Modelling of Complex Systems I

Systems as data-flow machines *

Simon Bliudze[1,**], Hugo Gimbert[2], and Daniel Krob[2]

[1] VERIMAG, Centre Équation, 2 av de Vignate, 38610, Gières, France
Simon.Bliudze@imag.fr
[2] LIX, École Polytechnique, 91128, Palaiseau, France
{gimbert,dk}@lix.polytechnique.fr

Abstract. We develop a unified functional formalism for modelling complex systems, that is systems composed of a number of components, including software and physical devices. Continuous time is modelled as a discrete sequence by using non-standard analysis techniques.
We define systems as generalised Turing machines with temporised operation, and input and output channels. Behaviours of systems are represented by transfer functions. A transfer function is said to be *implementable* if it is associated with a system. This also allows to define *computable* functions on reals.
We show that these notions are robust: the class of implementable transfer functions is closed under abstraction and integration (cf. Th. 2); the class of computable functions includes analytical functions whose coefficients are computable, and is closed under addition, multiplication, differentiation and integration (cf. Th. 3). In particular, the class of computable functions includes solutions of dynamical and Hamiltonian systems defined by computable functions. Hence, our notion of system is suitable for modelling physical systems.

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

1 Introduction

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

* Ces travaux ont été réalisés dans le contexte du projet Usine Logicielle du pôle System@tic Paris-Région avec le soutien de la Direction Générale des Entreprises, du Conseil Régional d'Ile de France, du Conseil Général des Yvelines, du Conseil Général de l'Essonne et du Conseil Général des Hauts de Seine.

** Part of this work was realised while this author was at LIX, École Polytechnique.

or computer science. Think for instance of dynamical, mechanical, Hamiltonian, hybrid, holonomic, embedded or distributed systems. 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, one may therefore naturally think that it is totally impossible to give any precise mathematical definition of a system, even though a few number of authors tried to propose global frameworks for discussing systems. In particular, one should point out [**?**, **?**, **?**], where one can find different generic definitions for continuous systems and [**?**] for a typical software approach where systems are defined through their capacity of being specified in a given specification language.

Our paper intends, however, to address this theoretical challenge. It indeed tries to propose what could be the basis of a general system theory — in the line of the classical algorithmic complexity theory, systems being here defined as a new kind of computing timed-machines — by going back to the very fundamentals, that is to say by looking for *a unified definition of a system* from which the different existing system models could be deduced.

Our main idea is to integrate in a common "machine" model the key features that can be found in quite all concrete "systems": firstly, the fact that systems are fundamentally characterised by a temporal behaviour 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* other smaller sub-systems and *abstracting* the result of this integration process (i.e. manipulating system parameters that are functions of sub-systems parameters).

We therefore propose here a formal definition of a system which intends to capture in a common framework both continuous and discrete systems, which are the two major types of systems that can be found in practice, especially in industrial applications. The key point on which our approach relies is a common (discrete) modelling of time, based on the use of a non-standard model of real numbers. Making this (very strong) change allows us indeed to take into account in the same way both physical systems and computer systems, while keeping several natural system intuitions. The behaviours of our systems are described using transfer functions, thus defining the class of implementable transfer functions. This class of functions enjoys several fundamental properties. First, it is closed under abstraction and integration, cf. Th. 2, which is typically required in the context of software engineering. Moreover, the sub-class of computable functions is stable under addition, multiplication, differentiation, integration and contains analytical functions whose coefficients are themselves computable. Thus, solutions of dynamical and Hamiltonian systems defined using computable functions are themselves computable. This proves that our notion of system is also suitable for taking into account physical systems.

Due to the fact that our systems 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 can 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.

2 Non-Standard Analysis

To develop a global (discrete) unified framework for dealing both with continuous and discrete systems, we go back to the 18th century representation of real numbers (see [**?**, **?**]).

Indeed, in the 17th and 18th centuries the common idea of real numbers was different from the modern one. The reasoning of the Differential Calculus was carried out with the help of the quantities called "infinitesimals". For instance, to compute the derivative of a given function $f(x)$, one would consider the *increment* of this function given an *infinitesimal* increment dx to x. Thus the derivative $f'(x)$ was defined by setting

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

For example, applying this reasoning to $f(x) = x^2$, one obtains the following computation

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

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

The notion of infinitesimal was formalised in 1960s by Robinson (see [**?**]), by defining a set ${}^*\mathbb{R}$ of *non-standard reals*,[3] which is a real-closed field that contains all usual real numbers, but also the infinitesimal reals (i.e. the non-zero non-standard real numbers that have their absolute value strictly less than any $r \in \mathbb{R}_*^+$) and their inverses, which are the infinitely great reals, i.e. those with an absolute value strictly greater than any usual real number $r \in \mathbb{R}$ (see Fig. 1).

