

Declarative Hierarchical Controllers

Wolf Kohn

Boeing Computer Services
Scientific Computing and Analysis
Seattle, WA 98124-0346

1 Introduction

This paper presents a new general purpose feedback controller for driving a system through complex tasks. The proposed controller, termed Declarative Hierarchical Controller, is based on a theory of knowledge-based controllers developed by the author [1, 2, 3]. In this theory, the plant dynamics, control requirements, and goal dynamics are declared in an axiom base. The actuator commands, which are functions of the sensor and goal command signals, are generated on-line as side effects of showing whether a theorem (Task Theorem) representing the system task logically follows from the equational axiom base. That is, the controller is an on-line mechanical theorem prover whose inference mechanism is based on equation-solving over a Variety. The Task Theorem is constructed by an on-line planner as a conjunction of primitive lemmas, each of which is a carrier of an elementary control action. The control actions are multiplexed in time so that at each instance one and only one is active.

The theorem prover of the controller operates as follows:

- At each controller sample interval, the Task Theorem and the corresponding active axioms generate a set of simultaneous equations in which the variables are Actuator commands and the Controller state. This set is referred to as the Active Set.
- The Active Set is used by the inferencer to build a procedure for computing instance values of the variables. This procedure is a locally finite automaton over the Rational variety.
- The automaton is executed to compute instances of the commanded actions.

In addition, the theory provides an algorithm for transforming the axiom base and the inferencer into a recursive hierarchy for its efficient implementation and for satisfying real time and architectural constraints.

The paper illustrates the theory of Declarative Hierarchical Controllers with a robot manipulator control under End Effector force constraints. It also gives

some stability, robustness and complexity results. The general controller capabilities are illustrated with a particular robot: A planar 3-link manipulator robot controller for painting the inside surface of a 2-dimensional balloon with deforms elastically under contact force. The task will also be constrained by End Effector angular velocity and position constraints (over-determined problem). The problem has no know solutions using conventional control and planning schemes.

The rest of the paper is organized into 5 sections: Section 2 presents an overview of knowledge-based controllers, Section 3 presents declarative controllers, Section 4 discusses the main elements of multiplexing action in the context of declarative controllers, Section 5 illustrates the concept with an example, and in Section 6 some conclusions are established.

2 Overview of Knowledge-Based Controllers

Declarative Rational Controllers are a class of knowledge-based feedback digital controllers.

As for conventional controllers, the function of a declarative controller is to generate an action (e.g., actuator command) as a function of sensor data, stored data and goal command data. In symbols, let T be the set of natural numbers, called the time set. Let G, S, X, A be semimodules over a common semiring H . A controller is fully characterized by two functions ρ, Ψ :

$$\begin{aligned} \rho &: G \times S \times X \times T \rightarrow A \\ \Psi &: G \times S \times X \times A \times T \rightarrow X \end{aligned} \tag{1}$$

called the control law [4].

The semimodules G, S, X, A are referred to respectively as the goal space, the sensor space, the internal storage space and the action space.

The control law (ρ, Ψ) is determined by the requirements, the characteristics of the systems to be controlled (the plant) and the characteristics of the goals the system is supposed to achieve.

In conventional controllers the control law (ρ, Ψ) is explicitly implemented as an algorithm for generating actions as a function of time. Specifically, let $s(k)$, $g(k)$, and $x(k)$ be the sensor data, goal data, and internal storage data at the current sample time $k \in T$. Then the control action $a(k + 1)$ and the updated internal storage $x(k + 1)$ are given by

$$\begin{aligned} a(k + 1) &= \rho(g(k), s(k), x(k), k) \\ x(k + 1) &= \Psi(g(k), s(k), x(k), a(k), k) \end{aligned} \tag{2}$$

In knowledge-based controllers in general, and in declarative controllers in particular, the control law is not constructed explicitly. Instead a knowledge base containing the requirements, plant dynamics representation and goal characteristics is constructed. At run time the controller generates the action signals $\{a(k) \mid k = 0, 1, \dots\}$ by an inference procedure [5] which operates on the knowledge base at each sample time to find an instantiation of the control action to be implemented in the next sample time.

Some of the potential advantages of declarative controllers over conventional ones are [6]: simplified design process, multiplicity of control laws, built-in adaptability, hooks and scars and dependency on computational characteristics. These are briefly discussed next.

Simplified Design Process: In principle, given an inferencing shell and a compatible specification for knowledge representation, the design process consists of collecting the rules or frames that capture the control and computational requirements and those aspects of the plant and goal that are relevant to the design and the encoding of them. The actual “algorithm” implementing the control is inferred from the knowledge base at run time.

Multiplicity of Control Laws: Since the control requirements are explicitly declared in the knowledge base, the controller may have more than one control law if knowledge is encoded so that it can select which one to execute as a function of external (goal) commands.

Built-in Adaptability: Parameters or structural characteristics of the plant can be easily declared in generic form and appropriate instances of them can be generated as a function of sensory data at run time [7].

Hooks and Scars: Typically the knowledge base of a knowledge base controller is composed of independent discrete elements (clauses, rules, frames, etc.) which are only connected at run time during inference. Therefore adding or deleting elements does not destroy the logic of the controller.

