

Behavioral Programming*

Michael S. Branicky

Electrical Engineering and Computer Science Department
Case Western Reserve University, Cleveland, OH 44106-7221 U.S.A.
branicky@alum.mit.edu

Abstract

Motivated by biology and a study of complex systems, intelligent behavior is typically associated with a hierarchical structure, where the lowest level is usually characterized by continuous-variable dynamics and the highest by a logical decision-making mechanism. Consistent with biological findings, we instantiate a particular version of such a hierarchy: a “Middle-out Architecture.” Significant there is the plasticity of circuits on both layers, as well as plasticity in their interactions. Intelligent systems must accomplish two broad tasks within this architecture. Based on experience, they must tune their lower-level, continuous, sensorimotor circuits to produce local behaviors that are viable and robust to slight environmental changes. Predicated on focus of attention on sensory input, they must modulate and coordinate these local behaviors to maintain situational relevance and accomplish global goals. Hence, if we are to understand learning and intelligent systems, we must develop ways of *behavioral programming* (taking symbolic descriptions of tasks and predictably translating them into dynamic descriptions that can be composed out of lower-level controllers) and *co-modulation* (simultaneously tuning continuous lower-level and symbolic higher-level circuits and their interactions). Herein, we begin the study of the first of these problems.

Introduction

Though manipulators have gotten more dextrous and sensors more accurate, though processors have gotten faster, memory cheaper, and software easier to write and maintain, truly agile engineering systems that learn and exhibit intelligent behavior have not been demonstrated. The substrate is not lacking; a theoretical approach yielding engineering principles is needed.

Both top-down (cognitive) and bottom-up (reactive) approaches to learning and intelligent systems (LIS) have yielded successes, but only for highly-complex programs performing high-level tasks in highly-structured

environments and for simple agents performing low-level tasks in mildly-changing environments, respectively. In contrast, humans and animals possess a middle-level competence between reactive behavior and cognitive skills. They are complex, constrained, and must work in environments whose structure changes.

Thus, the problem of producing LIS confronts us directly with building a theory that bridges the gap between the traditional “top down” and “bottom up” approaches. The missing “middle out” theory we propose requires the development of models and mathematical tools that bridge the gap between “top-down” and “bottom-up” methods.

Motivated by biology and a study of complex systems, intelligent behavior is typically associated with a hierarchical structure. Such a hierarchy exhibits an increase in reaction time and abstraction with increasing level. In both natural and engineered systems the lowest level is usually characterized by continuous-variable dynamics (acting upon and producing continuous signals) and the highest by a logical decision-making mechanism (acting upon and producing discrete symbols). Consistent with biological findings, we instantiate a particular version of such a hierarchy: a “Middle-out Architecture.” Significant there—and below—is the plasticity of circuits on both layers, as well as plasticity in their interactions.

Intelligent systems must accomplish two broad tasks within this architecture. Based on experience, they must tune their lower-level, continuous, sensorimotor circuits to produce local behaviors that are viable and robust to slight environmental changes. Predicated on focus of attention on sensory input, they must modulate and coordinate these local behaviors to maintain situational relevance and accomplish global goals. Hence, if we are to understand LIS, we must develop ways of

- Behavioral programming. Taking symbolic descriptions of tasks and predictably translating them into dynamic descriptions that can be composed out of lower-level controllers.
- Co-modulation. Simultaneously tuning continuous lower-level and symbolic higher-level circuits and their interactions.

* This work was supported by the National Science Foundation, under grant number DMI97-20309.

In control theory and computer science, a mathematics of hybrid systems—those combining continuous and discrete states, inputs, and outputs, e.g., differential equations coupled with finite automata—is emerging that enables the careful study of such questions. Most importantly, we have pioneered a theory and algorithms for optimal control of hybrid systems (Branicky *et al.*, 1998; Branicky, 1995) that are foundational in solving the above mixed symbolic-reactive interdependent optimization problems of LIS. Herein, we study their application to the problem of behavioral programming.

