Nonlinear Systems, Automata, and Agents: Managing their Symbolic Data Using Light Weight Persistent Object Managers*

R. L. Grossman†
Laboratory for Advanced Computing
University of Illinois at Chicago

W. Kohn
Intermetrics

A. Nerode‡
Mathematical Sciences Institute
Cornell University

November, 1994

Abstract

A hybrid system is a network of continuous input-output systems and discrete digital automata. Since hybrid systems contain both continuous and discrete data, implementations must manage both types of data. It is currently a challenge to manage the discrete data of a hybrid system. One approach discussed in this paper is to use "light-weight" persistent object managers and specialized algorithms exploiting them.

This paper also considers external agents whose role is to introduce new digital automata into a hybrid system when the hybrid system is no longer meeting the desired performance specification. One means of implementing external agents is to inference from a set of variables, predicates and rules using MetaProlog. It has been an open problem as to the best way to manage the predicates of the agent programs to speed up computation of control automata. An alternative discussed here is to use a light-weight persistent object manager to manage a persistent object store containing variables, predicates, and rules and specialized software tools to perform the inferencing.

1 Introduction

Hybrid systems are interacting networks of continuous, generally nonlinear, input-output systems and discrete automata. In a typical example, a physical process or plant gives rise to a continuous input-

*For more information, contact Robert L. Grossman, Laboratory for Advanced Computing, University of Illinois at Chicago, 851 South Morgan Street, Chicago, IL 60607, USA, +1 312 413 2176; +1 312 996 1491 fax, grossman@uic.edu.

†This research was supported in part by NASA grant NAG2-513, DOE grant DE-FG02-92ER25133, and NSF grants IRI 9224605 and CDA 9303433.

‡Sponsored in part by Army Research Office contract DAAL03-91-C-0027, DARPA-US ARMY AMCCOM (Picatinny Arsenal, N. J.) contract DAAA21-92-C-0013 to ORA Corp., SDIO contract DAAH04-93-C-O113, NSF grant DMS 93-06427311.

output system. A digital program, viewed as a digital automaton, enforces some desired behavior of the plant by selecting appropriate plant laws and appropriate control laws for the actuators which are used to control the plant. Hybrid systems have emerged as a powerful methodology for modeling large complex systems [2].

A fundamental problem in hybrid systems is the design of digital automata which can select appropriate plant models or modes and appropriate control laws to achieve desired performance. The problem is how to find algorithms which, applied to ordinary differential equations describing the dynamics of a plant and specifications for the desired plant performance, extract digital control programs that force the state trajectories of the system to obey the performance specifications. See Figure 1.

A second fundamental problem is how to do this in a robust fashion. A simple hybrid system consists of one digital control automaton used to control one continuous system. In use, the digital control automaton may fail to meet performance requirements for control of continuous system state. It may need to be replaced on-line when deviations from performance specifications are detected. Kohn and Nerode introduced the concept of an agent to accomplish this task. An agent monitors the simple hybrid system consisting of its continuous system with system controller (actuator) and digital control automaton. The agent looks for non-compliance with specifications and uses this and other historical information to compute, and occasionally install, a new finite control automaton. To repeat, the purpose of introducing the concept of agent is to implement appropriate adaptive control in the face of unmodelled dynamics and other uncertainties by computing and installing a new finite control automaton to replace the old one on the fly, when performance specifications for the system are violated. See Figure 2.

In this paper, we review the concept of a hybrid system and of an external agent. A particularly important challenge is developing appropriate technology to manage the symbolic data required by an automaton and an external agent, for large real time or near real time applications. Using "light-weight" low overhead, high performance persistent object managers to manage this data is a promising approach [1] and [3]. In this paper, we discuss some of the implications of this approach.

2 Hybrid Systems

In this and the following section, we introduce our basic ideas with a simple example. We work formally. That is, we are not concerned with smoothness of vector fields, controls, or trajectories. We assume that all continuous structures have enough smoothness for our constructions. Our example has three components: an input-output system, an automaton, and an agent. This yields a hybrid system consisting of the first two components. This hybrid system is is sole our concern in this section. In the following section, we extend consideration to a hybrid system with an external agent.