Dependency on Computational Characteristics: The knowledge base may include characteristics of the architecture in which the controller runs. These characteristics are treated in the same way as control requirements.

On the other hand, general knowledge-based controllers suffer from three serious drawbacks which have dampened their popularity among control designers. These are stability verification, performance verification, and computational complexity. These are briefly discussed next.

Stability Verification: This is understood as a test on the controller specification to determine whether it will drive the system to the current desired goal while satisfying the control requirements in a finite interval of time. In most of the knowledge-based control schemes that have been proposed, no such test is available.

Performance Verification: Since the control law function is not explicitly given, it is very difficult and in most cases practically impossible to guarantee a performance level for all the possible inference paths.

Computational Complexity: Many knowledge-based control systems are implemented on expert system shells which are not well-suited for real-time implementation. The theory of declarative hierarchical controllers was developed to address some of the special requirements for robot control systems. Their functionality possesses the good characteristics of knowledge-based controllers discussed previously. In addition, the theory provides for effective stability and performance tests and feasibility tests for (with current hardware) computational requirements.

The central objective of knowledge-based controllers in general, and of Declarative Hierarchical Controllers in particular, is the generation of *autonomous*

Figure 1: Declarative Rational Controller

feedback policies.

Autonomous feedback policies are trajectories of control actions and control action generators that are causally measurable in the Σ -algebra defined by the sensory data. This means that the control law is not completely pre-specified, as in conventional control systems, but rather it is inferred locally at run time as a function of the environment, the stated goal and the control specifications.

The degree of autonomy is a quantitative measure of the amount of knowledge that is generated at run time to complete a local instance of the control law. This measure is the central discriminator between conventional control systems and knowledge-based control systems.

A descriptive overview of declarative rational controllers is given in the next section.

3 Declarative Rational Controllers

This section presents an overview of Declarative Hierarchical Controllers. Space constraints prevent the detailed discussion of the mathematical basis behind some of their characteristics. These can be found in some of the references [8, 9].

The structure of a Declarative Hierarchical Controller is illustrated in Figure 1. It is composed of four functional elements: a *Knowledge Base*, an *Inference Mechanism*, a *Theorem Planner* and an *Adapter*. In the next paragraphs those elements are briefly described.

Knowledge Base: This element consists of a set of equational, first-order logic equational clauses [10] with some extension whose characteristics are described next.

A clause in the knowledge base of a declarative rational controller with sensor semimodule S , control action semimodule A , goal semimodule G , and internal

storage semimodule X and time set T , is one or more Horn clauses of the form

$$P_i(\underline{x}(t)) \Leftarrow e_1^i(\underline{x}(t), \underline{y}(t)) \wedge \dots \wedge e_s^i(\underline{x}(t), \underline{y}(t)) \quad (3)$$

where \Leftarrow is the logical implication, \wedge is the logical And, and $x(t), y(t)$ are sets of mappings of the form: Let $U = G \times S \times X \times A$,

$$\begin{aligned} \underline{x}(t) &= (x_1(t), \dots, x_n(t)) \\ \underline{y}(t) &= (y_1(t), \dots, y_m(t)) \end{aligned}$$

with

$$\begin{aligned} x_l &\in U & l &= 1, \dots, n \\ y_l &\in X & l &= 1, \dots, m \end{aligned} \quad (4)$$

The $\{x_l\}$ are called the external variables and the $\{y_l\}$ are called the internal variables.

In (3), P_i is termed the clause head. The logical interpretation of (3) is $P_i(\dots)$ is true if $e_1^i(\dots)$ is true and \dots $e_s^i(\dots)$ is true. The $e_j^i(\dots)$ are referred to as the terms of the body of the i th clause.

Each term $e_j^i(\underline{x}(t), \underline{y}(t))$ is exactly one of 5 possible forms:

- a) an equational term
- b) an inequational term
- c) a partial order term
- d) a clause name
- e) a frozen term

- a) An equational term is an expression of the form

$$w(\underline{x}, \underline{y}) \approx v(\underline{x}, \underline{y}) \quad (5)$$

where w and v are polynomic terms associated in an algebraic variety V_R defining the domain of the control system.

In an equational term, the variables and parameter sequences are subsequences of variables and parameters, respectively, the sequences appearing in the associated clause head.

The semantics of an equational term e_{name}^i is a subset E_{name}^i of a cartesian product of the universe U in which the variables and parameters take values. In symbols this subset can be expressed as

$$E_{\text{name}}^i(\underline{y}) = \{\underline{x} \mid \text{length}(\underline{x}) = n, \underline{x} \in U^n, w(\underline{x}, \underline{y}) \approx v(\underline{x}, \underline{y})\}$$

where length is a function that computes sequence length. Note that the set E_{name}^i is a function of the parameter sequence \underline{y} . This set is a rational set.