Outline of Paper. In the next section, we motivate our “Middle-out Approach” to learning and intelligent systems. In Section 3, we summarize a formal model framework, *hybrid dynamical systems*, that can be used to model hierarchical systems, from sensor-actuator dynamics to logical decision-making. In Section 4, we review some results on optimal control of hybrid systems that is applicable to solving problems of behavioral programming. Section 5 discusses the behavioral programming idea in more detail and makes the connection with hybrid systems theory explicit.

“Top-Down” vs. “Bottom-Up” vs. “Middle-Out”

Currently, there are two unreconciled approaches to learning and intelligent systems:

- the top-down, cognitive, “good old-fashioned AI” approach;
- the bottom-up, reactive, “emergence of behavior” approach.¹

The first approach has shown successes in structured environments, primarily where search dominates, such as theorem proving or chess playing. In short, “think globally.” The second is newer and has shown successes in less-structured environments, primarily where sensorimotor control dominates, such as robotic gait generation or simulated organisms.² In short, “(re)act locally.”

Our experimental, theoretical, and engineering studies of learning and intelligent systems suggests that neither of these approaches is fully adequate. To illustrate, we describe the problems in applying them to a real-world “intelligent” engineering system and problem, that is error-recovery in an agile manufacturing workcell, in which random errors are inevitable because of statistical mechanical variation:

- **Top-down approach:** Develop a detailed specification and plan for the entire process; run the process,

¹Examples of each are Cyc [D.B. Lenat, CYC: A large-scale investment in knowledge infrastructure. *Comm. of the ACM*, 38:11, 1995] and Cog [R.A. Brooks and L.A. Stein. Building brains for bodies, MIT AI Lab Memo #1439, August 1993].

²D. Terzopoulos et al., Artificial fishes: Autonomous locomotion, perception, behavior and learning in a simulated physical world, *Artificial Life*, 1:327–351, 1995.

keep an error log, categorize errors and define specific response algorithms; incorporate these as choice points in the robots’ control. This is an ineffective approach because task domains change relatively quickly, and the number of possible error conditions is potentially infinite.

- **Bottom-up approach:** Incorporate a range of local reflexes that allow the robots to respond rapidly and automatically to small perturbations in the task. However, this approach does not deal with the larger problem of defining sequences of appropriate actions in the face of a novel problem, optimizing these responses as a function of experience, and adjusting them flexibly to unforeseen contingencies.

The top-down approach tacitly assumes that locally relevant behavior will be produced by thinking globally—and exhaustively. The bottom-up approach is predicated on the belief that global behavior will “emerge” by combining many small behavioral loops. More generally, these two approaches have yielded successes, but to date only for highly-complex programs performing high-level tasks in highly-structured environments (top-down) or simple agents performing low-level tasks in mildly-changing environments (bottom-up).

However, humans and animals occupy a middle-level of competence, lying between the domains of successful top-down and bottom-up systems. They are *responsive*; they must close sensorimotor loops to deal with changing environments. Also, procedural tasks and procedural learning require *sequences* of actions, not just simple behavioral maps. Finally, they require *quick* responses, achieved by *modulating* previously learned behaviors and the current focus of attention. Waiting for globally optimal plans or exhaustively analyzed sensor data from higher-levels means failure. We do not see much effort “in the middle,” either from top-down researchers building down, or bottom-up researchers building up. One might argue that such research is unnecessary because future developments of top-down and bottom-up theory will lead to seamless matching between levels. Our group’s working hypothesis is that there is no *a priori* reason to believe that a purely symbolic and purely reactive approach can be seamlessly matched without the development of “middle-out” theory and practice. For example, there is no (nontrivial) finite automata representation of certain simple two-dimensional, quasi-periodic dynamical systems (Di Gennaro *et al.*, 1994). Therefore, the *goal* of middle-out theory is to achieve a better understanding of how natural LIS function and how artificial LIS can be effectively implemented by filling the gap between top-down and bottom-up knowledge. Our *approach* to developing this new theory supplies the following missing elements of the two more traditional approaches:

Middle-Out Approach. Develop ways of taking symbolic descriptions of tasks and *predictably* translating them into dynamic descriptions that can be

composed out of sensorimotor controllers.

