. Indeed, if one follows the postulates by the book, the only limitation upon evolutions is that they be unitary operators. Then, according to Nielsen's argument [11], it suffices to consider the unitary operator $U = \sum |i, h(i) \oplus b\rangle\langle i, b|$, with i over integers and b over $\{0,1\}$, to have a counter-example.
The paradox between Deutsch's argument and Nielsen's argument is only an apparent one; both arguments are valid; the former applies specifically to Quantum Turing Machines, the latter applies to full-blown quantum theory. Nevertheless, this leaves us in a unsatisfactory situation: if the point about the Quantum Turing Machine was to capture Quantum theory's computational power, then it falls short of this aim, and needs to be amended! Unless Quantum theory itself needs to be amended, and its computational power brought down to that of the Quantum Turing Machine?

Computable quantum theory. Quantum theory evolutions are about possibly infinite-dimensional unitary operators and not just matrices — for a good reason: even the state space of a particle on a line is infinite-dimensional. Can this fact be reconciled with the physical Church-Turing thesis, at least at the theoretical-level? Mathematically speaking, can we allow for all those unitary operators we need for good physical reasons *and at the same time* forbid the above $U = \sum |i, h(i) \oplus b\rangle\langle i, b|$, but for good physical reasons as well? These are the sort of questions raised by Nielsen [11], who calls for a programme of finding the non-ad-hoc, natural limitations that one could place upon Quantum theory in order to make it computable: we embark upon this programme of a computable Quantum theory.

Our result. The idea that physically motivated limitations lead to the physical Church-Turing thesis has, in fact, already been investigated by Gandy [7]. Although some similarities exist, Gandy's proof of the Church-Turing thesis serves different goals from those of the proof by Dershowitz and Gurevich [4], as it is based not on an axiomatic notion of algorithm, but on physical hypotheses. In Gandy's proof, one finds the important idea that causality (i.e. bounded velocity of information), together with finite density of information, could be the root cause of computability (i.e. the physical Church-Turing thesis). More generally, Gandy provides a methodology whereby hypotheses about the real world have an impact upon the physical Church-Turing thesis; an idea which can be transposed to other settings: we transpose it to Quantum theory. A formal statement and a proof of the result are available in [1].

Future work. This result clarifies when it is the case that Quantum theory evolutions could break the physical Church-Turing thesis or not; a series of examples shows that it suffices that one of the above hypotheses be

dropped. This draws the line between the positive result of [5] and the negative results of [8, 10, 11]. Because these hypotheses are physically motivated, this is a step along Nielsen's programme of a computable Quantum theory. Further work could be done along this direction by introducing a notion of 'reasonable measurement' [12], or investigating the continuous-time picture as started by [13, 14]. Prior to that however this work raises deeper questions: Is the bounded density of information really compatible with modern physics? For instance, can we really divide up space into pieces under this condition, without then breaking further symmetries such as isotropy?

Bigger picture. The question of the robustness of the physical Church-Turing thesis is certainly intriguing; but it is hard to refute, and fundamentally contingent upon the underlying physical theory that one is willing to consider. For instance in the General Relativity context a series of paper explain how 'hypercomputation' might be possible in the presence of certain space-times [6, 9]. Beyond this sole question however, we find that it essential to exhibit the formal relationships that exist between the important hypotheses that we can make about our world. Even if some of these hypotheses cannot be refuted, whenever some can be related to one another at the theoretical level, then one can exclude the inconsistent scenarios.

Acknowledgments

We are particularly thankful to Vincent Nesme and Reinhard Werner, since their paper with one of the authors [2, 3] contains a rather central ingredient to our proof. We would like to thank Alexei Grimbaum and Zizhu Wang for several comments, as well as Simon Perdrix, Mehdi Mhalla and Philippe Jorrand.

References

[1] P. Arrighi and G. Dowek. The physical Church-Turing thesis and the principles of quantum theory. Pre-print arXiv:1102.1612, 2011.

[2] P. Arrighi, V. Nesme, and R. Werner. Unitarity plus causality implies localizability. *QIP 2010, ArXiv preprint: arXiv:0711.3975*, 2010.

[3] P. Arrighi, V. Nesme, and R. Werner. Unitarity plus causality implies localizability (Full version). *Journal of Computer and System Sciences*, 2010.

[4] N. Dershowitz and Y. Gurevich. A natural axiomatization of the computability and proof of Church's thesis. *The Bulletin of Symbolic Logic*, 14(3), 2008.

[5] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences (1934-1990)*, 400(1818):97–117, 1985.

[6] G. Etesi and I. Németi. Non-Turing computations via Malament–Hogarth space-times. *International Journal of Theoretical Physics*, 41(2):341–370, 2002.

[7] R. Gandy. Church's thesis and principles for mechanisms. In *The Kleene Symposium*, Amsterdam, 1980. North-Holland Publishing Company.

[8] M. Gu, C. Weedbrook, A. Perales, and M.A. Nielsen. More really is different. *Physica D: Nonlinear Phenomena*, 238(9-10):835–839, 2009.

[9] M. Hogarth. Non-Turing computers and non-Turing computability. In *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, volume 1994, pages 126–138. JSTOR, 1994.

[10] T.D. Kieu. Computing the non-computable. *Contemporary Physics*, 44(1):51–71, 2003.

[11] M. A. Nielsen. Computable functions, quantum measurements, and quantum dynamics. *Phys. Rev. Lett.*, 79(15):2915–2918, Oct 1997.

[12] A. Peres. *Quantum theory: concepts and methods*. Kluwer Academic Publishers, 1993.

[13] W. D. Smith. Church's thesis meets quantum mechanics. Available on CiteSeerX doi=10.1.1.49.7057, 1999.

[14] R. Werner and V. Sholz. Church-Turing thesis and quantum mechanics. Private communication., 2010.

Two Representations of Music Computed with a Spatial Programming Language

Louis Bigo[1,2], Antoine Spicher[2], Olivier Michel[2]

[1]*IRCAM, 1 place Igor Stravinsky, 75004 Paris, France*
[2]*LACL, Université Paris-Est, 61 rue du Général de Gaulle, 94010 Créteil, France*
louis.bigo@ircam.fr, {antoine.spicher,olivier.michel}@u-pec.fr

Abstract

In this long abstract, we show how some musical objects and some algorithmic problems arising in musical theory can be rephrased in spatial terms. This leads to the development of meaningful spatial representations and of efficient spatial programs. The corresponding developments have been implemented in MGS, a rule-based spatial programming language.

Spatial computing has proven to be a fruitful paradigm for the (re-)design of algorithms tackling problems embedded in space or having a spatial extension. While music can be seen according to various viewpoints, we propose in this paper a few studies of paradigmatic theoretical music problems from a spatial computing perspective.

Musical theory has received much interest by mathematicians that have found a field to develop algebraic methods to solve, in a very elegant way, problems of enumeration and classification of musical structures. Indeed, the algebraic nature of many musical formalizations has been very early assessed: from the equal temperament to combinatorial properties of the integral serialism and modal techniques. However, the topological properties of those objects have been rarely considered (nevertheless, see [M+02, Tym06]).

Spatial computing distinguishes spaces in computations either as a *resource* or a *result*. In the following, we propose to illustrate each of these two points of view in the field of theoretical music.

The MGS Programming Language. MGS is an experimental programming language (see [GM01b, Gia03]) investigating the use of rules to compute with spatial data structures. MGS concepts rely on well established notions in algebraic topology [Mun84] and have been validated through applications in autonomic computing and in the modeling of complex dynamical systems.

In MGS, all data structures are unified under the notion of *topological collection* [GM02]. A topological collection is a *cellular complex* labeled with arbitrary values. The cellular complex acts as a container and the values as the elements of the data structure. A *cellular complex* is a space built by gluing together more elementary spaces called *cells*. A cell is the abstraction of a space with some *dimension*. In a cellular complex, cells are organized following the *incidence relationship*.

Topological collections are transformed using sets of rules called *transformations*. A rule is a pair `pattern => expression`. When a rule is applied on a topological collection, the sub-collections matching with the `pattern` are replaced by the topological collection computed by the evaluation of `expression`. There exists several ways to control the application of a set of rules on a collection but these details are not necessary for the comprehension of the work presented here. A formal presentation of the rewriting mechanism is given in [SMG10].

Space for Musical Representation. In this first application, we illustrate how a musical piece can be seen as a computation taking place on specific *resource* space. We focus here on the notion of *Tonnetz* [Coh97, Choa, Chob] and its use for the characterization of tonal music. We show that the topological collection of GBF (Group Based Fields), when carefully defined, makes it possible to describe chords progressions. A GBF [GM01a] is a topological collection that generalizes the usual notion of array, by labeling a cellular complex generated by a mathematical group.

The Neo-Riemannian representation (named after Hugo Riemann, a musicologist) has the ambition to show proximity between notes in terms of consonance instead of chromatism as in the traditional staff. We focus on the strong algebraic strucutre of musical intervals [Lew87] and suggest a spatial representation of a consonance relationship based on Cayley graphs. The properties of the resulting tone networks depend on the definition of consonance. A musical sequence represented in different networks exhibits more or less regularities. We are interested in formalizing these regularities and understanding if a network can characterize signature of a musical piece or style. More, the exploration of tone networks associated to these representations brings elements on scales generation and instruments conception.

Space as Musical Representation. A mathematical analysis of music generally relies on the definition of a *model*, that is a mathematical object, whose specific properties describe faithfully some musical characteristics. In the previous case, we followed this approach by defining topological spaces used to exhibit some musical properties. We focus here on the process used to build such a mathematical model of music. Thus, we consider here spaces as *results* of a spatial computing. In particular, we describe a self-assembly mechanism used to define a spatial representation of a tonality based on triadic chords [M+02]. Then, we propose to use this mechanism for the study of a tonality representation based on four-note chords and the analysis of a chord series to F. Chopin.

More detailed applications and other uses of spacial computing for musical analysis are described in [BSM10, BGS11].

Acknowledgements. The authors are very grateful to M. Andreatta, C. Agon, G.Assayag and J-L. Giavitto from the REPMUS team at IRCAM, J-M. Chouvel and M. Malt for endless fruitful discussions. This research is supported in part by the IRCAM and the University Paris Est-Créteil Val de Marne.

References

[BGS11] Louis Bigo, Jean-Louis Giavitto, and Antoine Spicher. Building spaces for musical objects. In *Mathematics and Computation in Music*, Communications in Computer and Information Science. Springer, 2011.

[BSM10] Louis Bigo, Antoine Spicher, and Olivier Michel. Spatial programming for music representation and analysis. In *Spatial Computing Workshop 2010*, Budapest, September 2010.

[Choa] Jean-Marc Chouvel. Analyser l'harmonie – aux frontières de la tonalité.

[Chob] Jean-Marc Chouvel. Au-delà du système tonal.

[Coh97] Richard Cohn. Neo-riemannian operations, parsimonious trichords, and their "tonnetz" representations. *Journal of Music Theory*, 41(1):pp. 1–66, 1997.

[Gia03] J.-L. Giavitto. Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In *Rewriting Technics and Applications (RTA'03)*, volume LNCS 2706 of *LNCS*, pages 208 – 233, Valencia, June 2003. Springer.

[GM01a] J.-L. Giavitto and O. Michel. Declarative definition of group indexed data structures and approximation of their domains. In *Proceedings of the 3nd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP-01)*. ACM Press, September 2001.

[GM01b] Jean-Louis Giavitto and Olivier Michel. MGS: a rule-based programming language for complex objects and collections. In Mark van den Brand and Rakesh Verma, editors, *Electronic Notes in Theoretical Computer Science*, volume 59. Elsevier Science Publishers, 2001.

[GM02] J.-L. Giavitto and O. Michel. Data structure as topological spaces. In *Proceedings of the 3nd International Conference on Unconventional Models of Computation UMC02*, volume 2509, pages 137–150, Himeji, Japan, October 2002. Lecture Notes in Computer Science.

[Lew87] David Lewin. *Generalized musical Intervals and Transformations*. Yale University Press, 1987.

[M+02] G. Mazzola et al. *The topos of music: geometric logic of concepts, theory, and performance*. Birkhäuser, 2002.

[Mun84] James Munkres. *Elements of Algebraic Topology*. Addison-Wesley, 1984.

[SMG10] Antoine Spicher, Olivier Michel, and Jean-Louis Giavitto. Declarative mesh subdivision using topological rewriting in mgs. In *Int. Conf. on Graph Transformations (ICGT) 2010*, volume 6372 of *LNCS*, pages 298–313, September 2010.

[Tym06] D. Tymoczko. The geometry of musical chords. *Science*, 313(5783):72, 2006.

The legacy of Fourier, Poincaré and Einstein about Relative Computation

Françoise Chatelin

Ceremath, Université Toulouse 1
CERFACS, 42 avenue G. Coriolis, 31057 Toulouse Cedex 1, France
chatelin@cerfacs.fr

Hypercomputation, that is nonlinear computation in real multiplicative Dickson algebras $A_k \cong \mathbb{R}^{2^k}$, is developed in (Chatelin 2011 a).For $k \geq 2$ (resp. $k \geq 3$) multiplication is not commutative (resp. associative). However addition remains both associative and commutative.

The situation changes in an essential way when computation is merely linear but there exists a *relator* which rules the way any two numbers are to be *added*. This kind of relating computation will be defined in precise terms. It includes the special case of an *explicit metric reference* consisting of a positive *finite* number λ, $0 < \lambda < \infty$. The classical structure of an abelian additive group is weakened by considering an addition whose commutativity and associativity are controlled by the relator. A physically meaningful example was provided a century ago by 3D-Special Relativity (Einstein) where the role of λ as a metric reference is played by c, the speed of light, and the relator is a plane rotation.

In the early days of Special Relativity it was soon recognised that hyperbolic geometry underlies Einstein's law of addition for admissible velocities (Varičak 1910, Borel 1914) creating the relativistic effect known today as *Thomas precession* (Silberstein 1914, Thomas 1926).

