The Use of Supervenience in Dynamic-World Planning[1]

Lee Spector

School of Communications and Cognitive Science
Hampshire College, Amherst, MA 01002
lspector@hamp.hampshire.edu

James Hendler

Department of Computer Science
University of Maryland, College Park, MD 20742
hendler@cs.umd.edu

Abstract

This paper describes the use of *supervenience* in integrating planning and reaction in complex, dynamic environments. Supervenience is a form of abstraction with affinities both to abstraction in AI planning systems and to partitioning schemes in hierarchical control systems. The use of supervenience can be distilled to an easy-to-state constraint on the design of multilevel dynamic-world planning systems: world-knowledge up, goals down. We present the *supervenience architecture* which embodies this constraint, and contrast it to the subsumption architecture of Brooks. We describe the performance of an implementation of the supervenience architecture on a problem in the HomeBot domain, and we conclude with a discussion of the role that supervenience can play in future dynamic-world planning systems.

Introduction

The "classical" AI planning paradigm, based on the plan-then-execute model of action and on assumptions of a static world and an omniscient reasoner, is inadequate for complex, dynamic environments. The rejection of this paradigm has generated a body of work that we call *dynamic-world planning*. One of the developing trends in dynamic-world planning is toward systems based on distributed control structures, many of which include both "deliberative" and "reactive" components (Maes 1990). For reasons of efficiency and modularity these systems are often partitioned into layers of abstraction.

Two general ideas of abstraction have hitherto played an important role in AI research. The first involves the division of systems into "levels of detail" as in ABSTRIPS (Sacerdoti 1974). This type of *simplification* abstraction has received considerable recent attention, including demonstrations of its utility for reducing the size of problem-solving search spaces (e.g., (Knoblock 1991), (Prieditis & Janakiraman 1993)). A second form of abstraction is popular in robotics and in research on large-scale cognitive models; in this form of abstraction, computational *processes* are divided into levels, providing flexibility of behavior through the use of modularity and parallelism.

Supervenience is a species of abstraction that we believe to be important for systems that must integrate high-level reasoning with real-time action. Simplification abstraction is a special case of supervenience, and the search-reduction benefits of ABSTRIPS-style systems are sometimes available in supervenient planning systems as well. The generality of supervenience also allows, however, for uses of abstraction similar to those available in blackboard architectures and in other multilevel control systems.

The central idea of supervenience is that representations at lower levels of abstraction are epistemologically "closer to the world" than those at higher levels, and that the representations at higher levels therefore *depend* on those at lower levels. The higher levels may contain representations that are simplifications of low-level, sensory reports, but they may just as well contain representations that are complex, structurally rich aggregates that have no unified representation at lower levels. In contrast to ABSTRIPS-style systems, in which higher levels must be simplifications of the lower levels, levels of supervenience may be dissimilar in various ways so long as the proper dependence relation holds. The thesis is that it is this *dependence*, and not the more restrictive notion of simplification, that allows for the flexible integration of deliberation and reaction.

The concept of supervenience applies naturally to multilevel computational architectures in which the higher levels are coupled to the world *through* the lower levels. In such cases the privileged status of the lower levels (vis-à-vis access to the world) can be used to advantage. We have formalized the supervenience relation in the context of nonmonotonic reasoning systems, using the concept of "defeasibility" in nonmonotonic systems to spell out the appropriate notion of "dependence." Supervenience is defined to be the case in which lower levels can defeat higher level facts but not vice versa (Spector 1992); this can be abbreviated as "assertions up, assumptions down," and reformulated for implementation purposes as "world knowledge up, goals down." The bottom line for system-builders is this: Low levels should "know" enough to be right about, and to act upon, their assertions. High levels should configure (e.g., provide goals for) lower levels, but should not override knowledge determined to be true by the lower levels. Lower levels may need to monitor for goal-changes from above, but not for changes in world knowledge from

[1]This work was supported in part by a grant from the Lemelson National Program in Invention, Innovation, and Creativity, and in part by the University of Maryland Institute for Systems Research (an NSF-supported research center).

above. Inter-level communication procedures can take advantage of this restriction. The knowledge/action "domains" of each level should be chosen with this restriction in mind.

