Hybrid Knowledge Systems *

V.S. Subrahmanian†

Abstract

In this paper, we describe an architecture, due to the author, called *hybrid knowledge systems* (HKS, for short) that can be used to inter-operate between (1) a specification of the control laws describing a physical system (in particular, this could include specifications such as those of Brockett and/or Nerode and Kohn, but is not limited to those), (2) a collection of databases, knowledge bases and/or other data structures reflecting information about the world in which the physical system being controlled resides, (3) observations (e.g. sensor information) from the external world, and (4) actions that must be taken in response to external observations.

1 Introduction

Deductive databases that interact with, and are accessed by, reasoning agents in the real world (such as logic controllers in automated manufacturing, weapons guidance systems, aircraft landing systems, land-vehicle maneuvering systems, and air-traffic control systems) must satisfy a number of diverse, and often conflicting criteria. In this paper, we will describe a software architecture called *hybrid knowledge systems* (HKS, for short) that supports intelligent real-time reasoning in domains such as control systems. In particular, we will show how, given the physical equations governing the dynamics of a control system, as well as the control laws governing the application of control actions, it is possible for our framework to be used as a platform for developing an intelligent, real-time control system. In other words, this platform enables a smooth integration of the knowledge of a control engineer, and the database technology in the HKS system. In particular, this platform enables HKSs to act as a mediator between database systems, and methods for specifying the dynamics of hybrid control systems (e.g. the frameworks of Brockett [2] and/or the Kohn-Nerode framework).

2 The HKS Architecture for Intelligent Control

In [10], Subrahmanian has outlined an architecture for supporting intelligent real-time reasoning systems. This architecture, known as *hybrid knowledge systems* (HKS, for short) is

*This research was supported by the Air Force Office of Scientific Research under Grant Nr. F49620-93-1-0065 and by the Army Research Office under grant DAAL-03-92-G-0225.

†Institute for Advanced Computer Studies, Institute for Systems Research and Department of Computer Science, University of Maryland, College Park, Maryland 20742. E-mail: vs@cs.umd.edu

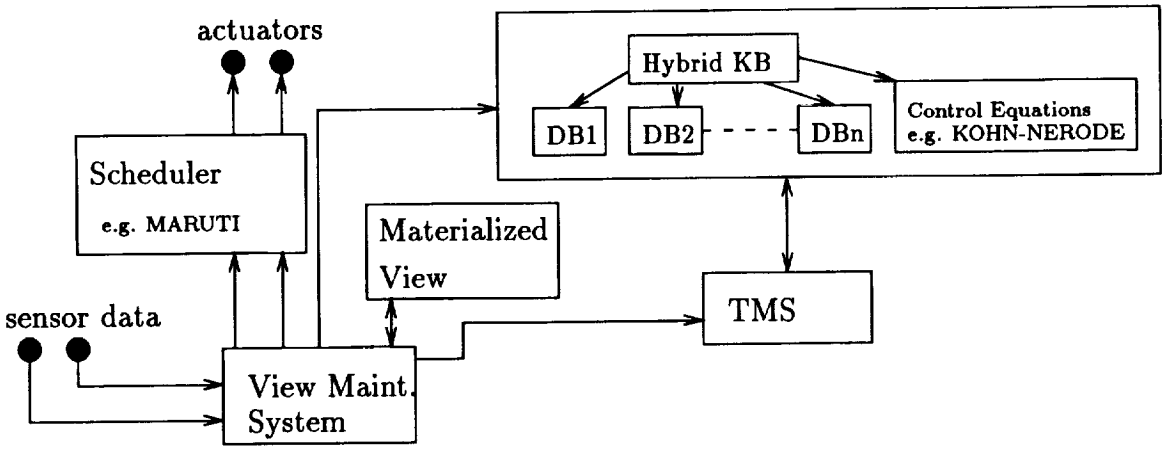

Figure 1: Architecture for Intelligent Support for Real-Time Control Systems

built upon integrating multiple data sources (e.g. sensors), knowledge sources (e.g. knowledge bases, and databases), data structures, and constraint systems. As differential equations are just constraints, the control axioms, and the computations involved in control systems can be represented in the HKS architecture. We describe below, the individual components of the HKS architecture, using the Cruise missile example introduced earlier to illustrate the basic ideas.

