Research program:
“Theories of computations for continuous systems. Applications to models of computations, to distributed computing, and to algorithmic complexity.”

Olivier Bournez
January 26, 2016

1 Preamble

This document presents my research program on theories of computations for continuous systems, and their applications to models of computations, to algorithmic complexity of problems in verification and control theory, and to distributed computing.

I first present theories of computations for continuous systems, and their various motivations. I then present some examples of systems on which I already worked. I then describe some results of my own going in the direction of this research program. Finally, I present four main research objectives, which I think important for the next few years, the fourth being personally more recent, but in relation with a true orientation that I am currently trying to insufflate to my work.

The arguments in this text are actually developed in greater depth and length in several chapters of my habilitation (habilitation à diriger les recherches, HDR) [?]. My HDR is indeed mostly a development of this research program: chapters 1 and 2 are about motivation. Chapter 3 is a survey on the state of the art about theories of computations for continuous time systems, with perspectives and discussions (also published as a book chapter in [?]). Chapter 6 is a conclusion discussing main questions.

2 Motivations

The overall objective of my work is to understand theories of computations for continuous systems.

Continuous systems A very general mathematical object for describing systems is given by the model of dynamical systems. In the general case, a dynamical system is given by $\mathcal{H} = (X, f)$, where $X$ is a space, and $f : X \to X$ a transition function. In discrete time systems, trajectories correspond to iterations of transition function $f$, and in continuous time systems to solutions of ordinary differential equation $x' = f(x)$.

I focus on systems that may be continuous in several manners: considered systems have at least their space continuous, i.e. say space $X$ is the set of real numbers $\mathbb{R}$, or Euclidean space $\mathbb{R}^n$ of dimension $n$. I then consider both discrete and continuous time systems. This last possibility, that introduces another aspect of continuity, leads to differential equations and their computational properties.

Several points of view One first interest of the dynamical system model is its modeling power. Dynamical systems have indeed proved their interest for the modeling of many systems in physics, biology, mathematics, and more generally in applied sciences. From this general point of view, understanding theories of computations for continuous systems provides a way to understand algorithmic complexity of natural
problems for the modeled systems. For example, this gives a way to understand algorithmic complexity of verification and control of systems.

From a more restrictive point of view, many continuous machines or devices can be modeled by dynamical systems. Discussing theories of computation for dynamical systems gives then ways to understand the computational power of those machines.

We detail these points of view now.

“Machine” point of view This is this latter point of view, more restrictive, and centered on computational machines, which led to the first developments on theories of computations for continuous systems: do not forget that first ever built universal machines were analog, and not-digital.

For example the first model of a universal machine is the model of the General Purpose Analog Computer (GPAC) of Claude Shannon [?]. The GPAC was a model of the Differential analyzer, which was a mechanical machine of that time, built for the first time in 1931 [?]. In a more recent point of view, the GPAC can be proved to be a faithful model of what can be computed by today's analog electronics [?, ?].

Continuous models also include formal neural networks models [?], models from verification [?, ?], and models arising from distributed computing, about which we will come back. All these models are continuous time models. Continuous space discrete time models include recursive analysis [?], or Blum Shub Smale computational model [?]. Continuous models can also be presented as being between classical discrete computational models and quantum models.

In the classical discrete world, Church thesis postulates that what is calculable, in an intuitive meaning, is computable by a Turing machine. This thesis is often confused with another thesis, sometimes called Church physical thesis, called thesis M in [?], which postulates that what can be calculated by a physical machine must be computable by a Turing machine. A third thesis, less formal, would say that all what can be calculated by a mathematical model must be computable by a Turing machine. Observe that the first thesis, is historically about the power of formal deduction, at least in original paper [?], and that the second talks about physical machines, and thus about of our physical world. The third talks somehow about mathematical models of our physical world. See [?] for discussions and a very beautiful demonstration of the historical semantic slip from first thesis to other theses. One can say that three theses have to do with what one considers to be “reasonable” in these fields.

By the existence of these theses, the situation for discrete models is relatively well understood, since as soon as a model is sufficiently powerful, it is equivalent to Turing machines. These theses are well admitted partly because there are many equivalence results between models.

For continuous systems, the situation is far from being also clear. There is no equivalent of Church thesis, or of these theses. There are in particular very few comparison and equivalence results between continuous models. The situation is clearly more delicate than in the discrete world: for example, functions computable in recursive analysis must be continuous, whereas simple discontinuous functions are computable in the model of Blum Shub and Smale. Such results seem to make Utopian any result of equivalence for continuous discrete time models. For continuous space and time models, we think that that is perhaps possible, in particular following some of our recent results: see for example [?] and following discussions.

In any case, understanding properties of computational models helps to understand what must be considered as reasonable for continuous systems. That contributes to understand limits of our formal deduction methods for continuous systems, in relation with first thesis, limits of physical machines, in relation with second thesis, and limits of current models of our physical world, in relation with third thesis. Several results obtained at this date for continuous systems indeed point out very simple, paradoxical or philosophical points related to each one of these aspects.

“Continuous System Theory” point of view In a more general point of view, one does not restrict to continuous systems that necessarily correspond to computational machines. Discussing theories of computations for continuous systems allows then understanding algorithmic complexity of natural problems for these systems.
The spectrum of systems that can thus be studied is enormous, since most systems of this world can be modeled by dynamical systems.

On the level of the properties, I restrict mainly to properties such as reachability properties: “given a dynamical system $H$, a starting point $x_0$, and a subset $X' \subseteq X$, determine if there is a trajectory starting from $x_0$ which reaches $X''$, and stability properties: “given a dynamical system $H$, decide if all trajectories go towards the origin”.