But, despite Maxwell's valiant efforts (Maxwell 1871), Hamilton's noncommutative $\times$ of 4-vectors was still unacceptable for most scientists at the dawn of the 20th century. Therefore Einstein's noncommutative $+$ of 3-vectors (representing relativistically admissible velocities) was fully inconceivable: Einstein's vision was much ahead of its time! A more understandable version of Special Relativity was conceived by Minkowski in 1907, by dressing up as physical concepts the Lorentz transformations in the field of quaternions $\mathbb{H}$ which had been introduced by Poincaré in 1905, see (Walter 1999, Auffray 2005). This is the version adopted until today in physics textbooks.

Einstein's intuition was left dormant for 80 some years until it was brought back to a new mathematical life in the seminal paper (Ungar 1988). During 20 years, Ungar has crafted an algebraic language for hyperbolic geometry which sheds a *natural light* on the physical theories of Special Relativity and Quantum Computation (Ungar 2008). We export some of Ungar's tools developed for mathematical physics into mathematical computation in a *relating* context. The reward of the shift of focus from physics to computation is to gain insight about the *geometric* ways by which information can be organically processed in the mind when *relation* prevails. This processing exemplifies the computational thesis posited in (Chatelin 2011 a,c).

Then we show how the seminal ideas of Fourier, Poincaré and Einstein blend together harmoniously to explain many features of computation in Nature and in the human mind.

1) The Fourier analysis of complex signals leads to the Fourier transform whose complex spectrum lies on the unit circle, with 4 eigenvalues $+1, -1, i, -i$ which are the 4 units of the Gaussian ring of complex integers.

2) The Poincaré approach to relativity bears on the Lorentz transformations in the algebra $\mathbb{H}$ of quaternions, using a noncommutative $\times$. Relative significant computation evolves from $\mathbb{H}$ to $\mathbb{G}^2$, where $\mathbb{G}$ is the algebra of octonions.

3) The Einstein perspective on relativity is based on a noncommutative $\oplus$, yielding a geometric 2-fold information potential with 2 types of geodesics. The potential lies in a $\mathbb{R}^3$-framedmetric cloth, a non-euclidean space which is a computational construct exhibiting some features attributed to either hyperbolic or elliptic axiomatically defined geometries. By mixing these 3 views with the quadratic logistic iteration, we are able to propose an algorithmic mechanism which can serve as a working model for the law of causality that we experience in Nature and mind (Chatelin 2011 d).

References

[1] Auffray J.P. (2005). **Einstein et Poincaré. Sur les traces de la relativité**, 2ème édition, Le Pommier, Paris.

[2] Atlan H. (2011). **Le vivant post-génomique, ou qu'est-ce que l'auto-organisation?**, Odile Jacob, Paris.

[3] Baez J. (2001). The octonions, *Bull. AMS* , **39**, 145-205.

[4] Borel E. (1913). La cinématique dans la théorie de la relativité, *CR Acad. Sc. Paris* **157**, 703-705.

[5] Borel E. (1914). **Introduction géométrique à quelques théories physiques**, Gauthier-Villars, Paris.

[6] Calude C., Chatelin F. (2010). A dialogue about Qualitative Computing, *Bull. EATCS* **101**, 29-41.
[7] Chatelin F. (2011 a). **Qualitative Computing: a computational journey into nonlinearity**, World Scientific, Singapore.
[8] Chatelin F. (2011 b). Polymorphic Information Processing in weaving computation: An approach through cloth geometry, Cerfacs Tech. Rep. TR/PA/11/27.
[9] Chatelin F. (2011 c). A computational journey into the mind, *Natural Computing*, To appear.
[10] Chatelin F. (2011 d). On relating computation, Cerfacs Tech. Rep. TR/PA/11/?.
[11] Poincaré H. (1905). Sur la dynamique de l'electron, *Rend. Circ. Matem. Palermo*, **21**, 17-175.
[12] Thomas L. (1926). The motion of the spinning electron, *Nature* **117**, 514.
[13] Ungar A.A. (1988). The Thomas rotation formalism underlying a nonassociative group structure for relativistic velocities, *Appl. Math. Lett* **1**, 403-405.
[14] Ungar A.A. (2008). **Analytic hyperbolic geometry and Albert Einstein's special theory of relativity**, World Scientific, Singapore.

Toward Morphogenetic Engineering: Biological development as a new model of programmed self-organization

René Doursat[1,2]

[1]*Institut des Systèmes Complexes, Paris Ile-de-France (ISC-PIF), France*
[2]*Centre de Recherche en Épistémologie Appliquée (CREA), École Polytechnique and CNRS, Paris, France*
rene.doursat@polytechnique.edu

Multicellular organisms are rather unique examples of natural systems that exhibit both self-organization and a strong architecture. Similarly to other bio-inspired methodologies in which principles derived from neural networks, genetics or ant colonies are routinely used in machine learning, stochastic optimization or combinatorial search problems, can we also export the precise self-formation capabilities of biological development to a new generation of algorithmic methods and technological systems?

I present here two related studies in "Embryomorphic Engineering" (an instance of a broader field called "Morphogenetic Engineering"). First, a 2-D/3-D multi-agent model of biological development, based on virtual gene regulation networks, chemical gradients and mechanical forces which can be applied to the self-aggregation or self-assembly of robotic swarms into specific and reproducible superstructures. Second, a related N-D multi-agent model of self-construction of complex but precise graph topologies by "programmed attachment", based on dynamical opening of node ports, spread of gradients and creation of links with potential applications in socio-technical systems composed of a myriad of peer-to-peer mobile devices and human users. In all cases, the challenge is to design, *e.g.* through an evolutionary search, the proper set of local rules followed by each agent of a complex system on how to interact with the other agents and the environment in order to produce global functional architectures.

References

[1] René Doursat. Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering. In R.P. Würtz, editor, *Organic Computing*, chapter 8, pages 167–200. Springer-Verlag, 2008.

[2] René Doursat. Programmable architectures that are complex and self-organized: From morphogenesis to engineering. In S. Bullock, J. Noble, R. Watson, and M.A. Bedau, editors, *In Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems (Alife XI)*, number 6506 in LNCS, pages 181–188. MIT Press, August, 5-8 2008.

[3] René Doursat. Facilitating evolutionary innovation by developmental modularity and variability. Generative and Developmental Systems track (GDS '09). In F. Rothlauf, editor, *Proceedings of the 18th Genetic and Evolutionary Computation Conference (GECCO '09)*, number 6506 in LNCS, pages 683–690. ACM, July, 8-12 2009.

[4] René Doursat and M. Ulieru. Emergent engineering for the management of complex situations. In A. Manzalini, editor, *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems (Autonomics '08)*, number 14 in ICST, pages 291–303, September, 23-25 2008.

Polynomial interpretation of stream programs

Hugo Férée[1], Emmanuel Hainry[2,5], Mathieu Hoyrup[3,5], Romain Péchoux[4,5]

[1]*ENS Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France*
[2]*Université Henri Poincaré, Nancy-Université, France*
[3]*INRIA Nancy – Grand Est, Villers-lès-Nancy, France*
[4]*Université Nancy 2, Nancy-Université, France*
[5]*LORIA, B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France*
hugo.feree@ens-lyon.fr,{hainry,hoyrup,pechoux}@loria.fr

Stream languages including lazy functional languages like Haskell allows the programmer to represent functionals, functions over functions. From this perspective, they can be understood as a way to simulate type 2 functions. There are many works in the literature that study computability and (polynomial time) complexity of such functions [5, 14]. The implicit computational complexity (ICC) community has proposed characterizations of such complexity classes using function algebra and types [8, 9, 15]. These results are reminiscent of former characterizations of type 1 polynomial time functions [2, 4, 12] that led to other ICC works using polynomial interpretations.

Polynomial interpretations [11, 13] are a well-known tool used in the termination analysis of first order functional programs for several decades. Variants, like sup-interpretations and quasi-interpretations [3], which allow the programmer to perform program complexity analysis have emerged in the last ten years. One of their drawbacks is that such tools are restricted to first order programs on inductive data types. The paper [7] was a first attempt to adapt such a tool to co-inductive data types and, more precisely, to stream programs. We provide a second order variation of this interpretation methodology that fits to stream computation by replacing usual polynomials with type 2 polynomials (basically polynomials with function variables) to interprete programs.

It allows us to characterize exactly the set of functions computable in polynomial time by Unary Oracle Turing Machine (UOTM), that is functions computable by machines using oracles where the oracle has only unary input. It can also be used in order to characterize the set of functions computable in polynomial time by Oracle Turing Machine (OTM), that is shown to be equivalent to the BFF algebra in [9].

The first characterization has two advantages. First, it gives a new and intuitive notion of stream computational complexity in terms of Turing Machine. Second, it shows that this natural class can be easily captured using an adaptation of the interpretation tool. Using this tool we can analyze functions of this class in an easier way (based on the premise that it is practically easier to write a first order functional program on streams than the corresponding Turing Machine). The drawback is that the tool suffers from the same problem as type 1 polynomial interpretation: the difficulty to automatically synthesize the interpretation of a given program (see [1]).

The latter characterization gives a natural view of a well-know complexity class BFF, just by changing the interpretation domain, that is allowing exponential arguments for type 2 variables. It illustrates that the first characterization on UOTM is natural and flexible because it can be easily adapted to other complexity classes. Finally, it can be interpreted as a negative result showing that the BFF class, whose purpose is to study functions from $\mathbb{N} \to \mathbb{N}$, is surprisingly not well-suited to describe stream polynomial complexity.

We also go one step further showing that these tools can be adapted to characterize the complexity of functions computing over reals defined in Recursive Analysis [10], since real numbers can be seen as streams of rational numbers and the polynomial time complexity defined using UOTMs matches the common definition of polynomial time complexity for real functions.

Details of this work can be found in [6].

References

[1] Roberto M. Amadio. Synthesis of max-plus quasi-interpretations. *Fundamenta Informaticae*, 65(1):29–60, 2005.

[2] Stephen Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational complexity*, 2(2):97–110, 1992.

[3] Guillaume Bonfante, Jean-Yves Marion, and Jean-Yves Moyen. Quasi-interpretations. *Theor. Comput. Sci.. to appear.*

[4] Alan Cobham. The Intrinsic Computational Difficulty of Functions. In *Logic, methodology and philosophy of science III: proceedings of the Third International Congress for Logic, Methodology and Philosophy of Science, Amsterdam 1967*, page 24. North-Holland Pub. Co., 1965.

[5] Robert L. Constable. Type two computational complexity. In *Proc. 5th annual ACM STOC*, pages 108–121, 1973.

[6] Hugo Férée, Emmanuel Hainry, Mathieu Hoyrup, and Romain Péchoux. Interpretation of stream programs: characterizing type 2 polynomial time complexity. In O. Cheong, K-Y. Chwa, and K. Park, editors, *21st*

International Symposium on Algorithms and Computation (ISAAC '10), number 6506 in LNCS, pages 291–303. Springer, 2010.

[7] M. Gaboardi and R. Péchoux. Upper Bounds on Stream I/O Using Semantic Interpretations. In *Computer Science Logic*, pages 271–286. Springer, 2009.

[8] Robert J. Irwin, James S. Royer, and Bruce M. Kapron. On characterizations of the basic feasible functionals (Part I). *J. Funct. Program.*, 11(1):117–153, 2001.

[9] Bruce M. Kapron and Stephen A. Cook. A new characterization of type-2 feasibility. *SIAM Journal on Computing*, 25(1):117–132, 1996.

[10] Ker-I Ko. *Complexity theory of real functions*. Birkhauser Boston Inc. Cambridge, MA, USA, 1991.

[11] D.S. Lankford. On proving term rewriting systems are noetherien. tech. rep., 1979.

[12] Daniel Leivant and Jean-Yves Marion. Lambda calculus characterizations of poly-time. *Typed Lambda Calculi and Applications*, pages 274–288, 1993.

[13] Z. Manna and S. Ness. On the termination of Markov algorithms. In *Third hawaii international conference on system science*, pages 789–792, 1970.

[14] Kurt Mehlhorn. Polynomial and abstract subrecursive classes. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 96–109. ACM New York, NY, USA, 1974.

[15] Anil Seth. Turing machine characterizations of feasible functionals of all finite types. *Feasible Mathematics II*, pages 407–428, 1995.

[16] Klaus Weihrauch. *Computable analysis: an introduction*. Springer Verlag, 2000.

Membrane Computing – Programming and Verification

Marian GHEORGHE[1,2]

[1]*Department of Computer Science, University of Sheffield*
Regent Court, Portobello Street, Sheffield S1 4DP, UK
[2]*Department of Computer Science, University of Pitesti*
Str Targu din Vale 1, 110040 Pitesti, Romania
M.Gheorghe@dcs.shef.ac.uk

1 Introduction

Membrane systems (also called P systems), introduced in [12], represent a (relatively new) computational model inspired by the structure, composition and functions of the living cell or more complex biological entities like tissues, organs and organisms. In the last ten years the area of membrane computing has been under intensive research and development; many aspects of this computational paradigm have been considered: computability and complexity of many different variants [13], various applications to modelling natural or engineered systems, connections with other computational models [6]. Some more recent investigations in this area are concerned with programming approaches on P systems ([7], [16]), formal semantics ([5], [1], [11]) and formal verification and testing of these systems ([9], [10]). Formal verification has been utilised for various variants of membrane systems by means of rewriting logic and the Maude tool [1] or, for stochastic systems [3], probabilistic temporal logics and PRISM [8].

Membrane systems have been used to specify various sorting algorithms and different parallel architectures [4], measure complexity aspects related to solving NP-complete problems [15] and describe synchronization and other distributed problems [2].

In this presentation we will focus on discussing some facts regarding the programmability of membrane systems by referring to a set of poweful basic primitives for rewriting and communication between compartments, adequate data structures, a flexible modular structure and appropriate verification mechanisms.