2.1 A Snapshot at Time 0

At time 0, the hybrid knowledge base (to be described in detail in Section 2.3) integrates a number of databases, and data domains – one of these consists of a set of numerical differential equations reflecting the dynamics of the physical system being controlled. For example, if we wish to control a missile, this set of equations reflects the control laws used to guide the missile. For those readers who are not control engineers, the dynamics of a control system (e.g. a missile) are specified by (multiple) sets of differential equations reflecting different trajectories (e.g. motions of the missile) of the system being controlled. Each of these sets of equations is called a control law. If one wants to change the trajectory of the system (e.g. the directionality of the missile), one must vary the control law currently being used and determine/specify the time for which that control law is applied.

The initial trajectory of the control system is known, and is denoted by Traj(0) – intuitively, this is an assignment of values to all variables that are used to describe the parameters of the system.

2.2 A Snapshot at Time $t > 0$

Suppose t is an instant of time that occurs during the working of the physical system being controlled. In a missile control example, for instance, t may be a point of time after the

missile has been launched, but before it has completed its mission.

At time t, the HKS architecture would:

- Maintain a small set of facts – this set of facts reflects the current "state" of the environment. In the missile control example, this set of facts includes the position, $\mathsf{POS}(t)$, of the missile, and the position, $(x'(t), y'(t), z'(t))$ of the target at time t. This set of facts is called a *materialized view*. It may, or may not, be consistent with the (relatively static) set of databases integrated by the hybrid knowledge base.
- In addition, at time t, new information may come in, specifying that the the actual values of the variables involved has changed from $\mathsf{Traj}(t)$ to $\mathsf{Traj}(t+1)$. For example, when considering missile control, this may reflect the fact that target has moved from its previous location to a new location.
- Using this information (which reflects a *request to update* the materialized view), the rules in the hybrid knowledge base are used to incrementally determine which set, Act of actions (selected from an available set of actions that the control system can execute) should be performed. Using a specification of the control laws (that will, presumably, be provided by control engineers), the HKS will use these actions to determine the new trajectory. Note that the control laws reside in one of the domains integrated by the hybrid knowledge base[1] and we are not generating them on the fly in real-time; rather, we are selecting certain (possibly parametrized) control laws to apply using the rules in the HKS.

In the rest of this section, we will: (1) explain the basic ideas behind the hybrid knowledge base paradigm [10, 8], and (2) show how control systems can be modeled using the architecture given here, and (3) explain what a materialized view is.

2.3 The Hybrid Knowledge Base Component

Nerode and Subrahmanian have introduced the concept of a hybrid knowledge base for integrating information in multiple data structures and multiple database paradigms. Key to the definition of hybrid knowledge bases is that of a *constraint domain*, described below.

2.3.1 Constraint Domains

Definition 1 Suppose S is a set. The *function space* generated by S, denoted $\mathbf{Func}(S)$, is the smallest set satisfying the following conditions:

1. $S \in \mathbf{Func}(S)$
2. for all integers $n, m \geq 1$, every function $f : S^n \longrightarrow S^m$ is in $\mathbf{Func}(S)$ and
3. if $\emptyset \neq G_1, G_2 \subseteq \mathbf{Func}(S)$, then every function $g : G_1 \rightarrow G_2$ is in $\mathbf{Func}(S)$.

[1] The role of the TMS, or Truth Maintenance System has not been elucidated here. It will be discussed later, in Section 2.5.