These decision problems are of first interest in many fields, like verification and control theory. For example, in verification, discussing the algorithmic complexity of the first problem allows to understand the algorithmic complexity of verification of safety properties, since a system is safe, if and only if there is not trajectory starting from an initial state which reaches the subset of the non-correct states [?]. One just need to evaluate the volume of literature these last years about the frontier between computability and non-computability for verification of hybrid systems to be convinced of the practical interest of this question: see for example [?, ?, ?, ?, ?, ?, ?].

Discussing algorithmic complexity of problems for dynamical systems have impacts well beyond verification and control theory: for example, for VLSI design (see for example [?]), neural networks [?], the study of chaos in dynamical systems (see for example [?]), optical models (see for example [?]).

Historically, the model the most developed to discuss computability and complexity of problems over real numbers is recursive analysis. It was introduced by Turing [?], Grzegorczyk [?], and Lacombe [?]. It aims at discussing effective constructions in mathematics and it gives clues to understand algorithmic complexity of many constructions in mathematics: see [?] for a recent monograph.

But recursive analysis approach turns out to be not relevant to discuss problems mentioned above. For example, on can show that reachability is undecidable, but actually even for systems whose dynamics is $y' = 0$, i.e. for systems without dynamics: see for example [?]. What is revealed by such an uncomputability proof is not really something in relation with the dynamics of the considered dynamical systems, but mostly a discussion related to coding problems. Somehow, many results of non-computability for dynamical systems are in this vein [?]. One can term these results “static undecidability” results, following the terminology of [?].

A more relevant alternative is as follows. Since a machine of Turing (or any other discrete model of computations) can be considered as a discrete dynamical system, if one succeeds to embed this dynamical system into a continuous dynamical system, one obtains immediately that the trajectories of the continuous dynamical system have at least all the richness of the behaviors of the embedded system. This leads to “dynamic undecidability” results, which allow to discuss in a really finer manner algorithmic complexity of problems for dynamical systems. For example, such approaches were used to prove results of uncomputability for dynamical systems in [?], for electronics in [?], for neural networks in [?], for verification in [?], for control theory of hybrid systems [?], or to study computation theory for continuous systems [?].

Producing classes of systems which are robust, realistic, and which avoid the phenomena of undecidability is of first importance. That allows in particular to guarantee the termination of automatic model-checking verification techniques [?].

Several dynamical systems models experience hyper-computation phenomena: For example, not only Turing machines can be embedded in piecewise constant differential equation systems [?], but actually also accelerating Turing machines [?]. This implies that the verification of safety properties for these systems is hard for all levels of the arithmetic hierarchy [?]. This type of phenomena occurs for many other classes: see for example [?], [?], [?], [?] for results of hyper-undecidability of the same type for other mathematical models, and for example [?] for such phenomena in models from theoretical physics.

In every case, these models can be accused to be non-robust, non-realistic, far from physical reality, and on can claim that such phenomena do not occur in “reality” [?, ?, ?]. It is then important to understand whether uncomputability (and hyper-uncomputability) results remain true for robust systems. The folk conjecture that realistic models of noise prevent undecidability on bounded domains is not clear: see for example the articles [?, ?, ?, ?, ?, ?, ?], that consider various models of noises (probabilistic, not-deterministic, etc) but with contradictory results.

Even if one restricts to problems known to be decidable, time and space contraction phenomena make very
difficult the discussion of complexity problems for continuous systems. For example, if there are computability theories for continuous systems, there are only preliminary works about complexity theories [?, ?], and the complexity of decision problems for continuous systems is largely left open. Understanding the algorithmic complexity of natural problems for continuous systems is of first importance, in particular to discuss efficiency of verification and control theory methods for hybrid and continuous systems.

Complements This presentation is of course extremely short. With Manuel Campagnolo, I recently wrote a survey [?] on computation theories for continuous-time systems. I suggest to my reader to refer to it for a better understanding of all above mentioned points. This survey is also chapter 3 of my habilitation. You can also refer to chapter 1 and 2 of my habilitation for all points related to motivations.

3 Three examples of systems

In order to make our discussion not too much abstract, we will now present three classes of continuous systems, on which we worked, with the risk of over-simplifying the discussion.

A model of analog electronics It is well known that one can realize an integrator circuit using an ideal operational amplifier, a condenser and a resistance: the voltage at the output of the integrator is the integral of the voltage at the input.

If one can conceive that this is possible to realize an analog circuit that realizes an integration in real time, one can easily conceive that this is possible to associate several integrators to build a circuit that will generate function \( y(t) = \cos(t) \) in real time \( t \): one just need to impose, through a feedback connexion, that \( y \) is equal to the opposite of the integral of its integral.

Understanding functions that can be generated by such analog circuits, leads to the General Purpose Analog Computer (GPAC) model from Claude Shannon [?], whose basic units are not only integrator units, but also constant units, adder units, and multiplier units. GPAC model was introduced as a model of machines at that time, in particular of the differential analyzer [?]. Of course, in 1941, there were no operational amplifiers, but the differential analyzers were made of basic blocks that carried out these elementary operations, and the GPAC model can be proved to remain a faithful model of what is computable (generated) by today’s analog electronics [?, ?].

The characterization from [?] that claims that a function is GPAC generated iff it is differentially algebraic is incomplete, as observed in [?, ?, ?]. For the more robust class of GPAC considered in [?], the following stronger property is satisfied: a scalar function \( f : \mathbb{R} \to \mathbb{R} \) is generated by a GPAC if and only if it is a component of a solution of an ordinary differential equation system \( y' = p(t, y) \) where \( p \) is a vector of polynomials. We will come back this type of systems, that we call polynomial differential equations.