b) An inequational term is an expression of the form

$$w(\underline{x}, \underline{y}) \not\approx v(\underline{x}, \underline{y}) \quad (6)$$

where w , v , \underline{x} , and \underline{y} are defined as in (5). The semantics of an inequational term e_{name}^i is a subset E_{name}^i of a cartesian product of the universe U defined as follows: Let D_{name}^i be the following set:

$$D_{\text{name}}^i(\underline{y}) = \{\underline{x} \mid \text{length}(\underline{x}) = n, \underline{x} \in U^n, w(\underline{x}, \underline{y}) \approx v(\underline{x}, \underline{y})\} \quad (7)$$

Then $E_{\text{name}}^i(\underline{y})$ is defined as the compliment of $D_{\text{name}}^i(\underline{y})$ with respect to U^N .

$$E_{\text{name}}^i(\underline{y}) = U^N - D_{\text{name}}^i(\underline{y}) \quad (8)$$

This set is rational.

c) A partial order term is an expression of the form

$$w(\underline{x}, \underline{y}) \leq_{\alpha} v(\underline{x}, \underline{y}) \quad (9)$$

where \leq_{α} is a partial order over one or more of the polynomial algebras associated with the algebras in the rational variety.

The semantics of a term of the form of (9) is the relation that is the set $E_{\text{name}}^i(\underline{y})$ defined by

$$E_{\text{name}}^i(\underline{y}) = \{\underline{x} \mid \text{length}(\underline{x}) = n, w(\underline{x}, \underline{y}) \leq_{\alpha} v(\underline{x}, \underline{y})\} \quad (10)$$

If the lattice L_{α} corresponding to the partial order \leq_{α} is modular, the set defined by (10) is rational.

d) A clause name is the head of a clause in the knowledge base. Its semantics is the intersection of the rational sets of the terms in its body. This includes recursive clauses. For example, a clause of the form

$$p(x, y) \Leftarrow w_1(x, y) \approx v_1(x, y) \wedge w_2(x, y) \approx v_2 \wedge p_3(x, y)$$

has semantics given by the set

$$E_p(y) = E_p^1(y) \cap E_p^2(y) \cap E_p^3(x, y)$$

where the semantics of the third term is the rational set associated with a clause.

e) A frozen term is an expression of the form

$$\text{rat}(\text{relation}, \text{polynomial}_1, \text{polynomial}_2, \text{variables}, \text{parameters}) \quad (11)$$

where *relation* is either \approx or $\not\approx$ or \leq_{α} , and *polynomial*₁ and *polynomial*₂ are polyomic expressions.

A frozen term is a term associated with a clause in which w is activated, i.e., unfrozen, as a consequence of the instantiation of another clause during an inference process. Any term in a clause can be frozen during inference. The purpose of this capability is to allow for *context* and *covering* of alternative semantics, i.e., rational sets for a concept.

The terms w, v in (5) and (6) are polynomials [10] over an algebra $B = \langle U, \Omega \rangle$ where U , the universe of the algebra, is given in (4), and Ω is the set of primitive operations:

$$\Omega = \{+, \cdot, 0, 1, [f_r \mid r \in R]\}. \quad (12)$$

These operations satisfy the following axioms:

- (i) $(U, +, 0)$ is a commutative monoid with unit 0
- (ii) $(U, \cdot, 1)$ is a monoid with unit 1
- (iii) For all a, b, c in U ,

$$\begin{aligned} (a + b) \cdot c &= a \cdot c + b \cdot c \\ a \cdot (b + c) &= a \cdot b + a \cdot c \end{aligned}$$

That is, $\langle U, +, \cdot, 0, 1 \rangle$ is a semiring.

- (iv) The set $[f_r \mid r \in R]$ is a set of unary operations of the algebra, referred to as the *Custom Operators*. Their axioms and computation values are determined for *each controller* by the clauses in the knowledge base. This means that each declarative controller C has associated with it a unique algebra $B_C = \langle U_C, \Omega_C \rangle$. The custom operators are the basis for the local construction of the control law under composition.
- (v) The algebra $\langle U, \Omega \rangle$ satisfies the central factorization principle. This principle states that a term w constructed by composition from primitives in Ω is either a primitive or else can be expressed in finitely many different ways in terms of derived operations of algebra on some of its elements.

The denotational semantics of the clauses defined in (3) are one of the following:

- 1) A conservation principle, or
- 2) An invariance principle, or
- 3) A control constraint.

Conservation principles are logic statements about dynamic behavior of the controller, associated plant or goal. These principles serve analogous roles to the ones played by mass momentum and energy conservation principles in mechanics and thermodynamics.

Invariance Principles are logic statements establishing constants of motion in a general sense. Examples include logic formulations of stationarity principles and geodesics.

Both conservation and invariance principles are characterized by equations or inequations valid in the controller algebra and therefore can be written as clauses of the form of (3).

Figure 2: Clause Knowledge Base of a Declarative Rational Controller

Finally, control constraints include actuator and sensor limitations and the control requirements. These can either be written in equational form or else written in terms of more general Horn clause forms. If a clause is not in the form of (1) it can be transferred to a set of clauses of that form using Colmerauer's construction which is compatible with the algebraic structure of elements of V .

The clause database is organized in a nested hierarchical structure as illustrated in Figure 2. The bottom of the hierarchy contains the equations that characterize the variety V , termed Laws of the Variety.

