

Hierarchical Control Systems for Autonomous Space Robots

Wolf Kohn

Boeing Computer Services Company
Engineering and Scientific Services Division
206-865-3665 wolfk@boeing.com

Thomas Skillman

Boeing Electronics Company
High Technology Center
206-865-3056 toms@boeing.com

Abstract

This paper presents a design procedure for hierarchical robot controllers which are representable by a class of automata known as locally finite topological automata. At the heart of the design procedure is an algorithm known as the factorization algorithm which allows the decomposition of the controller into hierarchies that are abstractions of the control specifications for the robot controller. The paper also discusses some possible implementation architectures for the controller.

Introduction

This paper describes ongoing research towards the development of a procedure for designing generalized controllers for autonomous and teleoperated robots. The paper will propose a generic architecture for robot controllers and then discuss how given control specifications, a goal class, and a representation of the dynamics of the robot, the procedure determines feasible instances of the proposed generic architecture.

The effective implementation of these controllers for real-time systems will also be discussed, and the mapping of functional specification of the controller to the operational specification of the controller will be described.

The design of controllers for autonomous robots often requires the consideration of specifications and goals which are significantly different than the ones encountered in the design of the conventional signal-

based controllers.

For example, consider the following aerospace design scenario [1]: A 6 DOF manipulator is to be used to change modules in a satellite on orbit. For this problem, the plant of the system is characterized by the manipulator and actuator dynamics, that is, a set of coupled second-order differential equations. The goal is specified by a semantic net that contains the steps in the maneuver (e.g., remove left and right screws, remove lid, locate module, etc.) and the strategies for malfunctions and subgoal failure recovery.

It is immediately clear from this example that there is a great structural difference between the dynamic representations of the goal, a semantic net, and that of the manipulator and actuators, differential equations. This disparity is characteristic of robot control design problems. To tackle the design problem, virtually every scheme starts with a hierarchical architecture for the structure of the controller [2, 3, 4]. An example of a typical hierarchical control architecture is shown in figure 1.

?????

with s continuous with respect to the topologies in W and S .

Finally, the control specifications are represented by a set of clauses in a first-order logic theory [7] characterizing the desired behavior of the robot. These clauses include basic performance specifications such as stability and robustness and task-dependent specifications such as speed of response, terminal accuracy and trajectory shape constraints.

The control specifications also include an inference mechanism called a realization rule which transforms

A topological locally finite automaton is a tuple $\langle \Sigma, Q, \delta, B, Y \rangle$ where Σ is the input space, Q is the state space, δ is the state transition function, B is the output function, and Y is the output space. Σ , Q , and Y are topological spaces. The state transition function

$$\delta : Q \times \Sigma \rightarrow Q$$

is continuous in the induced topology on $Q \times \Sigma$ and satisfies the following local conditions. For every q in Q the map

$$\delta(q, \cdot) : \Sigma \rightarrow Q \quad (3)$$

has a range that is finite or at the most countable, and for all σ_1, σ_2 in Σ ,

$$\delta(q, \sigma_1 \sigma_2) = \delta(\delta(q, \sigma_1), \sigma_2) \quad (4)$$

the space Σ is provided with a binary operation \cdot that is continuous in the topology of Σ and such that the pair (Σ, \cdot) form as a semigroup.

The output function B is a continuous map

$$B : Q \rightarrow Y$$

A locally finite topological automaton $\langle \Sigma, Q, \delta, B, Y \rangle$ is a realization for a controller map C of the form of (2) if the following conditions hold:

$$\begin{aligned} \Sigma &= G \times S \\ Y &= D \times E \end{aligned} \quad (5)$$

and if for $g \in G$, $x \in X$, $s \in S$ such that $C(g(x), s)$ is defined, then

$$C(g(x), s) = B(\delta(q, (g(x), s))) \quad (6)$$

for some $q \in Q$.

The controller behavior is fully characterized by the locally finite automaton realizing its input output map. This automaton operates by inferring the state transition and output from the clauses characterizing the control performance specification.

