Hybrid Systems and Real-World Computations

Oded Maler*
VERIMAG
Miniparc ZIRST
38330 Montbonnot, France
Oded.Maler@imag.fr

1 Introduction

In a preface to a book on computational geometry, J.T. Schwartz [Sc87] wrote:

Till now, computer science has been largely combinatorial and symbolic having the manipulation of patterns and tables of data as its principal content. In robotics, however, computer science makes contact with real-world geometric and physical phenomena such as compliance of elastic bodies, the frictional phenomena that occur when two bodies come in contact, errors in modeling which are inevitable in the real world, the sudden changes of state which occur when two bodies collide unexpectedly, and so forth. It is to be expected that much interesting new computer science will emerge from contact with these rich conceptual domains and, in particular, that computer science will become more traditionally mathematical and "continuous".

The theory of hybrid systems (HS), can be considered as an important step toward the realization of this vision. By a hybrid system I mean a system which is a product of the interaction between a discrete state-transition system (an automaton, a program) and a continuous differential dynamical system. In this informal talk I will try to explain what this theory is about and in what aspects it differs from other computational theories. In order to do so I will first (section 2) point out explicitly the underlying assumptions of traditional computability theory concerning the phenomena it is supposed to model. In section 3 I will show how these assumptions are not valid in what I call "real-life" computations, where the computer is embedded in the physical world. In section 4 I will show how reactive and real-time systems can be considered as preliminary steps toward hybrid systems.

Finally by mentioning some trends and developments in computer science, AI, robotics and cognitive science, I hope to put HS in a broader perspective and show you that the transition to HS might be even more significant than you think. Since, unlike other participants, I did not have any astonishing new results on HS since the announcement of this workshop[2] the talk will be completely non-technical, and the uninterested ones among the audience can find comfort in other presentation in this workshop.

---

* This talk was presented in the first workshop on hybrid systems, Lyngby, Denmark, 1992.
[2] Later I had something, see [MP93].,

2 Computations

Before digital computers came into existence, the only real instance of the phenomenon of computation was some type of human activity. And it must be said that this activity is only the tip of the iceberg of the operation of the nervous system – and we will come back to this point when we discuss AI and robotics. For the purpose of this discussion the computation

$$12 + 13 = 25 \tag{1}$$

or the deduction

$$(\forall x, y) husband(x, y) \Rightarrow wife(y, x), husband(aa, bb) \vdash wife(bb, aa) \tag{2}$$

are symbol manipulations of the same sort. The basic assumptions in this process of computation are:

1. The input and the output are symbols, formal entities, and the "computer" can access them directly and accurately.
2. Even if these symbols denote something in the "real" world, such as 12 apples or a certain married couple (and this is presumably our motivation to perform the computation) still we behave as if the real-world has frozen since the encoding of the input. Thus if the encoding was faithful and the computations were correct (in whatever sense) then the results can be decoded and stand for something true and still relevant about the real world.
   In other words, we ignore the possibility that by the time our computations terminate, aa and bb might be divorced and most of the apples turned into Calvados.

Still another way to see it is by declaring that by investigating computability in the classical sense we are interested in computations that are:

- Not related to the real-world.[3]
- Related to the stationary or the slowly-changing features of the world, e.g., calculate the number of angels that can stand on a pin-head (assuming it is a universal constant).
- Performed off-line, that is, it is assumed that the computation is taking place either before or after the process it is supposed to model. The computation is *isolated* from the processes of acquiring the input from the world and returning the output to it.

The mathematical models of computability theory are indeed tailored for these phenomena: if we look at a typical Turing machine, what we see is a sequence of dead symbols on the input tape, additional symbols that denote the transition function and an abstract head working in a cycle of discrete steps of read-interpret-move-write. The theory is not concerned with any possible change in the input symbols (or in what they stand for) *after* the computation has been started. It also treats the

[3] Which some of you will admit in private.

operations of symbol reading or writing and head movement as the atomic actions which can be performed in a perfect manner.

And this is almost what traditional computability theory (and its derived complexity theory) is all about. An interesting and fascinating phenomenon, yet a very *marginal* one compared to the whole spectrum of natural phenomena. Even if we ignore physics, chemistry and most of biology and restrict our domain of interest to the *information-processing* activities performed by the central nervous systems (CNS) of humans, still "classical" off-line computation constitutes a negligible fraction.