The history of supervenience and its relation to other forms of abstraction in the literature can be found in (Spector 1992). A discussion that relates this work to the neuropsychology of human planning can be found in (Spector & Grafman, in press).

The Ice Cube Problem

Consider a household robot in the midst of a routine chore. Suppose that an ice cube is on the floor in the middle of a room, and that at some point it is seen by the robot. Any assumption of a built-in "ice cube reaction rule" is problematic; ice cubes do not always present problems, and any reaction rule that checks all contextual circumstances to determine if there *is* a problem upon each sighting of an ice cube would be computationally complex if constructible at all. We should therefore assume that no such rule exists, and hence we should expect no immediate response to the sighting of the cube. Assuming that the robot has sufficient causal knowledge to infer that the ice cube will melt over time, however, and assuming that it can also infer that puddles may be dangerous, it would be appropriate for the robot to *eventually* make such inferences, and to alter its behavior accordingly.[2] The robot should then suspend work on its current task, take care of the potential safety hazard, and then resume the original task with as little replanning as possible.

Neither "replanning" systems nor "purely reactive" systems are capable of producing appropriate behavior in this problem. Replanning systems are driven by failures, but in the Ice Cube problem no failure occurs. The change in behavior after seeing the ice cube must be triggered by inferences, not by simple conditions that may be detected in the world. Purely reactive systems preclude, by definition, the use of reasoning processes that could infer the necessary causal connections at runtime.[3]

Problems like the Ice Cube problem make it clear that reasoning and reaction must be *integrated*, and not just *combined*, in systems that are to produce appropriate behavior in complex, dynamic environments. Approaches to such integration generally involve concurrency, and often divide computational components into "levels" in order to organize the complex systems that result (Gat 1991). The notions of "level" that have been used in this context form a large and heterogeneous set; one of the goals of our research is to articulate principles that underlie good choices of levels, and good strategies for inter-level communication, for systems that integrate reasoning and action.

[2]Of course, the ice cube may be hazardous before it melts as well; in any event we may presume that reasoning is required to infer the existence of a present or future hazard.

[3]Examples of systems that we consider to be "purely reactive" are those described in (Agre & Chapman 1987) and (Schoppers 1987).

The Supervenience Architecture

The *supervenience architecture* specifies that a number of data/processing *levels* be arranged in a partial order, with the minimal elements of the partial order connected to sensors and effectors. The levels may run in parallel and communicate asynchronously. Each level should contain both procedural and declarative components, and should model the world, solve problems, and initiate actions at a particular level of abstraction. Translation systems, situated between planning levels, are to pass goals downward and information about the state of the world upward. This implements the asymmetric supervenience relation between levels—the *dependence* of high-level knowledge upon low-level knowledge.

Although the procedures at each level should possess certain general properties (such as interruptability at some level of temporal granularity) the architecture allows for the use of radically different planning methodologies at each level. For example, one might use a connectionist system at the lowest level, logic programming techniques at intermediate levels, and a planning system such as NONLIN at the highest levels.

Comparison to the Subsumption Architecture

The *subsumption architecture* of Brooks is a well known architecture for mobile robots (Brooks 1990). Brooks characterizes previous systems as decomposing behavioral problems into a series of "functional units" that can be illustrated by a series of "vertical slices." He prefers a decomposition into "task achieving behaviors," illustrated by a series of "horizontal slices." The supervenience architecture combines these two methods of decomposition (as do others, e.g. (Albus 1991)). The supervenience architecture allows, as does the subsumption architecture, the direct coupling of sensation to action at the lowest level of the system. The lowest levels can provide basic "competencies" without the intervention of higher, more complex levels. On the other hand, the supervenience architecture stipulates that the higher levels receive their knowledge of the world via the lower levels of the system, and that they pass commands for action (expressed as goals) through the lower levels as well. This accords with some of the principles of "vertically sliced" systems; for example, that the knowledge required for planning is provided via modeling from perception, and not from the sensors directly. The resulting mix of horizontal and vertical decomposition is quite practical—indeed, some examples of the subsumption architecture exhibit such a combination (Brooks 1990, 15–19).