That is, if q is its current state, the goal state is x , and g is the task goal, then the inference procedure alluded to above computes q_+ and (d, e) .

$$\begin{aligned} q_+ &= \delta(q, (g(x), s)) \\ (d, e) &= B(q) \end{aligned} \quad (7)$$

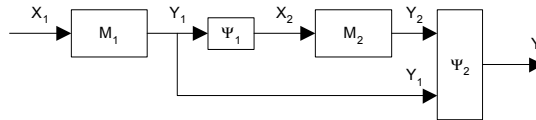


Figure 3: Modified wreath product of two machines

This is illustrated in figure 2(b). In general, even for a simple robot and elementary tasks, the number of clauses required to completely characterize the automaton associated with the controller is very large (~ 10000 clauses for a pick-and-place 3-axis manipulator). However, for a given state transition $\delta(q, (g(x), s))$, only a small percentage of all the clauses are involved in determining the new state. (That is the essence of the local finiteness of δ .)

Nevertheless, to determine which clauses are active, the inference procedure must examine all of them. This complexity issue makes it infeasible to implement directly the controller described above.

Several automata decomposition schemes can be considered to “parallelize” the implementation of the controller [11]. In this study, because of several computational and control performance advantages, a hierarchical decomposition of the controller based on the Krohn Rhodes decomposition algorithm for finite automata was selected.

The central element in the decomposition is an operation on automata known as the modified wreath product, which is an algebraic formalism for connecting two locally finite topological automata in a cascade configuration as shown in figure 3.

Let $M_1 = (\Sigma_1, Q_1, \delta_1, B_1, Y_1)$ and $M_2 = (\Sigma_2, Q_2, \delta_2, B_2, Y_2)$ be locally finite topological automata. Let Ψ_1 and Ψ_2 be given homeomorphisms

$$\begin{aligned} \Psi_1 &: Y_1 \rightarrow \Sigma_2 \\ \Psi_2 &: Y_1 \times Y_2 \rightarrow Y \end{aligned}$$

where Y is a topological space.

Then the modified wreath product of M_1 and M_2 , denoted by $M_{12} = M_1 \cdot M_2$, is the automaton $M_{12} =$

$(\Sigma, Q, \delta, B, Y)$ defined by the following identities

$$\begin{aligned}\Sigma &= \Sigma_1, & Q &= Q_1 \times Q_2 \\ \delta &: Q \times \Sigma \rightarrow Q\end{aligned}\quad (8)$$

where

$$\delta((q_1, q_2), \sigma_1) = (\delta_1(q_1, \sigma_1), \delta_2(q_2, \Psi_1(B_1(q_1)))) \quad (9)$$

and

$$B : Q \rightarrow Y$$

where

$$B((Q_1, Q_2)) = \Psi_2(B(q_1), B_2(q_2)) \quad (10)$$

The identities above imply that any output trajectory realizable by M_{12} is realizable by $M_1 M_2$.

In general, given n automata M_1, M_2, \dots, M_n and the corresponding morphisms $\Psi_{1i}, \Psi_{2i}, i = 1, \dots, n$, their cascade composition M is given by

$$M = (\dots (M_1 \cdot M_2) \cdot \dots M_n) \quad (11)$$

Note that the modified wreath product is not associative.

The factorization algorithm can now be formulated.

Given the controller automaton $M = \{G \times S, Q, \delta, M, D \times E\}$ defined by (5) and (6), find automata $M_1 = \{G_1 \times S_1, Q_1, \delta_1, M_1, D_1 \times E_1\}$ and $M_2 = \{G_2 \times S_2, Q_2, \delta_2, M_2, D_2 \times E_2\}$, homeomorphisms Ψ_1, Ψ_2 with $\Psi_1 : D_1 \times E_1 \rightarrow G_2 \times S_2$, $\Psi_2 : (D_1 \times E_1) \times (D_2 \times E_2) \rightarrow (D \times E)$, and a state homeomorphism $h : Q \rightarrow Q_1 \times Q_2$ such that the automaton M is covered by the modified wreath product of M_1 and M_2 . Symbolically,