3 Motor Control and Robotics

How different is the activity I have just discussed from what most of my brain is doing right now when I am talking to you (or typing this abstract). Consider just a simple action such as reaching some point with the tip of a finger: the CNS receives visual signals projected on the retina, as well as measurements from receptors attached to muscles and tendons. Based on those it "computes" and outputs certain nerve pulses that activate certain muscle groups and cause their contraction. All this is done in a manner which is completely embedded in the real-world: as the hand moves, the sensory "input" to the computation changes. On the other hand, the same "output" signal going from a motor-neuron to a muscle can have a completely different external effect if transmitted after some delay, or in another context.

This is most of what the CNS does, controlling in the loop of receptors and muscles, in order to facilitate our survival in the real world by moving toward pleasant and useful locations, running away from unpleasant ones, maintaining our posture and stability in the presence of gravitation and other disturbances. Nobody yet really understands how this is done – how by employing such huge ensemble of noisy sensors and effectors we achieve these magical capabilities to move our hand to a certain point and grasp an object. One can appreciate the complications by trying to build robots that mimic these capabilities: take several joints and motors, add some sensors for force, touch, stretch, light etc., and build a control program that continuously reads these sensors and commands the motors. What can be said about the behavior of such a program? In what language can it be expressed? How and under what assumptions could these properties be verified? What kind of computerized tools can be used in order to improve the development process of such artifacts? Such questions, at least to me personally, constitute the primary motivation for a theory of HS.

4 From Static to Dynamic Environments

I have described two extreme types of computational phenomena, the traditional "autistic" computation and the one embedded in the real-world. The shift from the former to the latter is not sharp but gradual and can be done along independent dimensions. I'll briefly summarize the way I view the main steps.

4.1 Reactive Systems

Harel and Pnueli in their often-quoted (and seldom read) paper [HP85] distinguished *transformational* systems (a static environment) from *reactive* ones. Intuitively, reactivity is intended to capture the phenomenon of an *ongoing interaction* between the environment and the computer. One may legitimately ask whether there is something new in this notion – why cannot we reduce a reactive computation into a classical transformational computation where all the actions of the environment and the computer are written on the input and output tapes respectively? Where does the reactivity hide in this model? The answer lies in the sequential interpretation of the input representation and its association with the *direction of time*. While general recursion theory can be viewed as dealing with functions from X^* to Y^*, the corresponding strings are considered as encodings of some countable domains, and *bear no temporal meaning*. Time does not exist during the process of input reading which can be completed before the computation starts.

In contrast, (deterministic) reactive machines define *monotone*[4] functions from $(X^*, \preceq)$ to $(Y^*, \preceq)$, where $\preceq$ is the prefix partial-order relation. Again, we can assign various meanings to $\preceq$ such as inclusion of intervals if we consider positional encoding of numbers, but the crucial step is by unifying $\preceq$ with the temporal order of occurrence of events. From computability point of view, what has been introduced is the notion of progressive (in time) knowledge concerning the input as well as progressive production of output. For a deterministic reactive machine M, we can associate a monotone function $f : X^* \rightarrow Y^*$, such that for every $x \in X^*$, $f(x)$ is the set of output actions performed by M after reading x. We can then view a reactive computation of M on input $x = x_1 x_2, \ldots, x_n \in X^*$ as an element of $(X \cup Y)^*$ defined by:

$$C_f(x) = x_1 \cdot f(x_1) \cdot x_2 \cdot (f(x_1)/f(x_1 x_2)) \ldots x_n \cdot (f(x_1 \ldots x_{n-1})/f(x_1 \ldots x_n))$$

where $/$ is defined by $x/xx' = x'$. Take another machine M' having a function f' such that $f(x) = f'(x)$, still $C_f(x) \neq C_{f'}(x)$ if for some $x' \prec x$, $f(x') \neq f'(x')$. In other words, two reactive computations on the same input, even if they finally produce the same output, are considered different if the interleaving of inputs and outputs are not the same.[5]

By making this finer distinction between performing an action *before* or *after* the occurrence of an environmental event we make our first contact with the sad (but apparently inevitable) facts of life, the irreversibility of Time and all that.

Note that in this formulation of reactivity we consider the most general environment, in the sense that it can generate all possible sequences of events. On the other hand, it is a rather limited class of "obedient" environments which generate

[4] And continuous or even Lipschitz, if you want.