Recapitulating, top-down and bottom-up approaches have contributed in method and principles, each in their own domain of applicability. Humans and animals possess a middle-level competence between reactive behavior and cognitive skills. A “middle-out” approach is necessary, but currently lacking. Recent advances in hybrid systems theory, summarized below, have us poised—theoretically and experimentally—to pursue such middle ground.

What are Hybrid Systems?

Hybrid systems involve both continuous-valued and discrete variables. Their evolution is given by equations of motion that generally depend on all variables. In turn these equations contain mixtures of logic, discrete-valued or **digital** dynamics, and continuous-variable or **analog** dynamics. The continuous dynamics of such systems may be continuous-time, discrete-time, or mixed (sampled-data), but is generally given by differential equations. The discrete-variable dynamics of hybrid systems is generally governed by a **digital automaton**, or input-output transition system with a countable number of states. The continuous and discrete dynamics interact at “event” or “trigger” times when the continuous state hits certain prescribed sets in the continuous state space. See Fig. 1.

Hybrid control systems are control systems that involve both continuous and discrete dynamics and continuous and discrete controls. The continuous dynamics of such a system is usually modeled by a controlled vector field or difference equation. Its hybrid nature is expressed by a dependence on some discrete phenomena, corresponding to discrete states, dynamics, and controls.

For the remainder of this section, we concentrate on modeling of hybrid systems. In particular, we introduce **general hybrid dynamical systems** as interacting collections of dynamical systems, each evolving on continuous state spaces, and subject to continuous and discrete controls, and some other discrete phenomena. The reader is referred to (Branicky, 1995) for more details.

The Basis: Hybrid Dynamical Systems Dynamical Systems

The notion of dynamical system has a long history as an important conceptual tool in science and engineering. It is the foundation of our formulation of hybrid dynamical systems. Briefly, a **dynamical system** is a system $\Sigma = [X, \Gamma, \phi]$, where X is an arbitrary topological space, the **state space** of Σ . The **transition semigroup** Γ is a topological semigroup with identity. The (**extended**) **transition map** $\phi : X \times \Gamma \rightarrow X$ is a continuous function satisfying the identity and semigroup properties (Sontag, 1990). A **transition system** is a dynamical system as above, except that ϕ need not be continuous.

Examples of dynamical systems abound, including autonomous ODEs, autonomous difference equations, finite automata, pushdown automata, Turing machines, Petri nets, etc. As seen from these examples, both digital and analog systems can be viewed in this formalism. The utility of this has been noted since the earliest days of control theory.

We also denote by “dynamical system” the system $\Sigma = [X, \Gamma, f]$, where X and Γ are as above, but the **transition function** f is the **generator** of the extended transition function ϕ .³ We may also refine the above concept by introducing dynamical systems with initial and final states, input and output, and timing maps.⁴

On to Hybrid . . .

Briefly, a hybrid dynamical system is an indexed collection of dynamical systems along with some map for “jumping” among them (switching dynamical system and/or resetting the state). This jumping occurs whenever the state satisfies certain conditions, given by its membership in a specified subset of the state space. Hence, the entire system can be thought of as a sequential patching together of dynamical systems with initial and final states, the jumps performing a reset to a (generally different) initial state of a (generally different) dynamical system whenever a final state is reached.

More formally, a **general hybrid dynamical system (GHDS)** is a system $H = [Q, \Sigma, A, G]$, with its constituent parts defined as follows.