In particular, two non-standard real numbers x and y are said to be *infinitely close* (denoted by $x \approx y$) if and only if $x - y$ is infinitesimal.

The field ${}^*\mathbb{R}$ is elementarily equivalent to $\mathbb{R}$, 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 (see [**?**] and [**?**] for more model theoretical insights into the non-standard analysis). 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 standard integers, contains infinitely great ones, having absolute value greater than any $n \in \mathbb{N}$.

More details about non-standard analysis can be found in [**?**] and [**?**]; a brief introduction can also be found in [**?**, Appendix A].

[3] A star on the left of a symbol as in ${}^*\mathbb{R}$ stands always for "non-standard", whereas on the right, it means that zero is not included, as in $\mathbb{R}^*$ or ${}^*\mathbb{R}^*$ (denoting respectively the usual and the non-standard sets of non-zero real numbers).

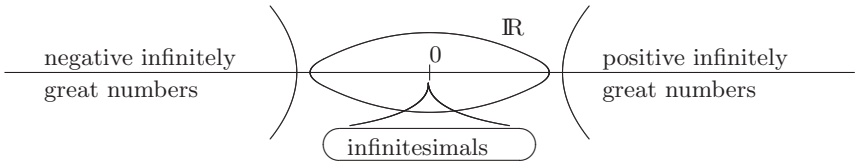


Fig. 1. Graphical representation of non-standard real numbers

2.1 Time Scales, Data Flows and Transfer Functions

Definition 1 (Time scales). *A time scale* $\mathbb{T}$ *is a subset of* ${}^*\mathbb{R}$ *for any* $t \in {}^*\mathbb{R}$*, the set* $\{t' \in \mathbb{T} \mid t' < t\}$ *has a maximum and the set* $\{t' \in \mathbb{T} \mid t' > t\}$ *has a minimum.*

Note that the condition, imposed on the time scales in the definition above, ensures that, for any $t \in \mathbb{T}$, its successor (denoted $succ_{\mathbb{T}}(t)$) and predecessor (denoted $pred_{\mathbb{T}}(t)$) can be uniquely defined.

Example 1 (Regular time scales). One important class of time scales consists of *regular time scales* [**?**, **?**]. Let $\tau \in {}^*\mathbb{R}^+_*$ be a strictly positive non-standard real number. The set $\mathbb{T}_\tau = {}^*\mathbb{Z}\,\tau$ is a regular time scale with step τ.

Such a time scale can be seen as a discrete series of clock ticks occurring at times $n \cdot \tau$, with $n \in {}^*\mathbb{Z}$ being a non-standard integer, potentially infinitely great.

$$-N\tau \;\ldots\; -2\tau \quad -\tau \quad 0 \quad \tau \quad 2\tau \;\ldots\; N\tau = \infty$$

It can be shown that we can recover usual continuous time by considering, for example, regular time scales with infinitesimal steps (i.e. with $\tau \approx 0$). □

Definition 2. *A time scale* $\mathbb{T}_\tau$ refines *another time scale* $\mathbb{T}_{\tau'}$ *(denoted by* $\mathbb{T}_{\tau'} \succeq \mathbb{T}_\tau$*), iff* $\mathbb{T}_{\tau'} \subseteq \mathbb{T}_\tau$*.*

Definition 3 (Data flow). *Let* A *be an arbitrary alphabet, we will denote by* $\overline{A}$ *this alphabet completed by a* blank symbol $\flat$*, i.e.* $\overline{A} = A \bigsqcup \{\flat\}$*. A* $\mathbb{T}$-data flow over A *is a function* $x : \mathbb{T} \to \overline{A}$*. The set of all* $\mathbb{T}$*-data flows over* A *is denoted* $A^{\mathbb{T}}$*.*

Although the value of a data flow only changes at discrete moments specified by its time scale, a data flow can be observed at any time.