In the subsumption architecture, the "reasoning" at each level is performed by a set of modules implemented as augmented finite state machines. The finite state machines are "hard wired"—they are created when the system is built and are never changed. The topology of the module network is also static. Regardless of the theoretical computational power of such systems, it is not at all clear how one would encode complex symbolic AI systems within these finite state machines—indeed, Brooks is interested in showing

how interesting behavior can emerge from the simplest of systems. The supervenience architecture, in contrast, allows for arbitrary processes at each level.

Bottom-up communication in the subsumption architecture is simply a matter of connecting the appropriate "wires" so that low-level data flows to high-level modules. Bottom-up communication in the supervenience architecture is similar, but a significant difference appears with respect to top-down communication. In the subsumption architecture, top-down communication occurs by "inhibition" of the outputs of low-level modules, and by the "suppression" and "replacement" of the inputs to low-level modules. In the case of inhibition, high-level modules simply cancel the effects of low-level modules that continue to run. In the case of suppression/replacement, higher-level modules *replace* the inputs of lower-level modules, thereby altering their behavior.

The use of suppression/replacement for knowledge about the world is entirely inconsistent with the principles of the supervenience architecture. In the supervenience architecture lower levels are epistemologically "closer to the world"—their knowledge cannot be defeated by knowledge from higher levels. Higher levels in the supervenience architecture communicate with lower levels via *goals*—they *configure* the behavior of the lower levels, but they do not *override* lower-level knowledge of the world.

The use of action inhibition is also inconsistent with the principles of the supervenience architecture. Lower-level procedures, on the basis of goals (perhaps obtained from higher levels) and knowledge of the world, compute and execute actions to achieve goals; they are presumed to have the necessary knowledge to achieve their goals *at their level*. The higher levels may provide goals, but the determination of appropriate *lower-level* actions to achieve such goals is the prerogative of the lower, not the higher, level.

It is possible to restrict the top-down communication in a subsumption system to conform with the principles of the supervenience architecture; the use of inhibition can be avoided, and inputs can be grouped as "goal" and "world" inputs, with suppression/replacement allowed only for the former. Such restrictions are not, however, part of the subsumption architecture philosophy.

Alternative architectures can be found in the literature, some of which use communication strategies that are more like supervenience than they are like subsumption (e.g., TCA (Lin, Simmons & Fedor 1989) and ATLANTIS (Gat 1991)). We believe that systems built on these architectures will scale up better than will subsumption systems, and that strict adherence to the supervenience constraint may provide further advantages.

Implementation

The Abstraction Partitioned Evaluator (APE) is an implemented dynamic-world planning shell based on the supervenience architecture. Here we list APE's specific levels and provide a brief discussion of the types of knowledge represented at each level; a more complete discussion can be found in (Spector 1992).

The levels used in APE are, from lowest to highest: perceptual/manual, spatial, temporal, causal, and conventional (see Figure 1). A linear order is imposed on the levels, although the supervenience architecture allows any partial order. The italicized annotations to Figure 1 are characteristic of the types of knowledge at each level; they show how a particular sighting of a smoke cloud might be cognized across the hierarchy.

The perceptual/manual level represents events as sensory reports and as operators for effector manipulation. The coupling of perception to action at this level is analogous to simple "reflex arcs" in animals. Although spatial and temporal data is *present* at this level, reasoning *about* space and time occurs elsewhere. The spatial level contains structures that organize perceptual data with respect to spatial relations such as *on*, *above*, *near* etc., along with operators for spatial reasoning (e.g., path planning). The temporal level augments spatial representations with temporal relations, allowing for reasoning about deadlines and temporal projection. While every level represents time in some manner (and certainly every level acts *in* time), only at the temporal level do representations such as "Betty is late" arise; lower levels simply tag representations with temporal information and perform actions as quickly as they can. It is at the temporal level that the system reasons *about* time and schedules actions to meet deadlines. The causal level contains representations that embody the agent's conception of the causal structure of the world, including causal rules and causally deduced facts. The conventional level contains knowledge about facts that are true only by convention; for example, that a certain hand gesture is a signal for a turn, that a dirty sock "belongs" in the hamper, or that a certain condition is considered undesirable or dangerous.

