Fundamenta Informaticae 1–24
IOS Press

Modelling of Complex Systems: Systems as dataflow machines

Simon Bliudze[*]
VERIMAG, Centre Équation
38610 Gières, France
Simon.Bliudze@imag.fr

Daniel Krob [C]
LIX, CNRS & École Polytechnique
91128 Palaiseau, France
dk@lix.polytechnique.fr

Abstract. We develop a unified functional formalism for modelling complex systems, that is to say systems that are composed of a number of heterogeneous components, including typically software and physical devices. Our approach relies on non-standard analysis that allows us to model continuous time in a discrete way.

Systems are defined as generalized Turing machines with temporized input, internal and output mechanisms. Behaviors of systems are represented by transfer functions. A transfer function is said to be *implementable* if it is associated with a system. This notion leads us to define a new class - which is natural in our framework - of *computable* functions on (usual) real numbers.

We show that our definitions are robust: on one hand, the class of implementable transfer functions is closed under composition; on the other hand, the class of computable functions in our meaning includes analytical functions whose coefficients are computable in the usual way, and is closed under addition, multiplication, differentiation and integration. Our class of computable functions also includes solutions of dynamical and Hamiltonian systems defined by computable functions. Hence, our notion of system appears to take suitably into account physical systems.

Keywords — Complex system; Hybrid system; Non-standard analysis; Physical system; Software system; System; System modelling; Temporized systems; Time scale; Turing machine.

[*]Part of this work was realised while this author was at LIX, École Polytechnique.
[C]Corresponding author

Contents

1. Introduction

1.1. Motivation

The notion of “system” is a typical generic — and unclear — concept which is used in many different contexts in totally different meanings. In “hard” sciences, one can find it for instance in mathematics, physics, electronics, control theory or computer science. Think for instance of dynamical, mechanical, Hamiltonian, hybrid, holonomic, embedded, concurrent or distributed systems (cf. [2, 4, 27, 31, 32, 38, 43]). In industrial applications, one will also speak of transportation, electro-mechanical, software or information systems, just to give a limited number of other examples.

Due to this both really huge and quite vague perimeter, it is therefore rather difficult to give any precise mathematical definition of a system that would cover all aspects of this concept. It indeed appears that only a few number of authors tried to propose global frameworks for discussing systems. These formalisms can moreover be roughly divided into two main categories. On one hand, one can find several “physical-oriented” formalisms (see for instance Sontag – [43] – who proposed his classical theory of systems for the purpose of control theory, but also [12, 41, 48] where other points of view of the same type are developed), even if they intend to cover both continuous and discrete aspects of systems ; on the other hand, one will find more “logical-oriented” frameworks, well adapted for describing software systems (see typically Lamport – [29] – for such an approach where systems are defined through their capacity of being described in a given specification language ; note of course that many other formalisms of the same kind can be found in the computer science literature, e.g. [10, 31, 42, 49]).

In all these different approaches, continuous and discrete systems are however quite difficult to manage together due to the fact that continuous and discrete time modelings cannot be mixed so easily. Several attempts to get round this problem were done during the two last decades leading in particular to the theory of hybrid systems that was developed jointly in control theory (see [43, 47]) and in computer science (see [2, 3, 30]). The main issue with this theory is however that the underlying formalism has some troubling properties – such as the Zeno effect which corresponds to the fact that an hybrid system can change of state an infinite number of times within a finite time – that one usually prefers to avoid in a robust modeling approach. Other interesting and slightly different attempts in the same direction can also be found in Rabinovitch and Trakhtenbrot (see [36, 44]) who tried to reconstruct a finite automata theory on the basis of a real time framework.

Everybody would therefore probably agree that there does not exist any commonly and widely agreed framework that integrates smoothly continuous and discrete systems (see [11] for a wonderful and quite exhaustive survey of all the existing research on this topic), the key problem beeing that one is usually always obliged to choose – at one moment or another – between the two underlying ways of modeling time. Our paper intends hence to bring a new contribution to this challenging problem by adressing it from another point of view. To this aim, we propose to revisit the formalisms for dealing with systems with two key ingredients – one old, Turing machines, and one new, non-standard analysis, the novelty being of re-visiting the old topic with the new one – that we will now discuss more in details.

- *Ingredient 1:* we will place us here in the line of the *algorithmic complexity theory* (in the meaning of the classical textbook [23] or more precisely in the continuation of the works described typically in [11]), systems being defined as a new kind of computing timed-machines. We shall indeed integrate in a common “machine” model the key features that can be found in quite all concrete "systems": first, the fact that systems are fundamentally characterized by a temporal behavior since they are always manipulating physical or data-*flows* (depending on their continuous or discrete structure) and, secondly, the fact that they always have an internal architecture since they are usually constructed by *integrating* (i.e. composing) other smaller sub-systems and *abstracting* the result of this composition process (this last notion being here taken in the meaning of abstract interpretation – see [13]). Note that "complex" systems can then just be seen as systems obtained through a large number of integration/abstraction processes [1] .

- *Ingredient 2:* to obtain a formal definition of a system which captures in a common framework both continuous and discrete systems, our key idea is to use a (discrete) modelling of time based on a *non-standard model of real numbers* (in the meaning of non-standard analysis – see [14, 17, 19, 20, 22, 39]). Making this quite strong step allows us indeed to take into account in the same way both physical and software systems, while keeping several natural system intuitions. Mixing this non-standard approach with the Turing-machine like ingredient which was described just above, leads us in particular to enlarge quite seriously the family of computable functions with respect to more classical approaches (see [9, 11, 33]): the analytical functions whose coefficients are Turing-computable are for instance computable in our formalism, which is usually not the case in other real-oriented computability frameworks.

With these two ingredients, we shall therefore revisit in this paper – and try to mix – both classical control theory approaches to system theory (see for example [43]) and complexity theory approaches (see for instance [11]) within a common *non-standard* point of view. This will lead us in particular to define a class of implementable transfer functions which enjoys several fundamental properties. First, it is closed under composition (cf. Theorem 3.5), which is a typical property required in the context of systems engineering. Moreover, its sub-class of computable functions is closed under addition, multiplication, differentiation and integration. These properties allow us to prove that solutions of dynamical and Hamiltonian systems defined using computable functions are themselves computable (in our meaning), which shows that our notion of system is suitable for taking into account physical systems.

Due to the fact that systems – as defined in our paper – are natural extensions of Turing machines, we obtained therefore a formal definition of systems where both software and physical systems can be modelled and described in a unified way. We believe that our formalism may be used as the basis of a complexity theory – still to be developed – for systems which appear now as a timed-extension (adding time) of Turing machines in the same way in some sense that Turing machines are themselves space-extension (adding memory) of classical automata.

[1] Note that the concept of "complex system" has presently two main meanings supported by two quite different communities. First, in the *computer science* sphere, it now refers more or less commonly to a type of system whose behavior can be obtained by a set of general rules that do not apply to any of his components (see for instance [34] for an introduction to the topic). This point of view leads in particular to difficult *behavioral* questions such as the problem of combining the behavior of different components to achieve the global behavior of a system in a given algebraic or logical framework (which is typically not addressed in our paper). Secondly, in *systems engineering* (see [26, 40, 46] for good introductory textbooks to this quite old subject which can be traced back to the early 50's), which is rather the academic and technical discipline to which we are naturally connected in this paper, complex systems refer to heterogeneous (i.e. mixing hardware, software and humanware) technical systems obtained through industrial design and integration processes. In this other direction, the key questions to solve are rather of *architectural* matter, that is to say how to design the more efficiently possible a system through an usually top-down approach (which is the systemic analysis point of view underlying our own approach). Our paper should therefore only be seen as a modest attempt to give sound mathematical bases to this last systems engineering tradition.

Let us finally add that this paper came out in part from S. Bliduze's dissertation (see [6, 8]) in the line of previous works of the two authors (cf. [7, 27]) on the same subject.

1.2. Non-Standard Analysis

To develop a global (discrete) unified framework for dealing both with continuous and discrete systems, let us first go back to the 18th century representation of real numbers (cf. the first annex of [28] for a very interesting reasoned introduction to the subject). In the 17th and 18th centuries, the common idea on real numbers was indeed quite different from the modern one. In differential calculus, reasonings were for instance typically carried out with the help of special quantities, called "infinitesimals", which had the strange property of "vanishing" when added with any usual real quantity (cf. the famous article of D'Alembert on the subject – see [16] – published in the 1754 Diderot-D'Alembert encyclopedia) [2]. For instance, one obtained in this historical framework the derivative $f'(x)$ of a given function $f(x)$ by considering the increment of the function given an *infinitesimal* increment dx of the variable x and applying directly (modulo infinitesimals) the following exact "computational" formula:

$$f'(x) = \frac{f(x+dx) - f(x)}{dx} .$$

For example, applying this mode of reasoning to $f(x) = x^2$ leads to the following computation

$$f'(x) = \frac{(x+dx)^2 - x^2}{dx} = \frac{2\,x\,dx + (dx)^2}{dx} = 2x + dx \approx 2x \, ,$$

where the last relation signifies that dx, being infinitesimal, *vanishes* in the final addition.

It took in fact around two centuries to be able to give a precise mathematical definition of the concept of infinitesimal which was only formalized by Robinson in the 60's (see [39]). He indeed introduced the set ${}^*\mathbb{R}$ of *non-standard real numbers* [3], which is a real-closed field that contains all usual real numbers, but also *infinitesimal real numbers*, i.e. non-zero elements of ${}^*\mathbb{R}$ that have their absolute value strictly less than any usual strictly positive real number $r \in \mathbb{R}_+^*$, and *infinite real numbers*, their inverses, i.e. those with an absolute value strictly greater than any usual real number $r \in \mathbb{R}$ (see Figure 1).