$$M < M_1 \cdot M_2 \quad (12)$$

Expression (12) means that any output trajectory in M is also an output trajectory in $M_1 \cdot M_2$. The state homeomorphism h defines the inequality in (12). The elements in M_1 and M_2 satisfy the following relations:

$$\begin{aligned}\delta_1 : Q_1 \times (G_1 \times S_1) &\rightarrow Q_1 \\ G_1 \times S_1 &= G \times S\end{aligned}\quad (13)$$

$$\delta_1(p_1^2(h(q)), (g(x), s)) = p_1^2(h(\delta(q, (g(x), s)))) \quad (14)$$

where $p_1^2 : Q_1 \times Q_2 \rightarrow Q_1$ is the projection homeomorphism,

$$\begin{aligned}B_1 : Q_1 &\rightarrow (D \times E) \\ B_1(p_1^2 h(q)) &= p_1^2(\Psi_2^{-1}(B(q)))\end{aligned}\quad (15)$$

$$\begin{aligned}\delta_2 : Q_2 \times (G_2 \times S_2) &\rightarrow Q_2 \\ \delta_2(h(p_2^2(q)), \Psi_1(B_1(p_1^2(q)))) &= p_2^2(h(\delta(q, \Psi_1(B_1(p_1^2(q)))))\end{aligned}\quad (16)$$

where $p_2^2 : Q_1 \times Q_2 \rightarrow Q_2$, and finally,

$$M_2(p_2^2(h(q))) = p_2^2(\Psi_2^{-1}(B(q))) \quad (17)$$

Note from (14) and (16) that both M_1, M_2 are homeomorphic images of M .

In order to formulate the decomposition algorithm, an additional concept is needed: the concept of automaton complexity.

Given an automaton M , its complexity is given by the function $a : M \rightarrow R$, R the real numbers, that computes the average number of state transitions possible from all the states of M . Since M is assumed to be locally finite, this function is well defined.

Now, the decomposition algorithm is defined: Given M , find M_1, M_2 such that

$$\min_{h, \Psi_1, \Psi_2} a(M_1 \cdot M_2)$$

subject to (13),(14),(15),(16),(17):

$$\begin{aligned}a(M_1) &\leq a(M) \\ a(M_2) &\leq a(M)\end{aligned}\quad (18)$$

That is, find a covering of M by the modified wreath product of M_1 and M_2 , i.e., find homeomorphisms h, Ψ_1, Ψ_2 such that the complexity of the covering is minimized and each individual automaton M_1, M_2 has no higher complexity than the original one M .

This algorithm has been implemented in Prolog and is currently being tested on several specific robot control design test cases.

The algorithm can be used recursively as follows: given M find M_1, M_2 according to (18), then apply the algorithm to M_1 and M_2 to either obtain

$M_{11}, M_{12}, M_{21}, M_{22}$ such that

$$\begin{aligned} M &< M_1 \cdot M_2 \\ M_1 &< M_{11} \cdot M_{12} \\ M_2 &< M_{21} \cdot M_{22} \end{aligned} \tag{19}$$

and so on.

This procedure is halten whenever for some $M_{i\dots k-1,k}$

$$a(M_{i\dots k-1,k}) > a(M_{i\dots k-1})$$

This halting condition is guaranteed to occur since each automaton generated is locally finite.

Notice that because of the fact that each automaton generated by the factorization algorithm is a homeomorphic image of the original one, each automaton is a controller for the robot at some level of abstraction determined by the chain of state homeomorphisms linking their state spaces.

As for the original controller automaton, both the state transition and output functions are realized by inferring on the appropriate clauses of the control specifications; but this is a subset of the original set because the only clauses that are required for an automaton dynamics in the decomposition chain are those in the preimage of the range of the corresponding homeomorphism. Any other clauses are in its kernel and do not affect the automaton dynamics.