Basically, $\mathbf{Func}(S)$ contains not only all functions from S^i to S^j for all $i, j \geq 1$, but also all *functions* on *sets* of functions. For example, if S is the set $\mathbf{R}$ representing the reals, then the definite integral

$$\int_a^b f(x)dx$$

may be viewed as a higher order function INT that takes as input, a function f, and the real numbers a and b, and returns as output, a real number, i.e.

$$INT(f, a, b) = f^*(b) - f^*(a)$$

where f^* is the integral of f. In the special case when $f(x) = (3x^2 + 4x + 5)$,

$$INT(f, a, b) = (b^3 + 2b^2 + 5b) - (a^3 + 2a^2 + 5a).$$

Definition 2 A *constraint domain* Σ is a triple (D, F, R) where:

- D is a non-empty set called the "domain of discourse" and
- $F \subseteq \mathbf{Func}(D)$ and
- R is a set of binary relations on $D \cup \mathbf{Func}(D)$.

Intuitively, a constraint domain Σ specifies a set D representing the domain of discourse over which we are working. The set F is the set of functionals (over the domain D) that are of interest, and the set R of relations over the functionals represents the kinds of relations we are interested in.

For example, if $\mathbf{R}$ (the set of reals) is our domain of discourse, we may have a relation $r \in R$ that says that if $f, g : \mathbf{R} \rightarrow \mathbf{R}$, then

$$f\, r\, g \leftrightarrow (\forall x \in \mathbf{R}) f(x) \leq g(x).$$

In this case, for the pair (f, g) to be in the relation r, both f and g must be unary functions on $\mathbf{R}$ and they must satisfy the above condition of "belowness."

As another example, we may consider equality as our relation, and express differential equations such as:

$$3\frac{dy}{dx} + 4 = x.$$

Here, just as integrals were considered to be functionals, differential operators may also be regarded as functionals.

The reader will notice that according to this definition, a constraint domain is a very general structure. This is indeed the case, and it was proved in [8] that many useful structures such as quadtrees, R-trees, relational databases, object oriented databases, etc. can be viewed as constraint domains.

Given a constraint model $\Sigma = (D, F, R)$, we associate with each element $d \in D$, a symbol d_s. With each $f \in F$, we associate a symbol f_s, and with each relation $r \in R$, a symbol r_s.

Definition 3 Given a constraint model $\Sigma = (D, F, R)$, we may define an atomic constraint as follows: if $r \in R$ is a relation, and α, β are in $\mathbf{Func}(S)$, then $(\alpha_s\, r_s\, \beta_s)$ is an *atomic* constraint. A *constraint* is defined as follows:

- Every atomic constraint is a constraint.
- If C is a constraint, then $\neg C$ is a constraint.
- If C, D are constraints, then $(C \,\&\, D)$ and $(C \vee D)$ are constraints.
- Nothing else is a constraint.

Definition 4 Given a constraint model $\Sigma = (D, F, R)$, and a constraint C, we may define the *satisfaction* of C by Σ, denoted $\Sigma \rhd C$, as follows:

- If C is the atomic constraint $(\alpha_s\, r_s\, \beta_s)$, then $\Sigma \rhd C$ iff $(\alpha\, r\, \beta)$, i.e. the pair $(\alpha, \beta) \in r$.
- If C is the constraint $\neg D$, then $\Sigma \rhd C$ iff it is not the case that $\Sigma \rhd D$.
- If C is the constraint $(D \,\&\, E)$, then $\Sigma \rhd C$ iff $\Sigma \rhd D$ and $\Sigma \rhd E$.
- C is the constraint $(D \vee E)$, then $\Sigma \rhd C$ iff $\Sigma \rhd D$ or $\Sigma \rhd E$.

Thus, for every constraint C, and any constraint model $\Sigma = (D, F, R)$, $\Sigma \rhd C$ or $\Sigma \rhd \neg C$.