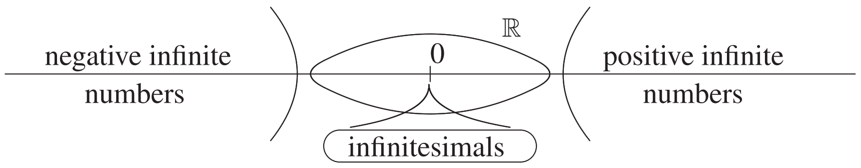

Figure 1. Graphical representation of the set ${}^*\mathbb{R}$ of non-standard real numbers.

[2] As Lakatos explains in [28], modern real analysis emerged largely in the 19th century – within Cauchy and Weierstrass works – from the difficulty of formalizing properly this not very rigorous (at this moment of history) notion.

[3] A star on the left of a symbol – as in ${}^*\mathbb{R}$ – stands always for "non-standard", whereas on the right, it just means that the single real number 0 is excluded from the considered set, as in $\mathbb{R}^*$ or ${}^*\mathbb{R}^*$ (that are denoting respectively the usual and non-standard sets of real numbers which are different from zero).

To convince the reader that there is nothing mysterious behind non-standard real numbers, we will now make a brief introduction to the topic in order to be – as much as possible – self-contained in this paper. The reader willing to learn more about non-standard analysis may of course refer to the following classical textbooks [14, 17, 19, 20, 22, 25] and papers [18, 21, 45]. Let us thus introduce the adaptation, due to Lindstrøm (cf. [14]), of the model-theoretical construction of $^*\mathbb{R}$ (see [5, 24]) which is probably the most intuitive for "standard" scientists. Using Zorn's Lemma, we should first consider an additive measure m on $\mathbb{N}$ – that we will fix throughout all this paper – with the following properties [4] :

- *Property 1:* for all $A \subset \mathbb{N}$, $m(A)$ is defined and is either 0 or 1,
- *Property 2:* $m(A) = 0$ for every finite subset A of $\mathbb{N}$,
- *Property 3:* $m(\mathbb{N}) = 1$.

The set of non-standard real numbers $^*\mathbb{R}$ is then defined as the quotient of the set $\mathbb{R}^{\mathbb{N}}$ of infinite sequences of usual real numbers by the equivalence relation $\equiv_m$ defined by setting:

$$(a_n)_{n\geq 0} \equiv_m (b_n)_{n\geq 0} \qquad \text{iff} \qquad m(\{n \in \mathbb{N}, a_n = b_n\}) = 1 \; .$$

Using the properties of an additive measure, one can now equip the set $^*\mathbb{R}$ with an addition, a product and an ordering by setting for every $a = [(a_n)_{n\geq 0}]$ and $b = [(b_n)_{n\geq 0}]$ (where [ ] denotes the class of a sequence of real numbers with respect to the equivalence relation $\equiv_n$):

- *addition:* $a + b = [(a_n + b_n)_{n\geq 0}]$,
- *product:* $a\, b = [(a_n\, b_n)_{n\geq 0}]$,
- *order:* $a < b$ iff $m(\{n \in \mathbb{N}, a_n < b_n\}) = 1$.

The reader will indeed easily establish that these operations are independent of the representatives chosen for defining them. Moreover it is also quite easy to show that these operations give a structure of totally ordered field to $^*\mathbb{R}$. Let us for instance prove that each non-zero element of $^*\mathbb{R}$ is invertible to give an example of the mode of reasoning to use. Let therefore $a = [(a_n)_{n\geq 0}]$ be an element of $^*\mathbb{R}$ different from its obvious zero element $0 = [(0)_{n\geq 0}]$. Being different from 0 means by definition that the measure $m(A)$ of the set $A = \{n \in \mathbb{N}, a_n = 0\}$ is equal to 0 (which implies that $m(\mathbb{N} - A) = 1$ according to the properties satisfied by m). Let us now consider the sequence $(b_n)_{n\geq 0}$ of $\mathbb{R}^{\mathbb{N}}$ defined by setting:

$$\begin{cases} b_n = 1/a_n & \text{for every } n \notin A, \\ b_n = 0 & \text{for every } n \in A. \end{cases}$$

It is therefore easy to see that one has $\{n \in \mathbb{N}, a_n\, b_n = 1\} = \mathbb{N} - A$, which immediately implies that the sequence $(a_n\, b_n)_{n\geq 0}$ is equivalent to the constant sequence $(1)_{n\geq 0}$ due to the fact that $m(\mathbb{N} - A) = 1$, which is obviously the neutral element 1 of the product of $^*\mathbb{R}$, i.e. that one has $a\, b = 1$.

The totally ordered field $^*\mathbb{R}$ contains of course the usual real numbers of $\mathbb{R}$ that can be identified to the classes of the constant sequences. However it also contains new interesting types of elements:

[4] In other words, m is a measure that separates between each subset of $\mathbb{N}$ and its complement, one and only one of these two sets being always of measure 1, when the other is then of measure 0.

- *infinitesimal* numbers, i.e. elements $x \in {}^*\mathbb{R}$ such that $-a < x < a$ for every real number $a \in \mathbb{R}$,
- *infinite* numbers, i.e. elements $x \in {}^*\mathbb{R}$ such that $x > a$ or $x < -a$ for every real number $a \in \mathbb{R}$ [5].

It is indeed easy to see that the equivalence class of any sequence of real numbers converging to 0 (resp. to $\pm\infty$) is infinitesimal (resp. infinite). To give more insight to the reader on the way to work with the above formalism, we shall now prove the following key property of non-standard real numbers.

Proposition 1.1. Any *finite* non-standard real number $x \in {}^*\mathbb{R}$ can be uniquely decomposed as a sum $x = a + \epsilon$ where a is a usual real number of $\mathbb{R}$ and ϵ is infinitesimal.

Proof: The unicity of the claimed decomposition is obvious. If one would have $a+\epsilon = b+\eta$ where a and b are two usual real numbers and ϵ and η two infinitesimals, one would indeed deduce that $a - b = \eta - \epsilon$. But, since this quantity is both a usual real number and an infinitesimal, it must be equal to 0, which shows that one both has $a = b$ and $\epsilon = \eta$ as expected.

On the other hand, let $x = [(x_n)_{n\geq 0}]$ be a finite non-standard real number. Let us then consider the subset B of $\mathbb{R}$ defined by setting $B = \{b \in \mathbb{R}, b < x\}$. Due to the finiteness of x, it is easy to see that B is a bounded set of real numbers which hence has a supremum a within $\mathbb{R}$. Let us then set $\epsilon = x - a$ and let us suppose that there exists a strictly positive real number $r \in \mathbb{R}$ such that $r < |\epsilon|$. If $\epsilon > 0$, we would have $r < x - a$, i.e. $a < a + r < x$, in contradiction with the fact that a is the supremum of B (there would be a bigger real number $a + r$ in B than a). On the other hand, if $\epsilon < 0$, we would have $r < a - x$, hence $x < a - r < a$, again contradicting the fact that a is the supremum of B ($a - r$ would be a small upper bound of B than a). Hence ϵ must be an infinitesimal, which ends our proof. $\square$

Let us now introduce some notations that will be useful in the sequel. Two non-standard real numbers x and y will first said to be *infinitely close*, which will be denoted by $x \approx y$, if and only if $x - y$ is infinitesimal. The previous result can therefore be restated as the fact that each finite non-standard real number is always infinitely close to a unique usual real number (which also implies that infinitesimal numbers are exactly the non-standard real numbers which are infinitely close to 0). Observe also that, among all non-standard real numbers, one can of course consider the set ${}^*\mathbb{Z}$ of *non-standard integers* that, on top of usual integers, contains infinite ones, having absolute value greater than any $n \in \mathbb{N}$.

We will finally end this introduction by giving (without any proof) some important model-theoretical properties of the set of non-standard real numbers (see [5, 19, 20, 22, 24] for more details). The first important property to know is the fact that the two totally ordered fields ${}^*\mathbb{R}$ and $\mathbb{R}$ are *elementarily equivalent*, which means that the first order logical properties of $\mathbb{R}$ and ${}^*\mathbb{R}$ (expressed in the logical theory of ordered fields) are exactly the same. This property is in fact a special case of the following more general result, called the *transfer principle* (see [14] for a proof and a totally rigorous statement).

Transfer principle. Let F be a set-theoretical formula involving only usual real variables which does not have any free variable. Then F is true iff ${}^*F$ is true, where ${}^*F$ stands for the formula on non-standard real numbers obtained from F by transfer, i.e. by changing each usual standard set a involved in F by its non-standard version ${}^*a$ (that is to say the $\equiv_m$-classes of the sequences of elements of a).

[5] An element of ${}^*\mathbb{R}$ which is not infinite is called *finite*.

As one can immediately see, the transfer principle is a very strong property of non-standard real numbers since it claims basically that every usual property of standard real numbers holds for non-standard real numbers up to replacing standard sets by their non-standard equivalents. It implies for instance that the recurrence principle can be immediately lifted to $^*\mathbb{R}$, up to working with $^*\mathbb{N}$ instead of $\mathbb{N}$. In the same way, the transfer principle shows that most of the classical properties (such as the existence of a non-standard integer part of any non-standard real number, of an upper bound for each bounded internal [6] non-standard subset of $^*\mathbb{R}$, etc.) of $\mathbb{R}$ still work for $^*\mathbb{R}$ up to replacing usual standard involved sets by their non-standard equivalents.

2. Time Scales, Data Flows and Transfer Functions

We are now in position to introduce the bases of the framework that we will use to deal with systems in the sequel. Note that most of our formalism is fundamentally a non-standard revisiting of quite classical concepts that were for instance synthesized in Sontag's classical textbook (cf. [43]) [7] . The new system properties brought by our non-standard approach will in fact mainly be seen in the last part of our paper.