2 Basic features

Some examples, including the synchronization problem and a solution to an NP-complete problem, will illustrate the approach. In these cases membrane systems with rewriting and communication rules are utilised and additional features like broadcasting facilities or electrical charges assiciated with compartments are considered.

In order to develop adequate formal verification procedures a specific class of Kripke structures have been associated with certain classes of P systems [10]. For a *transformation-communication P system*, as presented in Definition 1, a specific Kripke structure (see Definition 2) is defined.

Definition 1. *A* P system *is a tuple* $\Pi = (V, \mu, w_1, ..., w_n, R_1, ..., R_n)$*, where* V *is a finite set, called* alphabet*;* μ *defines the membrane structure, i.e., the hierarchical arrangement of* n *compartments called* regions *delimited by* membranes*; the membranes and regions are identified by integers 1 to* n*;* w_i*,* $1 \leq i \leq n$*, represents the initial multiset occurring in region* i*;* R_i*,* $1 \leq i \leq n$*, denotes the set of processing rules applied in region* i*.*

In this presentation another variant of P system is considered, namely the *P systems with electrical charges*. This is a simplification of the usual variant occurring in the literature [14]. Each compartment has a specific electrical charge $(+, -, 0)$ which can be changed by a communication rule. The set of electrical charges is denoted by H. The set of rules contains the following types of rules:

- $[u \to v]_b^h$;
- $u[]_b^{h_1} \to [v]_b^{h_2}$;
- $[u]_b^{h_1} \to v[]_b^{h_2}$;

where b indicates a compartment and $h, h_1, h_2 \in H$. The rules are applied in the normal way; for more details see [14].

In the sequel we will consider transformation-communication P systems using maximal parallelism, which is the normal semantics for such systems and asynchronous behaviour for P systems with electrical charges.

Definition 2. *A Kripke structure over a set of atomic propositions* AP *is a four tuple* $M = (S, H, I, L)$*, where* S *is a finite set of states;* $I \subseteq S$ *is a set of initial states;* $H \subseteq S \times S$ *is a transition relation that must be left-total, that is, for every state* $s \in S$ *there is a state* $s' \in S$ *such that* $(s, s') \in H$*;* $L : S \longrightarrow 2^{AP}$ *is an interpretation function, that labels each state with the set of atomic propositions true in that state.*

In general, the Kripke structure representation of a system consists of sets of values associated to the system variables. Assuming that $var_1, \ldots, var_n$ are the system variables and Val_i the set of values for var_i, with val_i a value from Val_i, $1 \leq i \leq n$, we can introduce the states of the system as

$$S = \{(val_1, \ldots, val_n) \mid val_1 \in Val_1, \ldots, val_n \in Val_n\}.$$

The set of atomic predicates are given by $AP = \{(var_i = val_i) \mid 1 \leq i \leq n, val_i \in Val_i\}$.

Based on this approach P system specifications have been mapped into specific Kripke structures associated with certain model checkers like NuSMV and SPIN. For such specifications certain properties have been checked and some testing techniques based on these formal specifications have been then developed in order to check the vailidy of the specifications. On longer term we aim to develop more appropriate and coherent specification languages based on the membrane computing paradigm, adequate temporal and spatial logics to express certain properties of the systems. Based on these, standard assertions can be included in the specifications made and certain properties expressed and verified in the associated query languages.

References

[1] Andrei, O., Ciobanu, G., Lucanu, D.: A rewriting logic framework for operational semantics of membrane systems. Theor. Comput. Sci. 373, 163–181 (2007)

[2] Bernardini, F., Gheorghe, M., Margenstern, M., Verlan, S.: How to synchronize the activity of all components of a P system? Int. J. Found. Comp. Sci. 404, 170–184 (2008)

[3] Bernardini, F., Gheorghe, M., Romero-Campero, F.J., Walkinshaw, N.: A hybrid approach to modeling biological systems. In: Eleftherakis, G., Kefalas, P., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing - 8th International Workshop, WMC 2007, Revised Selected and Invited Papers. LNCS, vol. 4860, pp. 138–159. Springer (2007)

[4] Ceterch, R., Sburlan, D: Membrane computing and computer science. In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Handbook of membrane computing, chap. 22, pp. 553–583. Oxford University Press (2010)

[5] Ciobanu, G.: Semantics of P systems. In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Handbook of membrane computing, chap. 16, pp. 413–436. Oxford University Press (2010)

[6] Ciobanu, G., Pérez-Jiménez, M.J., Păun, Gh. (eds.): Applications of Membrane Computing. Natural Computing Series, Springer (2006)

[7] Díaz-Pernil, D., Graciani, C., Gutiérrez-Naranjo, M.A., Pérez-Hurtado, I., Pérez-Jiménez, M.J.: Software for P systems. In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Handbook of membrane computing, chap. 17, pp. 437–454. Oxford University Press (2010)

[8] Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Hermanns, H., Palsberg, J. (eds.) 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2006. LNCS, vol. 3920, pp. 441–444. Springer (2006)

[9] Ipate, F., Gheorghe, M.: Testing non-deterministic stream X-machine models and P systems. Electr. Notes Theor. Comp. Sci. 227, 113–126 (2009)

[10] Ipate, F., Gheorghe, M., Lefticaru, R.: Test generation from P systems using model checking. J. Log. Algebr. Program., DOI:10.1016/j.jlap.2010.03.007 (2010)

[11] Kleijn, J., Koutny, M.: Petri nets and membrane computing. In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Handbook of membrane computing, chap. 15, pp. 389–412. Oxford University Press (2010)

[12] Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences 61, 108–143 (2000)

[13] Păun, Gh.: Membrane Computing: An Introduction. Springer-Verlag (2002)

[14] Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)

[15] Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Jiménez, A.: Complexity - membrane division, membrane creation. In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Handbook of membrane computing, chap. 12, pp. 302–454336. Oxford University Press (2010)

[16] Serbanuta, T., Stefanescu, G., Rosu, G.: Defining and executing P systems with structured data in K. In: Corne, D.W., Frisco, P., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing - 9th International Workshop, WMC 2008, Revised Selected and Invited Papers. LNCS, vol. 5391, pp. 374–393. Springer (2009)

MGS: *a declarative spatial computing programming language*[*]

Jean-Louis Giavitto[1], Antoine Spicher[2]

[1]*UMR 9912 STMS, IRCAM - CNRS - UPMC, 1 place Igor Stravinsky, 75004 Paris, France*
[2]*Université Paris-Est Créteil, LACL, 61 rue du Général de Gaulle, 94010 Créteil, France*
giavitto@ircam.fr;antoine.spicher@u-pec.fr

Abstract

We sketch the rationals of the MGS programming language. MGS is an experimental programming language developed to study the application of several spatial computing concepts to the specification and simulation of dynamical systems with a dynamical structure. MGS extends the notion of rewriting by considering more general structure than terms. The basic computation step in MGS replaces in a *topological collection* A, some subcollection B, by another topological collection C. A topological collection is a set of element structured by a neighborhood relationships describing an underlying space representing a data structure or constraints that must be fulfilled by the computation. This process proposes a unified view on several computational mechanisms initially inspired by biological or chemical processes (Gamma and the CHAM, Lindenmayer systems, Paun systems and cellular automata).

1 Dynamical Systems and their State Structures

A *dynamical system* (or DS in short) is a phenomenon that evolves in time. At any point in time, a dynamical system is characterized by a set of state variables. Intuitively, a DS is a formal way to describe how a point (the state of the system) moves in the *phase space* (the space of all possible states of the system). It gives a rule telling us where the point should go next from its current location. The evolution of the state over time is specified through a transition function or relation which determines the next state of the system (over some time increment) as a function of its previous state and, possibly, the values of external variables (input to the system).

This description outlines the change of state in time but does not stress that the set of state variables can also change in time and that the *a priori* determination of the phase space cannot always be done. This is a common situation in biology [FB94, FB96, Ros91] where such DS can be found in morphogenetic processes such as in plant growing, developmental biology or evolutionary processes. They also naturally appear in complex adaptive systems which exhibits dynamic networks of interactions and relationships not predefined aggregations of static entities (distributed systems, social networks, etc.).

This kind of systems share the property that the structure of the phase space must be computed jointly with the current state of the system. Note that if the set of state variables evolves in time, so does the transition function. We qualify such systems as $(\text{DS})^1$: *dynamical systems with a dynamical structure* [Gia03].

Computer Science has developed (or appropriated) many languages and tools to help model and simulate dynamical systems. However, the dynamic character of the structure raises a difficult problem: how to define a transition function when its set of arguments (the state variables) is not completely known at the specification time?

2 Space for Structuring Computation

The MGS project is an attempt to answer the previous question based on the assumption that the state of a $(\text{DS})^1$ can be dynamically structured in term of *spatial relationships* where only "neighbor" elements may interact.

Very often, a system can be decomposed into subsystems and the advancement of the state of the whole system results from the advancement of the state of its parts [GGMP02]. The change of state of a part can be intrinsic (*e.g.*, because of the passing of time) or extrinsic, that is, caused by some interaction with some other parts of the system.

The direct interactions of arbitrary elements in a system are not always allowed nor desirable and only "neighbor" elements may interact. For instance, for physical systems, subsystems are spatially localized and when a locality property[1] holds, only subsystems that are neighbors in space can interact directly. So the interactions between parts are structured by the spatial relationships of the parts. For abstract systems, in many cases the transition function of each subsystem only depends on the state variables of a small set of parts (and not on the

[*]This presentation relies partially on work presented already in [GM01c, GM02a, GMCS05, GS08b].

[1]The locality property states that matter/energy/information transmissions are done at a finite speed. This property is not always relevant, even for physical systems, for instance because processes may occurs at two separate time scales: changes at the fast time scale may appear instantaneous with respect to the slow one.

state variables of the whole system). In addition, if a subsystem s interacts with a subset $S = \{s_1, \ldots, s_n\}$ of parts, it also interacts with any subset S' included in S. This closure property induces a topological organization: the set of parts can be organized as an *abstract simplicial complex* [GMCS05].

The neighborhood relationship, which instantiates the locality property of the computations, can be formalized using concept from algebraic topology, see [GMCS05]. This relation represents *physical* (spatial distribution, localization of the resources) or *logical constraints* (inherent to the problem to be solved).

So, the idea is to describe the global dynamics by summing up the local evolutions triggered by local interactions. Note that two subsystems s and s' do not interact because they are identified *per se* but because they are neighbors. Such a feature enables the potential interaction of subsystems that do not yet exist at the beginning of the simulation and that do not know each other at their creation time.

3 Computing in Space, Space in Computation and Spatial Computing

We have described the rational behind the idea of using spatial relationships to structure the specification of a (DS)[1]. However, the use of spatial relationships can invade all domains of computing.

Spatial relationships adequately represent *physical constraints* (spatial distribution, localization of the resources) or *logical constraints* (inherent to the problem to be solved). Physical constraints are given by distributed computations where the computing resources are localized in different places in space.

Logical constraints arise when position and shape are input to computation or a key part of the desired result of the computation, *e.g.* in computational geometry applications, amorphous computing [AAC+00], claytronics [ABC+05], distributed robotics or programmable matter... to cite a few. More generally, logical constraints are given by the accessibility relation involved by a data structure [GM02a]. For instance, in a simply linked list the elements are accessed linearly (the second after the first, the third after the second, etc.). In arrays, a computation often involves only an element and its neighbors: the neighborhood relationships are left implicit and implemented through incrementing or decrementing indices. The concept of logical neighborhood in a data structure is not only an abstraction perceived by the programmer and vanishing at the execution. Indeed, the computation usually complies with the logical neighborhood of the data elements such that some primitive recursion schemes, e.g. catamorphisms and others polytypic functions on inductive types [MFP91] can be automatically defined from the data structure organization.

As an example, consider parallel computing. Parallel computing deals with both logical and physical constraints: computations are distributed on physical distinct computing resources but the distribution of the computation is a parameter of the execution, a choice done at a logical level to minimize the computation time, and does not depend on some imposed physical localizations induced solely by the problem to be solved.

[DHGG06] introduce the term *spatial computing* to recognize that space is not an issue to abstract away but that computation is performed distributed across space and that space, either physical or logical, serves as a mean, a resource, an input and an output of a computation.

4 Unifying Several Biologically Inspired Computational Models

One of our additional motivations is the ability to describe generically the basic features of four models of computation: Gamma (chemical computing) [BFL01], P systems (membrane computing) [Pău01], L systems [RS92] and cellular automata (CA) and related models of crystalline computing [Mar98, Tof04]. They have been developed with various goals in mind, e.g. parallel programming for Gamma, calculability and complexity issues for P systems, formal language theory and biological modeling for L systems, parallel distributed model of computation for CA (this list is not exhaustive). However, all these computational models rely on a biological or biochemical metaphor. They have been used extensively to model DS. So, it is then very tempting to ask if a uniform framework may encompass these formalisms, at least with respect to simulation purposes.

We assume that the reader is familiar with the main features of these formalisms. A Gamma program, a P or a L system and a CA can be themselves viewed abstractly as a discrete dynamical system: a running program can be characterized by a state and the evolution of this state is specified through *evolution rules*. From this point of view, they share the following characteristics:

Discrete space and time. The structure of the state (the multiset in Gamma, the membranes hierarchy in a P system, the string in a L system and the array in a CA) consists of a discrete *collection* of values. This discrete collection of values evolves in a sequence of discrete time steps.

Temporally local transformation. The computation of a new value in the new state depends only on values for a fixed number of preceding steps (and as a matter of fact, only one step).

Spatially local transformation. The computation of a new collection is done by a structural combination of the results of more elementary computations involving only a small and static subset of the initial collection.

"Structural combination", means that the elementary results are combined into a new collection, irrespectively of their precise value. "Small and static subset" makes explicit that only a fixed subset of the initial elements are used to compute a new element value (this is measured for instance by the diameter of the evolution rule of a P systems, the local neighborhood of a CA, the number of variables in the right hand side of a Gamma reaction or the context of a rule in a L system).