[5] In fact, the technical story is a bit more complicated, because if X itself is infinite than it should be encoded by some D^*, so the real model is of a two-dimensional tape, one dimension for the encoding of the domain (this dimension is neutral with respect to time) and one to denote the order of events, with respect to which we require monotonicity. In this case, a non-reactive system is a reactive system for which the second dimension is trivial.

their events only when the reactive machine is ready to process them. This seems in contrast with our real world: in terms of content, the real world is much more restricted – it obeys some regularities and not every wild sequence of events may happen. On the other hand, the world is much less patient concerning timing and scheduling. So proving something with respect to this environmental model amounts to worst-case analysis with respect to content and best-case analysis with respect to timing. But our particular world is not the best neither the worst of all possible ones.

4.2 Time is Money

Another independent departure from the autistic undisturbed view of computation is by considering the time it takes to perform a computation. When it is measured as the number of abstract computation steps, we can ask whether their number is finite or infinite, how it grows asymptotically with the size of the input, and all those fascinating questions which we leave for the best minds of our decade to struggle with.

Less interesting theoretically, but sometimes practically more important is the question of "real" time, how much it will take from the onset of the computation till the output is produced. The answer depends on the specific program and the quantitative time we assign to each operation of the underlying hardware. Thus, if we replace our old Turing machine by a new one having 100 times faster head-movement speed, we still compute *the same* function although we may get the result faster. In the context of a static environment, this has no theoretical significance. Questions of speed become meaningful only when the computation time goes beyond the range in which the environment can be approximated as static. Take, for example, a payroll program: whether it takes one hour or two to run it is rather meaningless; if however its computation time is in the scale of weeks, than it can be viewed as a real-time application.

This also explains how people dealing with Control (the area where real-life "computations" are currently practiced) treat computational phenomena. Based on their analysis of the dynamics of the environment, they can assume, that the change occurring within intervals smaller than, say, 5 milliseconds, are negligible. Thus, a computations that can be carried out within such an interval can be viewed as a good-old autistic transformational computation ignoring the dynamics of the environment. And my feeling is, that this is all they want from computers and computer scientists: "just supply us with a black box computing some f within t seconds, and don't interfere with our mathematics" – but maybe we will have to.

4.3 Combining Reactivity with Real-time

Recall that a specification is a set of equations on the input and output. An acceptable implementation in a given environment is a machine such that all the computations that may result from its interaction with the environment are solutions of those equations. In the case of reactive systems we can use temporal equations[6] on

[6] Or any other esoteric formalism.

$(X+Y)^*$ (or $(X+Y)^\omega$) which describe the acceptable interleavings of input-output events. In order to verify that an implementation satisfies the requirements we must use additional assumptions (e.g., synchrony hypothesis) which "guarantee" that the actual interleaving will be exactly as the reactive machine intends, that is, the machine is much faster than the environment and can generate as many output events as it wishes between any two input events.

All these consideration deal with qualitative time – the relative order of occurrences of events in the machine and in the external world. A more detailed analysis can be performed by using the very powerful and useful illusion of absolute time. Instead of comparing the machine with the environment, we measure each of them against some universal clock, and then can say that a computation step takes between l and u clock ticks or that environmental events are separated from each other by at least d clock events. This way, the "real" time can serve as a "common currency" for a more detailed and realistic analysis of the behavior of the implementation. Using this metric time is sometimes more practical because you are more likely to find a machine whose speed is defined in "absolute" seconds, rather than find one which is claimed to produce its output between the arrival of customer n and the arrival of customer $n+1$ for every n and for every request-generating process.

At the requirement level, by using real-time, we can also replace the messianic style of specification ("eventually") with a more concrete one ("2 weeks after the deadline"), and as I said before, by specifying this type of bounded response properties, we express implicitly some further assumptions concerning the dynamics of the environment – in this case, the patience threshold of proceedings editors.

To summarize, by augmenting reactivity with real-time we have a more detailed picture of the comparative dynamics between the implemented machine and its environment.

4.4 Hybrid Systems

Finally, when we come to HS, we try to incorporate as much of the environment as possible into our models. To this end, the environment is not considered anymore as a (time-constrained) generator of discrete events, but rather as *continuously changing* and as obeying some more-or-less specified rules such as differential equations. The previous robotics example when viewed as a hybrid system consisting of components that model the arm dynamics (conditional differential equations), the control program (a timed transition system), and interaction between the two via sensors and actuators. Discrete transitions can now be triggered by continuous variables that path certain thresholds, and those transitions, in turn, can influence the dynamics of the continuous component. For such a hybrid system we can try to predict the properties of the possible behaviors, e.g., "every object will be reached by the tip of the arm within t seconds".