APE's levels partition the system's knowledge into a set of connected, communicating systems, each of which exhibits limited expertise at a particular level of abstraction from the world. Note that this is *not* to say that the representations characteristic of one level are entirely absent from all other levels. For example, temporal representations exist at *all* levels of the system, even though explicit reasoning *about* time occurs only at the temporal level.

The planning procedures in APE are implemented as operators that are instantiated in a goal-driven fashion. APE

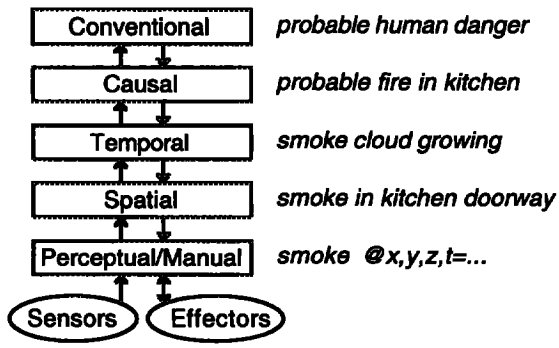

Figure 1. The specific levels of APE.

operators contain specifications for asynchronous parallel action networks, similar to Petri nets (Peterson 1981), that are incrementally expanded and executed. All parallelism in APE is currently simulated. Operator instances have priorities that can be passed through goals; if a goal is posted with a certain priority then that priority is inherited by operators triggered by the goal. These priorities are used by resource arbitrators to handle resource conflicts amongst operator instances. The between-level translation systems are implemented as collections of rule-based structures that run continuously, posting goals to lower levels on the basis of existing higher level goals, and manipulating lower level data to produce higher level knowledge. Additional features of the implementation are documented in (Spector 1992).

HomeBot and the Ice Cube

HomeBot is a simulated robot that "lives" in a one bedroom apartment. HomeBot is expected to perform household tasks such as cleaning and protecting the human residents from danger. Exogenous events may be triggered at any time, and HomeBot "knows" only that which it can infer from the data (descriptions of pre-recognized objects) that it receives through its sensors.

The HomeBot domain presents challenges for dynamic-world planning research because of the structural richness of the real-world events that can occur. While regimented domains such as Tileworld (Pollack & Ringuette 1990) are useful for certain quantitative studies, they do not give rise to high-level symbolic reasoning tasks with which real-world agents must contend. The Seaworld domain of (Vere & Bickmore 1990) and the Delivery Truck Domain of (Firby 1989) are both similar to the HomeBot domain in that realistic segments of the world are simulated to a level of detail intermediate between that of the real world and that of "toy" domains such as the Blocks World. A discussion of design issues for related test-bed environments can be found in (Hanks, Pollack & Cohen 1993).

The Ice Cube problem provides an example of the need for *integration* of reactive and deliberative components within dynamic-world planning systems. Recall that the appropriate behavior in the early stages of the Ice Cube problem is for HomeBot to continue the previous chore while reasoning about the ice cube. Once a hazard is inferred the previous chore should be suspended, and the hazard should be removed. After the hazard is removed work should resume on the previous chore with as little replanning as possible.

In our implementation high-level operators perform sufficient initial reasoning to set up monitors for "interesting" conditions with respect to their levels of expertise; for example, the conventional-level **earn-praise** operator monitors for "improper" or "hazardous" conditions. When either is found a goal is posted for its abolishment. Goals for the removal of improprieties are given **normal** priority, while goals for the removal of hazards are given **urgent** priority. This prioritization is a simple piece of conventional knowledge encoded into the **earn-praise** operator.