Definition 4 (Snapshots). *Let* $\mathbb{T}$ *be a time scale, let* x *be a* $\mathbb{T}$*-flow over an arbitrary alphabet* A*, and let* $t \in {}^*\mathbb{R}$ *be any non-standard real. The* snapshot of x at time t *is* $x::t$ *defined by:*

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

Definition 5 (Observational equivalence). *Let x be a $\mathbb{T}_1$-flow and y be a $\mathbb{T}_2$-flow over a common alphabet A. We say that x and y are* observationally equivalent *(denoted $x \simeq y$) if, for all $t \in {}^*\mathbb{R}$, we have $x :: t = y :: t$.*

Observable behaviour of a system is described by a corresponding transfer function. A transfer function can be seen as a real-time transformation of a data flow satisfying the causality condition.

Definition 6 (Transfer functions). *A transfer function on data flows from $\mathrm{In}^{\mathbb{T}^{\mathrm{i}}}$ and $\mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}$, is a function $\mathcal{F} : \mathrm{In}^{\mathbb{T}^{\mathrm{i}}} \to \mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}$ which has the causality property: for any $t \in {}^*\mathbb{R}$ and $x, x' \in \mathrm{In}^{\mathbb{T}^{\mathrm{i}}}$,*

$$(\forall t' < t, x :: t' = x' :: t') \implies (\mathcal{F}(x) :: t = \mathcal{F}(x') :: t) \quad .$$

The behaviour of a system may be described at various levels of precision. Forgetting part of the behaviour of a system consists in the *abstraction* operation, whereas a more precise description of a system is obtained by a *refinement* operation. These notions are described by Fig. 2 and formalised as follows:

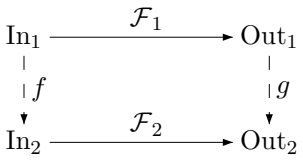


Fig. 2. Transfer function $\mathcal{F}_2$ is an abstraction of $\mathcal{F}_1$, conversely $\mathcal{F}_1$ is a refinement of $\mathcal{F}_2$

Definition 7 (Abstraction of transfer functions). *A transfer function $\mathcal{F}_2 : \mathrm{In}_2^{\mathbb{T}_2^{\mathrm{i}}} \to \mathrm{Out}_2^{\mathbb{T}_2^{\mathrm{o}}}$ is an* abstraction *of a transfer function $\mathcal{F}_1 : \mathrm{In}_1^{\mathbb{T}_1^{\mathrm{i}}} \to \mathrm{Out}_1^{\mathbb{T}_1^{\mathrm{o}}}$ if there exist two mappings $f : \mathrm{In}_1 \to \mathrm{In}_2$ and $g : \mathrm{Out}_1 \to \mathrm{Out}_2$, such that f is onto and for any data flow $x \in \mathrm{In}_1^{\mathbb{T}_1^{\mathrm{i}}}$ and any $t \in {}^*\mathbb{R}$,*

$$\mathcal{F}_2(f(x)) :: t = g(\mathcal{F}_1(x) :: t) \quad . \tag{1}$$

In this case, $\mathcal{F}_1$ is a refinement *of $\mathcal{F}_2$. Two transfer functions, $\mathcal{F}_1$ and $\mathcal{F}_2$, are* equivalent *if they mutually abstract each other.*

Notice that, for two transfer functions, $\mathcal{F}_1$ and $\mathcal{F}_2$, such that the set of output data flows of $\mathcal{F}_1$ coincides with the set of input data flows of $\mathcal{F}_2$ the composition function $\mathcal{F}_1 \circ \mathcal{F}_2$ is causal, hence it is a transfer function.

However, this notion of composition is not rich enough to capture the complexity of components integration arising in engineering processes, where integration of components is realised through *interfaces*. Interfaces are used to pipe various inputs and outputs of systems together. The integration of transfer functions using interface is depicted on Fig. 3 and formally defined as follows:

Definition 8 (Interfaces and integration of transfer functions). *Let* $\mathcal{F}_1 : \mathrm{In}_1^{\mathbb{T}_1^{\mathrm{i}}} \to \mathrm{Out}_2^{\mathbb{T}_2^{\mathrm{o}}}$ *and* $\mathcal{F}_2 : \mathrm{In}_2^{\mathbb{T}_2^{\mathrm{i}}} \to \mathrm{Out}_2^{\mathbb{T}_2^{\mathrm{o}}}$ *be two transfer functions. An interface for* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *is a tuple* $\Pi = (\mathrm{In}^{\mathbb{T}^{\mathrm{i}}}, \mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}, \pi^i, \pi^o)$ *where* π^i *and* π^o *are mappings:*

$$\pi^i : \mathrm{In} \times \mathrm{Out}_1 \times \mathrm{Out}_2 \to \mathrm{In}_1 \times \mathrm{In}_2$$
$$\pi^o : \mathrm{Out}_1 \times \mathrm{Out}_2 \to \mathrm{Out}$$

The integration of $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *through* Π *is the transfer function* $\mathcal{F} : \mathrm{In}^{\mathbb{T}^{\mathrm{i}}} \to \mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}$ *denoted* $\mathcal{F}_1 \rhd_\Pi \mathcal{F}_2$ *and defined as follows. For each input flow* $x \in \mathrm{In}^{\mathbb{T}^{\mathrm{i}}}$ *we define inductively the output flow* $\mathcal{F}(x) \in \mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}$ *together with two data flows* $x_1 \in \mathrm{In}_1^{\mathbb{T}_1^{\mathrm{i}}}$ *and* $x_2 \in \mathrm{In}_2^{\mathbb{T}_2^{\mathrm{i}}}$*:*

$$\begin{aligned} &\forall t \in \mathbb{T}_1^i, \quad x_1(t) = \pi_1^i(x::t, \mathcal{F}_1(x_1)::t, \mathcal{F}_2(x_2)::t) \\ &\forall t \in \mathbb{T}_2^i, \quad x_2(t) = \pi_2^i(x::t, \mathcal{F}_1(x_1)::t, \mathcal{F}_2(x_2)::t) \\ &\forall t \in \mathbb{T}^o, \quad \mathcal{F}(x)(t) = \pi^o(\mathcal{F}_1(x_1)::t, \mathcal{F}_2(x_2)::t) \ . \end{aligned} \tag{2}$$

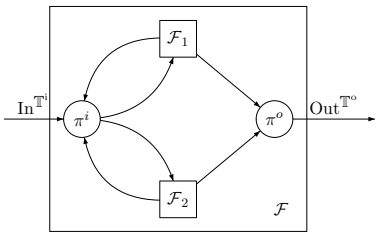


Fig. 3. Integration of two transfer functions $\mathcal{F}_1, \mathcal{F}_2$ through an interface $(\mathrm{In}^{\mathbb{T}^{\mathrm{i}}}, \mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}, \pi^i, \pi^o)$

Lemma 1. *Equation (2) correctly defines a transfer function.*

Proof. At any given moment t values of $x_1(t), x_2(t)$, and $\mathcal{F}(x)(t)$ only depend on $x_1(t'), x_2(t')$ and $\mathcal{F}(x)(t')$ for $t' < t$. Consequently, $\mathcal{F}$ is causal, and therefore it is a transfer function. □

3 Systems

In this section, we develop notions of a system, the transfer function associated with a system and integration of systems.

3.1 Definition

Definition 9 (Systems). *A* system S *is defined as the union of the following elements:*

- *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 *— data that belong to a given alphabet* In*, called the* input domain *of* S *(and also 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 *— data that belong to a given alphabet* Out*, called the* output domain *of* S *(and also denoted by* $\mathrm{Out}(S)$*);*

  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*.*
  - *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 non-standard finite set* Q *(also denoted by* $State(S)$*);*
  - *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;*
- *and, finally, an initial state given by the set of values that the corresponding components above are assigned at the moment zero.*

A graphical representation of a system defined in this way is given in Fig. 4.

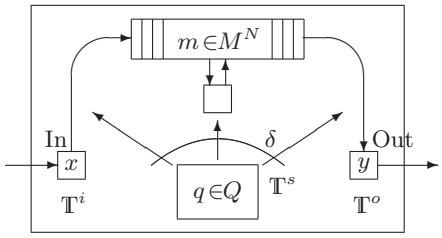


Fig. 4. Graphical representation of a system

The model defined in this way can be considered as a generalisation of a Turing machine, enhanced essentially by the following three notions

1. the input/output mechanism (the two respective channels),
2. temporisation (input, output, and internal time scales),
3. and the non-standard analysis approach, which allows to work with continuous values.