2.3.2 Hybrid Knowledge Bases

An *annotation* is a pair $[u, t]$ where u is a term ranging over the unit interval (i.e. either a real number in the unit interval, a variable ranging over the unit interval, or a complex term consisting of a unit-interval valued function applied to sub-terms that range over the unit interval) and t is a term ranging over *sets* of non-negative real numbers (i.e. t is either a set of non-negative real numbers, or t is a variable ranging over sets of non-negative real numbers, or t is a complex term consisting of a non-negative real-valued set-valued function applied to sub-terms of the same type). A natural ordering $\preceq$ on variable-free annotations is the pointwise ordering induced by $\leq$ and $\subseteq$. In other words,

$$[u, t] \preceq [u', t'] \text{ iff } u < u' \text{ and } t \subset t'.$$

Definition 5 If A is a usual atomic formula of predicate calculus (built out of ordinary variables, predicate symbols, and constant symbols) and $[u, t]$ is an annotation, then $A : [u, t]$ is an *annotated atom.* An annotated atom containing no occurrences of object variables is *ground.*

Intuitively, the annotated atom $A : [u, t]$ says that "A is true with certainty at least u at all time points in the set t." When $u = 1$ and $t = \mathbf{R}^+$, then we will simply write A instead of writing $A : [u, t]$.

Definition 6 A *constrained-clause* is a sentence of the form

$$A : [u_0, t_0] \leftarrow \Xi_1, \ldots, \Xi_m \parallel B_1 : [u_1, t_1] \,\&\, \ldots \,\&\, B_n : [u_n, t_n]$$

where $A, B_1, \ldots, B_n$ are atoms of the language L, Ξ_i is a constraint over Σ_i, and

$$A : [u_0, t_0] \leftarrow B_1 : [u_1, t_1] \,\&\, \ldots \,\&\, B_n : [u_n, t_n]$$

is an annotated clause. Ξ is called the *constraint part* of the above clause, and $A : [u_0, t_0] \leftarrow B_1 : [u_1, t_1] \,\&\, \ldots \,\&\, B_n : [u_n, t_n]$ is called the *annotated clause part* of the above formula.

2.3.3 Using Hybrid KBs to Generate the Snapshot at Time $t > 0$

The main purpose for which the hybrid knowledge base will be used is to determine, based upon changes in the trajectory of the system, what the new orientation of the missile ought to be – in particular, the hybrid KB must specify which of the available control actions should be applied.

We assume that there is a predicate, called change that specifies the change in the trajectory. Thus, at any given point in time, a fact of the form change(−) is added as an update to the materialized view, and a set of actions must be generated by the hybrid knowledge base. Observe that at time t, we must compute:

- What controls to apply, and
- How long these controls must be applied for.

2.4 The View Maintenance Component

For the purposes of this paper, a *view* is just a hybrid knowledge base. *Materialization* of a view (i.e. of a hybrid knowledge base) refers to the task of computing, and storing, parts of the unique least Herbrand model of the hybrid KB. As all hybrid KBs are just sets of constrained clauses, which are negation-free ([8] also studies the case when nonmonotonic modes of negation are present), such a unique least model is guaranteed to exist by results in [8]. Index structures can be built on the materialized view. Consequently, database accesses to materialized view tuples is much faster than by recomputing the view. Materialized views are especially useful for providing intelligent support to real-time control systems for the following reasons:

1. at time t, determining the current trajectory is a constant time retrieval operation,
2. the new trajectory at time $t+1$ can be viewed as an *update* to the view, saying that the atom Traj($\langle new \rangle$) should be inserted into the view, and the atom Traj($\langle old \rangle$) should be deleted from the view.
3. Subsequently, using incremental view maintenance techniques such as those described by Gupta, Mumick and Subrahmanian [3], these updates can be easily incorporated into the materialized view.

2.5 The Truth Maintenance Component

The primary reason for using view maintenance algorithms in real-time is that they have much better computational properties than truth maintenance algorithms. The view maintenance algorithms in [3] all have linear-time data complexity ; however, even for definite logic programs, truth maintenance is known to be NP-hard. Even though specific types of instances of NP-hard problems can, and often are, solved efficiently, it turns out (cf. [3]) that view maintenance is always computationally easier than truth maintenance. The reason for this is that if $T \models A$ (i.e. a set T of formulas has A as a logical consequence), and we wish to update T by asserting $\neg A$, then truth maintenance systems attempt to do two things: (1) prevent the derivability of A from T, and (2) attempt to establish which formulas that were provable from T are no longer provable from T (based on A being "false" as a result of the update). View maintenance algorithms only perform (2), and do not account for (1).