- Q is the set of **index states**, also referred to as **discrete states**.
- $\Sigma = \{\Sigma_q\}_{q \in Q}$ is the collection of **constituent** dynamical systems, where each $\Sigma_q = [X_q, \Gamma_q, \phi_q]$ (or $\Sigma_q = [X_q, \Gamma_q, f_q]$) is a dynamical system as above. Here, the X_q are the **continuous state spaces** and ϕ_q (or f_q) are called the **continuous dynamics**.
- $A = \{A_q\}_{q \in Q}$, $A_q \subset X_q$ for each $q \in Q$, is the collection of **autonomous jump sets**.
- $G = \{G_q\}_{q \in Q}$, $G_q : A_q \rightarrow \bigcup_{q \in Q} X_q \times \{q\}$, is the collection of (**autonomous**) **jump transition maps**. These are also said to represent the **discrete dynamics** of the HDS.

Thus, $S = \bigcup_{q \in Q} X_q \times \{q\}$ is the **hybrid state space** of H . For convenience, we use the following shorthand. $S_q = X_q \times \{q\}$ and $A = \bigcup_{q \in Q} A_q \times \{q\}$ is *the* autonomous jump set. $G : A \rightarrow S$ is *the* autonomous jump transition map, constructed component-wise in the obvious way. The **jump destination sets** $D = \{D_q\}_{q \in Q}$ are given by $D_q = \pi_1[G(A) \cap S_q]$, where π_i

³In the case of $\Gamma = Z$, $f : X \rightarrow X$ is given by $f \equiv \phi(\cdot, 1)$. In the case of $\Gamma = R$, $f : X \rightarrow TX$ is given by the vector fields $f(x) = d\phi(x, t)/dt|_{t=0}$.

⁴*Timing maps* provide a mechanism for reconciling different “time scales,” by giving a uniform meaning to different transition semigroups in a hybrid system.

is projection onto the i th coordinate. The **switching** or **transition manifolds**, $M_{q,p} \subset A_q$ are given by $M_{q,p} = G_q^{-1}(p, D_p)$, i.e., the set of states from which transitions from index q to index p can occur.

Roughly,⁵ the dynamics of the GHDS H are as follows. The system is assumed to start in some hybrid state in $S \setminus A$, say $s_0 = (x_0, q_0)$. It evolves according to $\phi_{q_0}(x_0, \cdot)$ until the state enters—if ever— A_{q_0} at the point $s_1^- = (x_1^-, q_0)$. At this time it is instantly transferred according to transition map to $G_{q_0}(x_1^-) = (x_1, q_1) \equiv s_1$, from which the process continues. See Fig. 3.

Dynamical Systems. The case $|Q| = 1$ and $A = \emptyset$ recovers all dynamical systems.

Hybrid Systems. The case $|Q|$ finite, each X_q a subset of R^n , and each $\Gamma_q = R$ largely corresponds to the usual notion of a hybrid system, viz. a coupling of finite automata and differential equations. Herein, a **hybrid system** is a GHDS with Q countable, and with $\Gamma_q \equiv R$ (or R_+) and $X_q \subset R^{d_q}$, $d_q \in Z_+$, for all $q \in Q$: $[Q, \{X_q\}_{q \in Q}, R_+, \{f_q\}_{q \in Q}, A, G]$, where f_q is a vector field on $X_q \subset R^{d_q}$.⁶

Changing State Space. The state space may change. This is useful in modeling component failures or changes in dynamical description based on autonomous—and later, controlled—events which change it. Examples include the collision of two inelastic particles or an aircraft mode transition that changes variables to be controlled. We also allow the X_q to overlap and the inclusion of multiple copies of the same space. This may be used, for example, to take into account overlapping local coordinate systems on a manifold.

Hierarchies. We may iteratively combine hybrid systems H_q in the same manner, yielding a powerful model for describing the behavior of hierarchical systems (cf. Harel’s statecharts).

... And to Hybrid Control

A **controlled general hybrid dynamical system (GCHDS)** is a system $H_c = [Q, \Sigma, A, G, V, C, F]$, with its constituent parts defined as follows.