Let us now consider the dynamics of a system in this model. To do so, we remark 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 the read/write head. The following definition introduces this idea in a formal manner.

Definition 10 (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 $d = (q, m, p, \tilde{y}) \in Q \times M^N \times {}^*\mathbb{N} \times \overline{\mathrm{Out}}$, 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.*

We can now introduce the dynamics of a system by considering a sequence of its instantaneous descriptions indexed by the moments on its internal time scale.

Definition 11 (Execution of a system). *An* execution *(or, alternatively, a* calculation*) of a system S* coupled with an input flow x *is the $\mathbb{T}^s$-flow $d = (q, m, p, \tilde{y})$ over $Q \times M^N \times {}^*\mathbb{N} \times \overline{\mathrm{Out}}$, defined as follows. Let $t\mathbb{T}^s$ and $t' = \min\{t' > t \mid t' \in \mathbb{T}\}$, then*

$$\begin{array}{ll} q(t') = q', & \\ m(t')_j = m(t)_j, & \text{for } j \neq p(t), \\ m(t')_j = m', & \text{for } j = p(t), \\ p(t') = p(t) + \Delta, & \\ \tilde{y}(t') = \tilde{y}(t), & \text{if } \tilde{y}\,' = \flat, \\ \tilde{y}(t') = \tilde{y}\,', & \text{if } \tilde{y}\,' \neq \flat, \end{array} \tag{3}$$

where τ_s is the time step of the internal time scale $\mathbb{T}^s$, and

$$(q', m', \Delta, \tilde{y}') = \delta\Big(q(t), m(t)_{p(t)}, x :: t\Big)\,,$$

with $m(t)_{p(t)}$ representing the value of the memory tape cell pointed by the read/write head at the moment t, and $x :: t$ being the snapshot of the input flow.

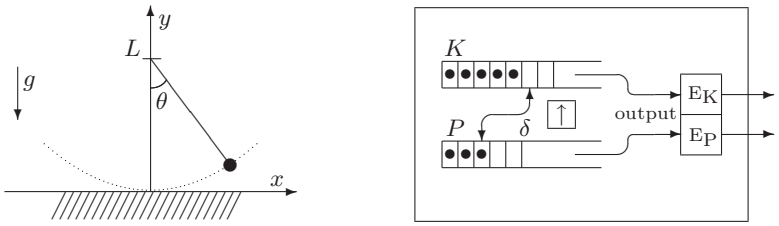
The system's output flow $y \in \mathrm{Out}^{\mathbb{T}^o}$ generated by x *is defined by:*

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

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 the transition function δ to the snapshot of the input flow, i.e. the latest received value.

In [**?**], we show that this notion of systems can be naturally generalised to include systems with multiple input/output channels and memory tapes, which we call *multi-systems*. As stated by Th. 1, in the following section, this generalisation does not increase computational power of the model.

Example 2 (Simple pendulum). Consider a pendulum consisting of a point mass m hanging on a rigid string of negligible mass and of length L (see Fig. 5-a).



(a) (b)

Fig. 5. Simple pendulum: mechanical (a) and systemic (b) representations

The motion of such a pendulum is described by the differential equation

$$\ddot{\theta} = -\frac{g}{L}\sin\theta \;, \tag{4}$$

where θ is the angle formed by the string and the y axis. By integrating (4) we obtain

$$\frac{1}{2}\,m(L\dot{\theta})^2 + mgL(1-\cos\theta) = C \,, \tag{5}$$

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 equation can be easily obtained from (5)

$$\left|\frac{d\,\mathbb{E}_K}{dt}\right| = \left|\frac{d\,\mathbb{E}_P}{dt}\right| = \sqrt{\frac{2\,\mathbb{E}_K\mathbb{E}_P}{L}\left(2g - \frac{\mathbb{E}_P}{mL}\right)} \,. \tag{6}$$

A system modelling such a pendulum is shown in Fig. 5-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 (thus the total energy is constant and equal to C). The system's behaviour consists then essentially in taking at each moment of time the number of quanta, defined by (6), from one tape and putting it on the other one, until the working tape is empty (see [**?**] for a detailed description). □

3.2 Transfer Functions and Equivalence of Systems

A transfer function can be naturally associated to any system. We call such transfer functions *implementable*.