2.1. Time Scales

The first key modification brought by non-standard analysis is of course related to the modelling of time.

Definition 2.1. Time references – A non-empty subset $\mathcal{T} \subset {}^*\mathbb{R}$ is called a *time reference* if and only if, for any $t \in {}^*\mathbb{R}$, the two sets $\mathcal{T}_{t-} = \{t' \in \mathcal{T} \mid t' < t\}$ and $\mathcal{T}_{t+} = \{t' \in \mathcal{T} \mid t' > t\}$ have respectively always a maximum and a minimum.

The conditions imposed on time references in Definition 2.1, ensures that the successor (resp. predecessor) of any $t \in \mathcal{T}$ – denoted by $succ_{\mathcal{T}}(t)$ (resp. by $pred_{\mathcal{T}}(t)$) – can be uniquely defined. Observe also that time references are clearly closed under union (but not under intersection).

Definition 2.2. Time scales – A non-empty subset $\mathbb{T} \subset {}^*\mathbb{R}^+$ is said to be a *time scale* if and only if it is the positive part of a time reference, i.e. if and only if there exists a time reference $\mathcal{T}$ such that one has $\mathbb{T} = \mathcal{T} \cap {}^*\mathbb{R}^+$. We shall use the term *moment* to denote the elements of a time scale.

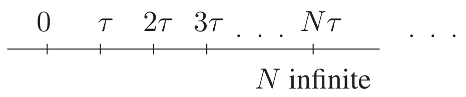

Figure 2. Graphical representation of a (non-standard) regular time scale.

One important class of time scales consists of *regular time scales*, i.e. of the type $\mathbb{T}_\tau = {}^*\mathbb{N}\,\tau$ where $\tau \in {}^*\mathbb{R}^*_+$ is a strictly positive non-standard real number, called the step of its corresponding regular time

[6] A subset $^*A$ of $^*\mathbb{R}$ is said to be *internal* if and only if there exist a sequence $(A_n)_{n\geq 0}$ of usual subsets of $\mathbb{R}$ such that $x \in {}^*A$ iff $x = [(a_n)_{n\geq 0}]$ with $a_n \in A_n$ for every $n \geq 0$.

[7] Many authors – such as typically Rabinovitch ([36]) who also re-visited the same concepts when introducing real-time aspects in finite automata theory – are working exactly in the same way.

scale. Such a time scale can be seen as a discrete series of clock ticks occurring at each time $n \cdot \tau$, where $n \in {}^*\mathbb{N}$ describes all non-standard finite and infinite integers (see Figure 2).

The next definition introduces now the notion of continuous time scales that will allow us to mimic naturally – and discretely – continuous time in a non-standard framework as we will see below.

Definition 2.3. Continuous time scales – A time scale $\mathbb{T}$ is said to be *continuous* if for any positive standard real number $r \in \mathbb{R}^+$, there exists a moment $t \in \mathbb{T}$ such that $t \approx r$.

We are now in position to show that we can recover usual continuous time by considering regular time scales with infinitesimal steps. This result is key since it means that we can model *discretely* continuous time in a non-standard framework, which is one of the core ideas on which this paper relies.

Proposition 2.1. A regular time scale with an infinitesimal step is continuous.

Proof: Let $\mathbb{T}$ be a regular time scale with an infinitesimal step $\tau \approx 0$ and let $r \in \mathbb{R}^+$ be any positive standard real. The non-standard version of Archimedes' property (use either the formalism of Section 1.2, either the transfer principle or look to Chapter A.6 of [4]) implies that there exists always a non-standard positive integer $N \in {}^*\mathbb{N}$ (here necessarily infinite) such that $N < r/\tau \leq N + 1$. We deduce therefore that $N\tau < r \leq (N + 1)\tau$ and $0 < r - N\tau \leq \tau \approx 0$. Hence $t \approx r$ with $t = N\tau \in \mathbb{T}$ as requested. □

We will also use in the sequel the notion of time scale refinement that we finally quickly recall.

Definition 2.4. A time scale $\mathbb{T}$ *refines* another time scale $\mathbb{T}'$ – denoted by $\mathbb{T} \preceq \mathbb{T}'$ – iff $\mathbb{T}' \subseteq \mathbb{T}$.

2.2. Dataflows

This new sub-section introduces the notion of dataflow, taken here again in a non-standard meaning, which will be another cornerstone of our system modelling approach due to the fact that we will consider here systems as dataflow transformers (and not as only data transformers as it is classically done).

Definition 2.5. Dataflow – Let $\mathbb{T}$ be a time scale. A $\mathbb{T}$-*dataflow* over an alphabet A is then just a mapping $x : \mathbb{T} \to A$. The set of all $\mathbb{T}$-dataflows over A will be denoted by $A^{\mathbb{T}}$.

The next definition intends to capture the fact that a dataflow can be observed at any positive time although its value only changes at discrete moments specified by its underlying time scale.

Definition 2.6. Snapshots – Let x be a $\mathbb{T}$-dataflow over A and let $t \in {}^*\mathbb{R}^+$ be a positive non-standard real number. The *snapshot of x at time t* is then the element of A, denoted by $x :: t$, which is defined by:

$$x :: t = x(t'), \text{ where } t' = \min\{u \in \mathbb{T} \mid t \leq u\}\,.$$

We can now introduce the following notion of product of dataflows that will be quite useful when we will consider multi-entry systems in the next section. The reader will easily check that the product of dataflows defined in this manner is an associative product.

Definition 2.7. Let $x \in A_1^{\mathbb{T}_1}$ and $y \in A_2^{\mathbb{T}_2}$ be two dataflows and let $\mathbb{T}$ be the time scale $\mathbb{T} = \mathbb{T}_1 \cup \mathbb{T}_2$ resulting of the union of the two time scales of these two dataflows. The *product* of x and y, denoted by $x \otimes y$, is then the $\mathbb{T}$-dataflow over $A_1 \times A_2$ defined, for any $t \in \mathbb{T}^+$, by setting:

$$(x \otimes y)(t) \overset{def}{=} (x :: t,\, y :: t)\,.$$

Let us finally introduce the following notion that we will use in the next section for discussing equivalence results between different systems definitions.

Definition 2.8. Observational equivalence – Two dataflows n-uples $x = (x_i)_{i=1\ldots n}$ and $y = (y_j)_{j=1\ldots n}$ are said to be *observationally equivalent* – which will be denoted by $x \simeq y$ – iff one has:

$$(x_1 \otimes \ldots \otimes x_n) :: t = (y_1 \otimes \ldots \otimes y_n) :: t \quad \text{for all } t \in {}^*\mathbb{R}^+\,.$$

Example 2.1. Let $\mathbb{T}_\tau$ be a regular time scale with step τ and let f be a $\mathbb{T}_\tau$-dataflow over an alphabet A. Consider the regular time scale $\mathbb{T}_{\tau/2}$ of step $\tau/2$. It is then clear that the $\mathbb{T}_{\tau/2}$-dataflow f' which repeats every value in f twice, is observationally equivalent to f.

2.3. Transfer Functions

Observable behavior of a system is described by its associated transfer function which can be seen as a real-time dataflow transformer satisfying the causality condition as given in the next definition. Note however that causality as defined below corresponds only to strong causality in a classical framework (see [43]). In the nonstandard analysis framework we took here, the usual concepts of causality and strong causality [8] are indeed collapsing now into the unique concept of (strong) causality. This is due to the fact that strong and non strong causality differ in a classical framework since systems can work there in null time, i.e. spend no time to pass from their input to their output. In a nonstandard framework, no system can work in zero time: there will be always at least an infinitesimal moment (which may model the zero time behaviors of usual systems) between an input and the corresponding output of a system, which as a consequence leads us to define only the following (strong) causality property as the unique type of causality that we will consider in our framework.

Definition 2.9. Transfer functions – Let In and Out be two alphabets and $\mathbb{T}^i$ and $\mathbb{T}^o$ be two time scales. Then a function $\mathcal{F} : \mathrm{In}^{\mathbb{T}^i} \to \mathrm{Out}^{\mathbb{T}^o}$ is said to be a *transfer function* iff it is causal, i.e. iff one has:

$$\forall\, t \in {}^*\mathbb{R}^+,\ \forall\, x, y \in \mathrm{In}^{\mathbb{T}^i},\ \Big(\forall\, 0 \le t' < t,\ x :: t' = y :: t'\Big) \implies \Big(\mathcal{F}(x) :: t = \mathcal{F}(y) :: t\Big)\,.$$

Observe that if $\mathcal{F}_1$ and $\mathcal{F}_2$ are two transfer functions such that the set of output dataflows of $\mathcal{F}_1$ coincides with the set of input dataflows of $\mathcal{F}_2$, then the composition $\mathcal{F}_1 \circ \mathcal{F}_2$ is also a transfer function. More generally, transfer functions are "composed" through interfaces as defined in the next definition. Note that we will in fact speak in full generality of *transfer function integration* – rather than composition

[8] Recall that a usual transfer function $x(t) \longrightarrow F(x(t); t)$ is causal (resp. strongly causal) if and only if for every $t \in \mathbb{R}$, one has $\forall\, t' \le t$, (resp. $t' < t$), $x(t') = y(t') \implies F(x(t); t) = F(y(t); t)$. A strongly causal transfer function is of course always causal, but the converse does not hold.

– due to the fact that our “composition” is a mixture between parallel and sequential compositions which models typically the possibility of redirecting dataflows from an output to an input of the two involved transfer functions. Integration of transfer functions is in fact only the transfer function counterpart of the system integration operator that plays a key role in systems modelling (cf. Section 3).

