Control systems vs reactive systems

Michel Sintzoff

Department of Computing Science and Engineering
Université catholique de Louvain

(Extended Abstract)

The present paper is made of reading notes. They report on our belated learning of classical approaches for designing control systems and reactive ones. These approaches concern dynamics, games, and their analysis and synthesis. They appear to provide a common and attractive setting. We end up by discussing a number of challenges in this area.

1 Dynamics

Control systems have been actively developed and investigated since the beginning of the industrial age. Remember Watt's flyball governor in steam engines, or even the water controller of Ktesibios' clock, twenty-three centuries ago in Alexandria. This field dramatically expanded during the last century, because of the impressive progress in systems for transportation, production and alas destruction. The recurring challenge is to guarantee the safety of systems while ensuring optimal use of resources.

In parallel, during the second half of last century, computing science and software engineering progressively evolved from a sequential, batch mode of thinking towards an interactive, cooperative and reactive view. A convergence with the control viewpoint was thus only natural: control systems are more and more implemented on computers, whereas computer systems interacting with the physical environment amount to digital controllers.

This common ground is known nowadays under the term of ”hybrid systems”. As a matter of fact, the standard structure of these hybrid systems still remains rather historical and accidental: the physical environment is described by continuous-time components whereas the controller is modelled by discrete-time ones. In this sense, the adjective ”hybrid” is pertinent. However, it suggests a basic difference between continuous and discrete time. We never really liked this.

To our mind, the reason of the convergence between control systems and reactive computing systems lies much deeper than in the mere use of computers to implement control. The essence of controlled as well as controlling components is given by their dynamics, be it in discrete or in continuous time. Hence, the relevant scientific ground on which to base the common development of control and reactive systems is the mathematics of dynamics. Controllers can then very well use continuous time, as typical in classical control engineering. Conversely, a controlled physical environment can rightly be modelled by a discrete-time subsystem, as done in computer-based simulation. Remember the latter field gave rise to object-oriented programming, cf. the Simula 67 language designed by O. Dahl and K. Nygaard.

Two observations illustrate our standpoint. Firstly, the concept of invariance is fundamental for the logical mastery of programs, including reactive ones. It has also been a standard technique in the qualitative analysis of dynamical systems, at least since H. Poincaré. Secondly, the concept of variance allows to ensure termination in programs; cf. Floyd’s well-founded decreasing functions. This concept boils down to a discretization of the classical concept of stability, used to guarantee asymptotic convergence of continous-time dynamics; cf. Lyapunov’s decreasing functions.

The pervasive, fundamental rôle of both invariance and variance for the qualitative analysis of complex dynamics has been a happy personal surprise, ten years ago. We do not tie computing science to discrete-time dynamics anymore. We rather consider discrete time as an abstraction of the continuous one. Admittedly, this is just a belated rediscovery of the classical analysis method of nonlinear dynamics by symbolic ones; see e.g. the books by S. Wiggins.

We thus deem the proper scientific context for the cooperation between control engineers and software engineers is the field of dynamical systems, rather than specific classes of differential equations or formal specifications. In fact, this viewpoint begins to appear in the context of hybrid systems; cf. a special issue of the Proceedings of the IEEE, July 2000. In the classical theory, the choice between discrete and continuous time is a choice between topologies, and is thus entirely free; see for instance the book by E. Akin.

2 Games

Control systems, as well as reactive or hybrid ones, involve the interplay between a controlled component and a controlling one. This corresponds to a game between an Opponent and a Proponent, respectively. Game theory is the classical theory of interaction between dynamics. Its central importance for control and reactive systems should thus not come as a surprise.

This view may be simple minded, but it helps. Interestingly enough, it has been rediscovered time and again. Our turn was a couple of years ago: quite late, indeed. It is sobering to revisit Zermelo's analyses of two games, in the beginning of last century: the discrete-time game of chess, and the continuous-time game of a tired swimmer against a wild river in a narrow canyon with an accessible island downstream. Actually, Zermelo already used a form of backwards inductive reasoning in that work.

In games, the dynamics can use discrete or continuous time, plays can be finite or infinite, goals can be qualitative or quantitative, and Proponent's moves react to Opponent's moves in sequence or in parallel.

In the discrete-time case, each move is a unit-time state transition. Finite discrete-time games were actively studied in economics. They are modelled using discrete mathematics, including induction principles, as elaborated e.g. by Zermelo, Morgenstern and von Neumann; see for instance the book by P. Morris. Infinite discrete-time games have been investigated mainly in logic and computing science, using the ω-automata introduced by Büchi forty years ago, and Knaster-Tarski's theorem on fixed points; see e.g. the book edited by E. Grädel, W. Thomas and Th. Wilke.

In finite and infinite continuous-time games, as pioneered by Isaacs fifty years ago, moves are infinitesimal and are typically defined by differential equations. Integration can then be seen as a continuous-time counterpart of induction. Yet, dynamics may very well use discrete time, which brings us back to discrete-time games. In this case, Isaacs even spoke of ”discrete differential games” to suggest the freedom in choosing the temporal basis.