Definition 12 (Implementable transfer functions). *Let S be a system with input flows in $\mathrm{In}^{\mathbb{T}^{\mathrm{i}}}$ and output flows in $\mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}$. Let*

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

be the mapping which associates to any input flow $x \in \mathrm{In}^{\mathbb{T}^{\mathrm{i}}}$ the output flow $y \in \mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}$ generated by S. Then $\mathcal{F}_S$ is causal and is called the transfer function associated with S. *We say that a transfer function $\mathcal{F}$ is* implementable *if there exists S such that $\mathcal{F} = \mathcal{F}(S)$.*

The notion of the transfer function associated with a system allows us to introduce the following equivalence relation on systems.

Definition 13. *Two systems S_1 and S_2 are said to be* weakly equivalent *iff the following 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 $In(S_1)$ we have*

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

where $\mathcal{F}_{S_1}(x_1)$ and $\mathcal{F}_{S_2}(x_2)$ are the corresponding output flows.

The following result states that the computational power of systems do not increase if we allow several memory tapes.

Theorem 1. *Each multi-tape system is weakly equivalent to some mono-tape system.*

Proof. The complete proof is presented in [**?**]. Here we only present the main idea. We proceed in a manner similar to that used by Hopcroft and Ullman in [**?**] for the proof of the analogous 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 CN equal parts, where N is the upper bound (which can be an infinitely great integer) on the memory size and C is a suitable constant (several sweeps of the memory tape can be required to simulate one step of the multi-system). □

The following theorem exhibits several desirable properties of the family of implementable transfer functions.

Theorem 2. *Up to weak equivalence, the class of implementable transfer functions is stable under abstraction and integration.*

Proof. Let us first prove stability under integration. Let $\mathcal{F}_1 : \mathrm{In}_1^{\mathbb{T}_1^{\mathrm{i}}} \to \mathrm{Out}_2^{\mathbb{T}_2^{\mathrm{o}}}$ and $\mathcal{F}_2 : \mathrm{In}_2^{\mathbb{T}_2^{\mathrm{i}}} \to \mathrm{Out}_2^{\mathbb{T}_2^{\mathrm{o}}}$ be two transfer functions and $\Pi = (\mathrm{In}^{\mathbb{T}^{\mathrm{i}}}, \mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}, \pi^i, \pi^o)$ an interface for $\mathcal{F}_1, \mathcal{F}_2$. Consider a multi-system S defined by the following elements:

- one input channel $\mathrm{In}^{\mathbb{T}^{\mathrm{i}}}$,
- one output channel $\mathrm{Out}^{\mathbb{T}^{\mathrm{o}}}$,
- three memory tapes: two tapes corresponding to each of the memory tapes of S_1 and S_2 and inheriting the same characteristics; a third tape with only one cell with symbol in $\mathrm{In}_1 \times \mathrm{In}_2$,
- the control mechanism defined by the product automaton of the control mechanisms of S_1 and S_2,
- the internal time scale, which is the union of the internal time scales of S_1 and S_2.

System S simulates both memory tapes of S_1 and S_2 with the two first memory tapes and use the memory cell to store at time t the value of $\pi^i(x :: t, \mathcal{F}_1(x_1) :: t, \mathcal{F}_2(x_2) :: t)$.

Let us now prove stability under abstraction. Let $\mathcal{F}_1 : \mathrm{In}_1^{\mathbb{T}_1^{\mathrm{i}}} \to \mathrm{Out}_1^{\mathbb{T}_1^{\mathrm{o}}}$ be an implementable transfer function and let S be a system such that $\mathcal{F}_1 = \mathcal{F}(S)$. Let $\mathcal{F}_2 : \mathrm{In}_2^{\mathbb{T}_2^{\mathrm{i}}} \to \mathrm{Out}_2^{\mathbb{T}_2^{\mathrm{o}}}$ be an abstraction of $\mathcal{F}_1$ and let $f : \mathrm{In}_1 \to \mathrm{In}_2$ be an onto mapping and $g : \mathrm{Out}_1 \to \mathrm{Out}_2$ be any mapping such that (1) holds. Since f is onto, there exists a mapping $h : \mathrm{In}_2 \to \mathrm{In}_1$ such that $\forall x \in \mathrm{In}_2, f(h(x)) = x$. Let $\mathcal{F}_h : \mathrm{In}_2^{\mathbb{T}_2^{\mathrm{i}}} \to \mathrm{In}_1^{\mathbb{T}_1^{\mathrm{i}}}$ and $\mathcal{F}_g : \mathrm{Out}_1^{\mathbb{T}_1^{\mathrm{o}}} \to \mathrm{Out}_2^{\mathbb{T}_2^{\mathrm{o}}}$ be the mappings that apply respectively h and g to any letter of the input flow. Precisely,