A model from verification Another model that I considered in my work is the model of piecewise constant derivative systems from [?]. The principle of the model is extremely simple: one considers continuous time dynamical systems \( x' = f(x) \), where function \( f \) is piecewise constant, that is constant on a finite number of polyhedral regions.

This model was proposed in [?] as a model for hybrid systems, i.e. for modeling systems that intermix continuous dynamics with discrete transitions.

This model, coming from verification, has several pedagogical virtues.

It shows that hybrid systems lead naturally to ordinary differential equations with discontinuous right hand side. Mathematical theory usually assumes continuous (and even Lipschitz) right hand side: see e.g. [?]. The natural concept of solution in this model [?] is different from most usual considered definitions for discontinuous differential equations [?].

It exhibits time and space contraction phenomena (called Zeno paradox in the verification community): accelerating Turing machines can be simulated in finite time, as soon as the dimension is sufficient [?, ?]. More generally, it also points out some paradoxical aspects of current models of our physical world [?, ?]. This model also allows discussing complexity of problems according to models of noise or robustness [?, ?].
A model from distributed computing  We think that theories of computations for continuous systems can help to contribute at a significant degree to the understanding of some recent models coming from distributed computing for sensor and telecommunication networks.

For example, let us mention the model of passively mobile sensor networks from [?] whose motivation is as follows. In several applications related to sensor networks, this is natural to assume passive mobility: one controls the programs in each sensor, but one does not control the interactions between sensors: consider for example, sensors that would be joined to access cards given to users. The model of population protocols, proposed to model such sensor networks [?], is now discussed in most selective conferences of distributed computing [?, ?, ?].

From the point of view of computation theory, a characterization of relations that are computable in this model was recently obtained [?]: that corresponds precisely to relations that are definable in Presburger arithmetic. In addition to this non-trivial technical result, this shows that this model is really different from traditional models from distributed computing, as well as from today’s models from computability and complexity.

The model of [?] is a discrete time model: at each round, a pairwise interaction occurs between agents. It is rather natural to consider an asynchronous version where interactions occur according to Poisson processes. The model becomes a way to describe (enormous) continuous time Markov chain, instead of (enormous) discrete time Markov chain. For the continuous time version, the computational power of the model remains open.

From the point of view of models, that shows that models from dynamics of populations, coming from biology, seem now relevant for understanding distributed computing.

More generally, continuous dynamical systems, and thus theories of computations for continuous systems, appear naturally as soon as the number of agents becomes large: indeed, as this is well known in applied sciences, it becomes relevant to go from a microscopic vision to a macroscopic description of systems, where one describes the state of the system by certain continuous parameters.

This approach, that can be termed the thermodynamic limit approach, is definitely not usual in distributed computing, at least to study their complexity and computability. One of the reasons is that for standard and traditional models of distributed computing, it is often non-relevant because of absence of space homogeneity. It becomes relevant in new models like population protocols [?] because of the natural assumption of passive mobility.

Considered models become very close to models from biology of populations, or to models from evolutionary game theory. Theories of computations for continuous systems give ways to understand their power.

4 Some personal results

Previous discussions show that understanding theories of computation for continuous systems, as well as understanding algorithmic complexity of natural problems for those systems, lead to compare models. Indeed, as we said, dynamic undecidability results are often based on embedding, and hence also on comparisons.

We now point out some selected personal results, which go in the sense of this research program.

Preamble: how to compare models  There are several ways to compare models.

1. The ideal case is when one succeeds to prove the equivalence of models, or the inclusion of models: one succeeds to prove that any function computed by a first model is computed by a second. This includes for example the results in [?] on the General Purpose Analog Computer that we proved equivalent to recursive analysis.

2. Unfortunately, this is not always possible to obtain so strong results, and a weaker way to compare models consists in discussing discretizations, in particular when one wants to compare a model of
computation on reals with a discrete computational model: one shows that the discretizations of the functions computed by the continuous model are computable by the discrete model. This includes for example the results in [?].

3. A third way, even less strong, consists in generalizing known results in classical complexity and computability to a continuous model of computation. This includes for example our results characterizing à la Bellantoni and Cook [?] all major complexity classes in the Blum Shub Smale [?], extended by Poizat [?], model of computation, over an arbitrary logical structure.

4. Finally, a fourth way, even less strong somehow, consists in discussing the algorithmic complexity of natural problems for considered systems: for example by discussing completeness and hardness results about the algorithmic complexity of reachability (verification) problems. This includes for example the results of [?].

We present now with more details, some results, going from approach 4. towards approach 1., i.e. in the sense of increasingly strong results.

Completeness Results Concerning last approach, we presented in [?], also in appendix A of my habilitation, a discussion of the algorithmic complexity of the reachability problem for various classes of dynamical systems. With various hypotheses on transition function \( f \) (continuous, analytical, computable, . . . ), we characterize the hardness of reachability problems for associated discrete and continuous time dynamical systems.

My thesis contains also a whole set of results in this direction [?].

Approach 4. also include results in [?, ?], about computability/non-computability of several stability problems, and results about decision of matrices problems discussed in [?].

Characterizations à la Bellantoni-Cook 92 Concerning third approach, I was interested for example in generalizing characterizations of complexity classes à la [?] in classical complexity to Blum Shub Smale model [?], via the co-supervision of the PhD thesis of Paulin Jacobé de Naurois.

The results of [?] shows that, if in primitive recursion schemas two types of arguments are distinguished, safe and normal arguments, with the conventions that a normal argument can become safe, but not vice versa; that parameters of recurrence must be in normal position, and that values of recurrence must be in safe position, then traditional polynomial time is captured: any function definable by such schemas is computable in polynomial time, and conversely any polynomial time computable function is definable by these schemas.