At the next level of the hierarchy, three types of clauses are stored: Generic Control Specifications, Plant Representation and Goal Class Representation.

The generic control specifications are clauses expressing general desired behavior of the system. They include statements about stability, complexity and robustness that are generic to the class of declarative rational controllers. These specifications are written by constructing clauses that combine laws of the variety using the Horn clause format described earlier.

The Plant Representation is given by clauses characterizing the dynamic behavior and structure of the plant, which include sensors and actuators. These clauses are written as conservation principles for the dynamic behavior and as invariance principles for the structure. As for the generic control specifications, they are constructed by combining a variety of laws in the equational Horn clause format.

The next level of the hierarchy involves the Control Performance Specifications. These are typically problem-dependent criteria and constraints. They are written either in equational Horn clause format or in rational tree format [4] which can be translated into this equational form.

Dynamic control specifications are equational Horn clauses whose bodies are modified as a function of the sensor and goal commands. (See Figure 1.)

Finally model builder realization clauses constitute a recipe for building a procedural model for variable instantiation and theorem proving. They serve as an interface to the inferencer whose operation will be discussed next.

Inferencer: This is an on-line equational theorem prover. The class of theo-

rems it can prove are represented by clauses of the form

$$\begin{aligned} \text{Theorem}(G(t), s(t), \underline{x}(t), A(t+1), \underline{x}(t+1)) \Leftarrow & \text{unify}(G(t), s(t), \underline{x}(t), \underline{y}(t)) \wedge \\ & \text{unify}(A(t+1), \underline{x}(t+1), \underline{z}(t+1)) \wedge \bigwedge_{k=1}^s P_{i_k}(\underline{z}(t+1), \underline{y}(t)) \quad (13) \end{aligned}$$

In expression (13), theorem is the clause head, t is the current time, $G(t)$, $S(t)$, and $A(t+1)$ correspond to the variables representing the goal command to the controller ($G(t)$) the sensor inputs ($S(t)$) and the actuator commands ($A(t+1)$). Note that the actuator commands to be generated are one unit of time ahead of current time.

In the right hand side of (13), unify is a special factual clause head whose function is to unify some of the external variables $X(t)$ with the input output variable $G(t)$, $S(t)$, $A(t+1)$, and $P_{i_j} \dots P_{i_k}$ are clause heads of clauses in the knowledge base.

The theorem represents the desired behavior at the current update time. The purpose of the inferencer is to determine whether the theorem logically follows from the clauses in the knowledge base. A side effect is to find values in the universe of the controller algebra for tuples of the form $(G(t), S(t), x(t), x(t+1), A(t+1))$ where $G(t), S(t), X(t)$ are given.

Note that the theorem is an encoding of a system of equations and inequations from the knowledge base. So proving the theorem is equivalent to solving this system.

The inference principle can be stated as follows: Let Σ be the set of clauses in the knowledge base. Let \neq represent one or more partial orders in the universe of the controller algebra. Then proving the theorem is equivalent to showing

$$\Sigma \vdash L \quad (14)$$

In principle, the proof can be accomplished by a sequence of applications of the following axioms:

1. Equality axioms

$$w \approx w \quad \text{identity} \quad (\text{a1})$$

$$\frac{w \approx v}{v \approx w} \quad \text{commutativity} \quad (\text{a2})$$

$$\frac{(w \approx v) \wedge (v \approx u)}{w \approx u} \quad \text{transitivity} \quad (\text{a3})$$

$$\frac{h \text{ an } n\text{-ary operation} \wedge w_1 \approx v_1 \wedge \dots \wedge w_n \approx v_n}{h(w_1, \dots, w_n) \approx h(v_1, \dots, v_n)} \quad \text{composition} \quad (\text{a4})$$

$$\frac{\rho \text{ be a substitution} \wedge w \approx v}{w\rho \approx v\rho} \quad (\text{a5})$$

$$\text{The equality clauses in the knowledge base are valid.} \quad (\text{a6})$$

for $w_1, \dots, w_n, v_1, \dots, v_n$ polynomic terms. Briefly, a1 establishes that every polynomic form is equal to itself, a2 and a3 are self-explanatory, and a4 says that if h is any arbitrary derived n -ary operation in the associated algebra, then the equality of n pairs of terms is preserved by the operation. In a5 the symbol ρ , termed a substitution, is an equation of the form

$$x \approx u(\underline{z}, \underline{y}) \quad (15)$$

where x is a variable, \underline{z} is a sequence of variables *not* containing x , and \underline{y} is a sequence of parameters.

As an example of the operation of the substitution inference principle, suppose that

$$w(x_1, x_2) \approx v(x_1, x_2, x_3, y)$$

and let ρ be of the form

$$x_1 \approx u(x_5, y')$$

Then upon the application of the substitution principle,

$$w(u(x_5, y'), x_2) \approx v(u(x_5, y'), x_2, x_3, y)$$

which is denoted by

$$w\rho \approx v\rho$$

Finally, a6 establishes that there is no conflict between the knowledge base clauses and the inference principle of equality.