Input-Output System. The role of the input-output system is to describe the time evolution of a continuous physical process which can be controlled. More precisely, given a control $t \mapsto u(t)$, the time evolution of the state is given by a map $t \mapsto x(t, u(t))$ which satisfies an ordinary differential equation

$$\dot{x}(t) = F_\gamma(x(t), u(t)),$$

where F is a vector field.

Automaton. The role of the automaton is to describe the time evolution of a discrete process, with discrete input and output symbols. A digital program is an example of such an automaton. More precisely, we assume that the automaton has a space of states S, accepts input symbols β, and that its dynamics are specified by a transition map

$$\Omega^* \times S \longrightarrow S,$$

where the Ω^* is the semigroup of all words generated by the input symbols β. After each state transition, the automaton outputs a discrete symbol γ.

Hybrid Systems. A hybrid system consists of one or more input-output systems, interacting with one or more automata.

Example 1. *Mode Shifting.* As a simple example, we return to the setting of the opening paragraph. We follow [6] and consider a simple hybrid system in which the digital automaton selects appropriate control laws and shifts modes of the input-output system. That is, assume that for each discrete output symbol γ of the automaton, there is a triple consisting of

1. a time τ_γ
2. a control $t \mapsto u_\gamma(t)$
3. a vector field $F_\gamma(x(t), u(t))$

This data determines a flow in the state space. That is, if the nonlinear input-output system was at the point x_0 in the state space, then this data uniquely specifies a new point x_1 in the phase space defined by flowing along the vector field F_γ for time τ_γ:

$$x_1 = \exp \tau_\gamma F_\gamma \cdot x_0,$$

where $F_\gamma = F(x(t), u_\gamma(t))$.

The flow of the hybrid system is defined by repeating this cycle: the automaton accepts another discrete input symbol, transits states, and outputs a new discrete output γ', which in turn is used to select a new closed loop mode.

The intended interpretation is easy: the automaton, viewed as a discrete digital process, is used to select a mode and a closed loop control for a continuous input-output system. See Figure 1.

Since a hybrid system is a network of communicating continuous and digital devices, at any time the system has a hybrid state, the simultaneous state of all plants and digital control devices. The hybrid state may be viewed from two complementary viewpoints, each of value for different purposes. The first viewpoint is that of the continuous world. From the continuous point of view, the states of the continuous plants naturally evolve in continuous time. We may also think of the states, inputs, and outputs of the digital device as piecewise constant functions of continuous time, changing abruptly at the endpoints of open time intervals. Thus the hybrid state can be viewed as evolving in a suitable continuous space of hybrid states. The second viewpoint is that of the discrete world. From the discrete point of view, the trajectories of the plant may *themselves* be viewed as discrete objects, which can be interpreted as the states of an appropriate discrete automaton. There is a natural discrete automaton with the same behavior as the network consisting of the automaton summarizing the continuous plant behavior and the digital controlling automata. Part of the charm of the subject arises from the interplay between the discrete and continuous viewpoints.

3 External Agents

We now describe a simple type of agent and its interactions with hybrid systems.

Agents. An agent is specified by a collections of variables, constants $\xi = \xi_1, \xi_2, \ldots,$, function symbols, and functional terms defined from them, predicates

$$q(\xi_1, \xi_2, \ldots,),$$

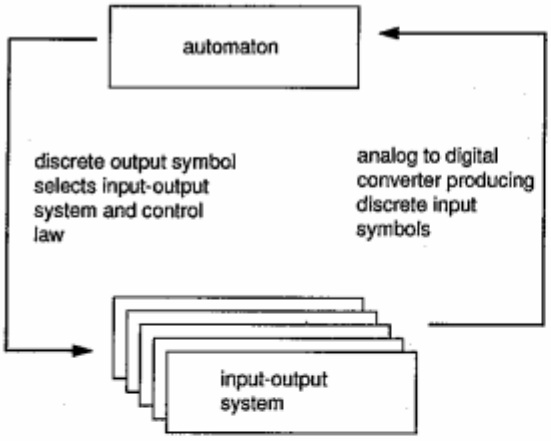

Figure 1: A simple hybrid system in which the discrete digital automaton selects an appropriate input-output system and control law.

and rules

$$p \ :- \ q_1, q_2, \ldots, q_k,$$

where the symbol $:-$ denotes implication the comma denotes logical conjunction, and p is also a predicate.