Similar characterizations were later on obtained for other complexity classes, like polynomial space [?], non-deterministic polynomial time [?], levels of the polynomial hierarchy [?], and parallel classes [?, ?].

Through the co-supervision of Paulin de Naurois’s PhD thesis, I was interested in generalizing these results to the BSS model of computation. This model of computation was proposed by Blum Shub and Smale in 1989 in [?] as a model to measure the algebraic complexity of problems on reals. It was generalized thereafter to a model of computations on an arbitrary logic structure by Poizat in [?]. The principle is to consider Turing machines whose tapes can contain elements of the underlying domain of the structure, and who can carry out the operations of the structure, as well as test relations of the structure at unit cost. Classes like polynomial time, non-deterministic polynomial time, polynomial hierarchy, alternating polynomial time, etc can then be defined [?].

We proposed characterizations of all major complexity classes in this model in terms of recursion schemas à la Bellantoni and Cook 92. We characterized polynomial time in [?], polynomial parallel time in [?], levels of the polynomial hierarchy in [?], alternating polynomial time in [?].

Since traditional complexity can be seen as the restriction of complexity in this model to finite structures [?, ?], these results generalize classical characterizations to arbitrary structures. For some of the classes, these results offer stratifications of the classical results: for example we characterize alternating polynomial time, digital alternating polynomial time, and polynomial parallel time on an arbitrary structure, whereas
for finite structures all these classes correspond to polynomial space. Let us observe that it is not possible
to define in the general case polynomial space in the model of Blum Shub and Smale [?].

These results provide strong links between computations over the reals, or over arbitrary logical structures,
and classical logic. From the programming point of view, a way to interpret all these results is to see
computability over an arbitrary structure as a programming language with extra operators coming from an
external library. This observation and its potential for the construction of methods, to derive automatically
properties of programs, in the spirit of [?, ?, ?], is also one of the motivations of this work.

\textbf{R-recursive functions} Concerning the second approach, which consists in discussing the discretization of
computable functions, I obtained a whole set of results in connection with the classes of R-recursive functions
introduced in [?], through the supervision of the PhD thesis of Emmanuel Hainry.

The original paper of Moore [?] presents extremely interesting and very original ideas to understand
computations on reals, which can be presented in the following way: since there is no universally accepted
concept of machine in the continuous world, why not circumvent the problem using machine independent
characterizations of classes in classical computability and complexity.

The idea of [?] is indeed simple: observing that the analog of a schema of primitive recursion in computability
is a definition by an ordinary differential equation (a schema of integration), and that computable
functions in classical computability correspond to the smallest class of functions which contains certain basic
functions and which is closed by composition, primitive recursion and minimization, Cris Moore proposed to
focus on the class of functions, called R- recursive, containing basic functions, closed by composition, inte-
gration and minimization. Unfortunately, the hierarchies suggested by Moore suffer from several problems,
that can be corrected by replacing the minimization schema by a limit schema [?, ?].

We were interested in an alternative approach which consists in considering R-primitive recursive func-
tions, as in [?, ?]: Campagnolo, Moore and Costa proposed to consider a class \( \mathcal{L} \) which is defined from
certain basic functions, and closed by composition and linear integration, rather than by a general inte-
gration schema. They major result is that the discrete part (discretization) of the functions of this class
Corresponds to the class of elementary functions of classical discrete computability. This result was general-
ized later on by same authors for other classes of classical computability, up to primitive recursive functions
[?].

With my PhD student Emmanuel Hainry, we proved that it is possible to go further and go up to
computable functions. We defined a rather natural minimization schema, so that the smallest class \( \mathcal{H} \)
containing certain basic functions and closed by composition, (controlled) linear integration, and this schema
of minimization has its discrete part which corresponds to computable functions [?].

We went then much further, because we proved that it was possible to define a natural limit schema, so
that the addition of this limit schema to class \( \mathcal{H} \) presents a characterization of computable functions from
recursive analysis over the reals, for functions of class \( \mathcal{C}^2 \) over a compact domain [?]. Papers [?, ?, ?] generalize
also these results to the levels of the Grzegorczyk hierarchy, and to elementarily computable functions in
recursive analysis.

In other words, we show that computable functions from recursive analysis correspond to the smallest class
of functions containing certain basic functions, and closed by composition, linear integration, minimization
and a limit schema.

These results relate in a fine way two à priori distinct computational models over the reals: recursive
analysis and R-recursive functions.

But mainly, to our knowledge, this is the first time that computable functions of recursive analysis are
characterized in a machine independent way, through analysis. In particular, this has a high pedagogical
value since this shows that one can define these classes without discussing type 2 or higher order Turing
machines.

\textbf{The General Purpose Analog Computer vs Recursive Analysis} Concerning the first approach, we
proved that the GPAC of Claude Shannon (and thus analog electronics) has the same power as recursive
analysis (and thus somehow as Turing machines) in [?].
Since results of Shannon in [?], it was believed that the GPAC was a model weaker than digital ones, and in particular than Turing machines, and recursive analysis. Indeed, Shannon claims in [?] that functions generated by GPAC must be differentially algebraic. The existence of functions, like Γ Euler function, or ζ Riemann functions, that are computable in recursive analysis but not differentially algebraic [?], was historically used to argue on the weakness of the GPAC model, and more generally on the weakness of analog models of computations compared to digital ones. Remember the long historical polemic between partisans of digital versus analog models [?, ?].

However, with Manuel Campagnolo, Daniel Graça, and Emmanuel Hainry, we proved that is actually more due to the concept of GPAC-generation rather than to a limitation of the GPAC. Indeed, in the concept of computation considered in [?], it is required that functions must be computed in real time, whereas in the model of the recursive analysis, an arbitrary time to compute a function is allowed.