$$\mathcal{F}_h((x_t)_{t\in\mathrm{In}_1})(t') = (h(t))_{t\in\mathrm{In}_1} :: t' \ ,$$

and $\mathcal{F}_g$ is defined similarly. Then clearly $\mathcal{F}_h$ and $\mathcal{F}_g$ are implementable. Moreover, $\mathcal{F}_2$ is the composition of the transfer functions $\mathcal{F}_g$, $\mathcal{F}_1$ and $\mathcal{F}_h$. Since these transfer functions are implementable and since composition is a special case of integration the above result implies that $\mathcal{F}_2$ is implementable. □

4 Computable Real Functions

We now consider a special case of implementable transfer functions: the class of *computable functions* on reals.

First, observe that, for $\varepsilon \in {}^*\mathbb{R}$, any non-standard real can be approximated up to ε, by a flow of quanta. Given a time scale $\mathbb{T} \subseteq {}^*\mathbb{R}$, we encode $x \in {}^*\mathbb{R}$ as a flow $\widetilde{x} \in \{+, -, 1\}^{\mathbb{T}}$ with the only non-blanc symbols being a sign '+' or '−' followed by a contiguous sequence of N ones, such that $N\varepsilon \leq |x| < (N+1)\varepsilon$.

Given $\varepsilon \in {}^*\mathbb{R}$, any transfer function $\mathcal{F} : \{+, -, 1\}^{\mathbb{T}} \to \{+, -, 1\}^{\mathbb{T}}$ induces 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, when $\varepsilon \approx 0$, this encoding is injective on $\mathbb{R}$. In the following, we fix $\varepsilon = 1/E$, where $E \in {}^*\mathbb{N}$ is an infinitely great positive integer.

Definition 14 (Computable functions). *A function* $f : \mathbb{IR} \to \mathbb{IR}$ *is* computable *if there exists an implementable transfer function* $\mathcal{F} : \{+,-,1\}^{\mathbb{T}} \to \{+,-,1\}^{\mathbb{T}}$*, such that, for any* $x \in \mathbb{IR}$*, we have* $f(x) \approx \mathcal{F}_\varepsilon(x)$*.*

A function f *is computable with* bounded error*, if there exists an* error bound $B \in {}^*\mathbb{N}$*, such that* $B\varepsilon \approx 0$ *and, for any* $N \in {}^*\mathbb{Z}$*, holds*

$$N\varepsilon \text{ is finite } \implies |\mathcal{F}_\varepsilon(N\varepsilon) - {}^*f(N\varepsilon)| < B\varepsilon \ , \tag{7}$$

where ${}^*f$ *is the non-standard version of* f *(cf. [?]).*

Theorem 3. *The set of bounded error computable functions is closed under addition, multiplication, and integration. Moreover addition and multiplication preserve the order of the error bound.*

Proof. 1. Implementation of addition is straightforward. To implement the multiplication of positive real numbers, $x \approx N\varepsilon$ and $y \approx M\varepsilon$, where N and M are two non-standard integers, recall that $\varepsilon = 1/E$, and therefore $xy \approx (MN/E)\varepsilon$. Observe now that $N, M < E^2$, and therefore their product can always be computed by a system with the internal time scale refining the target one by cE^2, where $c \in \mathbb{N}$ is a standard integer constant. Applying the Euclidean division algorithm, we obtain two non-standard integers Q and R such that $MN = QE+R$, and $0 \le R < E$. Thus,

$$xy \approx \frac{MN}{E}\varepsilon = Q\varepsilon + \frac{R}{E}\varepsilon \approx Q\varepsilon \ .$$

Proof of the error bound order preservation is straightforward.