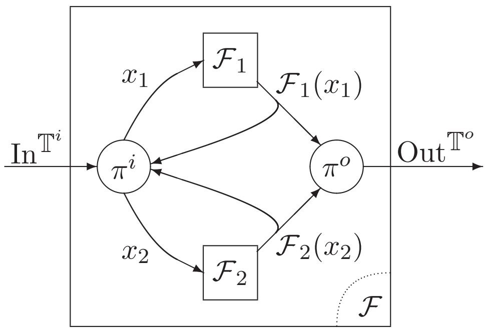

Figure 3. Two transfer functions $\mathcal{F}_1$ and $\mathcal{F}_2$ integration through an interface $(\mathrm{In}^{\mathbb{T}^i}, \mathrm{Out}^{\mathbb{T}^o}, \pi^i, \pi^o)$.

Definition 2.10. Interfaces and integration of transfer functions – Let $\mathrm{In}_{1,2}$ and $\mathrm{Out}_{1,2}$ be four alphabets which are all products of other alphabets [9] with two distinguished elements $\flat_{1,2}$ in $\mathrm{Out}_{1,2}$, let $\mathbb{T}^i_{1,2}$ and $\mathbb{T}^o_{1,2}$ be four time scales and let finally $\mathcal{F}_1 : \mathrm{In}_1^{\mathbb{T}^i_1} \to \mathrm{Out}_1^{\mathbb{T}^o_1}$ and $\mathcal{F}_2 : \mathrm{In}_2^{\mathbb{T}^i_2} \to \mathrm{Out}_2^{\mathbb{T}^o_2}$ be two transfer functions. An *interface* for $\mathcal{F}_1$ and $\mathcal{F}_2$ is then a quadruple $\Pi = (\mathrm{In}^{\mathbb{T}^i}, \mathrm{Out}^{\mathbb{T}^o}, \pi^i, \pi^o)$ where:

- In and Out are two alphabets which are products of other alphabets in such a way that the alphabets In1 and In2 (resp. the alphabet Out) are products (resp. is a product) of some of the alphabets involved in the alphabet products defining In, Out_1 and Out_2 (resp. Out_1 and Out_2),
- $\mathbb{T}^i$ and $\mathbb{T}^o$ are any arbitrary time scales,
- $\pi^i = (\pi^i_1, \pi^i_2)$ is a pair of componentwise projections [10] between the following product sets:

$$\pi^i_1 : \mathrm{In} \times \mathrm{Out}_1 \times \mathrm{Out}_2 \to \mathrm{In}_1 ,$$
$$\pi^i_2 : \mathrm{In} \times \mathrm{Out}_1 \times \mathrm{Out}_2 \to \mathrm{In}_2 ,$$

- $\pi^o : \mathrm{Out}_1 \times \mathrm{Out}_2 \to \mathrm{Out}$ is a componentwise projection between the involved product sets.

The *integration* of $\mathcal{F}_1$ and $\mathcal{F}_2$ through Π is then the transfer function $\mathcal{F} : \mathrm{In}^{\mathbb{T}^i} \to \mathrm{Out}^{\mathbb{T}^o}$, denoted by $I_\Pi(\mathcal{F}_1, \mathcal{F}_2)$, which is defined by setting for every $x(t) \in \mathrm{In}^{\mathbb{T}^i}$ (see Figure 3):

$$\mathcal{F}(x)(t) \overset{def}{=} \pi^o(\mathcal{F}_1(x_1) :: t, \mathcal{F}_2(x_2) :: t) \text{ for every } t \in \mathbb{T}^o, \tag{1}$$

[9] In other words, all these alphabets are therefore equal to some product $A_1 \times \ldots A_n$ of other alphabets $(A_i)_{i=1\ldots n}$.
[10] A componentwise projection is any surjective mapping π from a product set $A_1 \times \cdots \times A_n$ onto one of its “partial-product” sets $A_{i_1} \times \cdots \times A_{i_k}$ such that one has $\pi(a_1, \ldots, a_n) = (a_{i_1}, \ldots, a_{i_k})$ for every n-uple $(a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$.

where x_1 and x_2 are respectively the $\mathbb{T}_1^i$ and $\mathbb{T}_2^i$-dataflows inductively [11] defined by setting:

- *Initial conditions:*

$$\left\{\begin{array}{rcl} x_1(\min(\mathbb{T}_1^i)) & = & \pi_1^i(x::\min(\mathbb{T}_1^i), \flat_1, \flat_2), \\ x_2(\min(\mathbb{T}_2^i)) & = & \pi_2^i(x::\min(\mathbb{T}_2^i), \flat_1, \flat_2), \end{array}\right. \tag{2}$$

- *Induction relations:*

$$\left\{\begin{array}{rcll} x_1(t) & = & \pi_1^i(x::t, \mathcal{F}_1(x_1)::t, \mathcal{F}_2(x_2)::t) & \text{for every } t \in \mathbb{T}_1^i - \{\min(\mathbb{T}_1^i)\}\,, \\ x_2(t) & = & \pi_2^i(x::t, \mathcal{F}_1(x_1)::t, \mathcal{F}_2(x_2)::t) & \text{for every } t \in \mathbb{T}_2^i - \{\min(\mathbb{T}_2^i)\}\,. \end{array}\right. \tag{3}$$

The next result shows that Definition 2.10 is consistent which needs both to ensure the consistency of the above definition and the fact that it defines really a transfer function.

Proposition 2.2. For any transfer functions $\mathcal{F}_1$, $\mathcal{F}_2$ and any interface Π as in Definition 2.10, the integration $I_\Pi(\mathcal{F}_1, \mathcal{F}_2)$ of $\mathcal{F}_1$ and $\mathcal{F}_2$ is always a well-defined transfer function.

Proof: We will take here all notations of Definition 2.10. The reader will then first easily check that Equations (2) and (3) define well inductively the two dataflows x_1 and x_2 due to the fact that $\mathcal{F}_1$ and $\mathcal{F}_2$ are transfer functions (write x_1 and x_2 as sequences of unknown variables over In_1 and In_2 and look at the equations involving these variables given by Equations (2) and (3)). The only thing to prove now is then that the integration of two transfer functions is itself a transfer function. Going back to the defining Equation (1) and using the fact that $\mathcal{F}_1$ and $\mathcal{F}_2$ are causal functions, this last property results then immediately of the fact that for every $t \in \mathbb{R}^+$, one has $x_1(t) = x_1'(t)$ and $x_2(t) = x_2'(t)$ as soon as $x(t') = x'(t')$ for every $t' < t$ (with obvious notations), as one may easily check. □

As the reader may easily check, the notions of interfaces and integration can be extended without difficulties to any finite number of transfer functions. This generalization will in particular be used in the next example that we will now explore in order to illustrate all previous concepts.

Example 2.2. Water Tank – We shall revisit in our nonstandard framework a well known example of the hybrid systems and control theory literature. Let us hence fix first some regular continuous time scale $\mathbb{T}$ with infinitesimal time step ε that will be used to define all the transfer functions of our example.

We shall therefore consider the system consisting of a water tank where water arrives at a variable rate $w_i(t) \geq 0$ (with $t \in \mathbb{T}$) through one single pipe. The water leaves through another (output) pipe at rate $w_o(t)$ (with $t \in \mathbb{T}$) controlled by a valve (see Figure 4-a) whose position will be given by $v(t) \in [0, 1]$ (with $t \in \mathbb{T}$), 0 and 1 modelling respectively here the fact that the valve is closed or open. The initial position at time $t = 0$ of the valve is equal to some constant $v(0) = V_0$. The maximal throughput capacity of the output pipe is C, thus its actual throughput at each moment $t \in \mathbb{T}$ is $C\,v(t)$. The valve is controlled by a sensor measuring the level $l(t)$ of water in the tank, which aims at keeping this level in a given interval $[L_1, L_2]$ (the initial water tank level L_0 belongs therefore to this interval). For simplicity, we assume that there is always enough water in the tank to saturate the output pipe and that the incoming flow does not exceed the output pipe's capacity, i.e. that one has always $\max(w_i(t),\ t \in \mathbb{T}) \leq C$.

[11] Inductively is taken here in the sense of the nonstandard induction property satisfied by ${}^*\mathbb{N}$ according to the transfer principle.

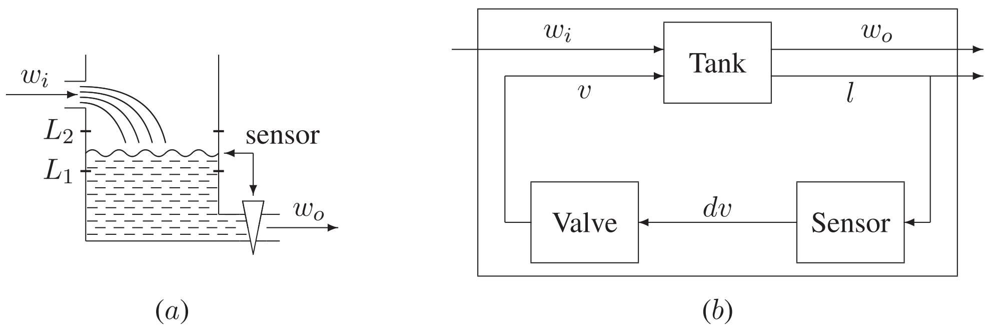

Figure 4. The Water Tank (a) and an associated simplified systemic model (b).