In particular, we were interested in the question because Daniel Graça proved that Γ function is in fact GPAC-computable if one takes a concept of “convergent computations” similar to the one considered in recursive analysis.

We proved that if one does not restrict to real-time computations in GPAC, i.e. not to GPAC generated functions, but one considers convergent computations as in recursive analysis, then a function over the reals over a compact domain is GPAC computable iff it is computable in the sense of recursive analysis [?].

Notice that somehow, that shows that current digital models, including Turing machines, are in fact completely equivalent to analog models, and even to 1931 machines [?, ?], at least from a theoretical computability point of view.

5 Perspectives

I present now some perspectives, in the form of research objectives for the next years about this research program.

The fourth objective, is more recent, and also more speculative, but corresponds to a direction that I am currently trying to insufflate to my work.

5.1 Objective A: Understand whether there may exist a unifying concept for continuous time systems similar to Church thesis

The situation is far from being as clear as for discrete models. Although it was shown that some continuous time models have some hypercomputational properties, all these results are based on the use of a certain infinite quantity of resources, like time, space, precision or energy.

It is often then claimed that “reasonable” continuous time models can not compute more than Turing machines [?, ?, ?, ?]. Actually, this is not so easy to prove: see all discussions in the survey [?]. Personally, since robust continuous-time analytical systems can provably simulate Turing machines over an unbounded space [?], I think that analog and digital computations are equivalent from the computability point of view.

Actually, following several recent results, in particular the proof of the equivalence between projections of solutions of polynomial differential equations, GPAC computable functions, and computable functions in the sense of recursive analysis, we think that there may exist a unifying concept for continuous time computations similar to what exists in classical computability theory.

Recall that classical computability deals with functions over the integers or over an alphabet, i.e. over discrete spaces, whereas continuous time computability is about functions over the reals. Classical Church thesis never claimed to cover computability over the reals, or computations by continuous time dynamical systems.

We have a candidate to be this unifying concept for continuous time systems: (projections of solutions of) polynomial differential equations.

A first argument is the universality of the model, from a modeling power point of view. Indeed, for example, all the examples considered in monograph [?], who is a reference for the theory of continuous time dynamical systems, or in monographs like [?, ?], that present an overview of models used in biology
and in physics, correspond to polynomial differential equations, or can be shown to be equivalent to higher
dimensional polynomial differential equations. This include Lorenz system, Lotka-Volterra prey-predators
systems, or epidemiological models.

Another argument is that this class of functions has strong stability properties. For example, functions
projections of solutions of polynomial differential equations are stable by addition, subtraction, multiplication,
and division, as well as by composition [?]. In addition, one can show the following strong result: any
system \( x' = f(t, x) \), where \( f \) is the composition of polynomials and of functions in the class, is equivalent to
a higher dimensional system \( x' = p(x, t) \) where \( p \) is a vector of polynomials [?]. This can be used to prove
very quickly that many ordinary differential equations can be turned to (higher dimensional) polynomial
differential equations.

These arguments show the universality of the model, from a modeling power point of view. One expects
such a result in the spirit of a Church thesis: in classical computability, the concept of computable function
subsumes the concept of computations by majority of natural models.

To have a concept like Church theses, one also expects strong links with concepts of machines. We
already said that solutions of polynomial differential equations correspond to what is computable by General
Purpose Analog Computer (GPAC) of Shannon, i.e. by machines of the time of Shannon (e.g. Differential
analyzers), but also by modern analog electronics circuits.

In addition, one may expect continuous-time computability to be related to classical computability.
Previous results show also that there are some strong links between the two: an unary real function over a
compact domain is computable in the classical sense (recursive analysis) if and only if it corresponds to a
projection of a solution of a polynomial differential equation [?].

There is thus indeed a natural, and relevant candidate for a Church thesis and all its variants for contin-
uous time systems. If there is no unifying concept like Church thesis for continuous time systems, all these
results show at least that polynomial differential equations are natural counterparts to Turing machines.
Better understand seems vital, from the point of view of theories of computations in general, but also to
understand models of our physical world, and the power of machines of our physical world.

5.2 Objective B: Understand if there is a nice and well-founded complexity
theory for continuous time systems

Up to this date, most attention in literature about theories of computations for continuous time systems is
on computability, rather than on complexity: see surveys[?, ?].

The obstacles to go towards a complexity theory are known: on one hand, there are time and space
contraction phenomena that make null and void any simple definition of a complexity class for continuous
time systems. Indeed, in the general case, any computation which can be carried out in a given time, can
be carried out in unit time, by using a space or time contraction, i.e. by a suitable change of variable: see
for example [?, ?, ?] for incarnations of this phenomenon on mathematical models, and [?] on models of
theoretical physics.

Another obstacle is the low number of proved equivalence results. Indeed, what legitimate polynomial
time in classical complexity is the fact that many models can be proved equivalent modulo a polynomial-time
slowdown in the worst case. For continuous time models, there are only a few comparison results.

In survey [?], we discuss rather deeply difficulties that arise when trying to build a complexity theory for
continuous time systems.

Several directions have already been experimented, but at this date there has not been an agreement
between authors on basic definitions such as computation time or input size. Results are either derived from
concepts which are intrinsic to the continuous time systems under study or related to classical complexity
theory.

One can thus distinguish the theories developed for general systems, where one tries to define a concept
of time valid for computations by differential equations, and theories defined for dissipative systems, like
Hopfield neural networks [?], where one restricts to asymptotically stable systems, and where concepts
of complexity close to what is considered in circuit complexity theory in classical complexity are mainly
considered. First approach is very natural in verification or in control theory, whereas second approach is more natural for neural networks models for example [7].