Normal chore-oriented behavior is generated by sub-

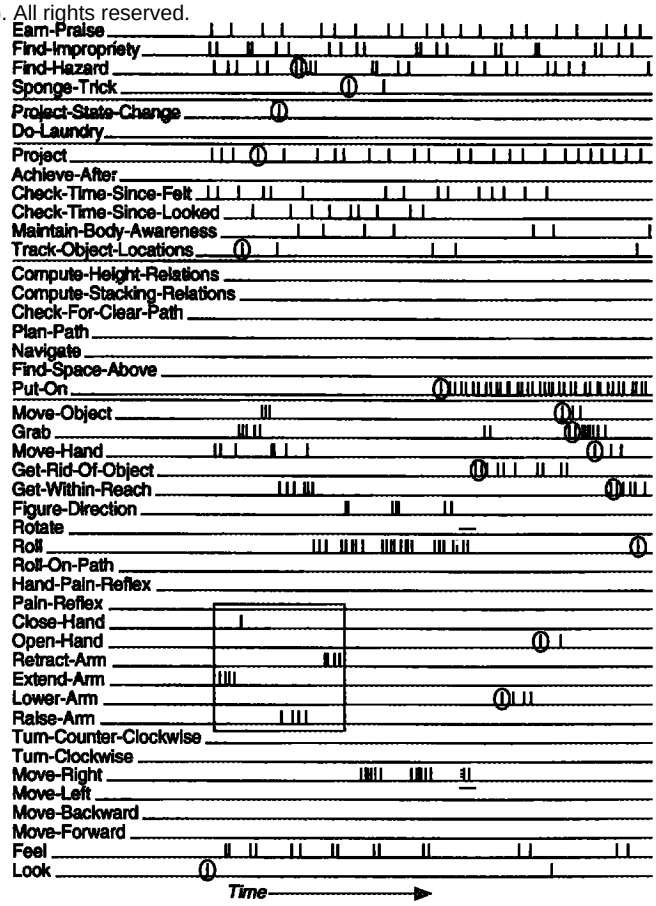

Figure 2. Operator activations for part of the Ice Cube problem.

goaling to discover and abolish improper conditions. In the run described here, the improper condition is a dirty sock on the floor. A representation of the sock propagates to the causal level, where the **do-laundry** operator generates a plan to clean the sock via a series of goals that will decompose into temporal, spatial, and perceptual/manual tasks.

Figure 2 shows an operator activation graph for a segment of the run. The activity of operators is indicated by short vertical lines; time is represented by the horizontal axis. Each vertical line indicates an explicit allocation of CPU time to an operator instance, but operator instances may be "working" via the activity of demons even when no activations are shown. The double horizontal lines separate levels; from top to bottom these are the conventional, causal, temporal, spatial, and perceptual/manual levels. The initial (left) portion of the graph shows HomeBot simultaneously picking up the sock and noticing the ice cube. The large square indicates the actions of picking up the sock (extending the arm, closing the hand, and then raising and retracting the arm), while the circles show the flow of activity relating to the ice cube. The left-most, lower-most circle indicates the **look** operation during which the ice cube is first seen. The subsequent circles show the propagation of the representation of the ice cube to the causal level, at which a state change (melting) is projected, and to the conventional level, at which the

change is recognized as a hazard. The **sponge-trick** operator contains a remedy for ice cube hazards: put a sponge on the cube to absorb the water as the ice melts. Although this solution is contrived it is not "hard wired"—once the goal of preventing the puddle is posted, more realistic operators could also be employed to discover a solution. The more important point, however, is that the solution is not hard wired *to the conditions in the world*. The fact that **sponge-trick** is applicable requires inference of the hazard, and the inference processes progress in parallel with action.

The later (right) portion of the graph shows the decomposition and execution of the plan for moving the sponge. The circles indicate the major events in this process, while the tall rectangle indicates the suspension of the previous task. Note that work on the previous task continues until the hazard-prevention task is decomposed to the point at which a resource conflict occurs. (The **roll** and **get-rid-of-object** operators both use the **arm-control** resource.) Once the operator instances for the new task have seized control of the robot's arm, the sock is dropped (via **get-rid-of-object**, **lower-arm**, and **open-hand**) and actions are initiated for the tasks of retrieving and moving the sponge. By the end of Figure 2 a new instance of the **roll** operator has begun to execute.