Example 2. *Automata Shifting through Variable Update.* Returning to the setting of Example 1, we now assume that there is also an external agent whose role is to shift, select, or extract an appropriate automaton. More precisely, assume that the agent's variables $\xi_1, \xi_2, \ldots$ are functions of the state of the automaton and the state of the input-output system. Therefore, as the state of the automaton and the state of the input-output system evolve, the values of the agent's variables change. Physically, one can imagine that the automaton and input-output system have sensors which update the agents' variables. See Figure 2.

As the variables change their values, the truth values of the predicates $q_1, q_2, \ldots$ change, and hence so do the values of the predicates $p_1, p_2, \ldots$ which are determined by rules.

For simplicity we assume assume that the role of the predicates p_i is to select automata number i. In other words, just as the automaton selects the control law for the input-output system, the agent selects the appropriate automaton from a suitable collection of automata. Of course, there are much more general ways for the predicates and rules to select an automaton.

The role of each agent is to observe when its plant with digital controller fails to meet the performance specification for that plant, and then to compute a new digital controller and put it into place for the plant. The agent is viewed as operating off-line or, put in another way, to operate on a time scale which is "slow" compared to the "fast" time scale needed for on line control in the real time hybrid system.

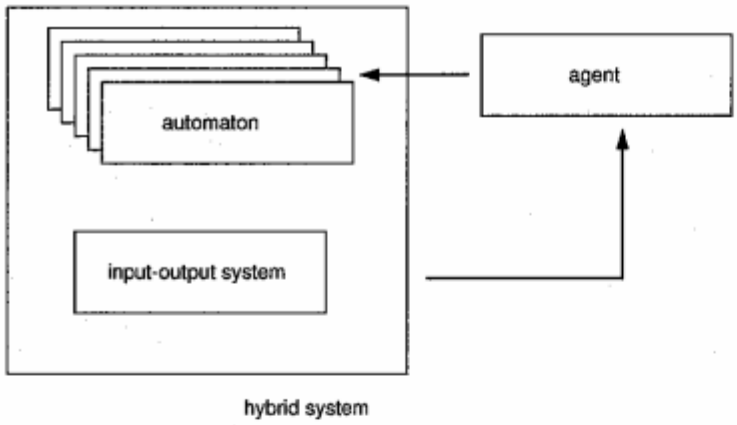

Figure 2: A simple example of a hybrid system with an external agent, in which the role of the external agent is to select an appropriate automaton on the basis of updating variables and evaluating rules and predicates.

4 Multiple Agents

In the previous two sections, we examined a very simple example of a hybrid system and of a hybrid system with external agent. In practice, two simple variants of this approach are often used.

Example 3. *Agents Which Use Historical Data and Other Variants.* Many other variants are possible. Perhaps most important is the ability to use historical data about the behavior of the input-output system and automaton. This data may be coded into appropriate variables, which are then used by the predicates and rules. The efficient management of this data is an important component in the design of high performance hybrid systems and one of the reasons light weight persistent object managers become important.

Another important variant is when measurements of the input-output system and of the digital automaton are used to add new predicates to agents. We also note that for some applications, such as the expanding networks which arise in distributed interactive simulation and interactive multimedia, agents also have to be capable of spawning new agents and of deactivating agents.

Example 4. *Multiple Agents—MAHCA.* Recently Kohn and Nerode have developed a multiple agent distributed hybrid control architecture (MAHCA) for extracting control programs for systems of cooperating agents controlling distributed hybrid systems ([8], [9], [10], [14] and [13]).