- Q , A , and S are defined as above.
- $\Sigma = \{\Sigma_q\}_{q \in Q}$ is the collection of controlled dynamical systems, where each $\Sigma_q = [X_q, \Gamma_q, f_q, U_q]$ (or $\Sigma_q = [X_q, \Gamma_q, \phi_q, U_q]$) is a controlled dynamical system as above with (extended) transition map parameterized by **control set** U_q .
- $G = \{G_q\}_{q \in Q}$, where $G_q : A_q \times V_q \rightarrow S$ is the **autonomous jump transition map**, parameterized

⁵We make more precise statements in (Branicky, 1995).

⁶Here, we may take the view that the system evolves on the state space $R^* \times Q$, where R^* denotes the set of finite, but variable-length real-valued vectors. For example, Q may be the set of labels of a computer program and $x \in R^*$ the values of all currently-allocated variables. This then includes Smale’s tame machines.

by the **transition control set** V_q , a subset of the collection $V = \{V_q\}_{q \in Q}$.

- $C = \{C_q\}_{q \in Q}$, $C_q \subset X_q$, is the collection of **controlled jump sets**.
- $F = \{F_q\}_{q \in Q}$, where $F_q : C_q \rightarrow 2^S$, is the collection of **controlled jump destination maps**.

As shorthand, G , C , F , V may be defined as above. Likewise, jump destination sets D_a and D_c may be defined. In this case, $D \equiv D_a \cup D_c$.

Roughly, the dynamics of H_c are as follows. The system is assumed to start in some hybrid state in $S \setminus A$, say $s_0 = (x_0, q_0)$. It evolves according to $\phi_{q_0}(\cdot, \cdot, u)$ until the state enters—if ever—either A_{q_0} or C_{q_0} at the point $s_1^- = (x_1^-, q_0)$. If it enters A_{q_0} , then it *must* be transferred according to transition map $G_{q_0}(x_1^-, v)$ for some chosen $v \in V_{q_0}$. If it enters C_{q_0} , then we *may* choose to jump and, if so, we may choose the destination to be any point in $F_{q_0}(x_1^-)$. In either case, we arrive at a point $s_1 = (x_1, q_1)$ from which the process continues. See Fig. 4.

Control results for this model are derived in (Branicky *et al.*, 1998); they are summarized in Section 4.

Definition 1 (Admissible Controls) *The admissible control actions available are the continuous controls $u \in U_q$, exercised in each constituent regime; the discrete controls $v \in V_q$, exercised at the autonomous jump times (i.e., on hitting set A); and the intervention times and destinations of controlled jumps (when the state is in C).*

Hybrid Control

Theoretical Results. We consider the following optimal control problem for controlled hybrid systems. Let $a > 0$ be a **discount factor**. We add to our model the following known maps:

- **Running cost**, $k : S \times U \rightarrow R_+$.
- **Autonomous jump cost and delay**, $c_a : A \times V \rightarrow R_+$ and $\Delta_a : A \times V \rightarrow R_+$.
- **Controlled jump (or impulse) cost and delay**, $c_c : C \times D_c \rightarrow R_+$ and $\Delta_c : C \times D_c \rightarrow R_+$.

The total discounted cost is defined as

$$\int_T e^{-at} k(x(t), u(t)) dt + \sum_i e^{-a\sigma_i} c_a(x(\sigma_i), v_i) + \sum_i e^{-a\zeta_i} c_c(x(\zeta_i), x(\zeta'_i)) \quad (1)$$

where $T = R_+ \setminus (\bigcup_i [\tau_i, \Gamma_i))$, $\{\sigma_i\}$ (resp. $\{\zeta_i\}$) are the successive pre-jump times for autonomous (resp. impulsive) jumps and ζ'_i is the post-jump time (after the delay) for the j th impulsive jump. The **decision or control** variables over which Eq. (1) is to be minimized are the *admissible controls* of our controlled hybrid system (see Def. 1). Under some assumptions (the necessity of which are shown via examples) we have the following results (Branicky *et al.*, 1998):