Considering these shared characteristics, the main difference between the four formalisms appears to be the organization of the collection. The abstract computational mechanism is always the same:

1. a subcollection A is selected in a collection C;
2. a new subcollection B is computed from the collection A;
3. the collection B is substituted for A in C.

We call these three basic steps a *transformation*. In addition to transformation specification, there is a need to account for the various constraints in the selection of the subcollection A and the replacement B. This abstract view makes possible the unification in the same framework of the above mentioned computational devices. The trick is just to change the organization of the underlying collection.

Constraining the Neighborhood. There is *a priori* no constraint in the case of Gamma: one element or many elements are replaced by zero, one or many elements. The means that the topology underlying a multiset makes all elements connected.

In the case of P systems, the evolution of a membrane may affect only the immediate enclosing membrane (by expelling some tokens or by dissolution): there is a *localization* of the changes. The topology of P systems are nested multiset: in a membrane all element are connected but membranes are organized in a strict hierarchy.

L systems are based on string (or sequence of elements). This also exhibit locality: the new collection B depends only on the context of A (the immediate neighbors in the string) and it is inserted at the place of A and not spread out over C.

For CA, the changes are not only localized, but also A and B are constrained to have the same shape: usually A is restricted to be just one cell in the array and B is also one cell to maintain the array structure.

5 MGS

MGS generalizes the previous formalisms by considering collection of elements with arbitrary neighborhood relationships and by developping a suitable notion of rewriting of such collections.

We see a collection as a set of *places* or *positions* organized by a *topology* defining the *neighborhood* of each element in the collection and also the possible subcollections. To stress the importance of the topological organization of the collection's elements, we call them "topological collection".

The topology needed to describe the neighborhood in a set or a sequence, or more generally the topology of the usual data structures, are fairly poor [GM02a]. So, one may ask if the machinery needed is worthwhile. Actually, more complex topologies are needed for some biological modeling applications or for self-assembly processes [GS08a, SMG11]. And more importantly, the topological framework unify various situations. notions, see [GM01b].

Now, we come back to our initial goal of specifying the dynamical structure of a DS. A collection is used to represent the state of a DS. The elements in the collection represent either entities (a subsystem or an atomic part of the DS) or messages (signal, command, information, action, etc.) addressed to an entity. A subcollection represents a subset of interacting entities and messages in the system. The evolution of the system is achieved through transformations, where the left hand side of a rule typically matches an entity and a message addressed to it, and where the right and side specifies the entity's updated state, and possibly other messages addressed to other entities. If one uses a multiset organization for the collection, the entities interact in a rather unstructured way, in the sense that an interaction between two objects is enabled simply by virtue of their both being present in the multiset (the topology underlying a multiset makes all elements connected). More organized topological collections are used for more sophisticated spatial organization.

We do not claim that topological collection are a useful theoretical framework encompassing all the previous formalisms. We advocate that few notions and a single syntax can be consistently used to allow the merging of these formalisms *for programming* purposes.

To go Further. The interested reader may refer to [GM02a, GM01a, GMC02] for the spatial approach of data structure. The mathematical concepts behind the MGS approach are presented in [GM01b, Spi06, GS08b, SMG10]. The modeling and the simulation of $(DS)^1$ are exposed for example in [GGMP02, Gia03, SMG04, GS08b]. Applications in systems biology are presented in [GM03, SM07, MSG09, SMG11]. Algorithmic applications are sketched in [GM01c, MJ05, SMG10]. Relation with novel approaches of computing, like P systems or autonomic computing, are showed in [GM02c, DHGG06, GS08a, GS08b, GMS08]. Further examples and documentation can be found at the MGS home page:

mgs.spatial-computing.org

Acknowledgments. The authors would like to thanks Olivier Michel for stimulating discussions and implementation works. They are also grateful to H. Klaudel, F. Pommereau, F. Delaplace and J. Cohen for many questions, encouragements and sweet cookies. This research is supported in part by the CNRS, the ANR projects AutoChem and SynBioTIC.

References

[AAC+00] Harold Abelson, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F. Knight, Radhika Nagpal, Erik Rauch, Gerald Jay Sussman, and Ron Weiss. Amorphous computing. *CACM: Communications of the ACM*, 43, 2000.

[ABC+05] Burak Aksak, Preethi Srinivas Bhat, Jason Campbell, Michael DeRosa, Stanislav Funiak, Phillip B. Gibbons, Seth Copen Goldstein, Carlos Guestrin, Ashish Gupta, Casey Helfrich, James F. Hoburg, Brian Kirby, James Kuffner, Peter Lee, Todd C. Mowry, Padmanabhan Pillai, Ram Ravichandran, Benjamin D. Rister, Srinivasan Seshan, Metin Sitti, and Haifeng Yu. Claytronics: highly scalable communications, sensing, and actuation networks. In Jason Redi, Hari Balakrishnan, and Feng Zhao, editors, *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys 2005, San Diego, California, USA, November 2-4, 2005*, page 299. ACM, 2005.

[BFL01] Jean-Pierre Banâtre, Pascal Fradet, and Daniel Le Métayer. Gamma and the chemical reaction model: Fifteen years after. *Lecture Notes in Computer Science*, 2235:17–44, 2001.

[DHGG06] André De Hon, Jean-Louis Giavitto, and Frédéric Gruau, editors. *Computing Media and Languages for Space-Oriented Computation*, number 06361 in Dagsthul Seminar Proceedings. Dagsthul, http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=2006361, 3-8 september 2006.

[FB94] Walter Fontana and Leo W. Buss. "the arrival of the fittest": Toward a theory of biological organization. *Bulletin of Mathematical Biology*, 1994.

[FB96] W. Fontana and L. Buss. *Boundaries and Barriers, Casti, J. and Karlqvist, A. edts,*, chapter The barrier of objects: from dynamical systems to bounded organizations, pages 56–116. Addison-Wesley, 1996.

[GGMP02] J.-L. Giavitto, C. Godin, O. Michel, and P. Prusinkiewicz. *Modelling and Simulation of biological processes in the context of genomics*, chapter "Computational Models for Integrative and Developmental Biology". Hermes, July 2002. Also republished as an high-level course in the proceedings of the Dieppe spring school on "Modelling and simulation of biological processes in the context of genomics", 12-17 may 2003, Dieppes, France.

[Gia03] J.-L. Giavitto. Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In *14th International Conference on Rewriting Technics and Applications (RTA'03)*, volume 2706 of *LNCS*, pages 208–233, Valencia, June 2003. Springer.

[GM01a] J.-L. Giavitto and O. Michel. Declarative definition of group indexed data structures and approximation of their domains. In *Proceedings of the 3nd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP-01)*. ACM Press, September 2001.

[GM01b] J.-L. Giavitto and O. Michel. `MGS`: a programming language for the transformations of topological collections. Technical Report 61-2001, LaMI – Université d'Évry Val d'Essonne, May 2001.

[GM01c] Jean-Louis Giavitto and Olivier Michel. Mgs: a rule-based programming language for complex objects and collections. In Mark van den Brand and Rakesh Verma, editors, *Electronic Notes in Theoretical Computer Science*, volume 59. Elsevier Science Publishers, 2001.

[GM02a] J.-L. Giavitto and O. Michel. Data structure as topological spaces. In *Proceedings of the 3nd International Conference on Unconventional Models of Computation UMC02*, volume 2509, pages 137–150, Himeji, Japan, October 2002. Lecture Notes in Computer Science.

[GM02b] J.-L. Giavitto and O. Michel. Pattern-matching and rewriting rules for group indexed data structures. In *ACM Sigplan Workshop RULE'02*, pages 55–66, Pittsburgh, October 2002. ACM.

[GM02c] Jean-Louis Giavitto and Olivier Michel. The topological structures of membrane computing. *Fundamenta Informaticae*, 49:107–129, 2002.

[GM03] J.-L. Giavitto and O. Michel. Modeling the topological organization of cellular processes. *BioSystems*, 70(2):149–163, 2003.

[GMC02] J.-L. Giavitto, O. Michel, and J. Cohen. Pattern-matching and rewriting rules for group indexed data structures. *ACM SIGPLAN Notices*, 37(12):76–87, December 2002. Selected papers from the PPDP satellite workshops. Revised version from [GM02b].

[GMCS05] J.-L. Giavitto, O. Michel, J. Cohen, and A. Spicher. Computation in space and space in computation. In *Unconventional Programming Paradigms (UPP'04)*, volume 3566 of *LNCS*, pages 137–152, Le Mont Saint-Michel, September 2005. Spinger.

[GMS08] Jean-Louis Giavitto, Olivier Michel, and Antoine Spicher. *Software-Intensive Systems and New Computing Paradigms*, volume 5380 of *LNCS*, chapter Spatial Organization of the Chemical Paradigm and the Specification of Autonomic Systems, pages 235–254. Springer, november 2008.

[GS08a] Jean-Louis Giavitto and Antoine Spicher. *Systems Self-Assembly: multidisciplinary snapshots*, chapter Simulation of self-assembly processes using abstract reduction systems, pages 199–223. Elsevier, 2008. doi:10.1016/S1571-0831(07)00009-3.

[GS08b] Jean-Louis Giavitto and Antoine Spicher. Topological rewriting and the geometrization of programming. *Physica D*, 237(9):1302–1314, jully 2008.

[Mar98] Norman Margolus. Crystalline computing. In Kathleen P. Hirons, Manuel Vigil, and Ralph Carlson, editors, *Proc. Conf. on High Speed Computing LANL * LLNL*, pages 249–255, Gleneden Beach, OR, April 1998. LANL. LA-13474-C Conference, UC-705.

[MFP91] E. Meijer, M. Fokkinga, and R. Paterson. Functional Programming with Bananas, Lenses, Envelopes and Barbed Wire. In *5th ACM Conference on Functional Programming Languages and Computer Architecture*, volume 523 of *Lecture Notes in Computer Science*, pages 124–144, Cambridge, MA, August 26–30, 1991. Springer, Berlin.

[MJ05] O. Michel and F. Jacquemard. *An Analysis of a Public-Key Protocol with Membranes*, pages 281–300. Natural Computing Series. Springer Verlag, 2005.

[MSG09] Olivier Michel, Antoine Spicher, and Jean-Louis Giavitto. Rule-based programming for integrative biological modeling – application to the modeling of the λ phage genetic switch. *Natural Computing*, 8(4):865–889, December 2009.

[Pău01] Gheorghe Păun. From cells to computers: computing with membranes (P systems). *Biosystems*, 59(3):139–158, March 2001.

[Ros91] Robert Rosen. *Life itself: a comprehensive inquiry into the nature, origin, and fabrication of life*. Columbia Univ Pr, 1991.

[RS92] G. Ronzenberg and A. Salomaa, editors. *L systems: from formalism to programming languages*. Springer Verlag, February 1992.

[SM07] Antoine Spicher and Olivier Michel. Declarative modeling of a neurulation-like process. *BioSystems*, 87(2-3):281–288, February 2007.

[SMG04] Antoine Spicher, Olivier Michel, and Jean-Louis Giavitto. A topological framework for the specification and the simulation of discrete dynamical systems. In *Sixth International conference on Cellular Automata for Research and Industry (ACRI'04)*, volume 3305 of *LNCS*, Amsterdam, October 2004. Springer.

[SMG10] Antoine Spicher, Olivier Michel, and Jean-Louis Giavitto. Declarative mesh subdivision using topological rewriting in mgs. In *Int. Conf. on Graph Transformations (ICGT) 2010*, volume 6372 of *LNCS*, pages 298–313, September 2010.

[SMG11] Antoine Spicher, Olivier Michel, and Jean-Louis Giavitto. *Understanding the Dynamics of Biological Systems: Lessons Learned from Integrative Systems Biology*, chapter Interaction-Based Simulations for Integrative Spatial Systems Biology. Springer Verlag, February 2011.

[Spi06] Antoine Spicher. *Transformation de collections topologiques de dimension arbitraire – Application à la modélisation de systèmes dynamiques*. PhD thesis, Université d'Evry, December 2006.

[Tof04] Tommaso Toffoli. A pedestrian's introduction to spacetime crystallography. *IBM Journal of Research and Development*, 48(1):13–30, 2004.

An Optical Approach to Computation

Sama Goliaei

SML Lab, Electrical and Computer Engineering Department, Tarbiat Modares University, Tehran, Iran
goliaei@modares.ac.ir

Abstract. Using optics in computing systems which refers to as optical computing, is an actively growing field in different areas such as data processing and data transmission. Two different approaches are used in optical data processing to design optical processors. The first approach is to simulate conventional electronic processors by designing optical logic gates. The second approach is to design optical processors, basically different from electronic processors based on physical properties of light, such as high parallel nature, splitting abilities, and high speed of light which may yield to obtain more computational power and efficiency.

In continuous to many other efforts to design optical processors in recent years, we have provided an optical method to solve the 3-satisfiability problem, using existence of several different wavelengths in light rays. We have considered each group of wavelengths as a possible value-assignment for the given 3-satisfiability formula, and we have used optical filters produced in preprocessing phase to drop wavelengths not satisfying the given formula. At the end, existence of some wavelengths in the outgoing light ray indicates that the 3-satisfiability formula is satisfiable. The method constructs polynomial number of optical filters in preprocessing phase taking exponential time. After preprocessing phase, the method requires polynomial time and exponential number of photons to solve each problem instance.

We have also provided an optical method to solve the graph 3-colorability problem. The method starts from some light rays considering each group of them as a possible 3-coloring of the graph, and then drops light rays corresponding to improper colorings using optical filters. Similar to the previous method, the optical filters are created in preprocessing phase in exponential time and the method takes polynomial time and exponential number of photons to solve each problem instance.