In [9] the hybrid system is modeled as a "carrier manifold", with coordinates representing both physical, logical and goal variables. The evolution of the hybrid system is a trajectory on this carrier manifold. They have shown how to formulate cooperating autonomous software agents controlling a hybrid system and meeting dynamical and logical constraints and goals as a single distributed relaxed variational problem, and then developed methods to solve the latter problem for the required controls.

In other words, in this approach the input-output system, the automaton, and the agent are all viewed from a continuous viewpoint.

5 Persistent Object Stores

The success of the theory of hybrid systems and hybrid systems with external agents is critically dependent on the development of the appropriate technology to implement the ideas described above. In this section, we focus on one technology which is emerging as a key component for engineering large complex hybrid systems controlled by external agents: the use of persistent object stores to store and manage the discrete data associated with the problem. This idea was first developed in [1] and [3].

Data which exists independently of the process which creates it is said to be persistent; otherwise, it is called transient. Persistence is most familiar as a feature of databases, but recently persistence is emerging as basic service provided by next generation operating systems. Using distributed, low overhead, high performance persistent object managers is proving useful in a variety of applications from digital libraries to high performance simulations [7].

A persistent object manager is a software tool which creates, accesses and updates persistent objects. By a "light-weight" persistent object manager we mean one in which no additional functionality is provided. Persistent object managers form the core of an object oriented database in which additional functionality, such as transactions, back up, recovery, concurrency, etc., is present.

One of the problems in using a database to manage discrete symbolic data for computationally intensive tasks for hybrid systems is that traditional general purpose databases incur too much overhead to provide acceptable performance. A basic alternative is to use "light weight" software tools to provide persistence and to manage persistent data [4].

For example, Figure 3 from [5] compares four different technologies for computationally intensive queries on high energy physics data: file based access using a memory management system called YBOS, a commercial relational database, a commercial object oriented database, and persistent object manager called PTool. The PASS Project developed three application specific benchmarks (mmm, bs, and avg) appropriate for the management and analysis of high energy physics data [5]. The histogram measures the time in seconds for tests on 109 Megabytes of data running on a Sun Microsystems SparcStation 1 with Sun OS 4.1.3. The advantages of using a persistent object manager are clear.

Persistent Object Stores of Trajectories. As mentioned above, hybrid systems may be viewed from two complementary viewpoints: the discrete viewpoint in which the continuous system is viewed as a discrete collection of continuous objects, such as trajectory segments; and the continuous viewpoint, in which the discrete automaton is viewed as a continuous system with discrete transitions.

From the discrete viewpoint, persistent object managers appear, for example, as a mechanism to manage physical collections of trajectories objects. Once managed in this framework, it is a simple task to interface discrete digital automata to select trajectory segments with desired performance specifications or to provide appropriate control laws to the nonlinear input-output system. See [3] for further discussion.

Example 5. *TrajectoryStore.* In [3] a path planning algorithm for hybrid systems is described. In this algorithm, during preprocessong large numbers of trajectory segments for a hybrid system are computed and used to populate a persistent object store of trajectory segments. Subsequently, a request to approximate a desired flight path is interpreted as a query on the persistent store of trajectory segments. The query returns a sequence of trajectory segments which approximate the desired flight path. Near real time performance is obtained by using appropriate indices. One strength of this approach is that the actual controls required can be retrieved along with the trajectory segments at no additional cost.