Further graphs (see (Spector 1992)) show the eventual completion of the hazard-prevention task and the resumption of the laundry task. When the sponge is dropped the **hand-control** resource is released, allowing for the resumption of operator instances that had previously been suspended. The instance of the **move-object** operator that was involved in moving the sock failed during the hazard-prevention task; it had posted a demon to monitor **'(holding dirty-sock)** that fired, triggering failure, shortly after the sock was dropped. The operators working on subgoals of **move-object** (for example, **roll**) were also terminated at that time. The goal of moving the sock remained, however, and a new instance of **move-object** was instantiated, leading also to an instance of **grab**. This instance of **grab** was suspended as soon as it requested the **arm-control** resource. When the sponge is dropped and the **arm-control** resource once again becomes available, the suspended **grab** operator is resumed. The resumption of the laundry task is fairly simple because the intervention of the hazard-prevention task caused failures only at the lowest level; none of the components of the laundry task at other levels of representation were affected. This localization of the failure to the lowest level derives from our choice of levels, which in turn derives from the concept of supervenience. It is also interesting that higher-level components of the hazard-prevention task (inferring that the hazard has indeed been prevented) are able to continue in parallel with the resumption of the laundry task.

The Value of Supervenience

While HomeBot is capable of integrating planning and action in interesting ways, one might still question the role that *supervenience* plays in the resulting performance. The implementation provides a number of facilities that help to explain the observed behavior: the Petri-net basis of operator expansions, simulated parallelism at several levels of granularity, the resource arbitrators, etc. Such facilities are not mandated by the concept of supervenience *per se;* they are features of the supervenience architecture and of the implemented APE shell in which HomeBot was programmed. So while it is clear that HomeBot is able to solve new classes of dynamic-world planning problems, it is important to understand how the use of supervenience supports this ability.

The role of supervenience in HomeBot is twofold: first, the communication between levels adheres to the "world knowledge up, goals down" constraint; and second, the choice of specific levels for APE was derived from a philosophical analysis of, for example, the supervenience of spatial knowledge on perceptual/manual knowledge (Spector 1992). The latter point should be of interest to others who have suggested that the concept of supervenience will play an important role in the future of AI (e.g., (Pollock 1989)). Supervenience is a relation that philosophers have developed to span the gulf from low-level physicalistic descriptions of the world to high-level knowledge. It is precisely this gulf that must be spanned in the mental representations of the thinking robots of the future, and hence it is interesting that a set of levels based on supervenience has turned out to be useful in building a dynamic-world planner that operates in a knowledge-rich environment.

The adherence to the "world knowledge up, goals down" constraint is a more concrete manifestation of supervenience about which more can be said. How useful is this constraint? It appears to conflict with the principles of the popular subsumption architecture, so we can expect that further evidence of its utility would have a real impact on system design. HomeBot is a demonstration that adherence to the supervenience constraint is *consistent* with the production of complex, reactive behavior in rich dynamic worlds; more work will be required to show that systems that adhere to the constraint will ultimately outperform those that do not. We envision two lines of attack on this question: the first is to build pairs of systems that differ, as far as this is possible, only by the allowed forms of inter-level communication. Our ability to evaluate the utility of supervenience will improve to the extent that we can isolate the effects that communication constraints have on system performance. We expect to find that the regimented communications in the supervenient systems allow for optimizations not possible in the non-supervenient systems.

The second line of attack is to analyze successful dynamic-world planners to discern implicit level-structures and communication patterns about which their implementors might not even be aware. Such analysis can be applied both to the hand-crafted systems of other planning researchers and to the results of new forms of automatic programming. Koza, for example, has used genetic programming techniques to produce programs that he considers to be instances of the subsumption architecture (Koza 1992).

It is conceivable that further analysis of the resulting programs will show that they use communication strategies that are more consistent with supervenience than with subsumption. Such a result—that the fitness-proportionate selection methods of genetic programming favor supervenience over subsumption—would be strong evidence of the utility of the "world knowledge up, goals down" constraint. Of course, it is also possible such analyses will produce the opposite result; we are currently developing the analytical tools to conduct a study along these lines (Spector 1994).