2. Inequation axioms

Let v and w be polynomic terms in an FPS algebra. The inequation inference principles are:

$$\frac{v \not\approx w}{w \not\approx v} \quad \text{symmetry} \quad (b1)$$

$$\frac{v \not\approx w}{\frac{v \approx w}{set} \wedge \text{complement}(set, U, set^1) \wedge \frac{set^1}{v^1 \approx w^1}} \quad \text{definition} \quad (b2)$$

$$\text{The inequality terms in the clauses of the knowledge base are satisfied.} \quad (b3)$$

b1 is clear. b2 says that if $v \not\approx w$ entails that the rational set associated with $v \not\approx w$ is set and the complement of set relative to the universe U of the semimodule is set^1 the semantics of the term $v^1 \approx w^1$.

In b2 one exploits the fact that rational sets are both mappings and sets and furthermore, if a set is rational its complement is also rational and therefore is generated by a polynomic equation, which is a member of the semimodule.

3. Partial order axioms

Let w , v and u be polynomic terms in an FPS algebra B . For each partial order \leq_α defined in B , the partial order inference principles are

$$w \leq_\alpha w \quad \text{identity} \quad (\text{c1})$$

$$\frac{w \leq_\alpha v \wedge v \leq_\alpha u}{w \leq_\alpha u} \quad \text{transitivity} \quad (\text{c2})$$

$$\frac{h \text{ an } n\text{-ary operation} \wedge h \text{ monotonic} \wedge w_1 \leq_\alpha v_1 \wedge \dots \wedge w_n \leq_\alpha v_n}{h(w_1 \dots w_n) \leq_\alpha h(v_1 \dots v_n)} \quad \text{composition} \quad (\text{c3})$$

$$\frac{\text{rho a substitution} \wedge w \leq_\alpha v}{w\rho \leq_\alpha v\rho} \quad \text{substitution} \quad (\text{c4})$$

The partial order terms in the clauses of the knowledge base are valid. (c5)

In addition to c1–c5, the following compatibility principle is given:

$$\frac{w \leq_\alpha v \wedge v \leq_\alpha w}{w \leq_\alpha v} \quad \text{compatibility} \quad (\text{ac1})$$

The meaning of these inference principles is similar to the corresponding principles for equality.

4. Convergence principles

The convergence principles are formal inference principles derived from the defining axioms of convergent sequences discussed in section 2.1. These principles, together with the limit principles, are to be given in e).

Both the convergence and limit principles are needed to determine solutions of equational systems in the presence of recursion.

Let B be a rational algebra, B^N the associated algebra of sequences with values in B , and C^N the set of convergence sequences that can be inductively built from $z(n)$ and $\bigwedge(n)$ using the following inference principles:

$$z(n) \wedge \bigwedge(n) \in C^N \quad \text{initiality} \quad (\text{d1})$$

$$\frac{a \in C^N \wedge b \in C^N}{(a +_s b) \in C^N} \quad \text{additive closure} \quad (\text{d2})$$

$$\frac{a \in C^N \wedge c \in B}{c \cdot a \in C^N \wedge a \cdot c \in C^N} \quad \text{scalar closure} \quad (\text{d3})$$

$$\frac{a \in C^N \wedge c \in B}{a_c \in C^N} \quad \text{shift closure} \quad (\text{d4})$$

where $z(n) = 0 \forall n$ and $n(n) = 1 \forall n$.

5. Limit axioms

These are given by the following inference clauses.

$$\lim(z) \approx 0 \wedge \lim(\wedge) \approx 1 \quad \text{initiality} \quad (\text{e1})$$

$$\frac{a \in C^N \wedge b \in C^N \wedge \lim a \approx A \wedge \lim b \approx B}{\lim(a +_s b) \approx A +_s B} \quad \text{additive preservation} \quad (\text{e2})$$

$$\frac{a \in C^N \wedge c \in B \wedge \lim a \approx A}{\lim(c \cdot a) \approx c \cdot a \wedge \lim(a \cdot c) \approx A \cdot C} \quad \begin{array}{l} \text{scalar multiplication} \\ \text{preservation} \end{array} \quad (\text{e3})$$

$$\frac{a \in C^N \wedge c \in B \wedge \lim a \approx A}{\lim(a_c) \approx A} \quad \text{shift preservation} \quad (\text{e4})$$

Note that in d) and e) the inference clauses include membership terms. These terms can be written as equational terms by including in the custom operations of the associated base algebra *Indicator* operations.

Let $a \in U$ be the universe of the base algebra an indicator operation f_a on the base algebra is a function $f_a : U \rightarrow U$ defined as follows:

$$f_a(x) = \begin{cases} 1 & a \approx x \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

Indicator operations on the associated rational algebra are defined in terms of indicator operations on the associated base algebra componentwise.

Because rational sets are subadditive, each rational set can be decomposed as the union (addition) of singletons that is in fact the structure of the semantics of the elements in the associated semimodule. Therefore, membership can be reduced to equational definitions of the form of (16).

The following theorem summarizes the inference procedure.