The transfer function of the complete water tank system has the input space $\mathrm{In} = [0, C]$ (incoming flow rate) and the output space $\mathrm{Out} = [0, C] \times [L_1, L_2]$ (output flow rate and current water level in the tank). It can be modelled as an integration of the three following transfer functions (see Figure 4-b):

1. The *tank* transfer function, T, taking on input the current values of the incoming water flow $w_i(t)$ and the position v(t) of the valve and sending on its output the corresponding output water flow $w_o(t)$ and water level $l(t)$ according to the following equations:

$$w_o(0) = C\,V_0,\;\; w_o(t+\epsilon) = C\,v(t) \;\text{ for every } t \in \mathbb{T}^*,$$
$$l(0) = L_0,\;\; l(t+\epsilon) = l(t) + (w_i(t) - w_o(t))\,\epsilon \;\text{ for every } t \in \mathbb{T}^*\;.$$

   The input and output spaces of T are then $\mathrm{In}_T = [0, C] \times [0, 1]$ and $\mathrm{Out}_T = [0, C] \times [L_1, L_2]$.

2. The *sensor* transfer function, S, taking on input the water level $l(t)$ and sending on its output a valve position adjustment $dv(t)$ defined from some given equation as follows [12] :

$$dv(0) = 0,\;\; dv(t+\epsilon) \;=\; \mathrm{sign}\,\left(l(t) - \frac{L_1 + L_2}{2}\right)\,\epsilon \;\text{ for every } t \in \mathbb{T}^*\;.$$

   The input and output spaces of S are then $\mathrm{In}_S = [L_1, L_2]$ and $\mathrm{Out}_S = \{-\epsilon, 0, \epsilon\}$.

3. The *valve* transfer function, V, taking on input the adjustment $dv(t)$ and providing on its output $v(0) = V_0$ at time $t = 0$ and the value $v(t)$ given by the following formula:

$$v(t+\epsilon) = \begin{cases} v(t) + dv(t) & \text{if } v(t) + dv(t) \in [0,1]\,, \\ v(t) & \text{if } (v(t) = 0 \wedge dv(t) = -\epsilon) \text{ or } (v(t) = 1 \wedge dv(t) = \epsilon)\;, \end{cases}$$

   for $t \in \mathbb{T}^*$. The input and output spaces of V are then $\mathrm{In}_V = \{-\epsilon, 0, \epsilon\}$ and $\mathrm{Out}_V = [0, 1]$.

[12] Where sign denotes the function from $\mathbb{R}$ into $\{-1, 0, 1\}$ defined by setting $\mathrm{sign}\,(0) = 0$, $\mathrm{sign}\,(x) = 1$ for every $x > 0$ and $\mathrm{sign}\,(x) = -1$ for every $x < 0$.

We will not enter at this point in the consistency of our model, i.e. on the fact that all the limits of the input and output spaces of our transfer functions are well defined [13] since we only wanted to illustrate the transfer function integration mechanism. As one can see on Figure 4-b, the transfer function of the full tank water system is indeed given by the integration of the transfer functions T, S and V through the interface $\Pi = (\mathrm{In}^{\mathbb{T}}, \mathrm{Out}^{\mathbb{T}}, \pi^i, \pi^o)$ with componentwise projections

$$\begin{array}{rcl} \pi^i : \mathrm{In} \times \mathrm{Out}_T \times \mathrm{Out}_S \times \mathrm{Out}_V & \longrightarrow & \mathrm{In}_T \times \mathrm{In}_S \times \mathrm{In}_V \\ (w_i; w_o, l; dv; v) & \longmapsto & (w_i, v; l; dv) \end{array}$$

and

$$\begin{array}{rcl} \pi^o : \mathrm{Out}_T \times \mathrm{Out}_S \times \mathrm{Out}_V & \longrightarrow & \mathrm{Out} \\ (w_o, l; dv; v) & \longmapsto & (w_o, l)\,. \end{array}$$

3. Systems

We shall now revisit classical system theory in the framework of non standard analysis. This will lead us to redefine the notion of system and the associated transfer function and integration operator concepts.

Notation 3.1. For any alphabet A, we will denote in the sequel by $\overline{A}$ the completion of A by a blank symbol $\flat$, that is to say $\overline{A} \overset{def}{=} A \cup \{\flat\}$.

3.1. Definition

We are now in position to introduce our "mechanical" system model in the line of the usual Turing machine mechanism which is essentially enhanced by adding to it input/output mechanisms and temporisation (though input, output and internal time scales) within a non-standard analysis approach which allows to work with continuous values [14] .

Definition 3.1. Systems – A *system* S is defined as the union of the following elements (see Figure 5):

- an *input/output mechanism* that consists respectively of:
  - an *input channel* x capable of receiving — only at moments that belong to a given time scale $\mathbb{T}^i$ called the *input time scale* — elements that belong to a given alphabet In, called the *input domain* of S (which will also be denoted by $\mathrm{In}(S)$),
  - an *output channel* y capable of emitting — only at moments that belong to a given time scale $\mathbb{T}^o$ called the *output time scale* — elements that belong to a given alphabet Out, called the *output domain* of S (which will also be denoted by $\mathrm{Out}(S)$),

[13] The only thing to check is in fact that one always has $l(t) \in [L_1, L_2]$ which can be proved by supposing that the property is not respected and considering the first moment where one of the L_1 or L_2 limit is broken in order to get a contradiction.
[14] Even if our system mechanism can be seen as an extension of the usual Turing machine mechanism, it should not considered as a full generalization. In the classical framework, S. Akl (see for instance [1]) showed indeed that standard physical real time destroys universality, a key classical property of Turing machines. This last result shows therefore that it is not totally obvious to see whether our own system formalism preserves universality, which is an important problem that we will hence let open (the most probable result in this way being however probably negative).

each of these channels is represented by a single memory cell containing the last received value and overwritten at each moment on the corresponding time scale;

- an *internal memory* composed of
  - a *memory tape* indexed by the set $^*\mathbb{N}$ of non-standard positive integers containing symbols from a given alphabet M (also denoted by $M(S)$) called the *memory domain* of S, with a size being bounded by a (possibly infinite) non-standard integer $N \in {}^*\mathbb{N}$,
  - a read/write head *current position indicator* $p \in {}^*\mathbb{N}$ (with $p < N$);
- a *control mechanism* defined by
  - an *internal time scale* $\mathbb{T}^s$,
  - an *internal state set* which is just an arbitrary standard finite set Q (also denoted $State(S)$),
  - a *transition function* $\delta : Q \times M \times \mathrm{In} \rightarrow Q \times M \times \{-1, 0, +1\} \times \overline{\mathrm{Out}}$, that allows to update at each moment on the internal time scale the system's current state, the element of the internal memory at the current head position, the head position itself and the system's output (the output value computed by δ can be blank when $t \in \mathbb{T}^s \setminus \mathbb{T}^o$, i.e. at the moments on the internal time scale that do not belong to the output time scale);
- a set of *initial values* to which the concerned above components are assigned at their respectively associated initial moments.

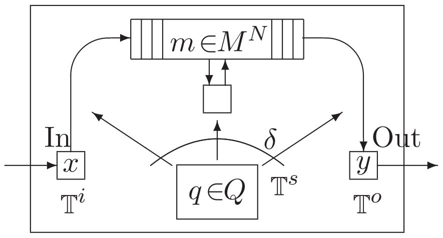

Figure 5. Graphical representation of a system.

Let us now consider the dynamics of a system in this model. Observe first that the internal configuration of a system is described at each moment of time by the system's state, the contents of its memory tape, and the position of its read/write head. The following definition introduces this idea formally.

Definition 3.2. Instantaneous description of a system – Let S be a system. An *instantaneous description* of S at a given moment $t \in \mathbb{T}^s$ on its internal time scale is a quadruple $d = (q, m, p, \tilde{y})$ of $Q \times M^N \times {}^*\mathbb{N} \times \overline{\mathrm{Out}}$, where q, m, p, and $\tilde{y}$ represent respectively the internal state, the full contents of the memory tape, the position of the read/write head and the symbol to be emitted on the output of S at the moment t (equal to blank if it does not exist).

We can now introduce the dynamics of a system (in the meaning of Definition 3.1) defined as a coherent sequence of instantaneous descriptions indexed by moments on its internal time scale.

Definition 3.3. Execution of a system – Let S be a system. An *execution* of S for a given input flow x is then the $\mathbb{T}^s$-dataflow $d = (q(t), m(t), p(t), \tilde{y}(t))_{t\in\mathbb{T}^s}$ over $Q \times M^N \times {}^*\mathbb{N} \times \overline{\text{Out}}$ which is defined inductively (in the non-standard meaning) as follows:

- the initial value of d at time $t_0 = \min(\mathbb{T}^s)$, i.e. the 4-uple $(q(t_0), m(t_0), p(t_0), \tilde{y}(t_0))$, is equal to the initial internal state, memory content, read/write head position and output value of S,
- let now $t \in \mathbb{T}^s$ and suppose that the 4-uple $d(t) = (q(t), m(t), p(t), \tilde{y}(t))$ is defined; we can then define $d(t')$ for $t' = succ_{\mathbb{T}^s}(t)$ by setting:

$$d(t') = (q(t'), m(t'), p(t'), \tilde{y}(t')) \overset{def}{=} (q', m'', p(t) + \Delta, \tilde{y}'), \tag{4}$$

where one has first $(q', m', \Delta, \tilde{y}') = \delta(q(t), m(t)_{p(t)}, x :: t)$, with $m(t)_{p(t)}$ and $x :: t$ respectively being the values of the memory tape cell pointed by the read/write head and the snapshot of the input flow x at moment t, and secondly $m"_j$ equal to $m(t)_j$ for $j \neq p(t)$ and to m'_j for $j = p(t)$.