For general systems, rather than formulating the problem as a real computability theory problem, following [7], perhaps this is better to formulate it in terms of a problem connected to numerical methods for solving ordinary differential equations. It is well known that the error made by Euler’s method for solving differential equation $y' = f(y)$, $y(0) = x$ at precision $\varepsilon$ on $[0, T]$, assuming that round off errors are bounded by $\sigma$, with $N$ steps is given by

$$||y(T) - y_N^*|| \leq \frac{h}{\lambda} \left[ \frac{R}{2} + \frac{\sigma}{h^2} \right] (e^T \lambda - 1),$$

where $y_N^*$ is the approximation after $N$ steps, $h$ is the step size, $\lambda$ is the Lipschitz constant for $f$ on $[0, T]$, and $R = \max\{||y''(t)||, t \in [0, T]\}$. From this bound, one can easily deduce that the number $N$ of steps to simulate the system is polynomial in $R$ and $\frac{1}{h}$, but exponential in $T$ [7]. This exponential dependence in $T$, is an obstacle to define the time of computation by a continuous time system as variable $t$: indeed, there is an exponential slowdown between computation by the differential equation, and its simulation by numerical methods.

One can think that Euler’s method is too basic, but in fact, the same phenomenon of exponential dependence in $T$ holds in all known numerical methods in numerical analysis [7]. By adding ad hoc conditions (for example assumptions on the growth of the solutions), one can build a method that eliminates the exponential dependence in $T$, but at the price of non-natural definitions [7].

Is it possible to obtain nicer results, or is this exponential dependence in $T$ inherent to numerical methods? In the first case, this would lead to faster numerical methods for solving ordinary differential equations. A complexity theory for general systems would then perhaps be possible. In the second case, that would be the proof that continuous time systems can carry out (at least in theory) some computations faster than classical discrete models.

Here are some research directions about objective B in relations with my personal results.

First of all, in relation with our results which link computable functions in the sense of recursive analysis with $\mathbb{R}$-recursive functions, we think that a way to tackle the problem is to try to characterize polynomial time computable functions over the reals as a subclass of $\mathbb{R}$-recursive functions. Indeed, in recursive analysis, it is possible to define a concept of complexity: one asks that approximations of reals at precision $2^{-n}$ must be computable in time polynomial in $n$. It is perhaps possible to characterize algebraically polynomial time computable functions over the reals, using known characterizations of polynomial time computable functions over the integers in classical complexity, like [7, 8].

Notice that previous discussion on the time required to solve an ordinary differential equation by numerical methods, lets think that time parameter $t$ behaves in a way different from other parameters. Perhaps it is then natural to indeed copy the idea of Bellantoni and Cook [7] of distinguishing two types of arguments, safe and normal arguments, when building a complexity theory for continuous time systems.

We proved that GPAC computable functions correspond to computable functions in the sense of recursive analysis. The simulation given by the proof seems to yield a strong connexion between the time of the computation by the GPAC and the time of the corresponding Turing machine. Can one establish such an equivalence at the level of complexity, and not only at the level of computability? For example, by considering that in the concept of computability by GPAC, the error must be given by the inverse of a polynomial function in the parameter $t$?

Finally, if the class of functions solutions of polynomial differential equations allows to characterize in an elegant way the concept of “reasonable” continuous-time computation, as we conjecture above, maybe it would be natural to formulate a theory of complexity based on these functions.

Of course, this work must be carried out in relation with existing literature to date on the subject. For dissipative systems, Gori and Meer proposed in [7] a framework to discuss the minimizers of a Lyapunov function $E$. Actually, Gori and Meer’s proposed framework is rather abstract, avoiding several problems connected to what one might expect of a true complexity theory for continuous time computations. Nonetheless, it has the great advantage of not relying on any particular complexity measure for the computation of trajectories.
However, one would like to understand the complexity of approaching the minima of energy functions, which correspond to the equilibria of dynamical systems. First steps toward this end have been investigated in [?], where dissipative systems with exponential convergence are explored. However, on the one hand the extension to general dissipative systems is not clear, and on the other hand, the only example which seems manageable at to date by this framework seems to be the flow suggested in [?] to solve linear programming problems, studied more deeply in [?] and [?].

An alternative to all these approaches is given by the approach of Costa and Mycka [?]: even if it is not possible to define a complexity theory for continuous systems, perhaps it is possible to transfer major problems of classical complexity into continuous systems. In particular, Costa and Mycka propose two classes of real recursive functions such that their inequality would imply $P \neq NP$ [?], [?].

5.3 Objective C: Better understand the effects of noise and unprecisions on computations

The works which we carried out up to date, as well as most of the literature on theories of computations for continuous systems disregard the impact of noise and unprecision on computations in continuous time systems. That constitutes a recurrently criticized and pointed weakness of the research in the field [?, ?].

Notice that considering computations in unbounded precision is rather similar to consider Turing machines with infinite tapes: the construction of machines with infinite tapes is not possible, like the construction of machines with infinite precision.

The obstacles to understand easily the effects of noise and unprecisions are known. First, there are several models of noise, and not fully understood relations between models.

Second, there are some recurrent conjectures, but sometimes with contradictory results.

For example, it was conjectured on several occasions (it is a "folk-conjecture" in the field of verification for hybrid systems for example) that undecidability do not hold for “realistic”, “unprecise”, “noisy”, “fuzzy”, or “robust” systems. See for example [?], [?] for various statements of this conjecture, and [?] for discussions on various arguments that lead to this conjecture.

There is no consensus on what a realistic noise model is. A discussion of this subject would require to question what are good models of the physical world. In the absence of a generally accepted noise model, one can however consider various models for noise, unprecision or smoothness conditions, and investigate the properties of the resulting systems.