Theorem: Given a controller algebra B and the corresponding knowledge base Σ , then if L is the system associated with the goal theorem, then there exists an effective procedure for showing

$$\Sigma \vdash L$$

Outline of the proof: Since B satisfies the central factorization principle, by structural induction, a finite chain of applications of the axioms 1-, 2-, 3- will yield a proof.

The theorem above is an extension to rational algebras of a result for finite algebras due to Evans [11].

Although the theorem guarantees convergence of the proof, it is a nondeterministic, highly inefficient procedure. Therefore, for practical reasons, a more efficient but equivalent procedure has been developed [12]. This approach, which involves construction of a procedure for solving the system L , is discussed next.

The inferencer operates according to the following procedure.

- Step 1. Unify the appropriate subset of the external variables $\underline{y}(t)$ with the current value of $G(t)$, $S(t)$ and $\underline{x}(t)$.

- Step 2. Search the database for the equations and inequations associated with clauses $P_{i_j} \dots P_{i_k}$. The result of this step is a system of simultaneous equations and inequations in terms of the unknowns, the subset of $X(t)$ unified with $A(t+1)$. This system is referred to as the Active Set (A_S).
- Step 3. Convert the active set into a system of equations in the canonical form. This system is referred to as the Linear Set (L_S).

The object of step 3 is to rewrite the Active Set in a form which allows the construction of a procedural model which in turn can be used to compute instance values of the unknown. This procedural model is a Locally Finite Automaton. The canonical form of L_S is given by a system of equations of the form

$$\underline{X}(t) = \underline{E}(\underline{Y}(t))\underline{X}(t) + \underline{T}(\underline{Y}(t)) \quad (17)$$

where $\underline{E}(\cdot)$ is a matrix of appropriate dimensions with entries in the algebra of formal power series over the controller Algebra B : $B\langle M \rangle$, M is the locally finite monoid generated by the set of primitive custom unary operations in B , Φ_R , and $\underline{T}(\cdot)$ is a vector whose entries are elements of $U\langle M \rangle$.

- Step 4. Given the linear set (L_S), construct a Locally Finite Automaton (LFA) for solving it.
- Step 5. Execute the LFA to obtain the values, if any, for $A(t+1)$.

If the theorem (13) logically follows from the knowledge base (i.e., it is true), the inferencer procedure outlined will terminate on step 5 with a command value $A(t+1)$. If this is not the case, then the adapter is activated and the theorem is modified by the theorem planner according to a prespecified strategy.

Before proceeding to discuss the theorem planner, a brief outline of the theory behind steps 3–5 of the inferencer will be presented. This theory is the central element of the paper.

In general, given a knowledge base of Horn clauses and a goal, the process of proving the goal and finding instances of its variables can be carried out by a procedure, developed by Kowalski, known as resolution [13]. The Resolution Procedure is based on constructing a tree (the Resolution tree) with the goal as its root and with the appropriate predicates of the knowledge base at its branches.

The resolution procedure consists of two subprocedures, a Navigator and a Unifier. The Navigator is a strategy for traversing the tree. Most systems currently available use depth-first with backtracking as a strategy [14]. The unifier, usually a variant of Robinson's unification principle [15], is a general pattern-matching recursive algorithm, which, given a set of clause heads, generates the Most General Unifier (MGU) associated with them.

In contradistinction with the approach discussed above, the inferencer of a declarative controller builds a procedure for goal variable instantiation: a locally finite automaton.

A locally finite automaton is a non-deterministic machine with an arbitrary number of states (without loss of generality, it can be assumed that the number of its states is infinity) that satisfy the following conditions:

- a) There is a finite number of states that are initial states.
- b) There is a finite number of states that are terminal states.
- c) The behavior of the automation is the set of all paths from initial to terminal states (successful paths). This set is an element of a semimodule of formal power series over the Controller Algebra $B\langle M \rangle$.
- d) Every successful path involves a finite number of automaton edges. (This is the locally finite condition.)
- e) Every successful path represents a map of the semimodule space $G \times S \times A$ into itself, where G is the space of goals, S is the space of sensor signals and A is the space of actuator commands.
- f) The Automaton is provide with an input and an output function: The input function is of the form

$$I : G \times S \times X \rightarrow G \times S \times X \times A \quad (18)$$

$$I(g, s, x) = (g, s, x, \Phi)$$

where Φ is the additive identity in the semimodule A . The output function is of the form

$$O : G \times S \times A \rightarrow A \quad (19)$$

$$O(g, s, a) = a$$

That is, O is a projection function.

In synthesis, each successful path is a feasible control law.

The successful paths can be well ordered by any user-defined optimization criterion for selection of the actual control law to be executed.

Now the theorem planner is described. The theorem planner generates theorems of the form of (3) according to a prespecified strategy. A theorem remains in effect as long as it has truth value true. The theorem has truth value true if the system of equations Ls have at least one solution different from the empty set.

If the theorem is not true, the adapter (see Figure 1) activates the strategy procedure in the planner. This produces a modified theorem to be proved.

This concludes the description of the concept of declarative controllers. A more detailed description of this concept will appear in [16].

4 Multiplexing in Declarative Hierarchical Controllers

This section introduces the concept of multiplexing action in the context of declarative controllers and establishes its use for the control of robot manipulators. A detailed description of multiplexing action in the context of a robot application can be found in [17].