Taking all the notations introduced in the previous definition 3.3, the output flow $y \in \text{Out}^{\mathbb{T}^o}$ of a system S which is generated by a given input flow $x \in \text{In}^{\mathbb{T}^i}$ can now be defined by setting:

$$\forall t \in \mathbb{T}^o, \quad y(t) \overset{def}{=} \tilde{y} :: t\,.$$

Note 3.2. As at some moments on the internal time scale — those that do not belong to $\mathbb{T}^i$ — no new data is available on the input channel, one applies then the transition function δ to the snapshot of the input flow, i.e. the latest received value.

In [8], we showed that our notion of system can be naturally generalized to include systems with multiple input/output channels and memory tapes which are called *multi-systems*. We will need to work with such multi-systems in the next example that shows how a classical physical system can be modelled by a system in our meaning. Note however that this generalization does not increase the expressive power of our model as stated by Theorem 3.4 in the next sub-section.

Example 3.1. Simple pendulum – Consider a pendulum consisting of a point mass m hanging on a rigid string of negligible mass and of length L (see Figure 6-a) [15] . The motion of such a pendulum is described by the usual standard differential equation

$$\ddot{\theta} = -\frac{g}{L}\sin\theta\,, \tag{5}$$

where θ is the angle formed by the string and the y axis. By integrating (5) and multiplying by mL^2, we obtain then immediately the following energy preservation equation:

$$\frac{1}{2}\,m(L\dot{\theta})^2 + mgL(1-\cos\theta) = C\,, \tag{6}$$

where the first and the second summand in the left hand side represent respectively the kinetic, $\mathbb{E}_K$, and the potential, $\mathbb{E}_P$, energies of the system. The following relation can then easily be obtained from (6):

$$\left|\frac{d\,\mathbb{E}_K}{dt}\right| = \left|\frac{d\,\mathbb{E}_P}{dt}\right| = \sqrt{\frac{2\,\mathbb{E}_K\mathbb{E}_P}{L}\left(2g - \frac{\mathbb{E}_P}{mL}\right)}\,. \tag{7}$$

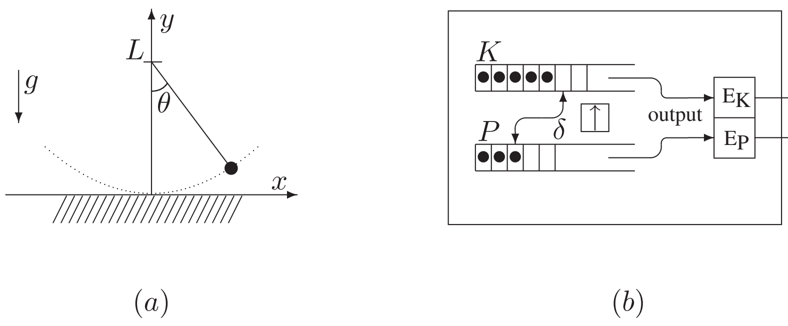

Figure 6. Simple pendulum: mechanical (a) and systemic (b) representations.

A system modelling such a pendulum is shown in Figure 6-b. It has two tapes K and P, each of them containing a (non-standard) finite number of infinitesimal quanta of energy. The two tapes together always contain the same infinitely great number $N \in {}^*\mathbb{N}$ of energy quanta (in such a way that the total energy contained in the two tapes is a constant equal to C). The system's behaviour consists then essentially in taking at each moment of time – on a given continuous time scale $\mathbb{T}$ – the number of quanta defined by (7) (which can simulated – using a finer time scale than $\mathbb{T}$ – within our system model due to the results of Section 4.1) from one tape and putting it on the other one, until the working tape is empty, at which moment the two tapes exchange their roles.

3.2. Equivalence of Systems

According to Definition 3.3, every system S with input flows in $\text{In}^{\mathbb{T}^i}$ and output flows in $\text{Out}^{\mathbb{T}^o}$ can be equipped with its *transfer function*

$$\mathcal{F}(S) : \text{In}^{\mathbb{T}^i} \to \text{Out}^{\mathbb{T}^o},$$

which is the mapping that associates to any input flow $x \in \text{In}^{\mathbb{T}^i}$ the output flow $y \in \text{Out}^{\mathbb{T}^o}$ generated by S. Conversely we shall call any transfer function that can be associated to a system *implementable*.

Definition 3.4. Implementable transfer functions – We say that a transfer function $\mathcal{F}$ is *implementable* if there exists a system S such that $\mathcal{F} = \mathcal{F}(S)$.

The notion of the transfer function associated with a system allows us to introduce the following key equivalence relation for systems in our meaning.

Definition 3.5. Weak system equivalence – Two systems S_1 and S_2 are said to be *weakly equivalent* if and only if the following characterizing properties hold simultaneously:

- In$(S_1) =$ In(S_2) and Out$(S_1) =$ Out(S_2),
- for any two input flows x_1 and x_2 over In(S_1), one has:

$$(x_1 \simeq x_2) \implies (\mathcal{F}_{S_1}(x_1) \simeq \mathcal{F}_{S_2}(x_2))$$

[15] Other interesting examples of non-standard revisits of classical physical problems can be found in [18, 19, 21, 45].

where $\mathcal{F}_{S_1}(x_1)$ and $\mathcal{F}_{S_2}(x_2)$ stand for the corresponding output flows provided respectively by the systems S_1 and S_2.

Note 3.3. The notion of weak equivalence can be obviously lifted to implementable transfer functions and will be used in this meaning in the sequel (cf. Theorem 3.5).

We are now in position to give the following result which states that the computational power of systems do not increase if we allow several memory tapes for a system.

Theorem 3.4. Each multi-tape system is weakly equivalent to some mono-tape system.

Proof: The complete proof is presented in [8]. Here we only present the main idea. We proceed in a manner similar to that used by Hopcroft and Ullman in [23] for the proof of the analog result for multi-tape Turing machines. The main additional difficulty of the proof resides in the constraint imposed by the input and output time scales. This is resolved by considering the internal time scale refining that of the original multi-system by dividing each step into $A \times N$ equal parts, where N is the upper bound (that can be an infinite integer of ${}^*\mathbb{N}$) of the memory size and A is a suitable (finite) integer constant (several sweeps of the memory tape can be required to simulate one step of the multi-system). $\square$

The following theorem is also key since it is the implementable transfer functions trace of the fact that systems are closed under an operator – called integration – which is underlying in its proof.

Theorem 3.5. Up to weak equivalence, implementable transfer functions are closed under integration.

Proof: Let $\mathcal{F}_1 : \mathrm{In}_1^{\mathbb{T}_1^i} \to \mathrm{Out}_1^{\mathbb{T}_1^o}$ and $\mathcal{F}_2 : \mathrm{In}_2^{\mathbb{T}_2^i} \to \mathrm{Out}_2^{\mathbb{T}_2^o}$ be two implementable transfer functions associated with the systems S_1 and S_2 and let $\Pi = (\mathrm{In}^{\mathbb{T}^i}, \mathrm{Out}^{\mathbb{T}^o}, \pi^i, \pi^o)$ be an interface for $\mathcal{F}_1$ and $\mathcal{F}_2$. Consider then the multi-system S defined by the following elements:

- its input and output channels are respectively equal to $\mathrm{In}^{\mathbb{T}^i}$ and $\mathrm{Out}^{\mathbb{T}^o}$,
- its internal memory consist in four distinct memory tapes: the two first tapes have respectively the same characteristics than the two memory tapes of S_1 and S_2 (that they simulate) when each of the two other tapes has only one single cell respectively with symbol in $\mathrm{In}_1 \times \mathrm{In}_2$ and $\mathrm{Out}_1 \times \mathrm{Out}_2$,
- its control mechanism works in an internal time refining sufficiently the internal times of S_1 and S_2 to allow the below simulations, has an internal state set equal to the product of the state sets of S_1 and S_2 by the special states product set $\{in_0, in_1, in_2\} \times \{out_0, out_1, out_2\}$ and consists in simulating in parallel the behavior of S_1 and S_2 (the special states being then equal to (in_0, out_0)) when storing or extracting $\pi^i(x :: t, \mathcal{F}_1(x_1) :: t, \mathcal{F}_2(x_2) :: t)$ and $\pi_o(\mathcal{F}_1(x_1) :: t, \mathcal{F}_2(x_2) :: t)$ respectively on the third and fourth memory tapes when necessary (the special states being then respectively equal to $(in_{1,2}, out_0)$ and $(in_0, out_{1,2})$ before going back to their nominal value (in_0, out_0)).

It is then easy to see that the transfer function of system S is observationally equivalent to the integration of the transfer functions of S_1 and S_2, as requested. $\square$

Note 3.6. The notions of interface and integration can be generalised to multi-systems. It is clear that the above theorem also holds for this generalisation.

4. A Notion of Computability on Reals

One of the main difference between the non standard approach and the classical approach reflects in the fact that the computable functions that we can define in our framework are much more rich that in the usual classical frameworks for computability on real numbers (see [11]).

4.1. Computable Real Functions

We shall now introduce a special case of implementable transfer functions, that is to say the class of *computable functions* on reals. To this aim, fix now $\varepsilon \in {}^*\mathbb{R}^+$ and observe first that any non-standard real number can be approximated by a flow of quanta up to ε. Given a non-standard time scale $\mathbb{T}$, we can indeed encode each $x \in {}^*\mathbb{R}$ as a flow $\widetilde{x} \in \{+, -, 1, \flat\}^{\mathbb{T}}$ defined as follows:

- $\widetilde{x}$ begins by a sign equal to '$+$' (when $x \geq 0$) or '$-$' (when $x < 0$),
- $\widetilde{x}$ is followed by a contiguous sequence of N times the symbol 1 where N is the unique (due to Archimedes's principle already used in the proof of Proposition 2.1) non-standard integer of ${}^*\mathbb{N}$ such that $N\,\varepsilon \leq |x| < (N+1)\,\varepsilon$,
- all the remaining symbols of $\widetilde{x}$ are equal to $\flat$.