A first approach is to consider probabilistic models of noise. This is the most usual approach in the neural network community. In [?], Maass and Orponen proved that the presence of bounded noise reduces the power of a large set of discrete time analog models to that of finite automata. This extends a previous result established in [?], [?]. However, for any common noise, such as Gaussian noise, which is nonzero on a sufficiently large part of the state space, systems are unable to recognize even arbitrary regular languages [?]. They recognize precisely the definite languages introduced by [?], as shown in [?] and [?]. If the noise level is bounded, then all regular languages can be recognized [?]. Somehow, these results seem to prove that one is still facing over-modeling problems, since even finite-state machines cannot be simulated with a Gaussian noise.

Another approach consists in modeling noise by non-determinism. This is the most usual approach in the verification community.

Fränzle defined in [?] an ad hoc concept of robustness and proved that this concept of robustness implies decidability for verification of safety properties (i.e., of the reachability problem). Later on, Asarin and Bouajjani proved in [?] that, if robustness is defined as not-sensitivity of the reachability relation by infinitesimal disturbances, then robustness implies decidability. All these results seem to go in the direction of the conjecture above.

But unfortunately, there is a whole set of results which go precisely in the contrary direction. A first example is that safety of a system is still undecidable if the transition relation of the system is open, as proved in [?], and [?]. Gupta, Henzinger and Jagadeesan consider in [?] a metric over trajectories of timed automata, and assume that if a system accepts a trajectory, then it must accept neighboring trajectories.
also. They prove that this notion of robustness is not sufficient to avoid undecidability of complementation for Timed automata. Henzinger and Raskin prove in [?]' that major undecidability results about verification of hybrid systems are still undecidable for robust systems in that sense. Finally, Asarin and Collins present in [?] a model of Turing machines exposed to a small stochastic noise, whose computational power have been characterized to correspond to \( \Pi^0_1 \), i.e. to non-computable relations.

Globally, up to this date, existing literature on the effects of noise and unprecisions, is mainly restricted to discrete time systems [?]. Extensions to continuous-time systems seem possible but remains to be made formally for most of mentioned results.

We propose to contribute to understand the frontier between decidability and non-decidability according to the considered model of noise. The underlying question is to understand if uncomputability results remain true for robust systems, and understand which models of noise and robustness lead to decidability. This is of first importance in the field of verification for example, since this question is strongly connected to the question of termination of automatic verification methods: if above conjecture is true, then the verification of realistic systems (of robust systems) would always be possible.

Moreover, up to this date the question was only addressed at the computability level, and not at the complexity level: for problems proved decidable (for example bounded time reachability), what is the influence of the concepts of noise on the complexity of problems? How does it grow with the precision, or the noise? A framework for that was proposed in [?] but up to this date the study remains completely to be made.

We think these questions at the heart of fundamental problems connected to very deep questions: how to model noise? What is a robust model? How to model uncertainty? How much is that relevant to discuss with formal models, and hence with certainty, models of noise and unprecision, and hence of uncertainty?

5.4 Objective D: Understand some new models (e.g. sensor and telecommunication networks) with theories of computations for continuous systems

The size of some distributed systems, like today’s networks, is now so huge that it seems necessary not to speak any more about individuals but about statistics, averages, and proportions. For example, when one describes the state of a network like the Internet, it is not relevant to describe the state of all individuals, but rather to describe the proportion, or numbers of individuals in such or such state. The instantaneous state of the system becomes better described by continuous macroscopic parameters than by its microscopic description in terms of individuals. The evolution of systems is then given by dynamics describing the evolution of these continuous macroscopic parameters.

We propose to work on the one hand on the microscopic/macroscopic approximation and its validity for distributed computing, and on the other hand on the computational properties of the obtained models. The global aim is to discuss with the tools from theories of computations, some new models from distributed computing and their computational properties.

The systems about which we propose to focus are those which lead naturally to dynamics close to models of biology of populations [?], or evolutionary game theory models [?, ?]: see [?] for example for discussions of the hypotheses that lead to these dynamics in applied sciences. The motivation is that these assumptions are naturally present in many new distributed systems, and hence that the macroscopic approximation becomes relevant.

The approach which consists in going, for a population of agents in interactions, from a microscopic vision, where one describes the behaviors and the interactions of each agent, to a macroscopic vision, where one describes the evolution of the population by global macroscopic parameters, is common in applied sciences, like physics, chemistry, and biology, etc [?, ?, ?].

This approach, that can be termed the thermodynamic limit approach, is definitely not as usual in distributed computing.

More precisely, the macroscopic vision is common for the description of distributed systems, like networks, and possibly of their dynamics, but it is definitely less usual for the programming and the discussion of their computational properties (complexity, computability, theory of computations).

One reason is that in many standard or classical models used in distributed computing, there is not
a space homogeneity, and hence the macroscopic approach is not-relevant. However, in several recently suggested models from distributed computing, in close links with some new applications and new systems, this approach become relevant.

For example, in the model of sensor networks of \[?\], already mentioned, the assumption of passive mobility (i.e. agents or actors do not control with which they will enter in interactions), implies that the system become naturally described by a macroscopic dynamic. Even if the model is completely deterministic, with local rules, the assumption of passive mobility implies a space homogeneity, which legitimates a macroscopic description. Somehow, these systems are programmed and discussed at the macroscopic level \[?, ?, ?\].

Whereas, all classical models from distributed computing are close to Turing machines or to cellular automata from the computability point of view \[?\], these new models are really different.

This can be observed by their computation theory: the computational power of the population protocol model of \[?\] has been characterized in terms of relations definable in Presburger arithmetic \[?\]. This corresponds to no known classical complexity or computability class, hence demonstrating that the model is really distinct from classical ones \[?, ?, ?\].