In summary, the hierarchical decomposition of the controller automaton M is given by

$$M < M_1 \cdot M_2 \cdot \dots \cdot M_n$$

where each M_i represents the level dynamics of the hierarchical controller.

The Realization Procedure

In order to develop an effective procedure for the factorization of a robot controller into a hierarchical structure, it is necessary to characterize the hierarchical levels. Although not necessary for the formulation of the procedure, it will be convenient for all the levels in the hierarchy to have the identical operational structure. This will allow for a recursive definition

of the overall controller which simplifies the factorization algorithm and the controller implementation. An operational structure that has proven satisfactory as a candidate for the level controllers in the hierarchy is described in [12].

This structure will be referred to as the canonical form of the controller. A block diagram depicting its main components is shown in figure 4. Each functional element in a level of the hierarchy is realizable by a locally finite topological automaton as described in the previous section.

The output of the realization algorithm will be a functional specification for each of the elements of the canonical controller for every level in the decomposition. This functional specification must be mapped into system architecture and then encoded into hardware and software to implement a real system. The mapping will reflect the system architecture on which the controller is to be implemented. Our approach is to model the system architecture as closely as possible after the hierarchical canonical form, providing a simple target for the mapping process. The benefits of this will be 1) a uniform mapping of data and operator representations, 2) simplified computation of interprocess communication rates and volumes, 3) unified data structures and accessors across levels, and 4) automatic verification of system back to initial control specifications.

The principal types of computation that will take place in the hierarchical controller are: (1) the abstraction of data representations, (2) the estimation of state from a model and input data, (3) the determination of the difference between two data structures, and (4) the resolution of non-determinism in goal structure.

The system architecture is being developed to meet real-time performance goals. The approach is to optimize performance by exploiting parallelism using data-driven (event) triggering of computations, supporting highly efficient abstract data structures and their accessors, and simplifying communication between computing elements.

The canonical form of the controller provides a regular structure to the functional specification of the controller. This structure can be exploited in a number of ways to implement a high performance real-

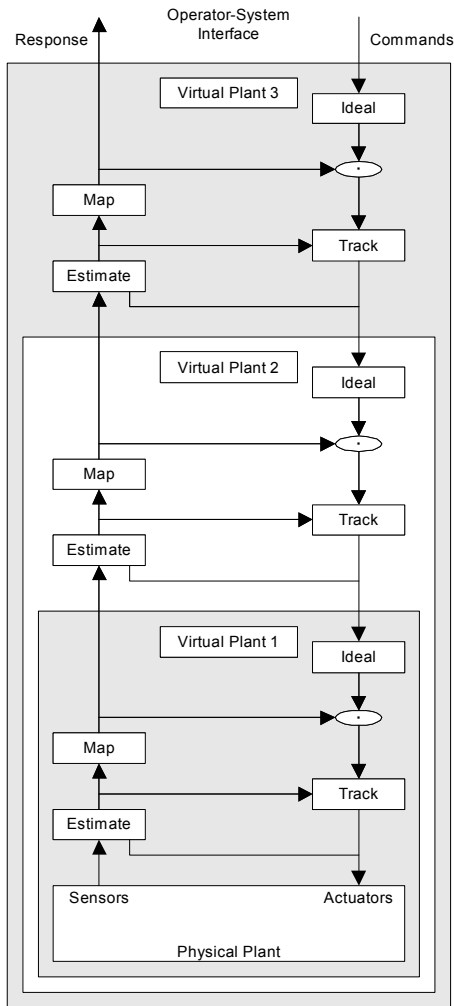


Figure 4: Recursive application of canonical form

time controller. Each level of the controller is computing in parallel, and each of the 5 elements within a level are computing in parallel. They are synchronized by the communication of their results to each other.

The computations in most elements can be data-driven. The ideal plant model will only compute a new desired behavior when its incoming goal structure changes. The error computation will only operate when one of its two inputs changes. The tracker will compute if the error changes or if the state changes. The state estimator is an exception because it may recompute the estimated state based on a fixed time interval, allowing it to compute changes in state that are not directly measured but are predicted by the model.