Let $E(\pm, 1, \flat)$ denote the image of all non-standard real numbers through this encoding, i.e. the set of all $\mathbb{T}$-dataflows on $\{+, -, 1, \flat\}$ that consist of a $\{+, -\}$ symbol, followed by an arbitrary number of 1's and finishing by an infinite sequence of $\flat$ symbols. Any transfer function $\mathcal{F} : \{+, -, 1, \flat\}^{\mathbb{T}} \to E(\pm, 1, \flat)$ induces then a function $\mathcal{F}_\varepsilon : {}^*\mathbb{R} \to {}^*\mathbb{R}$, as a composition of the above encoding, the transfer function $\mathcal{F}$ and the corresponding decoding. Notice that the previous encoding is injective on $\mathbb{R}$ when $\varepsilon \approx 0$. In the following, we shall therefore fix $\varepsilon = 1/E$ where $E \in {}^*\mathbb{N}$ is some infinite positive integer.

Definition 4.1. Computable functions – A function $f : \mathbb{R} \to \mathbb{R}$ is *computable* if there exists an implementable transfer function $\mathcal{F} : \{+, -, 1\}^{\mathbb{T}} \to E(\pm, 1, \flat)$, such that one has $f(x) \approx \mathcal{F}_\varepsilon(x)$ for every $x \in \mathbb{R}$. Moreover a computable function f is said to be computable with *bounded error* if there exists an *error bound* $B \in {}^*\mathbb{N}$ with $B\varepsilon \approx 0$ such that the following property holds for any $N \in {}^*\mathbb{Z}$:

$$N\varepsilon \text{ is finite } \Longrightarrow |\mathcal{F}_\varepsilon(N\varepsilon) - {}^*f(N\varepsilon)| < B\varepsilon \ , \qquad (8)$$

where ${}^*f$ stands for the non-standard version of f [16] .

Note 4.1. Observe that sequences of real numbers are functions with support $\mathbb{N}$. Hence *computable sequences* constitute a special case of computable functions.

The following result shows now that non-standard modelling gives raise to the same kind of computable functions than in most of the usual real-oriented computability frameworks (see [11]).

Theorem 4.2. The set of bounded error computable functions is closed under addition, multiplication, and integration. Moreover addition and multiplication preserves the order of the error bound.

[16] I.e. the function ${}^*f$ defined by setting ${}^*f([(x_n)_{n\geq 0}]) = [(f(x_n))_{n\geq 0}]$, going back to the formalism of Section 1.2.

Proof: 1. Closure under addition and preservation of the error bound order results immediately from the construction (let to the reader) of a two memory tapes system that computes the sum $(N+M)\,\varepsilon$ of two non-standard real numbers $N\,\varepsilon$ and $M\,\varepsilon$ with $N, M \in {}^*\mathbb{Z}$ (each of them being initially put on one of the system's tapes according to the above encoding).

2. Closure under multiplication will result immediately of the construction of a two memory tapes system that computes up to ε the product $(N\varepsilon) \times (M\varepsilon)$ of two non-standard positive real numbers $N\varepsilon$ and $M\varepsilon$ with $N, M \in {}^*\mathbb{N}$. Using the Euclidian division algorithm (that can be easily simulated by a system), one can compute two unique non-standard integers Q and R such that $MN = QE + R$ and $0 \leq R < E$ (where E is the infinite non-standard integer such that $\varepsilon = 1/E$). Since one clearly has

$$(M/E)\,(N/E) = (Q/E) + (R/E^2) \implies Q\,\varepsilon \leq M\,N\,\varepsilon^2 < Q\,\varepsilon + \varepsilon\,,$$

it happened that the positive non-standard real number $Q\varepsilon$ is hence a relevant approximation – up to ε – of the product $(N\varepsilon) \times (M\varepsilon)$ that can be system-computed as expected. Proof of the error bound order preservation is straightforward.

3. Let $[a, b] \subset \mathbb{R}$ be a usual interval and let $f : \mathbb{R} \to \mathbb{R}$ be a continuous standard function computable with some error bound $B \in {}^*\mathbb{N}$. For any infinite non-standard integer $N \in {}^*\mathbb{N}$, we then have

$$\int_a^b f(x)dx \approx \frac{1}{N} \Big(\sum_{i=0}^{N} {}^*f\big(a + i\,\frac{b-a}{N}\big) \Big)\ .$$

Taking $N = \lfloor \sqrt{E/B} \rfloor$ and applying Euclidean division algorithm as above, it is easy to verify that the right-hand side of this equation can be implemented, given an implementation of the integration of f. □

The two next results are now concentrating the main differences between our non-standard computability model and the usual real-oriented computability models.

Theorem 4.3. Let $f : \mathbb{R} \to \mathbb{R}$ be a uniformly continuous differentiable function, computable with error bound $B \in {}^*\mathbb{N}$. Then its derivative f' is also computable with bounded error.

Proof: Let $\mathcal{F}$ be an implementable transfer function such that $f(x) \approx \mathcal{F}_\varepsilon(x)$ for every $x \in \mathbb{R}$ (with associated error bound B). Recall first that the derivative $f'(x)$ can be approximated for any $x \in \mathbb{R}$ by the formula $f'(x) \approx ({}^*f\,(x+\delta) - {}^*f(x))\,/\delta$ where δ stands for any infinitesimal. Let us now consider a real number $x \in \mathbb{R}$ and let $M \in {}^*\mathbb{Z}$ be the unique infinite non-standard integer such that $0 \leq x - M\,\varepsilon < \varepsilon$. Then one has for any non-standard positive integer $N \in {}^*\mathbb{N}$ such that $\delta = N\varepsilon$ remains infinitesimal:

$$\left| \frac{{}^*f(M\varepsilon + N\varepsilon) - {}^*f(M\varepsilon)}{\delta} - \frac{\mathcal{F}_\varepsilon(M\varepsilon + N\varepsilon) - \mathcal{F}_\varepsilon(M\varepsilon)}{\delta} \right| < 2\,B\varepsilon/\delta\ . \qquad (9)$$

Consider $N = \lfloor \sqrt{EB} \rfloor$. We then have both $\delta = N\varepsilon \approx 0$ and $(B\varepsilon)/\delta = B/N \approx \sqrt{B\varepsilon} \approx 0$. Therefore $f'(x)$ can be always approximated by $(E/N)\,\big(\mathcal{F}_\varepsilon(x + N\varepsilon) - \mathcal{F}_\varepsilon(x)\big)$. Using this last formula in connection with Equation 9, it is now immediate to show that f' is computable with bounded error. □

Arguing as in the third point of Theorem 4.2, we can state the following remarkable computability result which is totally specific to the non-standard modelling we used here.

Theorem 4.4. Let $(a_n)_{n\in\mathbb{N}}$ be a sequence of real numbers computable with bounded error and such that the function $f(x) = \sum_{n\in\mathbb{N}} a_n\,x^n$ is analytical. Then f is computable with bounded error.

4.2. Dynamical and Hamiltonian Systems

A *dynamical system* $x : t \in \mathbb{R} \longrightarrow x(t) \in \mathbb{R}^n$ is defined by a differential equation of the type:

$$x(t_0) = x_0 \in \mathbb{R}^n \quad \text{and} \quad \dot{x}(t) = f(t, x(t)) \text{ for every } t \in \mathbb{R} \ . \tag{10}$$

If $f : [0,1] \times \mathbb{R}^n \to \mathbb{R}^n$ is a continuous bounded function, then Equation 10 admits a unique solution called the *trajectory* of the corresponding dynamical system (cf. [41]). The following theorem extends then the results of the previous sub-section to computability of dynamical systems.

Theorem 4.5. Let S be a dynamical system defined by a bounded continuous function f. If f is computable with bounded error, then the unique trajectory of S is also computable with bounded error.

Proof: The proof of this theorem is based on the non-standard proof of Peano theorem (see [6, Secn. A.4.2] or [14]) and techniques similar to those applied in the previous section. Let $N \in {}^*\mathbb{N}$ be a non-standard infinite integer. The key point is to express the solution of a dynamical system using the formalism of non-standard analysis. It appears that Equation (10) admits a solution $X : [0,1] \to {}^*\mathbb{R}$ defined inductively (in the non-standard meaning) by setting for every non-standard integer $0 \leq k \leq N$:

$$X(k/N) = x_0 + \frac{1}{N} \Big(\sum_{i=0}^{k-1} f(i/N, X(i/N)) \Big) \ . \tag{11}$$

Using Equation (11), we can build a system that computes X. This system takes analog encodings of real numbers of $[0,1]$ on its input that are encoded using the infinitesimal $\varepsilon = 1/N$ and is designed in such a way that when it reads the k^{th}-1, it stores the value $X(k/N)$ on its memory tape. To achieve this, the system simply adds to the non-standard integer value representing $X(k - 1/N)$ already stored on its memory tape the non-standard integer $\lfloor f((k-1)/N, X(k - 1/N)) \rfloor$. Since f is computable, this algorithm is easy to implement using techniques already used in the previous sub-section. $\square$

An important special case of dynamical systems is constituted by the *closed Hamiltonian systems*, defined by a system of differential equations of the form:

$$\begin{cases} p(t_0) = p_0, & \dot{p} = -\frac{\partial H}{\partial q} \\ q(t_0) = q_0, & \dot{q} = \frac{\partial H}{\partial p} \ , \end{cases} \tag{12}$$

where $H(p, q)$ is an Hamiltonian, modelling the total energy of the system. As an immediate consequence of the previous result, we get the following corollary which shows that most of the classical physical systems can be modelled by systems in our meaning. Note also that this result shows the key difference between our model and the classical ones since one can prove that solutions of Hamiltonian systems cannot be algorithmically computed in a usual computable framework (see [15, 35, 37]).