This completes the passage from a static environment, via an environment that makes discrete transitions, down to a continuously-changing one. On the other hand, by considering specific classes of evolution rules for the environment (e.g., observed properties of rigid objects obeying Newtonian mechanics), the class of possible sequences of events (predicates on the continuous variables) would be much more restricted than in the "most general" environment.

Exercise to the reader: why mathematical control theory (e.g., [So90]) is not sufficient? In other words, why can't we have only a continuous model plus a "non-constructive" control function augmented with some considerations of delays and sampling?

5 Relation to other Trends

I will conclude this presentation by mentioning some other contemporary scientific developments which one might (possibly but not necessarily) see as relevant to the discussion.

5.1 On-line algorithms

A currently active sub-field of algorithmic complexity theory is dealing with what is called dynamic and on-line algorithms. In these models, initially motivated by problems in operating systems such as optimal caching policies, the algorithm is required to solve an optimization problem in a changing environment or with a progressive knowledge of the input. For example, consider a traveling salesman in an Euclidean space who always knows only k cities in advance. The performance of such algorithms is compared to the optimal solution when all the information is given in advance (see for example [BLS92]). This sort of research can be seen as a preliminary step toward the extension of complexity theory along the reactivity dimension.

5.2 Qualitative physics

This is a sub-field of AI (see [WK89]) concerned with building models of complex devices (electrical circuits, refrigerators) which can be used for predicting their behaviors, diagnosing their faults, etc. The ultimate goal of this field is to mechanize the reasoning process of engineers and other experts. One of the approaches in this field is to model the devices by means of qualitative differential equation over discrete domains, such as $\{-, 0, +\}$, and try to predict their possible behaviors by simulation. This approach has been recently criticized (see [DS92] together with a long list of commentaries) for being too simplistic and ignorant of the wealth of existing mathematical techniques dealing with qualitative analysis of continuous dynamical systems (e.g., [HS62]). Such criticism might as well be applicable in the future to the theory of hybrid systems...

5.3 Discrete-event control

This development (e.g., [RW87]) is more anecdotal. It is a reformulation by control theorists of some aspects of the theory of discrete reactive systems using control terminology. Thus the environment is a *plant* and the program is a *supervisor* that can prevent some events from occurring in the plant. All the problems of reachability analysis, program synthesis, etc., are treated in the control framework of observability, controllability and optimal controller synthesis. Although initiated by

a prominent control theorist, this theory treats discrete event systems in a complete isolation from continuous processes, and thus, it cannot contribute much to the theory of HS.

5.4 Foundations of AI

AI is concerned with building models and imitations of intelligent behavior. For many years it has concentrated on the higher-level cognitive capabilities of humans such as problem-solving, chess-playing, abstract planning and natural language processing. The ultimate goal was to build a machine that could pass the Turing test, that is, will be able to communicate through a terminal without being distinguishable from a human being. During the last decade this "symbolic" approach to AI has been attacked from several directions.

One of the well-known arguments (and an endless source of diverging debates) was put forward by Searle [Se80] and is known as the "Chinese room" argument. It claims that it is possible to pass the Turing test without "understanding" and without being "really intelligent". The proof is by putting me (or you, or any non Chinese-speaking individual), inside a room together with a huge table which essentially is an effective description of a function from C^* to C^* where C is the Chinese alphabet. Suppose that using these instructions I can successfully communicate with Chinese-speaking persons – does this imply that I *understand* Chinese?

Without opening this can of worms, I'll just mention one analysis of this situation given by Harnad [Ha90] where he accepts Searle's argument for the case where the communication is entirely symbolic – Chinese characters in and Chinese characters out. In this case the symbols I read and write bear no meaning at all.[7] On the other hand, Harnad claims, Searle's argument fails if the machine passes the total Turing test (TTT) in which the interaction is not restricted to teletype communication but extended to full robotic capacities, and in that case, the symbols will be "grounded" on sensory-motoric experience from which they will take their meaning.