- A finite optimal cost exists for any initial condition. Furthermore, there are only finitely many autonomous jumps in finite time.
- Using the relaxed control framework, an optimal trajectory exists for any initial condition.
- For every $\epsilon > 0$ an ϵ -optimal control policy exists wherein $u(\cdot)$ is precise, i.e., a Dirac measure.
- The value function, V , associated with the optimal control problem is continuous on $S \setminus (\partial A \cup \partial C)$ and satisfies the **generalized quasi-variational inequalities (GQVIs)**, which are formally derived in (Branicky *et al.*, 1998).

Algorithms and Examples. We have outlined four approaches to solving the generalized quasi-variational inequalities (GQVIs) associated with optimal hybrid control problems (Branicky, 1995; Branicky and Mitter, 1995). Our algorithmic basis for solving these GQVIs is the generalized Bellman Equation: $V^*(x) = \min_{p \in \Pi} \{g(x, p) + V^*(x'(x, p))\}$, where Π is a generalized set of actions. The three classes of actions available in our hybrid systems framework at each x are the admissible control actions from Def. 1. From this viewpoint, generalized policy and value iteration become solution tools (Branicky, 1995).

The key to *efficient* algorithms for solving optimal control problems for hybrid systems lies in noticing their strong connection to the models of impulse control and piecewise-deterministic processes. Making this explicit, we have developed algorithms similar to those for impulse control (Costa and Davis, 1989) and one based on linear programming (Costa, 1991; Ross, 1992) (see (Branicky, 1995)). Three illustrative examples are solved in (Branicky, 1995). In each example, the synthesized optimal controllers verify engineering intuition.

Behavioral Programming

Others have noticed the powerful possibility of solving complex learning problem by composing predictable local behaviors to achieve predictable global goals. For example, (Krishnaswamy and Newman, 1992) presents a means to string goals together in order to accomplish global planning (with local obstacle avoidance guaranteed by the reflex and servo layers). Related work in procedural tasks such as juggling has proved successful (Burrige *et al.*, 1995). Our behavioral programming method starts by constructing families of controllers with guaranteed stability and performance properties. These constituent controllers can then be automatically combined on-line using graph-theoretic algorithms so that higher-level dynamic tasks can be accomplished without regard to lower-level dynamics or safety constraints. The process is analogous to building sensible speech using an adaptable, but predictable, phoneme generator. Such problems can be cast as optimal hybrid control problems as follows:

- Make N copies of the continuous state-space, one corresponding to each behavior, including a copy of the

goal region in each constituent space.

- Using as dynamics of the i th copy the equations of motion resulting when the i th behavior is in force.
- Allow the controlled jump set to be the whole state space; disable autonomous jumps. Impose a small switching cost.
- Solve an associated optimal hybrid control problem, e.g., penalize distance from the goal.

The result (if the goal is reachable) is a switching between behaviors achieving that achieves goal.

These problems can be solved by a variety of means, as outlined above. See also (Branicky and Mitter, 1995). Recently, though, we have extended the so-called “fast marching methods” of Sethian (Sethian, 1996) to the hybrid case. These hold promise for the efficient solution of certain classes of behavioral programming tasks. See (Branicky and Hebbar, 1999) for details.