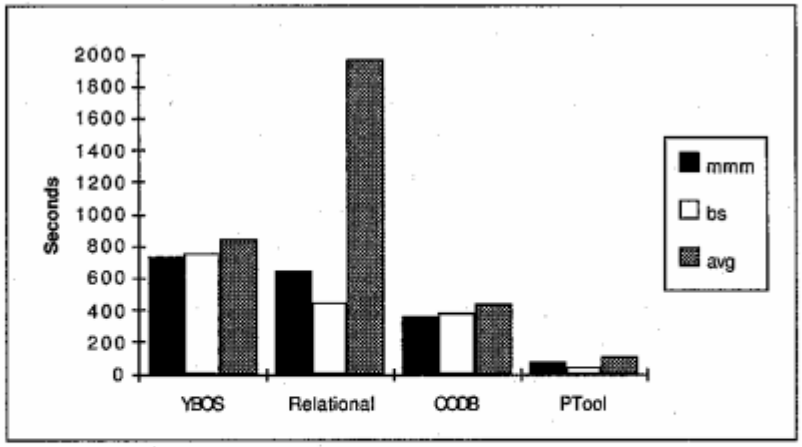

Figure 3: The chart compares four different technologies for computationally intensive queries on high energy physics data: file based access using a memory management system called YBOS, a commercial relational database, a commercial object oriented database, and a persistent object manager called PTool. The PASS Project developed three application specific benchmarks (mmm, bs, and avg) appropriate for the management and analysis of high energy physics data [5]. The histogram measures the time in seconds for tests on 109 Megabytes of data running on a Sun Microsystems SparcStation 1 with Sun OS 4.1.3.

Persistent Object Stores of Predicates. An external agent also has discrete symbolic data. It is natural to use a persistent object to manage this data. More specifically, a low overhead, high performance object manager can be used to manage the distributed collection of predicates required by a system of cooperating autonomous agents in order to provide the performance required for real time applications. High performance inferencing could then be achieved by the use of a companion software tool which would access predicate rules and facts stored by the persistent object manager and provide high performance inferencing. Thus we propose a "light weight" inferencer plus a distributed, persistent object manager as an alternative to Prolog or LISP for implementing distributed agents.

Example 6. *MAHCA.*

We return to the setting of Example 4 which uses multiple agents for the control of hybrid systems. The present implementation described in [10] for autonomous controlling agents uses compiled MetaProlog programs for each agent. The agents observe their local plant behavior and occasionally change the finite automaton controller of the plant when the performance specification is not met.

With currently available commercial implementations, MetaProlog has serious known limitations when used for real time control of large systems. There are two causes. One is the basic Prolog mechanism of backwards search. This one can be eliminated at some additional overhead by forward chaining implementations. But the other, more troublesome, cause is the necessity of retrieving and depositing rules and facts in large databases for online theorem proving in real time. In the approach described in [10], the agent proves the theorem, for a discretized optimization problem expressed in automaton equation terms, that there is a control law that is sufficiently optimal for use for the next interval of time. The witness substitution for that Prolog theorem is the desired next control law, a modern version of Green's trick.

What is crucial here is to eliminate the bottleneck due to having to consult a large database to prove this theorem, that is, to choose a control law that is essentially obtained by back propagation (backwards chaining) from the final goal. These databases of rules and facts can be very large because the dimension and size of the state space of the controlled systems can be large. Prolog based systems are not very fast in performing this task, as the lack of commercial exploitation of deductive database systems over a twenty-five year period attests. General purpose databases and MetaProlog currently carry overhead that makes such large real time applications hard to implement, necessitating disabling much of the Prolog mechanisms and using ad hoc database management techniques.

Using a light weight persistent object manager together with a light weight inferencer to manage predicate data, stored rules and facts, as proposed in [1] appears to be a promising approach.

6 Summary

A hybrid system is a network of continuous input-output systems and discrete digital automata. Since hybrid systems contain both continuous and discrete data, implementations must manage both types of data.

It is currently a challenge to manage the discrete data of a hybrid system. One approach discussed in this paper is to use "light-weight" persistent object stores and specialized algorithms exploiting them. A basic example of this approach is a class of path planning algorithms introduced in [3] which uses a light-weight persistent object manager called PTool to view a path planning algorithm as a query on a large persistent store of trajectory segments of hybrid systems.

In the context of hybrid systems, external agents were introduced in [10] and [11], as means of introducing new digital automata into a hybrid system when the hybrid system was no longer meeting the desired performance specification. Currently, the external agents used in [14] and [15] work by inferenc-

ing from a set of variables, predicates and rules using MetaProlog. It has been an open problem as to the best way to provide management for the predicates of the agent programs to speed up computation of control automata. An alternative introduced in [1] is to use a "light-weight" persistent object manager to manage a persistent object store containing variables, predicates, and rules and specialized software tools to perform the inferencing.