Key words: Unconventional Computing, Optical Computing, Optical Data Processing, 3-SAT Problem, Graph Coloring, 3-Colorability

References

[1] Dolev, S., Fitoussi, H.: Masking traveling beams:optical solutions for np-complete problems, trading space for time. Theoretical Computer Science **411** (2010) 837–853
[2] Goliaei, S., Jalili, S.: An optical wavelength-based solution to the 3-sat problem. In Dolev, S., Oltean, M., eds.: OSC 2009. Volume 5882 of LNSC., Springer-Verlag Berlin Heidelberg (2009) 77–85
[3] Goliaei, S., Jalili, S.: An optical solution to the 3-sat problem using wavelength based selectors. The Journal of Supercomputing (2010) 1–10
[4] Goliaei, S., Jalili, S.: Optical graph 3-colorability. In Dolev, S., Oltean, M., eds.: OSC 2010. Volume 6748 of LNSC., Springer-Verlag Berlin Heidelberg (2011) 16–22 to be appeared.
[5] Haist, T., Osten, W.: An optical solution for the traveling salesman problem. Optics Express **15**(16) (2007) 10473–10482
[6] Oltean, M.: Solving the hamiltonian path problem with a light-based computer. Natural Computing **7**(1) (2008) 57–70
[7] Oltean, M., Muntean, O.: Exact cover with light. New Generation Computing **26**(4) (2007)
[8] Oltean, M., Muntean, O.: Solving the subset-sum problem with a light-based device. Natural Computing **8**(2) (2009) 321–331
[9] Woods, D., Naughton, T.J.: An optical model of computation. Theoretical Computer Science **334**(1-3) (2005) 227–258

A General Framework for Computability[*]

Fabien Givors, Grégory Lafitte

Laboratoire d'Informatique Fondamentale de Marseille (LIF),
CNRS — Aix-Marseille Université,
39, rue F. Joliot-Curie, 13453 Marseille Cedex 13, France
{Fabien.Givors,Gregory.Lafitte}@lif.univ-mrs.fr

Abstract

Every recursively enumerable set of integers (r.e. set) is enumerable by a primitive recursive function. But if the enumeration is required to be one-one, only a proper subset of all r.e. sets qualify. Starting from a collection of total recursive functions containing the primitive recursive functions, we thus define a *sub-computability* as the structure of the r.e. sets that are one-one enumerable by total functions of the given collection. Notions similar to the classical computability ones are introduced and variants of the classical theorems are shown. The similarity between sub-computabilities and (complete) computability is surprising, since there are so many missing r.e. sets in sub-computabilities. This similarity hints at a general framework for computability, in which other computabilities, especially hyper-computabilities, can be embedded.

Many proofs in (basic and also more involved) computability rely on the algebraic structure of the enumeration of r.e. sets, partial functions, *etc.*, and not really *per se* on the notion of "being computable". The structure is provided by the properties of an acceptable enumeration, and consequences of being acceptable, *e.g.*, Kleene's second recursion theorem. One motivation behind the work of this article is to develop results similar to the classical computability ones (from basic results to Turing completeness issues) but in a setting verifying only a proper subset of the elementary properties of classical computability. In our case, this translates as the quest of developing computabilities with most of the nooks and crannies of classical computability without being all of computability. Here, a computability is meant to be collections of sets and functions which portray what we consider in this setting to be the r.e. sets and partial computable functions. Sub-computabilities are computabilities where these collections are a proper subset of the classical ones and is the main focus of this article. Higher computability can also be cast in this setting. Our setting can be seen as a toy model for computability, not based on models of machines, but on the algebraic structure of computability.

A sub-computability c is an enumeration of r.e. sets (called c-*enumerable*), which are one-one enumerated[1] by functions (called c-*fundamental*) in a sufficiently closed collection $\mathfrak{F}_{\mathsf{c}}$ (called the foundation of c) of total recursive functions. These fundamental functions somehow measure the effectiveness of the constructions used in computability. A partial recursive function is said to be *somewhat* c-*computable* if its graph is c-enumerable.

Sub-computabilities are like classical computability, the collection of all r.e. sets, but with holes punched in. Building on an enumeration $\mathbf{\Phi}^{\mathsf{c}}_{\cdot}$ of $\mathfrak{F}_{\mathsf{c}}$, we can respectively build canonical enumerations $\mathbf{W}^{\mathsf{c}}_{\cdot}$ (resp. $\varphi^{\mathsf{c}}_{\cdot}$) of c-enumerable sets (resp. somewhat c-computable functions).

An example of a sub-computability is p, the r.e. sets one-one enumerated by primitive recursive functions. Koz'minyh [2] in 1972 proved a key lemma on p. We generalize this lemma to any sub-computability. It gives insights into the behavior of c-enumerable sets and somewhat c-computable functions. Its corollaries make classical constructions possible, even if we do not have the general μ recursion operator. The (primitive recursive) effectiveness of these corollaries (and of most of our theorems) is especially useful.

Other sub-computability foundations (and thus sub-computabilities) stem from the field of *sub-recursion*. It provides a great number of examples of closed collections of ever more increasing total functions which do not contain all of the total recursive functions. This study is strongly entangled with proof theory and provides a clear picture of "what is provable and what is not" in a theory for which one is able to construct an *ordinal analysis*. In particular, proof theory has identified for many theories T the set of total recursive functions whose totality is provable in T. This connection gives another motivation to our work.

[*]The abstract presented in this paper has been made possible by the support of the French ANR grants *NAFIT* (ANR-08-DEFIS-008-01) and *EMC* (ANR-09-BLAN-0164-01).

[1]The complete definition also incorporates finite sets and $\varnothing$.

Studying sub-computabilities amounts to classifying r.e. sets by measuring the difficulty of enumerating them by way of sub-recursion. On that quest, one stumbles with the well-known at first surprising result that all r.e. sets are enumerable by primitive recursive functions. But if the enumeration is required to be one-one, the triviality evaporates: some r.e. sets are not one-one enumerable by primitive recursive functions, *e.g.*, the Ackermann function's range, but still many r.e. sets are primitively one-one enumerable, *e.g.*, the graph of the characteristic function of the range of the Ackermann function or the graph of a universal function.

If a set of integers is enumerable in a sub-computability, then it is recursively enumerable. If it is not enumerable, then it is either not recursively enumerable, or not feasibly enumerable in this sub-computability and necessitates more *power* to be seen as effectively enumerable. Thus, a sub-computability is an approximation of the classical computability: we have the same kind of results than in computability but with variations on the notions used; Classical (complete) computability is the *union* of all sub-computabilities.

For example, Post's theorem —stating that a set is recursive if and only if it is both recursively enumerable and co-recursively enumerable— does not hold anymore, leading the way to more restrictive notions of "being recursvive". Another example is Kleene's second recursion theorem, which only partially holds in various ways in our sub-computabilities.

We also define reducibilities which are refinements of Turing (or stronger) reducibilities and yield a structure of the associated degrees that looks very similar to the classical structure of the Turing degrees. One of the interesting aspects of this study is the simplicity of some of the proofs of the degree structure. To have this setting as a *real* toy model for computability, it would be nice to be able to have ways of transferring those results back in classical computability. Another application of sub-computabilities is using the strong links between proof theory and sub-recursion to tailor some reverse mathematics of computability and to partition the r.e. degrees according to the strength needed to prove their properties.

Our study is very different from other *sub-recursive degree* theories, especially when considering the fact that our objects of study are not necessarily recursive. Nevertheless, one of our future goals is to build bridges with these theories, *e.g.*, *honest sub-recursive degrees* (see [3, 4]), to be able to use their techniques and have their applications.

This abstract is a summary of [1], covering sub-computabilities, sub-reducibilities and related results concerning completeness and Post's problem. Our work uses elements of proof theory, in particular sub-recursion (see [6, 7]). A reference on higher computability is [8].

References

[1] F. Givors, G. Lafitte. *Holes punched computabilities.* 25 pages, soon at `http://www.lif.univ-mrs.fr/~lafitte/hpc.pdf` (February 2011)

[2] V.V. Koz'minyh. *On a presentation of partial recursive functions by compositions.* Algebra i Logika **11**(3), pp.270—294. In Russian. 1972.

[3] L. Kristiansen. *Papers on subrecursion theory.* PhD thesis, University of Oslo, 1996.

[4] L. Kristiansen. *A jump operator on honest subrecursive degrees.* Archive for Mathematical Logic **37**(2), pp.105—125, 1998.

[5] P. Odifreddi. *Classical Recursion Theory.* Volume I and II. North Holland - Elsevier, 1988 and 1999.

[6] W. Pohlers. *Proof Theory: The First Step into Impredicativity.* Springer, 2008.

[7] H.E. Rose. *Subrecursion: Functions and Hierarchies.* Vol. 9 of Oxford Logic Guides. Oxford University Press, USA, New York, 1984.

[8] G.E. Sacks. *Higher Recursion Theory.* Springer-Verlag, 1990.

Equivalence between Population Protocols and Turing Machines

Jonas Lefèvre

LIX, École Polytechnique, Palaiseau, France
jlefevre@lix.polytechnique.fr

Nowadays the size and the diversity of networks is greatly increasing. Very large networks of tiny agents, like cellphones for instance, becomes quite common. In such networks, it may become expensive in time or space to consider a unique identifier per device. To model such a situation, Angluin et al. introduce an abstract model of a network of devices: the population protocols [1].

In this model we consider a large population of anonymous and passively mobile agents with the computational power of finite automata. Every agent has the same transition rules. But we do not have any control on which pair of agent will interact. Due to the minimalistic nature of their model, the class of computable predicates was proved to be fairly small: it exactly is the class of semi-linear predicates, which are the predicates described by a formula from the Pressburger arithmetic. For more details, please read the survey [3]. In order to improve the power of the model, a lot of variants were proposed and studied: one-way or delayed communications [2], agent with identifier and a memory of size $\log(n)$ [5] [4], byzantine agents [5], and any combination.

The variant we study is a restriction of the interaction graph. The computational power increases when we delete edges in the interaction graph. Agents can not communicate with any agent anymore. In the case when the interaction graph is a string the model is equivalent to linear-space non-deterministic Turing machines. We are able to construct a hierarchy of population protocols by permitting more or less interaction within the population.

This work had lead us to study an other variant: the agent receive an identifier among a given set of identifiers, some agent may have the same identification. We hope to recycle the results of our first variant and to prove more. We aim to determine the set of computed formula and the equivalent Turing machine class in function of the quantity of identifiers.

References

[1] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *Proceedings of the 33rd annual ACM symposium on Principles of Distributed Computing (PODC '04)*, pages 290–299. ACM, 2004.

[2] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[3] James Aspnes and Eric Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 93:98–117, 2007.

[4] I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P.G. Spirakis. Passively mobile communicating logarithmic space machines. *ArXiv preprint arXiv:1004.3395*, 2010.

[5] Rachid Guerraoui and Eric Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP '09), Part II*, number 5556 in LNCS, pages 484–495. Springer-Verlag, 2009.

Points, Distances, and Cellular Automata: Geometric and Spatial Algorithmics

Luidnel Maignan

Alchemy team, INRIA Saclay-Île-de-France, Saclay, France
LRI (UMR CNRS 8623), University Paris-Sud, F-91405 Orsay, France
luidnel.maignan@inria.fr

Spatial computing aims at providing a scalable framework where computation is distributed on a uniform computing medium and communication happens locally between the nearest neighbors. Contrary to classical computation models based on Von-Neumann architecture, such a framework integrates from the start the fact that the communication time between two computation units is proportional to the distance separating them. We study the particular case of cellular automata, using a regular grid of computation units that update synchronously. As a first step towards generic computation, we propose to develop primitives allowing to structure the medium around a set of particles.

We consider three problems of geometrical nature. The first problem is analogous to load-balancing problems and considers moving the particles on the grid in order to uniformize their density [5]. This problem is solved specifically for the one dimensional grid and shows similarities with physical systems. Indeed implicit signals, corresponding to quantity of movements traveling through space, can be identified. They disappear either by cancellation with opposite signals or by leaving the space on the border.

The second problem is the construction of the convex hull of a set of particles [2, 6]. While this problem has already been tackled in the cellular automata framework, all solutions consider either that the space or the distance between the particle is bounded. They are also specific to particular cellular grids. The solution that we propose considers neither spatial nor distance bound, but uses nonetheless a finite number of states and a bounded neighborhood for any regular grid with an arbitrary number of dimensions.

The last problem is the construction of a connected proximity graph establishing connections between nearest particles [7]. The constructed graph is a generalization of the so-called Gabriel graphs, well-known in domains like clustering and wireless communication. Again, the solution that we propose considers no bound, uses a finite number of state, and works any regular multidimensional grids. In fact, the construction of this graph is a building block for the convex hull algorithms.

All these solutions are obtained by following the same global methodology. Our approach is to consider the metric space underlying the cellular automata topology and construct generic mathematical objects based solely on this metric [3]. This is the reason why the algorithms derived from the properties of those objects generalize over arbitrary regular grids. We implemented the usual ones, including hexagonal, 4 neighbors, and 8 neighbors square grids.

Also, all the solutions are based on the same basic component: the distance field, which associates to each site of the space its distance to the nearest particle [1, 4]. While the distance values are not bounded, it is shown that the difference between the values of neighboring sites is bounded, enabling encoding of the gradient into a finite state field. Our algorithms are expressed in terms of movements according to such gradient, and/or in terms of detection of local patterns in the gradient. This is the reason why our solutions use only a small finite number of states.

References

[1] *Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2008, Workshops Proceedings, October 20-24, 2008, Venice, Italy*. IEEE Computer Society, 2008.

[2] Stefania Bandini, Sara Manzoni, Hiroshi Umeo, and Giuseppe Vizzari, editors. *Cellular Automata - 9th International Conference on Cellular Automata for Research and Industry, ACRI 2010, Ascoli Piceno, Italy, September 21-24, 2010. Proceedings*, volume 6350 of *Lecture Notes in Computer Science*. Springer, 2010.

[3] Luidnel Maignan. *Points, Distances, and Cellular Automata: Geometric and Spatial Algorithmics*. PhD thesis, Université Paris-Sud 11, France, dec 2010.

[4] Luidnel Maignan and Frédéric Gruau. Integer gradient for cellular automata: Principle and examples. In *SASO Workshops* [1], pages 321–325.