Data abstraction is concerned with building data representations that cover all aspects of a less abstract representation. The automatic decomposition will determine what the useful abstractions are, and the implementation must support the storage, retrieval, and calculation of these abstractions.

The communication between the various computing elements must be supported efficiently. Note that the interprocess communication requirement is local, that is, most elements only communicate with a few fixed other elements. Also the nature of what is communicated will vary with the level of the hierarchy. The bottom levels may only be representing and communicating integer data. At the top of the hierarchy the elements may be communicating large abstract information structures.

Encoding onto Hardware and Software

Based on the mapping requirements, two system implementation architectures are currently being evaluated, 1) a blackboard paradigm [13] based multiprocessor system, and 2) a custom transputer multiprocessor architecture. The main elements of the blackboard system, shown in figure 5, are 1) a shared database, 2) distributed parallel processes, called knowledge sources, 3) a flexible process invocation

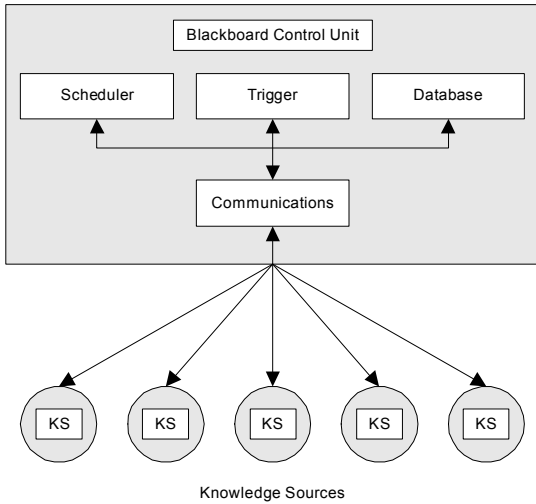


Figure 5: Generic blackboard architecture

mechanism, referred to as data-directed triggering, and 4) a process scheduler which can use analytic or heuristic methods. This paradigm has been used in ad hoc controllers such as CMU's Codger system on Alvan [14] and on Boeing's KBB system for the "Flying Eye" [15]. The paradigm can exploit large-grain parallelism and supports the concepts of multiple levels of abstraction and data-directed invocation. Recent work on multiple communication blackboards [11] may provide an additional performance increase by using a separate blackboard for each level of the hierarchy. The number of blackboards needed is typically driven by the bottleneck at the blackboard's shared database. Issues of custom VLSI implementation of these systems is currently under study at Boeing High Technology Center.

The other hardware system implementation being evaluated is based on dedicating computing hardware directly to each of the functional modules of each of the levels. A look at the canonical form shows that each module has at most a total of 4 input and output lines. This makes the 4-channel transputer an interesting candidate processor. Each transputer can be programmed to compute the necessary functions, and the high-speed serial channels can be used to

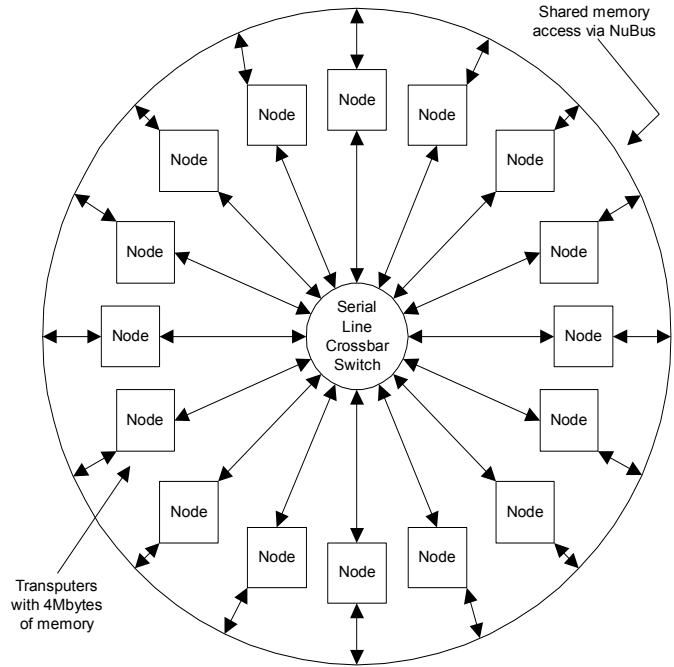


Figure 6: Transputer multiprocessor architecture

communicate messages and accomplish process synchronization.