This approach appears to be particularly promising for the multiple agent distributed hybrid control architecture (MAHCA) introduced in [12], [13] and [14]. In this approach, hybrid systems are modeled as a distributed mixed continuous-discrete optimization problem in which each agent has to recompute and replace control automata on line as needed, and the management of the predicate data is particularly challenging.

In this paper we have clarified the notion of agent as expressed in a persistent object store, and have proposed implementing agents by a combination of a light weight persistent object store plus a light weight inferencer for online extraction of control automata. We believe that this has the capacity to bypass the bottlenecks inherent in MetaProlog and in conventional and deductive databases.

This is an instance of the general principle that database implementations optimized for the class of queries actually used in the intended application have much to recommend them if real time and large databases are involved.

The principal advantage we see in this approach is its real time scalability to large applications. This has been demonstrated by for aircraft path planning [3] and for real time data retrieval for physical simulations [5] and [7].

References

[1] R. L. Grossman, "Managing persistent object stores of predicates," Oak Park Research Technical Report, Number 92-R1, May, 1992.

[2] R. L. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, "Hybrid Systems", Springer Lecture Notes in Computer Science, Springer-Verlag, Bonn, 1993.

[3] R. L. Grossman, D. Valsamis and X. Qin, "Persistent stores and hybrid systems," Proceedings of the 32st IEEE Conference on Decision and Control, IEEE Press, 1993, pp. 2298-2302.

[4] R. L. Grossman and X. Qin, "Ptool: a scalable persistent object manager," *Proceedings of SIGMOD 1994*, ACM, 1994, page 510.

[5] R. L. Grossman, X. Qin, D. Valsamis, W. Xu, C. T. Day, S. Loken, J. F. MacFarlane, D. Quarrie, E. May, D. Lifka, D. Malon, L. Price, "Analyzing High Energy Physics Data Using Databases: A Case Study," *Proceedings of the Seventh International Working Conference on Scientific and Statistical Database Management*, IEEE Press, 1994, to appear.

[6] R. L. Grossman and R. G. Larson, "An algebraic approach to hybrid systems," Journal of Theoretical Computer Science, Jan. 1995, to appear.

[7] R. L. Grossman, N. Araujo, X. Qin, H. Ramamoorthy, and W. Xu, "Managing physical folios of objects between nodes," Proceedings of the Sixth International Workshop on Persistent Object Systems, Springer-Verlag, 1995, to appear.

[8] W. Kohn and A. Nerode, "An autonomous control theory: an overview," 1992 IEEE Symposium on Computer Aided Control System Design (March 17–19, 1992, Napa Valley, CA), pp 204–210.

[9] W. Kohn and A. Nerode "Models for hybrid systems: automata, toplogies, controllability and observability", in R. L. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, "Hybrid Systems", Springer Lecture Notes in Computer Science, Springer-Verlag, Bonn, 1993.

[10] W. Kohn and A. Nerode "Multiple agent autonomous control", Proceedings of the 31st IEEE CDC, IEEE, pp. 2956–2966, 1993.

[11] W. Kohn and A. Nerode, "Multiple agent autonomous hybrid control systems", in "Logical Methods", Birkhauser, 1993.

[12] W. Kohn, A. Nerode, J. B. Remmel, and X. Ge, "Multiple agent hybrid control: carrier manifolds and chattering approximations to optimal control", CDC94.

[13] W. Kohn, A. Nerode, J. B. Remmel, A. Yakhnis, "Viability in hybrid systems", J. Theoretical Computer Science, to appear in 1995.

[14] A. Nerode, J. James, and Wolf Kohn, "Multiple agent reactive control of distributed interactive simulations," Proc. Army Workshop on Hybrid Systems and Distributed Simulation, Feb. 28-March 1, 1994.

[15] A. Nerode, J. James, Wolf Kohn, and N. DeClaris, Intelligent integration of medical models, Proc. IEEE Conference on Systems, Man, and Cybernetics, San Antonio, 1-6 Oct. 1994.