Conclusions

The concept of supervenience was useful in the construction of a dynamic-world planner that integrates planning and action. Supervenience provided guidance in partitioning the system's knowledge across multiple levels, and the "world knowledge up, goals down" constraint of the supervenience architecture was helpful in structuring the dynamics of the system. Further studies will help to quantify the value of supervenience for dynamic-world planners, and to arbitrate the conflicts between the supervenience and subsumption architectures.

Acknowledgments

Parts of this paper were adapted from (Spector 1992) and from (Spector and Hendler 1992). Valuable feedback was provided by many people, including V.S. Subrahmanian, Jim Reggia, Shekhar Pradhan, Don Perlis, Dana Nau, John Horty, Jordan Grafman, Michael Georgeff, Anand Rao, Martha Pollack, Rebecca S. Neimark and the members of the University of Maryland PLUS lab.

References

Agre, P. E., and Chapman, D. 1987. Pengi: An Implementation of a Theory of Activity. In Proceedings of the Fifth National Conference on Artificial Intelligence, AAAI-87, 268–272.

Albus, J. S. 1991. Outline for a Theory of Intelligence. *IEEE Transactions on Systems, Man, and Cybernetics* 21 (May/June): 473–509.

Brooks, R. A. 1990. A Robust Layered Control System for a Mobile Robot. In *Artificial Intelligence at MIT, Volume 2*, Winston, P. H., and Shellard, S. A., eds., 2–27. Cambridge, MA: The MIT Press.

Firby, J. R. 1989. Adaptive Execution in Complex Dynamic Worlds. Ph.D. diss., Dept. of Computer Science, Yale University.

Gat, E. 1991. Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots. Ph.D. diss., Virginia Polytechnic Institute and State University.

Hanks, S., Pollack, M., and Cohen, P. R. 1993. Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures. *AI Magazine* 14 (Winter): 17–42.

Knoblock, C. A. 1991. Search Reduction in Hierarchical Problem Solving. In Proceedings of the Ninth National Conference on Artificial Intelligence, AAAI-91, 686–691.

Koza, J. R. 1992. *Genetic Programming.* Cambridge, MA: The MIT Press.

Lin, L., Simmons, R. and Fedor, C. 1989. Experience with a Task Control Architecture for Mobile Robots, Technical Report CMU-RI-TR-89-29, The Robotics Institute, Carnegie Mellon University.

Maes, P., ed. 1990. *Designing Autonomous Agents.* Cambridge, MA: The MIT Press.

Peterson, J. L. 1981. Petri Net Theory and the Modeling of Systems. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Pollack, M. E., and Ringuette, M. 1990. Introducing the Tileworld: Experimentally Evaluating Agent Architectures. In Proceedings of the Eighth National Conference on Artificial Intelligence, AAAI-90, 183–189.

Pollock, J. L., 1989. *How to Build a Person: A Prolegomenon.* Cambridge, MA: The MIT Press.

Prieditis, A., and Janakiraman, B. 1993. Generating Effective Admissible Heuristics by Abstraction and Reconstruction. In Proceedings of the Eleventh National Conference on Artificial Intelligence, AAAI-93, 743–748.

Sacerdoti, E. D. 1974. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence* 5 (2): 115–135.

Schoppers, M. J. 1987. Universal Plans for Reactive Robots in Unpredictable Environments. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, IJCAI-87, 1039–1046.

Spector, L. 1992. Supervenience in Dynamic-World Planning. Ph.D. diss., Dept. of Computer Science, University of Maryland.

Spector, L. 1994. Genetic Programming and AI Planning Systems. In Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94. Forthcoming.

Spector, L., and Grafman, J. In Press. Planning, Neuropsychology, and Artificial Intelligence: Cross-Fertilization. In *Handbook of Neuropsychology*, Vol. 9, Boller, F., and Grafman, J., eds. Amsterdam: Elsevier Science Publishers B.V.

Spector, L., and Hendler, J. 1992. Planning and Reacting Across Supervenient Levels of Representation. *International Journal of Intelligent and Cooperative Information Systems* 1 (Numbers 3&4): 411–449.

Vere, S., and Bickmore, T. 1990. A Basic Agent. *Computational Intelligence* 6: 41-60.