[5] Luidnel Maignan and Frédéric Gruau. A 1d cellular automaton that moves particles until regular spatial placement. *Parallel Processing Letters*, 19(2):315–331, 2009.

[6] Luidnel Maignan and Frédéric Gruau. Convex hulls on cellular automata. In Bandini et al. [2], pages 69–78.

[7] Luidnel Maignan and Frédéric Gruau. Gabriel graphs in arbitrary metric space and their cellular automaton for many grids. *ACM Transaction Autonomous Adaptative Systems*, aug 2011, In Press.

About a message system for the tiles of the heptagrid of the hyperbolic plane

Maurice Margenstern

professor emeritus of Université Paul Verlaine − Metz
LITA, EA 3097, UFR-MIM, and CNRS, LORIA
Campus du Saulcy, 57045 Metz Cedex, France
margens@univ-metz.fr

This paper is a continuation of a topic considered in [1, 3] by the author.

In those papers, the author considered the possiblity for cells of a cellular automaton in the pentagrid or in the heptagrid of the hyperbolic plane to communicate. In this talk, we shall present a refinment of the protocol described in these papers and we shall present a small experiment performed in order to test several assumptions about a possible 'realization' of this message system.

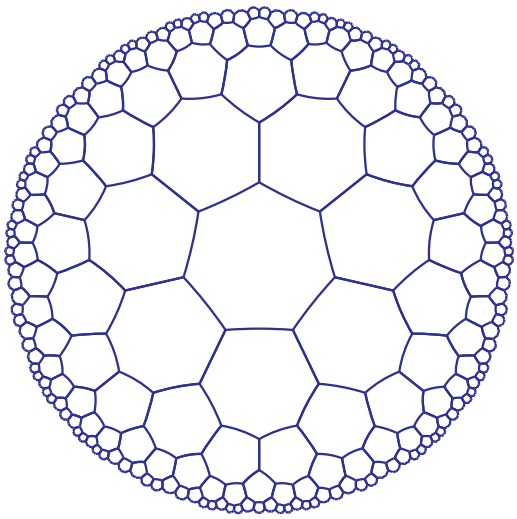


Figure 1. *The heptagrid, an illustration.*

We remind the reader that the heptagrid, see Figure 1, is one of the infinite family of tilings of the hyperbolic plane whose existence was established by Poincaré, see [2, 4]. We chose this tiling as it looks very close to the classical hexagonal tiling and as among the above mentioned tilings of the hyperbolic plane, it is the tiling with regular tiles whose interior angle is $\frac{2\pi}{3}$ for which the number of sides is the smallest.

The new protocol takes advantage of the implementation of an algorithm described in [3] in order to compute a shortest path between two tiles of the heptagrid. This implementation was performed by a computer program written in *ADA*95. This was one of the key points to implement the protocol and to proceed to the experiment. It is important to note that the experiment is necessarily very limited as the growth of the area of observation is exponential in the diameter of this area.

References

[1] M. Margenstern, On the communication between cells of a cellular automaton on the penta- and heptagrids of the hyperbolic plane, *Journal of Cellular Automata* **1**(3), (2006), 213-232.
[2] M. Margenstern, *Cellular Automata in Hyperbolic Spaces, Volume* 1*, Theory*, *OCP*, Philadelphia, (2007), 422p.
[3] M. Margenstern, *Cellular Automata in Hyperbolic Spaces, Volume* 2*, Implementation and computations*, *OCP*, Philadelphia, (2008), 360p.
[4] H. Poincaré, Théorie des groupes fuchsiens, Acta Math. **1** (1882), 1-62.

Beyond Determinism in Measurement-based Quantum Computation

Simon Perdrix

CNRS, Laboratoire d'Informatique de Grenoble, France
Simon.Perdrix@imag.fr

Information preserving evolutions play a fundamental role in quantum information processing. Unitary evolutions, which are deterministic reversible evolutions, are information preserving. In this presentation we consider two more general classes of information preserving evolutions. Indeed some non unitary evolutions which produce classical outcomes can be information preserving when one knows the classical outcomes. Such evolutions are called *equi-probabilistic* – when each classical outcome occurs with probability $1/2$ – or *constant-probabilistic* in the general case.

A priori non information preserving evolutions like quantum measurements can be combined in such a way that they induce an information preserving evolution, like in the quantum teleportation protocol [1] and in the one-way model [6] of quantum computation for instance. The former is a quantum protocol which transfers any quantum state from Alice to Bob, while the implementation of this protocol essentially consists in applying the so called Bell measurement. The latter is a universal model of quantum computation which consists in performing local measurements over a large entangled state represented by a graph and called graph state [3]. Any unitary evolution can be implemented by a one-way quantum computation. However, only few one-way quantum computations implement an information preserving evolution. A sufficient condition for a given graph state to guarantee a unitary evolution have been proved in [2, 5]: if the underlying graph has a certain kind of flow, called gflow, then a unitary evolution can be performed on the corresponding entangled state.

In this talk, in order to have a better understanding of the *flow of information* from the input to the output of a one-way quantum computation, I will present simple combinatorial conditions for equi-probabilistic and constant-probabilistic evolutions by means of excluded violating sets of vertices. I will show, in the particular case where the number of input and output qubits are the same, that graphs guaranteeing equi-probabilism and unitarity are the same.

This presentation is based on a joint work with Mehdi Mhalla, Mio Murao, Masato Someya, Peter Turner [4].

References

[1] C. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres and W. Wootters. *Teleporting an unknown quantum state via dual classical and EPR channels.* Phys. Rev. Lett. 70, 1895 (1993).

[2] D. E. Browne, E. Kashefi, M. Mhalla, and S. Perdrix. *Generalized flow and determinism in measurement-based quantum computation. New J. Phys.* **9**, 250, 2007.

[3] M. Hein, J. Eisert, and H. J. Briegel. *Multi-party entanglement in graph states. Phys. Rev. A* **69**, 062311, 2004.

[4] M. Mhalla, M. Murao, M. Someya, P. Turner. *Which graph states are useful for quantum information processing?* TQC'11, (to appear). 2011. arXiv:1006.2616.

[5] M. Mhalla, and S. Perdrix. *Finding optimal flows efficiently. ICALP proceeding Track A*, LNCS, 2008.

[6] R. Raussendorf and H. Briegel. *A one-way quantum computer. Phys. Rev. Lett.* **86**, 5188, 2001.

Solving Analytic Differential Equations in Polynomial Time over Unbounded Domains

Amaury Pouly

École normale supérieure de Lyon, LIP, 46 allée d'Italie, 69008 Lyon, France
amaury.pouly@ens-lyon.fr

We consider the following initial-value problem defined by an ODE

$$\begin{cases} x' = f(x) \\ x(0) = x_0 \end{cases} \tag{1}$$

where f is defined in some (possibly unbounded) domain.

In this paper we show that if $f : \mathbb{R}^n \to \mathbb{R}^n$ admits an analytic extension over $\mathbb{C}^n$ and $x : \mathbb{R} \to \mathbb{R}$ admits an analytic extension over $\mathbb{R}$ and both satisfies a very generous assumption about its growth rate, this solution can be computed in polynomial time from f and x_0 over $\mathbb{R}$. Actually, our constructions also works when considering solutions over $\mathbb{C}$ and assuming $f : \mathbb{C}^n \to \mathbb{C}^n$ analytic. Notice that, as it is well known, Equation (1) covers the case of an ODE of type $x' = f(t, x)$, as this latter case can be reduced to (1) by using a new variable x_{n+1} satisfying $x'_{n+1} = 1$.

Motivation 1 & Digression: Analog models of computation. We got to this result by trying to understand whether analog continuous-time models of computations do satisfy (some variant) of Church-Turing thesis: as such systems can usually be described by particular classes of ordinary differential equations, understanding whether they can compute more than Turing machines is equivalent to understanding whether they can always be simulated by Turing machines.

For example, the most well known example of analog model of computations is the General Purpose Analog Computer (GPAC) introduced by Claude Shannon in [7] as the idealization of an analog computer, the Differential Analyzer [1]. Shannon worked as an operator early in its career on these machines.

As it can be proved [4, 7] that any GPAC can be described by an ordinary differential equation of the form of (1) with f componentwise polynomial, proving that the GPAC does satisfy the Church-Turing thesis is equivalent to proving that solutions of such an ODE is always computable. It has been proved only recently that this holds for the case of f componentwise polynomial [5], [2]. Hence, the GPAC do satisfy the Church-Turing thesis. Notice that computability of solutions doesn't hold for general computable f [6], but in general known uncomputability results can only be obtained when the system is "ill behaved" (e.g. non-unique solutions in [6]). These kind of phenomena does not appear in models physically inspired by true machines like the GPAC.

Here, we are dealing with the next step. Do analog models like the GPAC satisfy the *effective* (in the sense of computable complexity) version of Church Turing thesis: all (sufficiently powerful) "*reasonable*" models of computations with "*reasonable*" measure of time are polynomially equivalent. In other words, we want to understand whether analog systems can provably (not) compute faster than usual classical digital models like Turing machines.

Taking time variable t as a measure of time (which is the most natural measure), proving that the GPAC can not compute more than Turing machines would require to prove that solutions of ODE (1) are always computable (in the classical sense) in a time polynomial in t, for f (componentwise) polynomial.

We here don't get exactly this result: for f componentwise polynomial, corresponding to GPACs, f is clearly analytic. But here, in our results, we have to suppose furthermore f to admit an analytic extension over $\mathbb{C}^n$. Although this case is stronger that when f is real analytic (it is well known that analyticity in the complex plane implies analyticity over the real line, but that the converse direction does not hold), we believe that our results are interesting on their own and provide a promising step towards the case of the real line.

Motivation 2: Recursive analysis. The results obtained in this paper turn out to be new and not known in a recursive analysis or classical computability or complexity perspective.

Being able to compute efficiently solutions of general ordinary differential equations is clearly of interest. Observe that all usual methods for numerical integrations (including basic Euler's method, Runge Kutta's methods, ...) do not provide the value of $x(t)$ in a time polynomial in t, whereas we do for general analytic functions under our hypotheses. Actually, as all these numerical methods falls in the general theory of n-order methods for some n, this is possible to use this theory (developed for example in [3]) to prove that none of them produce the value of $x(t)$ in a time polynomial in t. This has been already observed in [8], and claimed possible in [8] for some classes of functions by using methods of order n with n depending on t, but without a full proof. The method proposed here is different from the ideas proposed in this latter paper but prove that this is indeed possible to produce $x(t)$ in a time polynomial in t.

References

[1] V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.

[2] P. Collins and D. S. Graça. Effective computability of solutions of differential inclusions — the ten thousand monkeys approach. *Journal of Universal Computer Science*, 15(6):1162–1185, 2009.

[3] Jean-Pierre Demailly. *Analyse Numérique et Equations Différentielles*. Presses Universitaires de Grenoble, 1991.

[4] D. S. Graça and J. F. Costa. Analog computers and recursive functions over the reals. *J. Complexity*, 19(5):644–664, 2003.

[5] D.S. Graça, N. Zhong, and J. Buescu. Computability, noncomputability and undecidability of maximal intervals of IVPs. *Trans. Amer. Math. Soc.*, 361(6):2913–2927, 2009.

[6] M. B. Pour-El and J. I. Richards. A computable ordinary differential equation which possesses no computable solution. *Ann. Math. Logic*, 17:61–90, 1979.

[7] C. E. Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337–354, 1941.

[8] Warren D. Smith. Church's thesis meets the N-body problem. *Applied Mathematics and Computation*, 178(1):154–183, 2006.

A generic and modular signal machine solving satifiability problems[*]

Denys Duchier, Jérôme Durand-Lose, Maxime Senot

LIFO, University of Orléans, B.P. 6759, F-45067 Orléans Cedex 2, France
{denys.duchier,jerome.durand-lose,maxime.senot}@univ-orleans.fr

Abstract

Segments lines and their intersections in the Euclidean plane allow to realize non-trivial computations. Despite the simplicity of this abstract geometrical model of computation, it is possible to compute in the sense of Church-Turing, and even in an analog way. We present here a formalization of this idea into abstract devices: *signal machines*. To illustrate the power of these machines working on a continuous space-time, we present in this talk an efficient and geometrical solution to satifiability problems such as SAT or Q-SAT, by means of signals and their collisions. We also discuss complexities and propose a new measure for time complexity on signal machines: *the collision depth*, which is cubic for our proposed algorithm for Q-SAT.

Abstract Geometrical Computation. Signal machines, introduced in [Durand-Lose2003], take their origins in the world of cellular automata (see Fig.1). Indeed, signals and their interactions are very useful for studying problems and properties of cellular automata *e.g.* universality [Cook2004], synchronization [Mazoyer1996] or for implementing computations [Delorme and Mazoyer2002]. By abstracting the discrete nature of the cellular space-time to the continuity of the Euclidean plane, we can consider 1-dimensional and colored signals moving on the plane. Deterministic rules describe what happens when several signals meet, in function of their colors and speeds.

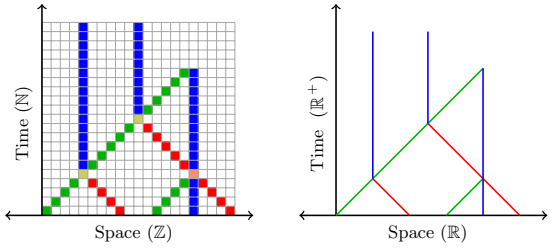


Figure 1: From cellular automata to signal machines.

Computations with colored segment lines in the Euclidean plane, their formalization into abstract devices— signal machines— and topological definitions of corresponding space-time diagrams constitute a larger class of models of computation, called *abstract geometrical computation* (*AGC*).

Other geometrical models of computation exist and allow to compute: Euclidean machines [Mycka and al.2006], colored universes [Jacopini and Sontacchi1990], geometric machines [Huckenbeck1989], piece-wise constant derivative systems [Bournez1997] ...