Let Δ be an interval in the nonnegative real line. Let $\Delta_1, \dots, \Delta_n$ be subintervals of Δ such that

$$\Delta = \Delta_1 \cup \dots \cup \Delta_n \quad (20)$$

A multiplexing action over Δ is a staircase function f_Δ over Δ taking values in the semimodule of control actions A such that

$$f_\Delta(t) = a_i \quad a_i \in A, t \in \Delta_i$$

A multiplexing action f over the nonnegative real line is a staircase function f_Δ over each interval Δ on it.

A feedback multiplexing action over Δ is a term of the form

$$O(w_i(g, s, x, a)) : I \times G \times S \times X \times A \rightarrow A \quad i = 1, \dots, n \quad (21)$$

where w_i is a polynomial of the controller algebra B , O is the output function of the controller automaton introduced in the previous section and I is the set $\{1, 2, \dots, n\}$.

A feedback multiplexing action over Δ with subintervals $\Delta_1, \dots, \Delta_n$ generates a multiplexing action $f_\Delta(t)$:

$$f_\Delta(t) = O(w_i(g, s, x, a_i)) = a_i \quad t \in \Delta_i, i = 1, \dots, n \quad (22)$$

A feedback multiplexing action over Δ is termed synchronous if the subintervals Δ_i , $i = 1, \dots, n$, are all equal. In this paper only synchronous multiplexing is considered.

Now the concept is particularized for robot manipulators.

Suppose that a manipulator has a DC motor at each of its joints. Each motor's armature voltage is given at every instant of time by exactly one of three possible controllers: a position controller, a rate controller or a force controller. Further suppose that the time line is divided into intervals of duration Δ and each Δ is further subdivided into 3 subintervals $\Delta_1, \Delta_2, \Delta_3$ at each Δ_i , $i = 1, 2, 3$, and for each joint only one of the 3 possible controllers' position, rate or force is active. Clearly this schema is a feedback multiplexing action over Δ if it is assumed that the three controllers for each joint are generated as functions of the form of (22).

Note that for an n -joint manipulator there are $3^{N+3} - 1$ different controllers. So the function of a declarative controller is to select for each Δ interval and for each joint the type of length -3 control sequence to be applied and then to determine the appropriate polynomial of the controller algebra for each element in the sequence.

Figure 3: Simplified Robot Geometry

As discussed in the previous section, this is accomplished by generalizing a locally finite automaton and simulating it to generate the values of the joint commands. The LFA at each Δ interval is a representation of the proof of the theorem characterizing the task to be accomplished by the manipulator in this time interval. This is illustrated with an example in the next section.

5 Example

The concept of multiplexing declarative controllers will be illustrated with a simple manipulator. The manipulator is a planar three-link chain with rotational joints and a rigidly attached effector (Figure 3). The central characteristics of the manipulator are:

- The links are assumed to be solid rigid long cylinders.
- The joints are driven by ideal permanent-magnet DC motors.
- Each joint is coupled to its driving motor by a “sloppy” gear box whose characteristics are shown in Figure 4. This gear box characteristic is similar to that in each joint of NASA’s Shuttle RMS.
- Each joint is equipped with position, rate and force sensors.
- A sensor, which detects the quadrant in which the end effector is with respect to base coordinates, is attached to the end effector.

The dynamics of the manipulator is represented in the knowledge base with the following Lagrangian conservation principle:

$$L = E - V - \sum_{i=0}^{3^{N+3}-1} V_i \quad (23)$$

where E is kinetic energy, V is the physical potential energy and V_i are virtual potentials representing the characteristics of the 3 types of controllers available at each joint in the multiplexing schema.

$$\begin{aligned}
\delta &= \text{Input} - N \times \text{Output} \\
N &= \text{gear ratio} \\
\text{Input} &= \text{motor rpm} \\
\text{Output} &= \text{Joint rpm} \\
\text{Output torque, } T &= \begin{cases} K1 \delta^2 \text{sign}(\delta) & |\delta| \leq \Delta \\ K2 \delta & |\delta| > \Delta \end{cases} \\
\text{sign}(\delta) &= \begin{cases} 1 & \delta \geq 0 \\ -1 & \delta < 0 \end{cases}
\end{aligned}$$

Figure 4: Joint Gearbox Characteristic

The maneuver to be accomplished by the robot is the following. The end effector is to follow a circular path exercising a constant force against it. The circular path deforms elastically under load. This is modeled as a preloaded spring distributed over the path. The stiffness of the spring is constant and uniform along the path. The base of the robot is not located at the center of the circular path.

In addition, tolerance for end effector position angular velocity and force are provided.

The controller also monitors the health of the manipulator with respect to failures such as joint runaways and tachometer failures and executes a crating maneuver if a failure is detected.