Many alternatives of the model are possible and natural, in particular in relations with applications: for example, as we said, for sensor networks, asynchronous pairings would be more natural. For all these alternatives, the power of the obtained model remains open.

To illustrate our matter, let us consider an example, which is an alternative of the previous population protocols: one considers a population of agents, in huge number. Each agent is in one out of four states. These agents enter in pairwise interactions according to following rules:

\[
\begin{align*}
\delta(q_1, q_2) &= q_2 \\
\delta(q_2, q_1) &= q_2 \\
\delta(q_3, q_2) &= q_3 : 1/2, q_4 : 1/2 \\
\delta(q_4, q_3) &= q_3 : 1/2, q_4 : 1/2
\end{align*}
\]

For example, first rule means that an agent in state \(q_1\) meeting an agent in state \(q_2\) becomes in state \(q_2\), and that this interaction occurs according to a homogeneous Poisson process of rate \(\beta\). The last rule means that an agent in state \(q_3\) meeting an agent in state \(q_4\) becomes in state \(q_3\) with probability \(1/2\), and \(q_4\) with probability \(1/2\), and that this interaction occurs according to a homogeneous Poisson process of rate \(\nu\).

By a classical reasoning from applied sciences, the macroscopic dynamics of the system, when the number of agents is large, is given by continuous dynamic

\[
\begin{align*}
S' &= -\beta SI \\
I' &= \beta SI - \nu I \\
R' &= \nu I
\end{align*}
\]

where \(S, I, R\) denote respectively the proportions of agents in states \(q_1, q_2, q_3\). If the reasoning to go from the microscopic dynamics to the macroscopic dynamic is easy, the mathematical proof of the validity of the transformation is not so: see \[?\] for a general discussion of the used assumptions, and for example \[?\] for the formal proof on this example.

Above equations are those of the SIR model of diffusion of a contagious illness, sometimes called the Kermack-McKendrick model \[?, ?\]. The important parameter is then the epidemic rate \(R = \beta S/\gamma\). If this threshold is strictly lower than 1, the infection will disappear. If it is strictly higher than 1, the infection will be spread without limits.

One thus sees that macroscopic dynamics is a game, in the sense of the game theory, of a fight between infected agents, in state \(q_2\), and healthy agents, in state \(q_3\). Macroscopic dynamics is an epidemiological model, i.e. a particular model of dynamics of populations \[?\], which can be also presented in the form of a dynamics of a game, from evolutionary game theory \[?, ?\]. These observations are not necessarily easy to derive when one only observes microscopic dynamics.

At the level of computation theory, to understand which microscopic dynamics corresponds to which macroscopic dynamics from biology of populations, or from evolutionary game theory, and conversely seems important.
In an orthogonal way, understanding what can be computed by such models also. Can one characterize the computational power of such continuous time models as in [?]? Let us take a new example: let us consider a system made up of agents which can have two states, and which still interact by pairs, according to a homogeneous Poisson process of fixed rate \( \lambda \), according to the following rules:

\[
\begin{align*}
\delta(+, -) &= + \\
\delta(-, +) &= + \\
\delta(+, +) &= + : 1/2, - : 1/2 \\
\delta(-, -) &= + : 1/2, - : 1/2
\end{align*}
\]

One can prove that a population of such agents will converge, whatever the initial state is, towards a state where the fraction of individuals of type + is given by \( \sqrt{2}/2 \) with very high probability, when numbers of individuals grows: the system is self-stabilizing, with the vocabulary of distributed computing [?, ?]. Let us say that such a protocol computes \( \sqrt{2}/2 \).

If all the coefficients in the rules are rational, it is easy to see that all real numbers computable by such protocols must be algebraic, since the macroscopic dynamics is given by a polynomial differential equation (here \( y' = -\lambda(\frac{1}{2} - y^2) \)).

Can one compute all algebraic numbers of \([0, 1]\)? Can one logically characterize the relation which connects the initial state to the final state in such protocols, like what was made for the population protocols of [?]? in terms of relations definable in some logic (Presburger arithmetic for population protocols)?

One also see the interest for algorithmic. Let us reconsider the previous example, where the discussion shows that the important parameter is the epidemic rate. Above a certain quantity (proportion) of adverse agents, the system collapses. Otherwise, it resists. Being able to take into account adversity in such a fine way is another strong motivation for this objective D.

Indeed current guarantees on our algorithms from distributed computing do not take into account the adversity at this level. At best, the studies are made at worst case, or on average, but without being generally able to characterize also finely, quantitatively, the tolerated maximum rate of unfavorable agents [?, ?].

I propose to contribute to this objective D, on the one hand by studying the transformation to go from a microscopic description to a macroscopic description in relations with models from distributed computing. The point is to understand the dynamics associated to a given model of microscopic rules, and to understand up to which point one can say that the trajectories of the macroscopic system represent the behavior of the system.

On the other hand, in an orthogonal way, to study the computational power of the obtained models. Can one characterize equilibria? Can we determine stability? Can one logically characterize the relations which connect the initial states to the equilibria for these systems?

Of course, one third underlying point is to understand suitable models, in relations with applications. This includes understanding which models seem reasonable for current and expected new systems (e.g. are population protocols a reasonable models of sensor networks [?]) as well as for their dynamics (e.g. are models of evolutionary game theory useful to model dynamics of distributed systems).

All these investigations are speculative, and correspond to personal work in progress, but are in fact rather close to the preceding points, since they are closely related to the study of theories of computations for particular continuous models, simply with different applications, and some distinct aspects. Notice that most population models are still polynomial differential equations models.

We think that theories of computations for continuous systems can contribute at a high degree to comprehension of distributed computing, and of current and future distributed systems.

References