Thus, once more, the distinction between discrete and continuous time appears to be secondary. Its premature introduction obscures understanding. Let us then call ”dynamical games” all games, with discrete or continuous time and with finite or infinite plays. In each case, we observe the same essential associations between minimization, universal quantification and Opponent moves, and between maximization, existential quantification and Proponent moves.

3 Optimal and winning strategies

Surely, we long thought, the distinction between discrete and continuous time remains essential for analyzing dynamical games in depth and for synthezing winning or optimal strategies. Again, to our recent wonderment, this proves wrong as well.

We first discuss methods concerning classical dynamics, i.e. non-interactive, single-agent ones as in solitary games. The optimization of continuous-time dynamics was initially tackled by the classical calculus of variations. In modern approaches, optimization problems for continuous-time dynamics are expressed by Hamilton-Jacobi differential equations. Solving such nonlinear systems is hard but not impossible, e.g. by using Pontryagin's principle of optimality.

Consider then discrete-time dynamics, as defined by algorithms, automata, or difference equations. In this case, the fundamental method for ensuring optimality is Bellman's principle of dynamic programming. This essentially amounts to an inductive, economical search throughout the solution space, but in general entails an exponential cost of computation.

Now, let us discretize the Hamilton-Jacobi equations defining a continuous-time optimization problem, and let us similarly discretize the method for solving these equations. The resulting discrete method of solution basically reveals itself as the inductive scheme of dynamic programming! This is why definitions of dynamical optimization problems, for discrete or continuous time, are sometimes presented as Hamilton-Jacobi-Bellman equations. This remarkable correspondence is developed e.g. in the books by A.A. Agrachev and Yu.L. Sachkov, by M. Bardi and I. Capuzzo-Dolcetta, by D.P. Bertsekas and by R. Vinter. Actually, the principle of dynamic programming holds for continuous-time dynamics as well. As a consequence, the trajectories which satisfy Pontryagin's principle of optimality provide characteristic curves for the Hamilton-Jacobi equation associated with Bellman's principle of dynamic programming. It is thus misleading to restrict the latter to discrete-time dynamics. After all, the concept of shortest path is independent of the time basis.

These observations can be transferred, lock, stock and barrel, to genuine games, viz. those with at least two adversaries. Note qualitative, winning strategies amount to simplified versions of quantitative, optimal strategies: the set of quantities is then reduced to a binary set of qualities. W.l.o.g., we assume optimality means maximality.

The maximization of the solitary player now becomes a logical interplay between the Proponent's maximization and its dual, the Opponent's minimization. The resulting maximin versions of the Hamilton-Jacobi equations, introduced by

Isaacs, are known as the Hamilton-Jacobi(-Bellman)-Isaacs equations. Accordingly, the maximin version of dynamic programming for games could be called the Bellman-Isaacs principle. In the case of infinite discrete-time games, a fundamental method for synthesizing winning strategies was discovered by Büchi and Landweber in the late sixties.

The main results on solving dynamical games, in continuous or discrete time, are presented e.g. in the books by T. Başar and G.J. Olsder and by J. Lewin, and in the book by E. Grädel et al. mentioned above. In the theories of dynamics and games, the methods for analysis and synthesis have been developed very much hand in hand. Specialized, effective techniques have been developed for control and hybrid systems. They mainly concern linear and finite-state dynamics, respectively.

4 Challenges

We should not imagine all is well. Not all problems are basically solved. Far from it.

A first challenge is to scale up. Current theories provide tools to tackle simple dynamical structures and properties, but not much more. On the other hand, an interactive distributed system amounts to a game with many players, teams, goals, rôles and levels. An example of this challenge is the design of a computer-based system for organizing road traffic and for assisting in the driving of vehicles, in order to maximize the survival of persons, the satisfaction of transportation needs, and the preservation of the environment. This can be seen as a land-based version of air-transportation control systems. We realize only too well the problems this raises.

It seems the successful approaches developed in software engineering are relevant for systems engineering at large. We know these approaches include hierarchical decomposition or composition, successive refinement or abstraction, and conjunction of viewpoints. But we also know the enormous difficulty in the effective implementation of good principles.

A second challenge is to establish a common, well-thought theoretical basis. Fundamental concepts in the analysis or synthesis of dynamics and games may well be shared. Yet, the actual techniques resulting from the historical developments vary too much. A crystal-clear unification of the existing methods is sorely needed.

A third challenge is the progressive elaboration of effective techniques for scaling up. For instance, the basic properties of success and optimality should be refined into specialized qualitative and quantitative constraints, including security and probability aspects. Concepts such as equilibria in multi-agent games should be adapted accordingly. Design processes should thus integrate negotiation techniques.

A fourth challenge is the adaptation or the development of systems for design support. Helpful support-systems now exist for the precision engineering of software. Is it feasible to modify these in order to tackle dynamics and interaction on a par? Should we rather start again from scratch, or integrate control-system design with software design?

The last but not least challenge is the effective and harmonious cooperation between computing scientists and applied mathematicians, as well as between software engineers and systems engineers from other disciplines. Software engineers and computer scientists maybe should go more than halfway in that direction. After all, computing science and engineering is the younger field and should genuinely value the other ones.