Our architecture separates truth maintenance and view maintenance into two phases. When real-time performance is desired and time is at a premium, view maintenance is performed; when additional time is available to analyze the cause of discrepancies between sensor information and the materialized view, then the hybrid knowledge base can be changed so as to ensure consistency with the materialized view; that is, truth maintenance is performed off-line (or when slack-time is available on the processors), view maintenance is performed in real-time. Hybrid Knowledge Systems present an architecture that supports intelligent real-time reasoning. In short, the HKS architecture shows how view maintenance techniques such as those of Gupta, Mumick and Subrahmanian [3], view materialization techniques such as those of Bell, Nerode, Ng and Subrahmanian [1], truth maintenance techniques, and efficient database mediation techniques [9, 8], and specification of control laws such as those of Brockett [2] and Kohn and Nerode [7].

3 Conclusions

In this paper, we have described the notion of a hybrid knowledge system, and shown how the HKS architecture can be used to support and seamlessly integrate the modes of computation required to provide intelligent support to real-time systems such as control systems. Complex reasoning systems of this kind need to be able to reason with multiple representations of data, knowledge, and reasoning paradigms. They must also have a facility whereby different models of control (e.g. [2, 7]) may be incorporated. The HKS paradigm provides the expressive power and facilities required for this purpose through the mechanism of hybrid knowledge bases [8, 10]. In addition to performing such modes of reasoning, real-time performance is also required of such systems in the presence of dynamic changes to the external world. We have shown how view maintenance algorithms in databases can be used to elegantly capture these phenomena.

We are currently developing two applications of HKSs to real-time control systems – one is in mobile robotics [4] at NIST, and the other is in missile control.

References

[1] C. Bell, A. Nerode, R. Ng and V.S. Subrahmanian. (1991) *Mixed Integer Programming Methods for Computing Nonmonotonic Deductive Databases*, to appear in: Journal of the ACM.

[2] R. Brockett. (1993) *Hybrid Models for Motion Control Systems*, to appear in: Perspectives in Control (eds. H.L. Trentelman and J.C. Willems), Birkhauser.

[3] A. Gupta, I. S. Mumick and V.S. Subrahmanian. (1993) *Maintaining Views Incrementally*, Proc. 1993 ACM SIGMOD Intl. Conf. on Management of Data, pps 157–166.

[4] J. Horst, E. W. Kent, H. Rifky and V.S. Subrahmanian. (1993) *Intelligent, Real-Time Robotic Reasoning with Hybrid Knowledge Systems*, draft manuscript.

[5] M. Kifer and V.S. Subrahmanian. (1989) *Theory of Generalized Annotated Logic Programming and its Applications*, Proc. 1989 North American Conf. on Logic Programming, MIT Press.

[6] M. Kifer and V. S. Subrahmanian. (1992) *Theory of Generalized Annotated Logic Programming and its Applications*, Journal of Logic Programming, Vol. 12, 4, pps 335–368, 1992.

[7] W. Kohn and A. Nerode. (1992) *An Autonomous Systems Control Theory: An Overview*, Invited Address. Proc. 1992 IEEE Symp. on Computer-Aided Control Systems Design, pps 204–210.

[8] J. Lu, A. Nerode, J. Remmel and V.S. Subrahmanian. (1993) *Towards a Theory of Hybrid Knowledge Bases*, Univ. of Maryland CS-TR-3037. Submitted for publication.

[9] V.S. Subrahmanian. (1992) *Amalgamating Knowledge Bases*, Univ. of Maryland CS-TR-2949, August 1992. Submitted to ACM Trans. on Database Systems, Aug. 1992. Revised May 1993.

[10] V.S. Subrahmanian. (1993) *Hybrid Knowledge Bases for Intelligent Reasoning Systems*, Invited Address, Proc. 8th Italian Conf. on Logic Programming (ed. D. Sacca), pps 3–17, Gizzeria, Italy, June 1993.