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

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

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 f. □

Theorem 4. *Let* $f : \mathbb{IR} \to \mathbb{IR}$ *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. For $x \in \mathbb{IR}$ and $\delta \approx 0$, we have $f'(x) \approx ({}^*f(x+\delta) - {}^*f(x))/\delta$. It can be shown that, for $\delta = N\varepsilon$ with an integer $0 < N < E$,

$$\left|\left({}^*f(x+N\varepsilon) - {}^*f(x)\right) - \left(\mathcal{F}_\varepsilon(x+N\varepsilon) - \mathcal{F}_\varepsilon(x)\right)\right| < cB\varepsilon \ ,$$

where c is finite. Consider $N = \lfloor\sqrt{EB}\rfloor$. We then have both $N\varepsilon \approx 0$ and $cB\varepsilon/(N\varepsilon) = cB/N \approx 0$. Therefore $f'(x) \approx (E/N)\cdot\left(\mathcal{F}_\varepsilon(x+N\varepsilon) - \mathcal{F}_\varepsilon(x)\right)$, where the right-hand side is clearly implementable. □

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

Theorem 5. *Let $(a_n)_{n\in\mathbb{N}}$ be a sequence of reals 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.*

Proof. The proof is similar to that of Th. 3. □

5 Dynamical and Hamiltonian Systems

A *dynamical systems* [**?**] is defined, in terms of differential equations, by

$$\dot{x}(t) = f(t, x(t)) \ , \tag{8}$$

with boundary condition $x(t_0) = x_0$, If $f : [0,1] \times \mathbb{R}^n \to \mathbb{R}^n$ is a continuous bounded function, such a system admits a unique solution (*trajectory*).

Theorem 6. *Let S be a dynamical system defined by a bounded continuous function f. If f is computable with bounded error, so is the unique trajectory of S.*

Proof. The proof of this theorem is based on the non-standard proof of Peano theorem (see, for example, [**?**, Secn. A.4.2]), and techniques similar to those applied in the previous section. We omit it here due to space limitations. □

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

$$\begin{cases} \dot{p} = -\frac{\partial H}{\partial q} \\ \dot{q} = \frac{\partial H}{\partial p} \end{cases} \ , \tag{9}$$

with boundary conditions $p(t_0) = p_0$ and $q(t_0) = q_0$, where $H(p, q)$ is the Hamiltonian, modelling the total energy of the system.

Corollary 1. *Let S be a closed Hamiltonian system, such that both $\partial H/\partial p$ and $\partial H/\partial q$ are bounded and continuous. If H is uniformly continuous and computable with bounded error, so is the unique trajectory of S.*

The discussion above allows us to reconsider the Simple pendulum example (Ex. 2), by decomposing it into simpler components.

Example 3 (Simple pendulum: compositional approach). Recall, that the motion of the pendulum, as described in Ex. 2, is modelled by the differential equation (4). Denoting $\varphi = \dot{\theta}$ the angular speed of the pendulum, this equation defines a Hamiltonian system

$$\begin{cases} \dot{\varphi} = -\frac{\partial H}{\partial \theta} \\ \dot{\theta} = \frac{\partial H}{\partial \varphi} \end{cases},$$

with $H(\varphi, \theta) = \frac{\varphi^2}{2} - \frac{g}{L}\cos\theta$, and the initial values $\theta(0) = \theta_0$ and $\varphi(0) = 0$.

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 is connected with two components computing respectively the Hamiltonian $H(\theta, \varphi)$ and the trajectories that it defines. (The component computing $H(\theta, \varphi)$ consists itself of several sub-systems, e.g. to compute the cosine function.) □

6 Conclusion

In this paper, we have presented a new model of timed systems encompassing classical models based on both continuous and discrete time. Along with the discrete values, common in computer science, our model allows to operate with physical quantities such as speed or energy of the system.

Behaviours of such systems are described by implementable transfer functions. The class of implementable transfer functions is stable under abstraction and integration. The associated class of computable real functions is also stable under several fundamental operators and includes the class of analytical functions. As a special case, trajectories of Hamiltonian and dynamical systems are computable under assumption of computability of system parameters. This shows that a large class of physical systems can be modelled in this framework.

Defining the notion of a system as a natural extension of Turing machines, leads 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 — if any — models have so far been developed that combine all the above mentioned characteristics. We believe that our model provides an elegant and powerful mathematical framework for modelling complex industrial systems.

The model we introduce opens numerous research directions, among which one can clearly identify two that have been, to some extent, presented in this paper: the complexity theory analogous to the classical one for Turing machines, and the calculability theory on real numbers. The latter also involves a comparative study with other existing theories, e.g. [**?**, **?**]. Finally, verification techniques have also to be developed for this model.