Signal machines can simulate Turing machines, and are thus Turing-universal [Durand-Lose2005]. They are also capable of analog computation by using the continuity of space and time to simulate analog models such as BSS's one [Durand-Lose2008, Blum and al.1989] or computable analysis [Durand-Lose2009, Weihrauch2000].

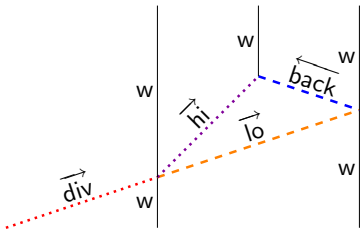
Signal machines. Each *signal* is an instance of a *meta-signal*, defining its type and its speed. When a set of signals collide, they are replaced by a new set of signals according to a matching *collision rule*. A rule has the form: $\{\sigma_1, \ldots, \sigma_n\} \to \{\sigma'_1, \ldots, \sigma'_p\}$ where all σ_i are meta-signals of distinct speeds as well as σ'_j. A *signal machine* is defined by a finite set of meta-signals and a set of collision rules. A signal machine is runned started from an initial configuration, *i.e.* a finite number of signals placed on the real line. Its evolution is representated geometrically as a *space-time diagram*, where space is always represented horizontally, and time vertically, growing upwards.

The geometrical algorithm displayed in Fig. 2 computes the middle: the new w is located exactly half way between the initial two w.

Computing in the fractal cloud. In this talk, to illustrate some abilities of signal machines, we present a geometrical algorithm solving Q-SAT—the problem of quantified boolean formula satisfiablity—which also provides a structure for solving other variants of boolean satifiablity.

[*]Details of this work can be found at http://arxiv.org/abs/1105.3454 ([Duchier and al.2011a]).

Meta-Signals	Speed
w	0
div	3
hi	1
lo	3
back	-3



Collision rules

$\{ \mathsf{w}, \mathsf{div} \} \to \{ \mathsf{w}, \mathsf{hi}, \mathsf{lo} \}$
$\{ \mathsf{lo}, \mathsf{w} \} \to \{ \mathsf{back}, \mathsf{w} \}$
$\{ \mathsf{hi}, \mathsf{back} \} \to \{ \mathsf{w} \}$

Initial configuration

$\mathsf{div} \ @ \ s < 0$
$\mathsf{w} \ @ \ 0$
$\mathsf{w} \ @ \ x > 0$

Figure 2: Geometrical algorithm for computing the middle.

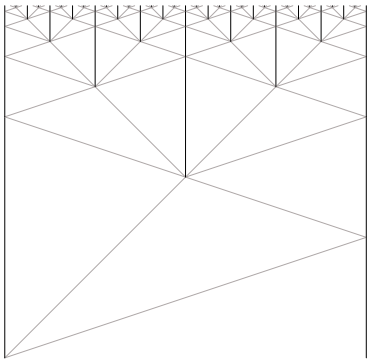


Figure 3: The fractal tree.

This construction is based on the previous algorithm computing the middle, which is used recursively to build the infinite binary tree of Fig. 3, the *fractal cloud*. This tree is then interpreted as a binary search tree, where the number of boolean variables determines the number of needed levels, and all possible boolean assignments can be tested in a massive parallel way. A computation in the fractal cloud is based on the *Map/Reduce* principle and follows three steps: distributing the computation along the tree, evaluating each case at the top level, and collecting all the results to yield the final answer.

Solving Q-SAT with a single machine. A Q-SAT formula is coded by signals and is propagated along the tree. Variables are representated by a set of signals so that at the i^{th} level, the variable x_i splits into two signals: a true signal going right and a false signal.

At the final stage, in each box (corresponding to each possible assignment of variables), there is no more variable but only true and false signals and signals coding boolean connectives. The unquantified formula is then evaluated in each case, and the results are aggregated with respect to the quantifiers of the input formula. The final answer—either the quantified formula is true or false—is given by the last signal going left at the top of the whole construction, which can be seen in Fig. 4.

We already provided geometrical algorithms in [Duchier and al.2010] and in [Duchier and al.2011b] to solve respectively SAT and Q-SAT, but in both cases, the machines were dependent on the input formula: a signal machine was generated in polynomial time for each boolean formula.

We present here a *single* signal machine solving Q-SAT for any instance coded in the initial configuration. We also provide a structure (the fractal cloud) and a *modular* approach (the tree, the propagation, the evaluation... can be programmed independently) permitting to solve easily other satisfiablity problems such as #SAT or MAX-SAT.

As all these constructions, bounded by the infinite binary tree, are made in constant width and time independently of the size of the formula, space and time are no longer appropriate complexity measures. We define a new measure for time complexity, better suited to the massive parallelism of signal machines: the *collision depth*. It is defined as the maximal number of consecutive collisions when we follow an ascending path through the whole diagram. Whereas the instance-specific constructions were in quadratic collision depth, the generic one is in cubic collision depth. This gives us an idea of the cost of genericity.

References

[Blum and al.1989] Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the AMS*, 21(1):1–46.

[Bournez1997] Bournez, O. (1997). Some bounds on the computational power of piecewise constant derivative systems. In *ICALP '97*, number 1256 in LNCS, pages 143–153.

[Cook2004] Cook, M. (2004). Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40.

[Delorme and Mazoyer2002] Delorme, M. and Mazoyer, J. (2002). Signals on cellular automata. In Adamatzky, A., editor, *Collision-based Computing*, pages 231–275. Springer-Verlag.

[Duchier and al.2010] Duchier, D., Durand-Lose, J., and Senot, M. (2010). Fractal parallelism: Solving SAT in bounded space and time. In Cheong, O., Chwa, K.-Y., and Park, K., editors, *21st International Symposium on Algorithms and Computation (ISAAC '10)*, number 6506 in LNCS, pages 279–290. Springer.

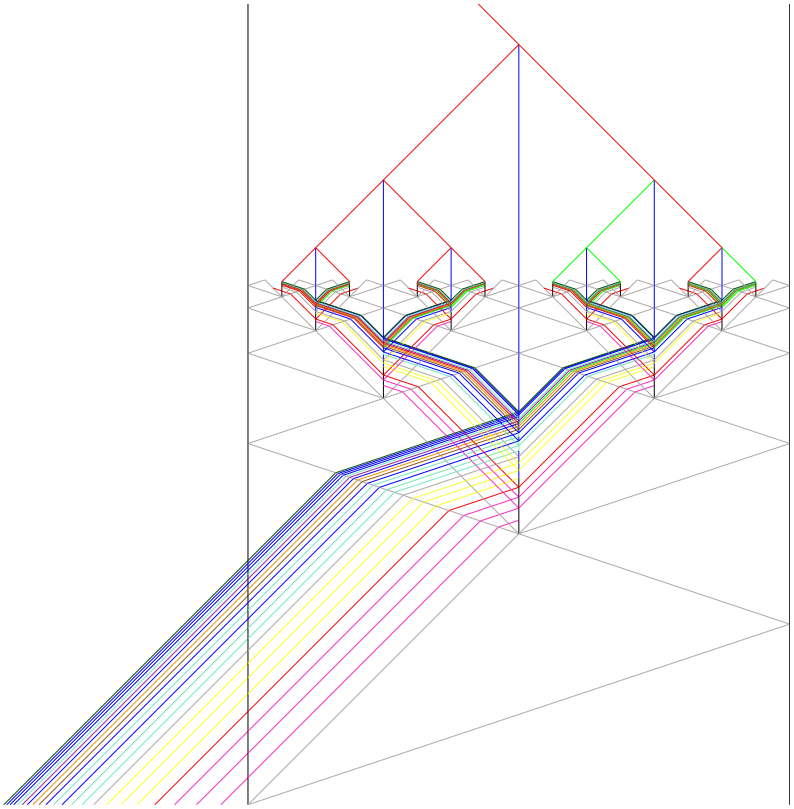


Figure 4: The whole diagram.

[Duchier and al.2011a] Duchier, D., Durand-Lose, J., and Senot, M. (2011). Computing in the fractal cloud: modular generic solvers for SAT and Q-SAT variants. *Arxiv preprint arXiv:1105.3454*. Available at http://arxiv.org/abs/1105.3454.

[Duchier and al.2011b] Duchier, D., Durand-Lose, J., and Senot, M. (2011). Solving Q-SAT in bounded space and time by geometrical computation. In *7th Int. Conf. Computability in Europe (CiE '11)*. To appear in the local booklet (abstracts and extended abstracts of unpublished papers).

[Durand-Lose2003] Durand-Lose, J. (2003). *Calculer géométriquement sur le plan – machines à signaux*. Habilitation à Diriger des Recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis. In French.

[Durand-Lose2005] Durand-Lose, J. (2005). Abstract geometrical computation: Turing computing ability and undecidability. In Cooper, B., Löwe, B., and Torenvliet, L., editors, *New Computational Paradigms, 1st Conf. Computability in Europe (CiE '05)*, number 3526 in LNCS, pages 106–116. Springer.

[Durand-Lose2008] Durand-Lose, J. (2008). Abstract geometrical computation with accumulations: Beyond the Blum, Shub and Smale model. In Beckmann, A., Dimitracopoulos, C., and Löwe, B., editors, *4th Conf. Computability in Europe (CiE '08) (extended abstracts of unpublished papers)*, pages 107–116. University of Athens.

[Durand-Lose2009] Durand-Lose, J. (2009). Abstract geometrical computation and computable analysis. In Costa, J. and Dershowitz, N., editors, *8th Int. Conf. on Unconventional Computation 2009 (UC '09)*, number 5715 in LNCS, pages 158–167. Springer.

[Huckenbeck1989] Huckenbeck, U. (1989). Euclidian geometry in terms of automata theory. *Theoret. Comp. Sci.*, 68(1):71–87.

[Jacopini and Sontacchi1990] Jacopini, G. and Sontacchi, G. (1990). Reversible parallel computation: an evolving space-model. *Theoret. Comp. Sci.*, 73(1):1–46.

[Mazoyer1996] Mazoyer, J. (1996). On optimal solutions to the firing squad synchronization problem. *Theoret. Comput. Sci.*, 168(2):367–404.

[Mycka and al.2006] Mycka, J., Coelho, F., and Costa, J. F. (2006). The Euclid abstract machine: Trisection of the angle and the halting problem. In Calude, C. S., Dinneen, M. J., Paun, G., Rozenberg, G., and Stepney, S., editors, *5th Int. Conf. on Unconventional Computation(UC 06)*, volume 4135 of *LNCS*, pages 195–206. Springer.

[Weihrauch2000] Weihrauch, K (2000). Computable Analysis: an Introduction. Springer Verlag.

Relativity in mathematical descriptions of automatic computations

Yaroslav D. Sergeyev

Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, 87030 Rende (CS), Italy
yaro@si.deis.unical.it

Abstract.
The Turing machine is one of the simple abstract computational devices that can be used to investigate the limits of computability. In this talk, they are considered from several points of view that emphasize the importance and the relativity of mathematical languages used to describe the Turing machines. A deep investigation is performed on the interrelations between mechanical computations and their mathematical descriptions emerging when a human (the researcher) starts to describe a Turing machine (the object of the study) by different mathematical languages (the instruments of investigation). Together with traditional mathematical languages using such concepts as 'enumerable sets' and 'continuum' a new computational methodology allowing one to measure the number of elements of different infinite sets is used in this paper. It is shown how mathematical languages used to describe the machines limit our possibilities to observe them. In particular, notions of observable deterministic and non-deterministic Turing machines are introduced and conditions ensuring that the latter can be simulated by the former are established.

Keywords. Theory of automatic computations; observability of Turing machines; relativity of mathematical languages; infinite sets; Sapir-Whorf thesis.

References

[1] J.B. Carroll, editor. *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf.* MIT Press, 1956.
[2] A. Church. *An unsolvable problem of elementary number theory.* American Journal of Mathematics, 58:345–363, 1936.
[3] S. Barry Cooper. *Computability Theory.* Chapman Hall/CRC, 2003.
[4] M. Davis. *Computability & Unsolvability.* Dover Publications, New York, 1985.
[5] P. Gordon. *Numerical cognition without words: Evidence from Amazonia.* Science, 306:496–499, 2004.
[6] A.N. Kolmogorov and V.A. Uspensky. *On the definition of algorithm.* Uspekhi Mat. Nauk, 13(4):3–28, 1958.
[7] Ya.D. Sergeyev. http://www.theinfinitycomputer.com. 2004.
[8] Ya.D. Sergeyev. *A new applied approach for executing computations with infinite and infinitesimal quantities.* Informatica, 19(4):567–596, 2008.
[9] Ya.D. Sergeyev. *Numerical computations and mathematical modelling with infinite and infinitesimal numbers.* Journal of Applied Mathematics and Computing, 29:177–195, 2009.
[10] Ya.D. Sergeyev. *Numerical point of view on Calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains.* Nonlinear Analysis Series A: Theory, Methods & Applications, 71(12):1688–1707, 2009.
[11] Ya.D. Sergeyev. *Counting systems and the First Hilbert problem.* Nonlinear Analysis Series A: Theory, Methods & Applications, 72(3-4):1701–1708, 2010.
[12] Ya.D. Sergeyev, A. Garro. *Observability of Turing Machines: a refinement of the theory of computation.* Informatica, 21(3):425–454, 2010.
[13] Ya.D. Sergeyev. *Lagrange Lecture: Methodology of numerical computations with infinities and infinitesimals.* Rendiconti del Seminario Matematico dell'Università e del Politecnico di Torino, 68(2):95–113, 2010.
[14] Ya.D. Sergeyev. *On Accuracy of Mathematical Languages Used to Deal With the Riemann Zeta Function and the Dirichlet Eta Function.* p-Adic Numbers, Ultrametric Analysis and Applications, 3(2):129–148, 2011.