A 36-transputer multiprocessor has recently been developed here. Figure 6 shows how the transputer serial communication lines are all routed through a software-programmable crossbar switch. The transputers have up to 4 megabytes of memory, each which is mapped into a global address space over its NuBus interface. This allows memory-to-memory communication as well.

The blackboard system approach offers the most flexibility and the possibility of dynamic processor load-balancing but at the cost of control overhead. The transputer may waste hardware resources in dedicating hardware to every function, but the control overhead is reduced. These and other issues are the focus of ongoing work.

Conclusions

Our objective is to develop the theory and tools necessary to build large controllers for complex systems that may or may not be directly supervised by an operator. We seek to automate the design process and improve the realtime performance of implementations.

Our approach is to develop (1) a theory of hierarchical control, (2) a unified representation of controller goals, requirements, and dynamics, (3) an automatic decomposition through factorization algorithm, (4) a simple mapping from the hierarchical controller specification into a system implementation, and (5) a high-performance architecture for realtime execution of the hierarchical controller.

This paper shows a factorization algorithm for obtaining hierarchical robot controller specifications given the desired system behavior. The algorithm is effective, that is, computable. It proceeds until the computational complexity of the structure of the controllers has been minimized. This algorithm is the first step in an overall procedure for the automated design of complex control systems, as shown in figure ??.

Acknowledgements

This research is being funded by IR&D Project BEC-488B.

References

- [1] Kohn, W., *Task Interpreter for Astrobot*, JAIPCC, Houston, Texas, 1985.
- [2] Jamshidi, J., *Large Scale Control Systems*, Springer-Verlag, New York, 1986.
- [3] Mesarovic, M. D., Marco, D., Tashahara, Y., *Theory of Hierarchical Multilevel Systems*, Academic Press, New York, 1970.
- [4] Brady, M., et al., eds., *Robot Motion, Planning and Control*, MIT Press, Cambridge, Mass., 1982.
- [5] Ginsburg, A., *Automata Theory*, Academic Press, New York, 1972.
- [6] Eilenberg, S., *Automata Languages and Machines, Volume B*, Academic Press, New York, 1976.
- [7] Loyd, R., *Introduction to Logic Programming Theory, Second Edition*, Springer-Verlag, New York, 1976.
- [8] Kohn, W., *A Declarative Theory for Rational Controllers*, Proceedings of the IEEE Control and Decision Conference, Los Angeles, December 1988.
- [9] Kohn, W., *A Rendezvous Expert System for STS, ROBEX*, Pittsburgh, Pennsylvania, July 1987.
- [10] Kowalsky, R., Van Embden, R., *The Semantics of Logic Programs*, Second IEEE Logic Programming Conference, Los Angeles, 1984.
- [11] Larson, F., *Large Space Systems*, John Wiley, New York, 1982.
- [12] Skillman, T. L., Kohn, W., *Blackboard Based Hierarchical Intelligent Control System*, AIAA Journal Guide. Cont. & Dyn., to appear.
- [13] Nii, P. H., *Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures*, The AI Magazine, Summer 1986.
- [14] Shafer, S., Stenz, A., Thorpe, C., *An Architecture for Sensor Fusion in a Mobile Robot*, IEEE International Conference on Robotics and Automation, 1986.
- [15] Skillman, T. L., *Distributed Cooperating Processes in a Mobile Robot Control System*, Proceedings of the Conference on Artificial Intelligence for Space Applications, November 13-14, Huntsville, Alabama, 1986.