References

- Branicky, M. S. 1995. Studies in Hybrid Systems: Modeling, Analysis, and Control. Doctor of Science dissertation. Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science.
- Branicky, M. S. and Hebbar, R. 1999. Fast marching for hybrid systems. In: *Proceedings IEEE Conf. Computer Aided Control Systems Design*, submitted, Kohala Coast, Island of Hawaii, HI.
- Branicky, M. S. and Mitter, S. K. 1995. Algorithms for optimal hybrid control. In: *Proceedings IEEE Conf. Decision and Control*, pp. 2661–2666, New Orleans, LA.
- Branicky, M. S.; Borkar, V. S.; and Mitter S. K. 1998. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions Automatic Control* **43**(1), 31–45.
- Burrige, R. R.; Rizzi, A. A.; and Koditschek, D. E. 1995. Toward a dynamical pick and place. In: *Proc. RSJ/IEEE International Conf. Intelligent Robots and Systems*.
- Costa, O. L. V. 1991. Impulse control of piecewise-deterministic processes via linear programming. *IEEE Transactions Automatic Control* **36**(3), 371–375.
- Costa, O. L. V. and Davis, M. H. A. 1989. Impulse control of piecewise deterministic processes. *Math. Control Signals Syst.* **2**, 187–206.
- Di Gennaro, S.; Horn, C.; Kulkarni, S. R.; and Ramadge, P. J. 1994. Reduction of timed hybrid systems. In: *Proceedings IEEE Conf. Decision and Control*, pp. 4215–4220 Lake Buena Vista, FL.
- Krishnaswamy, V. K. and Newman, W. S. 1992. On-line motion planning using critical point graphs in two-dimensional configuration space. In: *Proceedings IEEE International Conf. Robotics and Automation*, pp. 2334–2340, Nice, France.
- Ross, S. M. 1992. *Applied Probability Models with Optimization Applications*. Dover. New York.
- Sethian, J. A. 1996. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Material Sciences*, Cambridge University Press. New York.
- Sontag, E. D. (1990). *Mathematical Control Theory*. Springer. New York.

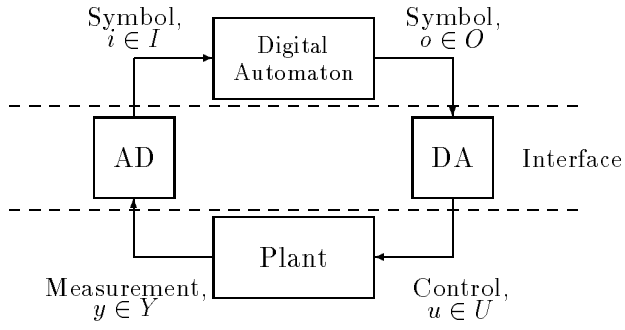


Figure 1: Hybrid System.

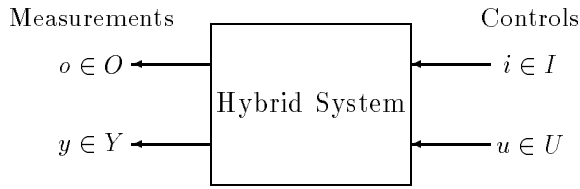


Figure 2: Hybrid Control System.

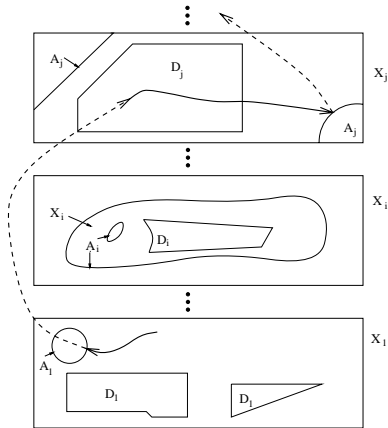


Figure 3: Example dynamics of GHDS.

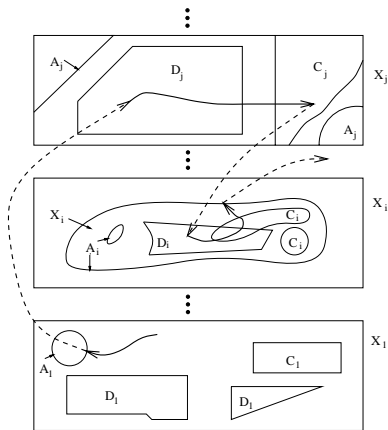


Figure 4: Example dynamics of GCHDS.