Recent Discussions in Cosmological Computation

Mike Stannett

University of Sheffield, UK
m.stannett@dcs.shef.ac.uk

Abstract

Andréka, Németi and their colleagues have developed a series of first order relativity theories, in which they consider the extent to which results from relativistic physics can be derived formally from various sets of basic axioms; some of the associated models are thought to permit systems that can solve formally undecidable problems by exploiting black hole geometries. In related work, we have recently started joint work looking at the occurrence of closed timelike curves (CTCs) in general relativistic models, as might exist for example as a consequence of traversable wormholes. In this talk I will discuss some of the consequences of CTCs, and their relevance to physical computation.

Why does 'time' seem so different to 'space'? This difference is fundamental to the way we experience the world, yet in view of relativity theory it seems almost paradoxical. Suppose we model spacetime as a $4d$ Minkowski space with coordinates (t, x, y, z). The effect of boosting an object's x-velocity by an amount v can then be represented in the usual way as a Lorentzian transformation

$$\begin{bmatrix} ct' \\ x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cosh\phi & -\sinh\phi & 0 & 0 \\ -\sinh\phi & \cosh\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ct \\ x \\ y \\ z \end{bmatrix} \quad \text{where } e^{\phi} = \sqrt{\frac{(1+v/c)}{(1-v/c)}}$$

or in other words, as a kind of *rotation* in spacetime. But consider what this implies. If an object can appear to *rotate* in spacetime, simply as a result of an observer changing their own relative motion, then space and time must be fundamentally similar in nature. Apart from the appearance of the negative sign in the metric, time should be no different to space.

But how different these seem. We appear to move through space at will, happily revisiting favourite locations as often as we wish. Yet time seems to flow inexorably forwards; the past is forever fixed, the future unknowable. The question arises, then, which interpretation (if either) is correct? Is time a unidirectional flow of a fundamentally different nature to space, or are time and space intrinsically identical types of thing?

This question was thrown into sharp relief by Gödel [Göd49], who demonstrated a rotating solution to Einstein's general relativistic equations which included *closed timelike curves* (CTCs). An observer traversing a CTC behaves like any other, moving always forwards in time, but the geometry of the spacetime in which they move leads them around a loop, so that they eventually revisit an event in their own past. For such an observer, the concepts of *past* and *present* become largely indistinguishable, since every future event on a CTC lies also in the past, and vice versa. While Gödel's solution appears to be unphysical (the exceptional isotropy of the cosmic background suggests the absence of a preferred rotation axis), it nonetheless highlights the possibility that CTCs exist within physically realistic universes, and maybe even our own. Unsurprisingly, given that CTCs appear to offer the possibility of 'time travel' and causality violations, their consequences have been the focus of considerable research in cosmology [Haw92, Tho93, Vis03], and more recently, in computer science [AW09].

In this talk we discuss the consequences of CTCs for computer science, taking as our theme various recent conversations with members of the geometric logic in Budapest.[3] The Budapest group have developed a series of first-order relativity theories, which provide a solid mathematical foundation for reasoning about cosmological behaviours [AMNS11]. We note that the nature of CTC 'time travel' is itself unclear, since it can be interpreted in two very different ways. In neither case, however, do CTCs, of themselves, necessarily entail causality violations. We also discuss the possibilities raised by CTCs both in terms of increased computational efficiency [Sta11], and regarding physical hypercomputation [ANS11].

References

[AMNS11] H. Andréka, J. X. Madarász, I. Németi, and G. Székely. A logic road from special relativity to general relativity. `arXiv:1005.0960v2` [gr-qc], 2011.

[ANS11] Hajnal Andréka, I. Németi, and G. Székely. Closed Timelike Curves in Relativistic Hypercomputation. Submitted to HyperNet 2011, Turku, June 2011, 2011.

[AW09] S. Aaronson and J. Watrous. Closed timelike curves make quantum and classical computing equivalent. *Proceedings of the Royal Society A*, 465(2102):631–647, 2009.

[3] I am especially grateful to Hajnal Andréka, István Németi, Judit X. Madarász and Gergely Székely for their input.

[Göd49] K. Gödel. An example of a new type of cosmological solution of Einstein's field equations of gravitation. *Rev. Mod. Phys.*, 21:447–450, 1949.

[Haw92] S. W. Hawking. The chronology protection conjecture. *Phys. Rev. D*, 46:603–611, 1992.

[Sta11] M. Stannett. Computation and Spacetime Structure. Submitted to Physics & Computation 2011, Turku, June 2011, 2011.

[Tho93] Kip S. Thorne. Closed Timelike Curves. Technical Report GRP-340, CalTech, 1993. http://www.its.caltech.edu/~kip/scripts/ClosedTimelikeCurves-II121.pdf.

[Vis03] Matt Visser. The quantum physics of chronology protection. In G. W. Gibbons, E. P. S. Shellard, and S. J. Rankin, editors, *The Future of Theoretical Physics and Cosmology: Celebrating Stephen Hawking's 60th Birthday*. Cambridge University Press, Cambridge, 2003.

Computing under Vagueness

Apostolos Syropoulos

Greek Molecular Computing Group, Xanthi, Greece
asyropoulos@yahoo.com

1 What is Vagueness?

Nowadays, the terms *fuzziness* and *vagueness* are used interchangeably, nevertheless, the term fuzziness is closely associated to fuzzy set theory and its accompanying logic. Since the use of the term vagueness precedes the use of the term fuzziness, I have opted to use this term in the title of this paper.

Vagueness is a very important notion. Unfortunately, most scientific theories, including computability theory, are "ostensibly expressed in terms of objects never encountered in experience" as Max Black [2] has pointed out. For example, there are no spheres in nature, but there are objects that are spheres to a certain degree. Bertrand Russell [8] has defined vagueness as follows:

Definition 1. *Per contra*, a representation is *vague* when the relation of the representing system to the represented system is not one-one, but one-many.

For instance, Russel suggests that a photograph which is so smudged that it might equally represent Brown or Jones or Robinson is vague. In addition, Russell and Black agree that vagueness should not be confused with *generality*, as the former applies in cases where there is lack of specification of boundaries. In addition, vagueness should not be confused with lack-of-information.

Black was the first who tried to formalize vagueness, but he did not managed to propose a full-fledged mathematical theory. This was achieved to a certain degree (!) by Zadeh who introduced fuzzy set theory.

2 Fuzzy Set Theory in a Nutshell

Fuzzy set theory was proposed by Lotfi Askar Zadeh [19] as an extension of ordinary set theory. Zadeh defined fuzzy sets by generalizing the memebrship relationship. In particular, given a universe X, he defined a fuzzy subset of X to characterized by a function $A : X \to \mathrm{I}$, where I is the unit interval. The value $A(x)$ specifies the degree to which some element x belongs to A. Despite its superficial similartiy to probability theory, fuzzy set theory is a different theory. Zadeh [18] has argued that the theories are different facets of vagueness. However, Bart Kosko [4] and other researchers, including this author, have argued that fuzzy set theory is more fundamental than probability theory.

Assume that $A, B : X \to \mathrm{I}$ are two fuzzy subsets of X. Then, $(A \cup B)(x) = \max\{A(x), B(X)\}$ and $(A \cap B)(x) = \min\{A(x), B(X)\}$. Also, if B is the complement of the fuzzy subset A, then $B(x) = 1 - A(x)$. A main deficiency of the theory is that Zadeh *fuzzified* the membership relationship, but he did not fuzzyfied the equality relationship.

In the years following the publication of Zadeh's paper, various researchers proposed and defined various fuzzy structures (e.g., fuzzy algebraic structures, fuzzy topologies, etc.). In particular, the concept of fuzzy languages was introduced by E.T. Lee and Zadeh [5]:

Definition 2. A fuzzy language λ over an alphabet S (i.e., an ordinary set of symbols) is a fuzzy subset of S^*.

If $s \in S^*$, then $\lambda(s)$ is the grade of membership that s is a member of the language.

3 Fuzzy Turing Machines

As expected, Zadeh [20] was the first researcher who mentioned fuzzy Turing machines and fuzzy algorithms or programs. According to Zadeh, a program is fuzzy if it contains fuzzy commands, that is, commands like the following one:

Make y approximately equal to 10, if x is approximately equal to 5.

Zadeh hinted about the way fuzzy programs can be executed, but it was Shi-Kuo Chang [3], Kokichi Tanaka and Masaharu Mizumoto [16], and Eugene S. Santos [10] who made precise the notion of fuzzy programs and their execution. Santos [9] was the first researcher who had given a formal definition of a fuzzy Turing machine:

Definition 3. A fuzzy Turing machine is a septuple $(S, Q, q_i, q_f, \delta, W, \delta_W)$ where:

1. S represents a finite non-empty set of input symbols,

2. Q denotes a finite non-empty set of states such that $S \cap Q = \emptyset$,
3. $q_i, q_f \in Q$ are the symbols designating the initial and final state, respectively,
4. $\delta \subset (Q \times S) \times (Q \times (S \times \{-1, 0, 1\}))$ is the next-move relation,
5. W is the semiring $(W, \wedge, \vee)$,
6. $\delta_W : (Q \times S) \times (Q \times (S \times \{-1, 0, 1\})) \to W$ is a W-function that assigns a degree of certainty to each machine transition.

Modern versions of this machine use t-norms and t-conorms instead of semirings. In particular, Jiří Wiedermann [17] has defined such a machine and proved that fuzzy languages accepted by these machines with a computable t-norm correspond exactly to the union $\Sigma_1^0 \cup \Pi_1^0$ of recursively enumerable languages and their complements. However, Benjamín Callejas Bedregal and Santiago Figueira [1] have shown that this very important result is not true in general. These researchers have partially shown also that there are no universal fuzzy Turing machines. Also, Yongming Li [6] has shown the nonexistence of an unrestricted universal fuzzy Turing machine.

4 Fuzzy P Systems

P systems [7] is a model of computation inspired by the way living cells function. Basically, a P system is structure that consists of nested, porous membranes that contain indistinguishable copies of objects. Attached to each compartment is a set of rewrite rules, that is, equations that roughly specify how the contents of a compartment should be modified. In particular, such rules may specify that copies of certain objects should be deleted or moved to another compartment or that copies of objects should be introduced from outside or be created out of thin air. Rules are applied in parallel in such a way that only optimal output is generated. When there is no more activity, the result of the computation is equal to the number of (copies of the) objects found in a designated compartment—the output compartment. P systems operate in a massively parallel way while they can interact with their environment.

Fuzzy P systems has been introduced by this author [12]. Typically, a P system is modelled by multisets (see [11] for an overview of the theory of multisets) and multiset rewrite rules that operate on these sets. If we consider that the multisets are actually fuzzy multisets, then we get a version of fuzzy multisets. Since the cardinality of fuzzy multisets is a real number, one can compute real numbers with fuzzy P systems. It is possible to generalize fuzzy P systems by replacing fuzzy multisets with L-fuzzy multisets or even by L-fuzzy hybrid sets, but it is not clear whether this will increase the computational power of the resulting system. However, it seems that these go beyond the Church-Turing barrier [13, 15].

The fuzzy chemical abstract machine [14] is model of computation that is similar to fuzzy P systems. However, the study of this model has just started!

References

[1] Benjamín Callejas Bedregal and Santiago Figueira. On the computing power of fuzzy turing machines. *Fuzzy Sets Systems*, 159:1072–1083, 2008.

[2] Max Black. Vagueness. An Exercise in Logical Analysis. *Philosophy of Science*, 4(4):427–455, 1937.

[3] Shi-Kuo Chang. On the Execution of Fuzzy Programs Using Finite-State Machines. *IEEE Transactions on Computers*, C-21(3):241–253, 1972.

[4] Bart Kosko. Fuzziness vs. Probability. *International Journal of General Systems*, 17(2):211–240, 1990.

[5] E.T. Lee and Lotfi Askar Zadeh. Note on Fuzzy Languages. *Information Sciences*, 1:421–434, 1969.

[6] Yongming Li. Fuzzy turing machines: Variants and universality. *IEEE Transactions on Fuzzy Systems*, 16:1491–1502, 2008.

[7] Gheorghe Păun. *Membrane Computing: An Introduction*. Springer-Verlag, Berlin, 2002.

[8] Bertrand Russell. Vagueness. *Australasian Journal of Philosophy*, 1(2):84–92, 1923.

[9] Eugene S. Santos. Fuzzy Algorithms. *Information and Control*, 17:326–339, 1970.

[10] Eugene S. Santos. Fuzzy and Probablistic Programs. *Information Sciences*, 10:331–345, 1976.

[11] Apostolos Syropoulos. Mathematics of Multisets. In C.S. Calude, Gh. Păun, Gr. Rozenberg, and A. Salomaa, editors, *Multiset Processing*, number 2235 in Lecture Notes in Computer Science, pages 347–358. Springer-Verlag, Berlin, 2001.

[12] Apostolos Syropoulos. Fuzzifying P Systems. *The Computer Journal*, 49(5):619–628, 2006.

[13] Apostolos Syropoulos. *Hypercomputation: Computing Beyond the Church-Turing Barrier*. Springer New York, Inc., Secaucus, NJ, USA, 2008.

[14] Apostolos Syropoulos. Fuzzy chemical abstract machines. *CoRR*, abs/0903.3513, 2009.

[15] Apostolos Syropoulos. On Generalized Fuzzy Multisets and their Use in Computation. To appear in the "Iranian Journal of Fuzzy Systems", 2011.

[16] Kokichi Tanaka and Masaharu Mizumoto. Fuzzy programs and their executions. In Lotfi A. Zadeh, King-Sun Fu, Kokichi Tanaka, and Masamichi Shimura, editors, *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*, pages 41–76. Academic Press, New York, 1975.

[17] Jiří Wiedermann. Characterizing the super-Turing computing power and efficiency of classical fuzzy Turing machines. *Theoretical Computer Science*, 317:61–69, 2004.

[18] Lotfi A. Zadeh. Discussion: Probability Theory and Fuzzy Logic Are Complementary Rather Than Competitive. *Technometrics*, 37(3):271–276, 1995.

[19] Lotfi Askar Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[20] Lotfi Askar Zadeh. Fuzzy Algorithms. *Information and Control*, 12:94–102, 1968.