Corollary 4.1. Let S be a closed Hamiltonian system such that both $\partial H/\partial p$ and $\partial H/\partial q$ are bounded continuous functions that are computable with bounded error. Then the unique trajectory of S is also computable with bounded error.

The discussion above allows us to reconsider the simple pendulum example (Example 3.1) by decomposing it into simpler components.

Example 4.1. Simple pendulum revisited – Let us first recall that the motion of the simple pendulum as described in Example 3.1 is modelled by the differential equation (5). Denoting $\varphi = \dot{\theta}$ the angular speed of the pendulum, this equation defines the following Hamiltonian system:

$$\varphi(0) = 0,\ \dot{\varphi} = -\frac{\partial H}{\partial \theta} \quad \text{and} \quad \theta(0) = \theta_0,\ \dot{\theta} = \frac{\partial H}{\partial \varphi}$$

with $H(\varphi, \theta) = \frac{\varphi^2}{2} - \frac{g}{L}\cos\theta$. Thus, another model of such a pendulum can be obtained by considering a system with two tapes storing respectively the values of the angle θ and angular speed φ. This system will use two "sub-systems" computing respectively the Hamiltonian $H(\theta, \varphi)$ and the trajectories that it defines, these components consisting themselves of other "sub-systems" (the system computing $H(\theta, \varphi)$ use for instance a sub-system to compute the cosine function).

5. Conclusion

In this paper, we presented a new family of timed systems encompassing classical approaches based on continuous and discrete time models. The key idea of our approach is to use a non-standard framework where continuous time is modelled discretely through continuous time scales. This change of modelling paradigm, joined with the fact that non-standard analysis allows to refine time scales without difficulties, allowed us to enlarge the perimeter of usual real-oriented computability frameworks. It indeed happens that a natural class of computable real functions in our formalism is closed under operations such as derivation and integration (which is usually not the case) and contains interesting functions such as most of the classical analytical functions. As a special case, trajectories of Hamiltonian and dynamical systems are computable under assumption of computability of their parameters. This shows that a large class of physical systems can be modelled in this framework. Defining a system as a natural extension of a Turing machine mechanism, lead us therefore to a formal definition of systems, where both software and physical systems can be modelled and described in a unified way. To the best of our knowledge, few models have so far been developed that combine all the above mentioned characteristics. We believe therefore that our model provides an elegant and powerful mathematical framework for modelling complex systems and hope that it will give new intuitions for further researches.

Acknowledgements

The authors would like to thank the anonymous referees for many valuable remarks that helped to seriously improve the first version of the present paper.

References

[1] Akl S., *Three counterexamples to dispel the myth of the universal computer*, Parallel Processing Letters, **16**, (3), 381–403, 2006.

[2] Alur R., Courcoubetis C., Halbwachs N., Henzinger T. A., Ho P. H., Nicollin X., Olivero A., Sifakis J., Yovine S., *The Algorithmic Analysis of Hybrid Systems*, Theor. Comp. Sci., **138** (1), 3–34, 1995.

[3] Alur R., Courcoubetis C., Henzinger T.A., Ho P.H., *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems*, Lect. Notes in Comp. Sci., **736**, 209–229, Springer, 1993.

[4] Bacon J., *Concurrent systems - An Integrated Approach to Operating Systems, Database, and Distributed Systems*, Addison Wesley, 1992.

[5] Barwise J., *Handbook of Mathematical Logic*, Studies in Logic and the Foundations of Mathematics, **90**, North Holland, 1977.

[6] Bliudze S., *A Framework for Studying Complex Industrial Systems: An Example Based on the UMTS Infrastructure*, Ph.D. Thesis, École Polytechnique, 2006, `http://www-verimag.imag.fr/~bliudze`.

[7] Bliudze S., Krob D., *Towards a Functional Formalism for Modelling Complex Industrial Systems*, ComPlexUs, in [Special Issue: Complex Systems – European Conference – November 2005 – Selected Papers – Part 1, Bourgine P., Képès F., Schoenauer M., Eds.], **2** (3–4), 163–176, 2004/2005.

[8] Bliudze S., Krob D., *Modelling of Complex Systems I — A Functional Approach: Time, Data and Systems*, Technical report, LIX, École Polytechnique, 2007,

[9] Blum L., Cucker F., Shub M., Smale S., *Complexity and Real Computation*, Springer, 1998.

[10] Börger E., Stärk R., *Abstract State Machines – A method for high-level system design and analysis*, Springer, 1998.

[11] Bournez O., Campagnolo M.L., *A Survey on Continuous Time Computations*, in ["New Computational Paradigms. Changing Conceptions of What is Computable", Cooper S.B., Löwe B., Sorbi A., Eds.], 383–423, Springer, 2008.

[12] Cha D., Rosenberg J., Dym C., *Fundamentals of Modeling and Analyzing Engineering Systems*, Cambridge University Press, 2000.

[13] Cousot P., Cousot R., *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*, in ["Conference Record of the Fourth Annual ACM Symposium on Principles of Programming Languages", Los Angeles, California], 238–252, ACM Press, 1977.

[14] Cutland N., *Nonstandard Analysis and its Applications*, London Mathematical Society Student Texts, **10**, Cambridge University Press, 1988.

[15] Da Costa N., Doria F., *Undecidability and incompleteness in classical mechanics*, Int. Journal of Theor. Physics, **30**, 1041–1073, 1991.

[16] D'Alembert Le Rond, J. dit, *Article "Différentiel"*, in [Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers, Diderot D., D'Alembert Le Rond J. dit, Eds.], **4**, 985–989, Briasson, David, Le Breton et Durand, Paris, 1754.

[17] Davis M. *Applied nonstandard analysis*, John Wiley, 1977.

[18] Delfini P., Lobry C. *The Vibrating String*, in ["Nonstandard Analysis in Practice", Diener F., Diener M., Eds.], Springer, 1995.

[19] Diener F., Diener M., *Nonstandard Analysis in Practice*, Springer, 1995.

[20] Diener F., Diener M., *Tutorial*, in ["Nonstandard Analysis in Practice", Diener F., Diener M., Eds.], Springer, 1995.

[21] Diener F., Diener M., *Ducks and rivers: three existence results*, in ["Nonstandard Analysis in Practice", Diener F., Diener M., Eds.], Springer, 1995.

[22] Diener F., Reeb G., *Analyse non standard*, Hermann, 1989.

[23] Hopcroft J. E., Ullman J. D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[24] Jensen C. U., Lenzing H., *Model Theoretical Algebra*, Gordon and Breach, 1989.

[25] Kanovei V., Reeken M., *Nonstandard analysis, axiomatically*, Springer Verlag, 2004.

[26] Kossiakoff A., Sweet W.N., *Systems Engineering – Principles and practice*, Wiley Series in Systems Engineering, 2003.

[27] Krob D., *Modelling of Complex Software Systems: A Reasoned Overview*, in ["Formal Techniques for Networked and Distributed Systems" (FORTE'06), Najm E., Pradat-Peyre J.-F., Donzeau-Gouge V., Eds.], **4229**, Springer, 2006.

[28] Lakatos I., *Preuves et réfutations*, Hermann, 1984.

[29] Lamport L., *Specifying Systems – The TLA+ Language and Tools for Hardware and Software Engineers*, Addison-Wesley, 2003.

[30] Lygeros J., *Lecture notes on hybrid systems*, Notes for an ENSIETA Workshop, 2004.

[31] Marwedel P., *Embedded System Design*, Kluwer, 2003.

[32] Meyer K.R., Hall G.R., *Introduction to Hamiltonian Dynamical Systems and the N-Body Problem*, Applied Mathematical Sciences, **90**, Springer, 1992.

[33] Moore C., *Recursion theory on the reals and continuous-time computation*, Theor. Comp. Sci., **162** (1), 23–44, 1996.

[34] Nicolis G., Nicolis C., *Foundations of Complex Systems*, World Scientific, 2007.

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

[36] Rabinovitch A., *A*utomata over Continuous Time, Theor. Comput. Sci., **300**, 331–363, 2003.

[37] Richardson D., *Some undecidable problems involving elementary functions of a real variable*, The Journal of Symb. Logic, **33**, (4), 514–520, 1968.

[38] Robert F., *Les systèmes dynamiques discrets*, Mathématiques et Applications, **19**, Springer, 1994.

[39] Robinson A., *Non Standard Analysis*, North Holland, 1966.

[40] Sage A.P., Armstrong J.E., *Introduction to systems engineering*, Wiley, 2000.

[41] Severance F. L., *System Modeling and Simulation: An Introduction*, John Wiley & Sons, 2001.

[42] Schneider K., *Verification of reactive systems – Formal methods and algorithms*, Springer, 1998.

[43] Sontag E., *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Textbooks in Applied Mathematics, **6**, Springer, 1998.

[44] Trakhtenbrot B.A., *Understanding Basic Automata Theory in the Continuous Time Setting*, Fundam. Inform., **62**, (1), 69–121, 2004.

[45] Troesch A., Urlacher E., *I. Analyse non standard et équations de Van der Pol. II. Perturbations singulières et analyse non standard*, Publications de l'IRMA, Strasbourg, 1977.

[46] Turner W.C., Mize J.H., Case K.E., Nazemetz J.W., *Introduction to industrial systems and systems engineering*, Prentice Hall, 1993.

[47] Zaytoun J., *Systèmes dynamiques hybrides*, Hermes, 2001.

[48] Zeigler B. P., Praehofer H., Gon K. T., *Theory of Modeling and Simulation — Integrating Discrete Event and Continuous Complex Dynamic Systems*, Academic Press, 2000.

[49] Zurawski R., *Embedded Systems Handbook*, CRC Press, 2006.