I mention these arguments not in order to motivate anyone in the audience to solve these open problems of mind and meaning, but rather to indicate another motivation for breaking the walls of the Turing ghetto, and let machines (and their underlying theories) interact with the external world in a non-symbolic manner. Similar arguments, formulated against the symbolic dogma of cognitive psychology have been raised by Hofstadter [Ho82] under the title: "waking up from the boolean dream". Within the connectionist movement Smolensky [Sm88] argued in favor of studying the manner in which "symbolic" events in the brain are realized by sub-symbolic activities of a complex neural-like dynamical systems. It is worth noting that technically, as is the case with HS, some branches of connectionism also require a mixture of continuous and discrete mathematical techniques.

5.5 Behavior-based robotics

Last but not least, let me mention another trend in AI which goes together with the abovementioned ones. This is the field of behavior-based robotics led by Brooks at

[7] We will not consider the question whether or not my words bear any meaning when I express myself in English...

MIT. In contrast with traditional symbolic AI which, as I said before, concentrated on abstract high-level activities, this approach suggests building intelligence from the bottom-up, and takes low-level creatures such as insects as the normative model for intelligent machine. The underlying ideology (the most recent version is [Br91]) is based on an observation, similar to the one I described at the beginning, that abstract reasoning is marginal with respect to the real-life "situated" computations involved in motor control, many of which we share with our evolutionary ancestors.

Unlike traditional AI approaches to robotics, which were based on a cycle of sensory reading, interpretation into a symbolic structure, reasoning and computing on this structure and then acting, Brooks suggested an alternative architecture ("subsumption", [Br86]) based on an asynchronous network of interacting automata connected to sensors, actuators and timers. Analyzing the behavior of systems based on such architectures is, I think, a real challenge for theoreticians of concurrent, reactive, and eventually hybrid systems. Even when we come back after this workshop to our dry formal systems, proof rules, and the rest of our particular tricks-of-the-trade, we should keep in mind some more "live" motivations for all this Hybrid System enterprise, and to my mind, being able to reason about the behavior of such robots is a good one.

6 Conclusion

Hybrid systems research might lead to interesting developments in computer science and in mathematics. It might as well be useful.

Acknowledgement

I would like to thank A. Pnueli and J. Sifakis for useful comments.

References

[Br86] R.A. Brooks, A Robust Layered Control System for Mobile Robots, *IEEE Journal of Robotics and Automation* RA-2, 14-23, 1986.

[Br91] R.A. Brooks, Intelligence without Reason, *Proc. IJCAI-91*, Sydney, 1991.

[BLS92] A. Borodin, N. Linial, and M.E. Saks, An Optimal On-line Algorithm for Metrical Task System, *J. of the ACM* 39, 745-763, 1992.

[HP85] D. Harel and A. Pnueli, On the Development of Reactive Systems, in K.R. Apt (Ed.), *Logics and Models of Concurrent Systems*, 477-498, Springer, Berlin, 1985.

[Ha90] S. Harnad, The Symbol Grounding Problem, *Physica D*, 42, 335-346, 1990.

[HS74] M.W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra*, Academic Press, 1974.

[Ho82] D.E. Hofstadter, Waking up From the Boolean Dream, or, Subcognition as Computation, *Scientific American*, 1982. (Reprinted in *Metamagical Themas.*

[SD92] E.P. Sacks and J. Doyle, Prolegomena to any Future Qualitative Physics, *Computational Intelligence* 8, 187-209, 1992.

[Sc87] J.T. Schwartz, Preface, in J.T. Schwartz and C.K. Yap (Eds.), *Algorithmic and Geometric Aspects of Robotics*, Lawrence Erlbaum, Hillsdale, NJ, 1987.

[Se80] J.R. Searle, Minds Brains and Programs, *Behavioral and Brain Sciences* 3, 417-424, 1980.

[Sm88] P. Smolensky, On the Proper Treatment of Connectionism, *Behavioral and Brain Sciences* 11, 1-22, 1988.

[So90] E.D. Sontag, *Mathematical Control Theory - Deterministic Finite Dimensional Systems*, Springer, Berlin, 1990.

[RW87] P.J. Ramadge and W.M. Wonham, Supervisory Control of a Class of Discrete Event Processes, *SIAM J. of Control and Optimization* 25, 206-230, 1987.

[WK90] D. Weld and J. de Kleer (Eds.), *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufman, San Mateo, 1990.