The theorem that characterizes this maneuver is given by

$$\begin{aligned}
\text{theorem}(\dots) &\Leftarrow \text{pos}(r, b, c, \epsilon_p) \wedge \\
&\quad \text{vel}(r, b, c, \epsilon_v) \wedge \\
&\quad \text{for}(r, b, K_{EE}, \epsilon_f) \wedge \\
&\quad \text{HMS}(\dots).
\end{aligned} \tag{24}$$

In (24) $\text{pos}(\dots)$ is the end effector position lemma, $\text{vel}(\dots)$ is the end effector velocity lemma, $\text{for}(\dots)$ is the end effector lemma, r is the radius of the circular path, b is the relative position of its center with respect to the base, c is the vector of joint command (multiplexing actions) and $\epsilon_p, \epsilon_v, \epsilon_f$ are the corresponding tolerances. $\text{HMS}(\dots)$ is the health monitoring lemma.

A multiplexing declarative controller for this robot was implemented in Quintus Prolog on a Sun 3-160 workstation. Following are some sample results of a typical run. First Δ was selected to be 6 milliseconds. $\Delta_i, i = 1, 2, 3$, was selected to be 2 milliseconds. The maneuver was completed in 6.48 seconds. The system ran 1.5 times faster than real time.

Figure 5a shows the angle described by the end effector with respect to the base as a function of kilosamples. Note that the path followed is nearly linear in angle (no detected backtracking). Figures 5b, 5c, and 5d show the position, velocity, and force potentials of the end effector with respect to kilosamples. Note that these potentials, after an initial transient, reach a steady state which

Figure 5a:

Figure 5b:

Figure 5c:

Figure 5d:

is maintained throughout the maneuver in spite of the fact that the manipulator go through several singularities as can be seen in Figure 5e that shows the path as detected by the quadrant sensor.

Notice also the effects of those singularities in the magnitude of the end effector angular velocity, Figure 5f.

In summary, the concept of declarative multiplexing controllers was used in solving a reasonably complex robot control problem and shows all the advantages predicted by the theory.

Finally the knowledge acquisition and coding took about 2 weeks.

6 Conclusions

A new class of knowledge-based controllers was introduced. The class termed declarative rational multiplexing controllers shows great promise for addressing

Figure 5e:

Figure 5f:

the verification issues discussed in Section 2.

In addition it was illustrated with the example, which is a representation of a wide class of robot control problems, that the concept yields to feasible implementations.

References

- [1] Kohn, W., "Declarative Theory of Rational Controllers," Proc. of IEEE CDC 1988, Vol. 1, pp. 130–136, Austin, TX, Dec. 7–9, 1988.
- [2] Kohn, W., "Hierarchical Control Systems for Autonomous Space Robots," Proc. of 1988 AIAA GN&C Conf., Vol. 1, pp. 382–390, Aug. 15–17, 1988, Minneapolis, Minnesota.
- [3] Kohn, W., Butter, J., Graham, R., "The Rational Tree Machine," Proc. Applications of Artificial Intelligence VII SPIE Vol. 1, pp. 264–274, Orlando, Fla., March 28–30, 1989.
- [4] Cadzow, M., "Discrete-Time and Computer Control Systems," Prentice Hall, Englewood Cliffs, NJ., 1970.
- [5] Morgenstern, L., "A First Theory of Planning, Knowledge and Action," Proc. of the 1986 Conf. on Reasoning about Knowledge: Theoretical Aspects, pp. 99–113, March 19–22, 1986.
- [6] Blanchard, D.C., Myers, R.M., "The Knowledge Representation Tool," Proc. of Robex 85, Houston, June 27–28, 1985, pp. 137–140.
- [7] Kohn, W., Carlsen, K., "Symbolic Design and Analysis in Control," Proc. of the 1988 Grainger lecture series at U. of Illinois, Urbana, May 23–25, 1989, pp. 40–52.

- [8] Kohn, W., “Declarative Control Theory,” accepted for publication in AIAA GN&D Journal, Feb. 1989.
- [9] Skillman, T., Kohn, W., “A Class of Hierarchical Controllers and their Blackboard Implementation,” accepted for publication in AIAA GN&D Journal, Sep. 1988.
- [10] Taylor, W., “Equational Logic,” in Universal Algebra by G. Grätzer, 2nd edition, Appendix 4, Springer Verlag, NY., 1979.
- [11] Evans, T., “An Algebra has a Solvable Word Problem iff it is Embeddable in a Finitely Generated Simple Algebra,” Algebra Universalis, 1978.
- [12] Kohn, W., Lecture Series on “Declarative Control and Rational Algebra,” HTC, June 2, Aug. 28, 1988, Seattle, Wa.
- [13] Lloyd, W., “Introduction to the Theory of Logic Programming,” 2nd ed., Springer Verlag, NY., 1987.
- [14] Yashuhara, A., “Theory of Recursive Function Theory and Logic,” Academic Press, 1966.
- [15] Kowalsky, T., “Logic Programming,” North Holland, 1982.
- [16] Kohn, W., “Declarative Hierarchical Controllers,” submitted to AIAA GN&D Journal, Feb. 1989.
- [17] Jurica, K., Kohn, W., Lai, D., “A Variable Configuration Controller for a Multipurpose Articulated End Effector,” Proc. of AIAA/NASA symposium on Automation Robotics and Advanced Computing for the National Space Program, pp. 11–18, Sept. 4–6, Washington D.C., 1985.